Assignment 3

CS 300

Daniel Michaels and Reuben Lewis

**Raspberry Pi Code:**

import RPi.GPIO as GPIO

import time

import paho.mqtt.client as mqtt

GPIO.setmode(GPIO.BCM)

SCL = 3

SDA = 2

icAddress = 0x18

registerAddress = 0x05

# Constants

BROKER = "test.mosquitto.org"

# BROKER = "iot.cs.calvin.edu"

PORT = 1883

QOS = 0

#IC Address is converted into Binary with R/W bit appended

icAddress = bin(icAddress)[2:]

icAddress = icAddress.zfill(7)

icAddress = [int(x) for x in icAddress]

icAddress.append(0)

#Register address is converted to binary

registerAddress = bin(registerAddress)[2:]

registerAddress = registerAddress.zfill(8)

registerAddress = [int(x) for x in registerAddress]

#getTemperature() uses sub-functions to bit bang the I2C for read

def getTemperature():

#Global Constants

global SCL, SDA, icAddress, registerAddress

#Set SCL for Output

GPIO.setup(SCL, GPIO.OUT)

#Send Start Sequence

sendStart()

#Write icAddress w/ WRITE Bit

icAddress[7] = 0

#Attempts to initiate I2C device at icAddress

if(writeData(icAddress) == False):

print("Address not Acknowledged by Sensor for initial write !!")

#Attempts to set sensor's register pointer to register Address

if(writeData(registerAddress) == False):

print("Register not Acknowledged by Sensor!!")

#Sends start sequence to initiate read

sendStart()

#Change R/W bit to Read

icAddress[7] = 1

#Tells I2C sensor that it is being communicated with

if(writeData(icAddress) == False):

print("Address not Acknowledged by Sensor for Data Read !!")

#Reads the two bytes seperately and acknowledes their reception

msbData = readData()

sendACK()

lsbData = readData()

sendNAK()

#Termina Communication with sensor

sendStop()

#Convert the binary list into a number and clear flags

UpperByte = ''.join(str(x) for x in msbData)

UpperByte = int(hex(int(UpperByte, 2))[2:] , 16)

UpperByte = UpperByte & 0x1F

LowerByte = ''.join(str(x) for x in lsbData)

LowerByte = int(hex(int(LowerByte, 2))[2:] , 16)

#Performs conversion to Celsius

if ((UpperByte & 0x10) == 0x10):

UpperByte = UpperByte & 0x0F

Temperature = 256 - (UpperByte \* 16 + LowerByte / 16)

else:

Temperature = (UpperByte \* 16 + LowerByte / 16)

print("Temperature in Celsius: ", Temperature)

return Temperature

#Start Sequence (toggles SCL and SDA per Timing Diagram)

def sendStart():

global SCL, SDA, icAddress

GPIO.setup(SDA, GPIO.OUT)

GPIO.output(SDA, True)

GPIO.output(SCL, True)

time.sleep(0.001)

GPIO.output(SDA, False)

time.sleep(0.001)

GPIO.output(SCL, False)

#Toggles SCL and SDA per Timing Diagram to write some list, data

def writeData(data):

global SCL, SDA, icAddress

GPIO.setup(SDA, GPIO.OUT)

for i in range(0,len(data)):

GPIO.output(SDA, data[i])

time.sleep(0.001)

GPIO.output(SCL, True)

time.sleep(0.001)

GPIO.output(SCL, False)

time.sleep(0.001)

#Check ACK

GPIO.setup(SDA, GPIO.IN)

GPIO.output(SCL, True)

time.sleep(0.001)

ackNak = GPIO.input(SDA)

GPIO.output(SCL, False)

if ackNak == True:

return False

else:

return True

#Toggles SCL and reads SDA per timing diagram for sensor register read

def readData():

GPIO.setup(SDA, GPIO.IN)

readByte = []

for bit in range(0, 8):

GPIO.output(SCL, True)

time.sleep(0.001)

readByte.append(GPIO.input(SDA))

GPIO.output(SCL, False)

time.sleep(0.001)

return readByte

#Sequence of IO toggles which communicate an acknowledge of reception of data

def sendACK():

