

CPE 4040: Data Collection and Analysis, Spring 2024

# **Laboratory Report #4**

## **Raspberry Pi Weather Station Application on ThingSpeak**

Team Members: Anindita Deb & Damisi Kayode

Electrical and Computer Engineering

Kennesaw State University

Faculty: Dr. Jeffrey L Yiin

Date of Lab Session: January 17, 2024

## I. Objective

1. Learn how to interface and read data from a temperature sensor.
2. Learn how to communicate with the popular ThingSpeak cloud platform.
3. Understand how to make HTTP client-server connection with Raspberry Pi.
4. Understand how to create a cloud server application with Raspberry Pi.
5. Learn how to control output of Raspberry Pi remotely from the cloud service

## II. Material List

Hardware:

- Temperature Sensor (we have the DHT-11)
- Breadboard, LED, and resistor (we used a 10k resistor)
- Power Cord

Software:

- Remote Desktop Connection
- ThingSpeak cloud Platform

## III. Lab Procedures and Results

1. We plug the power adapter to the Raspberry Pi and power up the system. We open Remote Desktop Connection (or SSH connection) on our laptop and connect to Raspberry Pi.

### Section 1: Connect and Test the Temperature Sensor

2. We open a terminal window and type `pip3 list` to see which Python packages are installed in our device.
3. We make sure the following packages are installed already: *urllib3*, *python-http-client*, *requests*. **Question:** What are those three packages intended for? Do Google searches and give a brief description for each package. The three packages – urllib3, python-http-client,

and requests – are essential for creating an HTTP client-server connection with the Raspberry Pi.

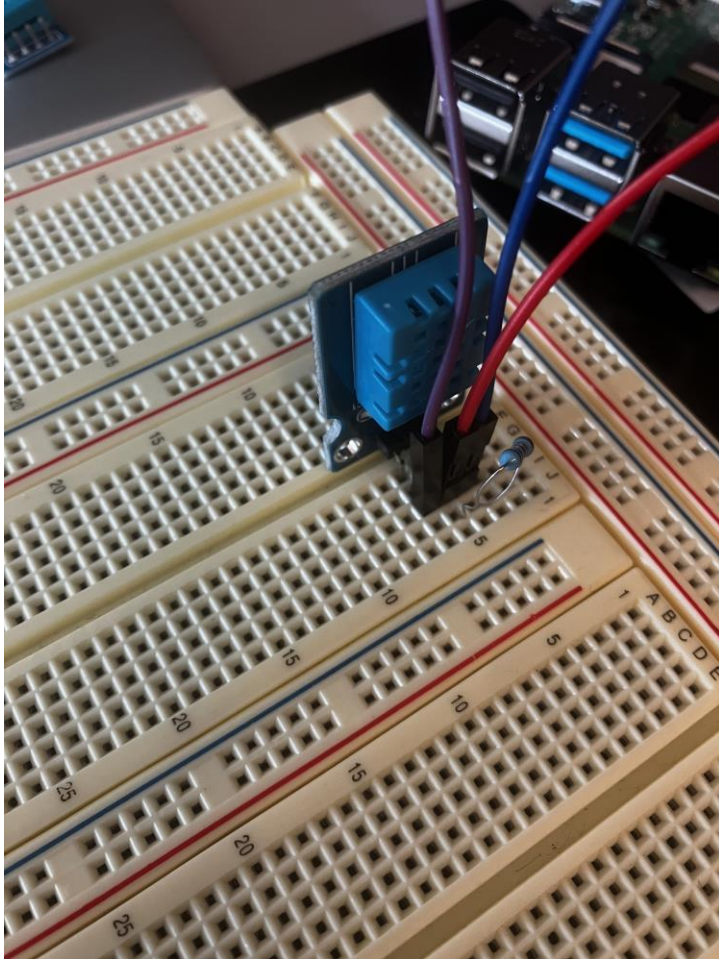
urllib3: A powerful HTTP client for Python with features such as connection pooling, thread safety, and support for various authentication schemes.

python-http-client: A simple HTTP client library for Python that facilitates making HTTP requests.

requests: A widely-used Python library for sending HTTP requests and handling responses. rewrite this a bit simpler.

Type `pip3 install <package name>` to install the packages. (if they aren't already installed)

4. We connect the temperature sensor to Raspberry Pi with the below connection diagram. The resistor we use is  $10k\Omega$  placed between Vcc and DATA pins of the sensor. This is a picture of our hardware setup.



