KENNESAW STATE UNIVERSITY

CPE 4040: Data Collection and Analysis, Spring 2023

Laboratory Report #N

Lab 2: MQTT Pub/Sub

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

We are tasked with learning how to create an MQTT Broker on the Raspberry PI. After going through the lab steps, we should be able to understand basic PUB/SUB messaging when using the Mosquitto MQTT package. Our final objective should be to fully understand how to use MQTT messaging options/topics when working with Mosquitto MQTT.

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 WinSCP

III. Lab Procedures and Results

--- Start of Procedure ---

1) Initial Raspberry Pi setup & MQTT Client/Broker

a. Start by connecting the Raspberry Pi to the power adapter and then power on the system.

b. Start a Remote Desktop Connection on your device and connect to the Raspberry Pi.

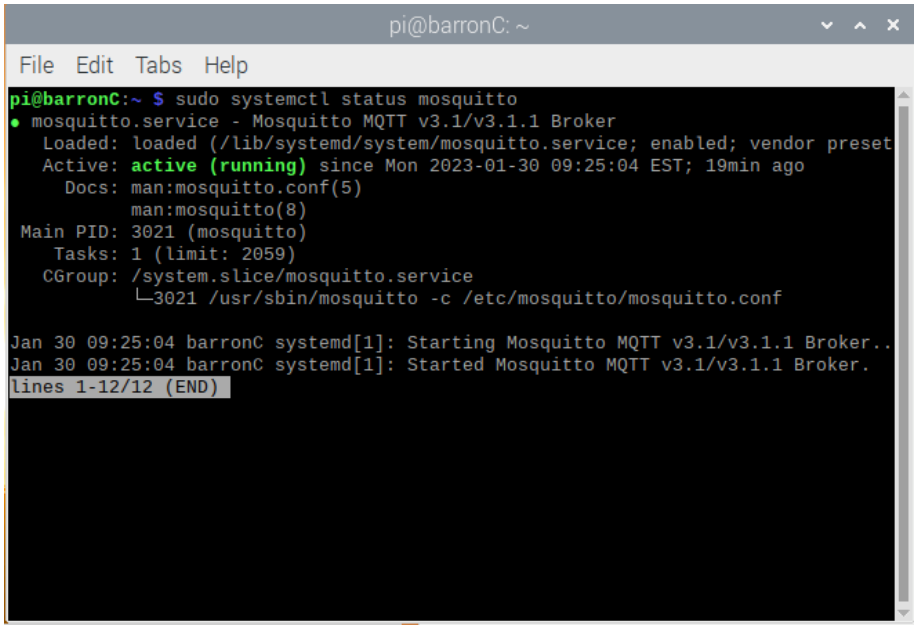
c. After logging into the Raspberry Pi, you need to open a terminal window. Next you will need to update and upgrade the available Linux packages using the command "**sudo apt-get update**" then "**sudo apt-get upgrade**".

2) To create the MQTT Client/Broker architecture we must first install the Mosquitto MQTT Broker with the command: "**sudo apt install mosquitto mosquitto-clients".**

