**Name: Austin Davis**

**Class: ECT 308**

**Lab #: 4**

Thingspeak

Thingspeak is an analytics platform that allows you to view and manipulate data streams. For example in this lab we fed Thingspeak data from our Arduino uno and had Thingspeak create a visualization of our data. While we used it for a personal project it has many industry capable features.

Thing speak allows for data manipulation analytics all on the cloud. Normally a data engineer would pull from the cloud to software on his station. With thingspeak it combines the two and allows most if not all algorithm development to be done on the cloud.

Website

[Austindavis.life](http://austindavis.life)/project1.html

Navigate to the projects page, hover over the thermistor project, and click the little computer symbol that pops up to view the live updating charts.

# Objective

Design and build a thermistor circuit that sens data to thingspeak then display data as a graph on a personal website

# Algorithm

1. Put thermistor into a breadboard
2. Take one end of the thermistor to power then the other end to a matching resistor
3. On the same lane as both the resistor and the thermistor take a wire to A0
4. Then take the end of the resistor and take that to ground
5. Then its time to begin the code
6. First, start by writing code that serial prints both Fahrenheit and celsius
7. Then add code to connect the wifi shield to the internet
8. Once connected to the internet send the data to thingspeak
9. Then use the code provided by thing speak to display the 2 graphs formed on a public website.

# Hardware & Software Used

**Hardware:**

(1) x Elegoo Uno R3

(1) x 830 tie-pointsbreadboard

(1) x Thermistor

(1) x Resistor

(6) x M-M wires (Male to Male jumper wires)

**Software:**

**Arduino Code:**

#include <ThingSpeak.h>

#include <SoftwareSerial.h>

#include <SparkFunESP8266WiFi.h>

const char mySSID[] = "linksys";

const char myPSK[] = "12345678";

int ThermistorPin = 0;

int Vo;

float R1 = 10000;

float logR2, R2, T, Tc, Tf;

float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 = 2.019202697e-07;

ESP8266Client client;

unsigned long myChannelNumber = 2050984;

const char \* myWriteAPIKey = "HB4YRSJS8AQVXR5E";

void setup() {

Serial.begin(9600);

initializeESP8266();

connectESP8266();

ThingSpeak.begin(client);

}

void loop(){

Vo = analogRead(ThermistorPin);

R2 = R1 \* (1023.0 / (float)Vo - 1.0);

logR2 = log(R2);

T = (1.0 / (c1 + c2\*logR2 + c3\*logR2\*logR2\*logR2));

Tc = T - 273.15;

Tf = (Tc \* 9.0)/ 5.0 + 32.0;

Serial.print(Tc);

ThingSpeak.setField (2, Tf);

ThingSpeak.setField(1, Tc);

int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

if(x == 200){

Serial.println("Channel update successful.");

}

else{

Serial.println("Problem updating channel. HTTP error code " + String(x));

}

delay(20000);

}

void initializeESP8266()

{

// esp8266.begin() verifies that the ESP8266 is operational

// and sets it up for the rest of the sketch.

// It returns either true or false -- indicating whether

// communication was successul or not.

// true

int test = esp8266.begin();

if (test != true)

{

Serial.println(F("Error talking to ESP8266."));

errorLoop(test);

}

Serial.println(F("ESP8266 Shield Present"));

}

void errorLoop(int error)

{

Serial.print(F("Error: ")); Serial.println(error);

Serial.println(F("Looping forever."));

for (;;)

;

}

void connectESP8266()

{

// The ESP8266 can be set to one of three modes:

// 1 - ESP8266\_MODE\_STA - Station only

// 2 - ESP8266\_MODE\_AP - Access point only

// 3 - ESP8266\_MODE\_STAAP - Station/AP combo

// Use esp8266.getMode() to check which mode it's in:

int retVal = esp8266.getMode();

if (retVal != ESP8266\_MODE\_STA)

{ // If it's not in station mode.

// Use esp8266.setMode([mode]) to set it to a specified

// mode.

retVal = esp8266.setMode(ESP8266\_MODE\_STA);

if (retVal < 0)

{

Serial.println(F("Error setting mode."));

errorLoop(retVal);

}

}

Serial.println(F("Mode set to station"));

// esp8266.status() indicates the ESP8266's WiFi connect

// status.

