Kamakshi jain
Sayan Brahma

Home Automation IOT Final Project

ENPM 809F
6th Dec 2019

Introduction

Currently real world objects like light bulbs are controlled by switches and there are no feedbacks for the appliances about their current states. Thus someone else in another room will not know about the state of the light bulb. However if we can introduce IOT implemented light bulbs or other appliances then these can be controlled from anywhere and the state of the appliances can also be known by everyone having the access.

Our project is all about creating a simple controllable MQTT sensor simulator with python. The aim of this project is too create a very simple 2 state sensor, that can be controlled externally using MQTT protocol.

The sensor could be used to simulate real world objects like light, doors, biometric locking systems, Air conditioners and other electrical appliances like kitchen exhausts that can have more than 2 states but for our project we have considered the ones with 2 states like light with on/off states and door with open or closed states.

MQTT

MQTT (Message queuing telemetry transport) is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium. For example, it has been used in sensors communicating to a broker via satellite link, over occasional dial-up connections with healthcare providers, and in a range of home automation and small device scenarios. It is also ideal for mobile applications because of its small size, low power usage, minimised data packets, and efficient distribution of information to one or many receivers.

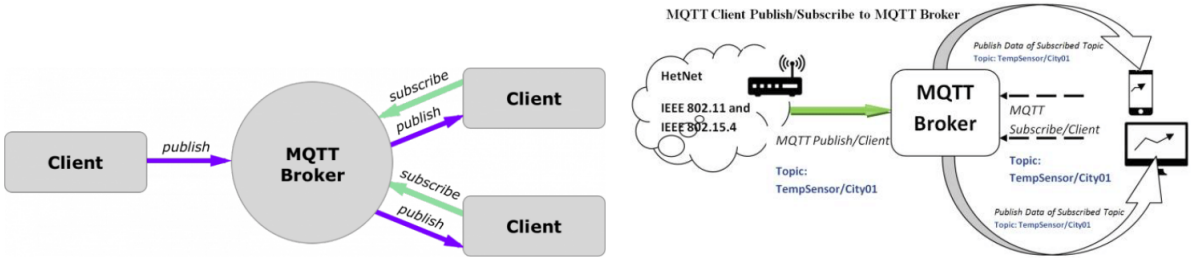

Figure 1: Overall Communication Sequence

MQTT is ideal for our project as in this scenario it allows the sensor to be controlled by multiple clients/locations and each one will be aware of the current state of the sensor.

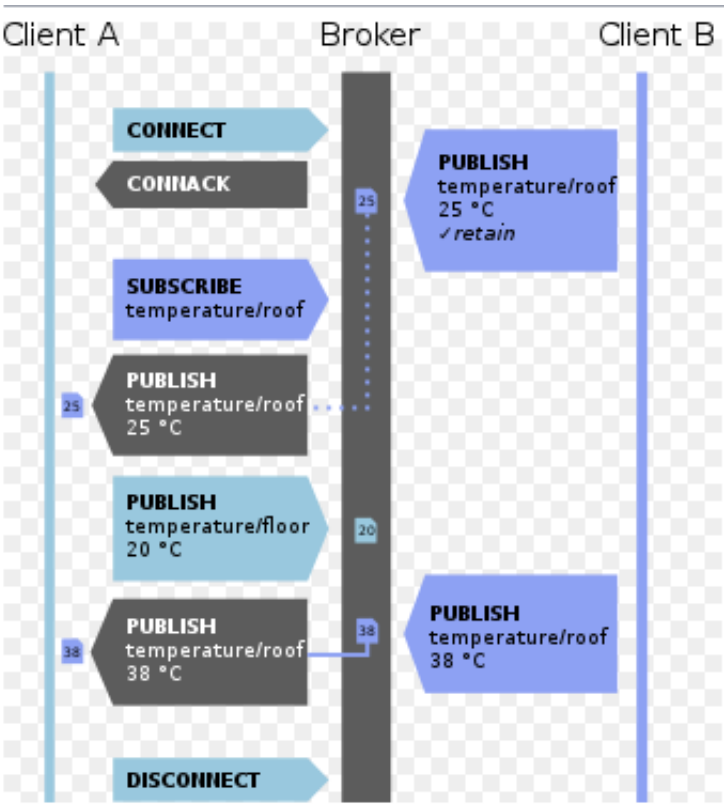

Figure 2: Image after Pre-processing

Technical Project brief

Tools Used:

- coding language - Python 3.6
- Operating system - Ubuntu 16.04
- Broker tool - mosqquitto pub tool
- Network protocol - MQTT
- Simulation - Command terminal in ubuntu

The way it works:

The sensor subscribes to a control topic and sits on a loop publishing its current state and waiting for a command to change its state. If the given command asks for a valid state change then the state is changed, else it is shown that the client tried to change the state but since the state is invalid so the state of the sensor remain unchanged.

