Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 132 (2018) 1611–1618

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Intelligence and Data Science (ICCIDS 2018)

Taking MQTT and NodeMcu to IOT: Communication in Internet of Things

Monika Kashyap*, Vidushi Sharma, Neeti Gupta

Gautam Buddha University,Greater Noida-201308 India

Abstract

Internet of Things (IoT) allow connection among devices using internet with the ability to gather and exchange data. These devices are usually attached with micro-controllers like Arduino, sensors, actuators and internet connectivity. In this context, Message Queuing Telemetry Transport protocol (MQTT) plays an important role to exchange the data or information between the devices in IoT without knowing the identities of each other. This paper presents different service models for communication in Internet of Things(IoT). Model A presents use of serial USB as transmission medium while Model B uses the Message Queuing Telemetry Transport protocol (MQTT) which deploy a Wi-Fi module (ESP8266-12) to connect the system to internet. For communication, concept of publisher and subscriber is used. Messages are published or subscribed with the help of a broker or server. This agent is in charge of dispersing messages to intent clients depending on the choice of the topic of a message. Broker in MQTT is also called server. Some brokers used in MQTT are: -Mosquitto, Adafruit, hiveMQ

E-mail address: mkashyap856@gmail.com, svidushee@gmail.com, neeti.gupta04@gmail.com

Keywords: Arduino, ESP8266, HiveMQ, Internet Of Things(IOT), Message Queuing Telemetry Transport (MQTT).

1. Introduction

Internet of Things (IoT) provide the ability to connect a large number of things or devices via internet. These things or devices have unique identities. IoT creates a smart environment by connecting devices with internet and equip them with the ability to gather and exchange data. These devices or gadgets are usually connected with micro-controllers, sensors, actuators and internet connectivity. Such gadgets may include regular household items like washing machines, refrigerators, sound systems, coffee makers, alarm clocks etc. Also, IoT application in smart cities include traffic monitoring, air and water pollution monitoring, electrical energy consumption monitoring etc. IoT provides a platform to different objects where they can communicate with each other while providing them the ability to self-organize. In IoT lightweight protocols like MQTT and CoAP are used for data transmission. Message Queuing Telemetry Transport (MQTT) is a protocol that is used in IoT for data transmission [1][2]. It is a publisher and subscriber based protocol which allows multiple devices to communicate with each other over a wireless network. The top most layer of TCP/IP is Application layer and MQTT protocol is used in this layer. In MQTT publisher and subscriber (or clients) do not need to know each other's identity. MQTT deliver the information from source to destination and implemented on the TCP layer. MQTT is preferably suited for the IoT nodes which have constrained abilities and assets. Any MQTT connection consider two types of agents: the first is clients of MQTT and another is MQTT broker server. The Information transmitted by protocol is known as application message. MQTT client refers to the devices or objects connected to the network that take part in communication or exchange messages through MQTT. The MQTT clients are named as publisher and subscriber. A publisher can send the application messages and subscriber can request for that application messages to get the information associated with that message. The broker permits the different clients to connect with each other. It acknowledges and transmits the application messages among different clients associated with it. MQTT client can be a sensor, mobile etc. In this paper we use HiveMQ as a broker. Wi-Fi is picked as the method of correspondence in the model and the devices are monitored using MQTT convention executed utilizing ESP8266.

In this paper [3], the creators depicted the current designs for home mechanization and proposed a home computerization architecture offering capacity to the entire modern IoT protocols. In [4], the authors have talked about the Arduino service interface programming model, a modern programming model that gives an administration deliberation to effectively compute new capabilities to microcontroller and furthermore offer help for networked boards utilizing a scope of systems, including MQTT, socket connections and so on. In [5], a model is intended to perform home computerization through SMS. GSM organize and the gadgets are connected utilizing a microcontroller. It likewise centers around the security angles in systems administration and proposes a protected, solid and versatile home mechanization framework.

The rest of the paper is composed as follows: System layout is displayed in section II. Implementation details are given in section III. Section IV demonstrates the results and analysis of the model. Section V represents the conclusion.