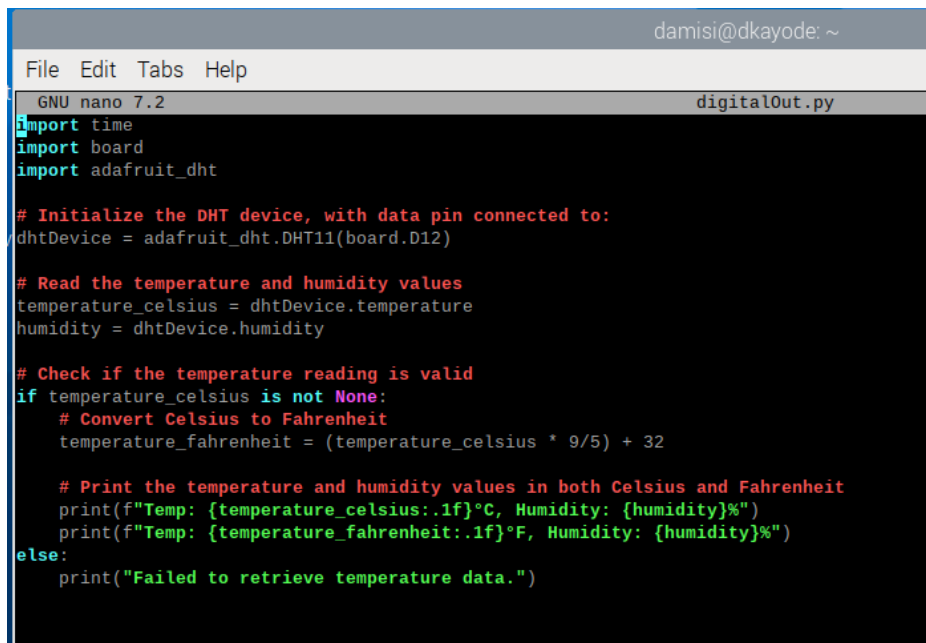
5. We install the DHT library which contains the device driver for DHT-11 or DHT-22 :

```
pip3 install adafruit-circuitpython-dht
```

```
sudo apt-get install libgpod2
```

**Question:** What is the library libgpod2 for? libgpod2: This library provides C library and tools for interacting with the GPIO character device.

6. We create a file (*digitalOut.py*) with the python code below to get sensor readings from our DHT-11.



```
damisi@dkayode: ~
File Edit Tabs Help
GNU nano 7.2 digitalOut.py
import time
import board
import adafruit_dht

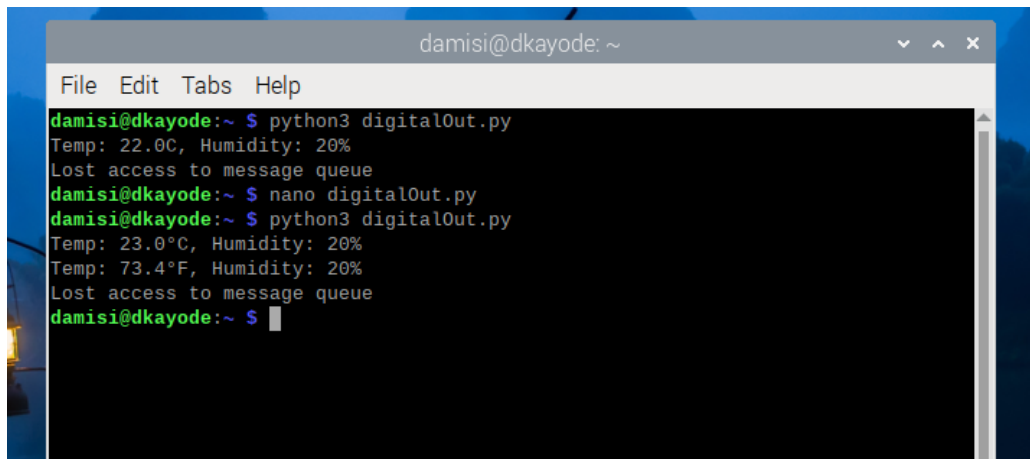
# Initialize the DHT device, with data pin connected to:
dhtDevice = adafruit_dht.DHT11(board.D12)

# Read the temperature and humidity values
temperature_celsius = dhtDevice.temperature
humidity = dhtDevice.humidity

# Check if the temperature reading is valid
if temperature_celsius is not None:
    # Convert Celsius to Fahrenheit
    temperature_fahrenheit = (temperature_celsius * 9/5) + 32

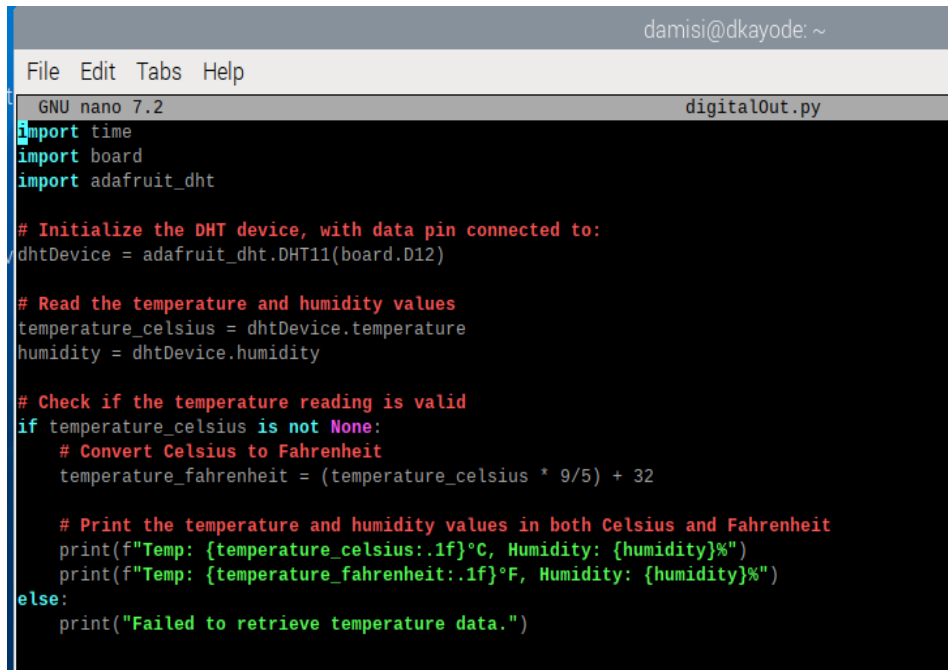
    # Print the temperature and humidity values in both Celsius and Fahrenheit
    print(f"Temp: {temperature_celsius:.1f}°C, Humidity: {humidity}%")
    print(f"Temp: {temperature_fahrenheit:.1f}°F, Humidity: {humidity}%")
else:
    print("Failed to retrieve temperature data.")
```

