Implementation of MQTT Protocol in Health Care Based on IoT Systems: A Study

Case Study

Roaa Wadullah Tareq
College of Engineering, Department of Computer Engineering,
University of Mosul, Mosul, Iraq
roaa.enp77@student.uomosul.edu.iq

Turkan Ahmed Khaleel
College of Engineering, Department of Computer Engineering,
University of Mosul, Mosul, Iraq
turkan@uomosul.edu.iq

Abstract *– Internet of things IoT systems have become one of the most promising technologies in all fields. Data transmission is one of the important aspects, and the tendency to messaging protocols is an important aspect of IoT systems. One of these most important protocols is MQTT. This protocol depends on the Publish/Subscribe model, and it is a lightweight protocol. Reliability, simplicity, quality of service levels, and being Resource-constrained make MQTT common in the IoT industry. This paper designed an IoT device that consists of the sensor MLX 90614 non-contact IR Temperature connected to a development board (Node MCU ESP8266). A person's temperature is one of the important vital signs. This system measures human temperature values and transmits the measured values to the Mosquitto broker by using the MQTT protocol in real-time. The technology used is Wi-Fi. The person or the doctor can read the patient's temperature remotely through a program (Flutter Android Client) representing the subscriber. Also, MQTT protocol control packets of the system were analyzed using Wireshark. The three levels of QoS were used in subscriber clients to compare the throughput. The results indicate that QoS2 is more reliable and offers more throughput but more delay. The results also show that the average round trip time (RTT) of the MQTT protocol is five milliseconds which means optimal performance for IoT applications.*

Keywords*: IoT, MQTT, Publish/Subscribe, Broker, QoS, MLX90614, NodeMCU ESP8266*

1. INTRODUCTION

The IoT needs a specific environment characterized by intelligence, as the Internet of Things device is any device that can be connected to the internet to collect data by sensors, process it, and send it over the internet to its specified endpoints. Fig. 1 shows the stages of dealing with data in IoT systems. IoT technology can connect any device to the internet in real-time at any time and from anywhere to control and analyze it [1, 2]. The IoT faces many issues that need to be addressed to properly implement it, including security and privacy and scalability, interoperability, and data management [3]. These are in addition to the lack of homogeneity of components with each other [4]. The application layer protocols of IoT responsible for transmitting data are important aspects. In the transmission of IoT, sensor data must use lightweight protocols and show bandwidth efficiency as these are fundamental features of the IoT. Moreover, it must show the efficiency of energy and capability of working with minimal hardware resources (like main memory and power supply) [5]. Internet access needs application protocols through UDP/IP or TCP/IP [6]. The protocols used in Internet of Things systems are as follows: MQTT (Message Queue Telemetry Transport), HTTP (HyperText Transport Protocol), AMQP (Advanced Message Queuing Protocol), COAP (Constrained Application Protocol), DDS (Data Distribution Service), XMPP (Extensible Messaging and Presence Protocol)[7]. One of the most important protocols used to transfer data in IoT systems is MQTT [8] which appeared in 1999 [9]. It was developed by Andy Stanford-Clark of IBM and Arlen Nipper of Arcom Control Systems [10]. MQTT protocol is considered OASIS standard [11] and M2M communication [12]. The publish/subscribe model used in the MQTT protocol makes it appropriate for M2M messaging [13]. One of the most important factors determining these M2M communications' performance is the messaging protocols designed for internet of things applications. The MQTT protocol uses default port 1883 and uses TCP/IP as transport [12]. MQTT uses Transport Layer Security/Secure Sockets Layer (SSL/TLS) as security [14]. It is considered one of the lightweight protocols used in devices with limited resources such as the IoT [12]. This protocol reduces the overhead costs and provides high communication efficiency for the internet of things as

it relies on "name-based routing" [6]. MQTT communication has two kinds of agents: the first is MQTT clients, and another is the broker of MQTT. The protocol-transmitted information is known as the application message. The clients of MQTT refer to the objects or devices connected to the internet that exchange messages or communication through MQTT. The clients of MQTT are known as subscribers and publishers. MQTT transfers the data from source (Publisher) to destination (Subscriber) through the broker. In MQTT, clients (or publisher and subscriber) do not require to cognize the identity of each other. Using an address named Topic, each message of data is published. A publisher can forward the application message while the subscriber can demand that application message to obtain its data. MQTT clients can be any device like a mobile, a sensor, Etc. The broker lets the various clients communicate with each other. It transmits and acknowledges the application messages between various clients connected to it. Fig. 2 shows the Publish/Subscribe model of the MQTT protocol.

