CPE 4040: Data Collection and Analysis, Spring 2023

# Laboratory Report #3

## Lab 3: Raspberry Pi with Cloud MQTT Broker

Team Members: Neal Jarzen & Clarence Barron

Electrical and Computer Engineering

Kennesaw State University

Faculty: Dr. Jeffrey L Yiin

Date of Lab Experiment: January 23, 2023

# I. Objective

The main purpose of this lab is to learn how to use the MQTT cloud broker and implement basic subscriber code in python. Also, we also learned how to remotely control the Pi digital output with the cloud platform.

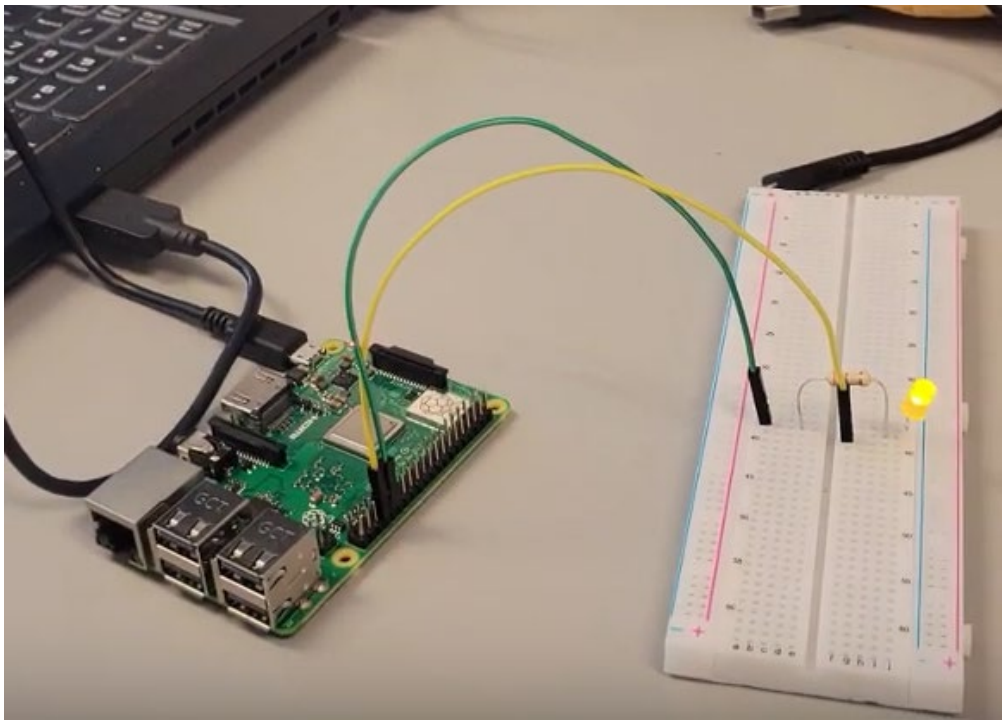
# II. Material List

1. Raspberry Pi 3 or 4
2. Power supply adapter
3. Micro SD card (16+GB)
4. Ethernet cable
5. (optional) USB Keyboard, mouse and HDMI monitor or TV
6. Install Putty, Advanced IP Scanner and WinSC 7.
LED and Resistors

# III. Lab Procedures and Results

1) After booting up and updating the packages to your Raspberry Pi, we need to get the packages called Paho MQTT. To install this package input "**pip3 install paho-mqtt**".

2) Next is the hardware set up. For this lab, we are going to utilize GPIO 12, which will be pin 32 on the Raspberry Pi and GND will be pin 34, which will be the pin below GPIO 12. Like as shown from the picture below:



3) After the configuration has been assembled, go back to the interface and create a python script called "*digitalout.py*" with the following code within the file:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(12, GPIO.OUT)

GPIO.output(12, GPIO.HIGH)

time.sleep(1)

GPIO.output(12, GPIO.LOW)
```

Then execute the chip. The use of the "***GPIO.setmode(GPIO.BCM)***" will activate the broadcom-chip specific pins on the pi. The light should turn on and off, and it will indicate that hardware is set up correctly as well. This is demonstrated by the video below:

https://www.youtube.com/watch?v=FMH6KmKJ-rI&feature=youtu.be

4) Once working, go to www.beebottle.com and create an account. Once, you have made the account go to the "My Channels" page and create a channel. Click "Create

New", on the top right of the page, and name the channel. In the created channel, create a resource name called "sensor".

5) Click on the channel and in red text, you will see something called "Channel Key". It should start with "token_xxxx". Go ahead and save this key. The key will be used for the pi and beebottle to communicate with each other when the code is triggered within the pi.
6) Next you will make a new Python program by entering the command "**sudo nano subscriber.py**" after that you will put the following code into the file. Put your own token string, channel name and resource name within the code.

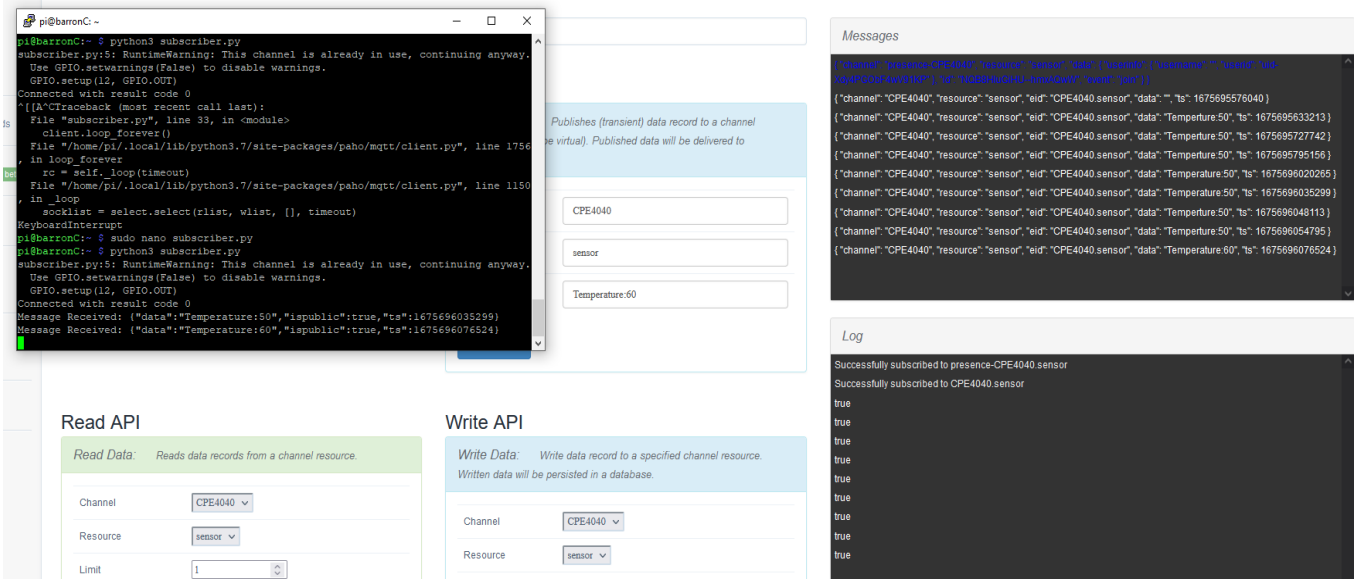
```
Import paho.mqtt.client as mqtt
import RPi.GPIO as GPIO