2. System layout

Figure 1 shows the working of the MQTT server. Message Queuing Telemetry Transport (MQTT) refers to a protocol that is used in IoT for data transmission.

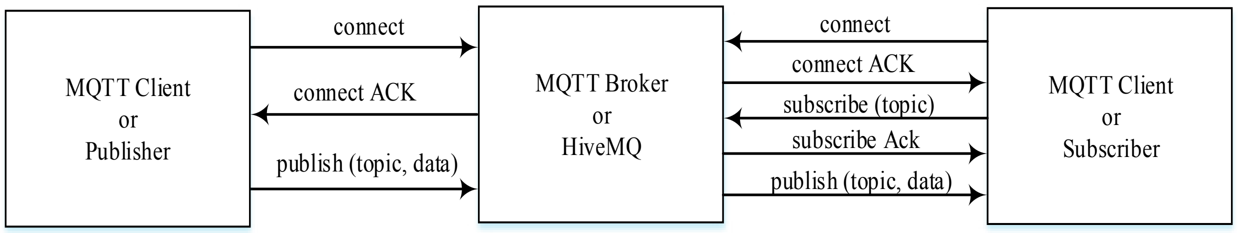


Fig. 1: Working of the system

The connection in MQTT consider the following types of operators: the first is MQTT client and another is called MQTT broker or we can say it MQTT server. The Information delivered by MQTT is known as application message. When MQTT client wants to broadcast or publish some information to the MQTT broker, the client needs to establish a connection with the MQTT broker. The client requests to broker to connect with it, then broker sends the acknowledgment of that connection request to client after connection establishment between the client and the MQTT broker. The client can send or publish the topic to the broker. Similarly, any other client who wants the information of the same topic as published by client 1, can send the request to connect to broker, then broker sends the acknowledgment, if the second client connect with the broker. It can subscribe to any topic on the broker side, or the client can publish a new message to broker. Therefore, this system allows the clients to communicate with each other without knowing each other. In this paper, we have presented some service models related to communication in IoT using MQTT like connecting one publisher and many subscribers to broker.

3. Implementation Details

This section gives the implementation details for various service models.

3.1. Network setup and System Implementation:

Requirement for setup:

i. Node MCU esp8266
ii. Arduino
iii. LED (Light Emitting Diode)
iv. Breadboard
v. Wires (Male to Female)
vi. Laptops with internet connection
vii. 420 OHM resistor

3.1.1. Service model A: Communication in IoT using Serial connection.

Connect client to Arduino(micro-controller) using a Serial connection(USB). In Figure 2, three led are connected to Arduino in pin number 3,4,5 respectively. These led turn off or on according to the instruction given by the system. This is the Service model of communication between the objects with the help of USB or wire or without Wi-Fi.

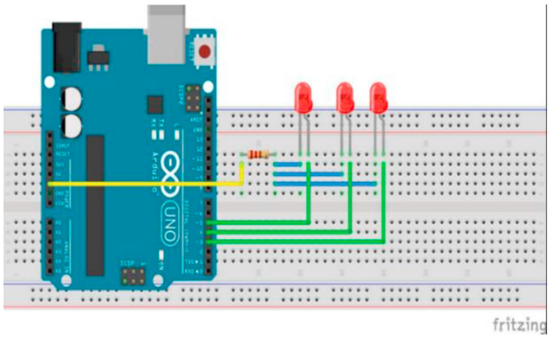


Fig. 2: connection of Arduino

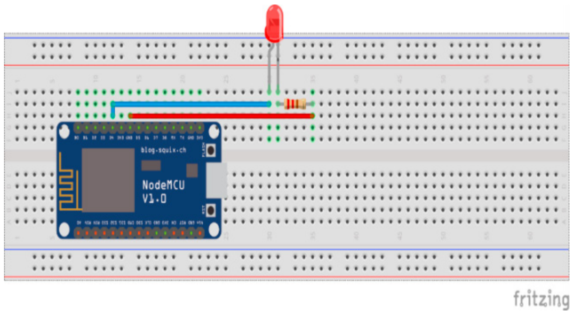


Fig. 3: LED Circuit for the subscriber side

3.1.2. Service model B: Communication in IoT using Wi-Fi or internet (MQTT protocol)

Setup an MQTT connection using NodeMcu shown in Figure 3 [6], [7]. In Figure 3, NodeMcu is connected to a LED on the breadboard. This NodeMcu is connected to a system (laptop or computer) and this setup is connected to MQTT broker for further communication to perform the following tasks:

