CPE301 – SPRING 2019

MIDTERM II

Student Name: Billy Maddex

Student #: 2000928390

Student Email: [maddex@unlv.nevada.edu](mailto:maddex@unlv.nevada.edu)

Primary Github address: [maddex@unlv.nevada.edu](mailto:maddex@unlv.nevada.edu)

Directory: Midterm2

Submit the following for all Labs:

1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/Midterm, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with youtube video links (see template).

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

Atmega328PB-XMINI

APDS9960 → PINC4/PINC5

ESP01 WiFi Module → PIND0/PIND1

1. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1**

/\*

\* Final.c

\*

\* Created: 12/12/2019 6:47:07 PM

\* Author : Billy

\*/

// 16MHz CPU clock

#define F\_CPU 16000000UL

// 115200 baud rate setting

#define URRR\_115200 8

#include <avr/io.h>

#include <util/atomic.h>

#include <util/delay.h>

#include <stdlib.h>

#include <stdio.h>

#include "i2cmaster.h"

// APDS read and write addresses

#define APDS9960\_WRITE 0x72

#define APDS9960\_READ 0x73

// 15s timer variable, interval is 14 because we count from 0

const int INTERVAL = 14;

int transmit\_time;

// variables to hold the sensor data

int16\_t raw\_R = 0;

int16\_t raw\_G = 0;

int16\_t raw\_B = 0;

int16\_t raw\_C = 0;

// variable to control data field for looping upload

// since only 1 value can be uploaded on each 15s interval, loop through all 4 of them

int field = 1;

// terminal output function declarations

void uart\_init(unsigned int ubrr);

void uart\_send\_string(char \*data);

// timer setup declaration

void timer\_init(void);

// ESP01 setup declaration

void esp01\_init(void);

// function to upload values

void upload\_value(uint16\_t value);

// APDS9960 sensor initialization

void init\_APDS9960(void);

// get readings from the APDS9960 sensor

void getreading(void);

// TIMER1 CTC interrupt handler

ISR(TIMER1\_COMPA\_vect)

{

// when transmit time is up, do something

if (transmit\_time-- == 0)

{

// send a value based on current field rotation

if (field == 1)

{

// red => field 1

upload\_value(raw\_R);

field++;

}

else if (field == 2)

{

// blue => field 2

upload\_value(raw\_B);

field++;

}

else if (field == 3)

{

// green => field 3

upload\_value(raw\_G);

field++;

}

else if (field == 4)

{

// ambient light => field 4

upload\_value(raw\_C);

field = 1;

}

// reset transmit\_time

transmit\_time = INTERVAL;

}

}

int main(void)

{

// configure terminal, baud rate 115200

uart\_init(URRR\_115200);

// configure the ESP01 module

esp01\_init();

// initialize i2c

i2c\_init();

// initialize sensor

init\_APDS9960();

// start the timer

timer\_init();

// enable global interrupt

sei();

// loop forever

while (1)

{

getreading();

\_delay\_ms(1000);

}

}

void upload\_value(uint16\_t value)

{

// output strings

char data\_size[20];

char report[80];

int report\_len = 0;

// generate output string from 'value' and set to upload to 'field' variable # field

snprintf(report, sizeof(report), "GET /update?api\_key=RGW36KTS9JXT0XM9&field%d=%d\r\n\r\n", field, value);

// get number of bytes to send

while (report[report\_len] != '\0')

{

report\_len++;

}

// generate SEND AT string based on byte size of report

snprintf(data\_size, sizeof(data\_size), "AT+CIPSEND=%d\r\n", report\_len);

// output to terminal

ATOMIC\_BLOCK(ATOMIC\_FORCEON)

{

uart\_send\_string("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n");

uart\_send\_string(data\_size);

uart\_send\_string(report);

}

}

// get readings from the APDS9960 sensor

void getreading(void){

i2c\_start(APDS9960\_WRITE);

// set pointer to CDATAL

i2c\_write(0x94);

i2c\_stop();

// read CDATAL

i2c\_start(APDS9960\_READ);

raw\_C = ((uint8\_t)i2c\_readNak());

i2c\_stop();

i2c\_start(APDS9960\_WRITE);