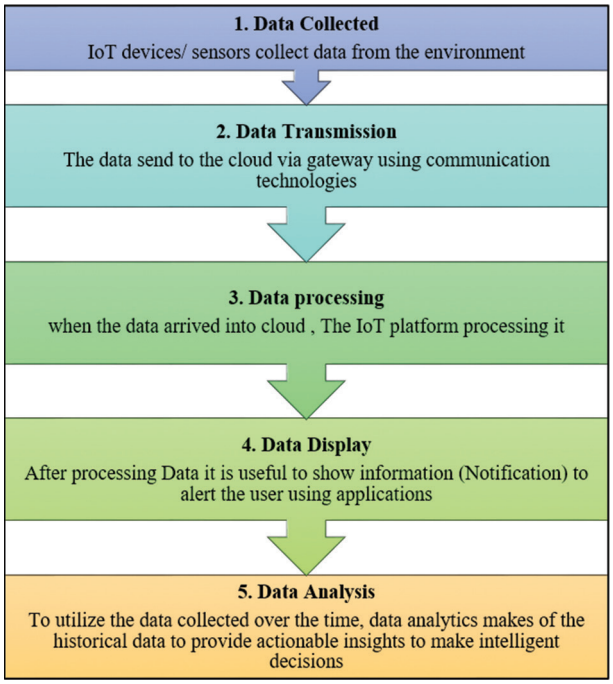

Fig. 1. The Stages of Dealing With Data In IoT Systems

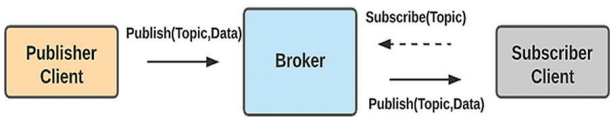

Fig. 2. Publish/Subscribe model of MQTT Protocol

Another prevalent protocol in web communication is HTTP. It utilizes a request/response model. This protocol is heavyweight, and it is a text protocol. This means that it sends massive size messages with high overhead. Instead of topics, HTTP utilizes the URI (Universal Resource Identifier). The server transmits data through URI, and the client receives it through the specific URI. Because HTTP is a text protocol, headers and payloads are determined by the programming technique or web server. The CoAP protocol, however, is an M2M and lightweight protocol. Like HTTP, the CoAP uses URI to send and receive data. It uses a request/response model. CoAP is a binary, like MQTT protocol which means that it needs 4 bytes fixed header and a small message of payloads up to a maximum size that varies depending on the programming technique or web server [15]. Table 1. shows the essential features and compares among MQTT, CoAP, and HTTP.

Table 1. Comparison among MQTT, CoAP, and HTTP

Protocol	MQTT	CoAP	HTTP
Designed by	IBM	IETF	IETF/ W3C
Messaging Model	Publish/ Subscribe	Request/Response	Request/ Response
Encoding Format	binary	binary	text
Transport Protocol	TCP	UDP	TCP
Hider Size	2 byte	4 byte	undefined
Security	TLS/SSL	DTLS, IPSec	TLS/SSL
Default Port	1883 (TCP Port) 8883 (TLS/SSL)	5683 (UDP Port) 5684 (DLTS)	80 (TCP Port) 443 (TLS/SSL)
QoS Reliability	At most once QoS 0, At least once QoS 1, Exactly once QoS 2	Confirmable Message Or Non-confirmable Message	Limited (via TCP)

In this paper, the body temperature data were collected using MLX90614 sensor as a publisher then sent to Mosquttio that uses as a broker by specific topic. Node MCU ESP8266 development board uses the technique Wi-Fi to transfer data. At the same time, a client was implemented using flutter. By the same topic, the client (doctor or person) subscribes to the broker to get the temperature value of the human in real-time. In addition, capture for the packets of MQTT was performed and analyzed it using Wireshark.

2. RELATED WORK

The MQTT protocol has been studied by many researchers. In this section, the most recent studies on this protocol are reviewed. The first direction relates to the implementation of the MQTT protocol in healthcare. The second direction is related to the study of the MQTT protocol, its analysis, and comparison with other protocols used in the fields of Internet of Things systems.

Sarierao and Prakasarao have been implemented a healthcare monitoring system using the MQTT protocol. The architecture of the Smart Healthcare System consists of microcontroller ESP32 using MQTT protocol. The sensors are used pulse rate, temperature sensor, spo2 sensor, and body movement sensor. This

healthcare system allows the doctor to remotely view a patient's vital signs on a web page and mobile app in real-time. All the doctor needs are internet access [16].

Another paper in the healthcare system has been implemented titled Smart Health Care System using IoT. It was used Node MCU ESP8266 with connecting sensors detecting (Heart Rate, Temperature, Fall Detection, and Step-Counter) and sends this data to a remote server through Wi-Fi using MQTT protocol. Heart rate: The device can be used to keep track of a person's heart rate. When the minimum threshold is reached, an individual can be automatically notified by doctors and family members. Using this device can save Millions of lives. Step counter: The device tracks the number of steps taken by the user so that he can exercise regularly. It also helps in calculating the number of calories in addition to knowing the body temperature and detecting the patient's fall [17].

The patient's vital signs monitoring system (heart rate and blood oxygen level) was designed using MAX30100 SPO2 sensor and connected to the ESP32 microcontroller using MQTT protocol to send these signs to be monitored via phone or computer. As a consequence of the publication Monitoring, vital signs of human hear based on IoT [18], the MQTT protocol is appropriate in health care applications for real-time monitoring of vital signs. Finally, on this part, Priyamvadaa suggested a utility to develop real-time body temperature readings using the MQTT protocol in healthcare. According to the results obtained, the researcher concluded that the MQTT protocol provides high data portability also energy efficiency, security, scalability, and reliability [19].

