Real-time Sensor Data Visualization on ThingsBoard Web Page using MQTT Protocol

A MINI PROJECT REPORT

18CSE345T- IOT ARCHITECTURE AND PROTOCOLS

Submitted by

Pragya Nidhi (RA2111032010001)

Arnav Aggarwal (RA2111032010002)

Shrey Nagori (RA2111032010008)

Alok Agnihotri (RA2111032010010)

Lakshya Kishnani (RA2111032010015)

Shaun Joshua (RA2111032010018)

Priyanshi Sharma (RA2111032010057)

Kanak Shilledar (RA2111032010060)

SCHOOL OF COMPUTING

BACHELOR OF TECHNOLOGY

COMPUTER SCIENCE AND ENGINEERING

(WITH SPECIALIZATION IN INTERNET OF THINGS)

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

(KATTANKULATHUR CAMPUS)

CHENNAI- 603203

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this mini project report "**Real-time Sensor Data Visualization on ThingsBoard Web Page using MQTT Protocol**" is the bonafide work of "**Pragya Nidhi (RA2111032010001), Arnav Aggarwal (RA2111032010002), Shrey Nagori (RA2111032010008), Alok Agnihotri (RA2111032010010), Lakshya Kishnani (RA2111032010015), Shaun Joshua (RA2111032010018), Priyanshi Sharma (RA2111032010057), Kanak Shilledar (RA2111032010060)**" who carried out the project work for **18CSE345T – IOT ARCHITECTURE AND PROTOCOLS** under my supervision at **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY,** Kattankulathur during the academic year 2023 – 2024.

SIGNATURE

Faculty In-Charge
Dr. Swathy R
Assistant Professor
Department of Networking and Communications
SRM Institute of Science and Technology

SIGNATURE

HEAD OF THE DEPARTMENT
Dr. Annapurani Panaiyappan. K
Professor and Head,
Department of Networking and Communications
SRM Institute of Science and Technology

TABLE OF CONTENTS

ABSTRACT

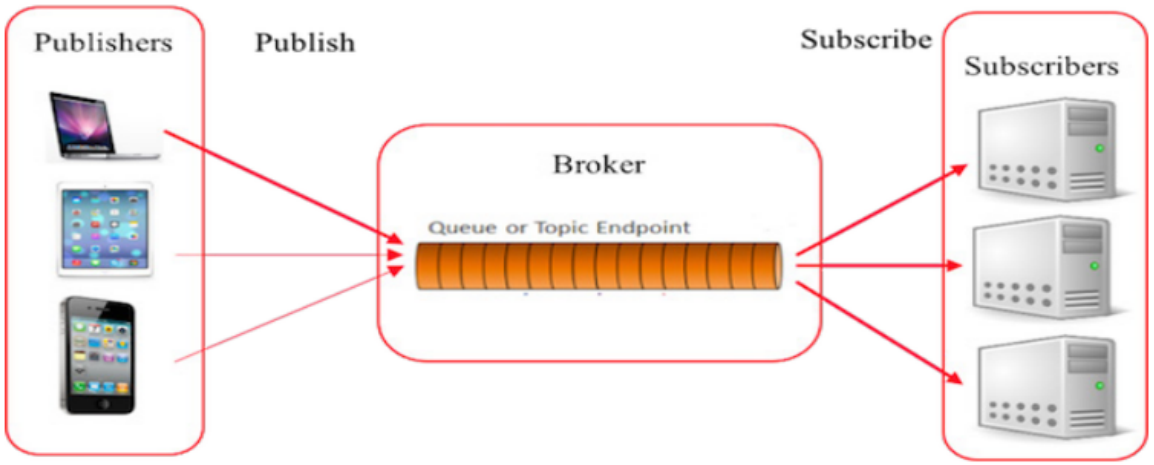
In this project, we propose a Python-based solution to push and display real-time sensor data on a ThingsBoard web page using the MQTT (Message Queuing Telemetry Transport) protocol. IoT (Internet of Things) devices often generate a wealth of sensor data, and it is crucial to visualize this data efficiently. We have designed a system that allows you to connect sensors to a Python program, which subsequently publishes the data to ThingsBoard, a popular open-source IoT platform, using MQTT.

The project comprises several key components: data collection from sensors, data formatting, establishing an MQTT connection, and the integration with ThingsBoard. The Python program will act as a bridge between the physical sensors and the ThingsBoard platform, enabling real-time monitoring and data visualization through a user-friendly web interface.

This project aims to provide a practical and accessible solution for developers and IoT enthusiasts looking to build a robust and flexible sensor data visualization system. The implementation will showcase the capabilities of MQTT and Python in the context of IoT applications while demonstrating the seamless integration of data from physical sensors to a web-based dashboard for monitoring and analysis.