1. Publish and subscribe a message using MQTT broker on the same machine.

2. Publish a message on MQTT broker and subscribe to a topic from the server to obtain the information.

3. Set up a publisher on one device and a subscriber on another device. LED circuit (as shown in figure 3) is connected on the side of subscriber. This circuit turn ON or OFF according to the message published by the client on the MQTT server. The purpose of this Service model is to build a MQTT publisher-subscriber system so that the publisher or client publishes digital values (0 and 1) to the broker and then the subscriber subscribe to the publisher's topic, information of that published message is then delivered to the subscriber through the broker. The subscriber's side LED will be turned ON or OFF according to the values entered on the client side.

4. Create one publisher and different subscribers, different publishers and one subscriber & different publishers and subscribers using MQTT along with it also observe that if there is a loss of data or lag while publishing messages from two subscribers at once. In this Service model, we have exploited the services of MQTT which can be used to communicate between several set of devices. Client publish a message and forward it to multiple subscribers. Also, we have presented a service model where client receive message from several subscribers and observe if there is any lag or loss of data due to simultaneous reception. In this manner, we have created a network of multiple publishers and subscribers.

The main difference between all the above service models will be in their code sketch.

Algorithm used:
Step 1: Connect to the Internet.
Step 2: If not connected to the internet, attempt reconnection.
Step 3: Else print the IP address of the device, on the serial monitor.
Step 4: Attempt MQTT connection with the specified MQTT Broker.
Step 5: Once connected we can publish the message on the broker.
Step 6: Devices which are connected to the broker also subscribe to a particular Topic, which are shown in broker's window.
This algorithm is applied for all the service model with MQTT in this paper.

Wiring Instructions for setup:
i. Take a breadboard.
ii. Insert the LED into that breadboard.
iii. Connect the cathode of the LED to the output pin of the Node MCU.

iv. The anode of the LED is connected to the ground (GND Pin) of Node MCU.
v. The Node MCU is powered via external USB.

A. Single publisher and multiple subscribers:
In case of multiple subscribers, the object name of a subscriber acts as the group identifier. MQTT broker will send information to only one member of the group at once and to all the members in a round robin fashion. Therefore, if we intend to have multiple subscribers, subscribing to the same publisher we must give different object names to the subscribers. Identification of the subscribers is independent of the network they are connected to.

B. Multiple publishers & a single subscriber:
Majority code will remain same. The only difference will be that the subscribers will subscriber to multiple topics i.e. the number of "client. Subscribe" will increase.

C. Multiple Publishers and Subscribers:
Majority of the code will remain same. The only difference will be that each publisher will publish multiple topics under "client. Publish" and each subscriber will subscribe to multiple topics using "client. Subscribe"

In all the service models discussed, we observed that though there are some lags, but there is no loss of data while transfer of multiple information over MQTT. This is the major advantage of MQTT over other protocols like HTTP.