// set pointer to CDATAH

i2c\_write(0x95);

i2c\_stop();

// read CDATAH

i2c\_start(APDS9960\_READ);

raw\_C |= ((uint8\_t)i2c\_readNak())<<8;

i2c\_stop();

i2c\_start(APDS9960\_WRITE);

// set pointer to RDATAL

i2c\_write(0x96);

i2c\_stop();

// read RDATAL

i2c\_start(APDS9960\_READ);

raw\_R = ((uint8\_t)i2c\_readNak());

i2c\_stop();

i2c\_start(APDS9960\_WRITE);

// set pointer to RDATAH

i2c\_write(0x97);

i2c\_stop();

// read RDATAH

i2c\_start(APDS9960\_READ);

raw\_R |= ((uint8\_t)i2c\_readNak())<<8;

i2c\_stop();

i2c\_start(APDS9960\_WRITE);

// set pointer to GDATAL

i2c\_write(0x98);

i2c\_stop();

// read GDATAL

i2c\_start(APDS9960\_READ);

raw\_G = ((uint8\_t)i2c\_readNak());

i2c\_stop();

i2c\_start(APDS9960\_WRITE);

// set pointer to GDATAH

i2c\_write(0x99);

i2c\_stop();

// read GDATAH

i2c\_start(APDS9960\_READ);

raw\_G |= ((uint8\_t)i2c\_readNak())<<8;

i2c\_stop();

i2c\_start(APDS9960\_WRITE);

// set pointer to BDATAL

i2c\_write(0x9A);

i2c\_stop();

// read BDATAL

i2c\_start(APDS9960\_READ);

raw\_B = ((uint8\_t)i2c\_readNak());

i2c\_stop();

i2c\_start(APDS9960\_WRITE);

// set pointer to BDATAH

i2c\_write(0x9B);

i2c\_stop();

// read BDATAH

i2c\_start(APDS9960\_READ);

raw\_B |= ((uint8\_t)i2c\_readNak())<<8;

i2c\_stop();

}

// configure the APDS9960 sensor

void init\_APDS9960(void)

{

// power up delay 150ms

\_delay\_ms(150);

// Enable Register, power on the device and the ALS

i2c\_start(APDS9960\_WRITE);

i2c\_write(0x80);

i2c\_write(0x03);

i2c\_stop();

// ADC Integration Time Register, 200ms, 72 cycles

i2c\_start(APDS9960\_WRITE);

i2c\_write(0x81);

i2c\_write(0xB6);

i2c\_stop();

// Control Register One, ALS and color gain = 1x

i2c\_start(APDS9960\_WRITE);

i2c\_write(0x8F);

i2c\_write(0x00);

i2c\_stop();

}

// configure and activate the timer

void timer\_init(void)

{

// initialize transmit\_time

transmit\_time = INTERVAL;

// configure interrupt handling

// enable TIMER1 COMPA interrupt

TIMSK1 |= (1 << OCIE1A);

// set TIMER1 to CTC operations

TCCR1B |= (1 << WGM12);

// TIMER1 COMPA = X

OCR1A = 15620;

// start TIMER1 with prescalar = 1024

TCCR1B |= (1 << CS12) | (1 << CS10);

}

// configure terminal output

void uart\_init(unsigned int ubrr)

{

// set buad rate based on ubrr

UBRR0H = (unsigned char)(ubrr>>8);

UBRR0L = (unsigned char)ubrr;

// set terminal to output

UCSR0B = (1 << TXEN0);

// set output mode

UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);

}

void uart\_send\_string(char \*data)

{

while((\*data != '\0'))

{

while (!(UCSR0A & (1 <<UDRE0)));

UDR0 = \*data;

data++;

}

// required delay between ESP01 UART commands

\_delay\_ms(1800);

}

// configure ESP01 and connect to WiFi

void esp01\_init(void)