7. We check whether the temperature and humidity readings are reasonable.



```
damisi@dkayode: ~
File Edit Tabs Help
damisi@dkayode:~ $ python3 digitalOut.py
Temp: 22.0C, Humidity: 20%
Lost access to message queue
damisi@dkayode:~ $ nano digitalOut.py
damisi@dkayode:~ $ python3 digitalOut.py
Temp: 23.0°C, Humidity: 20%
Temp: 73.4°F, Humidity: 20%
Lost access to message queue
damisi@dkayode:~ $
```

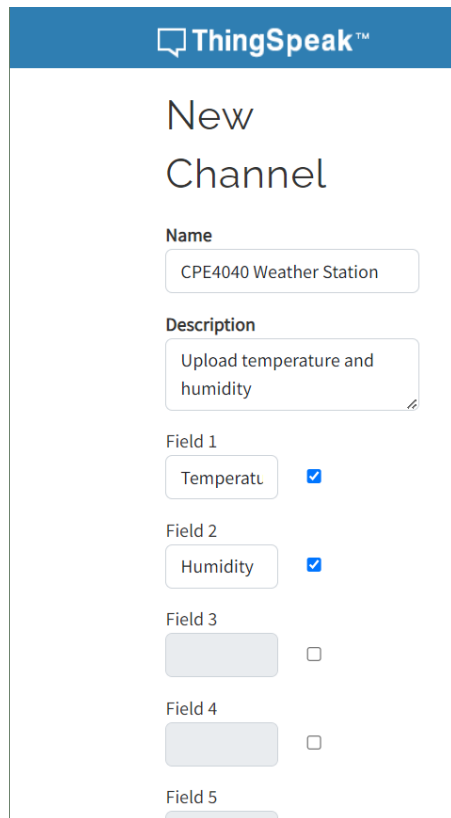
We add a small piece of code to convert temperature values from C (Celsius) to F (Fahrenheit).



```
damisi@dkayode: ~  
File Edit Tabs Help  
GNU nano 7.2 digitalOut.py  
import time  
import board  
import adafruit_dht  
  
# Initialize the DHT device, with data pin connected to:  
dhtDevice = adafruit_dht.DHT11(board.D12)  
  
# Read the temperature and humidity values  
temperature_celsius = dhtDevice.temperature  
humidity = dhtDevice.humidity  
  
# Check if the temperature reading is valid  
if temperature_celsius is not None:  
    # Convert Celsius to Fahrenheit  
    temperature_fahrenheit = (temperature_celsius * 9/5) + 32  
  
    # Print the temperature and humidity values in both Celsius and Fahrenheit  
    print(f"Temp: {temperature_celsius:.1f}°C, Humidity: {humidity}%")  
    print(f"Temp: {temperature_fahrenheit:.1f}°F, Humidity: {humidity}%")  
else:  
    print("Failed to retrieve temperature data.")
```

## Section 2: Create a ThingSpeak Channel for the MQTT Device

8. We sign up for a free ThingSpeak account using your KSU email.
9. In order to upload data to the ThinkSpeak Cloud, we create a **Channel** first. We set it up as shown in the figure below.



