How to MQtt Series

## Introduction

IoT (Internet of Things) devices have become an indistinguishable part of our daily lives edging from smart homes, smart bulbs to smart appliances; creators and developers are incorporating this technology to create a network of connected devices that makes our day-to-day life a little more exciting. All this has been made possible because of the ease of communication. There are many possible ways to communicate among devices, a single protocol that is commonly used is **Message Queuing Telemetry Transport** (MQTT).

MQtt was originally developed to link sensors on oil pipelines with satellites. It is a messaging application that supports asynchronous communication between parties. An asynchronous messaging protocol decouples the message sender and receiver both space and time, and hence is scalable in an unreliable network environment. It uses the publish and subscribe model.

## What is MQtt?

It is a light-weight and flexible network protocol that strikes the right balance for IoT developers.

* The lightweight protocols allow it to be implemented on both heavily constrained device hardware as well as high latency/limited bandwidth networks.
* Its flexibility makes it possible to support device application scenarios for IoT devices and services.

## Why MQtt?

It is lightweight and flexible, a key feature being its publish and subscribe model which decouples the publisher and consumer data, making the model asynchronous.

## Objectives

* To learn how to connect an IoT device (in our case, it is the NodeMCU module) to an MQTT broker
* To transfer data among the MQTT broker and NodeMCU.

## Literature Review

MQtt is a messaging standard protocol for IoT which is designed as an extremely light-weight publish/subscribe messaging transport that is ideal for connecting remote devices with unreliable networks, high latency and minimal bandwidth. It is applied in the automotive, manufacturing, telecommunication, oil and gas industries.

## How does MQtt work?

You need a client and a broker.

MQtt client can be any device, from a microcontroller to a fully-fledged server, which runs the MQtt library and is connected to the MQtt broker over any network.

MQTT broker is responsible for receiving all messages, filtering, decision making and sending messages to subscribed clients.

MQTT is based on TCP/IP, hence both client and broker are expected to have TCP/IP stack.

For this project, we aimed at connecting our NodeMCU module to a MQTT broker and send messages to the ESP8266. We wanted to control the lighting of an LED. To do this there are different methods in MQTT to indicate the desired actions to be performed on identified resources.

Resources can be files or the outputs of an executable, found on a server.

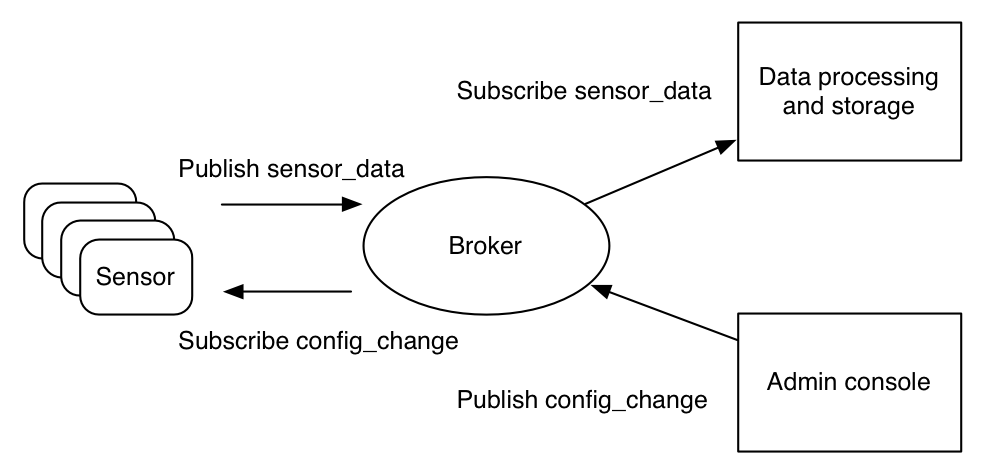
These methods include:

* Connect - waits for a connection to be established with the server.
* Disconnect - waits for the MQTT client to finish any work, which needs to to be done and for the TCP/IP session to disconnect.
* Subscribe -Requests the server to let the client subscribe to one or more topics.
* Unsubscribe -Requests the server to let the client unsubscribe from one or more topics.
* Publish - Returns immediately to the application thread after passing request to the MQTT client.

In our project we used the HIVEMQ broker; it is an open-source broker, which is light-weight and is suitable for use on IoT devices for communication.

To establish communication with the broker, we have to set up the hivemq broker. We used an Android application to publish and subscribe to the information with the broker. We installed an MQTT client from the app playstore which we used to set up the software side of the project; by subscribing a topic, then publishing the topic and finally writing a message to accompany the published topic. In our case, our message was either LEDON (to light the LED) or LEDOFF (to switch off the LED).

## Flow Diagram of MQTT Configuration



## Equipment - Bill of Materials

NodeMCU (ESP8266 module)

LED

220-ohm resistor

Connecting wires

Breadboard

Programming cable

## Methodology

Schematics featured in design file.

## Flowchart

CONNECT TO MQTT BROKER

SUBSCRIBE A TOPIC

PUBLISH A TOPIC

SEND TOPIC TO NODEMCU

RECEIVE TOPIC AND MESSAGE

IS MESSAGE RECEIVED?

IS MESSAGE LEDON?

LIGHT LED

IS MESSAGE LEDOFF?

LED OFF

Y

Y

N

Y

## Conclusion

From the research and review shown above, we were able to meet our objectives on the project. We used MQTT publish/subscribe model to control the lighting of our LEDs.

### Challenges

The only challenge we experienced during this project was to find a compatible MQTT broker to our NodeMCU module.

There was also the challenge of accessing the available wi-fi.

When considering scalability, there is a challenge when the system is needed to use a lager bandwidth; MQTT being a lightweight messaging protocol, can only be used in systems using low bandwidth.

When using MQTT protocol in a system when scaling up, developers need to ensure they configure secure environments for their system to avoid security breach in the system.

### Scalability

We can use IoT and MQTT to automate our homes, improve security by creating a system that allows the owner to monitor and authorize access to their homes remotely through their smartphones, iPad and PCs.

The developers might also decide to use broker clusters in the making of a distributed system and a more secure one at that. One of the most unique and sophisticated features of HiveMQ is its cluster capability that is ultra-reliable and is typically used in cloud environments for systems that must not fail and need linear scalability over time.