INTRODUCTION

Message Queue Telemetry Transport (MQTT) was introduced by IBM in 1999 and standardized by OASIS in 2013. It is designed to provide embedded connectivity between applications and middleware's on one side and networks and communications on the other side. It follows a publish/subscribe architecture, as shown in Figure, where the system consists of three main components: publishers, subscribers, and a broker. From IoT point of view, publishers are basically the lightweight sensors that connect to the broker to send their data and go back to sleep whenever possible. Subscribers are applications that are interested in a certain topic, or sensory data, so they connect to brokers to be informed whenever new data are received. The brokers classify sensory data in topics and send them to subscribers interested in the topics

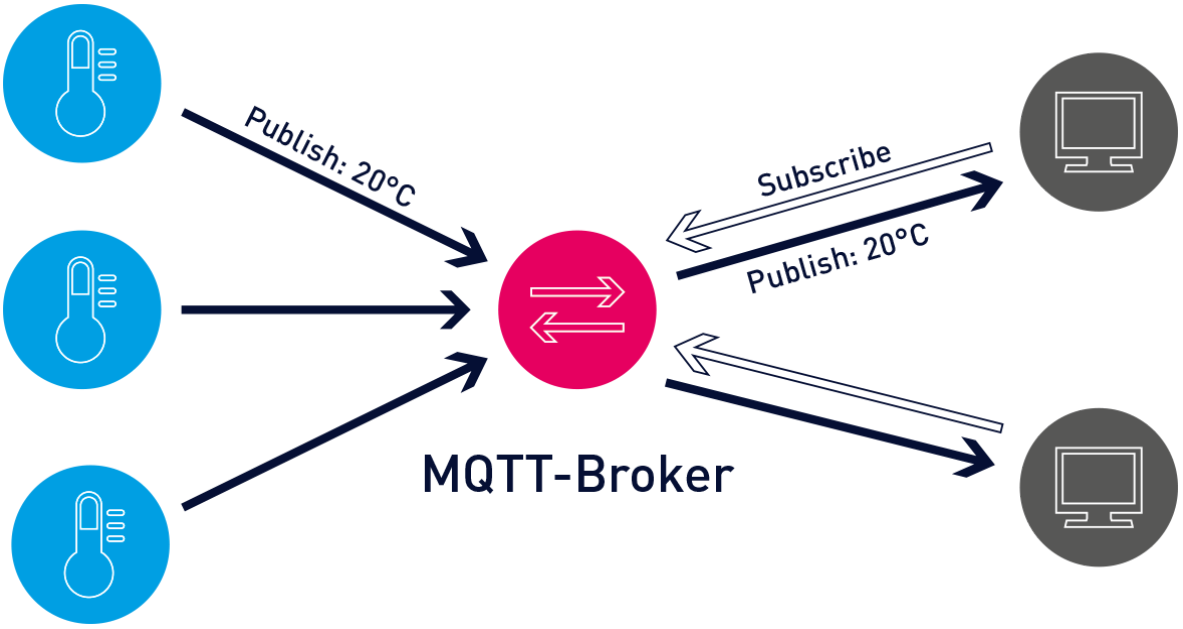


ARCHITECTURE

1.1 MQTT Architecture

MQTT runs on top of TCP/IP using a PUSH/SUBSCRIBE topology. In MQTT architecture, there are two types of systems: clients and brokers. A broker is the server that the clients communicate with. The broker receives communications from clients and sends those communications on to other clients. Clients do not communicate directly with each other, but rather connect to the broker. Each client may be either a publisher, a subscriber, or both.

MQTT is an event-driven protocol. There is no periodic or ongoing data transmission. This keeps transmission to a minimum. A client only publishes when there is information to be sent, and a broker only sends out information to subscribers when new data arrives.



1.2 Message Architecture

Another way MQTT minimizes its transmissions is with a tightly defined, small message construction. Each message has a fixed header of just 2 bytes. An optional header may be used but increases the size of the message. The message payload is limited to just 256 MB. Three different Quality of Service (QoS) levels allow network designers to choose between minimizing data transmission and maximizing reliability.

- QoS 0 – Offers the minimum amount of data transmission. With this level, each message is delivered to a subscriber once with no confirmation. There is no way to know if subscribers received the message. This method is sometimes referred to as "fire and forget" or as "at most once delivery." Because this level assumes that delivery is complete, messages are not stored for delivery to disconnected clients that later reconnect.
- QoS 1 – The broker attempts to deliver the message and then waits for a confirmation response from the subscriber. If a confirmation is not received within a specified time frame, the message is sent again. Using this method, the subscriber may receive the

message more than once if the broker does not receive the subscriber's acknowledgment in time. This is sometimes referred to as "at least once delivery".

- QoS 2 – The client and broker use a four-step handshake to ensure that the message is received, and that it is received only once. This is sometimes referred to as "exactly once delivery".

