

**Faculty of Engineering & Applied Science**

**Experiment Name:** **Leveraging MQTT communication for IoT applications**

**Experiment date: 11/16/2022**

**Group Number*: 4***

**Section CRN: 44432**

**Course Instructor: *Ramiro Liscano***

**Lab TA:*****Sifatul Mostafi***

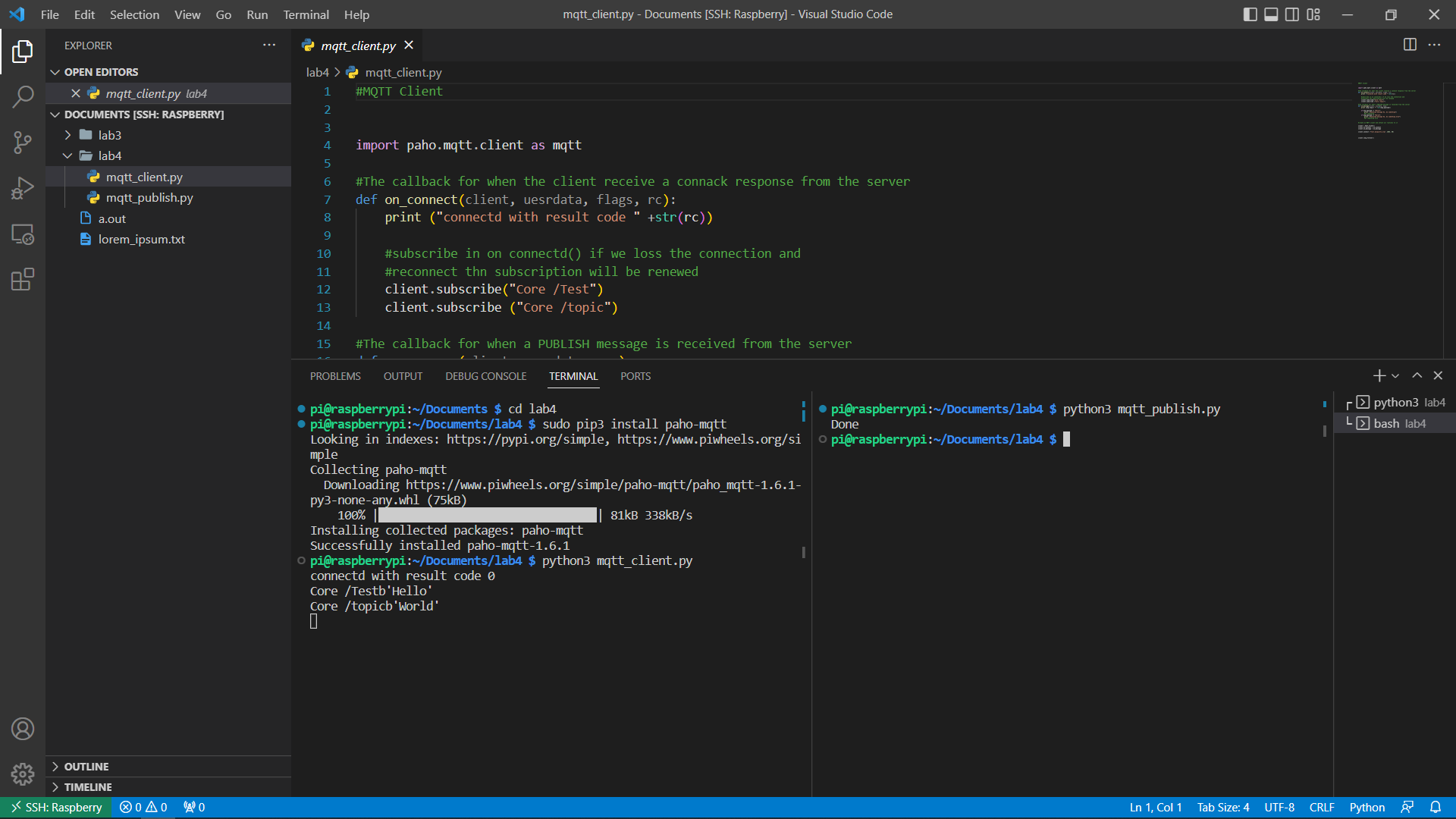
| **Student Name** | **Student Id** |
| --- | --- |
| Preet Patel | *100708239* |
| Tiwaloluwa Ojo | *100700622* |
| Waleed El Alawi | *100764573* |

**Learning Objective**

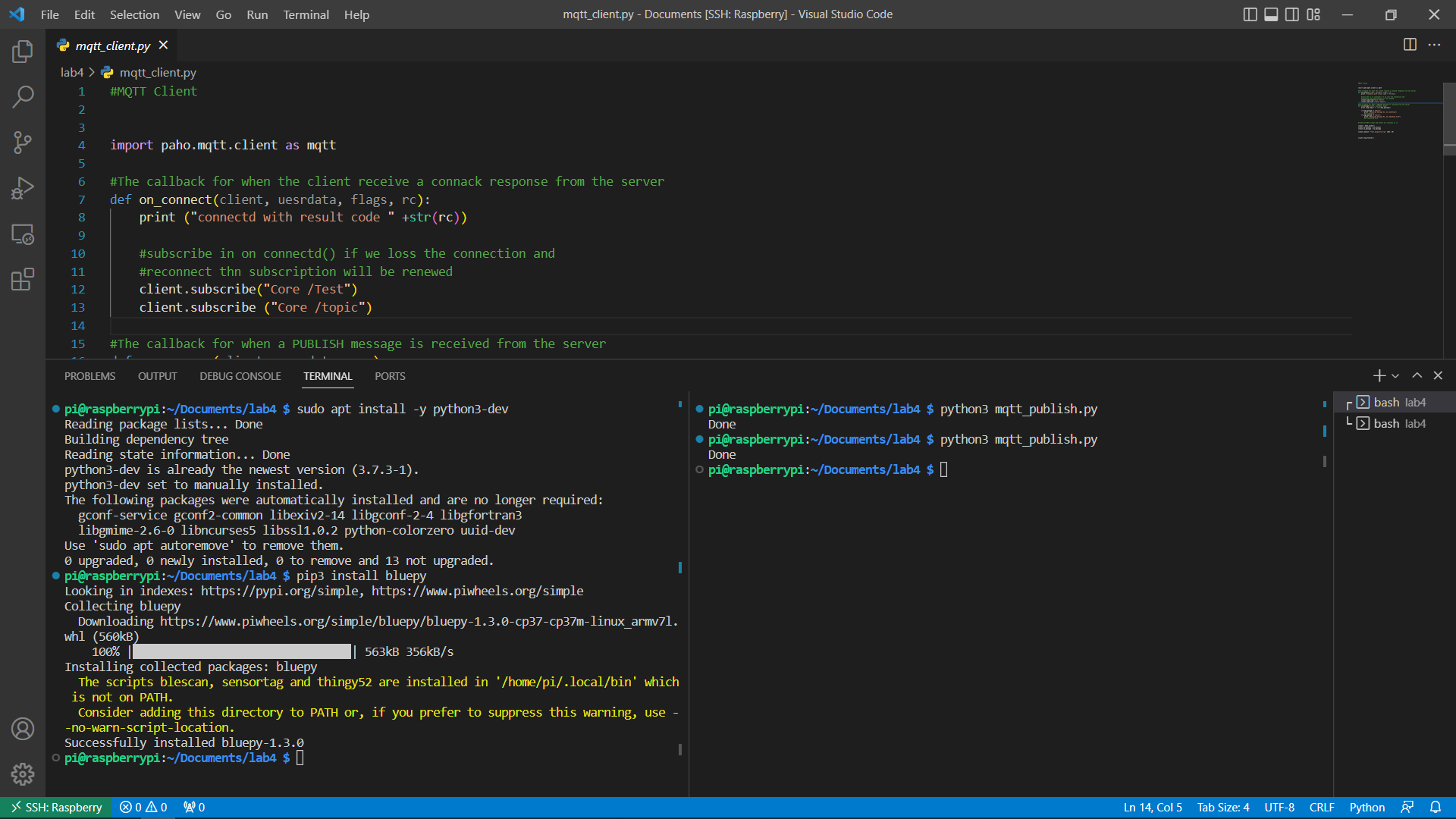
The objective of this lab is to install, learn and use MQTT, paho mqtt, and lastly mosquitto for sending data from one client to another. We learned to install and configure Mosquitto on the Raspberry Pi. We learned to use python code which leverages Mosquitto to test and send sensor data to another client.

