Evolution in Electrical and Electronic Engineering Vol. 3 No. 2 (2022) 611-616
© Universiti Tun Hussein Onn Malaysia Publisher's Office

EEEE

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/eeee
e-ISSN : 2756-8458

Smart Room System with Remote Access Using MQTT

Nick Cassidy Merunei[1], Chew Chang Choon[1]*

[1]Faculty of Electrical and Electronic Engineering,
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author Designation

DOI: https://doi.org/10.30880/eeee.2022.03.01.071
Received 20 January 2022; Accepted 31 May 2022; Available online 30 June 2022

Abstract: The smart building has become increasingly popular, especially among university and technology-based companies, as it offers greater efficiency in terms of energy management, and harnesses the power of facilities information that leads to better decision-making and more streamlined operations. From a room with a smart whiteboard to a room that automatically turns on its light for any visitor, it is clear that everything can be achieved with technology. This project aims to develop a smart room that is based on a webserver. Instead of using an application to monitor the room. This can be done by using the DHT11 temperature-humidity sensor, as this sensor can measure a range of temperature from 0^{o}C to 50^{o}C. In contrast, the humidity can be measured from 20% to 90% humidity level, with the accuracy of ±1^{o}C and ±1%. Data from the sensor is then sent to the ESP8266, which sends data to the webserver and finally to the user through the user interface. This process is programmed by using Arduino IDE. Next, through the user interface, the user can also switch on or off the light in the room remotely as long as the user is connected to the internet. The user can request data from the ESP8266, and the ESP8266 will turn the light on or off depending on the user. Generally, the user will be able to monitor the temperature and humidity in the room and control the light no matter where the user is, given both ESP8266 and the user are exposed to an internet connection.

Keywords: MQTT, ESP8266, DHT11 Temperature-Humidity Sensor

1. Introduction

A building-monitoring system is a system that is run through websites rather than apps, overcoming the problem faced by mobiles because smartphones use multiple operating systems. The developer must develop apps for each platform to be accessible to all administrators. The method comprises a Wi-Fi microchip and is website-based so that the user will access the website and monitor the room temperature anywhere as long as there is an internet connection.

Home automation or smart home (also known as domotics or domotica) is the residential extension of building automation and involves the control and automation of lighting, heating (such as

Corresponding author: chewcc@uthm.edu.my smart thermostats), ventilation, air conditioning (HVAC), and security, as well as home appliances such as washer/dryers, ovens or refrigerators/freezers that use Wi-Fi for remote monitoring. Modern systems generally consist of switches and sensors connected to a central hub sometimes called a "gateway" from which the system is controlled with a user interface that is interacted either with a wall-mounted terminal, mobile phone software, tablet computer or a web interface, often but not always via internet cloud services [1]. Smart room is an advanced extension of a commercial room to make life very comport for its residents. In the paper, the design of the smart room was discussed along with a block diagram and all hardware and software were discussed appropriately. The system was also implemented and tested and it comprises room accessories automation, a security system and a real-time weather status display [2]. Energy wastage is a common problem nowadays. The energy wasted because of human negligence is massive. In commercial buildings alone, where annual electricity is around $190 billion, around 30% of this energy goes to waste [3]. This smart system also allows the user to have remote access to the electronics inside the building, enabling the user to control and monitor the electronics anywhere as long as there is an internet connection.

From M. M. Froufe et al. [4] , the main goals of their research is to study the main driver that enhances a building's intelligence, as well as exploring the main system that is currently present in those building and search for the relationship between the driver and the systems [4]. While for proposed by Monika Kashyap et al. [5], a project is conducted to control LED by implementing MQTT, specifically using HiveMQ as the broker and connecting it to Arduino. ESP8266's GPIO is controlled by the MQTT client by requesting the sensor information and distributing the order[5]. Diego Salas Ugalde[6] researched the security of MQTT in terms of providing secure communication between a sensor and cloud, comparing it to a more traditional security protocol [6].

