CPE301 – SPRING 2019

MIDTERM 2

Student Name: Cody McDonald

Student #: 5000382538

Student Email: mcdonc4@unlv.nevada.edu

Primary Github address: <https://github.com/elev8rProcrastinator>

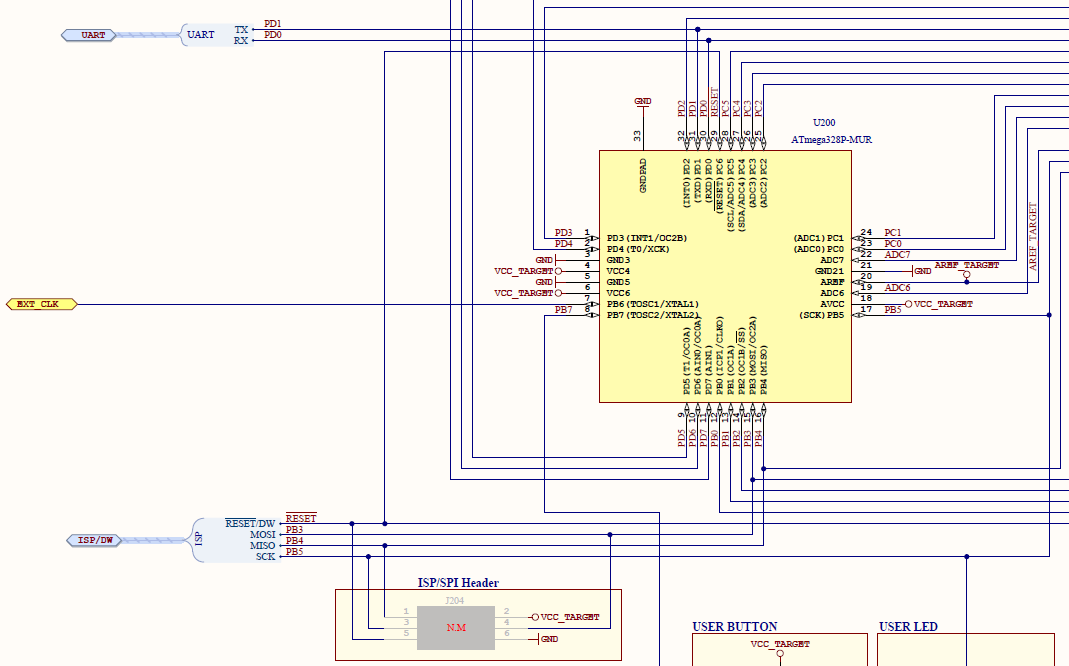
Directory: <https://github.com/elev8rProcrastinator/submission_da/tree/master/DA_Midterm>2

Submit the following for all Labs:

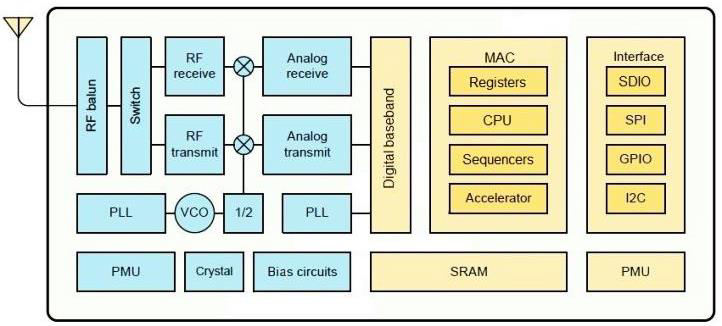
1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder 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**

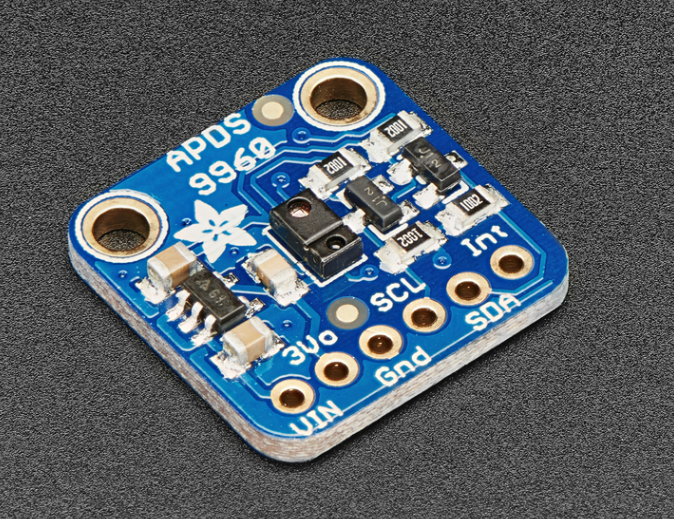
AtMini xplained:



