Benedict Odai

<u>Component 1</u>

Data was collected every 2 minutes for 2hrs each day. Totally to 10 hours' worth of data. The data was stored on google sheets.

Used IFTTT to send the data to the google sheets.

IFTTT: If This Then That is a service that allows a user to program a response to events in the world of various kinds.

First wrote a python script to connect to the Wi-Fi, and then sent a post request to IFTTT, which updated the google sheets with the current live temperature and humidity data from the ESP32. Each reading had a timestamp associated with it.

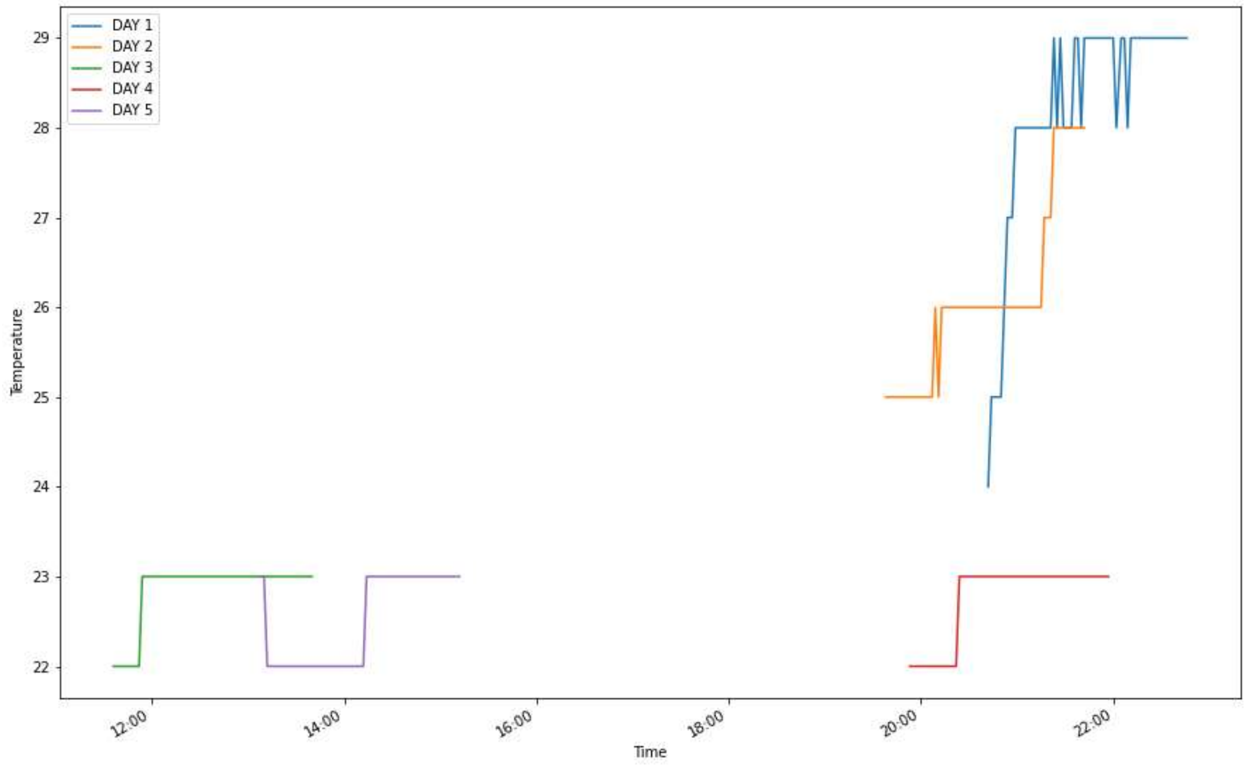
<u>Component 2</u>