The sensors considered with their possible states are:

- Light - on/off
- Door - closed/open

The default states are off and closed for the light and door respectively.
The commands are not case sensitive. Everytime the sensor changes its state it publishes a message stating its current state.

Modes for status update:

- Chatty Mode - In this mode the sensor publishes the state at a regular interval
- Non-Chatty Mode In this mode the sensor publishes only after the state is changed. By default the sensor publishes the status at a keep alive interval of 5 min.

Topics used:

- Topic prefix/sensor-name - used to publish sensor state
- Topic prefix/sensor-name/control/ - used to change state
- Topic prefix/connected/sensor-name/ - Used to indicated connection status 0 = disconnected and 1 = connected

And the way it works is that when is the sensor is connected it publishes 1 to the topic and before or when the sensor is disconnected it publishes 0. We can scan the topic connected/sensor-name to know the current connection status of the sensor.
For our project we used **mosqquittopub tool client** to publish messages. Although MQTT lens client is ideal for this purpose and also MQTT dashboard client (an Android app) as we can keep a track of messages going to the sensor as well coming out from a sensor.

Commands for executing

- To run the python file - **python3 simple-sensor.py**(Linux)
- Host name specification - **python3 simple-sensor.py -h ¡host name¿ - mqtt.eclipse.org**
- Use non chatty mode(i.e., to publish status after it has been changed) - it is by default - **python3 simple-sensor.py -h mqtt.eclipse.org -v**
- Use chatty mode(i.e., to publish status after every specified seconds) - **python3 simple-sensor.py -h mqtt.eclipse.org -v -i 20**

- To set the client name to testclient - **python3 simple-sensor.py -h mqtt.eclipse.org -n testclient** - By default it starts the light sensor
- To use the door sensor open/close - **python3 simple-sensor.py -h mqtt.eclipse.org -n testclient -s**
- Control the sensors we use - **mosquittopub -h mqtt.eclipse.org -t sensors/testclient/control -m** or use open for door sensor