2. Methodology

This section is divided into three parts that contain design, hardware development and server development. The main part of this project is to implement MQTT in monitoring the intended electronic and able to control them online.

2.1 Hardware development

In the initial stage of this project, it contains some key hardware component such as Node MCU ESP8266, temperature sensor, relay, Raspberry Pi and smartphone. The hardware design is separated into two parts which is the controlling part and the monitoring part. The smartphone and Node MCU ESP8266 belong to the controlling part, and the Raspberry Pi is the core of the monitoring part. Both of these parts work together to form the initial stage of the smart room monitoring system.

2.2 Software development

The project utilises the versatility of ESP8266 Node MCU as the microchip to accomplish the project. To program the ESP8266, Arduino IDE is required, and since it is a student-friendly software that can be installed on Windows, Linux and Mac OS, it can be used by anyone. The DHT11 temperature-humidity sensor is connected to the Node MCU, collecting data regarding temperature and humidity and giving almost accurate output to the user. A lightbulb that is connected to a relay is also connected to the Node MCU, giving the user freedom to turn it on or off.

The MQTT server is created by using Raspberry Pi as the platform. This server, particularly Mosquitto server, is used to transmit data between the user and the Node MCU. Figure 1 shows the hardware circuit design, while Figure 2 shows the Mosquitto server with Node-RED, where the server is activated on the Raspberry Pi by the user.

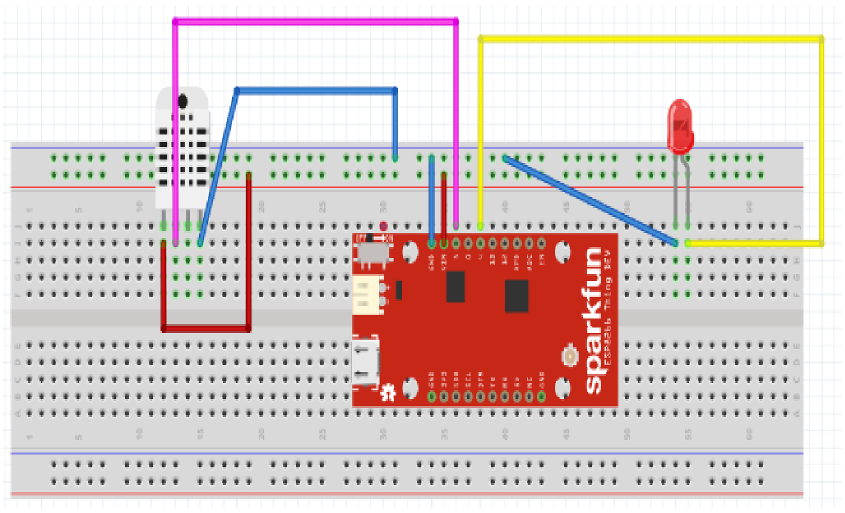
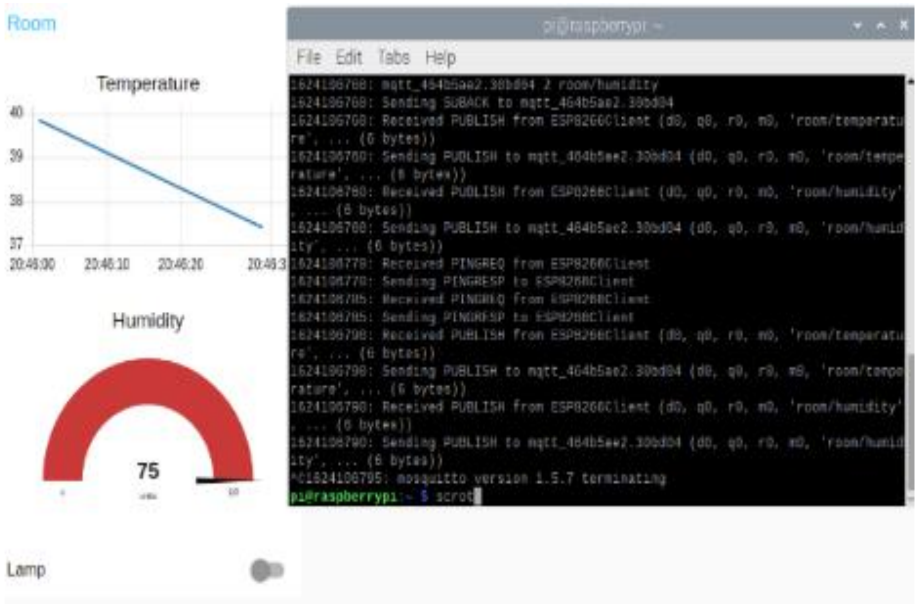


