KENNESAW STATE UNIVERSITY

CPE 4040: Data Collection and Analysis, Spring 2023

Laboratory Report #4

Lab 4: Raspberry Pi Weather Station Application on ThingSpeak

Team Members: Neal Jarzen & Clarence Barron

Electrical and Computer Engineering

Kennesaw State University

Faculty: Dr. Jeffrey L Yiin

Date of Lab Experiment: February 26, 2023

I. Objective

The objective of this lab is to operate a temperature sensor with the use of ThingSpeak. Also, we learned how to use a client-server connection with our RPi with the use of creating channels to record data from temperature.

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

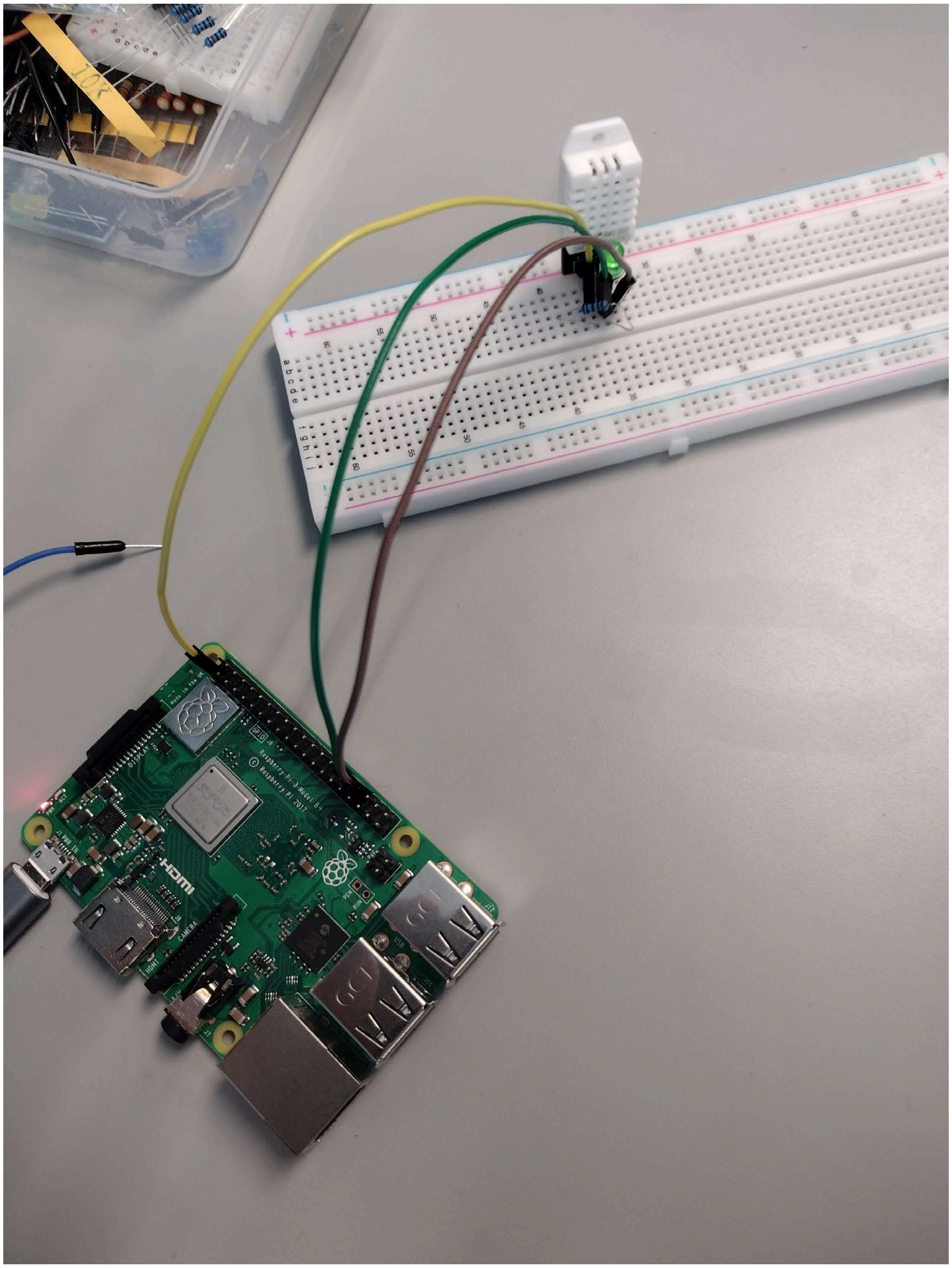
7. Temperature sensor (either DHT-11, DHT-22, or DHT-20).

8. 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 "**urlib, http-client, requests**". *urlib* is used to fetch URLs, while *http-client* is used to define classes that implement the client side of the HTTP and HTTPS protocols. Finally, *requests* are used to make HTTP requests using python which is simpler and friendlier for human interaction.

