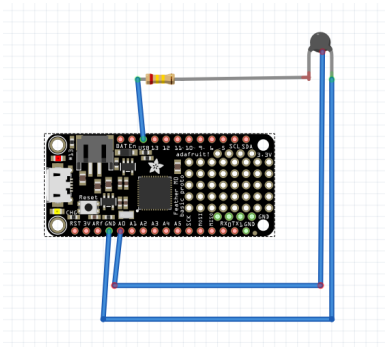
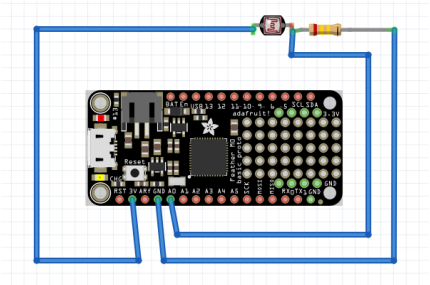
Brandon Martinez
Kevin Fang

Final Report: Project #1

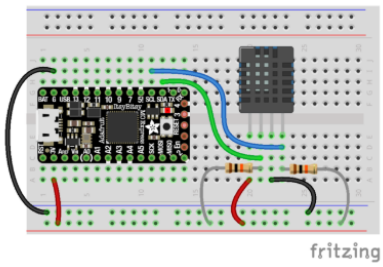
Before we started on the project, we gathered all of the previous information from our labs and planned on how to assemble all of the components into our system. We started on the first circuits we built in this class containing the thermistor, photocell, and later on, the humidity sensor. We combined the codes for the first two sensors and later downloaded the adafruit_bus_drive and am2320.mpy files into Circuitpy and implemented the corresponding I2C code. The circuits are shown below.



(I) Thermistor Circuit

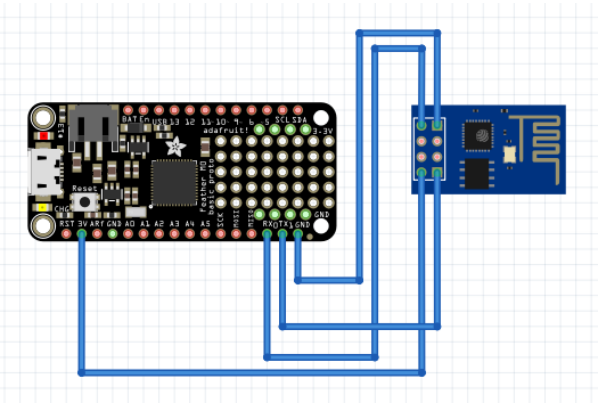


(II) Photoresistor Circuit



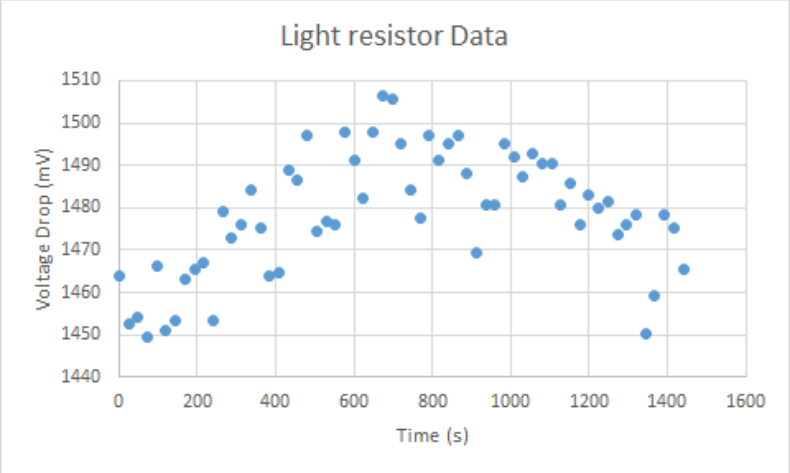
(III) Humidity Sensor Circuit

We also combined the circuit between the featherboard and the ESP8266. It sends the data from the microcontroller to the internet using UART protocol. It works with a certain order of HTTP commands shown in the code below. The code integrates the data values into the commands so that we can publish our findings online. The circuit of the ESP8266 is shown below where the Tx of the Feather is connected to the Rx of the ESP8266 and the Rx of the Feather to the Tx of the ESP8266. The ESP8266 is connected to 3.3V and GND from the board.