Figure 1: Hardware circuit design

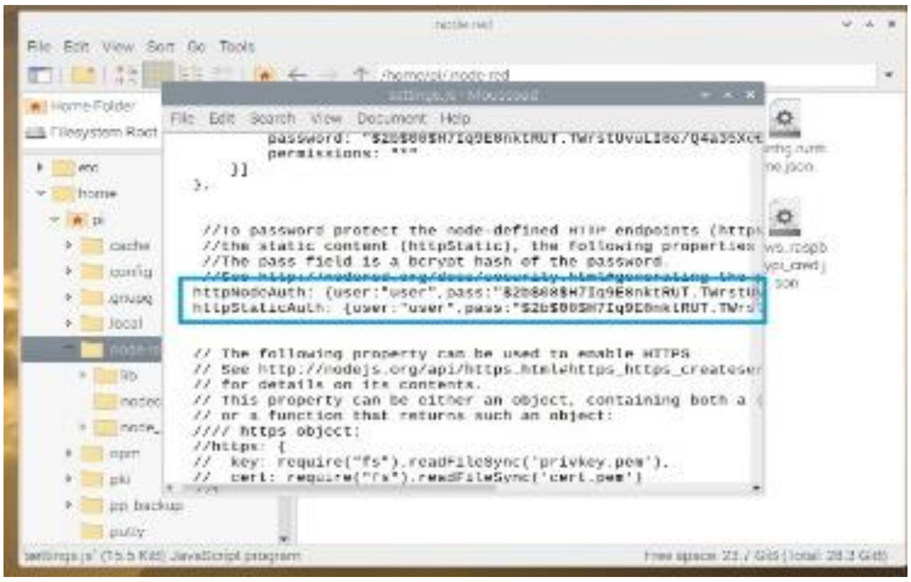


```
1624106768: mqtt_464b5ae2.30bd04 2 room/humidity
1624106768: Sending SUBACK to mqtt_464b5ae2.30bd04
1624106768: Received PUBLISH from ESP8266Client (d0, q0, r0, m0, 'room/temperatu
re', ... (6 bytes))
1624106768: Sending PUBLISH to mqtt_464b5ae2.30bd04 (d0, q0, r0, m0, 'room/tempe
rature', ... (6 bytes))
1624106768: Received PUBLISH from ESP8266Client (d0, q0, r0, m0, 'room/humidity'
, ... (6 bytes))
1624106768: Sending PUBLISH to mqtt_464b5ae2.30bd04 (d0, q0, r0, m0, 'room/humid
ity', ... (6 bytes))
1624106778: Received PINGREQ from ESP8266Client
1624106778: Sending PINGRESP to ESP8266Client
1624106785: Received PINGREQ from ESP8266Client
1624106785: Sending PINGRESP to ESP8266Client
1624106798: Received PUBLISH from ESP8266Client (d0, q0, r0, m0, 'room/temperatu
re', ... (6 bytes))
1624106798: Sending PUBLISH to mqtt_464b5ae2.30bd04 (d0, q0, r0, m0, 'room/tempe
rature', ... (6 bytes))
1624106798: Received PUBLISH from ESP8266Client (d0, q0, r0, m0, 'room/humidity'
, ... (6 bytes))
1624106798: Sending PUBLISH to mqtt_464b5ae2.30bd04 (d0, q0, r0, m0, 'room/humid
ity', ... (6 bytes))
^C1624106795: mosquitto version 1.5.7 terminating
pi@raspberrypi:~ $ scrot
```

