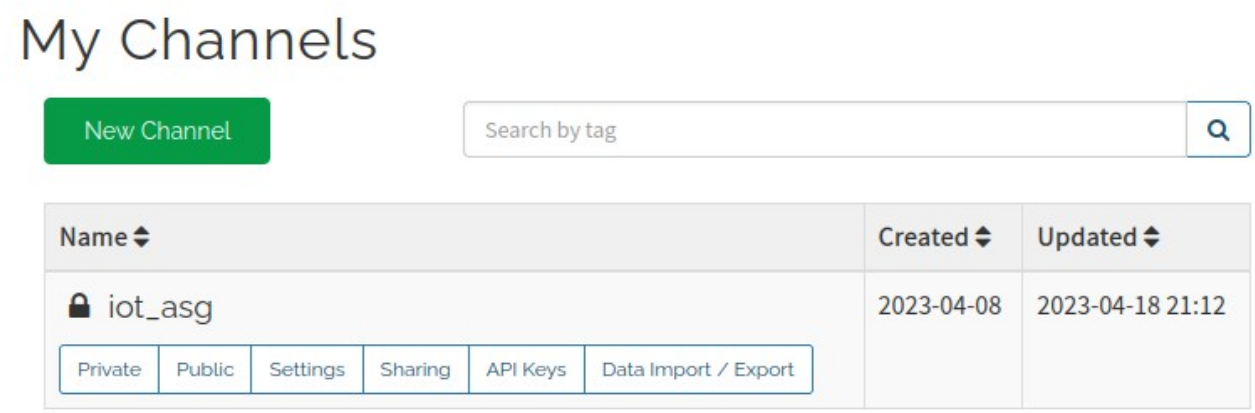
VIVEK KUMAR (19MCME18) **IOT ASSIGNMENT 2**

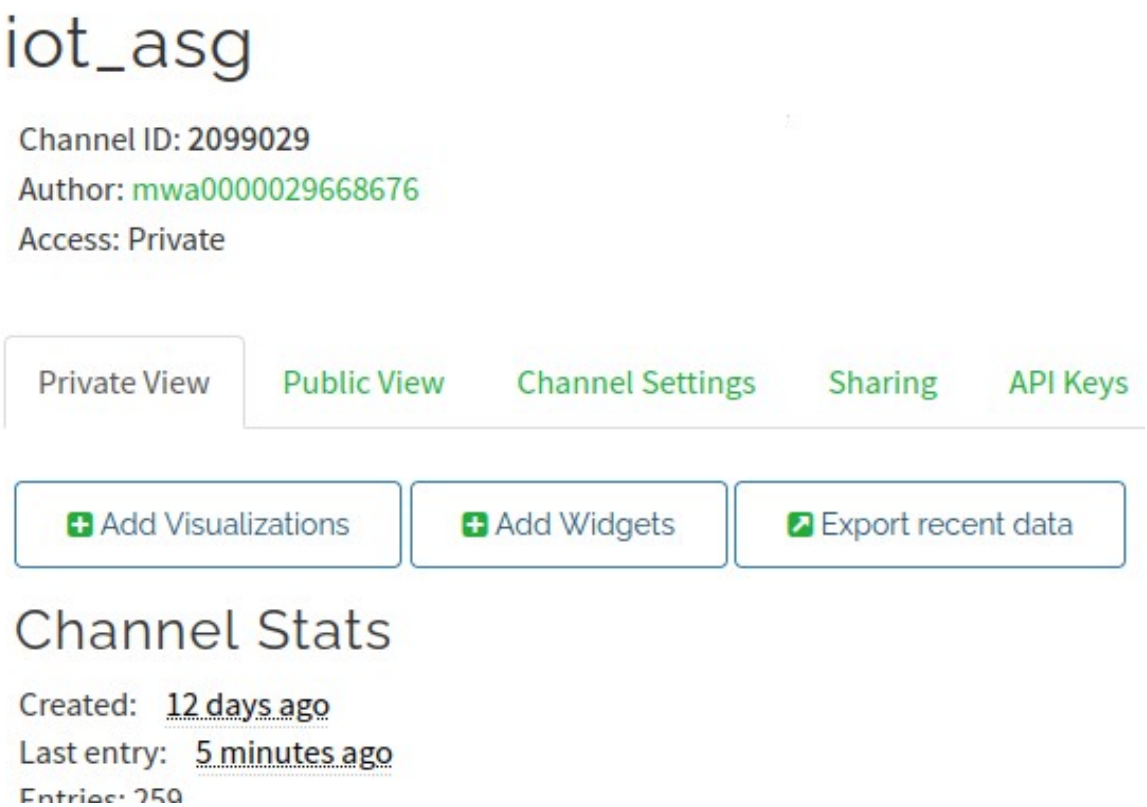
Q1: Develop a Python GUI application for Implementing the MQTT subscribe operation to the given demo MQTT publish operation using Thingspeak Cloud.