In the other direction, the protocol is covered and surveyed. It was explained the most importance of MQTT protocol in IoT, MQTT architecture, and existing problems in MQTT such as message expiry, security, ordering, and priority. Furthermore, the number of MQTT brokers was mentioned, with each having its own set of restrictions and none of them implementing data priority or future advancements for this protocol [20].

In another article, the architecture of the MQTT protocol is explored, as well as QoS levels, message format, and MQTT's scope. MQTT is essentially a binary data transfer protocol that supports a wide range of communication technologies. Its goal is to create a communication system that uses as little bandwidth as possible. MQTT uses the TCP protocol for transport and communicates over IP [12].

Another research detailed the MQTT protocol's architecture, QoS levels, message format, and MQTT scope. MQTT is primarily a binary data conduit that allows for a variety of communication methods. It's intended to provide a communications system with the least amount of bandwidth requirements. The MQTT protocol has been defined and compared to other IoT message protocols, such as CoAP, for transport. In addition, there have been tools available to aid in the execution of practical experiments and simulations. Experiments were performed to observe the communication delay between both MQTT and CoAP according to the results obtained. The QoS0 level of MQTT showed a lower delay than CoAP. Finally, the challenges and open issues in this field are examined [21].

In addition, it was introduced the various M2M communications protocols such as MQTT, CoAP, and AMQP that being used over the previous 20 years. The most widely used M2M/IoT protocol, MQTT, has been improving. The protocol was examined some of the most relevant research papers in the current literature reviews to highlight the key characteristics, benefits, and limits of this protocol and MQTT broker implementations concerning comparison to alternative IoT protocols. The results were presented findings of the current usage of MQTT and the areas of its application using different comparison tables and graphs. It arrived that an in-depth comparison of the characteristics of many brokers and clients libraries of MQTT in several taxonomy categories were utilized to enable the researchers and the users to choose implementation of MQTT based on the needs and appropriateness [22].

Finally, the MQTT protocol-based applications in IoT systems are addressed, and methods for controlling access and organizing data exchange across MQTT protocol contexts are proposed. According to the user's preferences and authorization policies, it was offered the framework of access control to manage data sharing across environments of MQTT [23].

3. USING MQTT PROTOCOL TO SENDING DATA TO THE CLOUD

MQTT stands for Message Queuing Telemetry Transport. It is considered a lightweight protocol, and it uses a server called a broker. The client(Publisher) sends the data as a message to the broker using a specific subject called a topic, indicating the data category. More than one client can receive the message from the broker via the same specified topic. Figure 3 shows the network of intercommunication using the MQTT protocol.

1. Message

A message is the basic unit of MQTT protocol communication; it contains basic information called the "topic" that constitutes the data exchanged between devices through the broker.

2. Topic

The topic is the basic information for determining the message to be sent and received. The structure of the topic is determined in the form of a hierarchy separated by a slash (/). The topic must always be specified when sending the message or when subscribing to receive it. Publisher client (sensor) publishes data with its topic, subscriber clients who want to receive it can assign the needed topic and receive the data [24].

3. Publish/Subscribe Paradigm

The publish/subscribe paradigm (or pub/sub) is an alternative to conventional client-server architecture. The publisher client and subscriber client never communicate directly with each other. Communication takes place via a third component called the broker. Several dimensions separate the publisher of the message from the subscriber, and they are as follows:

- Space decoupling: The publisher and the subscriber do not need to cognize each other (for instance: no interchange of IP address).
- Time decoupling: The publisher and the subscriber are not required to work at the same time.
- Synchronization decoupling: during receiving or publishing, operations do not need to be interrupted.

4. Broker

The broker represents the communication node between the publisher and the subscriber. It filters and organizes all arriving messages then distributes them accurately to subscribers using a specific topic interested in it [25]. There are many open-source brokers, the most important of which are Mosquito, Hive MQ, and Mosca. They differ in specifications and functions that must be taken into consideration in advance.

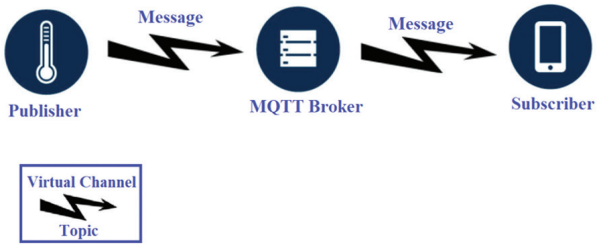

Fig. 3. The network for intercommunication using MQTT Protocol

4. QUALITY OF SERVICE QOS LEVELS IN MQTT PROTOCOL

The MQTT is an asynchronous protocol whose paradigm is based on Publish/Subscribe model. It flows through TCP/IP, connecting large numbers of control devices and remote sensors [26]. Devices are allowed to exchange data using a message broker. The broker of MQTT sends the message data to the subscribed clients and forwards, stores, prioritizes, filters, and publishes requests from the publisher client to the subscriber client. The chosen quality of service QoS level relies on the system. For example, suppose the system requires a constant data transfer. In that case, MQTT adapts QoS2 to deliver data even if there is a delay of time [27]. MQTT offers three levels of QoS [28]:

1. (QoS 0) --- At-most once. It is the simplest, fastest, and most unreliable level of QoS where the message data is transmitted to the subscriber client at-most-once (one time only). It is not saved or stored. Also, the publisher client does not receive any confirmation or information about delivering the message (no need to be acknowledged). If the subscriber client does not have a network connection or if the publisher is down or unable to receive the message, the packet is lost. The probability of repeated messages does not exist. The expression "fire and forget" describes (At-most once) QoS 0 level.

