ENVIRONMENTAL AND EARTH SCIENCES RESEARCH JOURNAL

A publication of IIETA

ISSN: 2369-5668 (Print), 2369-5676 (Online)
Vol. 4, No. 1, March 2017, pp. 23-28
DOI: 10.18280/eesrj.040105
Licensed under CC BY-NC 4.0

IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering
http://www.iieta.org/Journals/EESRJ

Study and implementation of environment monitoring system based on MQTT

Sumit Pal[1*], Sourav Ghosh[2], Sarasij Bhattacharya[3]

[*1]Department of Information Technology, Indian Institute of Engineering Science and Technology, Howrah
[2] Assistant Systems Engineer Trainee, Tata Consultancy Services, Kolkata
[3] Programmer Analyst Trainee, Cognizant Technology Solutions, Kolkata

Email: sumitpal808@gmail.com

ABSTRACT

In the past few years, technology has developed a lot in making our lives simpler. With the evolution of internet and mobile technology coupled with the parallel development of a variety of embedded systems, the focus towards a smart world has gained momentum and has given us a new concept, the Internet of Things. As a result, much of today's technology is automated and the developers are developing systems which gather data from various sensor systems which can be sent to any part of world through the Internet and can be used for a vareity of purposes, including controlling of devices. In this paper, we have attempted to study one such Internet protocol which makes such a communication possible, the MQTT protocol. Using an Environmental monitoring system, we will determine the viability of such a protocol for transmission of sensor data and then use the same data to control electronic devices. We also compare the MQTT protocol with the traditional HTTP protocol and attempt to find out which protocol is the better one.

Keywords: MQTT Protocol, Internet of Things, Mobile Technology, Embedded Systems, Communication.

1. INTRODUCTION

The IoT envisions hundreds or thousands of end-devices with sensing, actuating, processing and communicating.[1][2] In the early 2000's, Kevin Ashton was laying the groundwork for what would become the Internet of Things (IoT) at MIT's AutoID lab. If all objects in daily life were equipped with identifiers and wireless connectivity, these objects could be communicate with each other and be managed by computers. The Internet of Things will: [3]

(1) **Connect both inanimate and living things**: The types of items range from gas turbines, automobiles to living organisms such as plants, farm animals and people.

(2) **Use sensors for data collection:** In IoT, these sensors will connect to each other and to systems that can understand or present information from the sensor's data feeds.

(3) **Change what types of item communicate over an IP Network:** This information can be shared in real-time or collected and shared at defined intervals. [3]

2. THE MQTT PROTOCOL

MQTT or the Message Queuing Telemetry Transport Protocol is a simple and lightweight messaging protocol. It's publish/subscribe mechanism is designed to be easy to implement.[4] These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium [5]

The protocol runs over TCP/IP on the application layer in the OSI layer along with other protocols like CoAP, HTTP etc. in the 6LoWPAN stack.

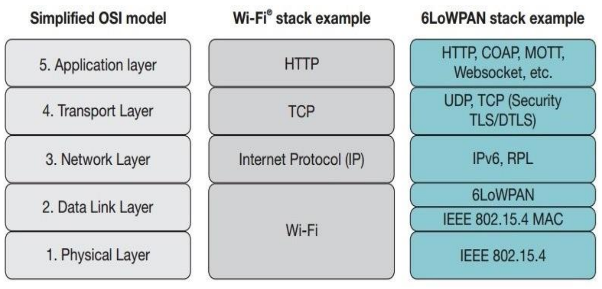


Figure 1.

Its features include:

- Use of the publish/subscribe message pattern which provides one-to-many message distribution and decoupling of applications.
- A messaging transport that is agnostic to the content of the payload.
- Three qualities of service for message delivery:

(1) "At most once", where messages are delivered according to the best efforts of the operating environment. Message loss can occur.
(2) "At least once", where messages are assured to arrive but duplicates can occur.
(3) "Exactly once", where message are assured to arrive exactly once.

- A small transport overhead and protocol exchanges minimized to reduce network traffic.