GPIO.setup(SDA, GPIO.OUT)

GPIO.output(SDA, False)

time.sleep(0.001)

GPIO.output(SCL, True)

time.sleep(0.001)

GPIO.output(SCL, False)

time.sleep(0.001)

#Sequence of IO toggles which communicate an acknowledge end of read op

def sendNAK():

GPIO.setup(SDA, GPIO.OUT)

GPIO.output(SDA, True)

time.sleep(0.001)

GPIO.output(SCL, True)

time.sleep(0.001)

GPIO.output(SCL, False)

time.sleep(0.001)

#Like start, this sequence indicates the termination of communication

def sendStop():

global SCL, SDA, icAddress

GPIO.setup(SDA, GPIO.OUT)

GPIO.output(SDA, False)

time.sleep(0.001)

GPIO.output(SCL, True)

time.sleep(0.001)

GPIO.output(SDA, True)

time.sleep(0.001)

GPIO.output(SCL, False)

# Callback when a message is published

def on\_publish(client, userdata, mid):

print("data published")

# Callback when a connection has been established with the MQTT broker

def on\_connect(client, userdata, rc, \*extra\_params):

print('Connected with result code='+str(rc))

if \_\_name\_\_ == '\_\_main\_\_':

# Setup MQTT client

client = mqtt.Client()

client.on\_connect = on\_connect

client.on\_publish = on\_publish

# Connect to MQTT broker and subscribe to the button topic

client.connect(BROKER, PORT, 60)

#Infinite loop to read and publish sensor data until KeyboardInterrupt

while (True):

try:

message = getTemperature()

# Publish message to broker

(result, num) = client.publish("dwm5/temperature", message, qos=0, retain=True)

if result != 0:

print("PUBLISH returned error:", result)

time.sleep(2)

except KeyboardInterrupt:

break

client.disconnect()

GPIO.cleanup()

**Raspberry Pi Code:**

import paho.mqtt.client as mqtt

# ------ Globals -------------

BROKER = "iot.cs.calvin.edu"

PORT = 1883

QOS = 0

TOPIC = 'dwm5/temperature'

temp = 0

# ------ Functions -------------

def c\_to\_f(c):

return c \* 9.0 / 5.0 + 32.0

def on\_connect(client, userdata, rc, \*extra\_params):

print("Connection to " + BROKER + " with result code !" + str(rc))

def on\_temp\_msg(client, data, msg):

global temp

if msg.topic == TOPIC:

temp = msg.payload

if \_\_name\_\_ == '\_\_main\_\_':

mqtt\_client = mqtt.Client()

mqtt\_client.on\_message = on\_temp\_msg

mqtt\_client.on\_connect = on\_connect

mqtt\_client.connect(BROKER, PORT, 60)

mqtt\_client.loop\_start()

mqtt\_client.subscribe(TOPIC, qos=QOS)

old\_temp = 0

print('Getting the temp from pi!')

while True:

try:

if temp != old\_temp:

print('The current temperature is %s C, or %s F' % (temp, c\_to\_f(float(temp))))

