

DOP: / / 2023

DOS: / / 2023

Experiment No: 10

Aim: - To study and demonstrate working of CoAP protocol in Contiki OS (simulator).

Hardware/Software: Raspberry Pi

Theory:

MQTT is simple, lightweight messaging protocol used to establish communication between multiple devices. It is TCP-based protocol relying on the publish-subscribe model. This communication protocol is suitable for transmitting data between resource-constrained devices having low bandwidth and low power requirements. Hence this messaging protocol is widely used for communication in IoT Framework.

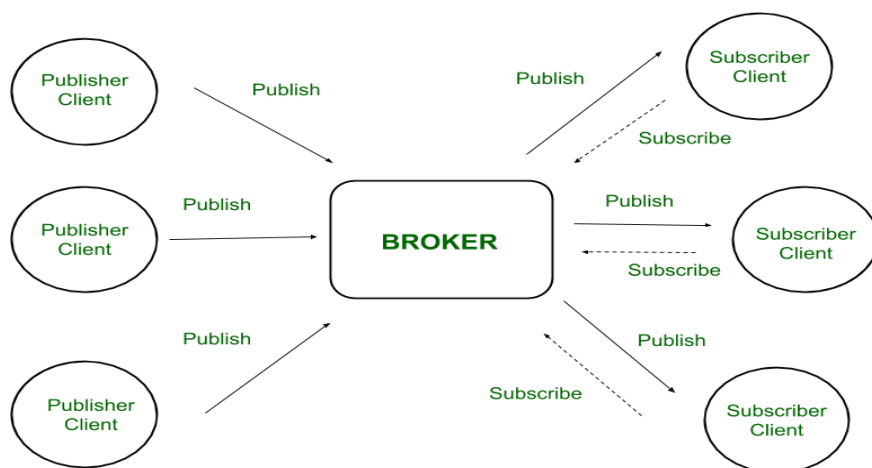
- **Publish-Subscribe Model:**

This model involves multiple clients interacting with each other, without having any direct connection established between them. All clients communicate with other clients only via third party known as Broker.

- **MQTT Client and Broker:**

Clients publish messages on different topics to broker. The broker is the central server that receives these messages and filters them based on the topics. It then sends these messages to respective clients that have subscribed to those different topics.

Hence client that has subscribed to a specific topic receives all messages published on that topic.



Here the broker is central hub that receives messages, filters them, and distributes them to appropriate clients, such that both message publishers, as well as subscribers, are clients.

Advantages:

1. Easy Scalability –

This model is not restricted to one-to-one communication between clients. Although the publisher client sends a single message on specific topic, broker sends multiple messages To all different clients subscribed to that topic. Similarly, messages sent by multiple such publisher clients on multiple different topics will be sent to all multiple clients subscribed to those topics. Hence one-to-many, many-to-one, as well as many -to-many communication is possible using this model. Also, clients can publish data and at the same time receive data due to this two-way communication protocol. Hence MQTT is bi-directional



protocol. The default unencrypted MQTT port used for data transmission is 1883. The encrypted port for secure transmission is 8883.

2. Eliminates insecure connections –

In a complex system where, multiple devices relate to each other, each device not only has to manage its connections with other devices but also must ensure that these connections are secure. But in the publish-subscribe model, the broker becomes central server managing all security aspects. It is responsible for the authentication and authorization of all connected clients.

3. Lightweight Communication –

Data transmission is quick, efficient, and lightweight because MQTT messages have small code footprint. These control messages have a fixed header of size 2 bytes and payload message up to size 256 megabytes

Topics:

In MQTT, topic is UTF-8 string that the broker uses to filter messages for each individual connected client. Each topic consists of one or more different topic levels. Each topic level is separated by forward slash also called topic level separator. Both topics and levels are case-sensitive.

MQTT HIVE MQ

HiveMQ is an MQTT broker and a client-based messaging platform designed for the fast, efficient, and reliable movement of data to and from connected IoT devices. It uses the MQTT protocol for instant, bi-directional push of data between your device and your enterprise systems.