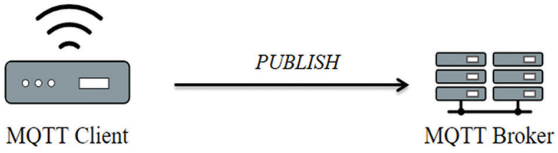

Fig. 4. Quality of Service (QoS0)

2. (QoS 1) --- At least once. The message of data is transmitted to the subscriber client At least once. An acknowledgment packet is used at this level of QoS. The publisher must use DUP flag for the duplicated data messages. The Publishing with QoS of level 1 needs two messages. The publisher publishes a data message to the subscriber if the publisher does not receive Acknowledgment from the subscriber. QoS keeps publishing the message until it receives the acknowledgment packet (PUBACK).

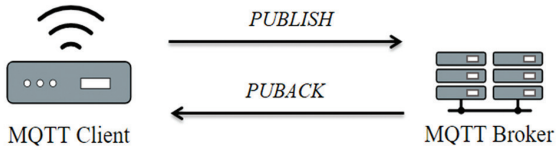

Fig. 5. Quality of Service (QoS1)

3. (QoS 2) --- Exactly once. QoS 2 is used to guarantee the data message delivery and to ensure safety. This level is the slowest as it needs four messages. While posting the message, two rounds of transition are utilized. The message must be stored to be processed by the publisher and the subscriber. In the first round, the publishing customer transmits a message of data to the subscriber client then waits for acknowledgment from the subscriber the message of data has been saved. If an acknowledgment is not received, the publisher sends the message until it receives the acknowledgment that message has been received. After that, the second round begins; the publisher sends PUBREL to inform the subscriber that the message can be processed and waits for an acknowledgment PUBREL of receipt from the subscriber and then deletes the message.

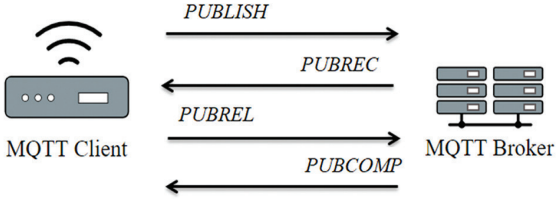

Fig. 6. Quality of Service (QoS2)

5. MQTT PROTOCOL FEATURES

Several points distinguish the MQTT protocol and make it suitable for IoT applications [29], including:

- It supports three levels of QoS to ensure message reliability.
- It uses bandwidth efficiently by packet agnostic, and it has a small overhead.
- The data is binary.
- The publish/subscribe method has capabilities like M2M communication. This technique also allows bi-directional communication.
- The communications are asynchronous.
- Anytime, it can publish/subscribe messages.
- It is suitable for limited-resource devices such as sensors for IoT systems.

6. SYSTEM DESIGN AND IMPLEMENTATION DETAILS

Hardware System Requirements:

1. Node MCU ESP 8266 development board
2. MLX 90614 Non-contact Temperature Sensor
3. Male to Male wires and USB Cable
4. Laptop

Software System Requirements:

1. Android Emulator in Flutter to execute Client
2. Arduino IDE
3. Mosquitto Broker
4. Wireshark to Capture Packets of MQTT Protocol

Table 4. MQTT Parameters

Parameter Used	
Broker	Mosquitto
Topic	Capture/Temperature
Port	1883
Subscriber	MQTT Client in Flutter

The design of the system is shown in Figure 7. The system consists of a sensor to measure the value of temperature in humans (MLX90614). This sensor which uses infrared (IR) and is non-contact, is connected to a Node MCU ESP8266 low-cost development board. Table 6 shows some specifications of Node MCU ESP8266 that distinguish it from other systems and make ESP8266 suitable for IoT projects due to its small size and its use of a full-stack of TCP/IP. ESP8266 accesses the internet through the router by using Wi-Fi technology and utilizing MQTT protocol. This system is designed to measure the human temperature as the measured data (temperature) is sent to the server (Mosquitto Broker [30]) by a topic that has been determined (Capture/Temperature). The person or doctor who monitors the patient's condition can see the measurement results in real-time from anywhere. The measured data is also stored in the server so that the results are saved as a database to be displayed again to check the patient's temperature at any time. In Table 5, we mention the properties of the sensor used MLX90614 while Table 6, specifications of Node MCU ESP8266.

Table 5. The Characteristics of MLX90614 Sensor

Operating Voltage	3v to 5v
Supply Current	1.5 mA
Range of Object Temperature	-70° C to 382.2°C
Range of Ambient Temperature	-40° C to 125°C
Accuracy	0.02°C
The distance among object and sensor	2cm to 5cm (approx.)