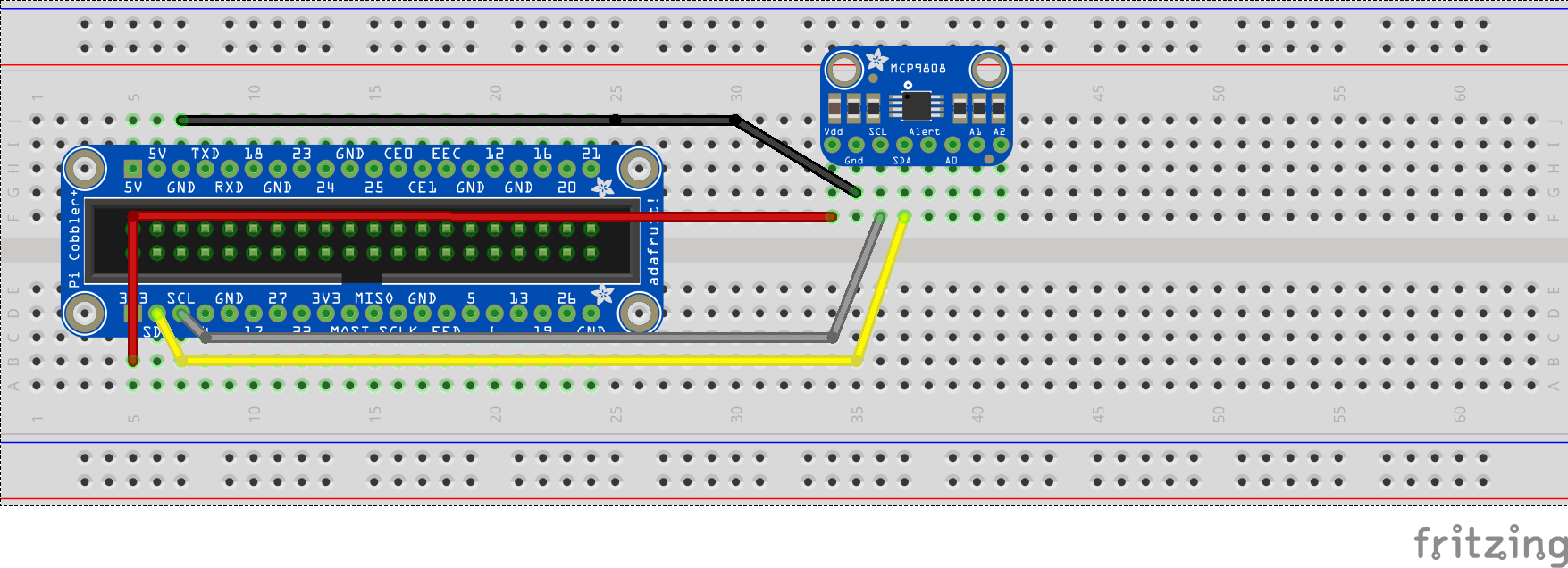
old\_temp = temp

except KeyboardInterrupt:

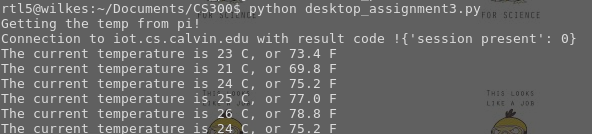
break

mqtt\_client.disconnect()