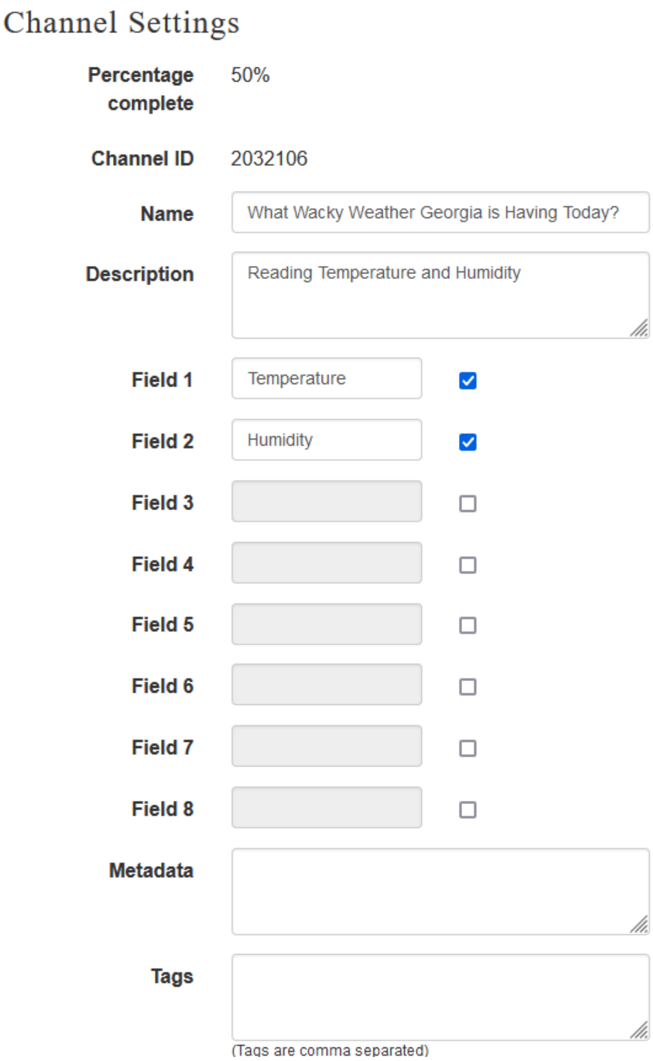
2) Now connect the temperature sensor to the Raspberry Pi according to the connection diagram shown below. There should be a resistor placed between the VCC and DATA pins of the sensor. Use a resistor of around 10kΩ



3) Afterward, install the DHT11/DHT22 drivers through the package “adafruit-circuitpython-dht" and download the libgpiod2 package as well. The libgpiod2 package will guarantee all allocated resources are freed after closing the device file descriptor, poll events, and setting and reading multiple values.
4) Next, you will need to create a file named “**digitalOut.py**” with the python code shown below to get sensor readings from your DHT11/DHT22.

```
import time
import board
import adafruit_dht

#Initialize the DHTdevice, with data pin connected to:
dhtDevice = adafruit_dht.DHT22 (board.D12)

#Print the temperature and humidity values to the terminal
temperature = dhtDevice.tem
humidity = dht.Device.humidity
        print(“Temp: {L.1f} C, Humidity: {}% “.format(temperature, humidity))
```

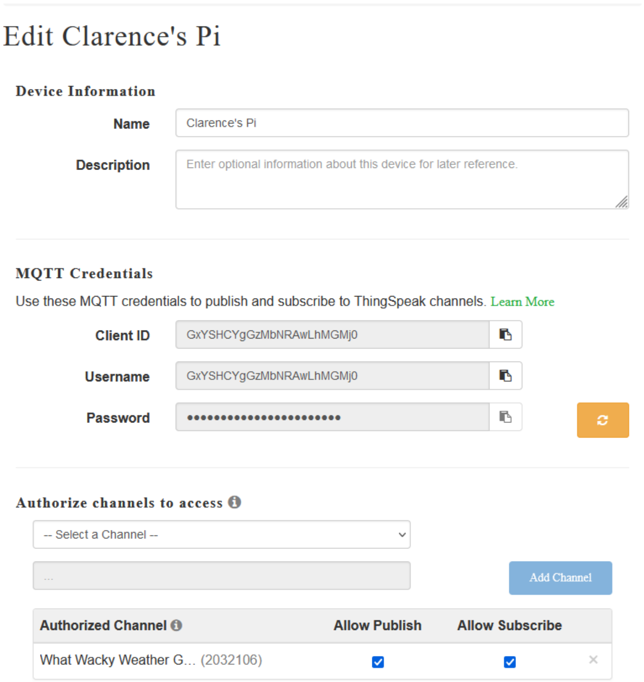
5) After setting up the digitalOut.py file to test your DHT11/DHT22, go to http://thingspeak.com and create an account. Afterward, go ahead and create a channel which will be at the top of the page. Click the “New Channel” button to create a new channel and fill out the name and description of the channel. In field 1 and 2, fill them out with Temperature and Humidity respectively as shown below.