// A return value of 1 indicates the device is already

// connected. 0 indicates disconnected. (Negative values

// equate to communication errors.)

retVal = esp8266.status();

if (retVal <= 0)

{

Serial.print(F("Connecting to "));

Serial.println(mySSID);

// esp8266.connect([ssid], [psk]) connects the ESP8266

// to a network.

// On success the connect function returns a value >0

// On fail, the function will either return:

// -1: TIMEOUT - The library has a set 30s timeout

// -3: FAIL - Couldn't connect to network.

retVal = esp8266.connect(mySSID, myPSK);

if (retVal < 0)

{

Serial.println(F("Error connecting"));

errorLoop(retVal);

}

}

}

**Website Code**

<div class="center">

<head>

<title>Data Collection Dashboard</title>

</head>

<body>

<table>

<tr><td colspan="2">

<h1 align="center" color="#00FFFF">Data Collection Dashboard</h1>

</td></tr>

<tr><td>

<iframe width="450" height="260" style="border: 1px solid #cccccc;" src="https://thingspeak.com/channels/2050984/charts/1?bgcolor=%23ffffff&color=%23d62020&dynamic=true&results=60&title=Celsius&type=line"></iframe>

</td>

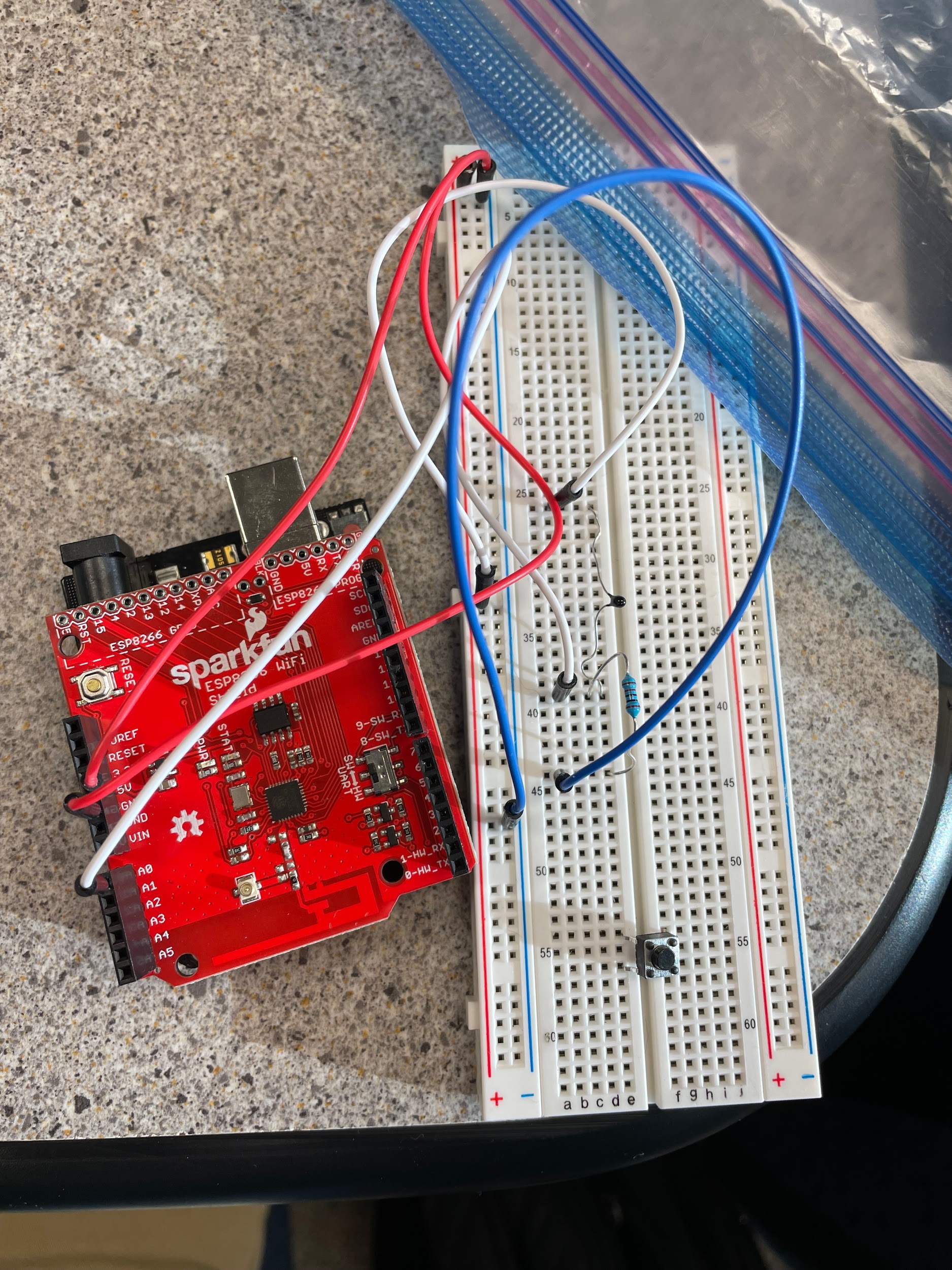
<td><iframe width="450" height="260" style="border: 1px solid #cccccc;" src="https://thingspeak.com/channels/2050984/charts/2?bgcolor=%23ffffff&color=%23d62020&dynamic=true&results=60&title=Fahrenheit&type=line"></iframe>