**Deliverables**

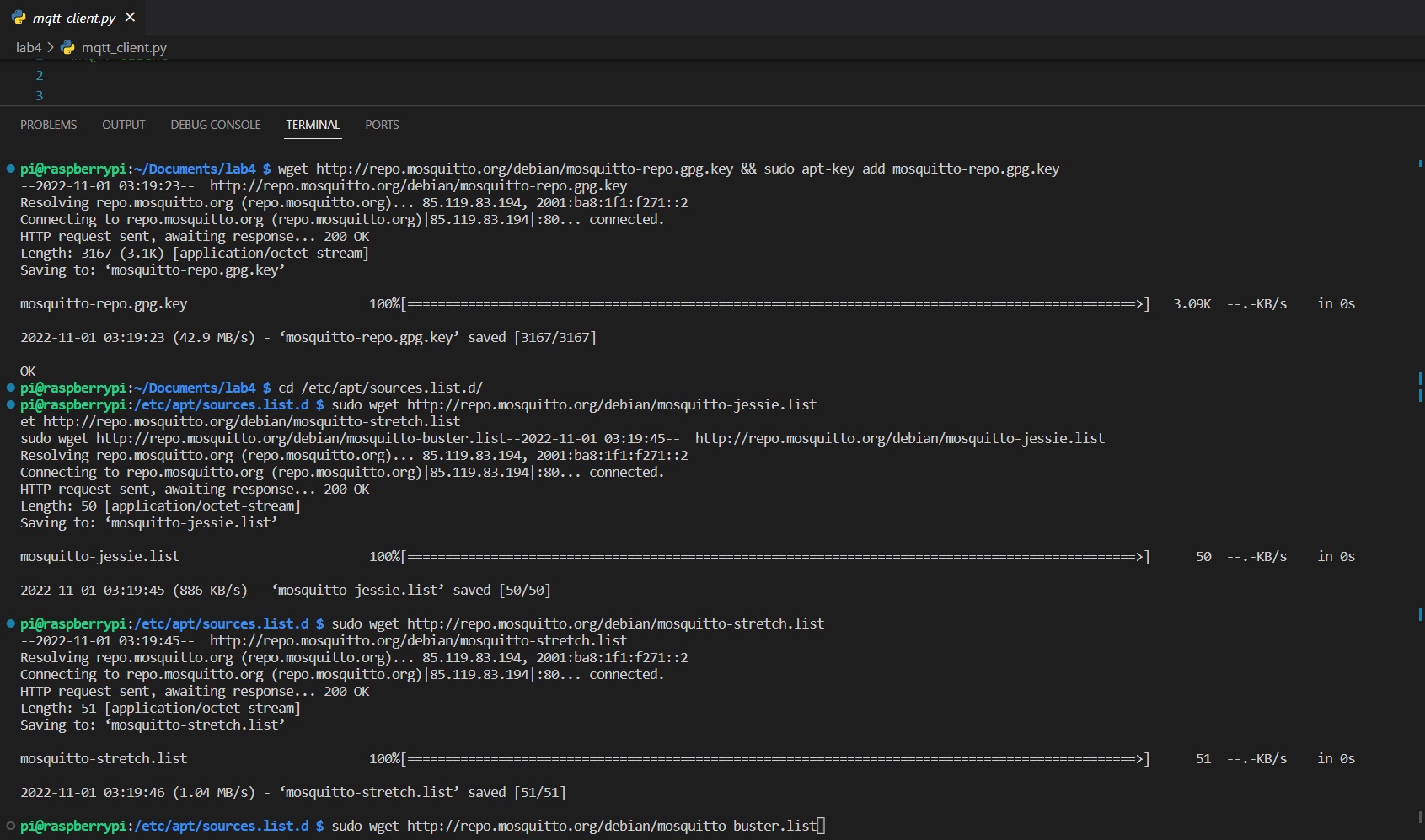
**Utilize bluepy library to communicate.**

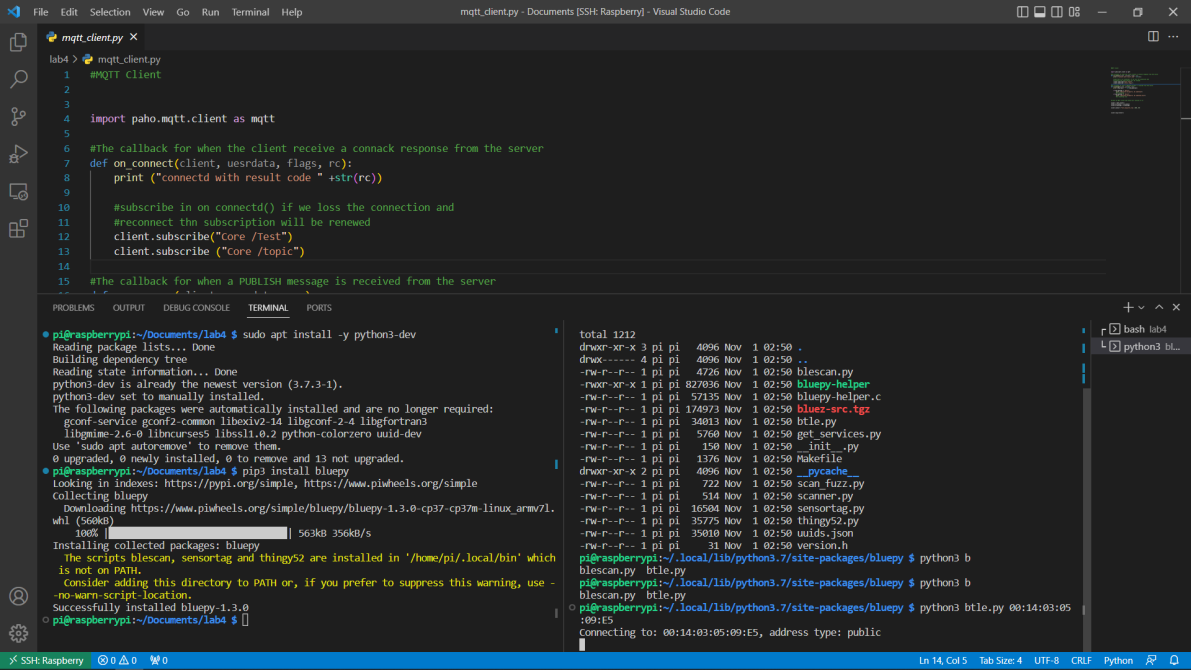
****

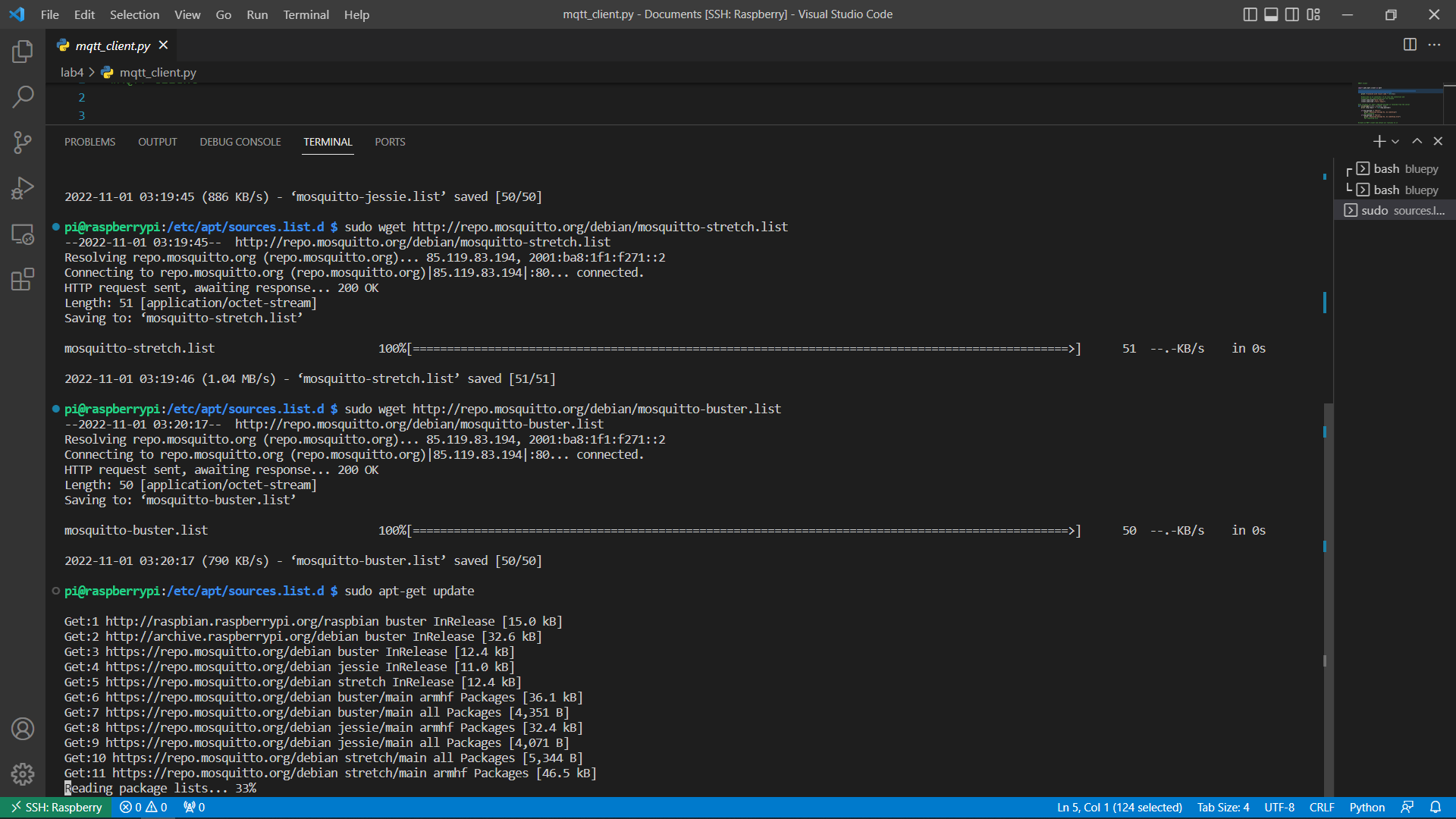
**Install paho-mqtt. Run the mqtt\_client.py to receive the msg stream from the mqtt\_publish.py. The topic the clients and publisher are using is ‘/Test’ & ‘topic’**