Channel Settings

Percentage complete	50%	
Channel ID	2032106	
Name	What Wacky Weather Georgia is Having Today?	
Description	Reading Temperature and Humidity	
Field 1	Temperature	☑
Field 2	Humidity	☑
Field 3		□
Field 4		□
Field 5		□
Field 6		□
Field 7		□
Field 8		□
Metadata		
Tags	(Tags are comma separated)	

6) Afterward, go to the "Devices" tab, and add your PI. Then, click "Add Channel" and add the channel you made under the device and hit "Save". After saving go to the Client ID and Client Password. Save those in a separate document as those will be used in a future step. You may also add Latitude and Longitude to your channel as well, but it is not needed. Make sure you write down the "Channel ID".



Edit Clarence's Pi

Device Information

Name: Clarence's Pi

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: GxYSHCYgGzMbNRAwLhMGMj0

Username: GxYSHCYgGzMbNRAwLhMGMj0

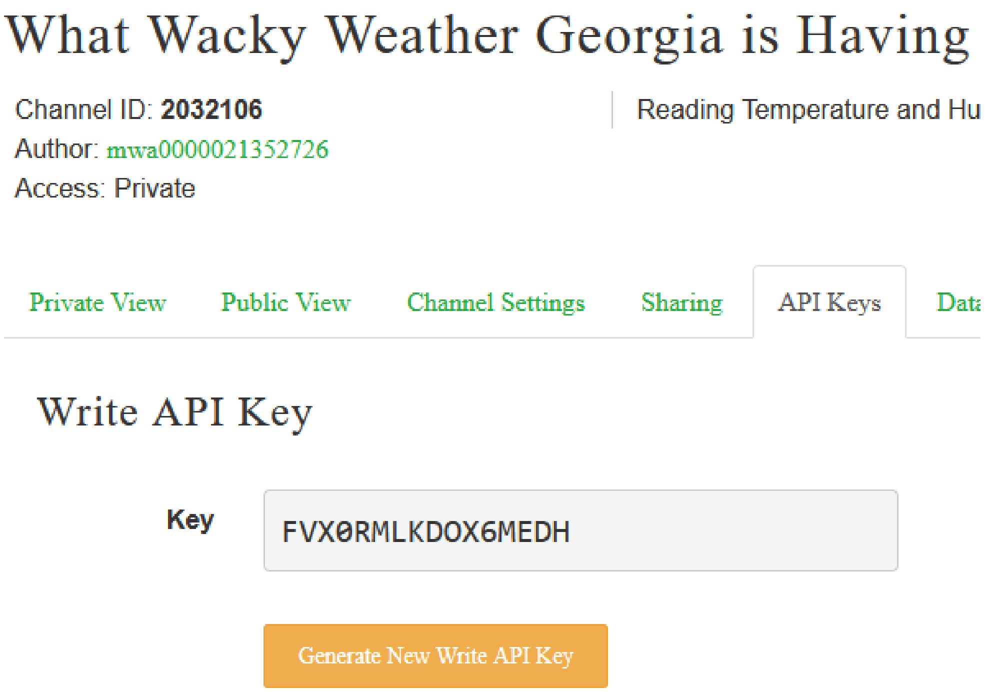
Password: •••••••••••••••••••••••

Authorize channels to access

-- Select a Channel --

Authorized Channel	Allow Publish	Allow Subscribe
What Wacky Weather G... (2032106)	☑	☑

7) Now, go into your channel and on the bar click “API Keys”. You will write down the “Write API Key” in a separate text document as this will be used for our code to communicate with the IoT website.