Figure 2: Mosquitto server with Node-RED

2.3 SSL and username authentication

Node-Red GUI is easily accessible to others as long as the user either connected to the same Wi Fi as the server or has the QR code that are generated by the dashboard. This can be prevented by securing the Node-Red with SSL and username authentication. Firstly, to secure the Node-Red, one need to create a self-signed certificate. Figure 3 shows the password and username setting.



```
        password: "$2b$08$H7Iq9E8nktRUT.TWrstUvuLI8e/Q4a35Xct
        permissions: "*"
    }]
},


 //To password protect the node-defined HTTP endpoints (https
 //the static content (httpStatic), the following properties
 //The pass field is a bcrypt hash of the password.
 //See http://nodered.org/docs/security.html#generating-the-
httpNodeAuth: {user:"user",pass:"$2b$08$H7Iq9E8nktRUT.TWrstU
httpStaticAuth: {user:"user",pass:"$2b$08$H7Iq9E8nktRUT.TWrs

// The following property can be used to enable HTTPS
// See http://nodejs.org/api/https.html#https_https_createser
// for details on its contents.
// This property can be either an object, containing both a
// or a function that returns such an object:
//// https object:
//https: {
//  key: require("fs").readFileSync('privkey.pem').
//  cert: require("fs").readFileSync('cert.pem')
```

Figure 3: Password and username setting

3. Results and Discussion

The hardware and the software must be able to be used and run successfully simultaneously. The temperature sensor must work properly and send the output data to the server through the ESP8266, to the user interface. The light must be able to be controlled by the user through the interface by switching it on or off, as the server will request data from the ESP8266. Everything is dependent on the Mosquitto broker, so just like other servers, it needs to be online in order to be accessible. This means, if the Raspberry Pi is turned off, so does the server. This is totally user-dependent, so it can only work as a demonstration. Once all the hardware is completed, the goal is to control the light, monitor the temperature and humidity. In order to prove the working software as well as hardware, the Raspberry Pi plays a crucial role in this project.

3.1 Results

Figure 4 shows the working graphic user interface. The GUI is comprised of humidity gauge meter, temperature chart, and a switch to the light. All of the data received by the ESP8266 are displayed here, and the user can interact with the switch to turn the light on or off.

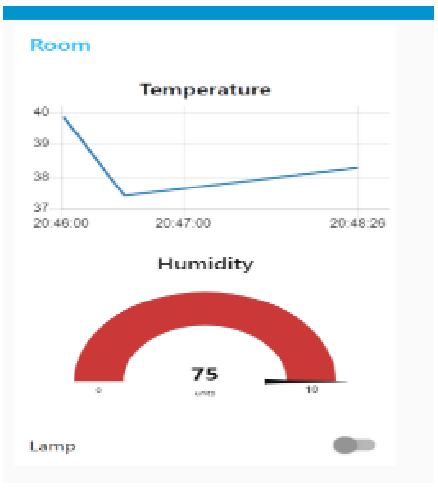
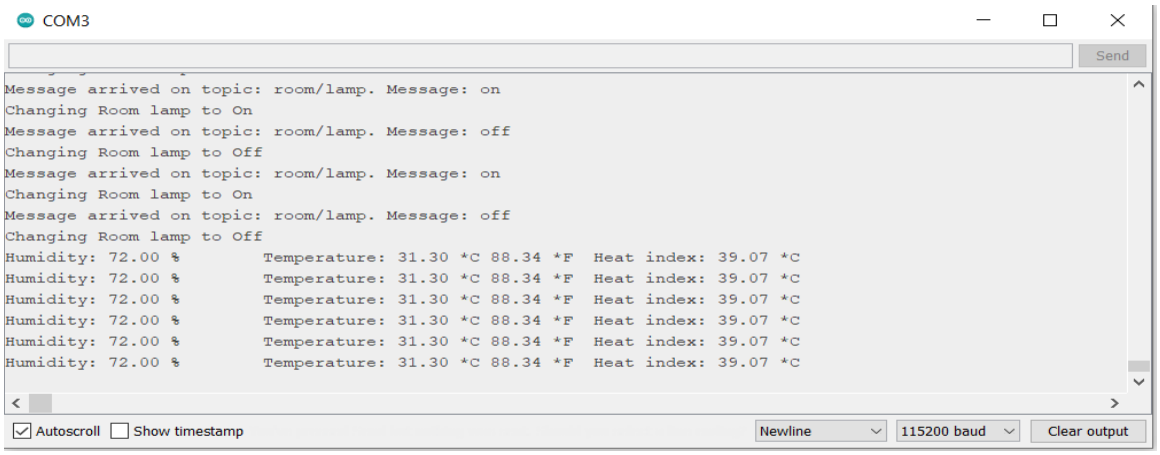