****

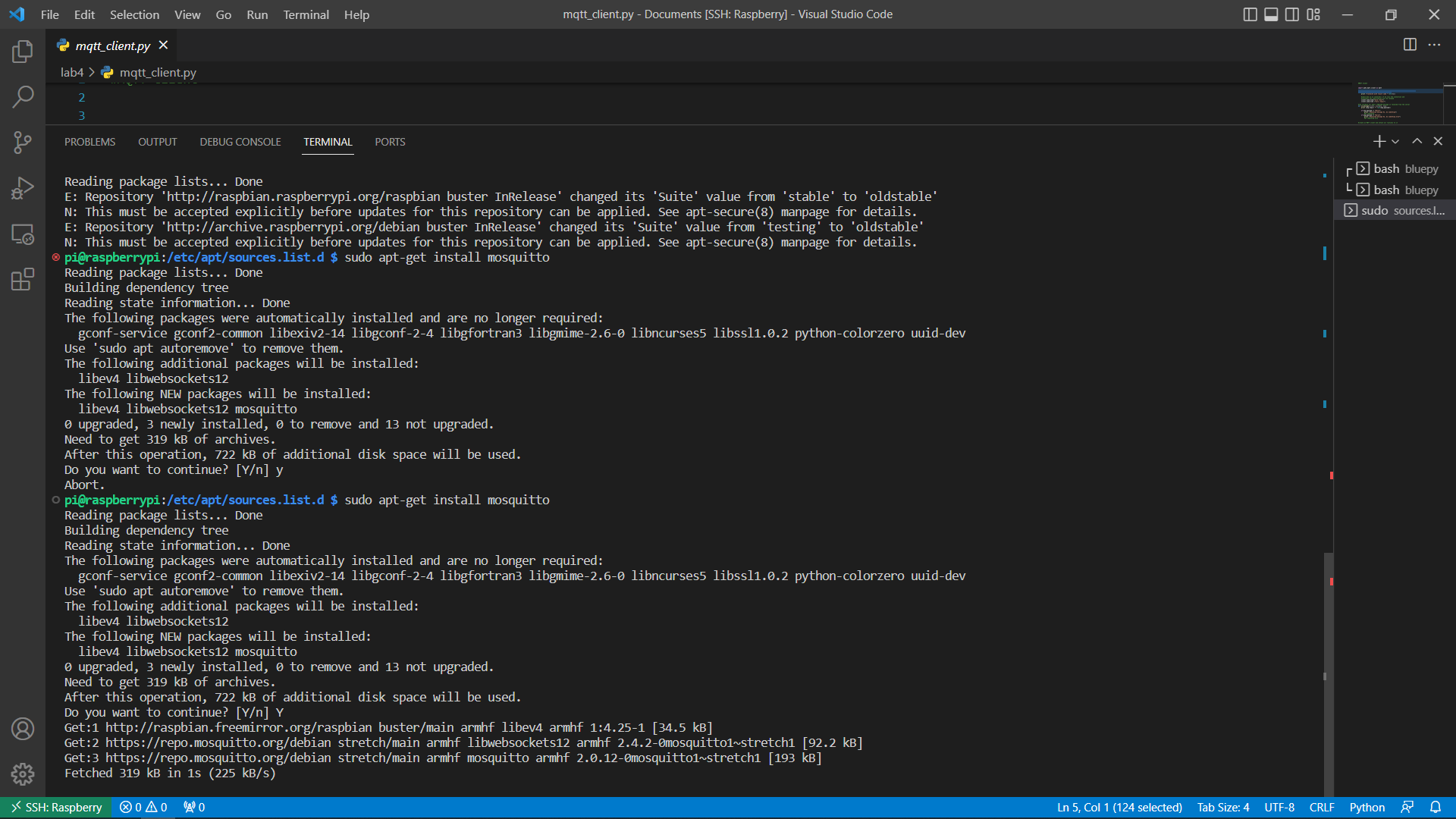
**Install bluepy python package and verify python3 is installed.**