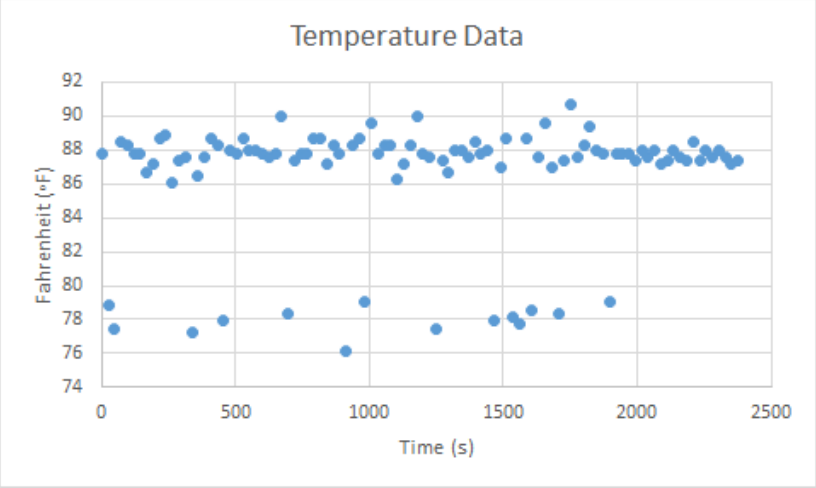


(IV) ESP8266 Circuit

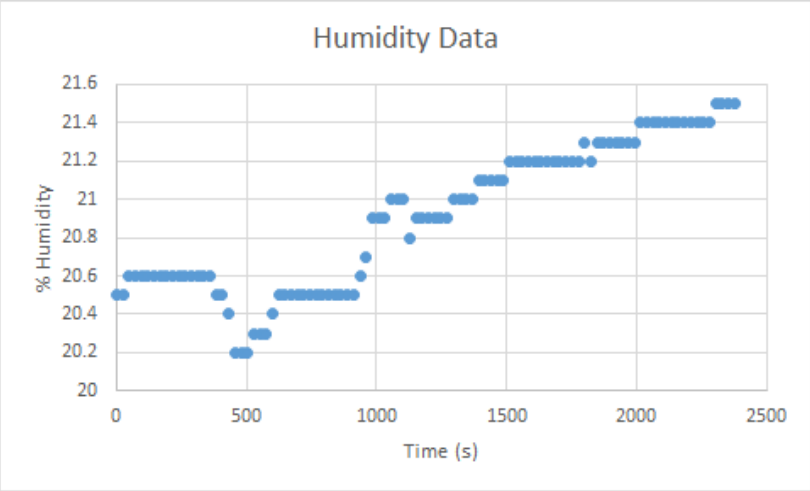
Throughout the project, we started from not a lot of information about what we were supposed to do in the project to create a sensor with photo resistance and a thermometer built into it. The photosensor gives us resistance value within ohms, depending on the condition of light, it will change value. Thermometer measures temperature originally in Celsius, but we convert the value in Celsius to Fahrenheit within our code. It measures the temperature based on how much voltage it drops at a certain temperature. We later on add on to this sensor's data, knowing that it gives higher results than expected, so we decide to calibrate it using a thermometer. After we ran the code, the feather was sending some random values. The wiring of the circuit is a tough part of the project. Throughout the project, we spent hours just on the wiring. We also had trouble working with the ESP8266 through the Feather, so we took several troubleshooting tests with the AD2 to see if we got feedback from the ESP8266. After many tests, we had our code and circuits working.