3.2. Node MCU esp8266 Version 1.0
Node MCU (see figure 4) is an open source firmware and development kit that helps in prototyping IoT products [8]. The programming of Node MCU is similar to that of Arduino IDE. It has the following features:

1. Arduino like hardware I/O
2. USB TTL included, plug and play
3. Lowest cost Wi-Fi Lowest cost Wi-Fi

Fig. 4: Node MCU esp8266

3.3. Software origination

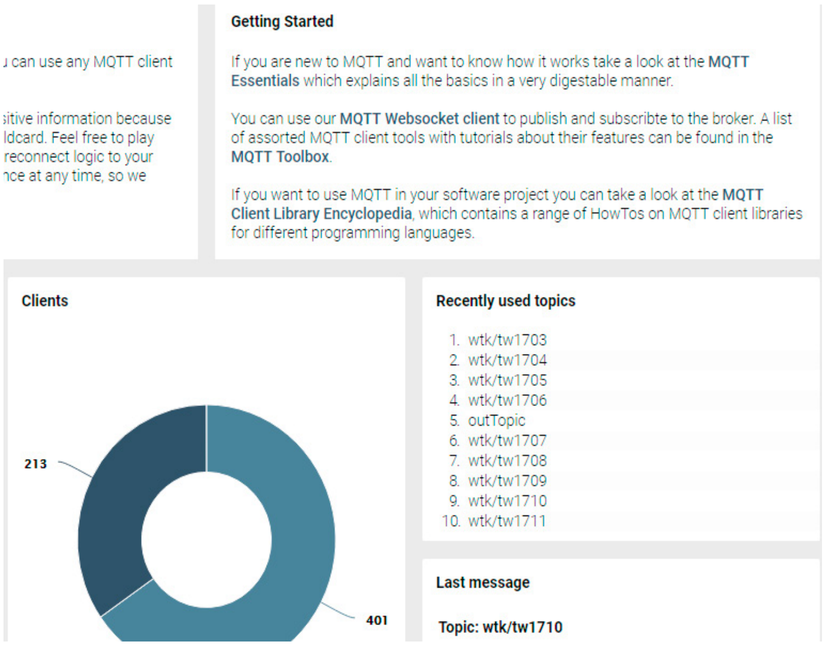
Arduino IDE is adopted as a software for programming in this paper. HiveMQ, an open source MQTT broker is used for the implementation. HiveMQ uses two services (pub and sub) to publish and subscribe to a messages. MQTT client request for the sensor information and distributes the orders to control GPIOs of ESP8266. MyMQTT, an android application is another MQTT client that associates with the Mosquitto MQTT broker and sends or distributes to a particular topic.

4. Results and Analysis

- In Service model A, Arduino is used for communication between computer and LED. In this, a client can connect to a computer using USB. There are three LEDs connected to the Arduino which blink according to the computer instructions. The communication here is done via serial USB rather than an internet or Wi-

Fi.

- In Service model B, MQTT server is set-up using HiveMQ. Client's published message i.e. "Out Topic" will appear on the HiveMQ broker's list of recently used topics. The message here is being published (see figure 5).


Getting Started

u can use any MQTT client

sitive information because ldcard. Feel free to play reconnect logic to your nce at any time, so we

If you are new to MQTT and want to know how it works take a look at the **MQTT Essentials** which explains all the basics in a very digestable manner.

You can use our **MQTT Websocket client** to publish and subscribte to the broker. A list of assorted MQTT client tools with tutorials about their features can be found in the **MQTT Toolbox**.

If you want to use MQTT in your software project you can take a look at the **MQTT Client Library Encyclopedia**, which contains a range of HowTos on MQTT client libraries for different programming languages.

Clients

Recently used topics

1. wtk/tw1703
2. wtk/tw1704
3. wtk/tw1705
4. wtk/tw1706
5. outTopic
6. wtk/tw1707
7. wtk/tw1708
8. wtk/tw1709
9. wtk/tw1710
10. wtk/tw1711

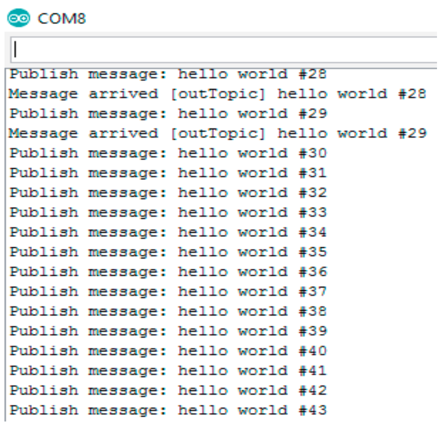
Last message

Topic: wtk/tw1710

Fig. 5: Published data on MQTT server (OutTopic)