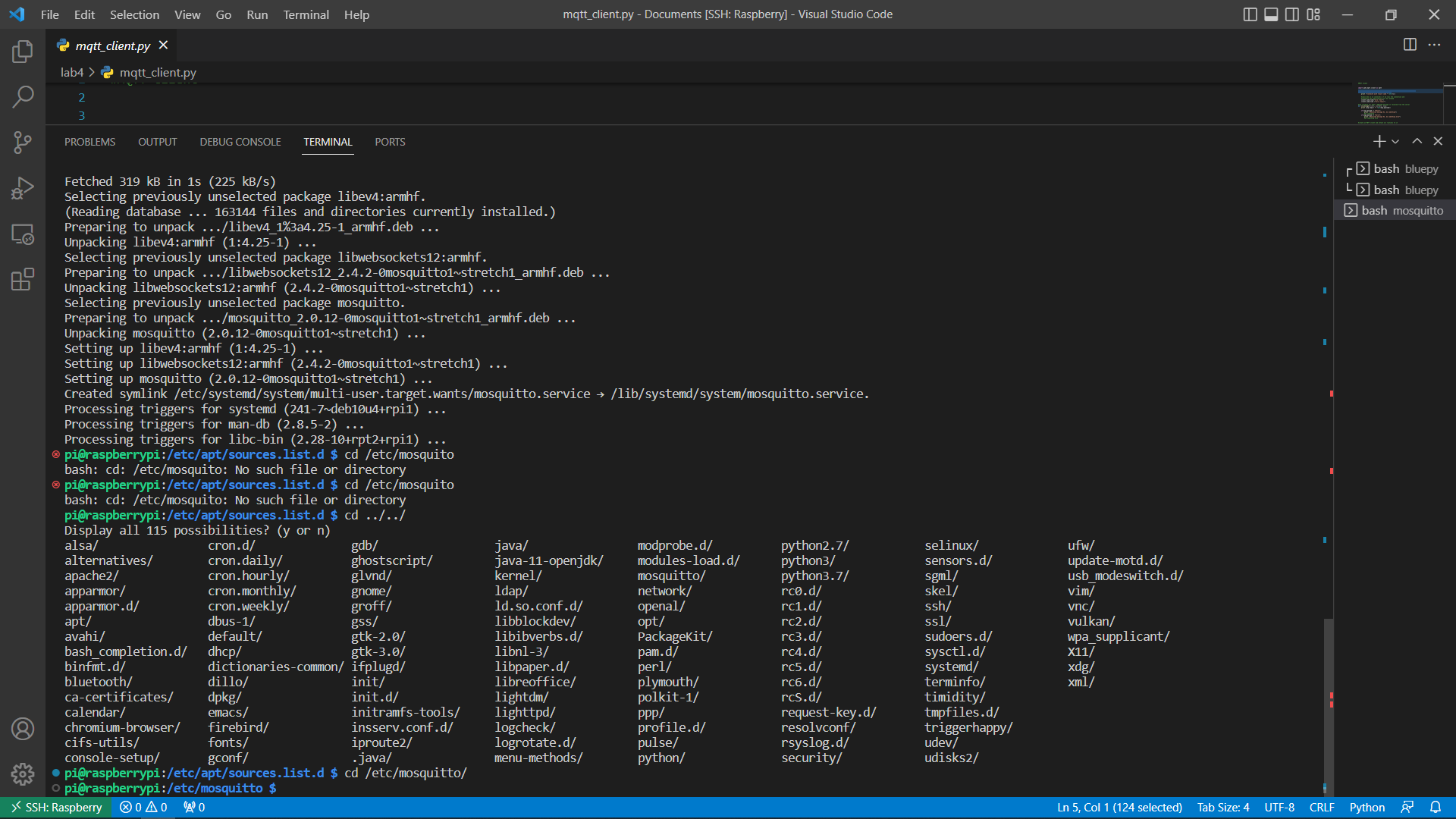
****

****

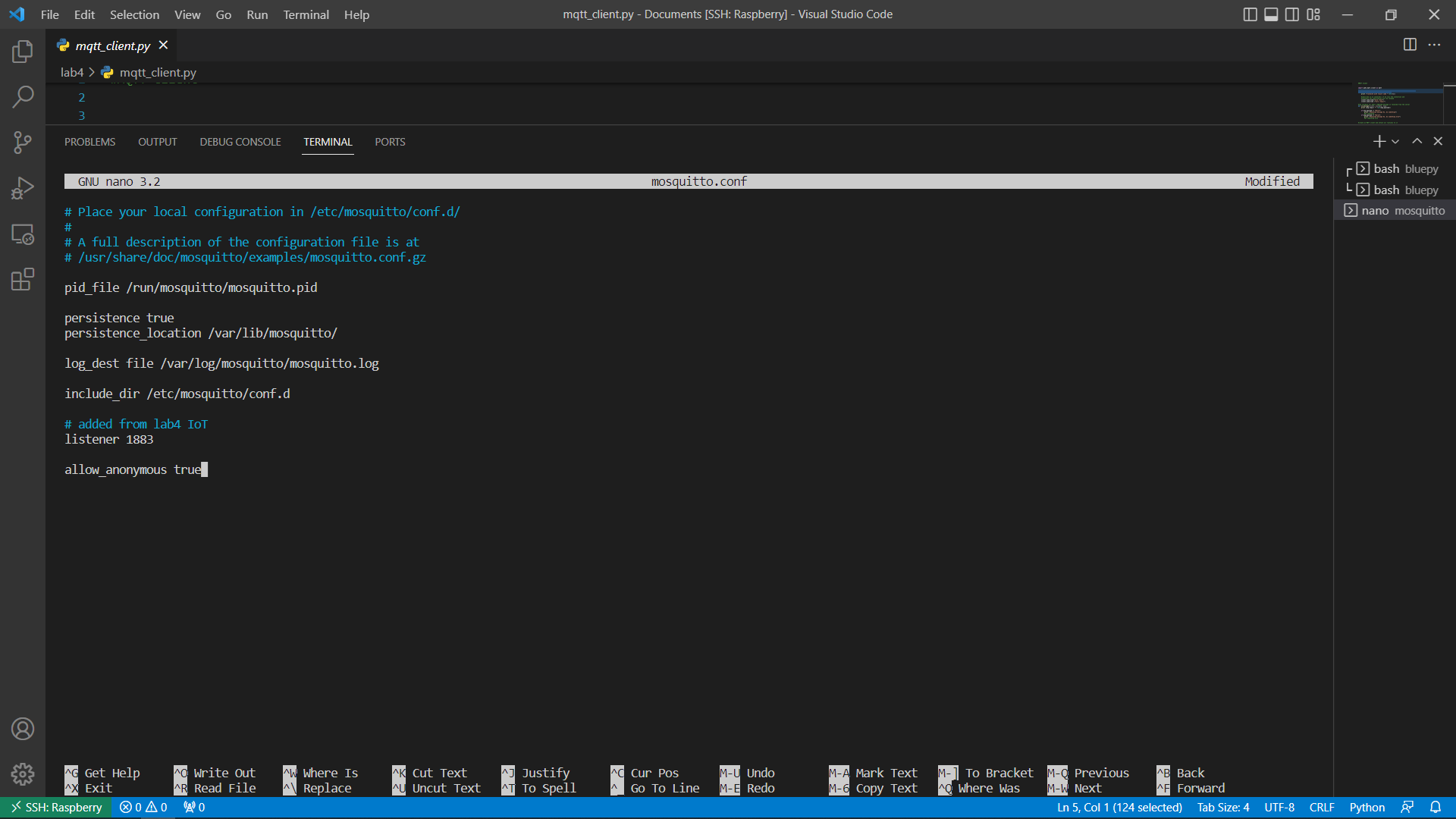
****

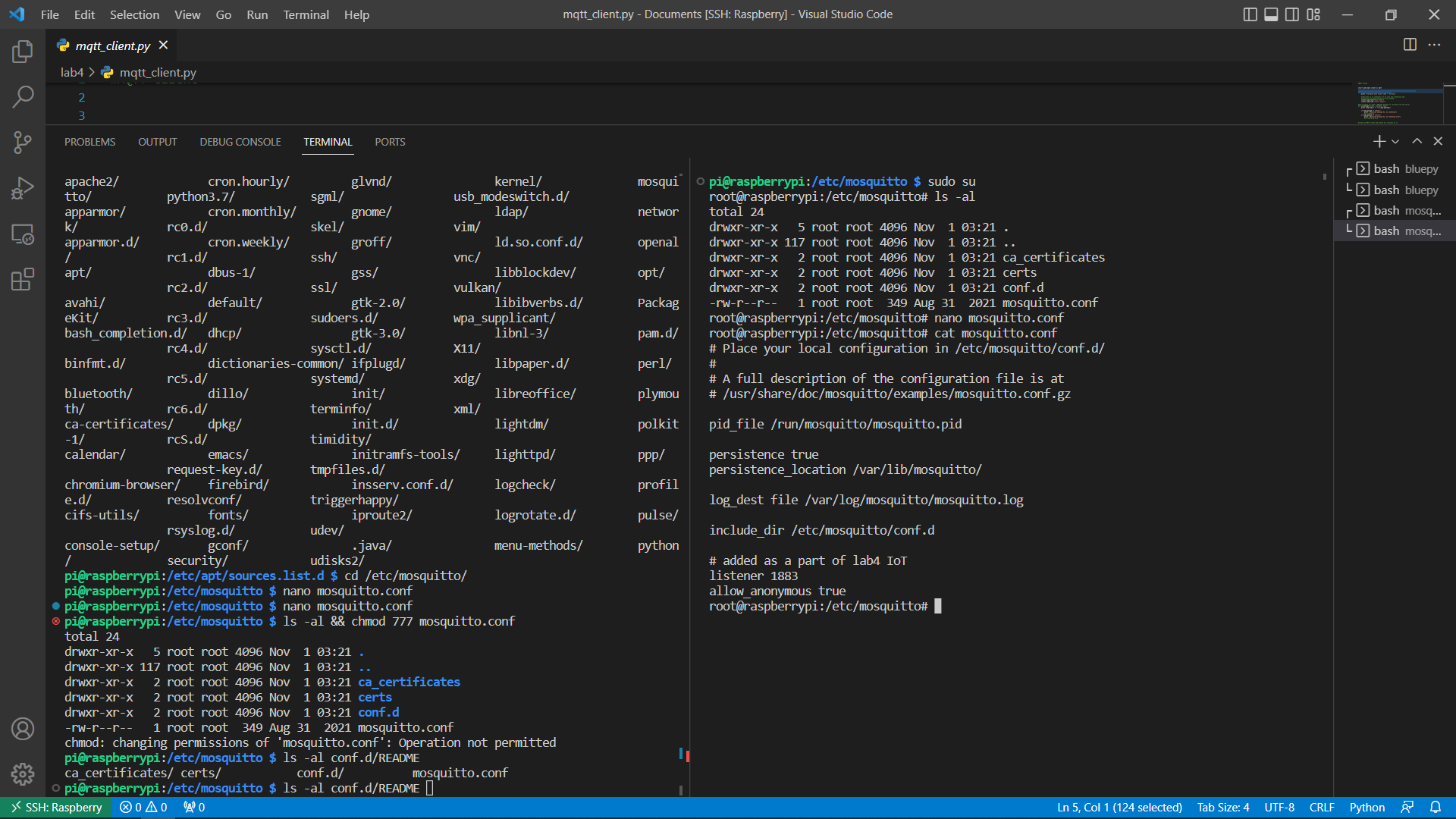
**Here we installed mosquitto, some other system dependencies and configure it.**