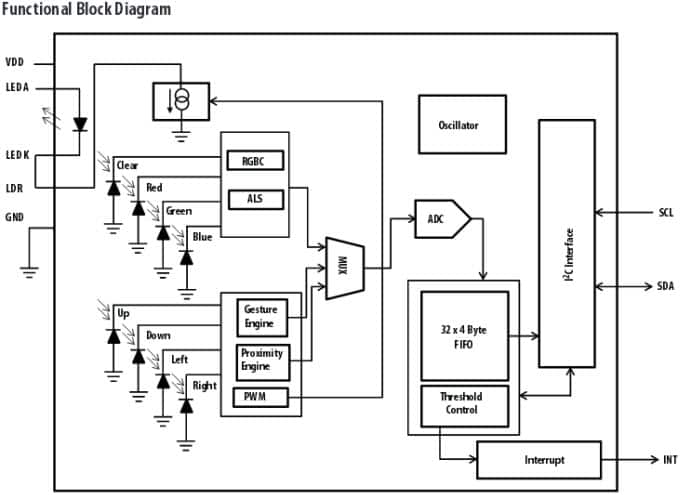
ESP8266:



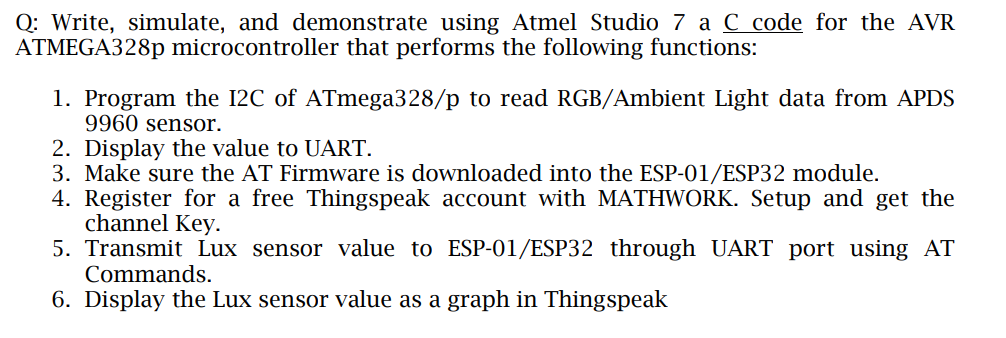
LUX RGB SENSOR(adps9660):



Functional block diagram for sensor connection:



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



/\*

\* CPE301\_Midterm2.c

\*

\* Created: 5/9/2019 12:15:07 PM

\* Author : Cody

\*/

#ifndef F\_CPU

#define F\_CPU 16000000UL

#endif

// Global constants for uart

#define BAUD 115200

#define FOSC 16000000

#define UBRR FOSC/8/BAUD-1

#define APDS9960\_WRITE 0x72

#define APDS9960\_READ 0x73

//include standard libraries

#include <avr/io.h>

#include <stdlib.h>

#include <stdio.h>

#include <util/delay.h>

#include <math.h>

//include custom libraries

#include "APDS9960\_def.h"

#include "i2c\_master.h"

//Function declarations

void getValues(void);

// void TIMER1\_init();

void init\_APDS9960(void);

void usart\_init();

void USART\_putstring(volatile unsigned char \*StringPtr);

//AT commands

volatile unsigned char AT[] = "AT\r\n"; // Test

volatile unsigned char CWMODE[] = "AT+CWMODE=3\r\n"; // Set Wi-Fi mode

volatile unsigned char CWJAP[] = "AT+CWJAP=\"Monarchs Palace\",\"quackquack\"\r\n"; // Get Wi-Fi info

volatile unsigned char CIPSTART[] = "AT+CIPSTART=\"TCP\",\"184.106.153.149\",80\r\n"; // Establish connection with ThingSpeak

volatile unsigned char CIPSEND[] = "AT+CIPSEND=104\r\n"; // Send Temperature

volatile unsigned char CIPMUX[] = "AT+CIPMUX=0\r\n"; // Enable connection

volatile unsigned char SEND\_DATA[] = "GET /update?key=ZDG1BP942G9NVEWD&field1="; // Get Write Key

volatile unsigned char RESET[] = "AT+RST\r\n"; // Get AT Firmware info

volatile unsigned char LINEBREAK[] = "\r\n"; // end of temperature transmission

volatile unsigned char CLOSE[] = "AT+CIPCLOSE\r\n";