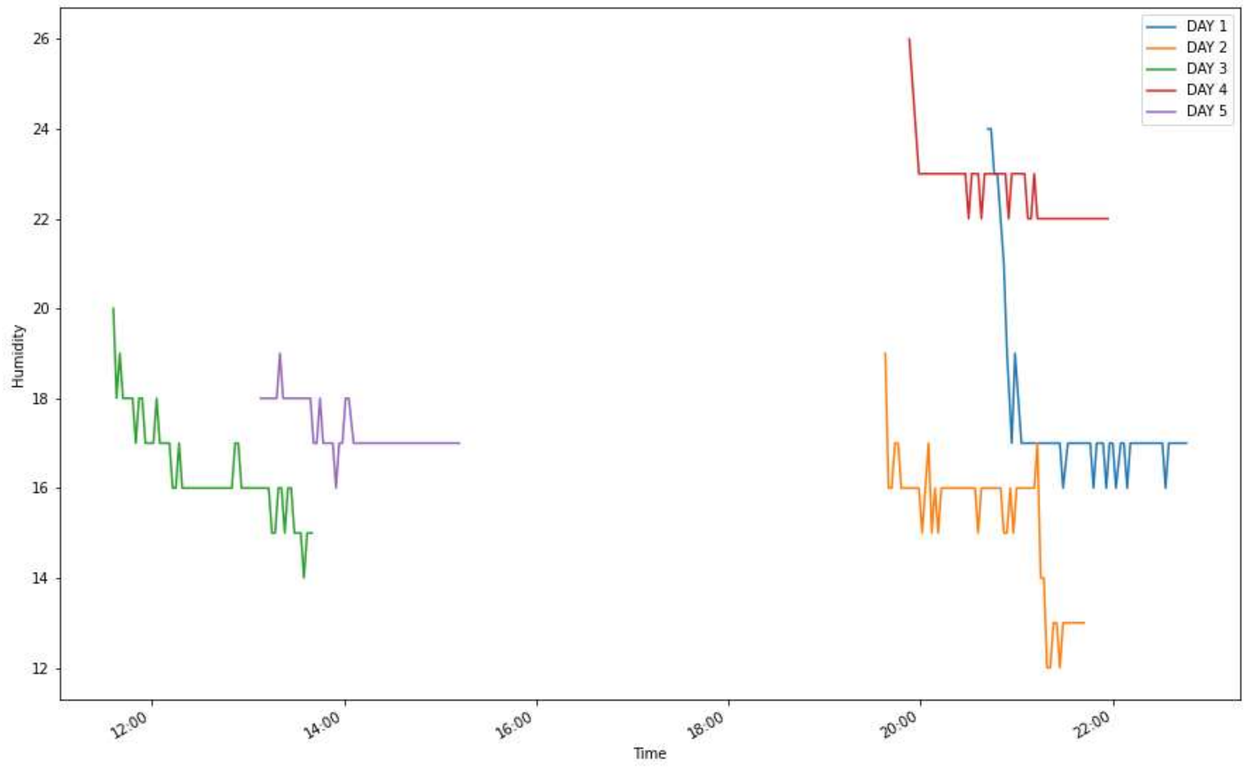
The data was then analyzed and visualized in python.

First, I extracted the data from the csv file using the python pandas library, and then the date was parsed in order to have it in the right format.

Then I extracted the first 61 rows from the dataset, to represent day1. Then I extracted the following 61 datasets for day2, and I continued to do the same for the remaining day 3, 4 and 5.

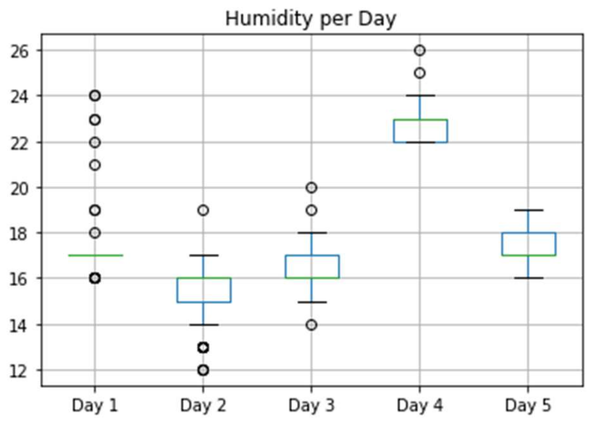
I then plotted the data on a graph. There was some difficulty getting the data to plot in the appropriate manner due to the way the matplotlib operated.