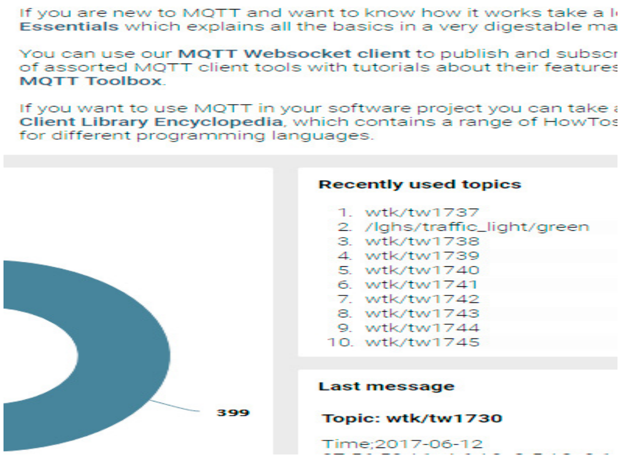
In figure 6, serial monitor of Arduino IDE is showing both the published and subscribed messages. Here the published message is 'outTopic' and the subscribed message is 'Hello World' and this published message appears in the Mqtt broker (mosquitto).

Figure 7 demonstrates subscription to a topic from the HIVE MQ broker. It is the subscriber's choice which topic he wants to choose; in our case we have used a topic: "Lghs/traffic_light/green"


COM8

```
Publish message: hello world #28
Message arrived [outTopic] hello world #28
Publish message: hello world #29
Message arrived [outTopic] hello world #29
Publish message: hello world #30
Publish message: hello world #31
Publish message: hello world #32
Publish message: hello world #33
Publish message: hello world #34
Publish message: hello world #35
Publish message: hello world #36
Publish message: hello world #37
Publish message: hello world #38
Publish message: hello world #39
Publish message: hello world #40
Publish message: hello world #41
Publish message: hello world #42
Publish message: hello world #43
```

Fig. 6: Published and subscribed data


If you are new to MQTT and want to know how it works take a l
Essentials which explains all the basics in a very digestable ma

You can use our **MQTT Websocket client** to publish and subscr
of assorted MQTT client tools with tutorials about their features
MQTT Toolbox.

If you want to use MQTT in your software project you can take a
Client Library Encyclopedia, which contains a range of HowTos
for different programming languages.

Recently used topics

1. wtk/tw1737
2. /lghs/traffic_light/green
3. wtk/tw1738
4. wtk/tw1739
5. wtk/tw1740
6. wtk/tw1741
7. wtk/tw1742
8. wtk/tw1743
9. wtk/tw1744
10. wtk/tw1745

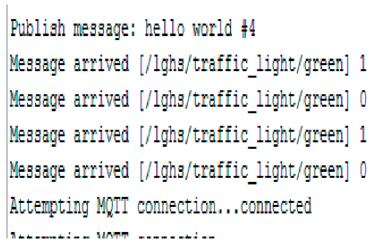
Last message

Topic: wtk/tw1730

Time:2017-06-12

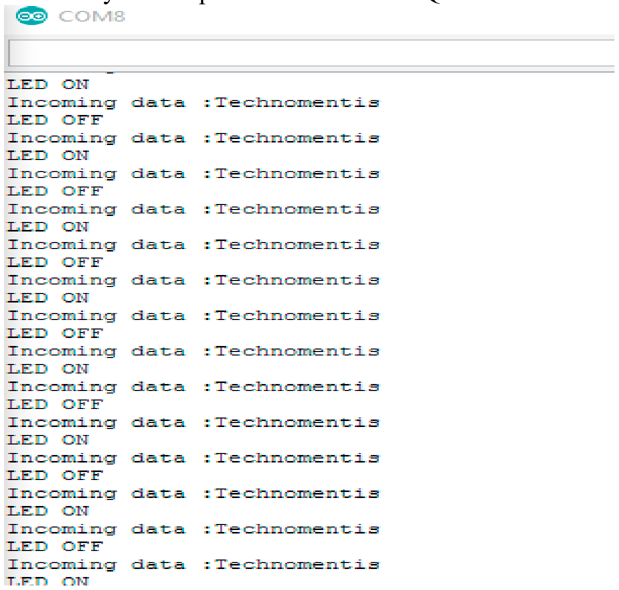
Fig. 7: The subscribed topic from the MQTT server