Procedure:

Program:

```
MQTT Client program
# Install MQTT Broker Server
# sudo apt-get install mosquitto
# Command line clients in case for debugging
# sudo apt-get install mosquitto-clients -y #
Install the MQTT Publisher
# sudo pip3 install paho-mqtt
import os
import sys
import time
import board
import adafruit_dht
import paho.mqtt.client as mqtt
import json
# Initial the dht device, with data pin connected
to:dhtDevice=adafruit_dht.DHT11(board.D19,use_pulseio=False)sensor_data= {'&#39;temperature
&#39;;: 0, '&#39;humidity&#39;;: 0}Server = '&#39;127.0.0.1&#39;;
client=mqtt.Client()
client.connect(Server,1883, 60)
client.loop_start()
```



```
if _____name_____ == '&#39;_____main____&#39;::
while True:
try:
# Print the values to the serial port temperature = dhtDevice.temperature humidity =
dhtDevice.humidity

Print("&quot;Temp: {:.1f} C Humidity: {}% &quot;
.format( temperature, humidity)) time.sleep(2.0)
sensor_data[&#39;temperature&#39;] = temperature sensor_data[&#39;humidity&#39;] = humidity
# Sending humidity and temperature data to ThingsBoard client.publish('&#39;test_channel&#39;,
.dumps(sensor_data), 1) time.sleep(5)
except RuntimeError as error:

# Errors happen fairly often, DHT&#39;s are hard to read, just keepgoing
print(error.args[0
])time.sleep(2.0)
continue
except
KeyboardInterrupt:
client.loop_stop()
client.disconnect()
print (&#39;Exiting Program&#39;)
exit()

HiveMQ MQTT Server
Program# Install the MQTT
Publisher
# sudo pip3 install paho-mqtt
# open the MQTT browser client in web browser
# http://www.hivemq.com/demos/websocket-client/

Impor
t os
impor
t sys
impor
t time
impor
t
board
import adafruit_dht
import paho.mqtt.client as
mqttimport json
# Initial the dht device, with data pin connected to:
dhtDevice = adafruit_dht.DHT11(board.D19, use_pulseio=False) sensor_data
= {&#39;temperature&#39;: 0, &#39;humidity&#39;: 0}MQTTServer =
&#39;broker.mqttdashboard.com&#39;
client = mqtt.Client()
```



Jawahar Education Society's Annasaheb Chudaman Patil College of Engineering, Kharghar, Navi Mumbai

```
client.connect(MQTTServer, 1883,
8000)client.loop_start()

if __name__ == '__main__':
    while True:

        try:
            # Print the values to the serial port
            temperature=dhtDevice.temperature
            humidity = dhtDevice.humidity
            print('Temp: {:.1f} C Humidity: {}%'.format( temperature,humidity))
            time.sleep(2.0) sensor_data['temperature'] =
            temperaturesensor_data['humidity'] = humidity

            # Sending humidity and temperature data to HIVEMQ
            client.publish('RPI4_MQTT', json.dumps(sensor_data), 1)time.sleep(5)

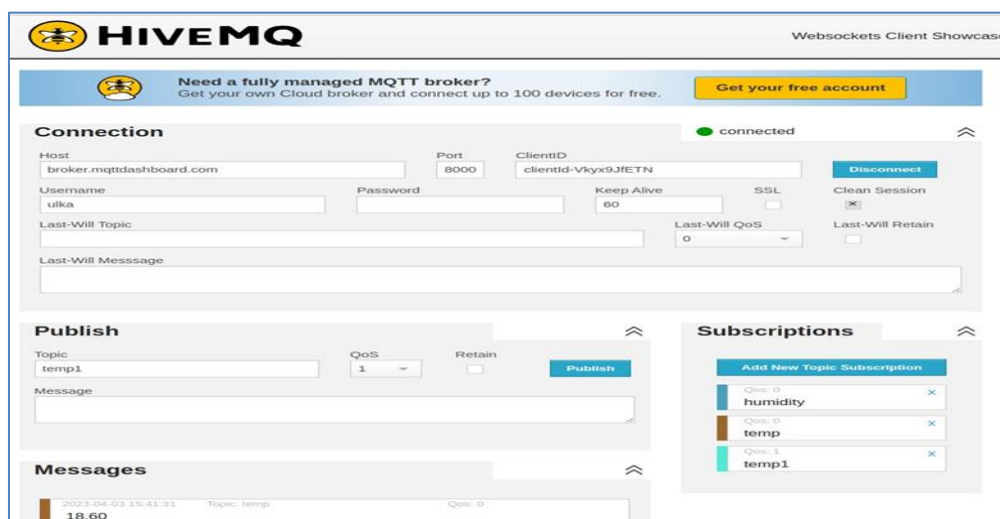
        except RuntimeError as error:

            # Errors happen fairly often, DHT's are hard to read, just keepgoing

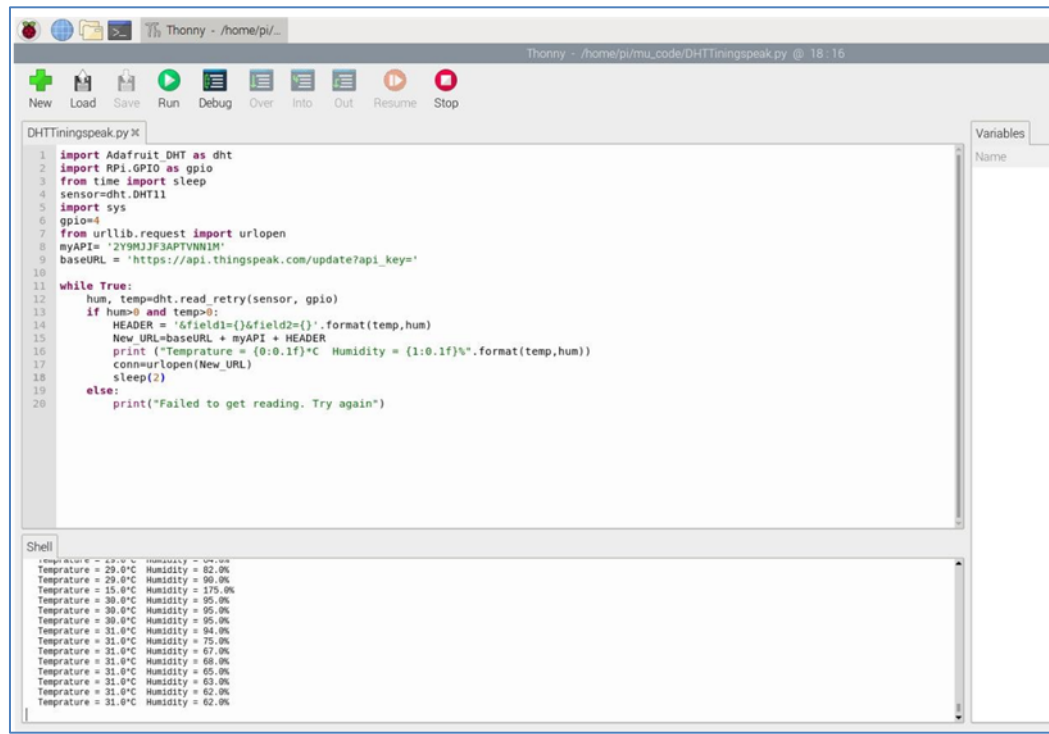
            print(error.args[0])time.sleep(2.0) continue

        except KeyboardInterrupt:
            client.loop_stop()
            client.disconnect()
            print ('Exiting Program')
            exit()
```