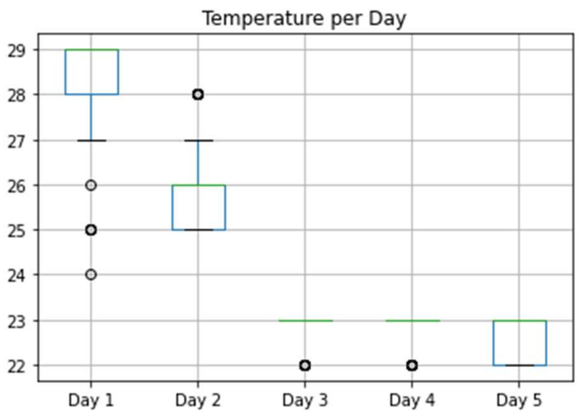




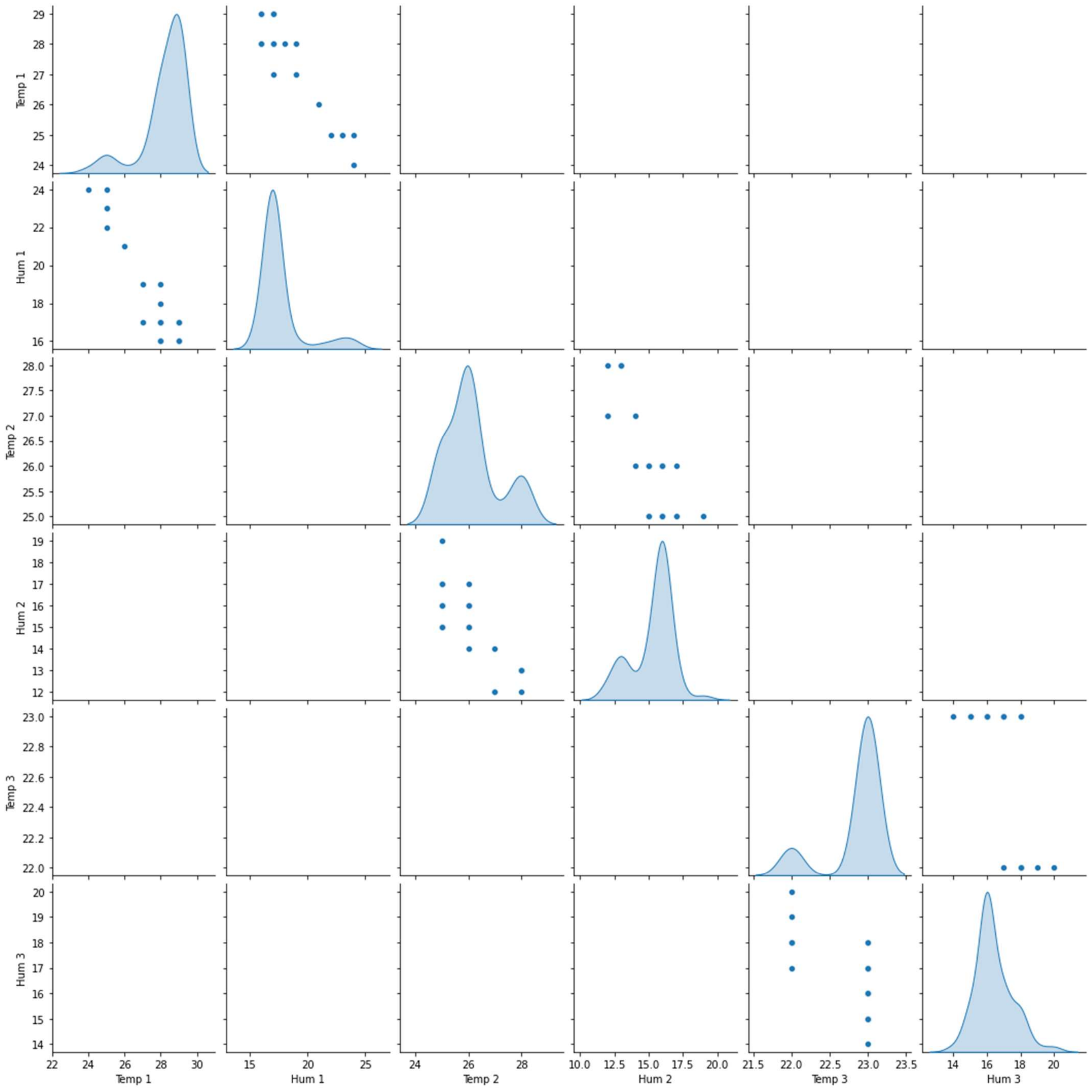
Graph of Humidity and time, for 5 different days. Shows that the humidity for day 1,2,3,5 were similar and consistent despite it being at different times, except for day 4 which is higher than the other days.