> Setup:

Creation of Channel on thingspeak:

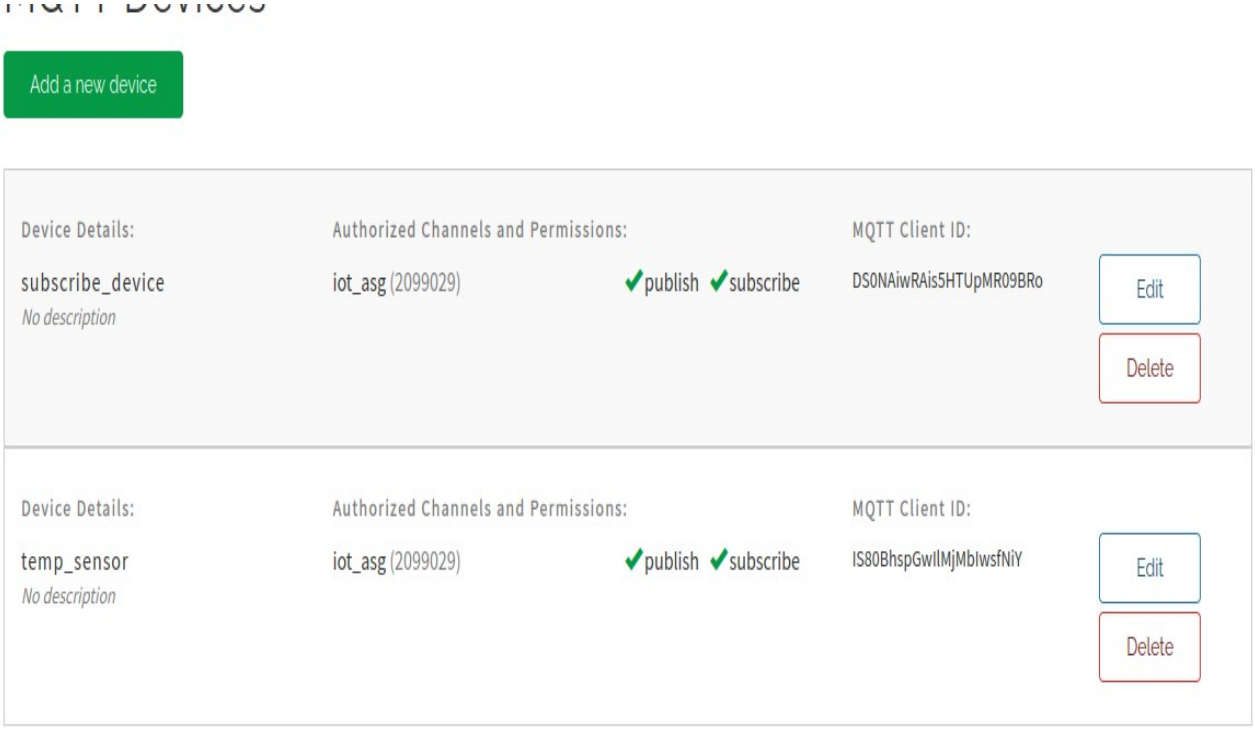


> Created a channel with ID 2099029.



VIVEK KUMAR (19MCME18) **IOT ASSIGNMENT 2**

> Created 2 mqtt device to publish and subscribe.
One with client ID `DS0NAiwRAis5HTUpMR09BRo`
and other with client ID `IS80BhspGwIlMjMbIwsfNiY`



> Permission for both publish and subscribe is given to these both devices.
For publishing, i used DS0NAiwRAis5HTUpMR09BRo client.