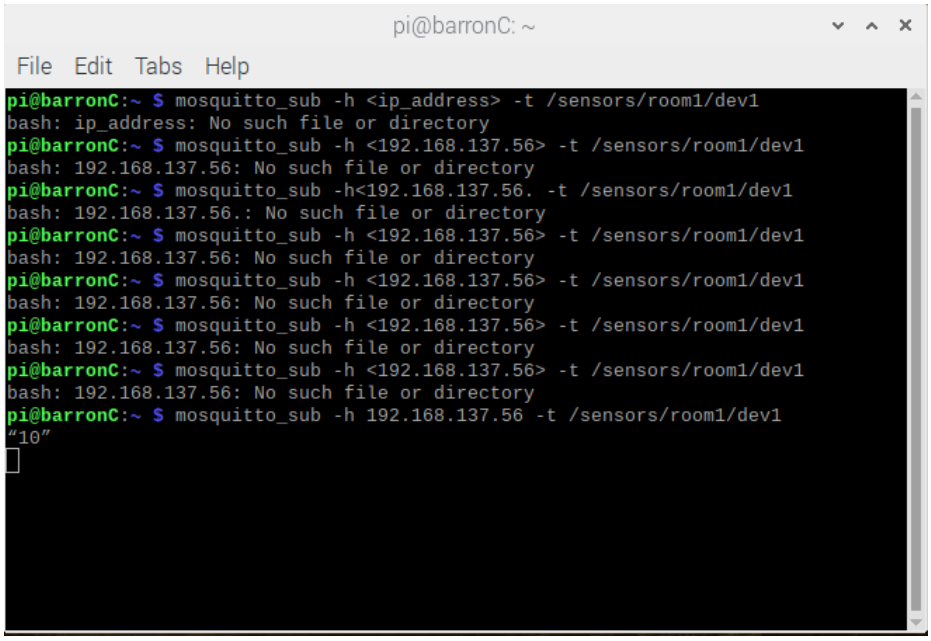
a**.** Once installed, open two more terminals along with the first terminal. The first terminal will serve as the broker, and the second and third terminals will serve as subscribers 1 & 2 respectively.

b. For the first terminal, enter the command "**sudo systemctl enable mosquitto**". The systemctl will activate the Mosquitto program with administrator level access. This will also make our first terminal the broker, and we can check to see if it is running with

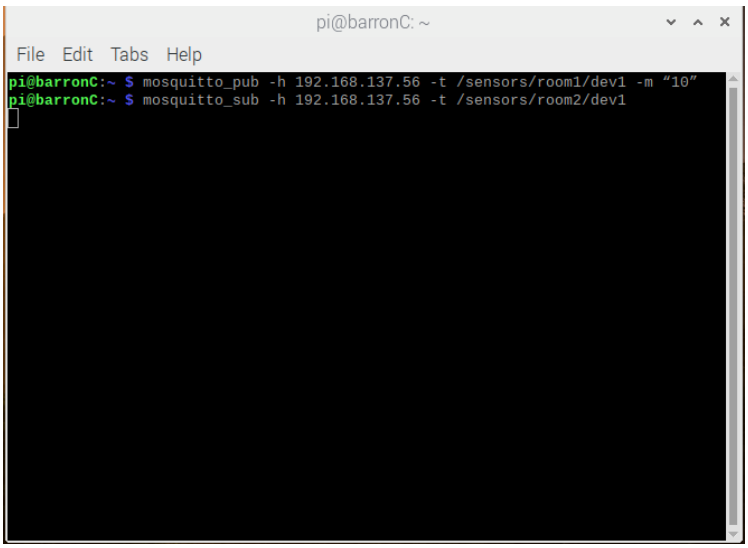
"**sudo systemctl status mosquitto**" and you will get the following information like the picture seen below.



c. For the other two terminals, for the first subscriber, which will be the second terminal, input the command: "**mosquitto_sub -h <ip_address> -t /sensors/room1/dev1**". Note that the IP address will be the IP from your Raspberry Pi.