Graph of Temperature over time., for 5 different days. Shows that the temperature for days 3,4 and 5 was much lower than the temperature for day 1 and 2.





Mean Humidity for the first 3 days was lower than the humidity for the later 3 days. The mean temperature for day 3,4 and 5 were the same. There were a lot of outliers for the Humidity and Temperature for day 1.



Scatter matrix to show the relationship between 2 variables. Temp1 signifying temperature for day 1,etc

Component 3

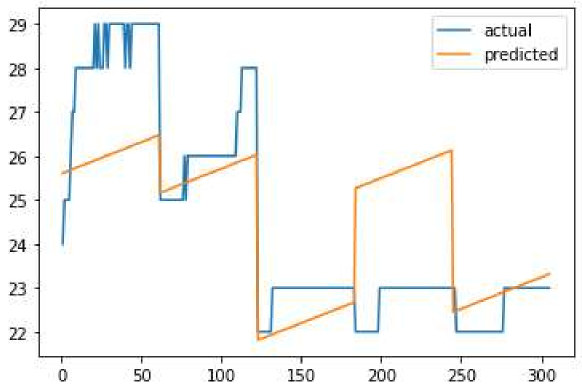
For Model 1:

M1 = skl_lm.LinearRegression()

X = data.minutes.values.reshape(-1,1)

y = data.Temperature

M1.fit(X,y)



intercept: 16.975363232639225

coefficient: [0.00695232]

Temperature = 16.975 + 0.006 * minutes + epsilon

Change in Temperature (y axis) over time. Number of observations (x-axis).

The predicted value for the Temperature is not accurate. It's very off from the actual value.

For Model 2:

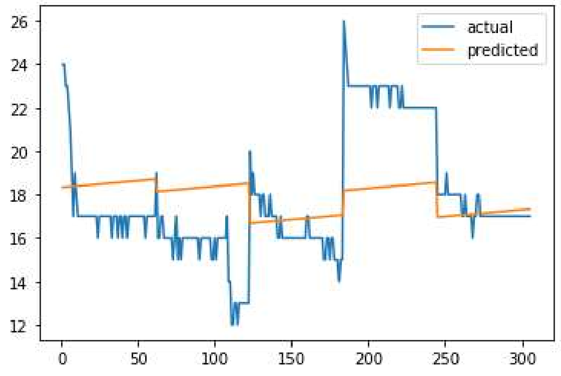
2. Create a linear regression model between Humidity and minutes. Call this model M2.

M2 = skl_lm.LinearRegression()

X = data.minutes.values.reshape(-1,1)

y = data.Humidity

M2.fit (X,y)



intercept: 14.571265813152287

coefficient: [0.00302584]

Humidity = 14.571 + 0.003 * minutes + epsilon

Change in Humidity (y axis) over time. Number of observations (x-axis).

Change in Humidity over time. There is a substantial difference between the predicted value for the humidity and actual value. Perhaps more variables are required to create a more reliable model.

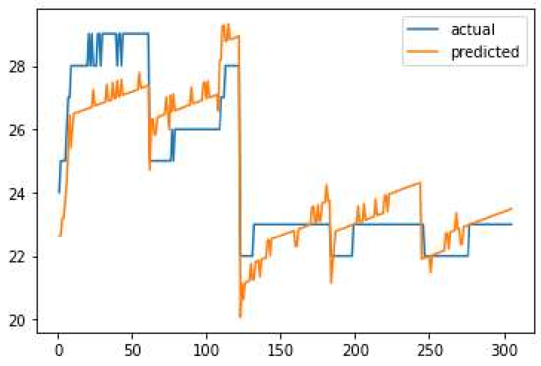
For Model 3:

```
M3 = skl_lm.LinearRegression()

X = data [['Humidity', 'minutes']]

y = data.Temperature

M3.fit(X,y)
```



coefficient: [-0.52591291 0.00854365]

intercept: 24.638580084048858

Temperature = 24.638 + -0.525 * Humidity +0.0085*minutes + epsilon

Change in temperature as a result of a change humidity and minutes.