MQTT applications run on mobile devices, such as smartphones and tablets. MQTT is also used for telemetry to receive data from sensors, and to control them remotely. The MQTT client library is small. The library acts like a mail box, sending and receiving messages with other MQTT applications that are connected to an MQTT server. Using this mechanism, MQTT applications conserve battery life.

3. MQTT MESSAGE FORMAT

Before going into the depth of our experiment and demonstrating how MQTT functions, there are a few components that we should be aware of. As said before, MQTT is a publish/subscribe protocol. It means that a user publishes something and another user must subscribe to it before receiving it. Thus, MQTT works on the exchange of messages from one user to the other. The messages of MQTT have a format. The format is as follows:

Table 1. MQTT message format

Bits								
Byte	0	1	2	3	4	5	6	7
1	Message Type				DUP	QOS Level		Retain
2	Remaining Length (1 to 4 Bytes)							
3to n	Optional: Variable Header Length							
n+1 to m	Optional: Variable Length Message Payload							

While the header might be little bit complicated to understand at the moment, we will soon explain it with respect to a sensor network scenario. We will show you how advantageous is MQTT especially when considering the mobile environment. In our experiments, we have used an open source microcontroller known as Arduino Uno. Open source hardware is a type of hardware whose design is made publicly available so that anyone can study, modify, distribute, make, and sell the design or hardware based on that design [6] We will be coupling with it an Ethernet shield and attach it with a DHT22 temperature and humidity sensor to record the temperature data. We will then use this data and then publish the data on the internet. Then, we will switch an electronics device with respect to that data we have received.

4. MORE DETAILS ABOUT PAPER TITLE AND AUTHOR INFORMATION

4.1 Paper title

Study and Implementation of Environment Monitoring System using MQTT

4.2 Author information

4.2.1 Name
Sumit Pal [1*]
Sourav Ghosh [2]
Sarasij Bhattacharya [3]

4.2.2 Affiliation
Master of Technology Student in Information Technology, Indian Institute of Engineering Science and Technology, Howrah
Assistant Systems Engineer Trainee, Tata Consultancy Services, Kolkata
Programmer Analyst Trainee, Cognizant Technology Solutions, Kolkata

5. THE EXPERIMENT

5.1 Devices Required

(1) Arduino Uno Microcontroller: The primary component of our project is the Arduino Uno micro controller. The Uno is a microcontroller board based on the ATmega328P. [8]
(2) Ethernet Shield: The second component is an Ethernet shield. The Arduino Ethernet Shield connects your Arduino to the internet in mere minutes.
(3) DHT22 temperature and humidity sensor: The DHT22 is a relative cheap sensor for measuring temperature and humidity.
(4) A Router or a direct Internet Connectivity: It is required to send our sensor data to the internet. For our purpose, we have used a router.
(5) Relay Module: is used for switching an electronics device based on the temperature data that we have received from the DHT22 temperature and humidity sensor.

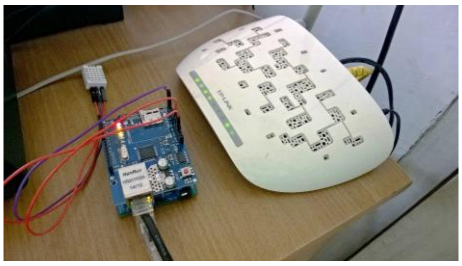


Figure 1. Arduino connected to the internet

5.2 The setup and working

The Relay module used has 4 pins, a ground or GND, 2 input pins or IN1 AND IN2 and a VCC pin. The 2 input pins are connected with the analog pins of Arduino marked A1 to A5. Similarly, the DHT22 has 3 pins, a GND, a data pin and a VCC pin. The data pin is connected to the digital pins of

Arduino marked 1 to 13. The ground and the VCC pins for all the 3 devices is the breadboard that we have used.

