CPE301 – SPRING 2019

MIDTERM 2

Student Name: Chris Barr
Student #: 2000682859
Student Email: barrc1@unlv.nevada.edu
Primary Github address: https://github.com/BarrChris
Directory: https://github.com/BarrChris/submission_da.git

Submit the following for all Labs:

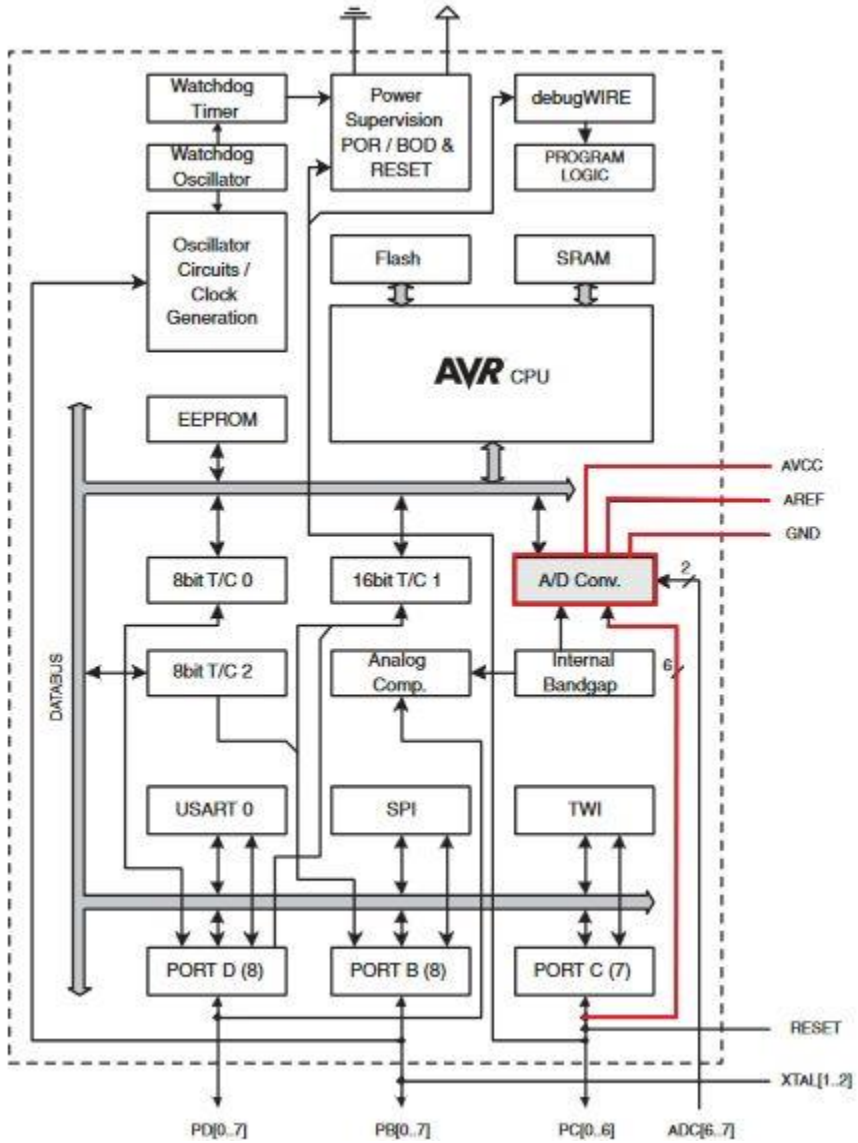
1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder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