**ThingSpeak™**

## New Channel

**Name**  
CPE4040 Weather Station

**Description**  
Upload temperature and humidity

**Field 1**  
Temperatu ☒

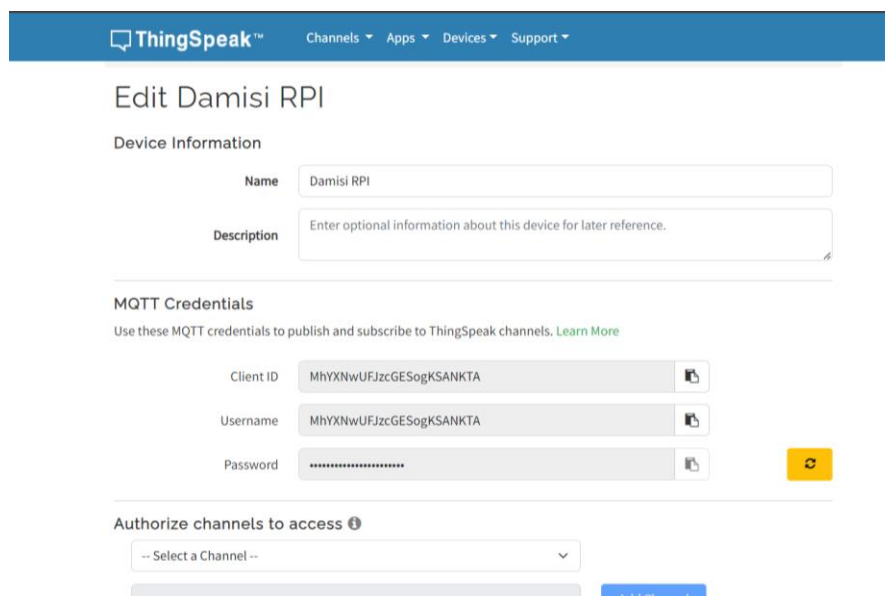
**Field 2**  
Humidity ☒

**Field 3** ☐

**Field 4** ☐

**Field 5**

10. In the “Devices” in the menu bar we create an MQTT device (our Raspberry Pi). We provide the device a name, add the channel that we just created, then click “Add Channel”. The MQTT credentials of this device will show up (Client ID, Username and Password). We save them in a separate text file as we will use them in Step 13.



**ThingSpeak™** Channels Apps Devices Support

## Edit Damisi RPI

**Device Information**

**Name** Damisi RPI

**Description** Enter optional information about this device for later reference.

**MQTT Credentials**  
Use these MQTT credentials to publish and subscribe to ThingSpeak channels. [Learn More](#)

**Client ID** MhYXNwUFJzcGESogKSANKTA

**Username** MhYXNwUFJzcGESogKSANKTA

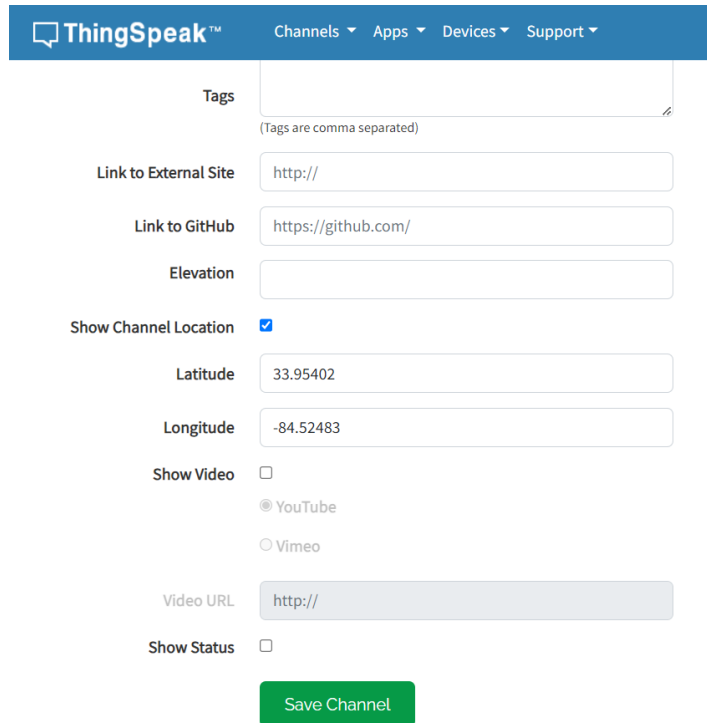
**Password** \*\*\*\*\*

**Authorize channels to access**

-- Select a Channel --

[Add Channel](#)

11. In "Channel Settings", scroll down and check the box "Show Channel Location". We find the coordinate points of our location from Google Maps and paste it into the corresponding boxes that are now available under Channel Settings. When done, we save the channel, and are redirected to the main page of the newly created channel. We write down "Channel ID", which will be used shortly.



The screenshot shows the ThingSpeak Channel Settings page. At the top is a blue navigation bar with the ThingSpeak logo and links for Channels, Apps, Devices, and Support. Below the navigation bar is a form with several fields: Tags (with a note that tags are comma separated), Link to External Site (pre-filled with http://), Link to GitHub (pre-filled with https://github.com/), Elevation, Show Channel Location (checked), Latitude (pre-filled with 33.95402), Longitude (pre-filled with -84.52483), Show Video (unchecked), Video URL (pre-filled with http://), and Show Status (unchecked). At the bottom of the form is a green "Save Channel" button.