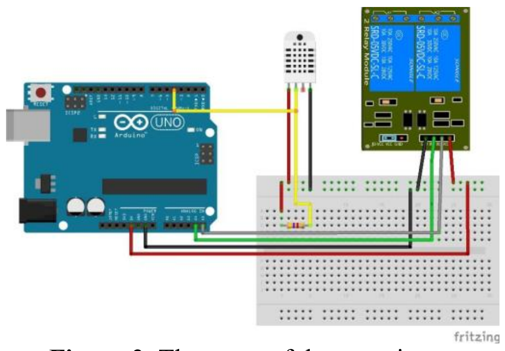


Figure 2. The setup of the experiment

5.3 Implementing MQTT

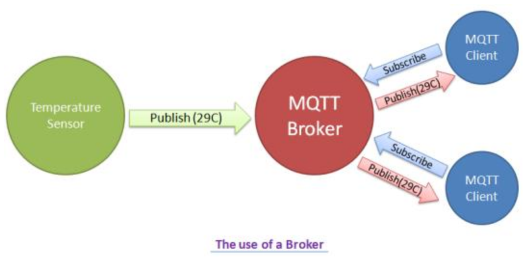


Figure 3. The use of a Broker

The publish/subscribe pattern (pub/sub) is an alternative to the traditional client-server model, where a client communicates directly with an endpoint. However, Pub/Sub decouples a client, who is sending a particular message (called publisher) from another client (or more clients), who is receiving the message (called subscriber). This means that the publisher and subscriber don't know about the existence of one another. This is done by a broker.

Another responsibility of the broker is the authentication and authorization of clients [9] There are two popular brokers that are present and they are the HIVE broker and the Mosquitto broker. For our experimental purposes, we have used both the HIVE and the Mosquitto broker. Using the HIVE broker, we have developed an android application that would help us to switch from our mobile phones and using Mosquitto, we have implemented in the desktop.

So at first, we have to publish our sensor data that we receive from the DHT22 server and it goes to the interested party who has subscribed to it. Now the question arises, how can it be possible that the broker sends the correct data to the correct party? For that, the broker has an interesting mechanism known as message filtering. It can be done in three ways:

(1) Subject-based filtering: The filtering is based on a subject or topic, which is part of each message. The receiving client subscribes on the topics it is interested in. [8]

(2) Content-based filtering: Content-based filtering is as the name already implies, when the broker filters the message based on a specific content filter-language. [8]

(3) Type-based filtering: When using object-oriented languages it is a common practice to filter based on the type/class of the message (event) [8]

The MQTT connection itself is always between one client and the broker, no client is connected to another client directly. The connection is initiated through a client sending a CONNECT message to the broker.

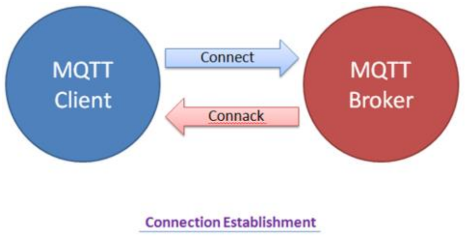


Figure 5. Connecting to MQTT Broker

Now, let's look at the MQTT CONNECT command message. A good-natured client will send a connect message with the following content among other things:

(1) Client ID: The client identifier (short ClientId) is an identifier of each MQTT client connecting to a MQTT broker.

(2) Clean Session: The clean session flag indicates the broker, whether the client wants to establish a persistent session or not.

(3) Username/Password: MQTT allows to send a username and password for authenticating the client and also authorization.

Now we come to the part of publishing and subscribing the data. The MQTT works on subscribe and publish mechanism just like the HTTP works on the request and response mechanism.

Publish: After a MQTT client is connected to a broker, it can publish messages. MQTT has a topic-based filtering of the messages on the broker, so each message must contain a topic, which will be used by the broker to forward the message to interested clients [8]

1. **Topic Name:** Is a UTF-8 string used by the broker to filter messages for each connected client.
2. **Quality of Service:** The Quality of Service (*QoS*) level is an agreement between sender and receiver of a message regarding the guarantees of delivering a message.
3. **Payload:** This is the actual content of the message.
4. **Packet Identifier:** The packet identifier is a unique identifier between client and broker to identify a message in a message flow.
5. **DUP flag:** The duplicate flag indicates, that this message is a duplicate and is resent because the other end didn't acknowledge the original message.

Subscribe: Publishing messages doesn't make sense if no one ever receives the message, or, in other words, if there are no clients subscribing to any topic.