What Wacky Weather Georgia is Having

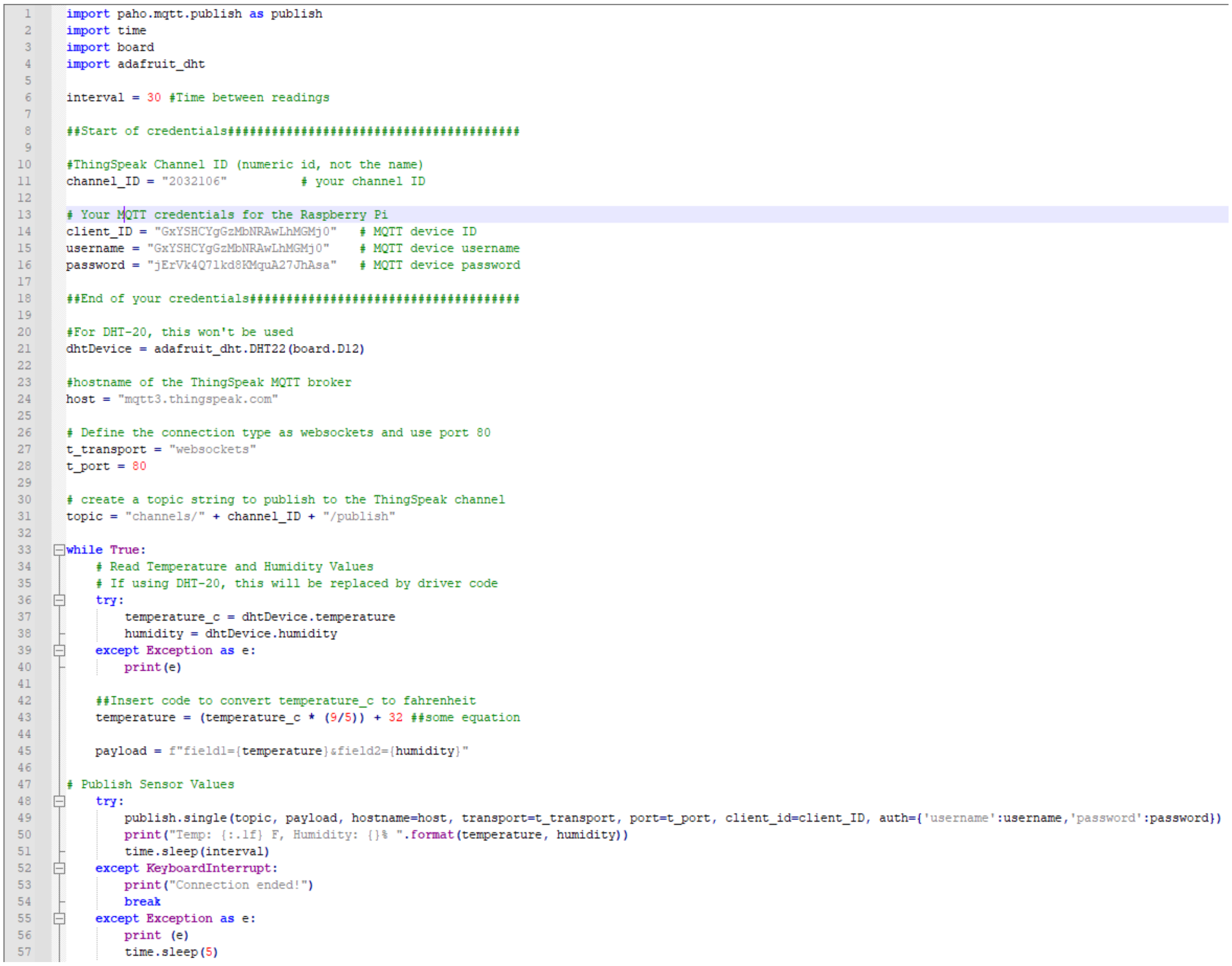
Channel ID: **2032106**
Author: mwa0000021352726
Access: Private

Reading Temperature and Hu

Write API Key

Key: FVX0RMLKDOX6MEDH

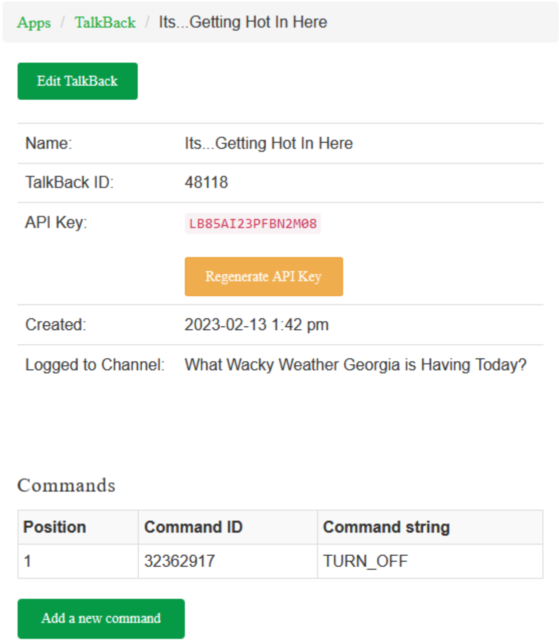
8) Modify your sensor reading code by adding the code shown below and then save it as “publish.py”.



```
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 = "2032106"          # your channel ID

# Your MQTT credentials for the Raspberry Pi
client_ID = "GxYSHCYgGzMbNRAwLhMGMj0"   # MQTT device ID
username = "GxYSHCYgGzMbNRAwLhMGMj0"    # MQTT device username
password = "jErVk4Q71kd8KMquA27JhAsa"   # MQTT device password

##End of your credentials####################################

#For DHT-20, this won't be used
dhtDevice = adafruit_dht.DHT22(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
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 ##some equation

    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)
```

9) Connect the LED to the GPIO12 pin of the RPi. When the temperature exceeds 75°F, the LED will turn on. If the temperature is below 75°F, the light will turn off.

10) With the use of the TalkBack app, which is also on ThingSpeak, we can make the LED turn on when it reaches the temperature threshold. Go to the TalkBack App and click “New TalkBack”. When created and given a name under “Log to Channel” insert your channel.