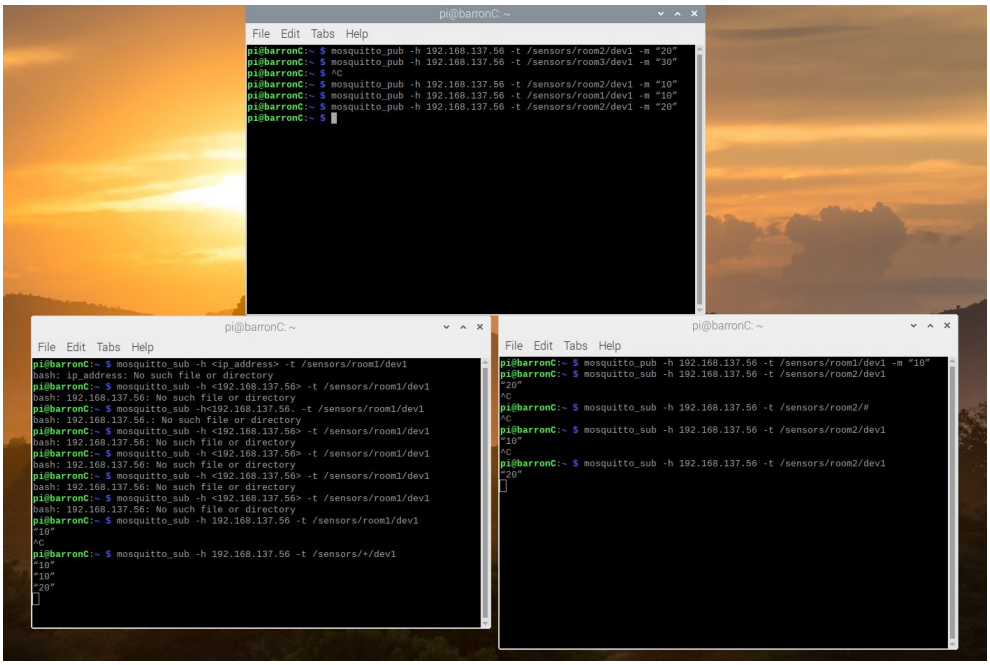
d. The same process that was used in 2c. will be the same here in the third terminal and it will be treated as subscriber 2. Input the following command: "**mosquitto_sub -h <ip_address> -t /sensors/room2/dev1**". Now the second and third terminals should be subscribers 1&2 respectively



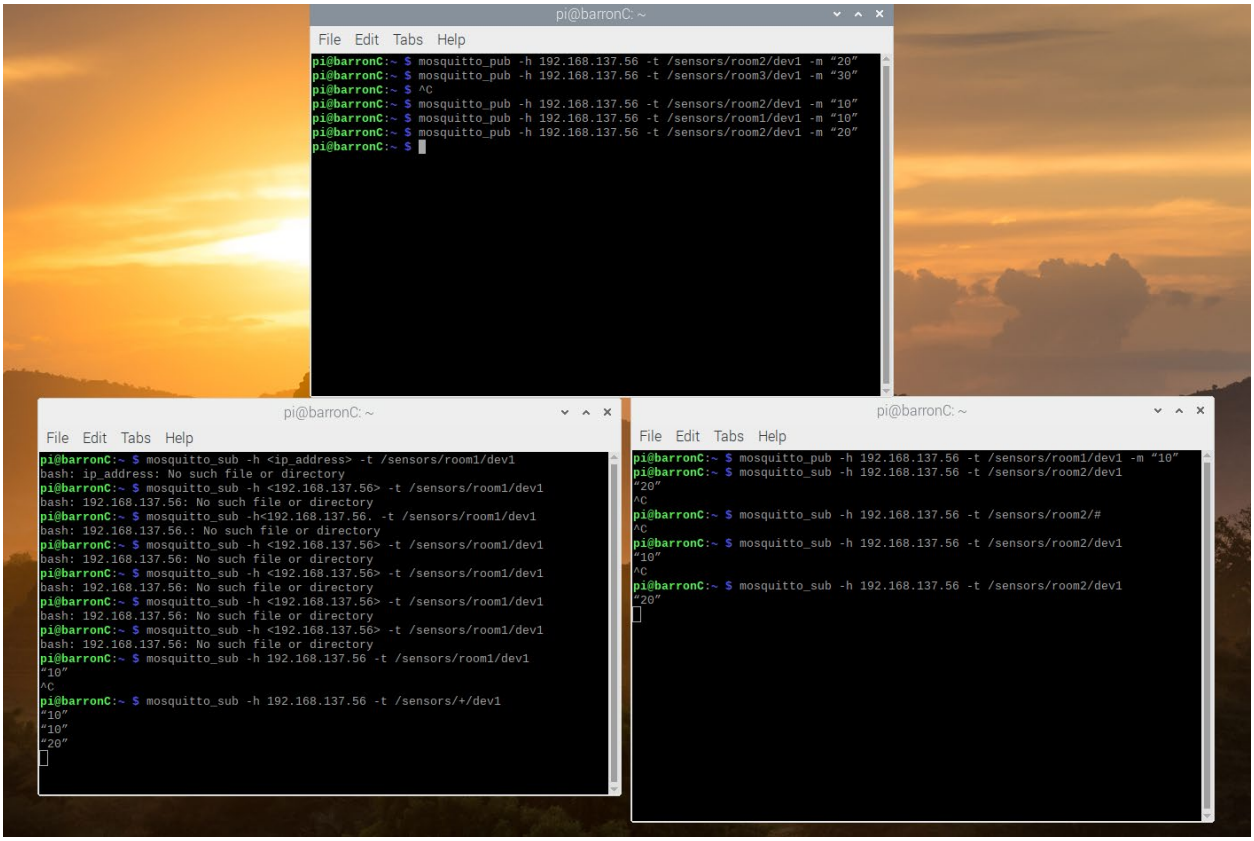
3) Using the first terminal, in order to publish content to each subscriber, they must be sending with the IP address that are given (which is the same as your PI) and the designated room and device number they were assigned. To send a message to each subscriber input the following two commands:

"**mosquitto_pub -h <ip_address> -t /sensors/room1/dev1 -m "10""**
"**mosquitto_pub -h <ip_address> -t /sensors/room2/dev1 -m "20""**

These two commands will send the number "10" to subscriber 1 (Second Terminal) and subscriber 2 (Third Terminal) respectively, since they transmit to the room and device number like as shown in the photo.



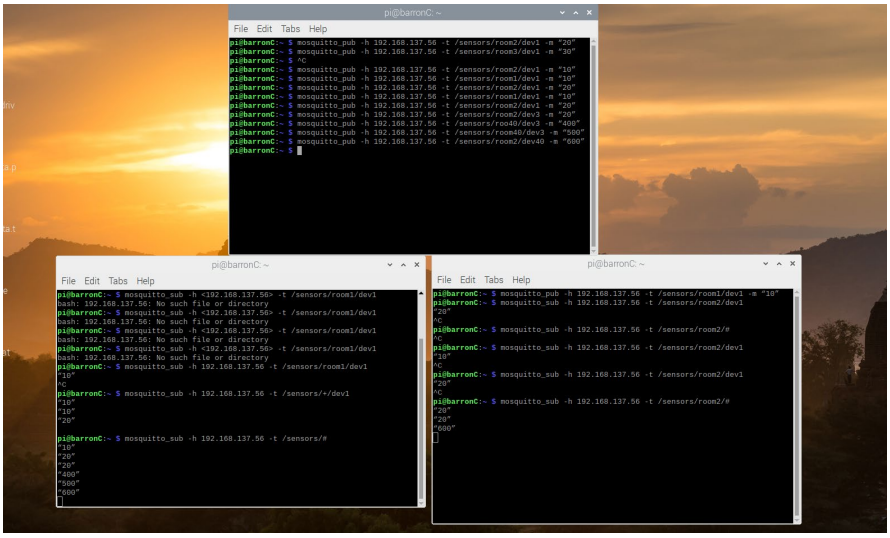
4) Exit Subscriber mode on Subscriber 1 by hitting Ctrl + C in the terminal. Then using the same command that was in step 2c. subscribe to a new topic but instead of /sensors/room1/dev1, subscribe to **/sensors/+/dev1**. This will make it so that Subscriber 1 will receive all the messages and Subscriber 2 will only receive the second message like in the photo shown below.



5) After exiting subscriber mode for both subscribers we will use the wildcard command which is a “#”.

a. To start we will need subscriber 1 to subscribe to the topic “**/sensors/#**”. For subscriber 2 we will subscribe to the topic “**/sensors/room2/#**”.

b. We are going to publish the messages in a similar method to earlier steps but this time we will be using different room numbers and device numbers for each case. After implementing this the results for Subscriber 1: it will receive messages regardless of the room or device but in the case of Subscriber 2: it would only receive the messages if was in the same room but it will receive messages across any devices.