(V) Data reading from the light resistor



(VI) Data reading from the thermistor

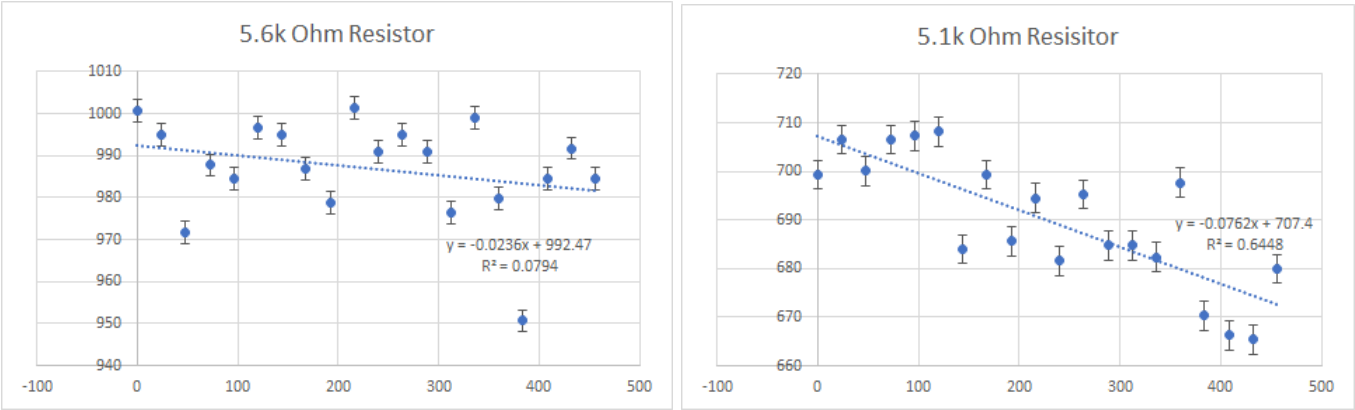


(VII) Data reading from the humidity sensor

The results come from the ThingSpeak Channel for 3 fields. Figure V represents the data produced by the light sensor in mV. Figure VI is the graph of the thermistor's data in Fahrenheit, and Figure VII is the graph of the humidity sensor's data as a percentage relative to normal humidity.

(Add-on) calibration of thermometer:
When the data is first sent onto the thingspeak, my data is around 80 plus or minus 4 Celsius, which is extremely high as the room temperature. I convert Celsius data into Fahrenheit using the equation of "`temperature = 9 * (Vout / .01) / 5 + 32`". After this conversion, I get a value consistent around 111 plus or minus 5 Fahrenheit. Then I looked at the thermometer from our class. It displays the same value of 74 Fahrenheit all the time. So, I looked at the data sent online for two minutes (it sent the value of temperature every 30 second), it was reading values of (111, 112.40, 113.53, and 108.34 Fahrenheit). So, I roughly calculate the average value of those numbers and use math equation "`(9 * (Vout / .01) / 5 + 32) *73/111.3175`" to calibrate data in consistent 73 plus or minus 1 Fahrenheit (room temperature on the thermometer for our lab room). Even though sometimes the data drop to around 67 Fahrenheit for two or three data, the data still went back to 73 plus or minus 1 Fahrenheit.

(Add-on) Data Precision:
We also worked on precision where we used different resistors to produce smoother results for the photo sensor. The data was measured for each resistor of 4.3k, 4.5k, 5.1k, 5.6k, and 6.2k Ohms. The best precise results were given by the 5.6k Ohms where the standard deviation was the lowest out of all the resistors and gave the least oscillation.



(VIII) Data readings from light resistor using a 5.6kΩ resistor v 5.1kΩ resistor

As displayed, the slope is almost near 0 where it aims for perfection in maintaining consistent results. There are more points closer to the line than there are farther from it. In addition, we would add a scale to see when the room is dark, normal lit, and very bright. We would use the same calibration approach from the thermistor to dictate between those scenarios.

Reflection:
During the project, we had faced a lot of problems. When I first ran the code, the data of both the photo sensor and thermistor were way off from the actual value. The photosensor was supposed to send values around 2200 ohms, but it sent values around 200 ohms. The temperature sensor keeps sending a consistent value of around 80 plus or minus 5 Celsius. For the photosensor, I found out that I forgot to change the value of 3.3V to 3300 mV. For the thermometer, I inserted a code to convert Celsius value to Fahrenheit and one time I had spent two hours on the and circuit and code, just to figure out why photo resistance is sending value of zero onto thingspeak. In the end, it is just because I didn't insert the reference resistance onto the photo sensor in order to let the feather read the resistance value of the photo sensor. These simple things on the circuit usually made us spend hours on the project (connect sensor into a different pin in feather from our code.) Overall, we learn much about electrical devices and how they work. We had a great endeavor of troubleshooting and identifying the problems to find a solution. Our answers in the preamble were answered with understanding but can be further explained with more experience.