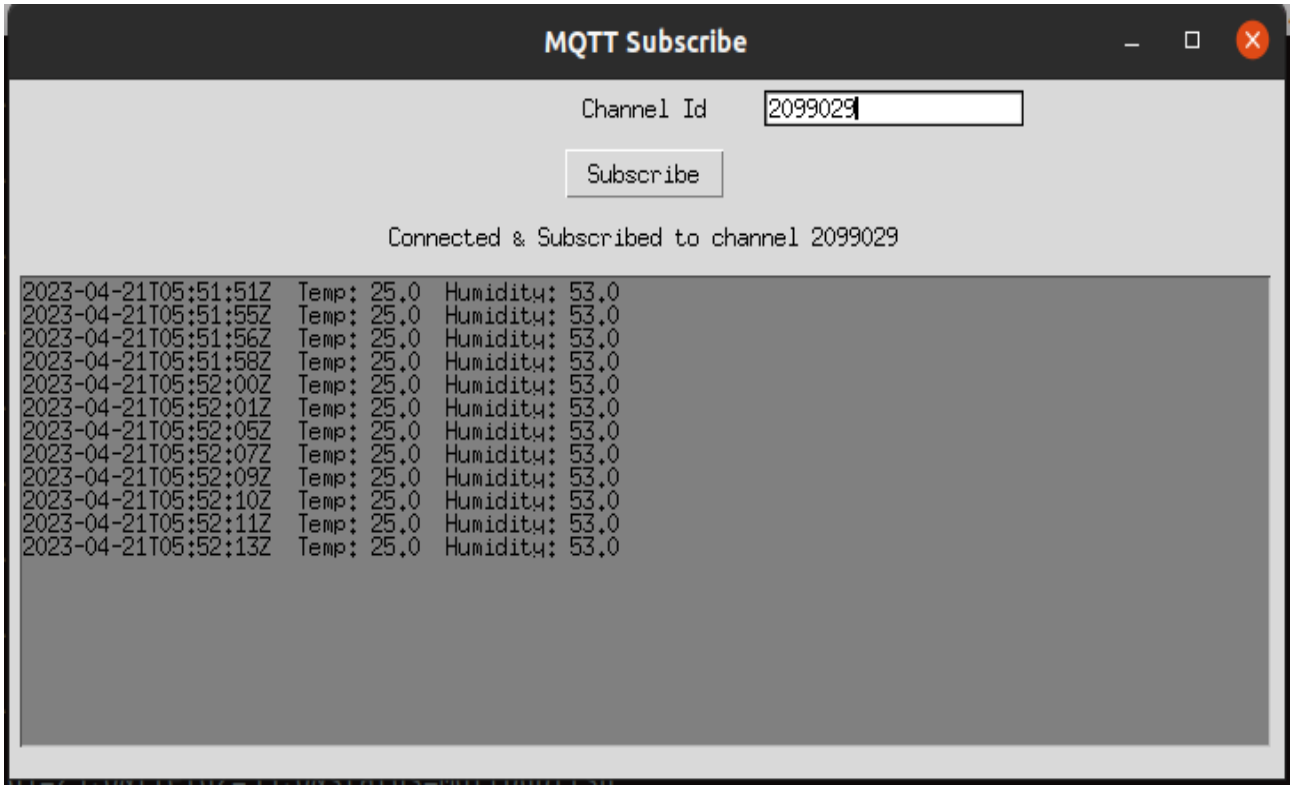
For subscription, i used IS80BhspGwIlMjMbIwsfNiY client.

Publishing Data:

```
pi@raspberrypi:~/19MCME18 $ python3 q1_pub.py
field1=25.0&field2=53.0&status=mqttpublish
(0, 1)
field1=25.0&field2=53.0&status=mqttpublish
(0, 2)
field1=25.0&field2=53.0&status=mqttpublish
(0, 3)
field1=25.0&field2=53.0&status=mqttpublish
(0, 4)
field1=25.0&field2=53.0&status=mqttpublish
(0, 5)
field1=25.0&field2=53.0&status=mqttpublish
(0, 6)
field1=25.0&field2=53.0&status=mqttpublish
(0, 7)
field1=25.0&field2=53.0&status=mqttpublish
(0, 8)
field1=25.0&field2=53.0&status=mqttpublish
(0, 9)
field1=25.0&field2=53.0&status=mqttpublish
(0, 10)
field1=25.0&field2=53.0&status=mqttpublish
```