//string for colors

volatile unsigned char RedStr[10];

volatile unsigned char GreenStr[10];

volatile unsigned char BlueStr[10];

*uint16\_t* redVal, greenVal, blueVal;

int main(void){

i2c\_init();

usart\_init(115200);

init\_APDS9960();

//Start up Esp

//Start AT communication

*\_delay\_ms*(1000);

USART\_putstring(AT); //send AT to the USART

//connect to network

*\_delay\_ms*(1000);

USART\_putstring(RESET); //reset ESP

*\_delay\_ms*(1000);

USART\_putstring(AT); //confirm communication

*\_delay\_ms*(1000);

USART\_putstring(CWMODE); //WiFi mode = 3

*\_delay\_ms*(1000);

USART\_putstring(CWJAP); //Send wifi login

while(1){

//getValues();

*\_delay\_ms*(1000);

USART\_putstring(CIPMUX); //Single connection point

*\_delay\_ms*(1000);

USART\_putstring(CIPSTART); // Connect to ThingSpeak

*\_delay\_ms*(1000);

USART\_putstring(CIPSEND); // Declare send length 50

*\_delay\_ms*(1000);

getValues();

USART\_putstring(SEND\_DATA); // Connect to proper key

USART\_putstring(RedStr); // Send adc data

USART\_putstring("&field2=");

USART\_putstring(GreenStr); // Send adc data

USART\_putstring("&field3=");

USART\_putstring(BlueStr); // Send adc data

*\_delay\_ms*(1000);

}

return 0;

}

void init\_APDS9960(void){

*uint8\_t* setup;

i2c\_readReg(APDS9960\_WRITE, APDS9960\_ID, &setup,1);

if(setup != APDS9960\_ID\_1) while(1);

setup = 1 << 1 | 1<<0 | 1<<3 | 1<<4;

i2c\_writeReg(APDS9960\_WRITE, APDS9960\_ENABLE, &setup, 1);

setup = DEFAULT\_ATIME;

i2c\_writeReg(APDS9960\_WRITE, APDS9960\_ATIME, &setup, 1);

setup = DEFAULT\_WTIME;

i2c\_writeReg(APDS9960\_WRITE, APDS9960\_WTIME, &setup, 1);

setup = DEFAULT\_PROX\_PPULSE;

i2c\_writeReg(APDS9960\_WRITE, APDS9960\_PPULSE, &setup, 1);

setup = DEFAULT\_POFFSET\_UR;

i2c\_writeReg(APDS9960\_WRITE, APDS9960\_POFFSET\_UR, &setup, 1);

setup = DEFAULT\_POFFSET\_DL;

i2c\_writeReg(APDS9960\_WRITE, APDS9960\_POFFSET\_DL, &setup, 1);

setup = DEFAULT\_CONFIG1;

i2c\_writeReg(APDS9960\_WRITE, APDS9960\_CONFIG1, &setup, 1);

setup = DEFAULT\_PERS;

i2c\_writeReg(APDS9960\_WRITE, APDS9960\_PERS, &setup, 1);

setup = DEFAULT\_CONFIG2;

i2c\_writeReg(APDS9960\_WRITE, APDS9960\_CONFIG2, &setup, 1);

setup = DEFAULT\_CONFIG3;

i2c\_writeReg(APDS9960\_WRITE, APDS9960\_CONFIG3, &setup, 1);

}

void getValues(void){

*uint8\_t* redVH, redVL;

*uint8\_t* greenVH, greenVL;

*uint8\_t* blueVH, blueVL;

unsigned char i;

char dummy[10];

// Read red value

i2c\_readReg(APDS9960\_WRITE, APDS9960\_RDATAH, &redVH, 1);

i2c\_readReg(APDS9960\_WRITE, APDS9960\_RDATAL, &redVL, 1);