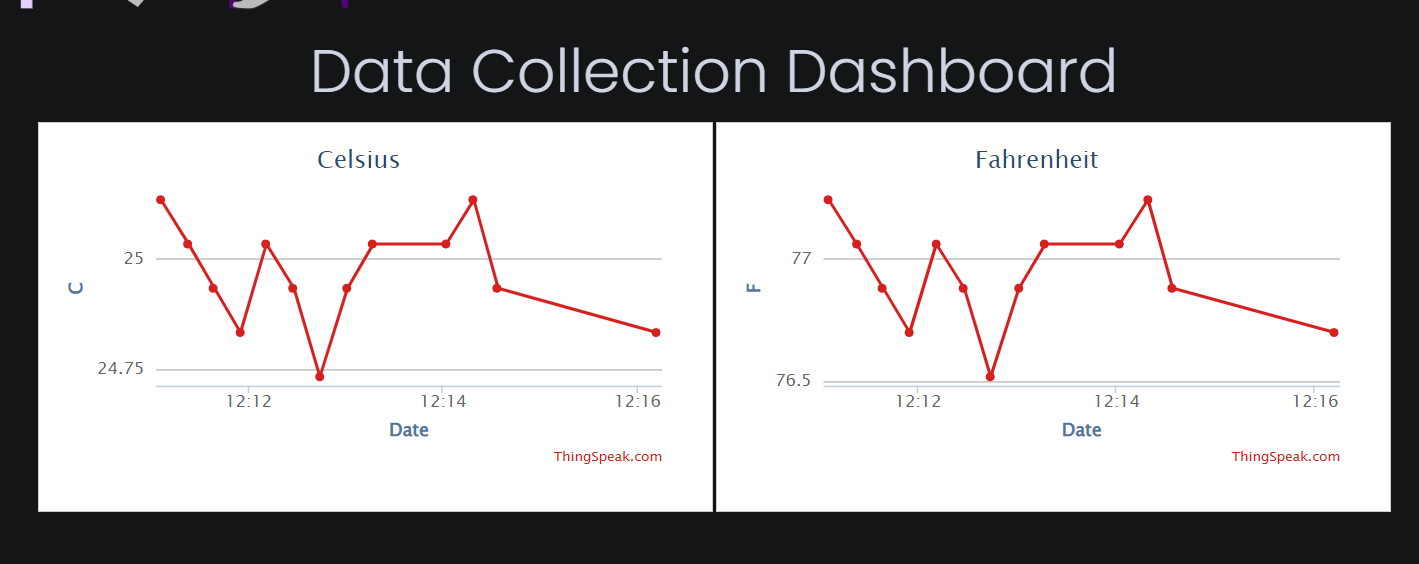
</td></tr>

</table>

</div>

Schematic of Project





# Conclusions (Test & Debug)

I am having an issue where my html code won't pull my style sheet correctly so the “center” style isn't working. Meaning the graphs are not centered for every device.