Table 6. The Specifications of Node MCU ESP8266

Microcontroller	32-bit Tensilica RISC CPU Xtensa LX106
Operating voltage	3.3 V
Wi-Fi	802.11 b/g/n
Clock speed	80 MHZ-160MHZ
Analog Input	1 pin
Digital I/O	16 pins

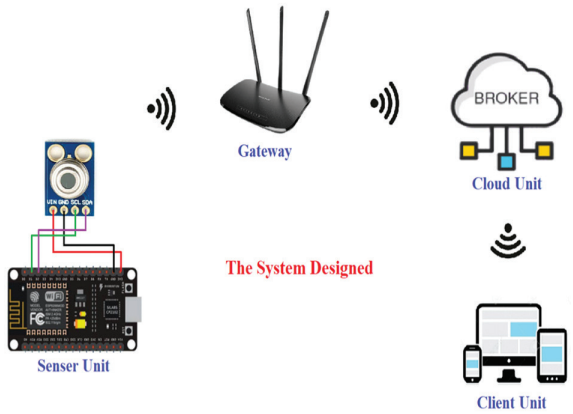

Fig. 7. The design of the system using the MQTT Protocol

7. CAPTURE MQTT CONTROL PACKETS USING WIRESHARK

The communication of MQTT protocol is the procedure of exchanging a series of control packets of MQTT [31]. This section explains the format of control packets for MQTT by capturing them using Wireshark. The control packet of MQTT consists of three parts, in the following order:

1. Fixed header: exists in all the control packets
2. Variable header: exists in some kinds of control packets
3. Payload: Also found in some control packets of MQTT

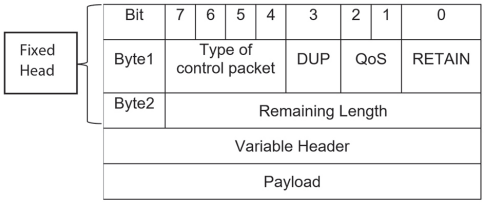

Fig. 8. Format of MQTT control packet

There are 14 kinds of MQTT control packets, and they are as follow:

1. CONNECT The first packet sent by the publisher client to the server after establishing the network through the client's communication with the server.

2. CONNACK: It is the acknowledgment of receipt of the communication packet sent by the server to confirm receipt of the connection packet, as it is the first packet by the server. If the time to receive this packet is exceeded, the customer will close the network connection.

3. PUBLISH: It is the packet that the publisher client and server can send to transfer data (application message).

4. PUBACK: It is an acknowledgment packet of PUBLISH packet with QoS level.

5. PUBREC: The response packet to the PUBLISH packet with the second level of quality of service (QoS 2).

6. PUBREL: It is a response packet to the PUBREC packet.

7. PUBCOMP: It is a response packet to the PUBREL packet.

8. SUBSCRIBE: The subscriber sends a subscribe packet using a specific subject assigned to obtain the application message.

9. SUBACK: It is an acknowledgment and processing packet for the SUBSCRIBE packet.

10. UNSUBSCRIBE: This packet sent by a subscriber to unsubscribe in topic

11. UNSUBACK: It is an acknowledgment packet for accessing an unsubscribe packet.

12. PINGREQ: Sending this packet determines if the client is alive.

13. PINGRESP: sending this packet determines if the server is alive. The server sends it to respond to the PINGREQ packet.