Subscribed Data shown on gui:

```
(base) vivek@vivek:~/sem-8/IOT/asg-2$ python3 q1_sub.py
Successfully connected to MQTT broker
Subscribed! woah!
```



>Publish code :

```
import paho.mqtt.client as mqtt
import Adafruit_DHT    Import "Adafruit_DHT" could not be resolved

channel_id = "2099029"
username = "DS0NAiwRAis5HTUpMR09BRo"
password = "P+cMvDo4uJJoBipUWIoeLRSL"
pub_topic = "channels/"+channel_id+"/publish"

client = mqtt.Client(client_id=username, transport="websockets")
client.username_pw_set(username, password)

client.connect("mqtt3.thingspeak.com", 80)

def sendData():
    try:
        hum, temp = Adafruit_DHT.read_retry(11, 4)
        data = "field1="+str(temp)+"&field2="+str(hum)+"&status=mqttpublish"
        print(data)
        val = client.publish(topic=pub_topic, payload=data)
        print(val)
    except Exception as e:
            print("connection error: ", e)


if __name__ == "__main__":
    while True:
        sendData()
        time.sleep(1)
```

>First, we initialize mqtt Client and then set user password for that device and connect them to broker on port.
> It read data from sensor through Adafruit_DHT.read_retry(11, 4) function and store it in data variable and send them to publish through client.publish().
> It keeps publishing data on an interval of 1 seconds as specified in time.sleep(1).

> Subscribe code With GUI:

```
import paho.mqtt.client as mqtt
import tkinter as tk
import json

broker = "mqtt3.thingspeak.com"
port= 80
username = "IS80BhspGwIlMjMbIwsfNiY"
clientId = "IS80BhspGwIlMjMbIwsfNiY"
password = "64zaWOjeHbKdc4PAC0+RlLOw"

def on_connect(client, userdata, flags, rc):
    if rc==0:
        print("Successfully connected to MQTT broker")
        status_label.config(text=f"Successfully connected to MQTT broker")
    else:
        print("Failed to connect, return code %d", rc)
        status_label.config(text=f"Failed to connect to MQTT broker")


def on_subscribe(client, userdata, mid, granted_qos):
    channelId = channelId_entry.get()
    print("Subscribed! woah!")
    status_label.config(text=f"Connected & Subscribed to channel {channelId}")

def on_message(client, userdata, message):
    msg = json.loads(message.payload.decode('ascii'))
    timestamp = msg['created_at']
    temperature = msg['field1']
    humidity = msg['field2']

    sensor_data_text.config(state=tk.NORMAL)
    sensor_data_text.insert(tk.END, f"{timestamp}  Temp: {temperature}  Humidity: {humidity}\n")
    sensor_data_text.config(state=tk.DISABLED)
```

> Stored all the credentials in variables which will be used for connecting client.
> Defined three function, on_connect will be executed when connection response was sent by broker, on_subscribe will be executed when broker responds to subscription request, on_message will be executed when client receives message for their subscribed topic.
> In on_message, I am updating sensor_data_text so the message will be shown in box as part of GUI.

```
client = mqtt.Client(client_id=clientId, transport="websockets")
client.username_pw_set(username, password)
client.on_connect = on_connect
client.on_subscribe = on_subscribe
client.on_message = on_message

def subscribe():
    channelId = channelId_entry.get()
    client.connect(broker, port)

    sub_topic = "channels/"+str(channelId)+"/subscribe"
    client.loop_start()

    client.subscribe(topic=sub_topic, qos = 0)

root = tk.Tk()
root.title("MQTT Subscribe")
root.minsize(300, 300)

channelId_entry = tk.Label(root, text="Channel Id")
channelId_entry.grid(row=1, column=0, columnspan=2,padx=5, pady=5)

channelId_entry = tk.Entry(root)
channelId_entry.grid(row=1, column=1,columnspan=1,  padx=5, pady=5,)


connect_button = tk.Button(root, text="Subscribe", command=subscribe)
connect_button.grid(row=3, column=0, columnspan=2, padx=5, pady=5)

status_label = tk.Label(root, text="")
status_label.grid(row=4, column=0, columnspan=2, padx=5, pady=5)

sensor_data_text = tk.Text(root, height=20, width=100, background='grey')
sensor_data_text.grid(row=5, column=0, padx=5, pady=5, columnspan=2)
sensor_data_text.config(state=tk.DISABLED)

root.mainloop()
```

> Here, tkinter gui is implemented where we have to write channel name and click on subscribe button to get data.
> After clicking subscribe button, it connects to client and subscribe topic on the channel Id given(if allowed).
> Then if data is being published, we get data in text box as updated by on_message function.