12. In "API Keys" in the manual bar and we write down the key in the "Write API Key" section.
13. We open the reference Python program, *publish.py*, and modify the code as instructed. When running the code, we see sensor data posting to our ThingSpeak channel in real time.



```

GNU nano 7.2 publish.py
import paho.mqtt.publish as publish
import time
import board
import adafruit_dht

interval = 30 #Time between readings

##Start of credentials#####

#ThingSpeak Channel ID (numeric id, not the name)
channel_ID = "2440843" # your channel ID

# Your MQTT credentials for the Raspberry Pi
client_ID = "MhYXNwUFJzcGESogKSANKTA" # MQTT device ID
username = "MhYXNwUFJzcGESogKSANKTA" # MQTT device username
password = "u78ET/wTha5VXkr1UTEZoecI" # MQTT device password

##End of your credentials#####

#For DHT-20, this won't be used
dhtDevice = adafruit_dht.DHT11(board.D12)

#hostname of the ThingSpeak MQTT broker
host = "mqtt3.thingspeak.com"

# Define the connection type as websockets and use port 80
t_transport = "websockets"
t_port = 80

# create a topic string to publish to the ThingSpeak channel

# create a topic string to publish to the ThingSpeak channel
topic = "channels/" + channel_ID + "/publish"

while True:
    # Read Temperature and Humidity Values
    # If using DHT-20, this will be replaced by driver code
    try:
        temperature_c = dhtDevice.temperature
        humidity = dhtDevice.humidity

    except Exception as e:
        print(e)

    ##Insert code to convert temperature_c to Fahrenheit
    temperature = (temperature_c * 9/5) + 32

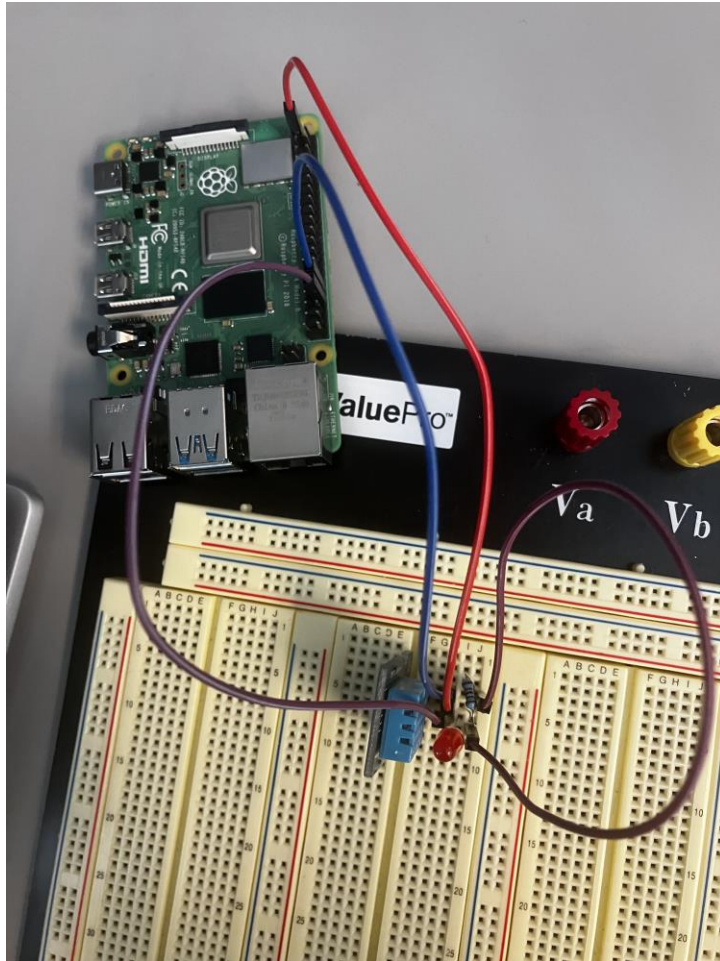
    payload = f"field1={temperature}&field2={humidity}"

# Publish Sensor Values
try:
    publish.single(topic, payload, hostname=host, transport=t_transport, port=t_port, client_id=client_ID, auth={'username':username, 'password':password})
    print("Temp: {:.1f} F, Humidity: {}% ".format(temperature, humidity))
    time.sleep(interval)
except KeyboardInterrupt:
    print("Connection ended!")
    break
except Exception as e:
    print(e)
    time.sleep(5)

```

**Note:** For channel ID, username and password, you will use the ones obtained in Step 10. In the *while* loop, include the modification that you did in Step 7 to convert the temperature unit to Fahrenheit.


14. Connect an LED to GPIO12 of Raspberry Pi with a serial resistor (the same thing you did in Lab 3).



### Section 3: Sending Alert from Your Weather Station

When the temperature exceeds 75°F, an alert will be sent to Raspberry Pi and the LED will be turned on. This will be achieved using three ThingSpeak applications, as shown in the flowchart below.

15. To generate an action when the temperature exceeds 75°F, we will use the **TalkBack** app. Go to Apps on the ThingSpeak home page and select TalkBack. Select the “New TalkBack” button, then enter a name to select our channel in the “Log to Channel” option. Save and exit.