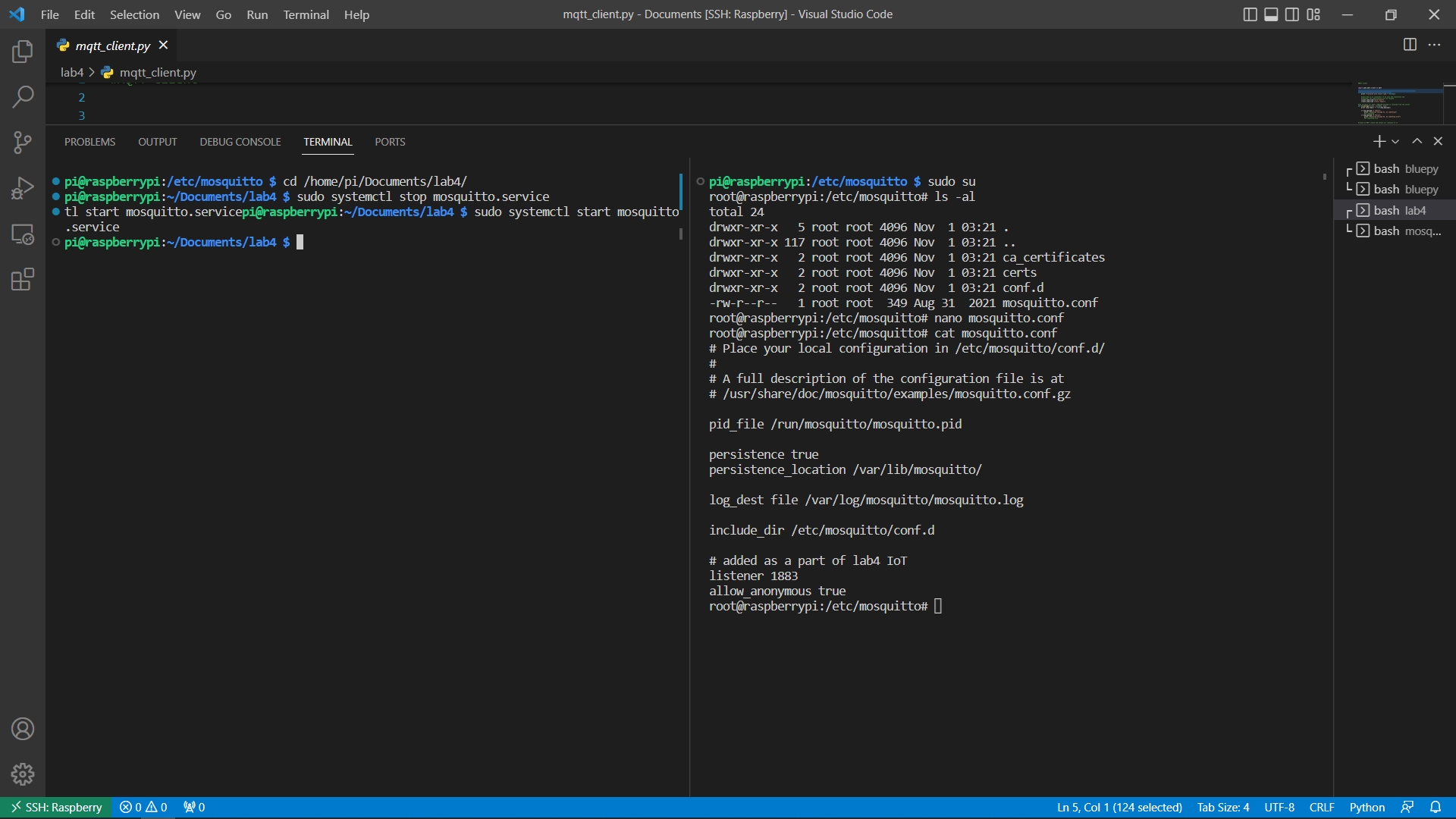


****

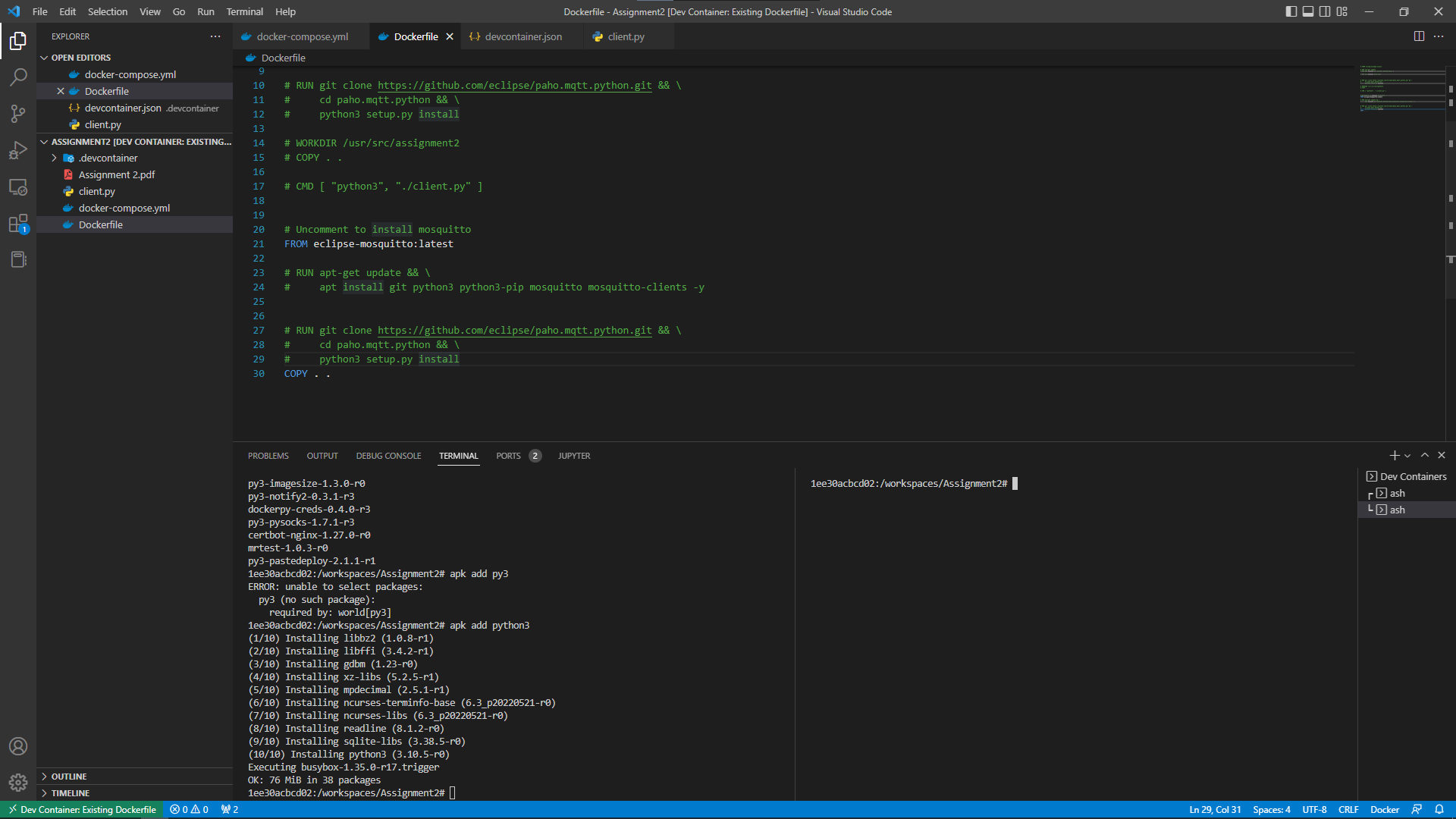
**Modified the port Mosquitto will listen on in the /etc/mosquitto/conf.d file. The file did not have write permission. We used chmod to enable it using superuser permissions.**