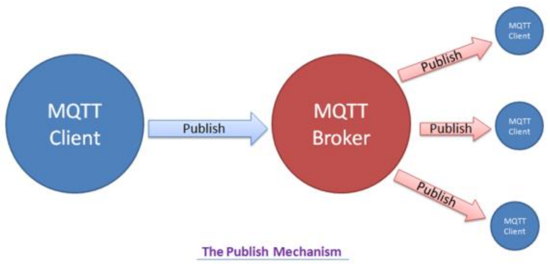


Figure 6. The publish and subscribe mechanism

1. **Packet Identifier:** The packet identifier is a unique identifier between client and broker to identify a message in a message flow.
2. **List of Subscriptions:** A SUBSCRIBE message can contain an arbitrary number of subscriptions for a client. Each subscription is a pair of a topic topic and QoS level. [7]

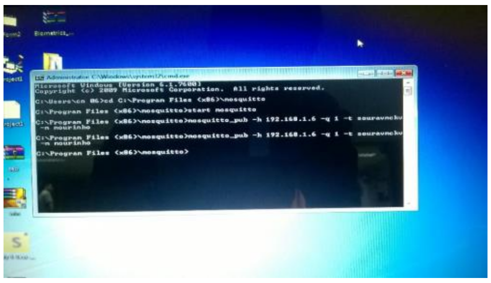
6. RESULTS

6.1 Using desktop and Mosquitto broker

Using the Mosquitto MQTT broker, we establish a server in a Desktop computer. The install the broker will act as a server using which we issue the Publish and Subscribe commands. Just like in a web server, clients (Desktop MQTT Client or Android MQTT app Client) will contact the server with subscription requests and once the connection is established, they will get the required data. One thing to be noted is that, the communicating parties also need to have the MQTT framework for the communication.

Publication can be done by the following command: Mosquitto_pub –t 'test/topic' –m 'helloWorld'

This is of course a very simple implementation of the publish command. Inside the publish command, we can specifically mention the hostname and also mention the QoS. Let us consider the machine on which we have published the data as PC1



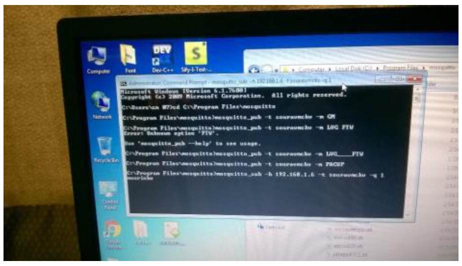
```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\cn 06>cd C:\Program Files (x86)\mosquitto

C:\Program Files (x86)\mosquitto>start mosquitto

C:\Program Files (x86)\mosquitto>mosquitto_pub -h 192.168.1.6 -q 1 -t souravmckv
 -m mourinho

C:\Program Files (x86)\mosquitto>mosquitto_pub -h 192.168.1.6 -q 1 -t souravmckv
 -m mourinho

C:\Program Files (x86)\mosquitto>
```

Figure 7. The PUB command in Mosquitto Server in PC1

After the publication is done from PC1, we can subscribe to the published data using the following command:

Mosquitto_sub –v –t 'test/topic'

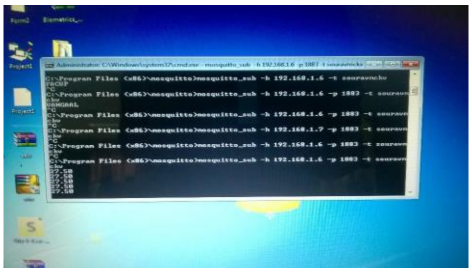
Let us consider the desktop from which we have subscribed to be PC2. Just like with the publish command, we can tweak it and include the host name and also the QoS



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\cn 07>cd C:\Program Files\mosquitto

C:\Program Files\mosquitto>mosquitto_pub -t souravmckv -m GM

C:\Program Files\mosquitto>mosquitto_pub -t souravmckv -m LVG FTW
Error: Unknown option 'FTW'.

Use 'mosquitto_pub --help' to see usage.