// Read green value

i2c\_readReg(APDS9960\_WRITE, APDS9960\_GDATAH, &greenVH, 1);

i2c\_readReg(APDS9960\_WRITE, APDS9960\_GDATAL, &greenVL, 1);

// Read blue value

i2c\_readReg(APDS9960\_WRITE, APDS9960\_BDATAH, &blueVH, 1);

i2c\_readReg(APDS9960\_WRITE, APDS9960\_BDATAL, &blueVL, 1);

redVal = (redVH << 8) | redVL;

greenVal = (greenVH << 8) | greenVL;

blueVal = (blueVH << 8) | blueVL;

// Set max threshold values

if (redVal > 255){

redVal = 255;

}

if (greenVal > 255){

greenVal = 255;

}

if (blueVal > 255){

blueVal = 255;

}

*itoa*(redVal, dummy, 10); //convert char to ascii

for(i = 0 ; i < 10 ; i++){

RedStr[i] = dummy[i]; //move converted ascii

}

*itoa*(greenVal, dummy, 10); //convert char to ascii

for(i = 0 ; i < 10 ; i++){

GreenStr[i] = dummy[i]; //move converted ascii

}

*itoa*(blueVal, dummy, 10); //convert char to ascii

for(i = 0 ; i < 10 ; i++){

BlueStr[i] = dummy[i]; //move converted ascii

}

}

void usart\_init() {

UBRR0H = ((UBRR) >> 8);

UBRR0L = UBRR;

UCSR0A |= (1<< U2X0); // divisor baud = 8

UCSR0B |= (1 << TXEN0); // Enable transmission

UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00); // 8 bits

}

void USART\_putstring(volatile unsigned char \*StringPtr)

{

while ((\*StringPtr != '\0')){ // Until it reaches the end of the line, it will keep looping