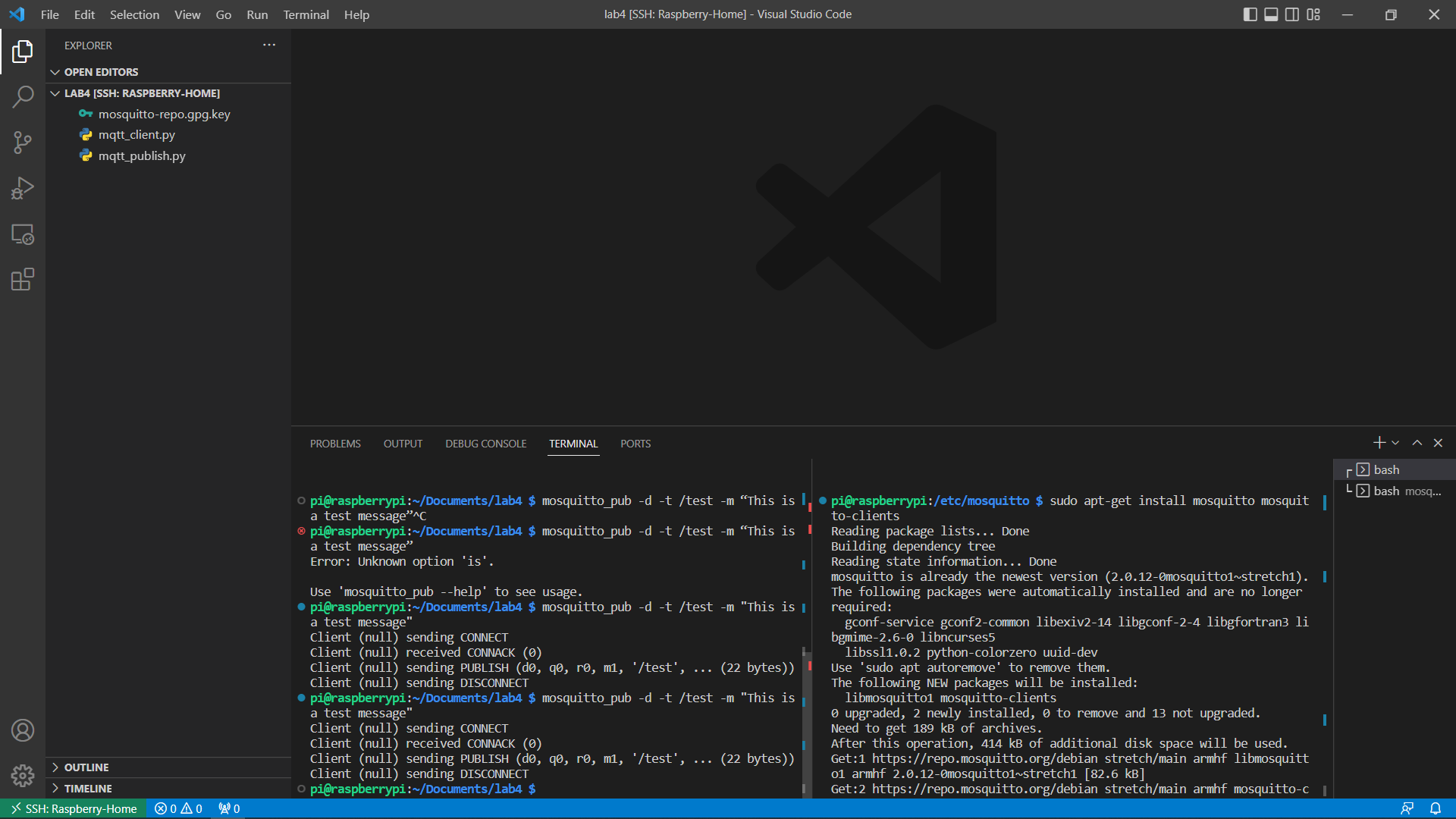


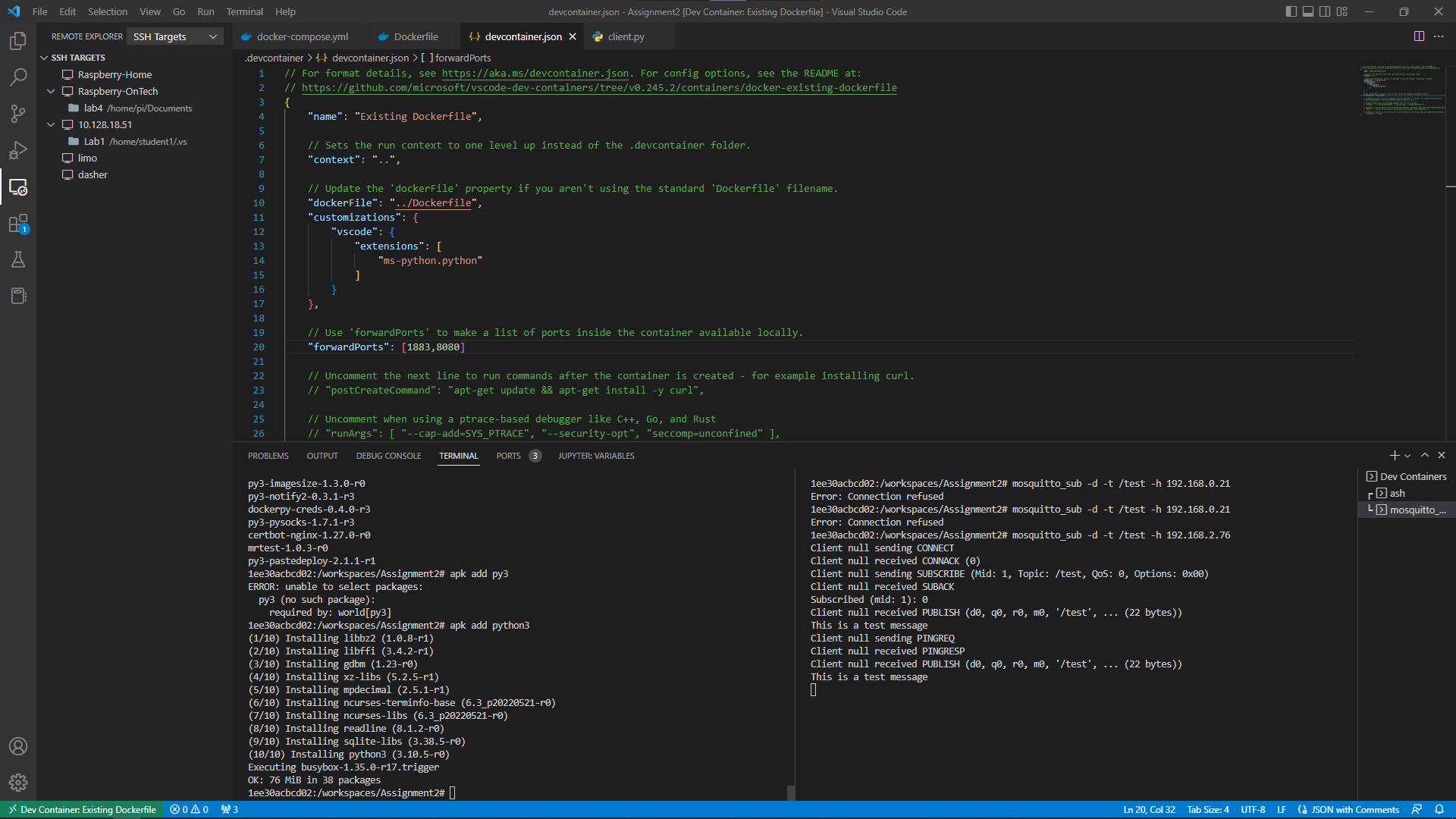


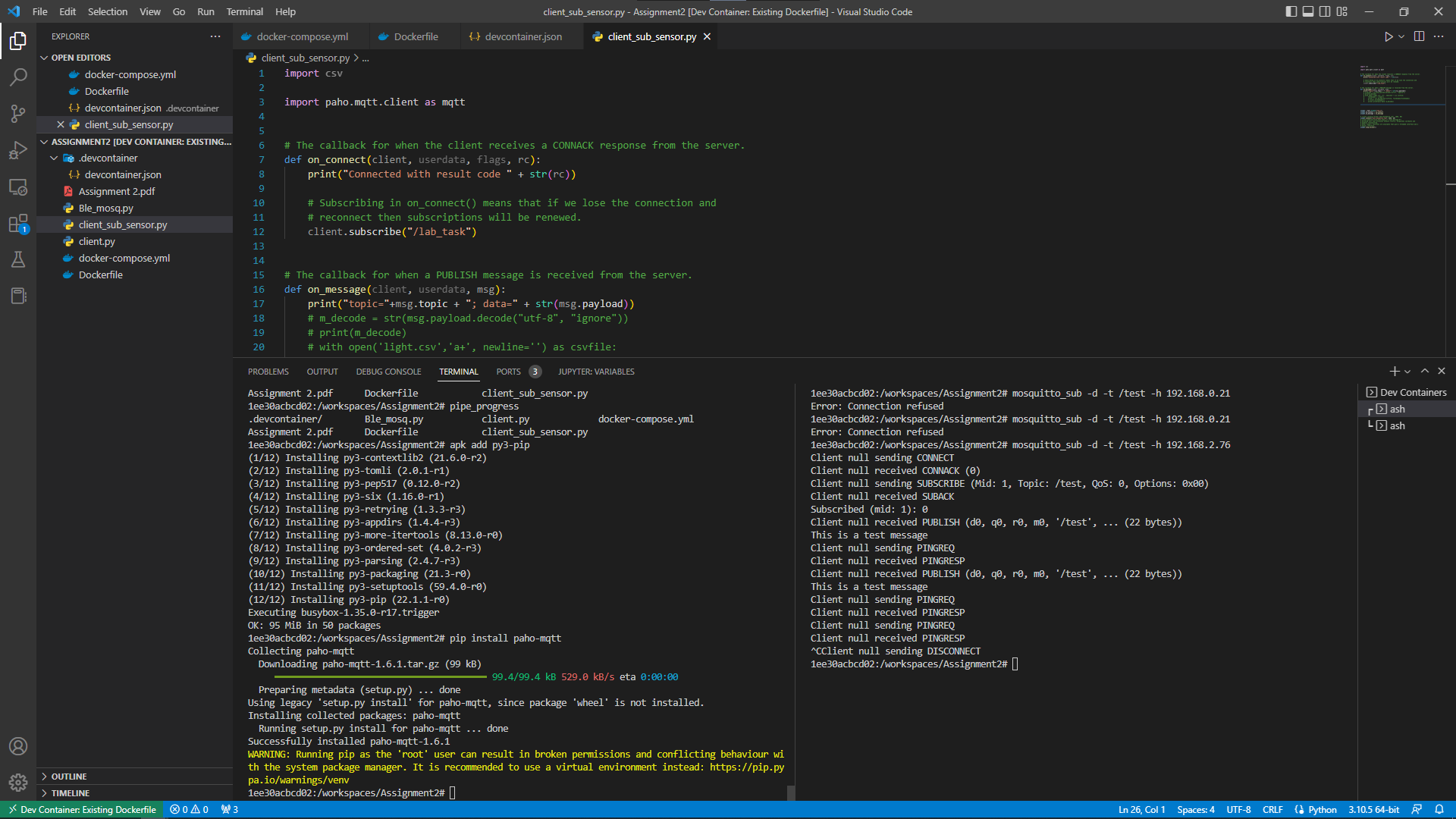