14. DISCONNECT: It is sent as the last packet to the server as it indicates a disconnection for the client.

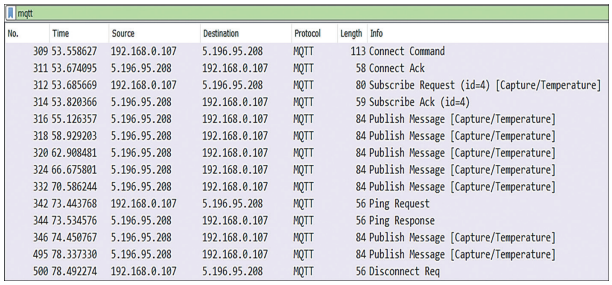

Fig. 9. Capture MQTT packets using Wireshark

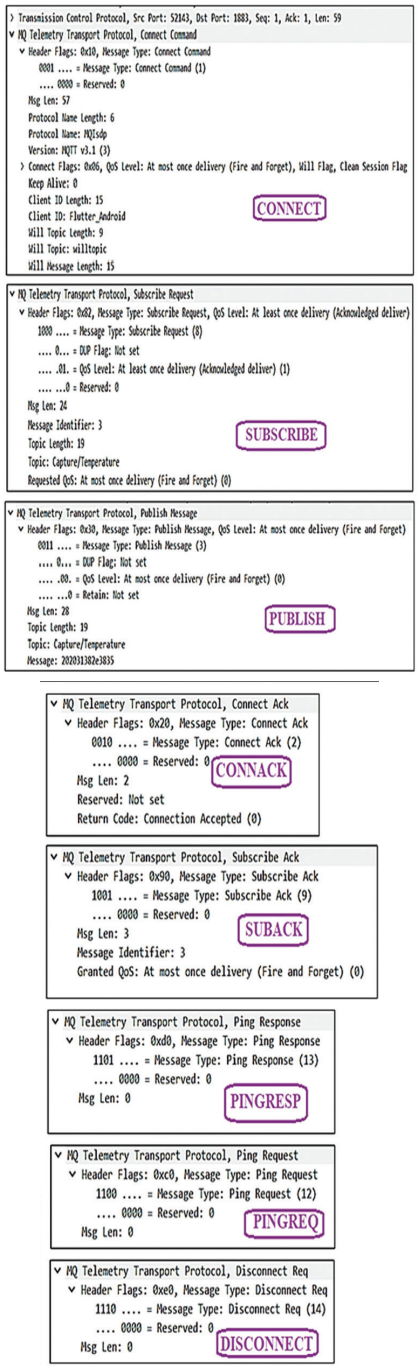

Fig. 10. MQTT control packets type using Wireshark

8. RESULTS AND DISCUSSION

The system was connected using MQTT protocol and was used to measure the human temperature. As the sensor senses the temperature, the readings, which represent the data, are published to the broker using a specific topic (Capture/Temperature) and picked up by a Mosquitto broker, which was chosen because it is open-source. The subscriber client has been implemented using flutter (software development) that the client can operate on the Android or Apple system. Fig. 11 shows the practical

connection of the Internet of Things (IoT) system, Fig. 12 shows serial monitors displaying the readings of temperature using QoS1 to publish data, while Fig.

13 shows the temperature readings in real-time using the MQTT BOX tool. The real-time human temperature reading was obtained using an Android Emulator that subscribes to the specific topic, as shown in Fig. 14.

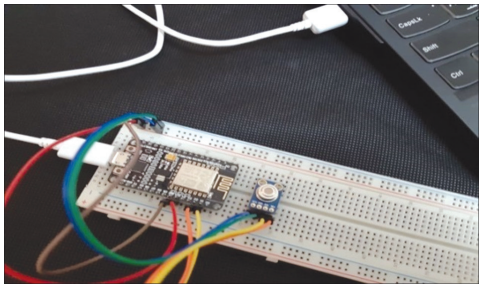

Fig. 11. IoT system (practical connection) NodeMCU ESP8266 with MLX90614 sensor

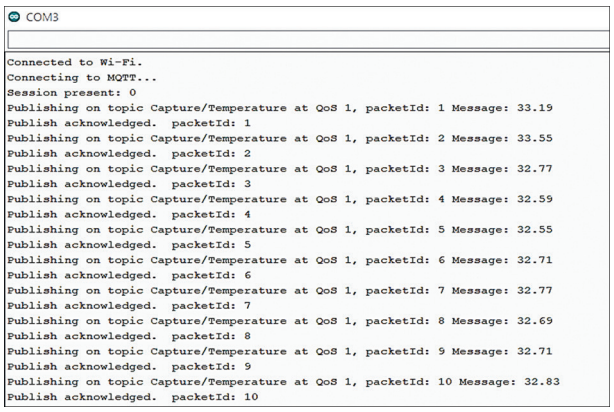

Fig. 12. The results of temperature on the serial monitor

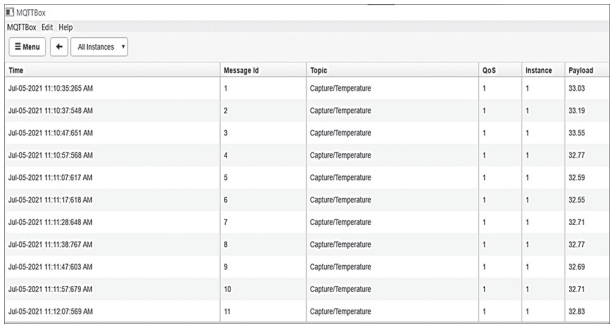

Fig. 13. The temperature in real-time using MQTTBox

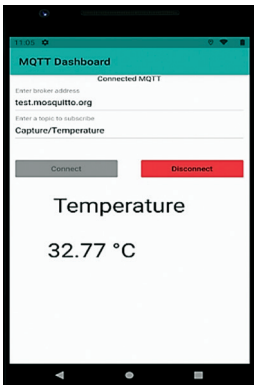

Fig. 14. Flutter Android Client

We noticed a quick response when subscribing to the specific topic using Wireshark, as shown in Fig. 15, which means that MQTT protocol has low overhead and provides high communication efficiency for the IoT system because it relies on "name-based routing". The feature of TCP Steam Graph, which exists in Wireshark, provides a whole record for RTT of each packet. RTT is the time needed for sending a packet and receiving acknowledgment packets. In a test of the duration of time in which RTT is ranged from 0 msec to 10 msec, the packets were captured between 1070-1110 sec, as shown in Fig. 16 and Fig. 17. The communication network speed uses the I/O graph of Wireshark, and the data were obtained at the three levels of QoS. Fig. 18 shows the detailed waveform of the bytes per sec for 100 sec. The throughput of this network means that the number of MQTT packets was successfully received. The results show that the highest level of QoS2 is more reliable and has higher throughput due to a 4-way handshake mechanism. MQTT protocol is distinguished from other protocols like HTTP and CoAP. It creates better packets and requires less time for transmission, even though CoAP is a UDP protocol [6] [29].