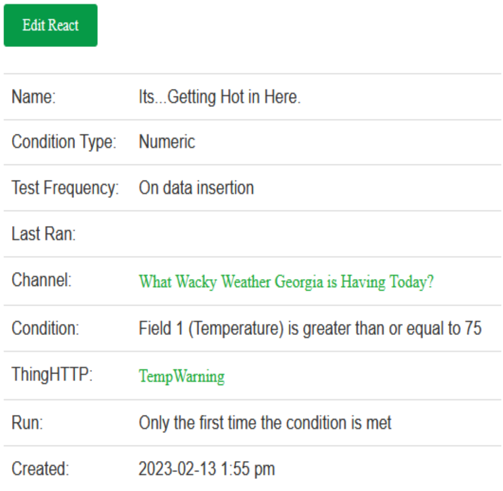
Apps / TalkBack / Its...Getting Hot In Here

Name:	Its...Getting Hot In Here
TalkBack ID:	48118
API Key:	LB85AI23PFBN2M08
Created:	2023-02-13 1:42 pm
Logged to Channel:	What Wacky Weather Georgia is Having Today?

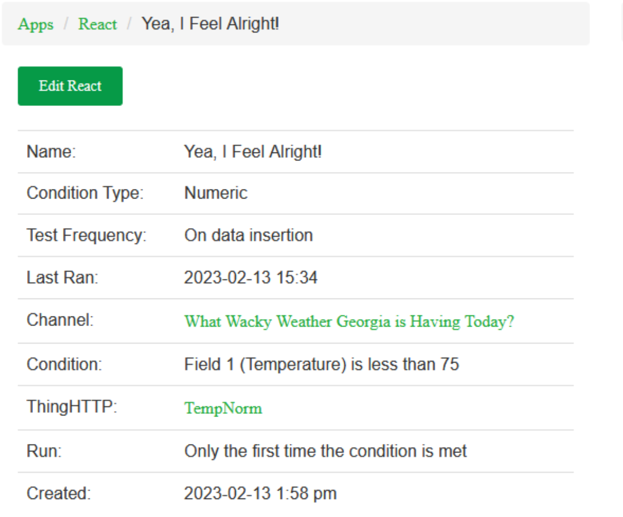
Commands

Position	Command ID	Command string
1	32362917	TURN_OFF

11) With the use of the React App, we can make actions happen with the use of the TalkBack app when it is called. Create two React channels related to the conditions in step 9. The two React Apps will follow similar fields as shown below, but one will be less than 75°F and the other will be greater than or equal to 75°F.



Name:	Its...Getting Hot in Here.
Condition Type:	Numeric
Test Frequency:	On data insertion
Last Ran:	
Channel:	What Wacky Weather Georgia is Having Today?
Condition:	Field 1 (Temperature) is greater than or equal to 75
ThingHTTP:	TempWarning
Run:	Only the first time the condition is met
Created:	2023-02-13 1:55 pm

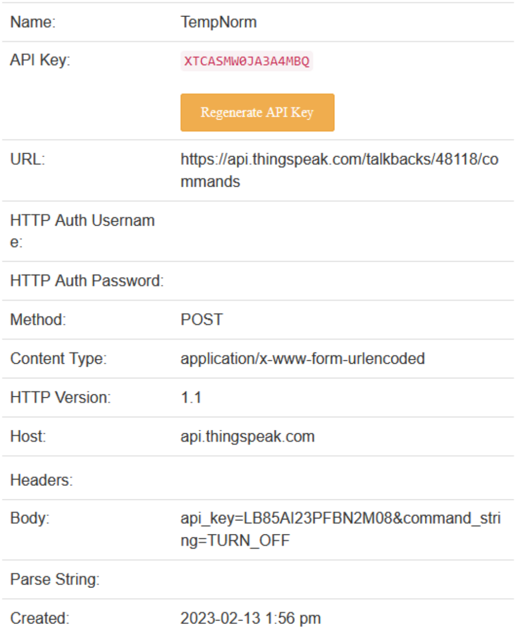


Apps / React / Yea, I Feel Alright!

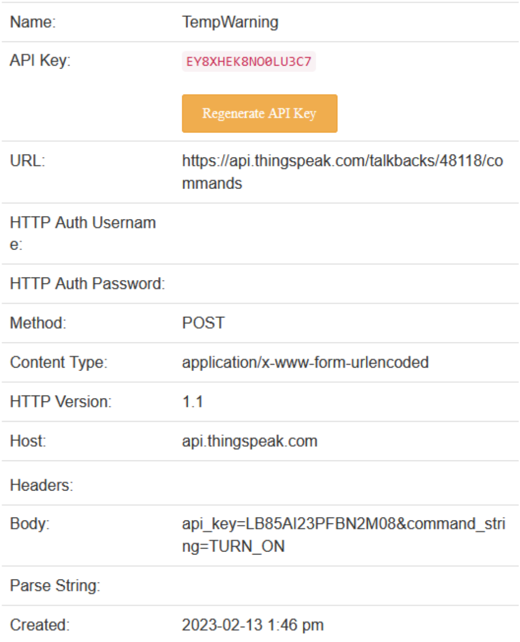
Name:	Yea, I Feel Alright!
Condition Type:	Numeric
Test Frequency:	On data insertion
Last Ran:	2023-02-13 15:34
Channel:	What Wacky Weather Georgia is Having Today?
Condition:	Field 1 (Temperature) is less than 75
ThingHTTP:	TempNorm
Run:	Only the first time the condition is met
Created:	2023-02-13 1:58 pm

12) When the conditions are met in the React app, it will trigger "**ThingHTTP**" app to generate special commands and post them to the talkback app. You will need to add

two ThingHTTP requests with the settings shown below for the purpose. In the URL section, we need to enter your Talkback ID, and in the body section, you will then enter your talkback API Key.