Figure 4: Graphic user interface

3.2 Discussions

The humidity level and the temperature level be easily observed in the GUI. The DHT11 temperature-humidity sensor will send data to the server every 30 seconds, which means it will still update data until the server is switched off. The switch, can be used to turn on or off the light source connected to the ESP8266. Figure 5 shows the data sent and received by the ESP8266 in Arduino IDE's serial monitor.

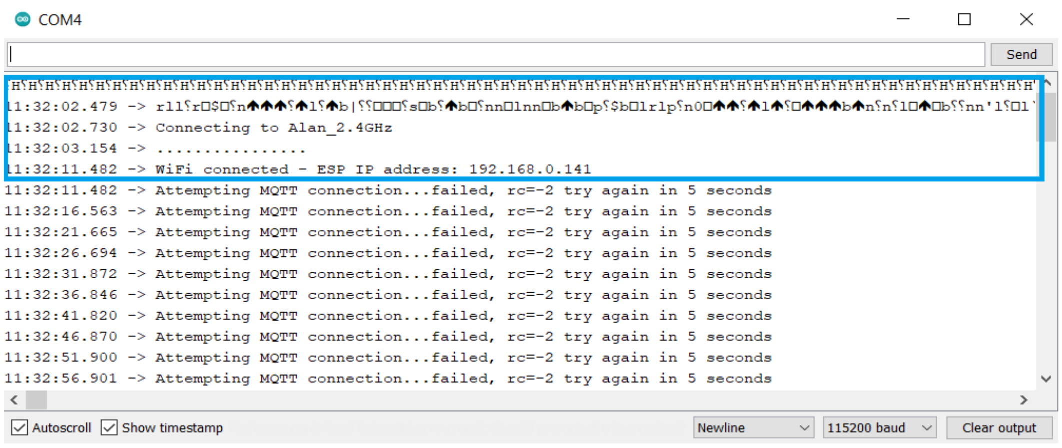


```
Message arrived on topic: room/lamp. Message: on
Changing Room lamp to On
Message arrived on topic: room/lamp. Message: off
Changing Room lamp to Off
Message arrived on topic: room/lamp. Message: on
Changing Room lamp to On
Message arrived on topic: room/lamp. Message: off
Changing Room lamp to Off
Humidity: 72.00 %        Temperature: 31.30 *C 88.34 *F  Heat index: 39.07 *C
Humidity: 72.00 %        Temperature: 31.30 *C 88.34 *F  Heat index: 39.07 *C
Humidity: 72.00 %        Temperature: 31.30 *C 88.34 *F  Heat index: 39.07 *C
Humidity: 72.00 %        Temperature: 31.30 *C 88.34 *F  Heat index: 39.07 *C
Humidity: 72.00 %        Temperature: 31.30 *C 88.34 *F  Heat index: 39.07 *C
Humidity: 72.00 %        Temperature: 31.30 *C 88.34 *F  Heat index: 39.07 *C
```