[Apps](#) / [TalkBack](#) / [Damisi rpi](#) / [Edit](#)

Name

Damisi rpi

API Key

FMV7PZV40GWWKL7K

Log to Channel

CPE4040 Weather Station (2440843) ▼

Commands

[Add a new command](#)

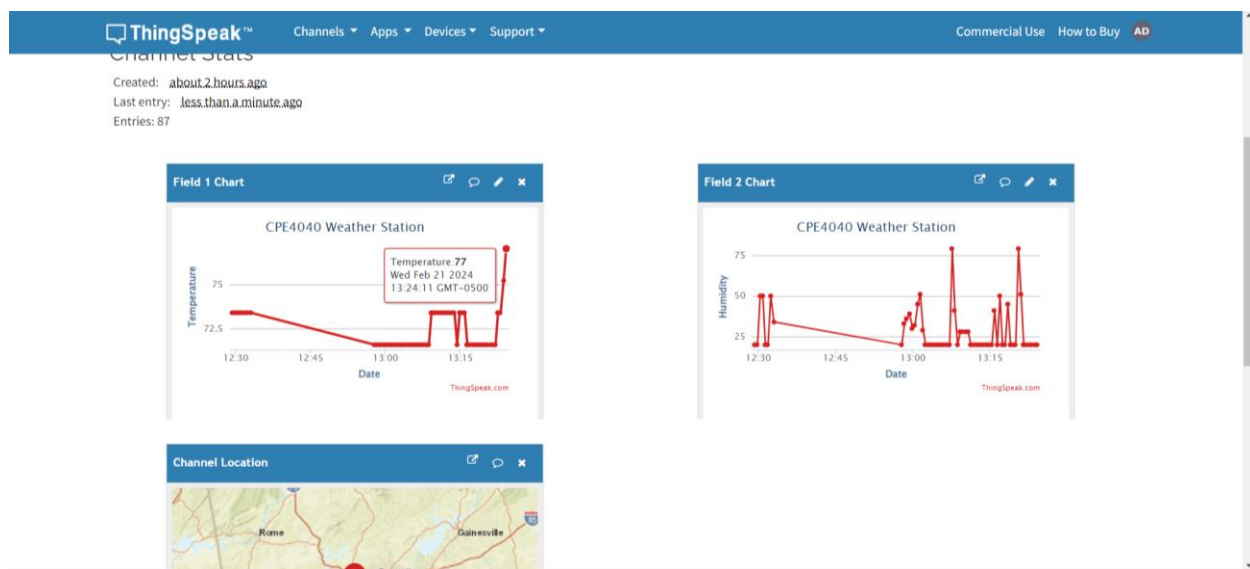
Save TalkBack

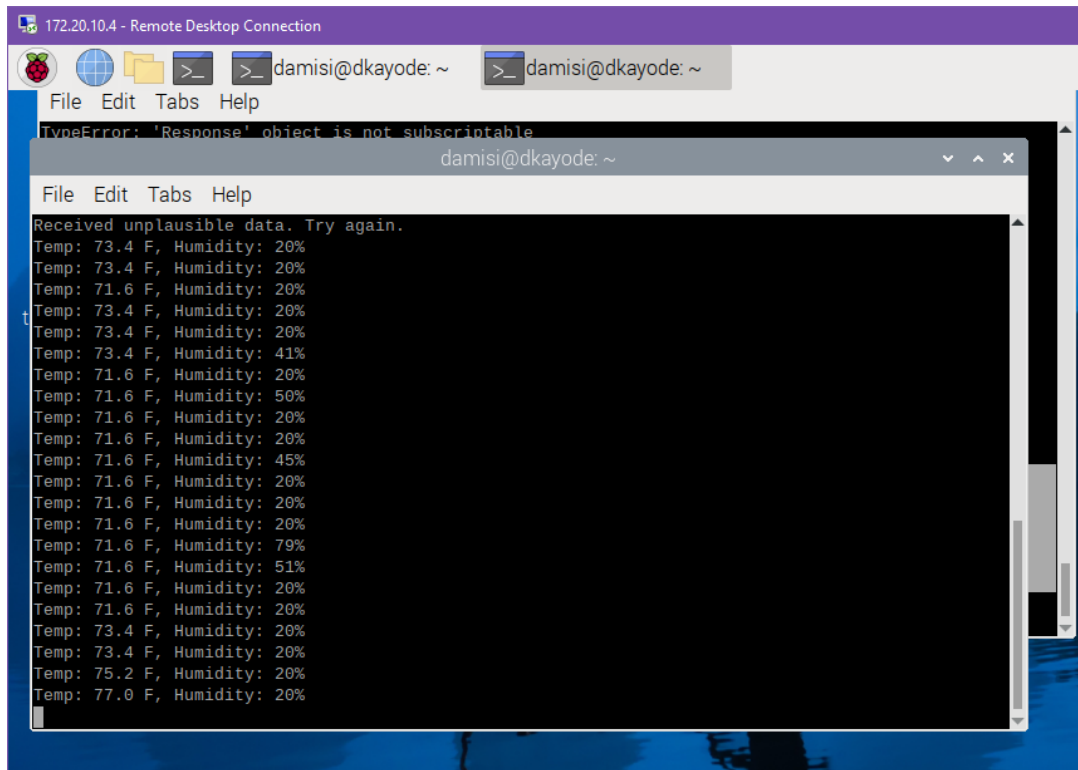
Want to delete this TalkBack?

Delete TalkBack

TalkBack is where Raspberry Pi periodically accesses and polls commands. To generate commands for the result of events, we use the **React** app in ThingSpeak.

16. We will create two new Reacts: one for the temperature exceeding above 75°F and one for the temperature dropping below 75°F. Both are similarly defined as in the figure below.