**Starting the mosquitto service**

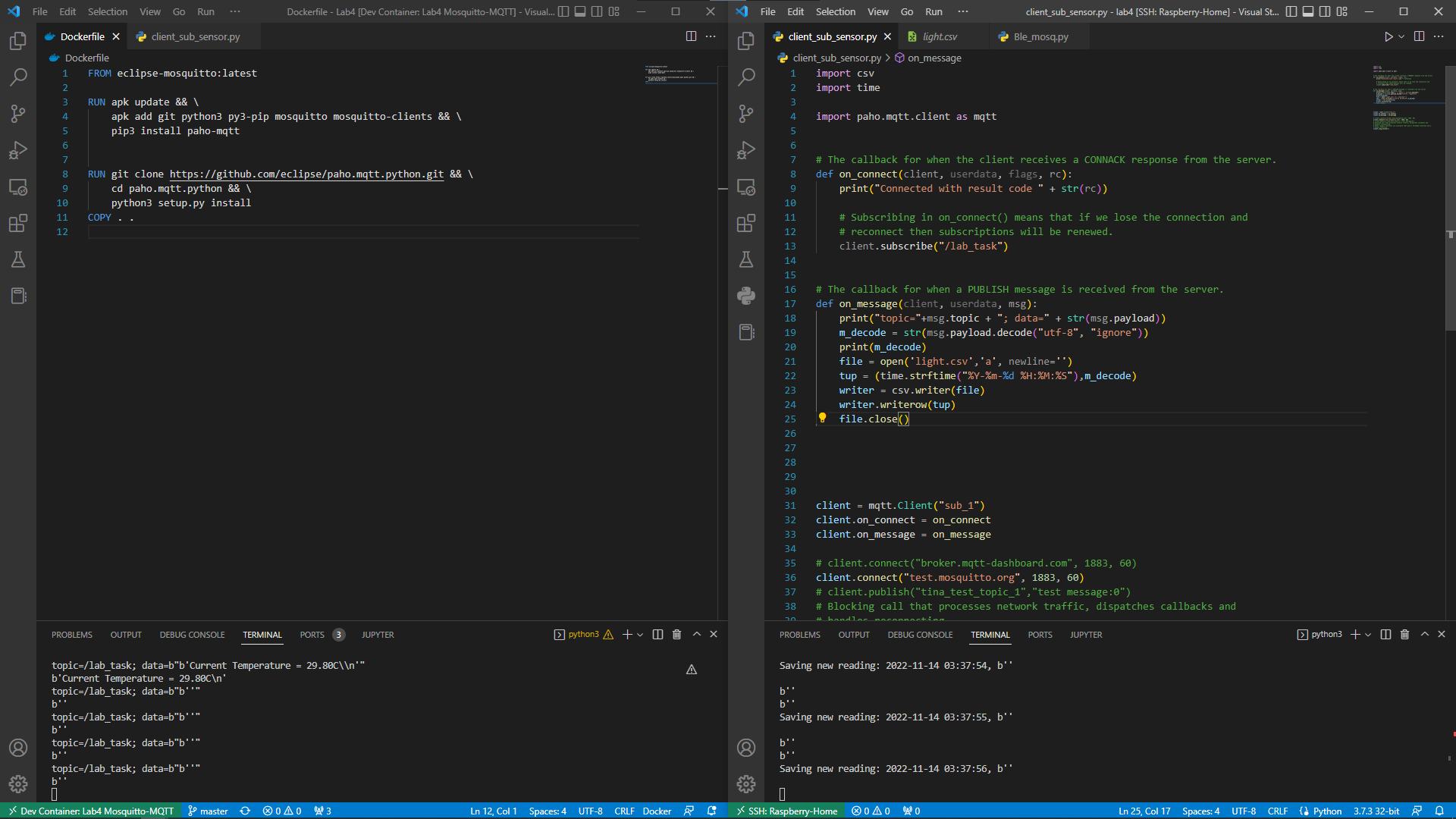
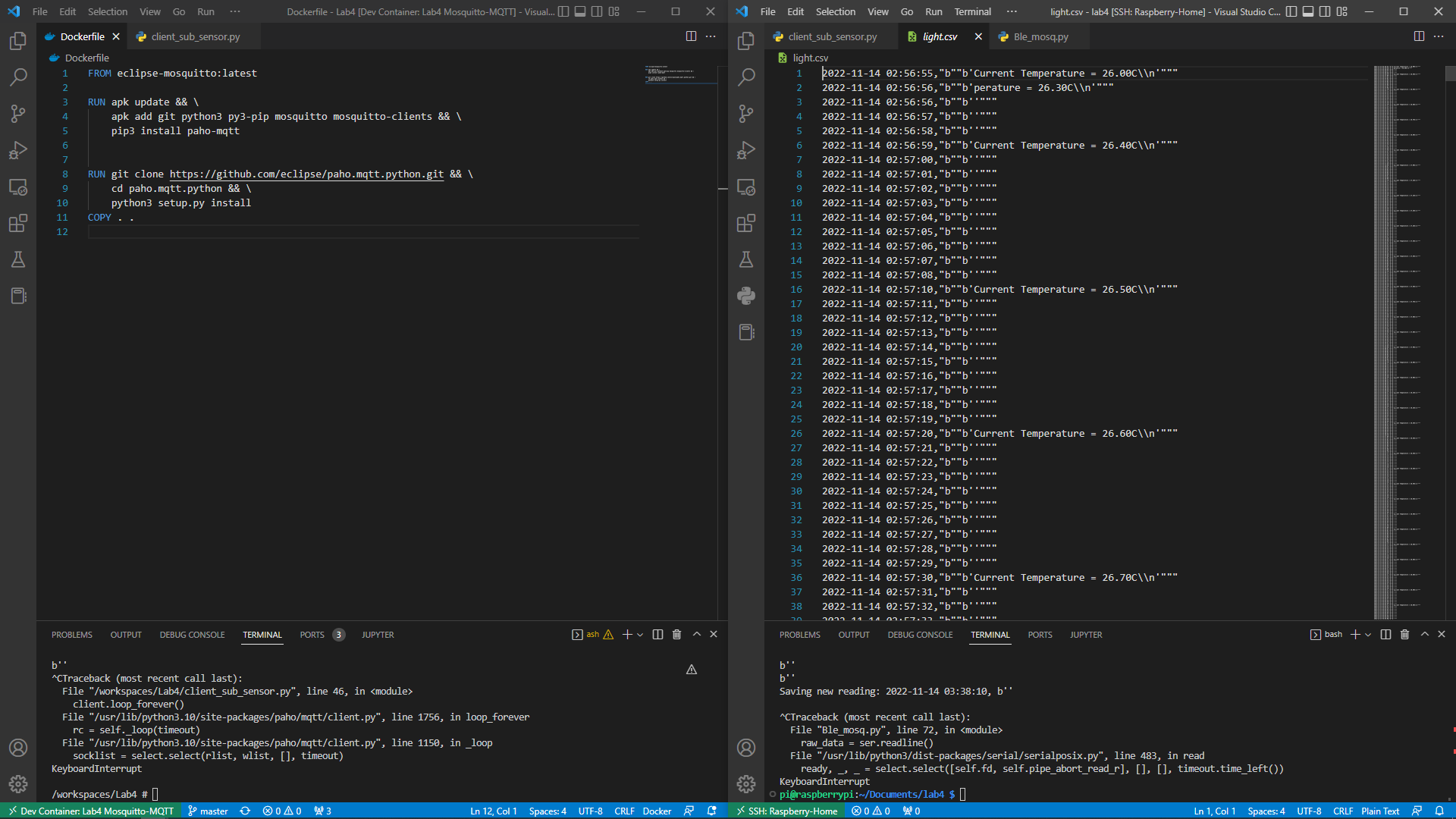