Q2: Develop a Restful API application to collect, store the DHT sensor data and run the application on a RPi.

> Code:

```python
from flask import Flask, render_template, request, jsonify
import requests
from markupsafe import escape
import datetime
import Adafruit_DHT    Import "Adafruit_DHT" could not be resolve

app = Flask(__name__)

now = datetime.datetime.now()
timestring = now.strftime("%Y-%m-%d %H:%M")
global keepPublishing
keepPublishing = True

@app.route("/")
def main():
    templateData = {
    "time": now,
    "title": "Rest_API",
    "isPublishing": False
    }
    return render_template("index.html", **templateData)

@app.route("/sensorData")
def sensorData():
    hum, temp = Adafruit_DHT.read_retry(11, 4)
    if hum is not None and temp is not None:
        return jsonify({'temperature': temp,'humidity':hum})
    else:
        return jsonify({'error': 'Sensor not working.'})
```

> Importing important modules such as Flask. I have made REST API application in flask and then in homepage i.e app.route(“/”) I am rendering my html page with provinding some content.

> In route(“/sensorData”), i am reading sensordata and returning as json for just api purpose.

```python
@app.route("/publish", methods=['POST'])
def publish():
    writeApi = request.form.get("writeApi")
    hum, temp = Adafruit_DHT.read_retry(11, 4)
    data = "field1="+str(temp)+"&field2="+str(hum)+"&status=mqttpublish"
    pub_data = "Temperature = {}, Humidity = {}".format(temp,hum)
    if hum is not None and temp is not None:
        requests.get("https://api.thingspeak.com/update?api_key="+writeApi+"&"+data)
        templateData = {
            "time": now,
            "title": "Rest_API",
            "isPublishing": True,
            "writeApi": writeApi,
            "data": pub_data
        }
    else:
        templateData = {
            "time": now,
            "title": "Rest_API",
            "isPublishing": False
        }
    return render_template("index.html", **templateData)

if __name__ == '__main__':
    app.run(debug=True)
```

> In publish route, i am reading data and making an REST API request to publish the data if valid write key is given.

HTML Template:

```
<!DOCTYPE html>
    <head>
        <title>{{title}}</title>
    </head>
    <body>
        <h1>Start Publishing Data</h1>
        <h2>Date and time on server: {{time}}</h2>
        <div>
            <form action="/publish" method="POST">
                <input type="text" name="writeApi" placeholder="Write API Key" />
                <button type="submit">Send Data</button>
            </form>
        </div>
        <div>
            {% if isPublishing == True %}
                <h2>Sensor Data published for writeApi {{writeApi}}</h2>
                <p>Data: {{data}}</p>
            {% else %}
                <h2>No Data Publishing</h2>
            {% endif %}
        </div>
    </body>
```

Outputs:
Terminal when app is running:

```
pi@raspberrypi:~/19MCME18 $ python3 q2.py
 * Serving Flask app "q2" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 124-393-664
127.0.0.1 - - [21/Apr/2023 16:44:32] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Apr/2023 16:44:33] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [21/Apr/2023 16:46:03] "POST /publish HTTP/1.1" 200 -
127.0.0.1 - - [21/Apr/2023 16:46:10] "POST /publish HTTP/1.1" 200 -
127.0.0.1 - - [21/Apr/2023 16:46:19] "POST /publish HTTP/1.1" 200 -
```

Application First Look:

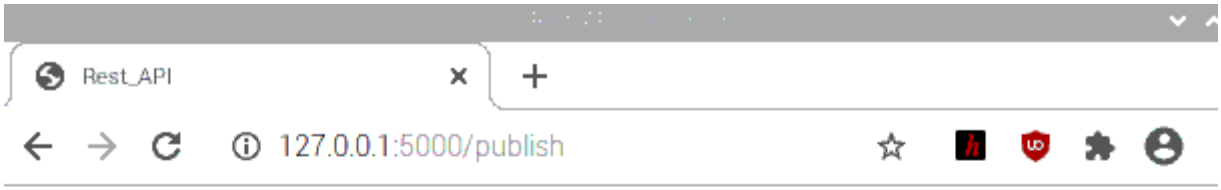
Start Publishing Data

Date and time on server: 2023-04-21 16:50:48.655245

Write API Key | Send Data

No Data Publishing

>App after writing API key and clicking send Data:



Start Publishing Data

Date and time on server: 2023-04-21 16:43:42.299983

Write API Key | Send Data

Sensor Data published for writeApi PBHIXPS6NETXO

Data: Temperature = 24.0, Humidity = 49.0

Q3: Develop a Python GUI application to collect and store sensor data locally in the MySQL DB (LAMP stack installed in the Raspberry Pi). Connect to Raspberry Pi+DHT11 Sensor.

> Code:

```
import time
import Adafruit_DHT    Import "Adafruit_DHT" could not be resolved
import mysql.connector    Import "mysql.connector" could not be resolved
import tkinter as tk
import datetime

myDb = mysql.connector.connect(
    host="127.0.0.1", user="19mcme18", password="passwd@123", database="19mcme18")
sql = "CREATE DATABASE IF NOT EXISTS 19mcme18"
myCursor = myDb.cursor()
myCursor.execute(sql)
tableSql = "CREATE TABLE IF NOT EXISTS data1 (timestamp timestamp, temperature double, humidity double)"
myCursor.execute(tableSql)

def readAndStore():
    def updateTable(temperature, humidity):
        insertSql = "INSERT INTO data1(temperature, humidity) VALUES ("+ str(
            temperature) + ","+str(humidity) + ")"
        myCursor.execute(insertSql)
        myDb.commit()

    try:
        hum, temp = Adafruit_DHT.read_retry(11, 4)
        data = "field1="+str(temp)+"&field2="+str(hum)
        print(data)
        now = datetime.datetime.now()
        if temp is not None and hum is not None:
            updateTable(format(temp, '.2f'), format(hum, '.2f'))
        else:
            updateTable(0.0, 0.0)

        sensor_data_text.config(state=tk.NORMAL)
        sensor_data_text.insert(tk.END, f"DHT11, {now}, Temperature: {temp}°C, Humidity: {hum}\n")
        sensor_data_text.config(state=tk.DISABLED)
```

>First, I connected to mysql database with credentials and create database and table if they don’t exists.

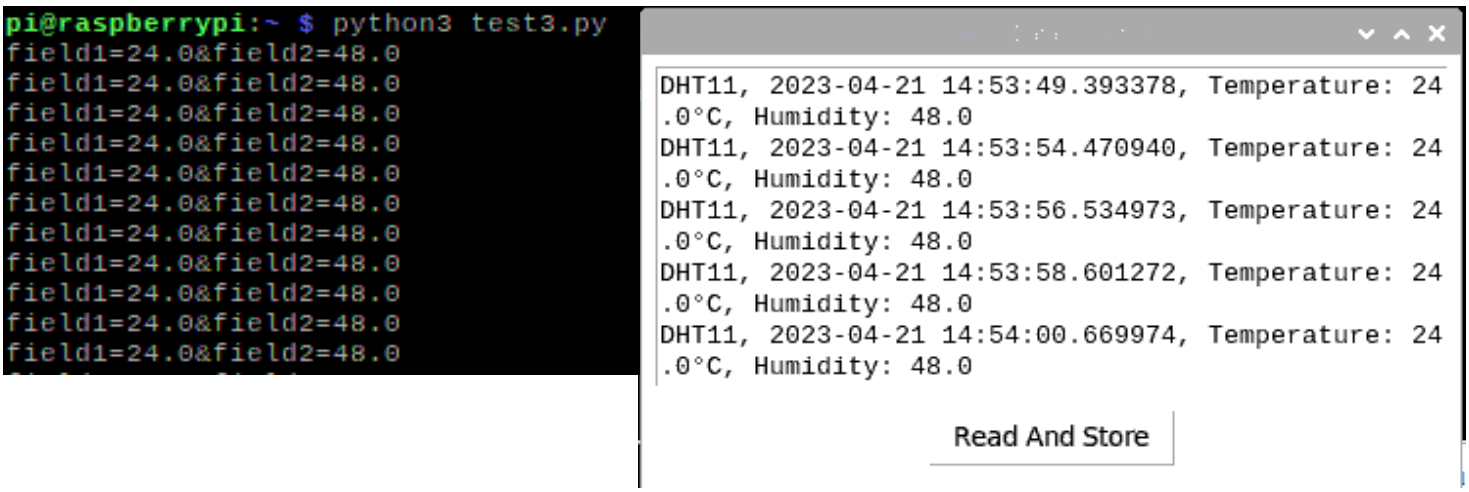
> Made a function readAndStore(), which reads data from DHT sensor, store in hum and temp variable and read current time, store in now variable and send to updateTable() function and update the data in gui textbox.