Figure 5: Arduino IDE's serial monitor

3.3 Analysis

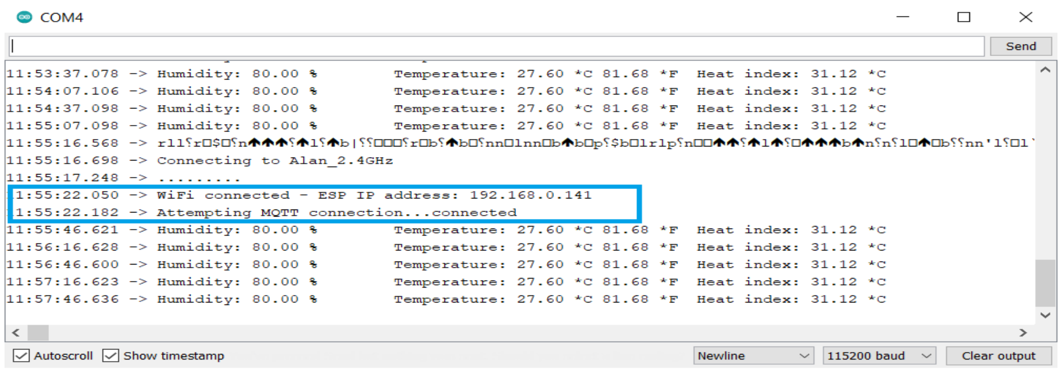
MQTT offers faster connection to the broker than HTTP, as according to 3G networks, throughput of MQTT is 93 times faster than HTTP's. The time taken for the ESP8266 to connect to the Wi Fi is approximately 9 seconds as shown in Figure 6.



```
H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H⸮H'
11:32:02.479 -> rll⸮r□$□⸮n⸮⸮⸮⸮⸮l⸮⸮b|⸮⸮□□□⸮s□b⸮⸮b□⸮nn□lnn□b⸮b□p⸮$b□lrlp⸮n0□⸮⸮⸮⸮l⸮⸮□⸮⸮⸮b⸮n⸮n⸮l□⸮□b⸮⸮nn'l⸮□l
11:32:02.730 -> Connecting to Alan_2.4GHz
11:32:03.154 -> ................
11:32:11.482 -> WiFi connected - ESP IP address: 192.168.0.141
11:32:11.482 -> Attempting MQTT connection...failed, rc=-2 try again in 5 seconds
11:32:16.563 -> Attempting MQTT connection...failed, rc=-2 try again in 5 seconds
11:32:21.665 -> Attempting MQTT connection...failed, rc=-2 try again in 5 seconds
11:32:26.694 -> Attempting MQTT connection...failed, rc=-2 try again in 5 seconds
11:32:31.872 -> Attempting MQTT connection...failed, rc=-2 try again in 5 seconds
11:32:36.846 -> Attempting MQTT connection...failed, rc=-2 try again in 5 seconds
11:32:41.820 -> Attempting MQTT connection...failed, rc=-2 try again in 5 seconds
11:32:46.870 -> Attempting MQTT connection...failed, rc=-2 try again in 5 seconds
11:32:51.900 -> Attempting MQTT connection...failed, rc=-2 try again in 5 seconds
11:32:56.901 -> Attempting MQTT connection...failed, rc=-2 try again in 5 seconds
```