Number of observations (X-axis)

The predicted value for the graph is not accurate, however it's not too far off, Making it a better model than the previous 2, M1 and M2.

4.

Mean Squared Error tells how accurate the calculated coefficient with respect is to the actual value.

Mean Squared Error of M1: 3.2356426675499463

Mean Squared Error of M2: 7.422252857487001

Mean Squared Error of M3: 1.1827633724322417

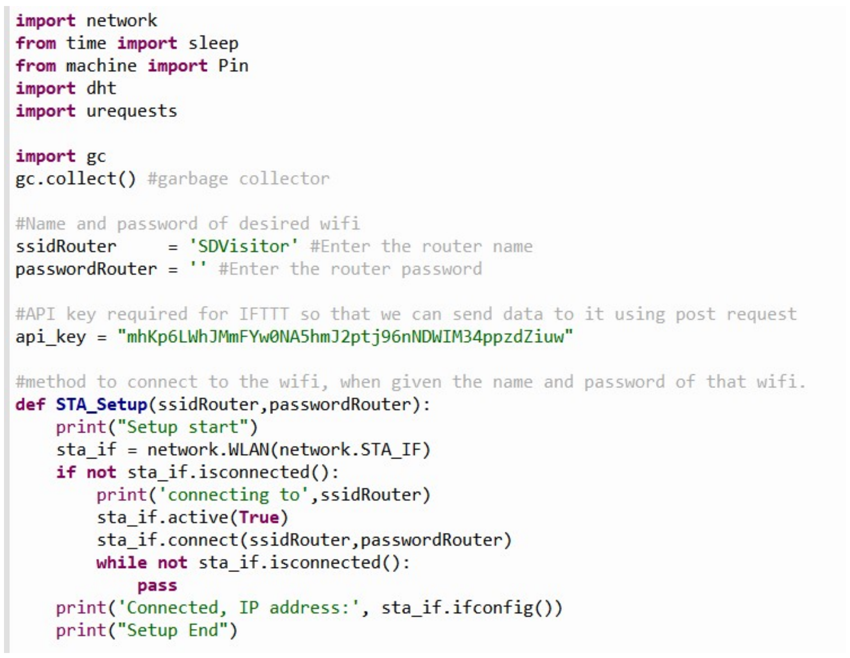
Based off the values we can see that M3 produces the least mean squared error, hence proving to be the best model.

AIC of model 1, model 2, and model 3 are -11.79 -13.45 -7.78

AIC estimates the quality of each model, relative to each of the other models.

The preferred model gives an AIC value close to zero. Hence, according to the AIC values model 3 is better.