{

uart\_send\_string("AT\r\n");

uart\_send\_string("AT+CWMODE=1\r\n");

uart\_send\_string("AT+CIPSTA=\"10.7.7.22\"\r\n");

uart\_send\_string("AT+CWJAP=\"Untempered Schism\",\"rycbar11\"\r\n");

uart\_send\_string("AT+CIPMUX=0\r\n");

}

1. **DEVELOPED MODIFIED CODE OF TASK 2 from TASK 1**

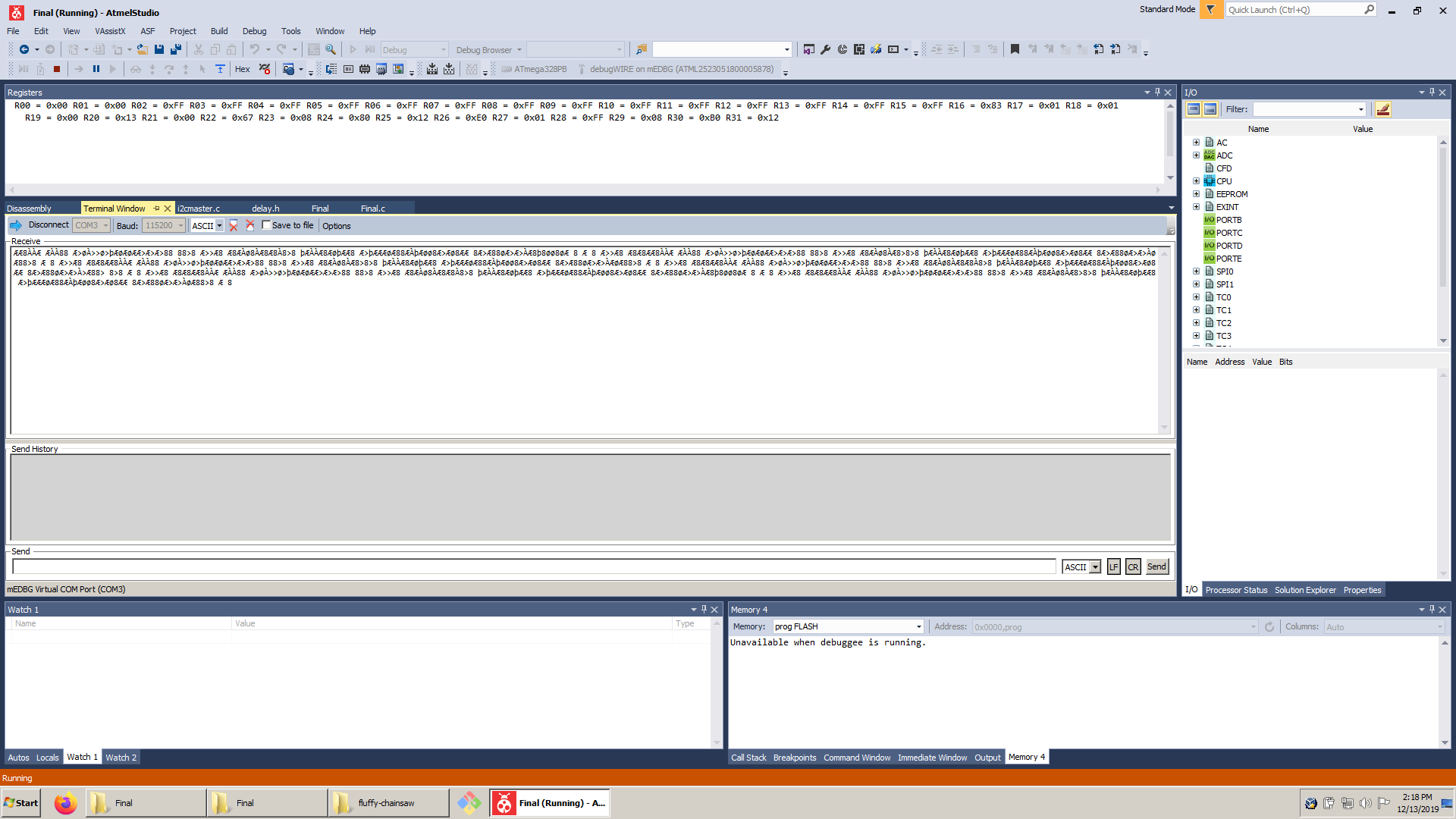
// no modified code

1. **SCHEMATICS**