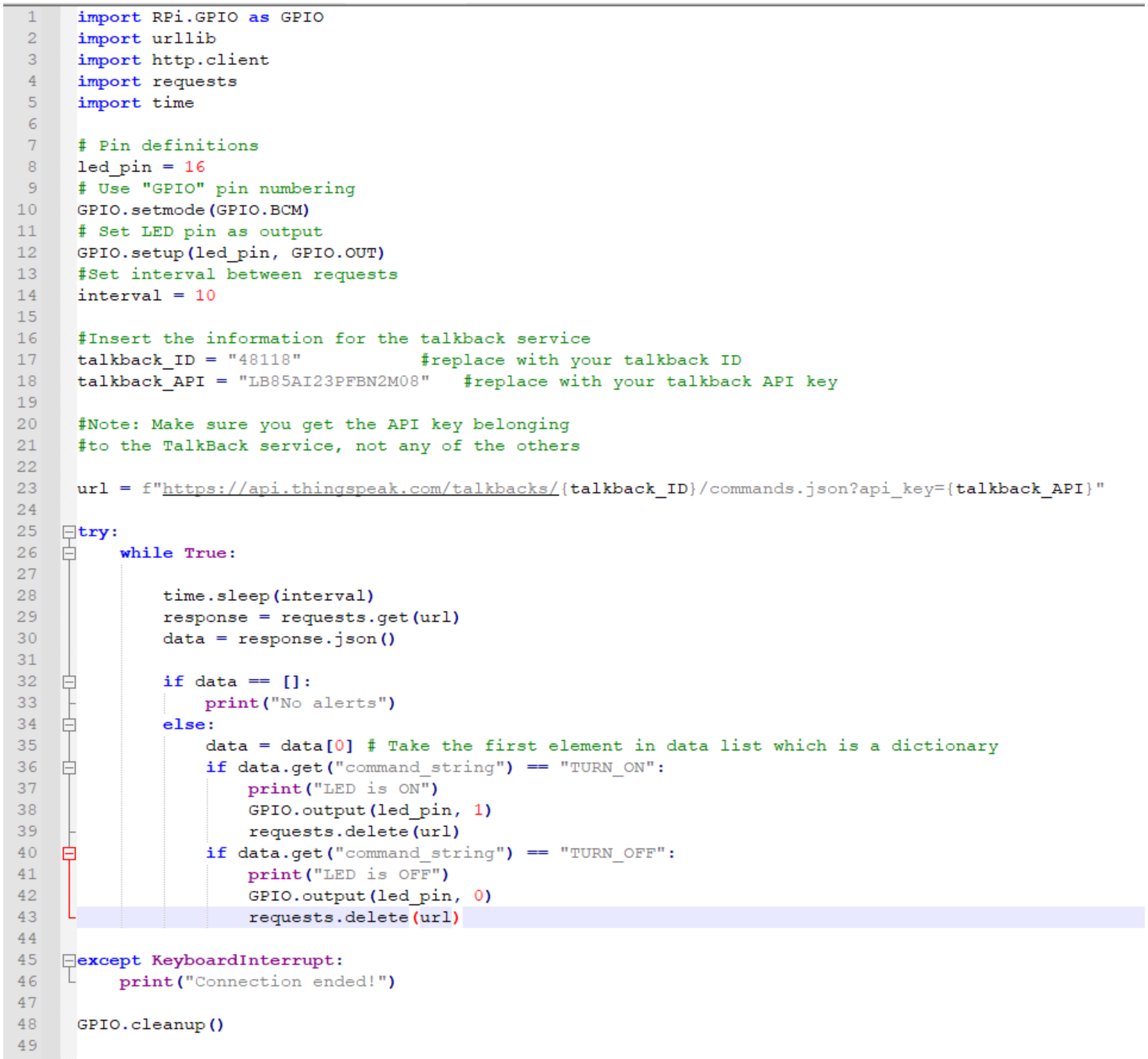


Name:	TempNorm
API Key:	XTCASMW0JA3A4MBQ Regenerate API Key
URL:	https://api.thingspeak.com/talkbacks/48118/commands
HTTP Auth Username:	
HTTP Auth Password:	
Method:	POST
Content Type:	application/x-www-form-urlencoded
HTTP Version:	1.1
Host:	api.thingspeak.com
Headers:	
Body:	api_key=LB85AI23PFBN2M08&command_string=TURN_OFF
Parse String:	
Created:	2023-02-13 1:56 pm