List of Components used

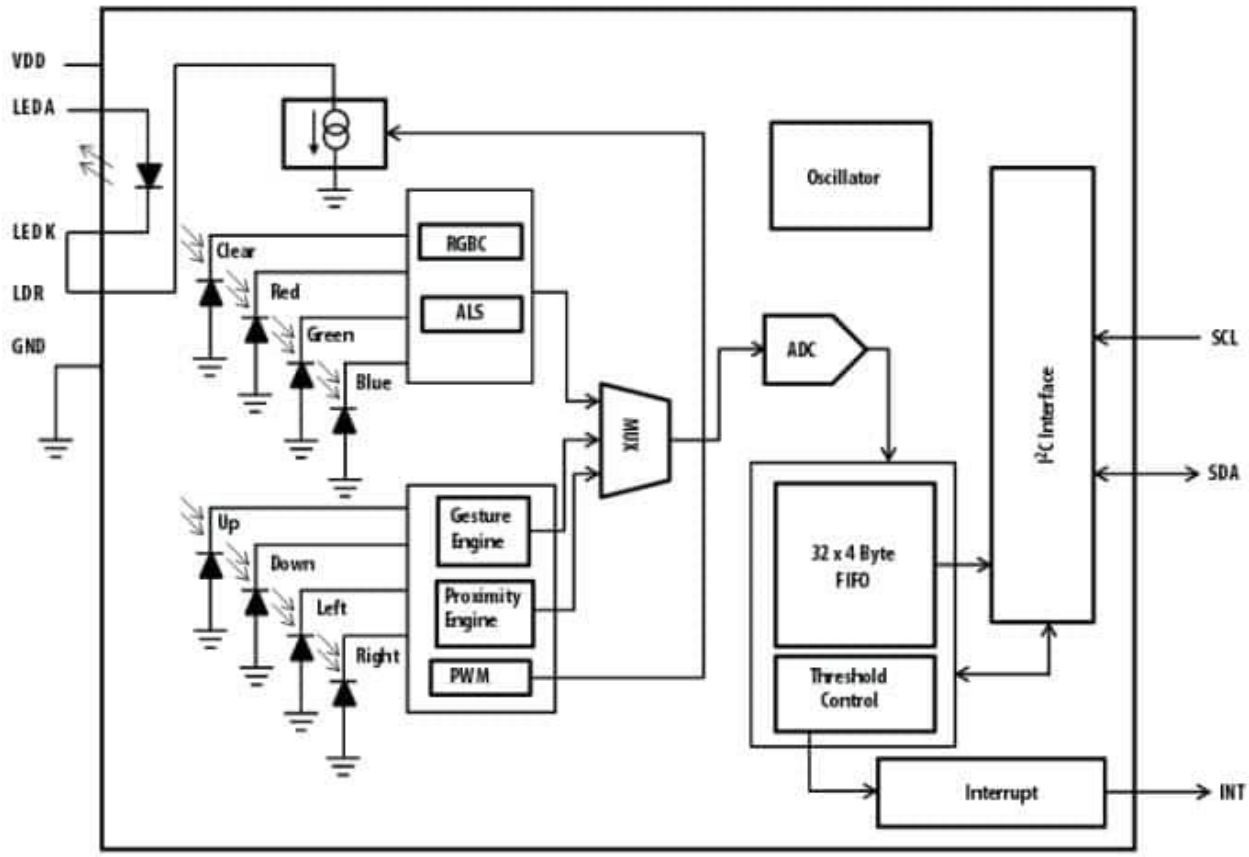
- 10 Wires
- Breadboard
- Atmega 328p
- FTDI Basic (testing purposes)
- ESP8266
- APDS9960