Result:



HiveMQ WebSockets Client Showcase



```

DHTTingspeak.py X
1 import Adafruit_DHT as dht
2 import RPi.GPIO as gpio
3 from time import sleep
4 sensor=dht.DHT11
5 import sys
6 gpio=4
7 from urllib.request import urlopen
8 myAPI= '2Y9KJ3F3APTVMN1M'
9 baseURL = 'https://api.thingspeak.com/update?api_key='
10
11 while True:
12     hum, temp=dht.read_retry(sensor, gpio)
13     if hum>0 and temp>0:
14         HEADER = '&field1={}&field2={}'.format(temp,hum)
15         New_URL=baseURL + myAPI + HEADER
16         print ("Temperature = {0:0.1f}*C Humidity = {1:0.1f}%".format(temp,hum))
17         conn=urlopen(New_URL)
18         sleep(2)
19     else:
20         print("Failed to get reading. Try again")

```

```

Shell
temp=29.0°C Humidity = 62.0%
Temperature = 29.0°C Humidity = 62.0%
Temperature = 29.0°C Humidity = 90.0%
Temperature = 15.0°C Humidity = 175.0%
Temperature = 30.0°C Humidity = 95.0%
Temperature = 30.0°C Humidity = 95.0%
Temperature = 30.0°C Humidity = 95.0%
Temperature = 31.0°C Humidity = 94.0%
Temperature = 31.0°C Humidity = 75.0%
Temperature = 31.0°C Humidity = 67.0%
Temperature = 31.0°C Humidity = 68.0%
Temperature = 31.0°C Humidity = 65.0%
Temperature = 31.0°C Humidity = 63.0%
Temperature = 31.0°C Humidity = 62.0%
Temperature = 31.0°C Humidity = 62.0%

```

Temperature and humidity showcase

Conclusion:

Thus, we were able to perform MQTT which is simple, lightweight messaging protocol used to establish communication between multiple devices. It is TCP-based protocol relying on the publish-subscribe model. We were able to setup the Coap Protocol in Contiki Simulator (OS) which uses MQTT Client and Broker.