C:\Program Files\mosquitto>mosquitto_pub -t souravmckv -m LVG____FTW

C:\Program Files\mosquitto>mosquitto_pub -t souravmckv -m FACUP

C:\Program Files\mosquitto>mosquitto_sub -h 192.168.1.6 -t souravmckv -q 1
mourinho
```

Figure 8. The SUB command in mosquitto server

After successfully writing down the publish and subscribe commands, we should receive the sensor data that we have obtained through the DHT22 sensor in PC2, the desktop from which we subscribed to the information published from PC1



```
C:\Program Files (x86)\mosquitto>mosquitto_sub -h 192.168.1.6 -t souravmckv
FACUP
^C
C:\Program Files (x86)\mosquitto>mosquitto_sub -h 192.168.1.6 -p 1883 -t sourav
ckv
VANGAAL
^C
C:\Program Files (x86)\mosquitto>mosquitto_sub -h 192.168.1.6 -p 1883 -t sourav
ckv
^C
C:\Program Files (x86)\mosquitto>mosquitto_sub -h 192.168.1.7 -p 1883 -t sourav
ckv
^C
C:\Program Files (x86)\mosquitto>mosquitto_sub -h 192.168.1.6 -p 1883 -t sourav
ckv
^C
C:\Program Files (x86)\mosquitto>mosquitto_sub -h 192.168.1.6 -p 1883 -t sourav
ckv
27.50
27.50
27.50
27.50
27.50
```

Figure 9. Receiving the temperature data

As we said before, we would like to switch the data. Using the data that we have received, we will switch on a fan to show that the MQTT protocol can help us in controlling devices easily. For this purpose, we will use this command.

Mosquitto_pub -h (hostname) –t (topicname) –q (qos) –m fanon

To turn the fan off, we use the following command:

Mosquitto_pub -h (hostname) –t (topicname) –q (qos) –m fanoff

After using both this commands, we see that the fan succesfully turned on and off with minimal delay.

6.2 Using mobile and HIVE broker

We developed and android application and in our application, we have used the HIVE broker. The application is coded in such a way that the temperature data will appear as push notifications whenever we have subscribed. Similarly, the options of switching is available in GUI inside the application itself.

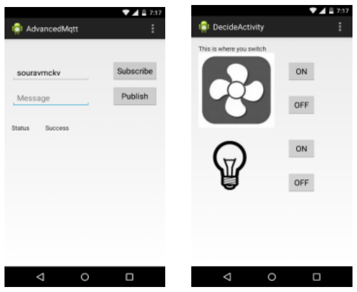


Figure 10. The GUI of our Android application

6.3 Result analysis

We now analyze the data that we have received to see how accurate the readings taken from the sensor are with respect to that of original reading.

We present a graphical representation of the temperature values which shows us that there is not much difference in between the temperatures and so the readings can be considered accurate for experimental purposes. Like the temperature data, we can measure the humidity as well. The DHT22 sensor is well equipped to measure the humidity data as well following the same procedures as we did for the temperature data. However, for experimental purposes, we represent only the temperature data.

Serial No	Temperature in F (sensor)	Temperature in F (Original)
1	27.5	28
2	27.5	28
3	27.5	28
4	28.2	28
5	28.3	28
6	28	28
7	29.7	30
8	29.7	30
9	29.7	30
10	29.7	30
11	30	30
12	30.2	30

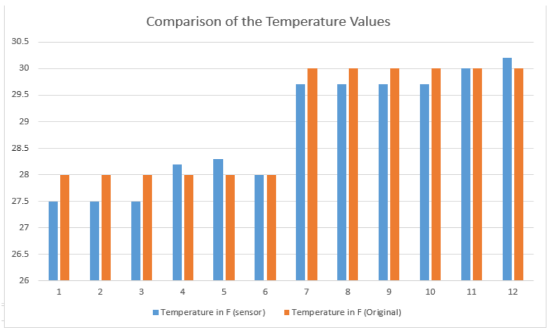


Figure 11. Graph to compare the accuracy of our data

7. COMPARISONS

So far, we have utilized a variety of tools to serve our purpose. We have used a different protocol known as MQTT instead of the traditional HTTP to serve our purpose. We also have used two different brokers in two different situations. Our purpose for this is to find out which broker is best suited for our purpose and also to learn if the MQTT protocol is indeed advantageous over the HTTP protocol

7.1 MQTT vs HTTP

MQTT	HTTP
1) MQTT is a lightweight publish/subscribe messaging protocol.	1) HTTP is designed as a request-response protocol for client-server computing.
2) It is very useful in Mobile Application Development.	2) It is not necessarily optimized for mobile and push capabilities