Test/Demo Cases

Light sensor + 1 client

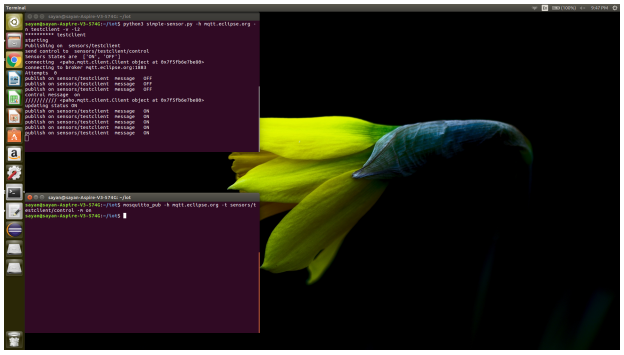

Figure 3: screenshot for 1 sensor with 1 client

Light sensor + 2 client

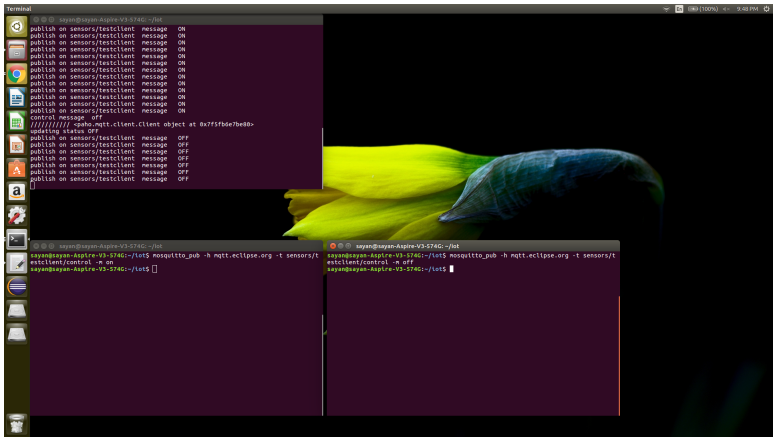

Figure 4: screenshot for 1 sensor with 2 clients

Light sensor + 2 clients + invalid status update

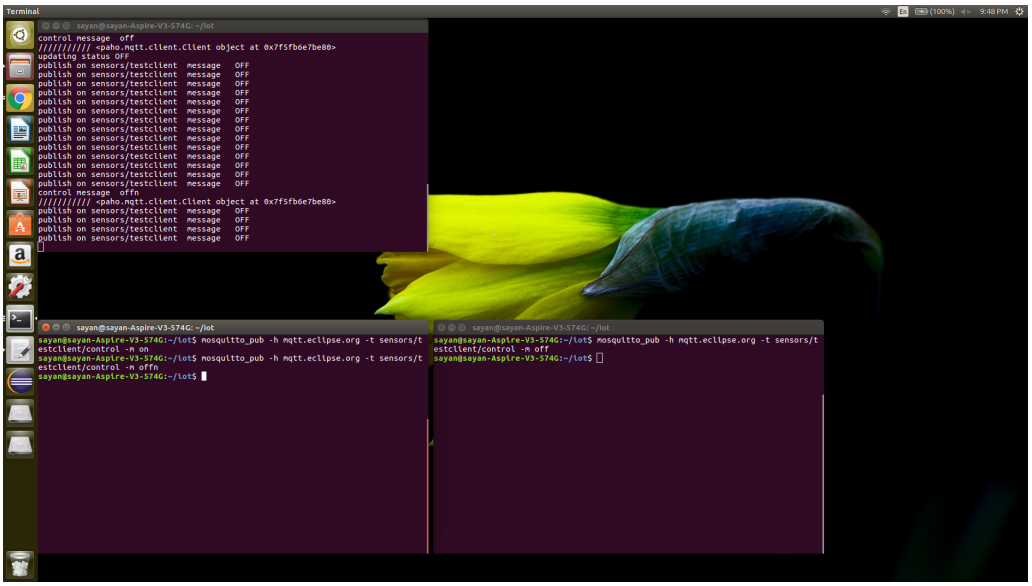

Figure 5: screenshot for 1 sensor with 1 client and an invalid status request

Light + door sensor + 1 clients each

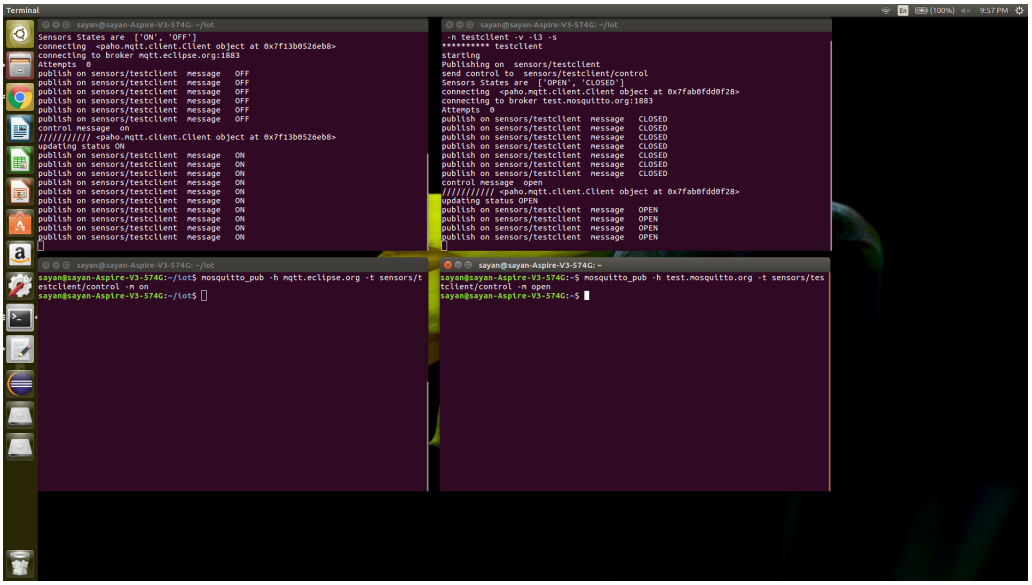

Figure 6: screenshot for 2 sensors with 1 client each

Conclusion and Future Work

- Many more 2 state sensors can be incorporated.
- The code can be expanded to be implemented in more the 2 state sensors, which will help us to attain different applications like - We can pull up the client voting system and update the state of the appliances to an intermediate level if involved with direct controlled actuators.
- Different biometric levels of security can be obtained for industrial as well as domestic purpose.

Deliverables:

Github repository link with all the code and launch instructions in readme file. Project Report - softcopy. Readme for the final submitted folder with all the detailed instructions and some screenshots from the demo.
Github Link : IOT Project