**Start our Ubunto Docker Container, install mosquitto and run the subscription in 5.2 step 12.**



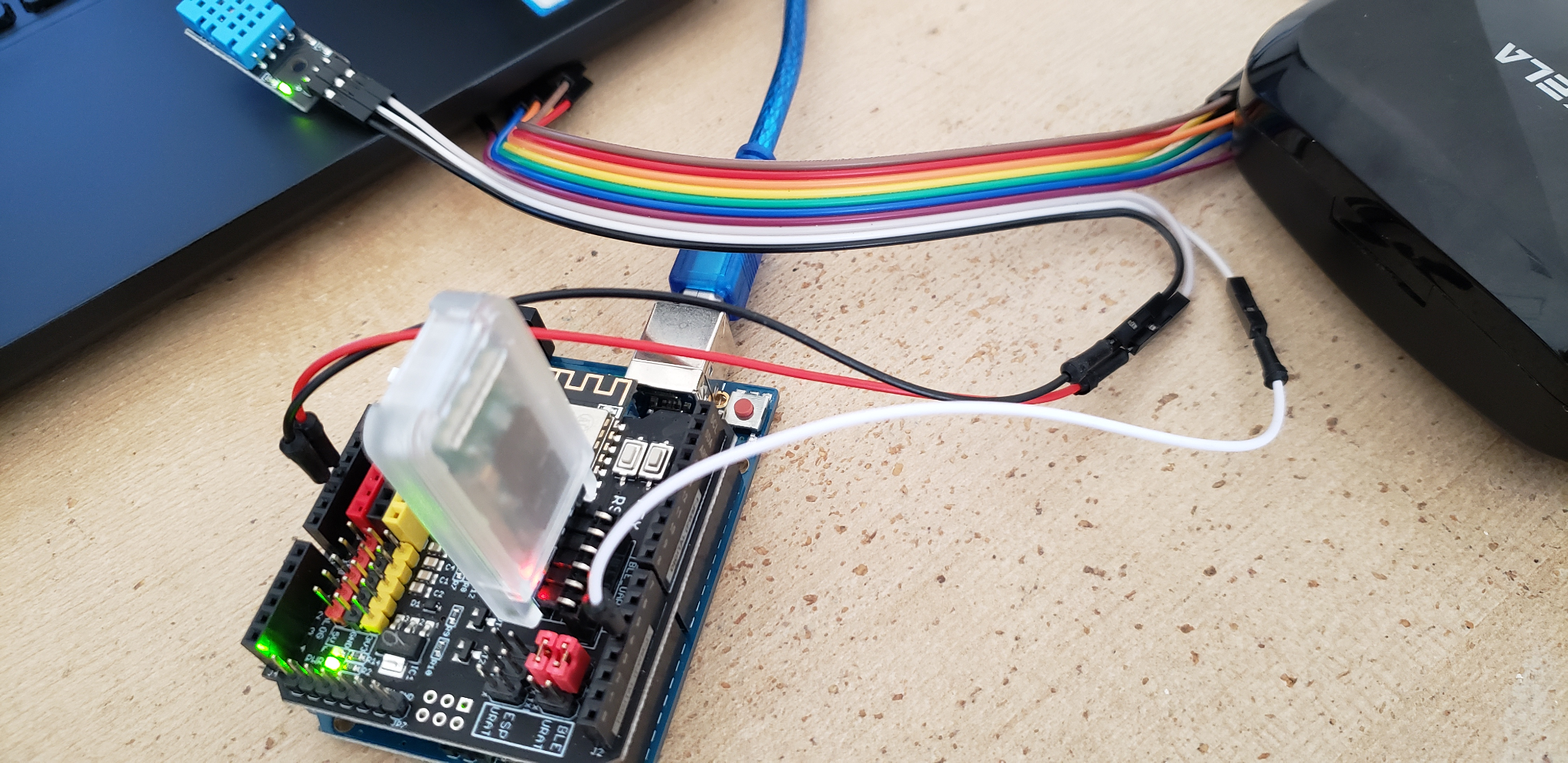
**Ran “sudo rfcomm bind /dev/rfcomm0 00:14:03:05:09:E5 1” to setup the device port for BLE messages.** 







**Ran a mosquitto subscriber service on the container while the Pi was the publisher of the mqtt messages. This was performed for both the “/test” and the “/lab\_task” topics. We were also able to send the output to a csv file as seen in the light.csv.**



**Conclusion**

In conclusion, we were able to install and utilize bluepy library to communicate, send messages between the raspberry Pi and Ubuntu OS using MQTT, Broker install mosquitto on the raspberry Pi The subscriber was Ubuntu machine(docker alpine container) while the Publisher was the raspberry Pi. We learned to use python code which leverages Mosquitto to test and send sensor data to another client. In the final task of the lab, we used BLE HC05 to communicate with raspberry Pi.

Attached below is the source code which we created for the purposes of this lab

**Dockerfile**

FROM eclipse-mosquitto:latest

RUN apk update && \

apk add git python3 py3-pip mosquitto mosquitto-clients && \

pip3 install paho-mqtt

RUN git clone https://github.com/eclipse/paho.mqtt.python.git && \

cd paho.mqtt.python && \

python3 setup.py install

COPY . .

**Client\_sub\_sensor**

import csv

import time

import paho.mqtt.client as mqtt

# The callback for when the client receives a CONNACK response from the server.

def on\_connect(client, userdata, flags, rc):

print("Connected with result code " + str(rc))

# Subscribing in on\_connect() means that if we lose the connection and

# reconnect then subscriptions will be renewed.

client.subscribe("/lab\_task")

# The callback for when a PUBLISH message is received from the server.

def on\_message(client, userdata, msg):

print("topic="+msg.topic + "; data=" + str(msg.payload))

m\_decode = str(msg.payload.decode("utf-8", "ignore"))

print(m\_decode)

file = open('light.csv','a', newline='')

tup = (time.strftime("%Y-%m-%d %H:%M:%S"),m\_decode)

writer = csv.writer(file)

writer.writerow(tup)

file.close()

client = mqtt.Client("sub\_1")

client.on\_connect = on\_connect

client.on\_message = on\_message

client.connect("test.mosquitto.org", 1883, 60)

client.loop\_forever()

**Ble\_mosq.py**

import serial

import numpy as np

import time

import paho.mqtt.client as mqtt

import serial, time

ser = serial.Serial("/dev/rfcomm0", 9600)

ser.timeout = 1

ser.setDTR(False)

ser.setRTS(False)

old\_data = 0

new\_data = 0

ser.flushInput()

ser.flush()

ser.flushOutput()

time.sleep(1)

# start out the mqtt client with port 1883

mosquitto\_client = mqtt.Client("pub\_1")

mosquitto\_client.connect("test.mosquitto.org", 1883, 60)

def saveReading(temperature):

print(temperature)

newReading = time.strftime("%Y-%m-%d %H:%M:%S") + ', ' + str(temperature) + '\n'

# publish the date on test\_leon

mosquitto\_client.publish("/lab\_task", str(temperature))

print('Saving new reading: ' + newReading)

print("Waiting for data...")

temp = []

start = False

while (True):

raw\_data = ser.readline()

try:

ser.flushInput()

new\_data = int(raw\_data)

ser.flush()

except ValueError:

pass

print (raw\_data)

saveReading(raw\_data)

if (old\_data != new\_data - 1 and old\_data != 0):

print('ERROR#####################################################')

old\_data = new\_data

time.sleep(0.01)