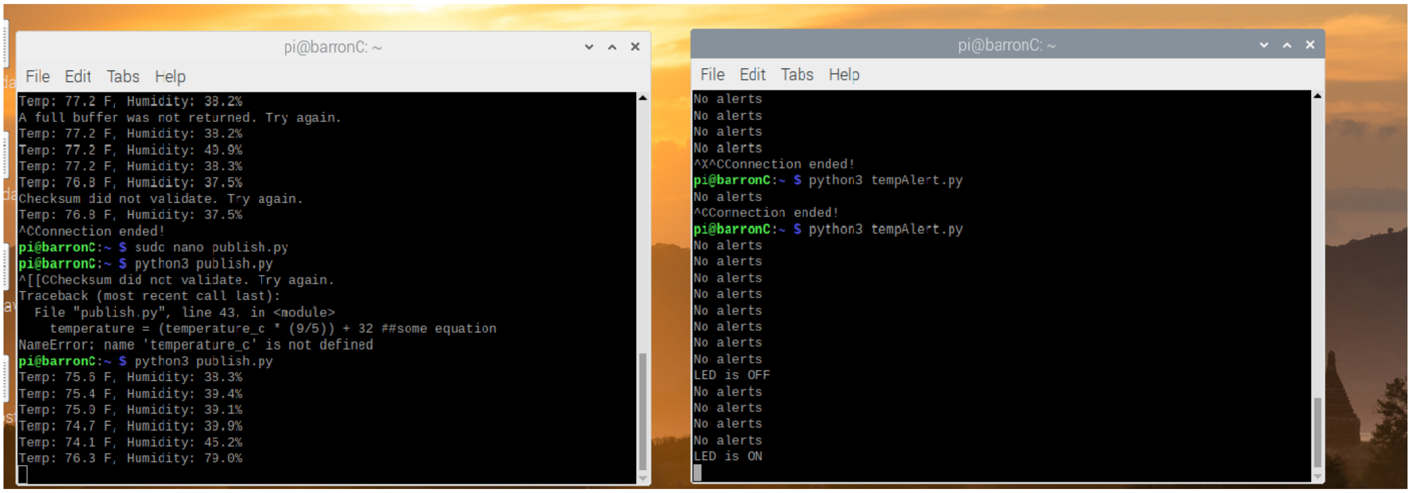


Name:	TempWarning
API Key:	EY8XHEK8NO0LU3C7 Regenerate API Key
URL:	https://api.thingspeak.com/talkbacks/48118/commands
HTTP Auth Username:	
HTTP Auth Password:	
Method:	POST
Content Type:	application/x-www-form-urlencoded
HTTP Version:	1.1
Host:	api.thingspeak.com
Headers:	
Body:	api_key=LB85AI23PFBN2M08&command_string=TURN_ON
Parse String:	
Created:	2023-02-13 1:46 pm

13) After setting up the ThingSpeak App, with the program, "***tempAlert.py***", this will allow the LED to turn on or off depending on the temperature. Also, the program will send messages to a second terminal. This will run while "***publish.py***" is running in the first terminal.



```python
import RPi.GPIO as GPIO
import urllib
import http.client
import requests
import time

# Pin definitions
led_pin = 16
# 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 = "48118"           #replace with your talkback ID
talkback_API = "LB85AI23PFBN2M08"   #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/{talkback_ID}/commands.json?api_key={talkback_API}"

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()

```

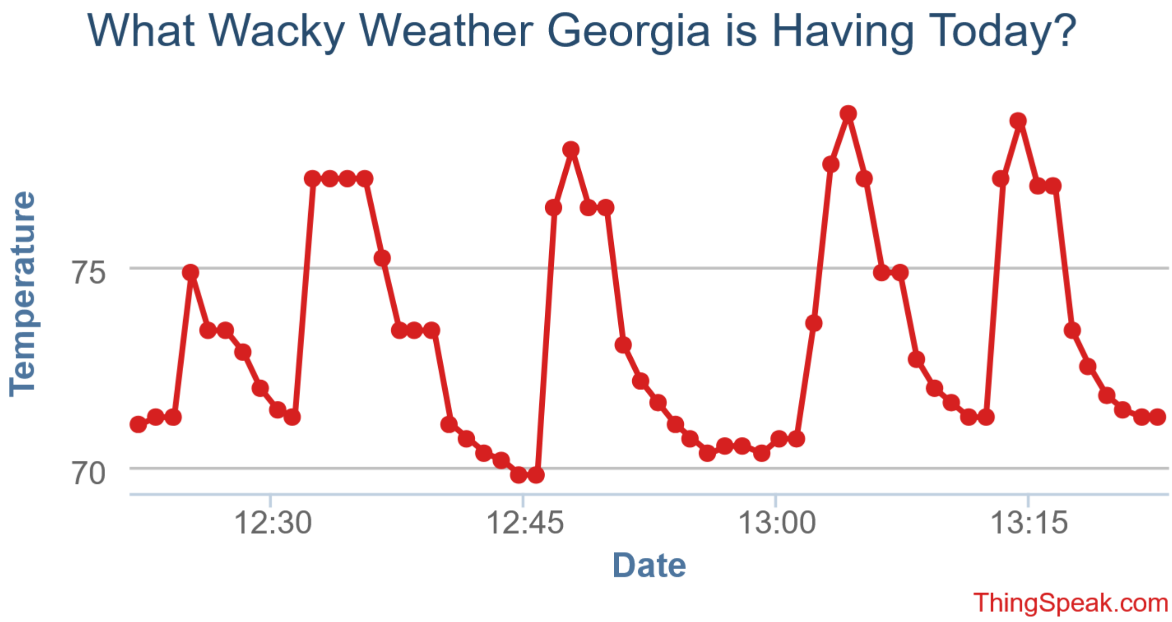


```
Temp: 77.2 F, Humidity: 38.2%
A full buffer was not returned. Try again.
Temp: 77.2 F, Humidity: 38.2%
Temp: 77.2 F, Humidity: 40.9%
Temp: 77.2 F, Humidity: 38.3%
Temp: 76.8 F, Humidity: 37.5%
Checksum did not validate. Try again.
Temp: 76.8 F, Humidity: 37.5%
^CConnection ended!
pi@barronC:~ $ sudo nano publish.py
pi@barronC:~ $ python3 publish.py
^[[CChecksum did not validate. Try again.
Traceback (most recent call last):
  File "publish.py", line 43, in <module>
    temperature = (temperature_c * (9/5)) + 32 ##some equation
NameError: name 'temperature_c' is not defined
pi@barronC:~ $ python3 publish.py
Temp: 75.6 F, Humidity: 38.3%
Temp: 75.4 F, Humidity: 39.4%
Temp: 75.0 F, Humidity: 39.1%
Temp: 74.7 F, Humidity: 39.9%
Temp: 74.1 F, Humidity: 45.2%
Temp: 76.3 F, Humidity: 79.0%
```