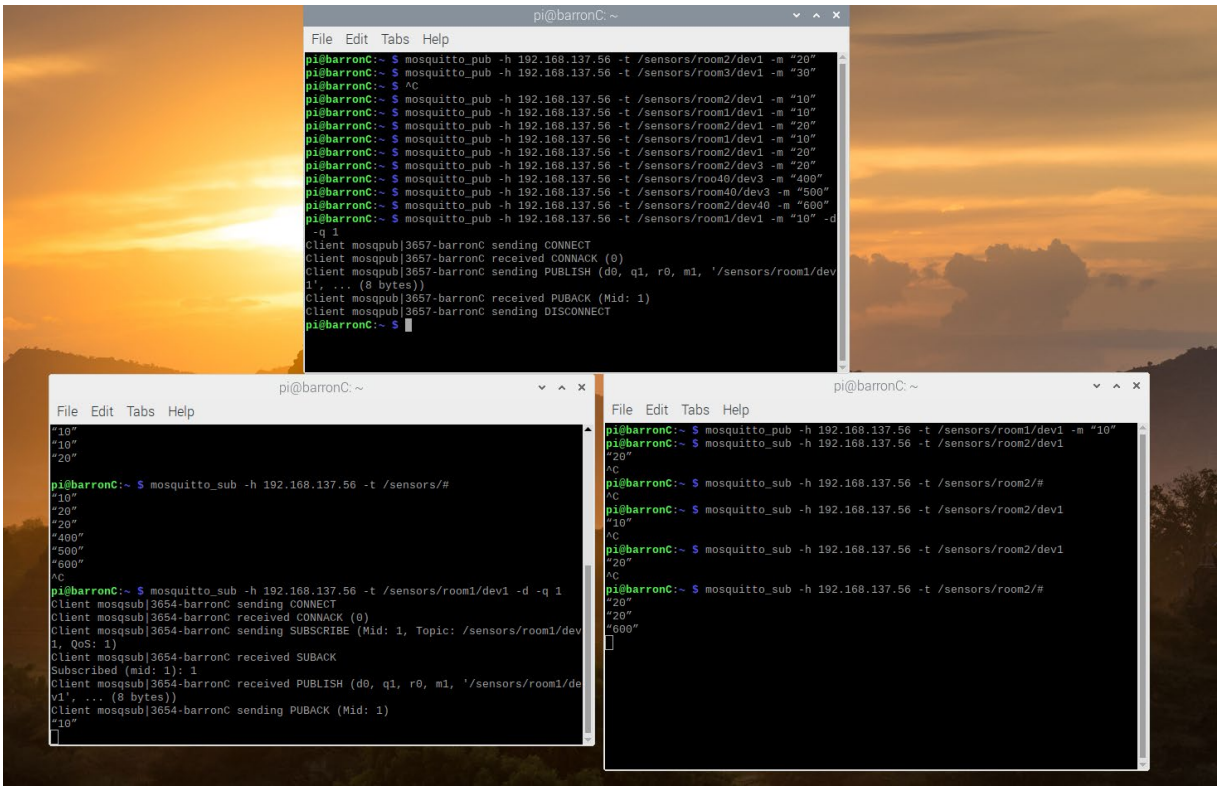


6) In this portion we will be handshaking messages and QoS

a. We start with the Publisher and Subscriber 1, we are going to enable debug messages using the “-d” option to see handshaking messages on both sides. We can also use the “-q” option to change the QoS flags (0, 1, 2) on both the Publisher and Subscriber side. Use the following commands and you should receive results like the pictures.

“**mosquitto_sub –h <ip_address> -t /sensors/room1/dev1 -d –q 1**”

“**mosquitto_sub –h <ip_address> -t /sensors/room1/dev1 -m “10” -d –q 1**”