Figure 6: ESP8266's connection time

Just as the server can go online in a matter of few seconds, the time taken for the ESP8266 to connect to the server can be done immediately. From the analysis, the time taken for the ESP8266 to connect to the server take an approximately 0.1 seconds as shown in Figure 7.



```
11:53:37.078 -> Humidity: 80.00 %        Temperature: 27.60 *C 81.68 *F  Heat index: 31.12 *C
11:54:07.106 -> Humidity: 80.00 %        Temperature: 27.60 *C 81.68 *F  Heat index: 31.12 *C
11:54:37.098 -> Humidity: 80.00 %        Temperature: 27.60 *C 81.68 *F  Heat index: 31.12 *C
11:55:07.098 -> Humidity: 80.00 %        Temperature: 27.60 *C 81.68 *F  Heat index: 31.12 *C
11:55:16.568 -> rll?r□$□?n????l??b|??□□□?r□b??b□?nn□lnn□b?b□p?$b□lrlp?n□□???l??□???b?n?n?l□?□b??nn'l?□l`
11:55:16.698 -> Connecting to Alan_2.4GHz
11:55:17.248 -> .........
1:55:22.050 -> WiFi connected - ESP IP address: 192.168.0.141
1:55:22.182 -> Attempting MQTT connection...connected
11:55:46.621 -> Humidity: 80.00 %        Temperature: 27.60 *C 81.68 *F  Heat index: 31.12 *C
11:56:16.628 -> Humidity: 80.00 %        Temperature: 27.60 *C 81.68 *F  Heat index: 31.12 *C
11:56:46.600 -> Humidity: 80.00 %        Temperature: 27.60 *C 81.68 *F  Heat index: 31.12 *C
11:57:16.623 -> Humidity: 80.00 %        Temperature: 27.60 *C 81.68 *F  Heat index: 31.12 *C
11:57:46.636 -> Humidity: 80.00 %        Temperature: 27.60 *C 81.68 *F  Heat index: 31.12 *C
```

Figure 7: MQTT's connection time

4. Conclusion

This project has successfully created a connection between the user and the ESP8266, allowing the user to monitor the temperature and humidity, as well as control the lightbulb using the user interface that has been set up. The temperature and humidity can be measured by using the sensor DHT11as input and connecting it to ESP8266. The DHT11 sensor can measure temperature from 0^o C to up to 50^o C, while the humidity that can be measured ranges from 20% of humidity level to 90% humidity level, with the accuracy of ±1^{o}C and ±1%. The input is transmitted to ESP8266, which is then sent to the server and finally to the user interface. The temperature and humidity can be tracked by the user from time to time, as the sensor transmits input to the ESP8266 to be processed. The processed information is then received by the server that has been set up beforehand, which then transmit the data to the user interface as the output. This user interface acts both ways, as the user can interact with the user interface. Smart Room System with remote access using MQTT is successfully developed by using the specification verified.

Acknowledgement

The authors would like to thank the Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia for its support.

References

[1] A. M. Sukumar, N. S. Thanjan, M. Varghese, and E. Engineering, "Abstract :," vol. 4, no. 1, pp. 24–39, 2017.

[2] M. Ibrahim, "Design and Implementation of Smart Room," no. May, 2020, doi: 10.22214/ijraset.2020.5327.

[3] C. Wilson, T. Hargreaves, and R. Hauxwell-Baldwin, "Benefits and risks of smart home technologies," *Energy Policy*, vol. 103, no. January, pp. 72–83, 2017, doi: 10.1016/j.enpol.2016.12.047.

[4] M. M. Froufe, C. K. Chinelli, A. L. A. Guedes, A. N. Haddad, A. W. A. Hammad, and C. A. P. Soares, "Smart Buildings: Systems and Drivers," *Buildings*, vol. 10, no. 9, p. 153, 2020, doi: 10.3390/buildings10090153.

[5] M. Kashyap, V. Sharma, and N. Gupta, "Taking MQTT and NodeMcu to IOT: Communication in Internet of Things," *Procedia Comput. Sci.*, vol. 132, no. Iccids, pp. 1611–1618, 2018, doi: 10.1016/j.procs.2018.05.126.

[6] D. S. Ugalde, "Security analysis for MQTT in Internet of Things," *Degree Proj. Comput. Sci. Eng.*, 2018.