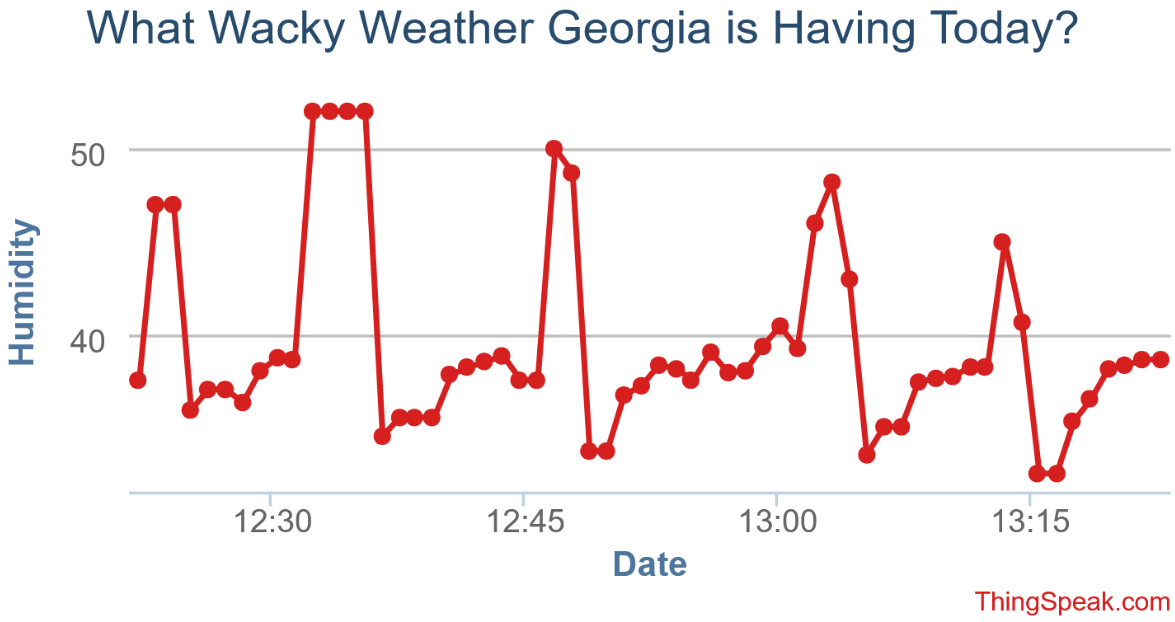
```
No alerts
No alerts
No alerts
No alerts
^X^CConnection ended!
pi@barronC:~ $ python3 tempAlert.py
No alerts
^CConnection ended!
pi@barronC:~ $ python3 tempAlert.py
No alerts
No alerts
No alerts
No alerts
No alerts
No alerts
No alerts
No alerts
LED is OFF
No alerts
No alerts
No alerts
No alerts
LED is ON
```

Video: https://youtu.be/UzBBvwGWv7s

Temperature Graph:



Humidity Graph:



IV. Conclusion

We were able to complete the lab successfully with minimal trouble after the hardware issues we were experiencing were alleviated. We were using the DHT-11 with a three-prong connector, we connected the sensor to the board correctly and made sure that the code we were executing was correct. But there was no response from the sensor, the sensor was supposed to be reading the temperature and then outputting it into the channel. Although we believed that we had a working sensor, this was not the case. We had to receive a substitute DHT22 for this lab. After we received the substitute hardware, we were able to get send alerts from our weather station and smoothly complete the rest of the lab.