After subscribing to a particular topic, the serial monitor of Arduino IDE will display the information associated with this topic (see figure 8). We may subscribe to any of the recently used topics from the HiveMQ.



```
Publish message: hello world #4
Message arrived [/lghs/traffic_light/green] 1
Message arrived [/lghs/traffic_light/green] 0
Message arrived [/lghs/traffic_light/green] 1
Message arrived [/lghs/traffic_light/green] 0
Attempting MQTT connection...connected
```

Fig. 8: Information display on serial monitor of subscribed topic

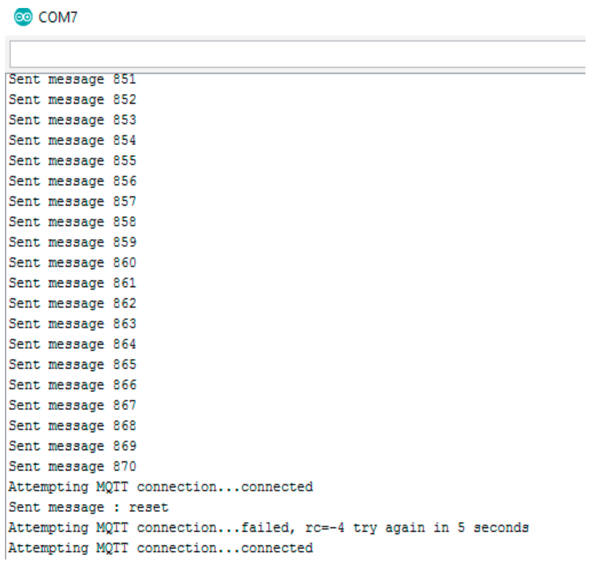


```
LED ON
Incoming data :Technomentis
LED OFF
Incoming data :Technomentis
LED ON
Incoming data :Technomentis
LED OFF
Incoming data :Technomentis
LED ON
Incoming data :Technomentis
LED OFF
Incoming data :Technomentis
LED ON
Incoming data :Technomentis
LED OFF
Incoming data :Technomentis
LED ON
Incoming data :Technomentis
LED OFF
Incoming data :Technomentis
LED ON
Incoming data :Technomentis
LED OFF
Incoming data :Technomentis
LED ON
Incoming data :Technomentis
LED OFF
Incoming data :Technomentis
LED ON
```

Fig. 9: Serial monitor output for LED status

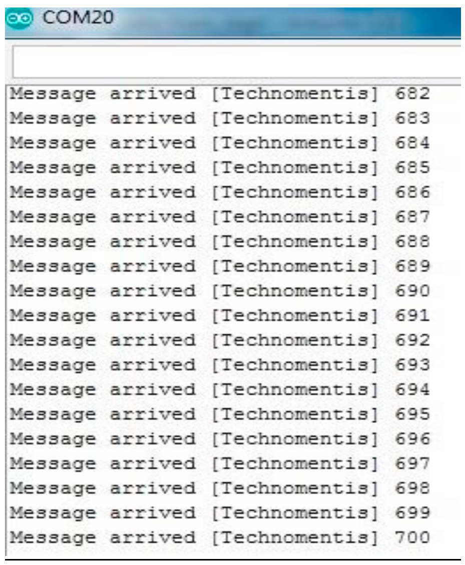
In Service Model B (point no. 3), LED will turn ON or OFF according to the command (e.g. ON or OFF) given by the publisher. The serial monitor is presented in figure 9:

Figure 10 shows the output of Service Model B (point no. 4). Publisher window shows the messages that are published by the client. Figure 11 shows the subscriber windows and the information of the subscribed topic.



```
Sent message 851
Sent message 852
Sent message 853
Sent message 854
Sent message 855
Sent message 856
Sent message 857
Sent message 858
Sent message 859
Sent message 860
Sent message 861
Sent message 862
Sent message 863
Sent message 864
Sent message 865
Sent message 866
Sent message 867
Sent message 868
Sent message 869
Sent message 870
Attempting MQTT connection...connected
Sent message : reset
Attempting MQTT connection...failed, rc=-4 try again in 5 seconds
Attempting MQTT connection...connected
```

Fig 10: Publisher's Serial Monitor



```
Message arrived [Technomentis] 682
Message arrived [Technomentis] 683
Message arrived [Technomentis] 684
Message arrived [Technomentis] 685
Message arrived [Technomentis] 686
Message arrived [Technomentis] 687
Message arrived [Technomentis] 688
Message arrived [Technomentis] 689
Message arrived [Technomentis] 690
Message arrived [Technomentis] 691
Message arrived [Technomentis] 692
Message arrived [Technomentis] 693
Message arrived [Technomentis] 694
Message arrived [Technomentis] 695
Message arrived [Technomentis] 696
Message arrived [Technomentis] 697
Message arrived [Technomentis] 698
Message arrived [Technomentis] 699
Message arrived [Technomentis] 700
```

Fig 11: Subscriber 1& 2's Serial Monitor

5. Conclusion

MQTT protocol permits the correspondence between devices. Considering the simplicity of remote web access through Wi-Fi, MQTT client application is based on ESP8266.This paper focussed on the execution of Arduino, NodeMcu and MQTT protocol and also describe different possible service models for communication among the IoT devices. The administration shows in this paper discuss the communication between client and server using wireless (Wi-Fi) and wired (USB) transmission medium. MQTT protocol is the main focused area in this work, we examine the different tasks using this protocol. We test this protocol by publish and subscribe the message on the same machine, different machine, and then we set up a publisher on one device and a subscriber on another device. LED is connected to the subscriber, this led ON or OFF according to the message published by the client on the broker and also examine the MQTT protocol with various cases like create one publisher and different subscribers, different publishers and one subscriber & different publishers and subscribers along with it also observe that if there is lost information or slack while distributing messages from two subscribers without a moment's delay. Thus we found that MQTT protocol is suitable for all cases defines above but speed of transfer information is less than the serial connection because it depends on the speed of Wi-Fi. But this protocol never lost the data while it is connected with Wi-Fi and it saves the data in a queue.

REFERENCES

[1] MQTT protocol specification. [Online] Available: "http://public.dhe. ibm.com/software/dw/web services/ws-mqtt/mqtt- v3r1.html".
[2] R A Atmoko, R Riantini, and M K Hasin. (2017) "IoT real time data acquisition using MQTT protocol" Journal of. Physics.: 853
[3] Nasrin S. and Radcliffe P. J. (2014) "Novel protocol enables DIY home automation." Telecommunication Networks and Applications Conference (ATNAC): 212–216.
[4] Gianluca Barbon, Michael Margolis, Filippo Palumbo, Franco Raimondi, Nick Weldin. (2016) "Taking Arduino to the internet of things: the ASIP programming model." computer communication: 1-15
[5] ElKamchouchi H. and ElShafee A. (2012) "Design and prototype implementation of SMS based home automation system." International Conference on Electronics Design, Systems and Applications (ICEDSA): 162–167
[6] MQTT version 3.1.1 becomes an oasis standard [On-line]. Available: https://www.oasis open.org/news/announcements/mqtt-version-3-1-1-becomes-an-oasis-standard.
[7] MQTT version 3.1.1 oasis standard. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.
[8] NodeMcu [Online]. Available: http://nodemcu.com/index en.html/
[9] HiveMQ [Online] Available: "http://www.hivemq.com"