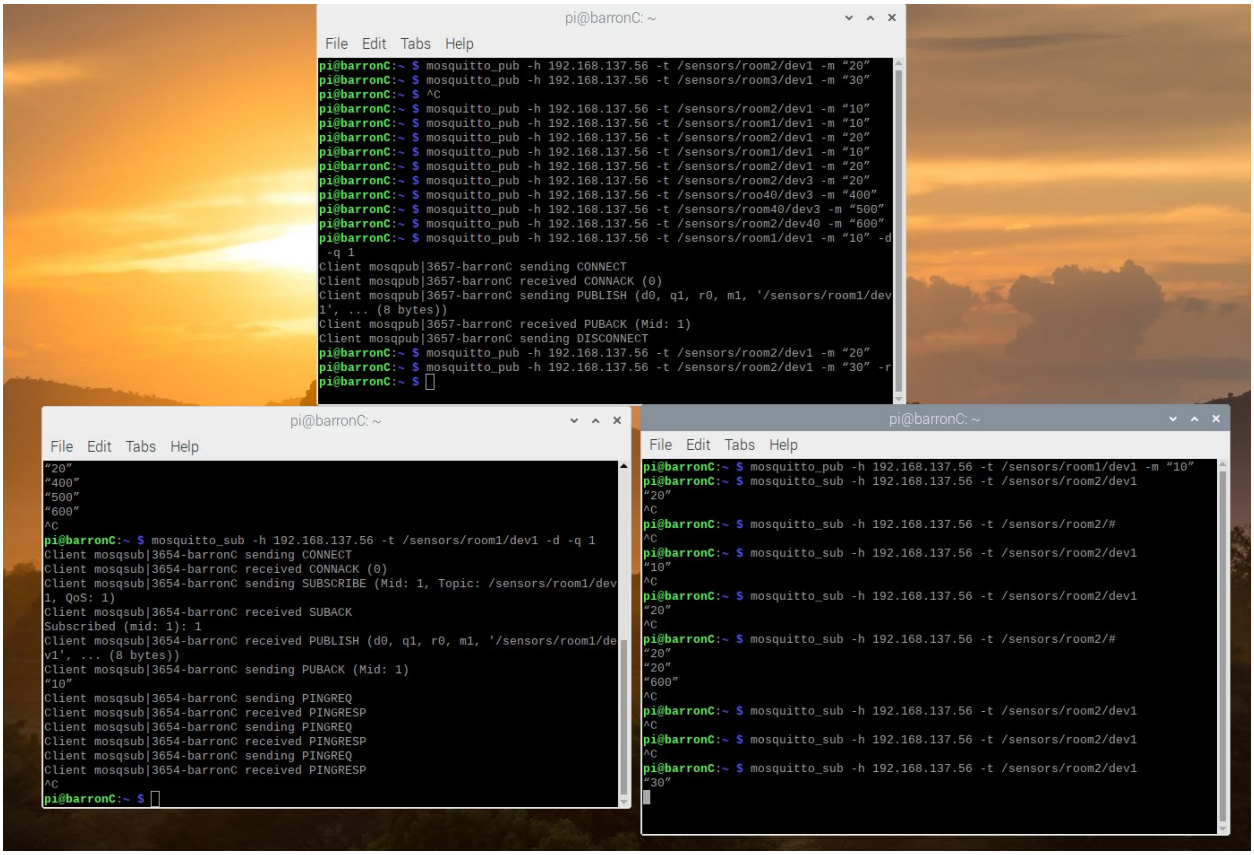
7) We are going to implement Retained messages

a. As a preliminary check we first need to check the Subscriber 2 Terminal and stop the client if it is still running. After that, we will go to the Publisher Terminal, we need to publish the second message stated earlier pertaining to room2. We will then go back to subscriber 2 and subscribe to the same topic. You should see that nothing should have been delivered to the subscriber during or after publishing.

b. We will now need to stop the client in Subscriber 2 terminal. The Publisher will now need to publish the same message from earlier but with the "-r" option added to the end of the command. The command is as follows

"**mosquitto_pub –h <ip_address> -t /sensors/room2/dev1 -m "30" -r**"

c. Now start the client back up in Subscriber 2 and you will now be able to see that the retained message will be delivered immediately when the client is subscribed to the same topic.



IV. Conclusion

Overall, this lab proved that networking, at least with the same machine, is very much possible and doable. Honestly, seeing how all three terminals communicate with each other really did set up some possibilities in the future when it comes to implementing it into more complex designs. There were no big issues encountered during the lab sessions and no real areas for improvement are needed for this lab given how straightforward it was, but a small nitpick can be explaining what some commands do like -d, -q, and –l commands a little better, but it is not necessary.