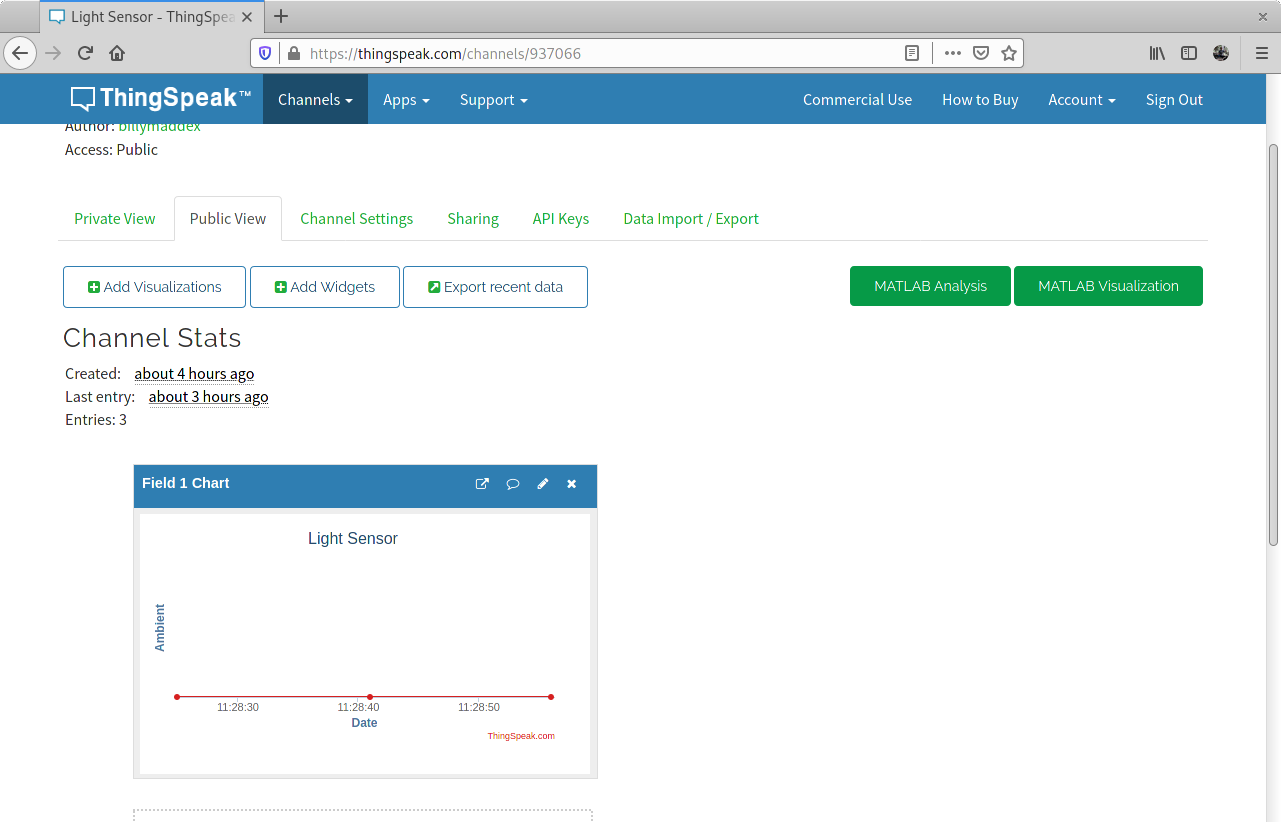
Use fritzing.org

1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

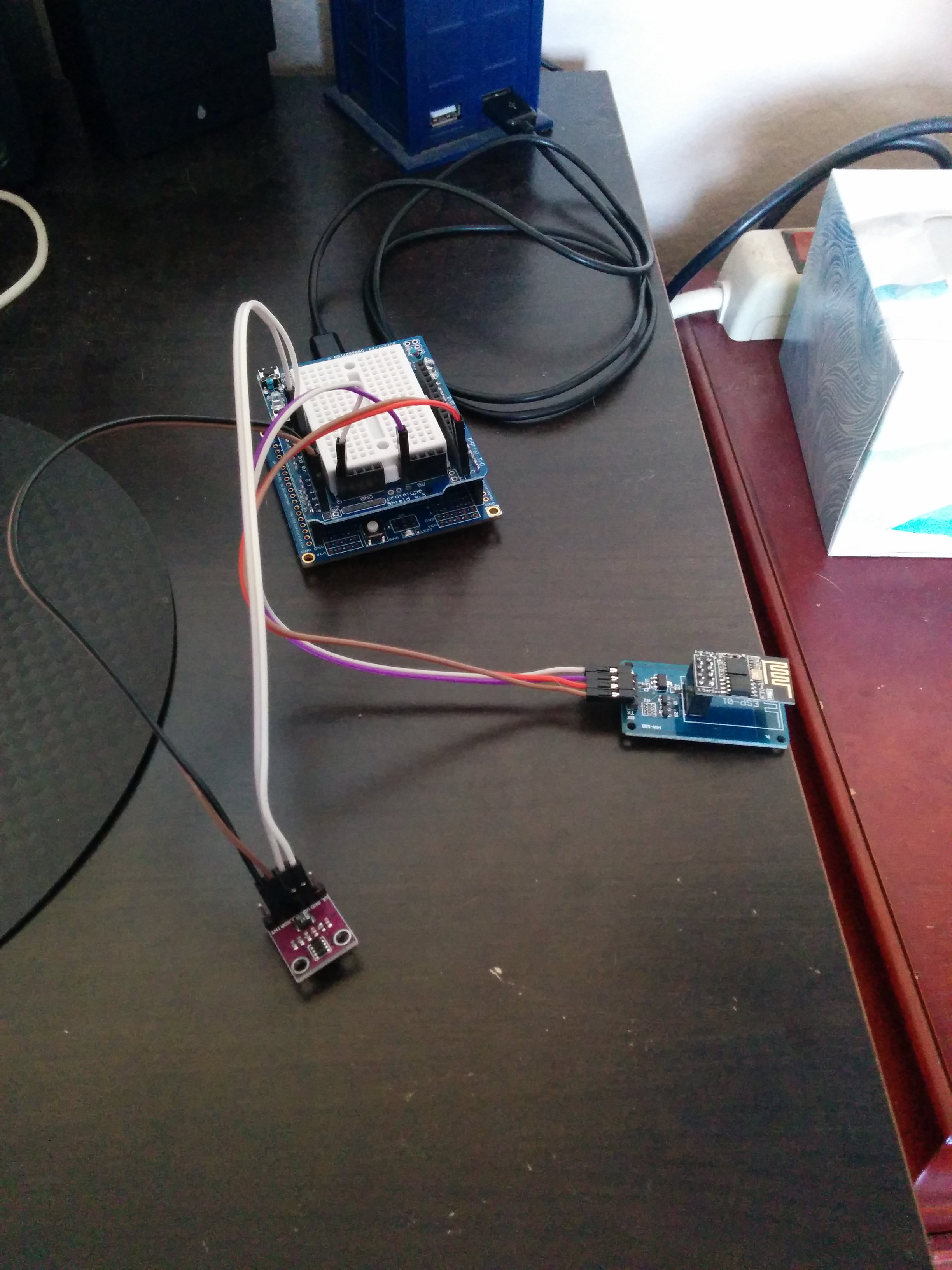
Something fried on my dev board at the end of this project, so the terminal output turned to gibberish, even on previous working code.



ESP01 Module no longer connecting to Thingspeak, tested with previously working Midterm I code but no longer functional



1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**



1. **VIDEO LINKS OF EACH DEMO**

https://youtu.be/uXP6SmCNgdc

https://thingspeak.com/channels/937066

1. **GITHUB LINK OF THIS DA**

<https://github.com/billymaddex/fluffy-chainsaw/tree/master/Midterm2>

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Billy Maddex