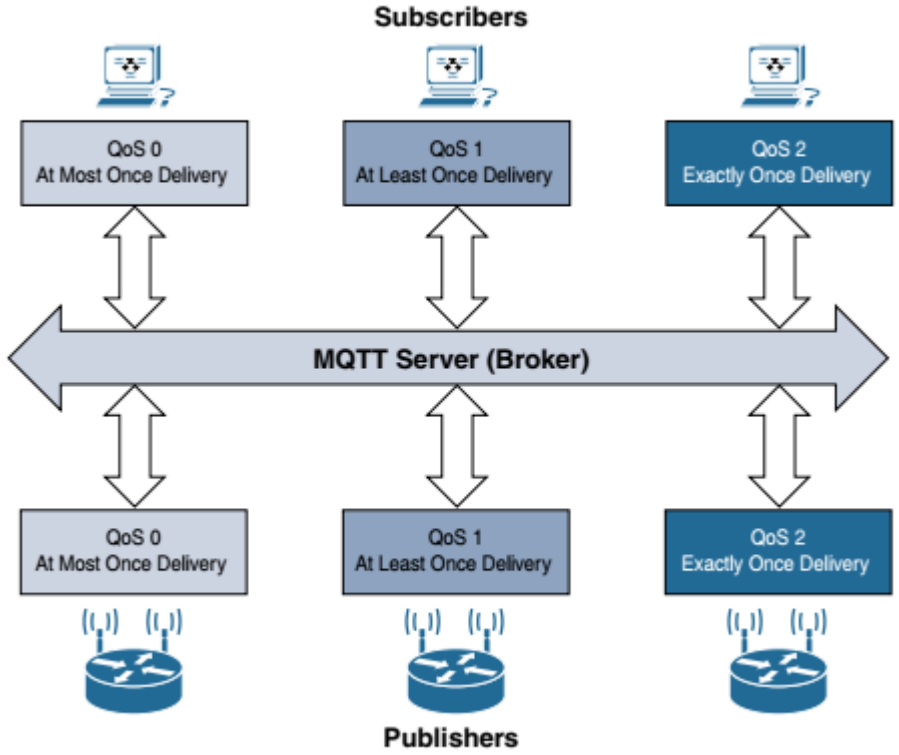


Figure 6-13 *MQTT QoS Flows*

For situations where communications are reliable but limited, QoS 0 may be the best option. For situations where communications are unreliable, but where the connections are not as resource limited, then QoS 2 would be the best option. QoS 1 provides a sort of best-of-both-worlds solution but requires that the application receiving the data knows how to handle duplicates.

For both QoS 1 and QoS 2, messages are saved or queued for clients that are offline and that have an established persistent session. These messages are resent (according to the appropriate QoS level) once the client is back online.

ALGORITHM

1.Import paho.mqtt.client, time and json.

2.Create a host server on thingsboard.io and get the access token.

3.Create a JSON object in the function and return it as a dictionary.

4.Run the web application using the run method.

5.MQTT Protocol client program to push and display temperature, humidity, wind speed data and rain sensor data (rainy or not) in a webpage of thingsboard.io using json format using paho-mqtt library.

PROGRAM & OUTPUT

4.1 Program

```
import paho.mqtt.client as paho #MQTT library
import os
import json
import time
from datetime import datetime

ACCESS_TOKEN='9ggCmmf4iNTcK2AsAJqD' #Token of your device
broker="demo.thingsboard.io" #host name
port=1883 #data listening port

def on_publish(client,userdata,result): #create function callback
     print("data published to thingsboard \n")
     pass
client1= paho.Client("control1") #create client object
client1.on_publish = on_publish #assign function to callback
client1.username_pw_set(ACCESS_TOKEN) #access token from thingsboard
device
client1.connect(broker,port,keepalive=60) #establish connection

while True:
    payload="{"
    payload+="\"Humidity\":60,";
    payload+="\"Temperature\":25";
    payload+="}"
    ret= client1.publish("v1/devices/me/telemetry",payload)
#topic-v1/devices/me/telemetry
    print("Please check LATEST TELEMETRY field of your device")
    print(payload);
    time.sleep(5)
```

4.2 Output

CONCLUSION

The **Real-time Sensor Data Visualization on ThingsBoard Web Page Using MQTT and Python** project has successfully demonstrated the capability to efficiently collect, process, and display sensor data on a web interface in real-time. Through the integration of Python, MQTT, and the ThingsBoard platform, this project has provided a practical solution for the Internet of Things (IoT) community.In summary, this project not only showcases the power of Python and MQTT in the realm of IoT but also underscores the significance of real-time data visualization. It provides a foundation for building more complex IoT applications and highlights the potential of sensor data in various domains, including smart homes, industrial automation, and environmental monitoring. As IoT continues to grow, this project serves as a valuable reference for those looking to harness the potential of sensor data in their own applications.

REFERENCES

1. https://thingsboard.io/docs/
2. https://mqtt.org/
3. https://pypi.org/project/paho-mqtt/
4. https://www.hindawi.com/journals/jece/2017/9324035/
5. https://thingsboard.io/docs/samples/raspberry/temperature/