References:
"Adafruit AM2320 Sensor," *Python & CircuitPython | Adafruit AM2320 Sensor | Adafruit Learning System*. [Online]. Available: https://learn.adafruit.com/adafruit-am2320-temperature-humidity-i2c-sensor/python-circuitpython. [Accessed: 05-Apr-2019].

Brandon’s code

```
import busio             #module setup
import time
import board
import adafruit_am2320
from analogio import AnalogIn

uart = busio.UART(board.TX, board.RX, baudrate=115200) #uart setup
i2c = busio.I2C(board.SCL, board.SDA)       #humidity sensor-i2c protocol
am = adafruit_am2320.AM2320(i2c)

temperature = 0           #parameters
Vout = 0                  #LM35 voltage drop
Vout2 = 0                 #Photocell voltage drop

analog_in = AnalogIn(board.A0)              #pins setup
analog_in2 = AnalogIn(board.A1)


def get_value(pin):               #voltage values setup for each pin
    return (pin.value)

def find():              #read uart feedback function
    if data is not None:
        data_string = ''.join([chr(b) for b in data])
        print(data_string, end="")
        time.sleep(3)
    return

while True:
    Vout = (int(get_value(analog_in)) * 5 / 65536)   #Vdrop from LM35
    Vout2 = ((int(get_value(analog_in2)) * 3300) / 65536) #photocell Vdrop
    temperature = 9 * (Vout / .01) / 5 + 32  #Convert LM35 Vdrop to Fahren
    data = uart.readline() #uart read all data feedback

    uart.write('AT+RST\r\n')                #HTTP Commands
    find()

    uart.write('AT+CWMODE=1\r\n')
    find()

    uart.write('AT+CWJAP="bucknell_iot",""\r\n')
    find()
```

```
    uart.write('AT+CIPMODE=0\r\n')
    find()

    uart.write('AT+CIPSTART="TCP","api.thingspeak.com",80\r\n')
    find()

    uart.write('AT+CIPSEND=150\r\n')
    find()

    uart.write("GET
/update?api_key=WESYI6EBFKIZP22L&field2="+str(Vout2)+"&field3="+str(temper
ature)+"&field4="+str(am.relative_humidity)+"\r\n")
    find()

    uart.write('AT+CIPCLOSE\r\n')
    find()
```

Kevin’s code:

```
import busio                 # module setup
import time
import board
#import adafruit_am2320
from analogio import AnalogIn

uart = busio.UART(board.TX, board.RX, baudrate=115200)
#i2c = busio.I2C(board.SCL, board.SDA)
#am = adafruit_am2320.AM2320(i2c)

temperature = 0
Vout = 0
Vout2 = 0

analog_in = AnalogIn(board.A1)
analog_in2 = AnalogIn(board.A0)
COMM1 = "GET
/update?api_key=7RV0SRFJ4TS76T66&field3="+str(int(temperature))+"\r\n"
COMM2 = "GET
/update?api_key=7RV0SRFJ4TS76T66&field2="+str(int(Vout2))+"\r\n"
```

```
byt = len(COMM1)
byt2 = len(COMM2)


def get_value(pin):
    return (pin.value)

def find():
    if data is not None:
        data_string = ''.join([chr(b) for b in data])
        print(data_string, end="")
        time.sleep(3)
    return

while True:
    Vout = (int(get_value(analog_in)) * 5 / 65536)
    Vout2 = ((int(get_value(analog_in2)) * 3300) / 65536)
    temperature = (9 * (Vout / .01) / 5 + 32)*74/111
    data = uart.readline()

    uart.write('AT+RST\r\n')
    find()

    uart.write('AT+CIPMODE=0\r\n')
    find()

    uart.write('AT+CIPSTART="TCP","api.thingspeak.com",80\r\n')
    find()

    uart.write('AT+CIPSEND=150\r\n')
    find()

      uart.write("GET/update?api_key=7RV0SRFJ4TS76T66&field2="+str(Vout2)+
      "&field3="+str(temperature)+"\r\n")
    find()

    uart.write('AT+CIPCLOSE\r\n')
    find()
```