-----------------------Code for ESP32-----------------------------------



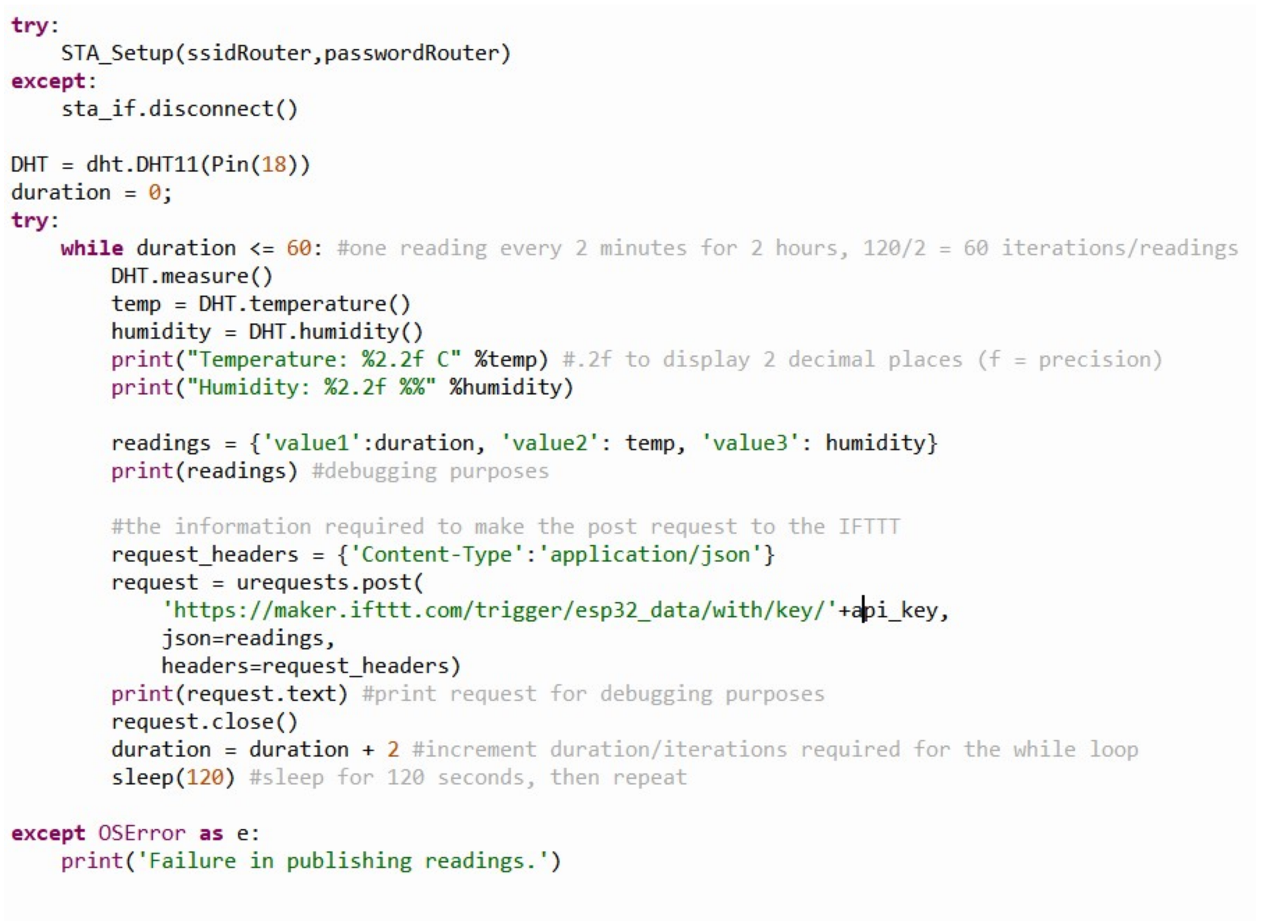
```
import network
from time import sleep
from machine import Pin
import dht
import urequests

import gc
gc.collect() #garbage collector

#Name and password of desired wifi
ssidRouter     = 'SDVisitor' #Enter the router name
passwordRouter = '' #Enter the router password

#API key required for IFTTT so that we can send data to it using post request
api_key = "mhKp6LWhJMmFYw0NA5hmJ2ptj96nNDWIM34ppzdZiuw"

#method to connect to the wifi, when given the name and password of that wifi.
def STA_Setup(ssidRouter,passwordRouter):
    print("Setup start")
    sta_if = network.WLAN(network.STA_IF)
    if not sta_if.isconnected():
        print('connecting to',ssidRouter)
        sta_if.active(True)
        sta_if.connect(ssidRouter,passwordRouter)
        while not sta_if.isconnected():
            pass
    print('Connected, IP address:', sta_if.ifconfig())
    print("Setup End")
```



```
try:
    STA_Setup(ssidRouter,passwordRouter)
except:
    sta_if.disconnect()

DHT = dht.DHT11(Pin(18))
duration = 0;
try:
    while duration <= 60: #one reading every 2 minutes for 2 hours, 120/2 = 60 iterations/readings
        DHT.measure()
        temp = DHT.temperature()
        humidity = DHT.humidity()
        print("Temperature: %2.2f C" %temp) #.2f to display 2 decimal places (f = precision)
        print("Humidity: %2.2f %%" %humidity)

        readings = {'value1':duration, 'value2': temp, 'value3': humidity}
        print(readings) #debugging purposes

        #the information required to make the post request to the IFTTT
        request_headers = {'Content-Type':'application/json'}
        request = urequests.post(
            'https://maker.ifttt.com/trigger/esp32_data/with/key/'+api_key,
            json=readings,
            headers=request_headers)
        print(request.text) #print request for debugging purposes
        request.close()
        duration = duration + 2 #increment duration/iterations required for the while loop
        sleep(120) #sleep for 120 seconds, then repeat

except OSError as e:
    print('Failure in publishing readings.')
```