> In updateTable() function, we write a query to insert the temperature and humidity value in database and execute the query.

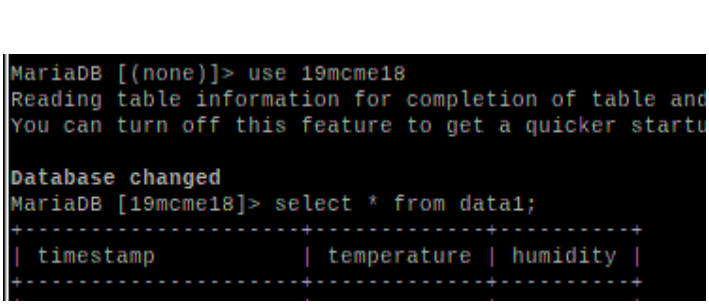
```
root = tk.Tk()
root.title("Sensor Data Collection")
root.minsize(300,300)

sensor_data_text = tk.Text(root, height=10, width=50)
sensor_data_text.grid(row=0, column=0, padx=5, pady=5, columnspan=2)
sensor_data_text.config(state=tk.DISABLED)

refresh_button = tk.Button(root, text="Read And Store", command=readAndStore)
refresh_button.grid(row=1, column=0, columnspan=2, padx=5, pady=5)

root.mainloop()
```

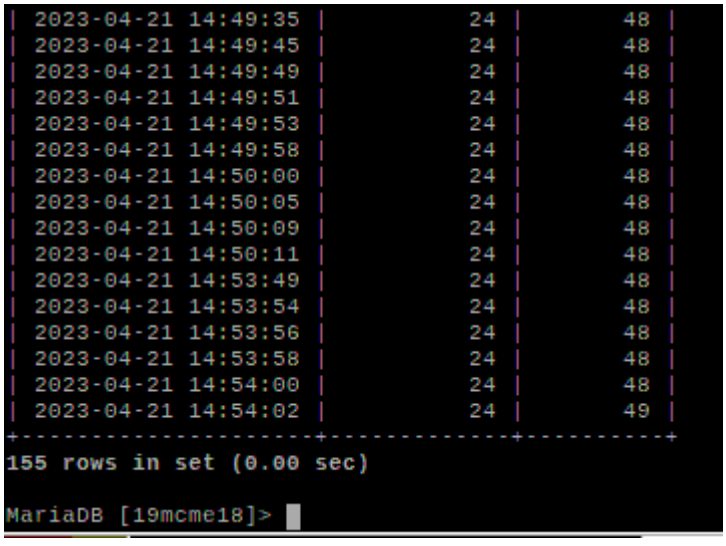
> This is python-gui built with the help of tkinter.

Program Running and output in terminal and GUI:



> Database entries:





Q4: Develop a program to perform image classification task by capturing the images using RPi using Tensorflowlite.

> Code:

```
import picamera    Import "picamera" could not be resolved
from time import sleep
import numpy as np
from tflite_runtime.interpreter import Interpreter    Import "tflite_runti
from PIL import Image

interpreter = Interpreter(model_path='mobilenet_v1_1.0_224_quant.tflite')
interpreter.allocate_tensors()
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

# camera = picamera.PiCamera()
# camera.start_preview()
# sleep(5)
# camera.capture("image.jpg")
# camera.stop_preview()

for i in range(0, 34):
    image = Image.open('img{}.jpg'.format(i))
    image = image.resize((224, 224))
    image = np.array(image)
    image = np.expand_dims(image, axis=0)
    image = image.astype(np.uint8)

    interpreter.set_tensor(input_details[0]['index'], image)
    interpreter.invoke()
    output = interpreter.get_tensor(output_details[0]['index'])
    output = np.squeeze(output)
    predicted_class_index = np.argmax(output)
    confidence = output[predicted_class_index]
```

> I have imported some important modules, loading the already downloaded model of mobilenet in my local directory and getting input and output details from interpreter.