The screenshot shows a Remote Desktop Connection window titled "172.20.10.4 - Remote Desktop Connection". Inside the window, there is a terminal window with a menu bar (File, Edit, Tabs, Help) and a title bar (damisi@dkayode: ~). The terminal displays a list of temperature and humidity readings. The first line shows an error: "TypeError: 'Response' object is not subscriptable". The subsequent lines show a series of "Temp: X.X F, Humidity: Y%" readings. The temperatures fluctuate between 71.6 F and 77.0 F, and the humidity fluctuates between 20% and 79%.

```
TypeError: 'Response' object is not subscriptable
Received unplausible data. Try again.
Temp: 73.4 F, Humidity: 20%
Temp: 73.4 F, Humidity: 20%
Temp: 71.6 F, Humidity: 20%
Temp: 73.4 F, Humidity: 20%
Temp: 73.4 F, Humidity: 20%
Temp: 73.4 F, Humidity: 41%
Temp: 71.6 F, Humidity: 20%
Temp: 71.6 F, Humidity: 50%
Temp: 71.6 F, Humidity: 20%
Temp: 71.6 F, Humidity: 20%
Temp: 71.6 F, Humidity: 45%
Temp: 71.6 F, Humidity: 20%
Temp: 71.6 F, Humidity: 20%
Temp: 71.6 F, Humidity: 79%
Temp: 71.6 F, Humidity: 51%
Temp: 71.6 F, Humidity: 20%
Temp: 71.6 F, Humidity: 20%
Temp: 73.4 F, Humidity: 20%
Temp: 73.4 F, Humidity: 20%
Temp: 75.2 F, Humidity: 20%
Temp: 77.0 F, Humidity: 20%
```

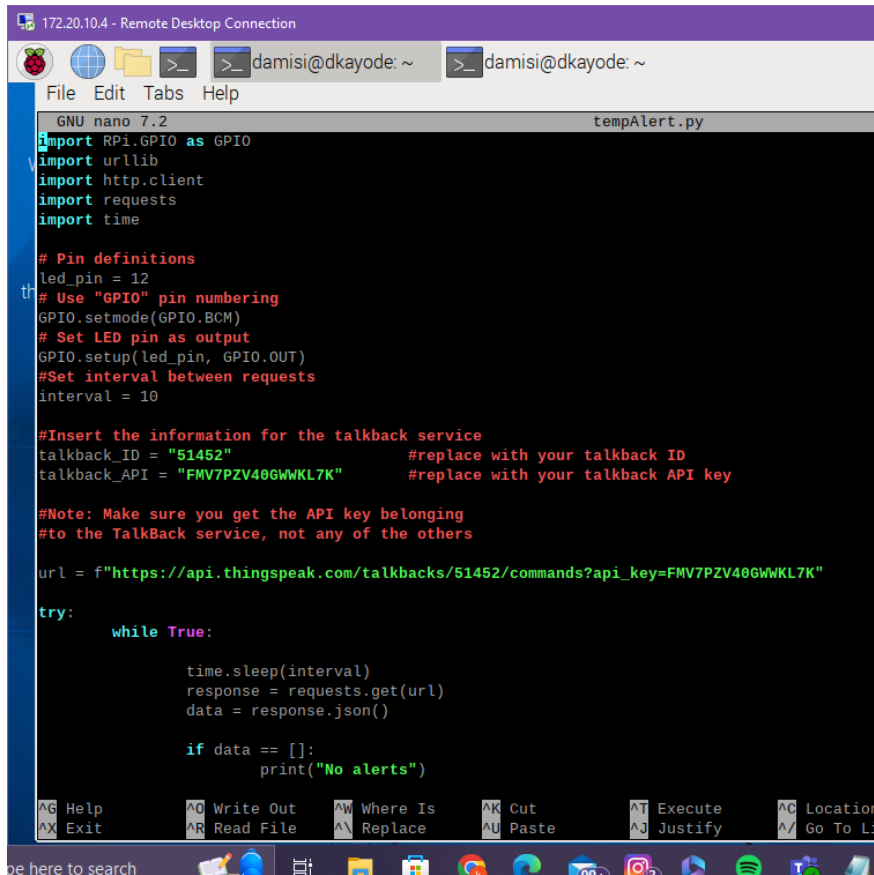
17. When the condition is met in **React**, a trigger will be sent to **ThingHTTP** to generate a POST command to **TalkBack**. We set up two separate ThingHTTP requests. For each request, fill in the configuration settings as illustrated in the figure below.

18. After setting up ThingSpeak, we open the reference Python program, *tempAlert.py*, and modify the code as instructed. It will make HTTP connection to poll commands from the TalkBack app to turn on or off the LED.

When executing the code, the LED should be turned on when the temperature goes above 75°F and turned off when it drops below 75°F. You can place your finger on the sensor or use a hair dryer to accelerate the heat-up!