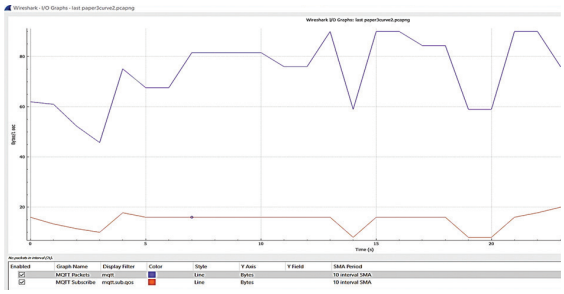

Fig. 15. The response when subscribing in the client

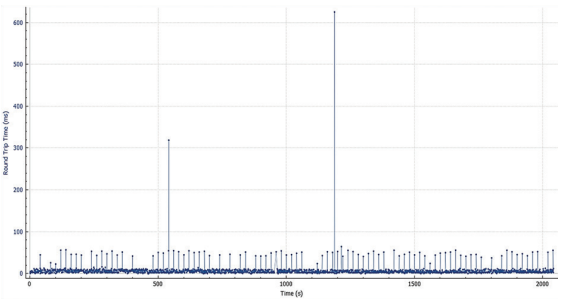

Fig. 16. RTT of the MQTT packets captured using Wireshark for 2000 sec.

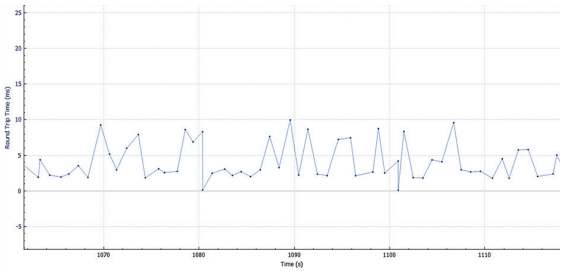

Fig. 17. RTT of the MQTT packets captured during 1070 sec - 1110 sec

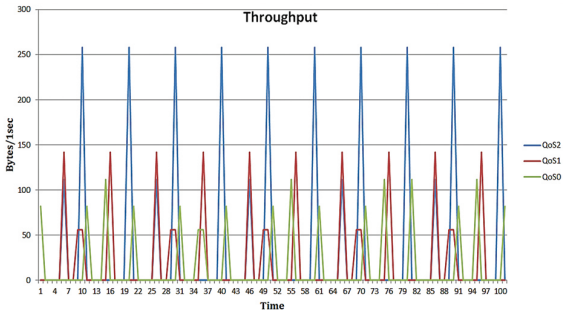

Fig. 18. Throughput for QoS levels

9. CONCLUSION AND FUTURE WORK

In this article, we suggested a system for monitoring the temperature of humans using the MQTT protocol in real-time as an IoT system can read the temperature from anywhere. The application that was implemented uses flutter and can be downloaded on Android and Apple systems. It was concluded that the protocol is a lightweight protocol that provides a high network connection and provides three levels of QoS to ensure reliability. The QoS2 is more reliable, and throughput but requires more time. The average RTT for the packets is approximately 5 msec, which could be considered the optimal response time for IoT applications. Future work can be aim is to expand and develop the system by introducing other sensors to measure the oxygen level and heart rate of patients using the MQTT protocol. Also, this system can be implemented with other Internet protocols such as CoAP and HTTP, and a comparison can be made between them in terms of performance.

10. ACKNOWLEDGEMENTS

The authors are grateful for the facilities provided by the University of Mosul's College of Engineering's Department of Computer Engineering, which contributed to improving the quality of this paper.

11. REFERENCES

[1] M. Al-Khafajiy et al., "Remote health monitoring of elderly through wearable sensors", Multimedia Tools and Applications, Vol. 78, No. 17, 2019, pp. 24681-24706.

[2] A. Mishra, A. Kumari, P. Sajit, and P. Pandey, "Remote web based ECG Monitoring using MQTT Protocol for IoT in Healthcare", International Journal of Advance Engineering and Research Development, Vol. 5, No. 04, 2018, pp.1096-1101.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications", IEEE communications surveys & tutorials, Vol. 17, No. 4, 2015, pp. 2347-2376.

[4] C. P. Kruger, A. M. Abu-Mahfouz, G. P. Hancke, "Rapid prototyping of a wireless sensor network gateway for the internet of things using off-the-shelf components", Proceedings of the IEEE International Conference on Industrial Technology, Seville, Spain, 17-19 March 2015, pp. 1926-1931.

[5] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, C. K.-Y. Tan, "Performance evaluation of MQTT and CoAP via a common middleware", Proceedings of the IEEE ninth international conference on intelligent sensors, sensor networks and information processing, Singapore, 21-24 April 2014, pp. 1-6.

[6] T. Yokotani, Y. Sasaki, "Comparison with HTTP and MQTT on required network resources for IoT", Proceedings of the International Conference on Control, Electronics, Renewable Energy and Communications, Bandung, Indonesia, 13-15 September 2016, pp. 1-6.

[7] E. Al-Masri et al., "Investigating messaging protocols for the Internet of Things (IoT)", IEEE Access, Vol. 8, 2020, pp. 94880-94911.

[8] R. Atmoko, R. Riantini, M. Hasin, "IoT real-time data acquisition using MQTT protocol", Journal of Physics: Conference Series, Vol. 853, No. 1, 2017, pp.1-6.