> Commented code is to capture image but for testing i am using already captured image from picamera.

```
predicted_class_index = np.argmax(output)
confidence = output[predicted_class_index]

with open('labels.txt', 'r') as f:
    labels = [line.strip() for line in f.readlines()]

predicted_class = labels[predicted_class_index]
print("For image name: img{}".format(i))
print('Predicted Class:', predicted_class)
print('Confidence:', confidence)
```

>Then i am reading labels.text, already downloaded with model and predicting class and confidence for each image.

Output:

```
pi@raspberrypi:~/19MCME18 $ python3 q4.py
For image name: img0
Predicted Class: bubble
Confidence: 56
For image name: img1
Predicted Class: pencil sharpener
Confidence: 134
For image name: img2
Predicted Class: iPod
Confidence: 127
For image name: img3
Predicted Class: ballpoint
Confidence: 65
For image name: img4
Predicted Class: mouse
Confidence: 255
For image name: img5
Predicted Class: digital clock
Confidence: 82
For image name: img6
Predicted Class: screw
Confidence: 40
For image name: img7
Predicted Class: ballpoint
Confidence: 68
For image name: img8
Predicted Class: cassette
Confidence: 60
For image name: img9
Predicted Class: pool table
Confidence: 100
For image name: img10
Predicted Class: hand blower
Confidence: 160
For image name: img11
Predicted Class: conch
Confidence: 34
```

```
For image name: img11
Predicted Class: conch
Confidence: 34
For image name: img12
Predicted Class: bathing cap
Confidence: 80
For image name: img13
Predicted Class: stethoscope
Confidence: 123
For image name: img14
Predicted Class: laptop
Confidence: 146
For image name: img15
Predicted Class: cellular telephone
Confidence: 110
For image name: img16
Predicted Class: pool table
Confidence: 20
For image name: img17
Predicted Class: pool table
Confidence: 21
For image name: img18
Predicted Class: screwdriver
Confidence: 214
For image name: img19
Predicted Class: ballpoint
Confidence: 82
For image name: img20
Predicted Class: loupe
Confidence: 76
```

```
For image name: img22
Predicted Class: iPod
Confidence: 180
For image name: img23
Predicted Class: racket
Confidence: 114
For image name: img24
Predicted Class: racket
Confidence: 72
For image name: img25
Predicted Class: fiddler crab
Confidence: 25
For image name: img26
Predicted Class: can opener
Confidence: 33
For image name: img27
Predicted Class: modem
Confidence: 74
For image name: img28
Predicted Class: steel drum
Confidence: 18
For image name: img29
Predicted Class: notebook
Confidence: 34
For image name: img30
Predicted Class: paddlewheel
Confidence: 39
For image name: img31
Predicted Class: pencil box
Confidence: 79
For image name: img32
Predicted Class: stethoscope
Confidence: 130
For image name: img33
Predicted Class: swimming trunks
Confidence: 69
```

Downloaded Model, labels.txt and images:

```
pi@raspberrypi:~/19MCME18 $ ls
image.jpg   img1.jpg   img30.jpg  labels.txt
img0.jpg    img20.jpg  img31.jpg  __MACOSX
img10.jpg   img21.jpg  img32.jpg  mobilenet_v1_1.0_224_quant_and_labels.zip
img11.jpg   img22.jpg  img33.jpg  mobilenet_v1_1.0_224_quant.tflite
img12.jpg   img23.jpg  img34.jpg  mobilenet_v1_1.0_224.tgz
img13.jpg   img24.jpg  img3.jpg   q3_1.py
img14.jpg   img25.jpg  img4.jpg   q3.py
img15.jpg   img26.jpg  img5.jpg   q4.py
img16.jpg   img27.jpg  img6.jpg   test2.py
img17.jpg   img28.jpg  img7.jpg   test.py
img18.jpg   img29.jpg  img8.jpg
img19.jpg   img2.jpg   img9.jpg
```