while (!(UCSR0A & (1 << UDRE0))); // Until UDRE0 goes high, it will keep looping

UDR0 = \*StringPtr; // UDR0 register grabs the value given from the parameter

StringPtr++; // but it does it by every character as shown here

}

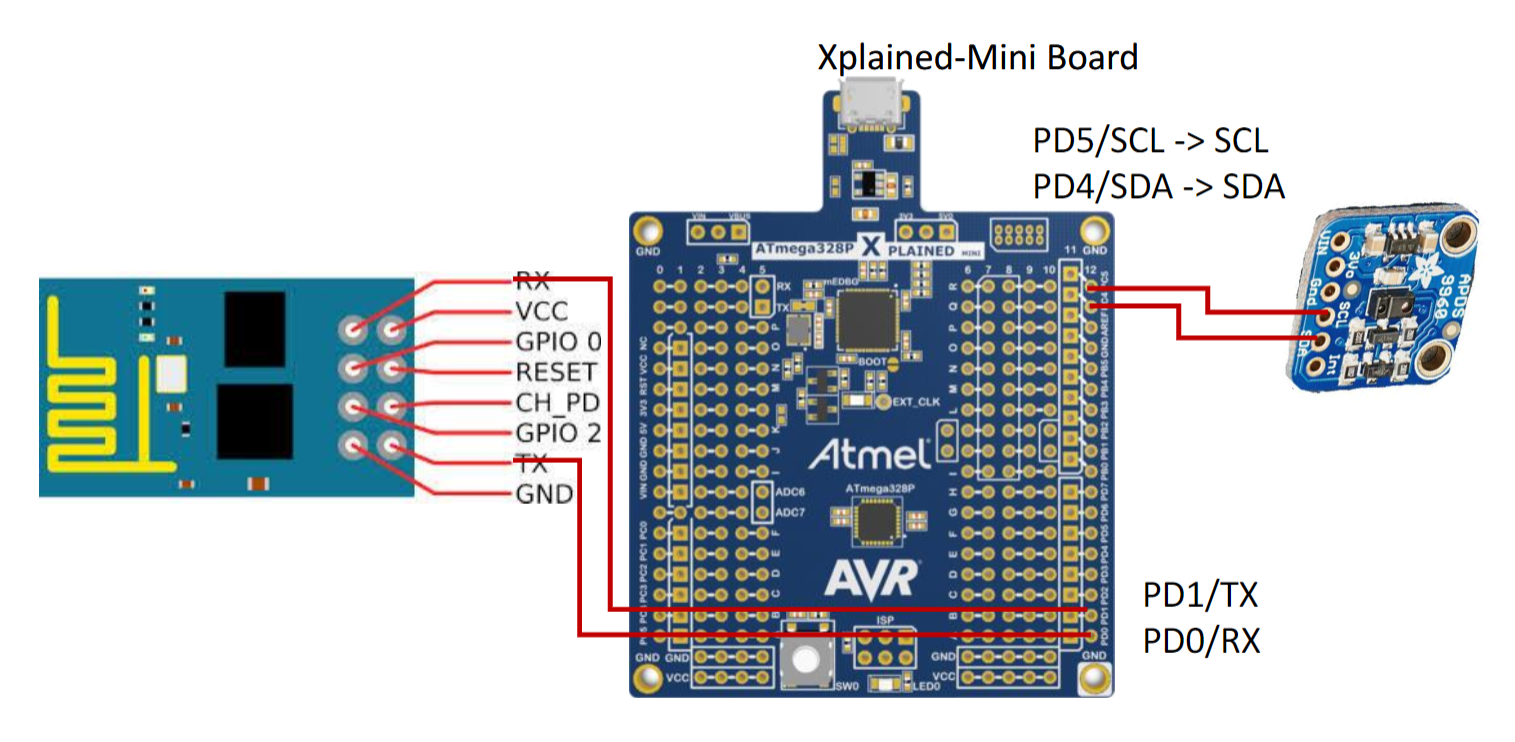
}

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

N/A

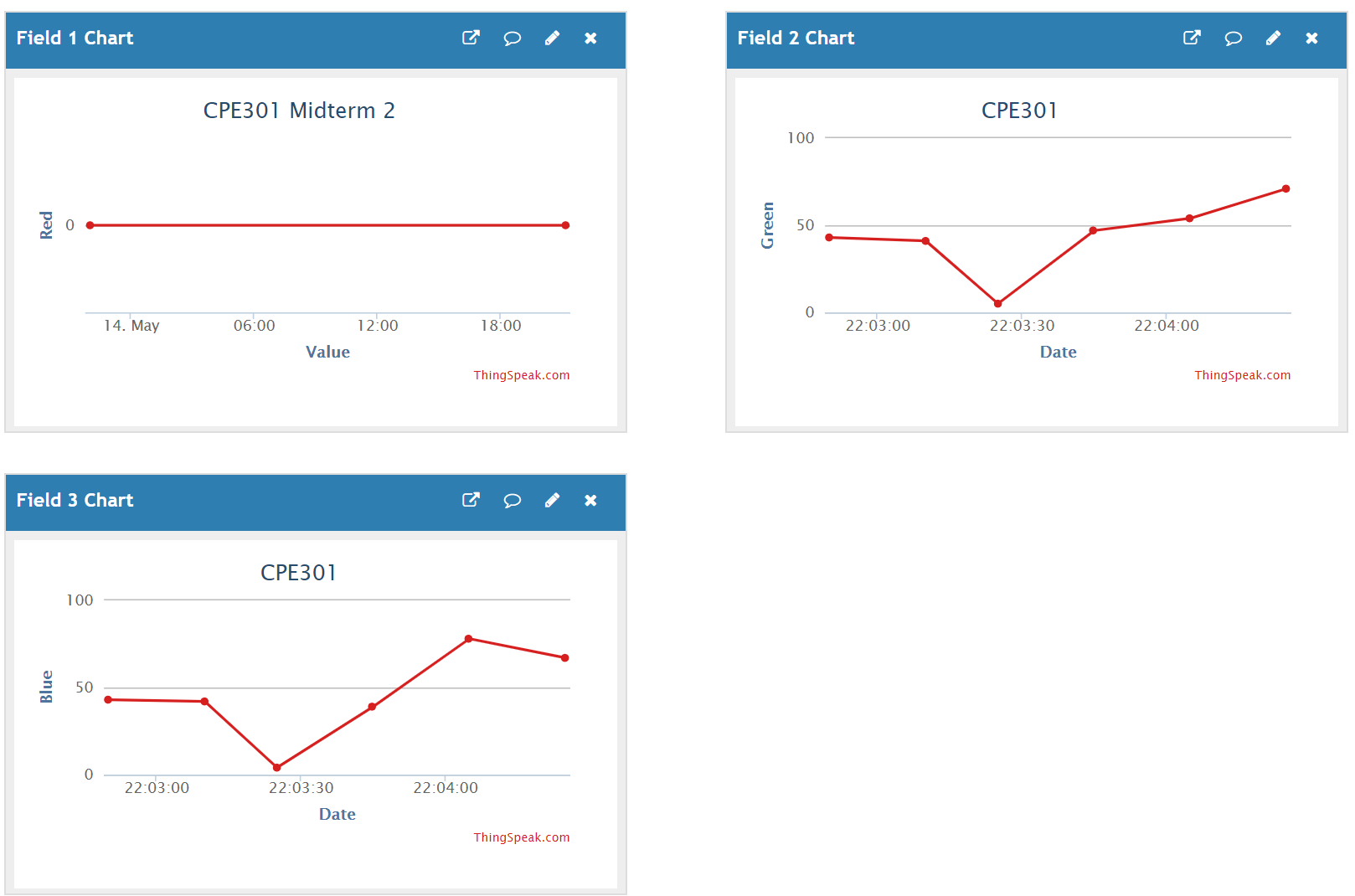
1. **SCHEMATICS**

Connection between the vital components for this assignment



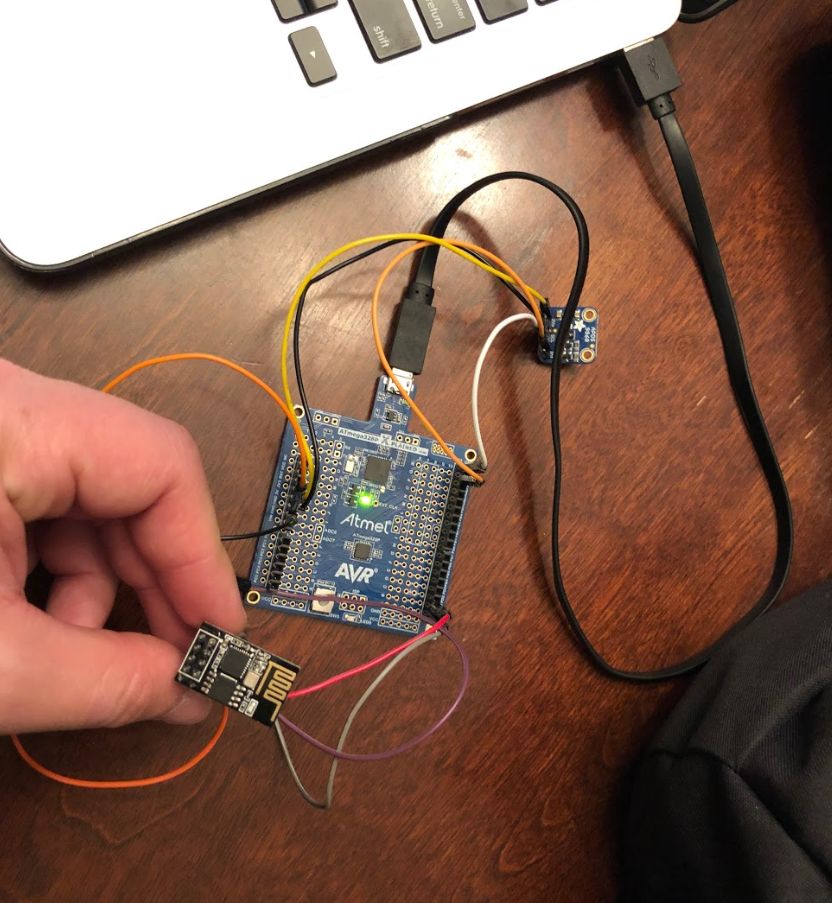
1. **SCREENSHOTS OF EACH TASK OUTPUT (THINGSPEAK OUTPUT)**

Thingspeak outputs. I may have too rough or shorted the Red sensor during setup since it doesn’t seem to output values anymore.



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

Board connection between the microcontroller, esp-01, and RGB LUX sensor:



1. **VIDEO LINKS OF EACH DEMO**

https://youtu.be/7J6V3lFsQJk

1. **GITHUB LINK OF THIS DA**

<https://github.com/elev8rProcrastinator/submission_da/tree/master/DA_Midterm>2

**Student Academic Misconduct Policy**

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

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

Cody McDonald