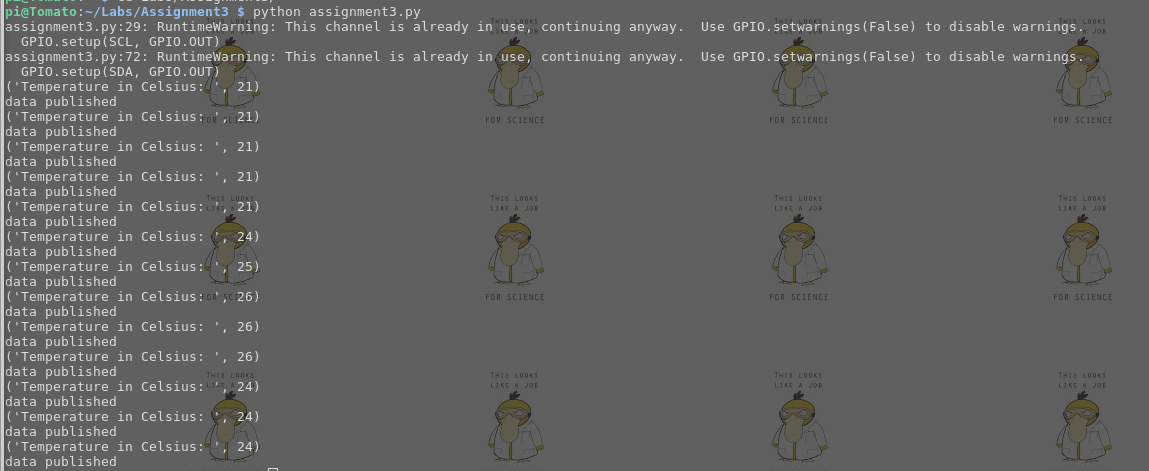
**Fritzing:**



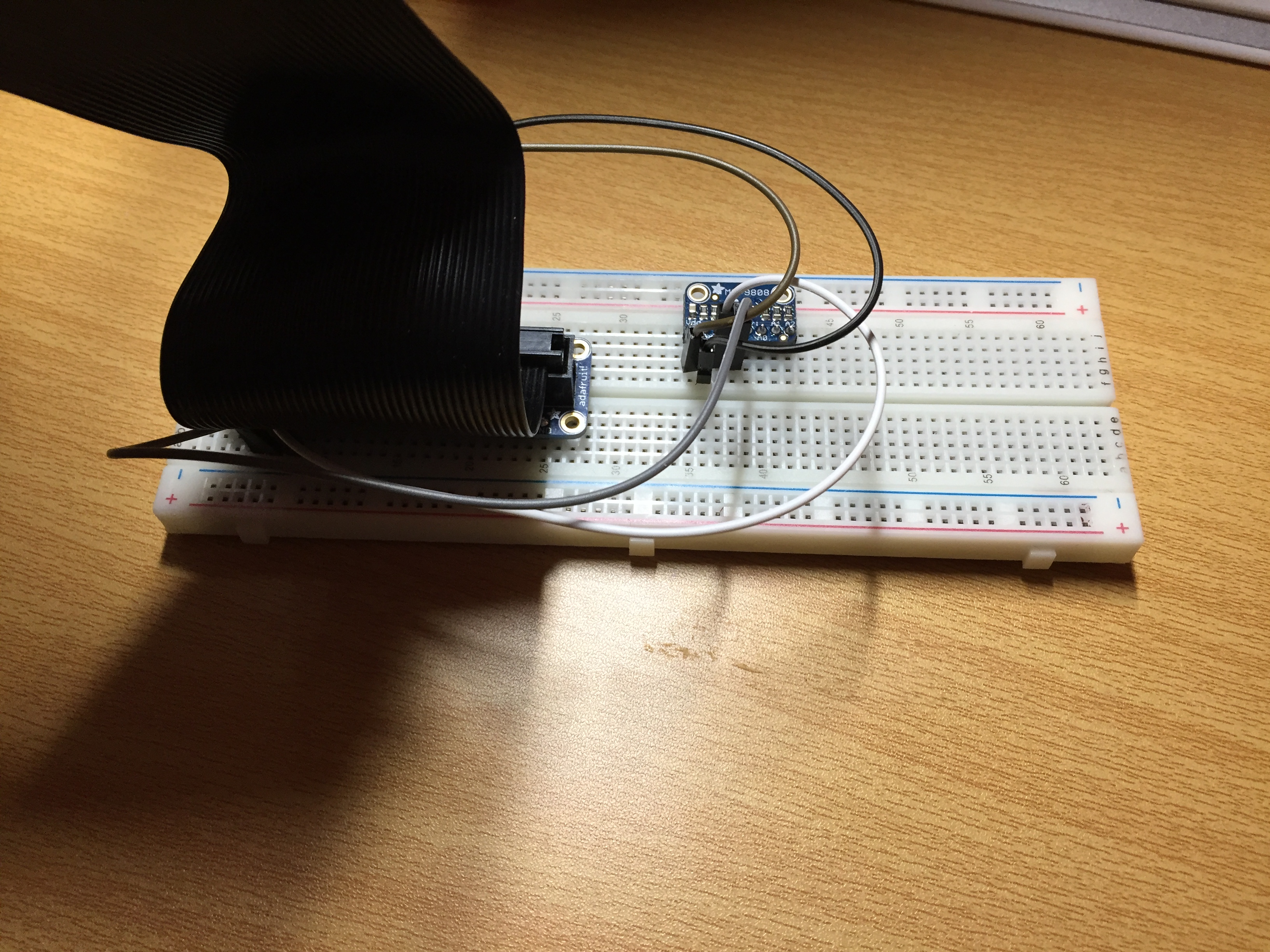