# This is the GPIO settings
GPIO.setmode(GPIO.BCM)
GPIO.setup(12, GPIO.OUT)

# This is the MQTT Connection Settings
host = "mqtt.beebotte.com"
port = 1883
password = ""
username = "<token_XXXXXXXXXXXX>"
channel = "<channel name>"
resource = "<resource name>"
defStr = "Temperature"

topic = channel + "/" + resource
def on_connect(client, userdata, flags, rc):
        print("Connected with result code "+str(rc))
        client.subscribe(topic,1)
```

```
def on_message(client, userdata, msg):
        if defStr in str(msg.payload.decode()):
        print("Message Received:", str(msg.payload.decode()))
elif "ledON" in str(msg.payload.decode()):
        print("LED ON!")
elif "ledOFF" in str(msg.payload.decode()):
        print("LED OFF!")
client = mqtt.Client()
client.username_pw_set(username)
client.connect(host,port,60)
client.on_connect = on_connect
client.on_message = on_message
client.loop_forever()
```

7) You will now need to run this code in a terminal using **Python3**. If the connection is established, a "connected" message with the result code 0 should show up on the screen. If you do not get this result code, then you will need to re-check the MQTT connection parameters and then try again.

8) In beebottle, go to your account settings and navigate to the "Access Management" page. Inside the page, you will find the secret key. Copy that secret key and go to your channel and paste the security key in the box. The secret key is needed so that the Pi and the cloud can communicate with each other under a secure connection between the two.

9) Next, you need to test your channel in the Beebottle cloud, fill in the channel name and resource name in the subscribe section and click on the "Subscribe" button. After filling in the same information into the publish section with arbitrary data, click on the "publish" button. You should now start observing the messages that start to appear in the message window. The string "Temperture:" will not show up in the message window due to the function definition of string and change it to what is has under the definition. As shown below:




10) After printing five messages, go back into the code and modify it to where when ledON or ledOFF is printed, the LED will turn on or turn off depending on what is published. Then run the code again and, upon publishing ledON or ledOFF, the LED

will turn on and off just like as shown in the video. The following is also the modified code done to the "*subscriber.py*" file.

pi@barronC: ~

```
  GNU nano 3.2

import paho.mqtt.client as mqtt
import RPi.GPIO as GPIO
# This is the GPIO settings
GPIO.setmode(GPIO.BCM)
GPIO.setup(12, GPIO.OUT)
# This is the MQTT Connection Settings
host = "mqtt.beebotte.com"
port = 1883
password = ""
username ="token_RqecIn5uZsULbAzR"
channel = "CPE4040"
resource = "sensor"
defStr = "Temperature"
topic = channel + "/" + resource

def on_connect(client, userdata, flags, rc):
        print("Connected with result code " + str(rc))
        client.subscribe(topic,1)

def on_message(client, userdata, msg):
        if defStr in str(msg.payload.decode()):
                print("Message Received:", str(msg.payload.decode()))
        elif "ledON" in str(msg.payload.decode()):
                print("LED ON!")
                GPIO.output(12, GPIO.HIGH)
        elif "ledOFF" in str(msg.payload.decode()):
                print("LED OFF!")
                GPIO.output(12, GPIO.LOW)

client = mqtt.Client()
client.username_pw_set(username)
client.connect(host,port,60)
client.on_connect = on_connect
client.on_message = on_message
client.loop_forever()
```

https://www.youtube.com/watch?v=RZIjTcWKB6E&ab_channel=PhozyVault

# IV. Conclusion

It was interesting watching the PI and BeeBottle communicate with one another through some code. In fact, it does bring up a bunch of ideas that this can be used with and possibly in conjunction with IFTTT's networking systems to create something great. The lab overall was not hard. Although, one critique we can make, which is merely a small nitpick, but can help over all regarding using code, is that when using the "def" functions and inputting code under it to have a reminder that it must be indented correctly or else the code will not work under the definition.