3) It is much more reliable, three Qos and patterns to avoid packet loss on client disconnection.	3) No quality of service in case of HTTP.
4) MQTT features faster response and throughput, and lower battery and bandwidth usage	4) HTTP is not optimized for low power usage and bandwidth usage is usually more.
5) MQTT is a very secure protocol and uses Username and Password in the CONNECT message.	5) Secure is a very big concern is case of HTTP and is not able to provide that level of security as much as MQTT.
6) If you know that you'll always be using a cellular mobile connection, then you should be good with MQTT.	6) There is some set of edge use cases of public Wifi where HTTP will be more robust.
7) MQTT uses less power to maintain an open connection, to receive messages and to send them.	7) In case of HTTP power usage is more.

After usage of both the protocols in our project we came to a conclusion that MQTT is a better option to choose

7.2 Mosquitto vs. HIVE in Brokers

HIVE MQTT Broker	Mosquitto MQTT Broker
1) HIVE is not Open Source and it has commercials involved	1) Mosquitto is Open Source and extremely light weight broker
2) Unreliability is an issue we have faced with HIVE	2) Mosquitto has so far proved useful for its stability and community support
3) HIVE and it ' s components are a bit heavier when compared to Mosquitto	3) The fact that Mosquitto is very light makes it efficient for transporting huge amounts of sensor data.

Even though both HIVE and Mosquitto find their uses, we feel that Mosquitto has some advantages and mostly due to the fact that it is lighter and open source

8. CONCLUSIONS

Using the MQTT protocol, we see that it is indeed possible to transmit sensor data from one device to the other which can also be used to control electronic devices. The data obtained from the sensor can also be considered accurate enough as the error does not exceed more than +0.5 or below -0.5. Thus, we can conclude that the MQTT protocol is an effective protocol for transmission of sensor data and systems can be built based on this protocol. We also see that the MQTT protocol has distinct advantages over the traditional HTTP protocol especially in the mobile environment and considering that mobile platforms are on the rise, MQTT protocol will be more popular in the years to come.

ACKNOWLEDGEMENT

I would like to thank Professor Nilay Kumar Nag for the guidance and the Department of MCA of MCKV Institute of Engineering for providing us with the support and infrastructure required for such a task. I would also like to thank International Conference on Advances in Science and Technology 2017 (ICAST 2017) for providing us with the platform for sharing our knowledge and findings.

REFERENCES

[1] Vasileios K., Periklis C., Francisco V.G., Jesus A.Z. (2015). A survey on application layer protocols for the internet of things, transaction on IoT and cloud computing.

[2] Kaukalias T., Chatzimisios P. (2014). Internet of Things (IoT) C Enabling technologies, applications

and open issues, Encyclopedia of Information Science and Technology (3rd Ed.), IGI Global Press.

[3] Lopez Research LLC (2013). An Introduction to the Internet of Things (IoT).

[4] Valerie L., Weng T.L., Leonardo O., Sweta R., Nagesh S., Rong X., Gerald K., Neeraj K., Stefan F., Martin K., Dave L.. IBM redbooks building smarter planet solutions with MQTT and IBM WebSphere MQ telemetry.

[5] HiveMQ. (2015). MQTT Essentials: Part 1 – Introducing MQTT, the HiveMQ Team.

[6] IBM Knowledge Center (2017). Introduction to MQTT.

[7] HiveMQ. (2015). MQTT Essentials Part 3: Client, Broker and Connection Establishment, the HiveMQ Team.

[8] HiveMQ. (2015). MQTT Essentials Part 4: Subscribe & UnSubscribe, the HiveMQ Team.