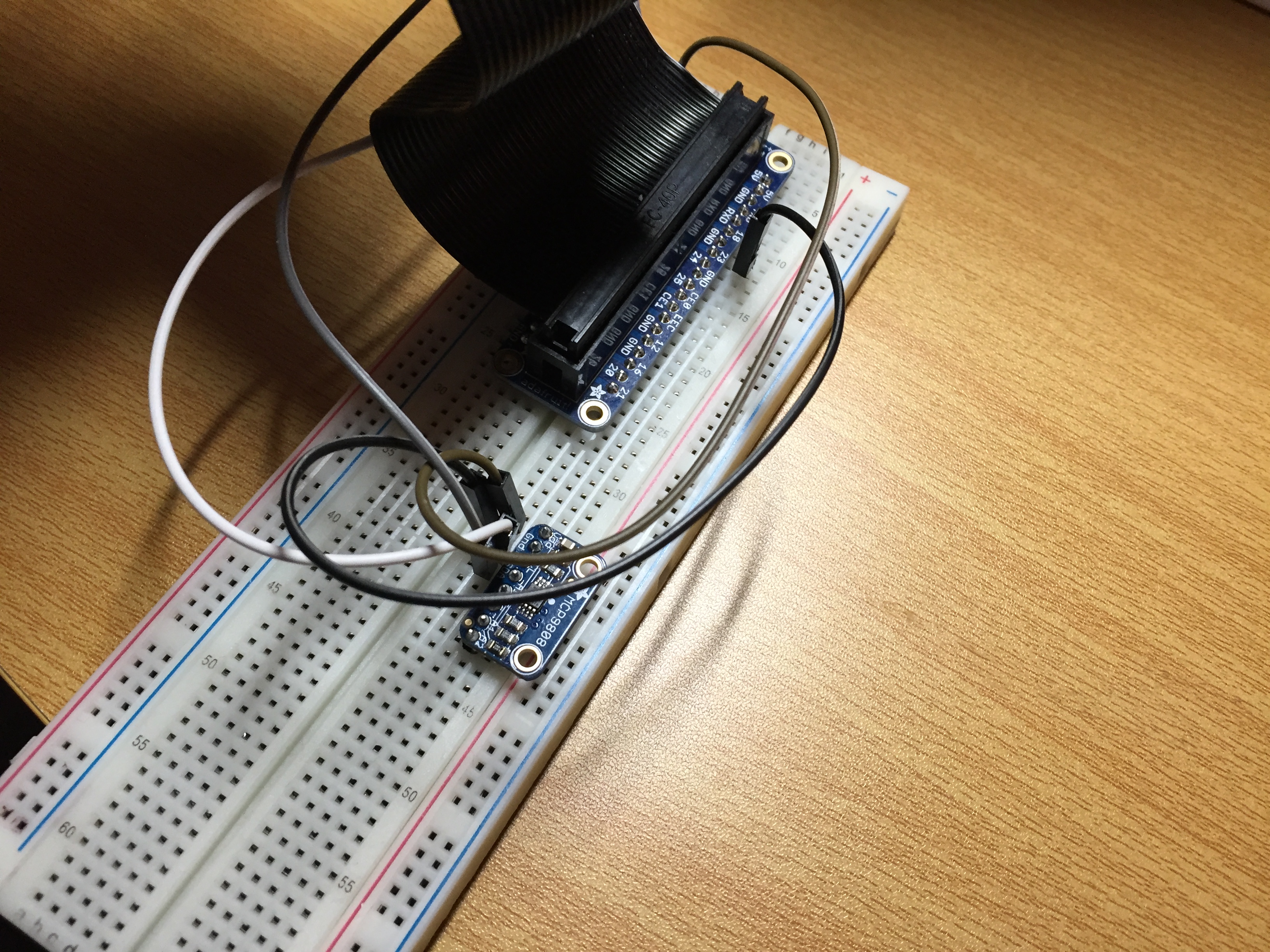
**Desktop App Output:**



**Pi Output/MQTT Publishing:**



**Actual Wiring:**

****