[9] K. Sugumar, "MQTT-A LIGHTWEIGHT COMMUNICATION PROTOCOL RELATIVE STUDY", Author Preprints, 2020, pp.1-5.

[10] B. Mishra, "TMCAS: An MQTT based collision avoidance system for railway networks", Proceeding of the 18th International Conference on Computational Science and Applications, Melbourne, VIC, Australia, 2-5 July 2018, pp. 1-6.

[11] O. Standard, "MQTT version 3.1. 1", Available at http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html, Vol. 1, 2014.

[12] M. B. Yassein, M. Q. Shatnawi, S. Aljwarneh, R. Al-Hatmi, "Internet of Things: Survey and open No.s of MQTT protocol", Proceedings of the International Conference on Engineering & MIS, 2017, pp. 1-6.

[13] I. Heđi, I. Špeh, A. Šarabok, "IoT network protocols comparison for the purpose of IoT constrained networks", Proceedings of the 40th International Convention on Information and Communication Technology, Electronics and Microelectronics, Opatija, Croatia, 22-26 May 2017, pp. 501-505.

[14] J. J. Anthraper, J. Kotak, “Security, Privacy and Forensic Concern of MQTT Protocol”, Proceedings of International Conference on Sustainable Computing in Science, Technology, and Management, Amity University Rajasthan, Jaipur-India, 2019, pp. 876-883.

[15] N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP", Proceedings of the IEEE international systems engineering symposium, Vienna, Austria, 11-13 October 2017, pp. 1-7.

[16] B. S. Sarierao, A. Prakasarao, "Smart healthcare monitoring system using MQTT protocol", Proceedings of the 3rd International Conference for Convergence in Technology, Pune, India, 6-8 April 2018, pp. 1-5.

[17] V. M. Rao, C. V. Shankar, K. G. Reddy, “Smart Health Care System using IOT”, International Journal of Innovative Technology and Exploring Engineering, Vol. 8, No. 6,2019, pp. 772-775.

[18] K. T. Kadhim, A. M. Alsahlany, S. M. Wadi, H. T. Kadhum, “Monitoring vital signs of human hear based on IoT”, Al-Furat Journal of Innovations in Electronics and Computer Engineering, Vol. 01, No. 2, 2020, pp. 9-13.

[19] R. Priyamvadaa, “Temperature and Saturation level monitoring system using MQTT for COVID-19”, Proceedings of the International Conference on Recent Trends on Electronics, Information, Communication & Technology, 2020, pp. 17-20.

[20] D. Soni, A. Makwana, “A Survey On MQTT: A Protocol Of Internet Of Things (IoT)”, Proceedings of the International Conference On Telecommunication, Power Analysis And Computing Techniques, 2017, pp. 1-5.

[21] S. Quincozes, T. Emilio, J. Kazienko, “MQTT Protocol: Fundamentals, Tools and Future Directions”, IEEE Latin America Transactions, Vol. 17, No. 9, 2019, pp. 1439-1448.

[22] B. Mishra, A. Kertesz, “The Use of MQTT in M2M and IoT Systems: A Survey”, IEEE Access, Vol. 8, 2020, pp. 201071-201086.

[23] P. Colombo, E. Ferrari, E. D. Tümer, “Regulating data sharing across MQTT environments”, Journal of Network and Computer Applications, Vol. 174, 2021,p. 102907.

[24] D. Soni, A. Makwana, "A survey on MQTT: a protocol of internet of things (IoT)", Proceedings of the International Conference On Telecommunication, Power Analysis And Computing Techniques, 2017, pp.1-5.

[25] L. Durkop, B. Czybik, J. Jasperneite, “Performance evaluation of M2M protocols over cellular networks in a lab environment”, Proceedings of the 18th International Conference on Intelligence in Next Generation Networks, Paris, France, 17-19 February 2015, pp. 70-75.

[26] S. Jaloudi, “Communication protocols of an industrial internet of things environment: A comparative study”, Future Internet, Vol. 11, No. 3, 2019, p. 66.

[27] J. E. Luzuriaga, J. C. Cano, C. Calafate, P. Manzoni, M. Perez, P. Boronat, “Handling mobility in IoT applications using the MQTT protocol”, Proceedings of the Internet Technologies and Applications, Wrexham, UK, 8-11 September 2015, pp. 245-250.

[28] J. Toldinas, B. Lozinskis, E. Baranauskas, A. Dobrovolskis, “MQTT Quality of Service versus Energy Consumption”, Proceedings of the 23rd International Conference Electronics, Palanga, Lithuania, 17-19 June 2019, pp. 1-4.

[29] B. H. Çorak, F. Y. Okay, M. Güzel, Ş. Murt, S. Ozdemir, "Comparative analysis of IoT communication protocols", Proceedings of the International symposium on networks, computers and communications, Rome, Italy, 19-21 June 2018, pp. 1-6.

[30] “Eclipse Mosquitto”, https://mosquitto.org/, Accessed on 7/10/2021.

[31] M. Houimli, L. Kahloul, S. Benaoun, “Formal specification, verification, and evaluation of the MQTT protocol in the Internet of Things”, Proceedings of the International Conference on Mathematics and Information Technology, Adrar, Algeria, 4-5 December 2017, pp. 214-221.