## CPE4040 Lab Report

You should run this code on another terminal while *publish.py* is still running.



```
172.20.10.4 - Remote Desktop Connection
damisi@dkayode: ~
tempAlert.py
GNU nano 7.2
import RPi.GPIO as GPIO
import urllib
import http.client
import requests
import time

# Pin definitions
led_pin = 12
# Use "GPIO" pin numbering
GPIO.setmode(GPIO.BCM)
# Set LED pin as output
GPIO.setup(led_pin, GPIO.OUT)
#Set interval between requests
interval = 10

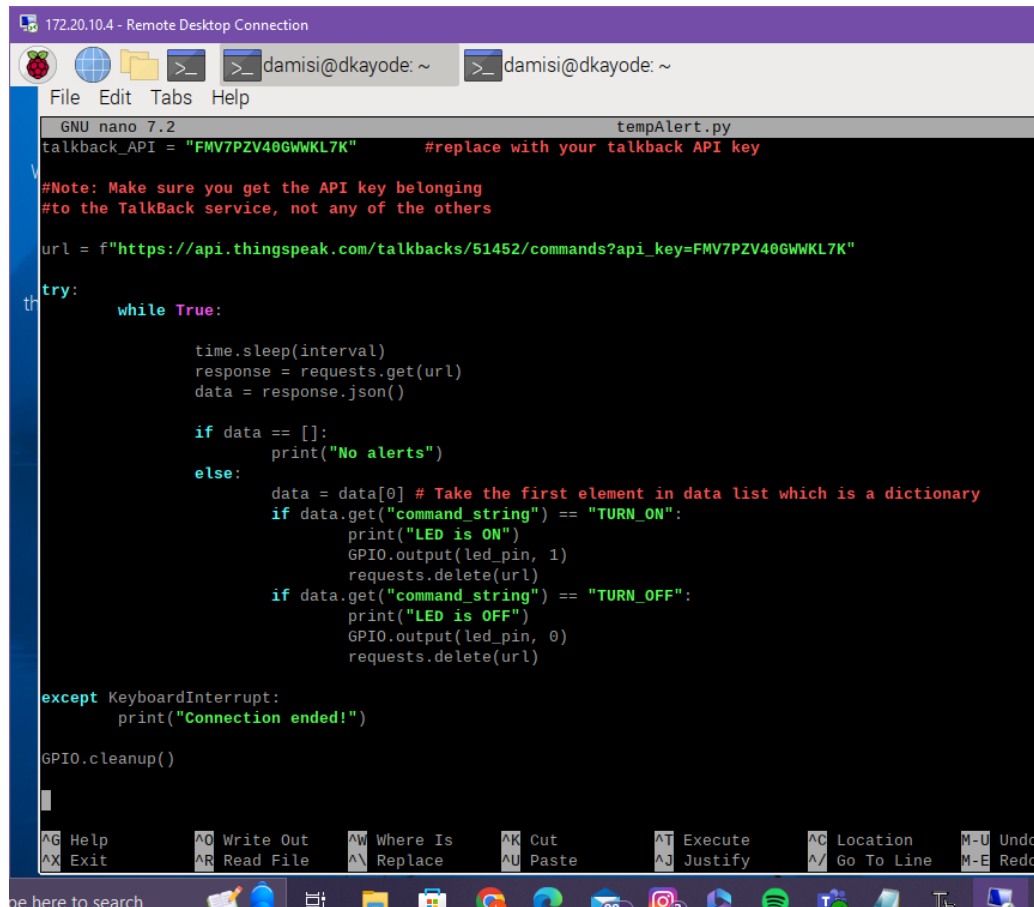
#Insert the information for the talkback service
talkback_ID = "51452" #replace with your talkback ID
talkback_API = "FMV7PZV406WWKL7K" #replace with your talkback API key

#Note: Make sure you get the API key belonging
#to the TalkBack service, not any of the others

url = f"https://api.thingspeak.com/talkbacks/51452/commands?api_key=FMV7PZV406WWKL7K"

try:
    while True:
        time.sleep(interval)
        response = requests.get(url)
        data = response.json()

        if data == []:
            print("No alerts")
```



```
172.20.10.4 - Remote Desktop Connection
damisi@dkayode: ~
File Edit Tabs Help
GNU nano 7.2 tempAlert.py
talkback_API = "FMV7PZV40GWWKL7K" #replace with your talkback API key
#Note: Make sure you get the API key belonging
#to the TalkBack service, not any of the others

url = f"https://api.thingspeak.com/talkbacks/51452/commands?api_key=FMV7PZV40GWWKL7K"

try:
    while True:
        time.sleep(interval)
        response = requests.get(url)
        data = response.json()

        if data == []:
            print("No alerts")
        else:
            data = data[0] # Take the first element in data list which is a dictionary
            if data.get("command_string") == "TURN_ON":
                print("LED is ON")
                GPIO.output(led_pin, 1)
                requests.delete(url)
            if data.get("command_string") == "TURN_OFF":
                print("LED is OFF")
                GPIO.output(led_pin, 0)
                requests.delete(url)

except KeyboardInterrupt:
    print("Connection ended!")

GPIO.cleanup()
```

## IV. Conclusion

This lab taught us how to create a Raspberry Pi Weather Station and use ThingSpeak for analyzing data. We got to do practical stuff like connecting a temperature sensor, making a cloud server, and using it to detect the temperature and humidity of the environment

We faced a problem while installing some Python packages using pip3. The regular method didn't work smoothly, so we used the 'break' command to force the installation. This was probably due to differences in the raspberry pi versions. To help others, it would be good to include troubleshooting steps in the lab guide, especially for common installation problems.