Block diagram with pins used in the Atmega328P

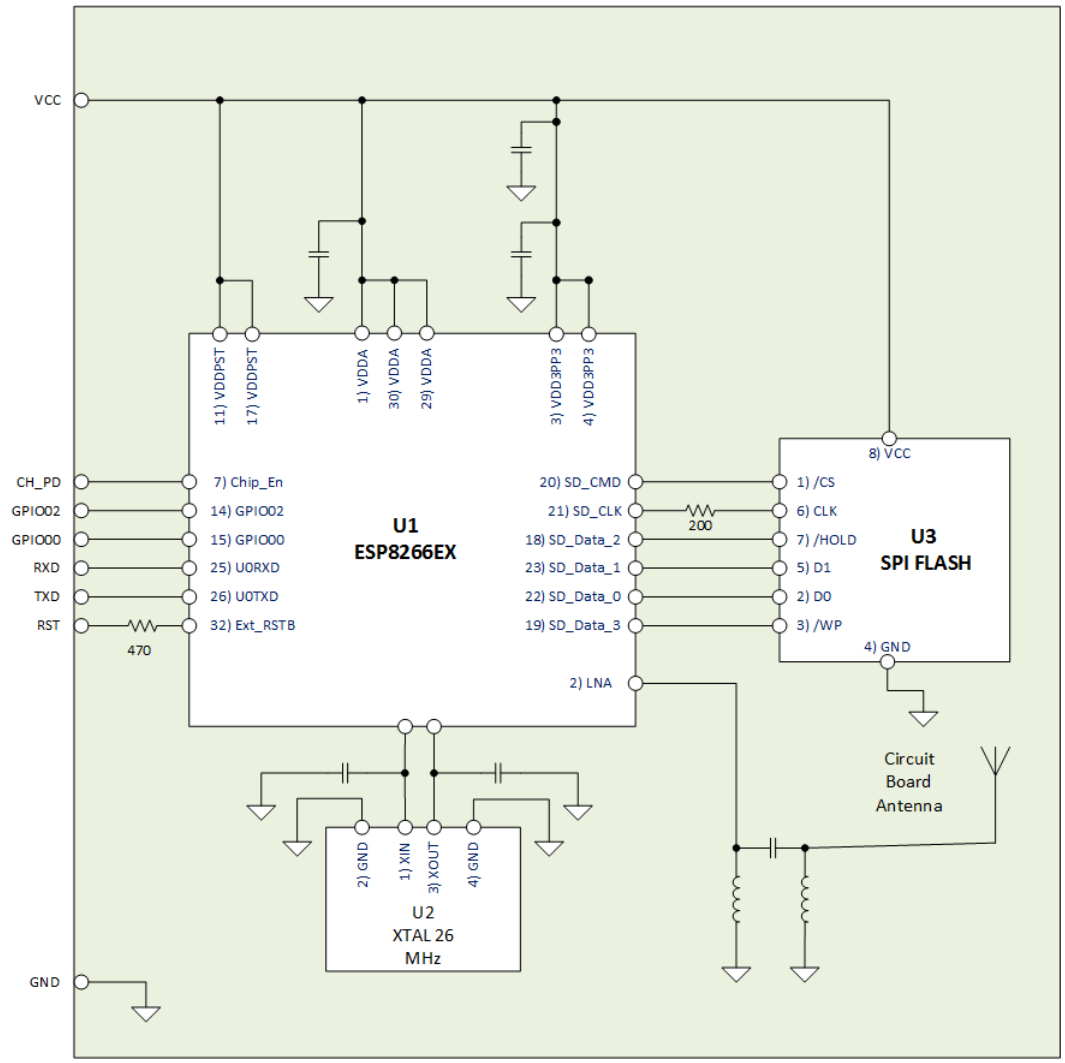


Used pins PC5 and PC4, used 3.3V for both the ESP and the APDS, and used PD0 and PD1 for TX and RX respectively.

Functional Block Diagram



Used SCL for the clock to be inputted to the APDS chip on PC5
Used SDA for the RGB data outputs on PC4



Used TX and RX to transmit and receive from the 328P to the ESP

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

```
/*
 * Midterm 2
 *
 * Created: 5/10/2019 3:23:52 PM
 * Author : Chris
 */
// NOTE: Did not tamper with the given 4 files, this is the main code!

#include <avr/io.h>
#include <stdio.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdlib.h>
#include <stdint.h>

#include "SparkFun_APDS9960.h"
```

```
#include "i2c_master.h"

#define F_CPU 16000000UL
#define BAUD 9600
#define FOSC 16000000
#define UBRREQ FOSC/16/BAUD -1

#define APDS9960_WRITE 0x72
#define APDS9960_READ 0x73

void UART_init (void);
void APDS_init (void);

int uart_putchar( char c, FILE *stream);
FILE str_uart = FDEV_SETUP_STREAM(uart_putchar, NULL , _FDEV_SETUP_WRITE);
void getreading(void);

uint16_t red;
uint16_t green;
uint16_t blue;

char sred[5];
char sgreen[5];
char sblue[5];

int main( void )
{
	UART_init(); // Initializes UART values
	APDS_init(); // Initializes APDS9960
	i2c_init(); // Initializes I2C
	stdout = &str_uart;

	red = 0;
	green = 0;
	blue = 0;

	// Checks AT commands (not needed)
	_delay_ms(2000);
	printf("AT\r\n");

	// Set AP's info which will be connect by ESP8266. (AP + Station Mode)
	_delay_ms(5000);
	printf("AT+CWMODE=3\r\n");

	// Connect to Internet
	_delay_ms(5000);
	printf("AT+CWJAP=\"[WIFI NAME]\",\"[WIFI P/W]\"\r\n");	// WIFI
SETTINGS LOCATED HERE

	while(1) // Constantly send values through the cloud until device turns
off
	{
		//
==========================================================================
==========================
		// Calls functions to connect to thingspeak, sets length of data
to be sent, sends the data values to cloud,
```

```
            // pauses till data goes through cloud
            //
================================================================================
============================

            // Enable Single Connection
            _delay_ms(5000);
            printf("AT+CIPMUX=0\r\n");

            // Start the connection to the cloud
            _delay_ms(5000);
            printf("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n");

            // Grab values from APDS chip
            // Set length of data to be sent (type of data)
            // Send values red, green, blue (in respective order) to field 1,
field 2, and field 3
            _delay_ms(5000);
            getreading();
            printf("AT+CIPSEND=104\r\n");
            printf("GET
https://api.thingspeak.com/update?api_key=[API_WRITE_KEY]&field1=0%05u&field2
=%05u&field3=%05u\r\n", red, green, blue); // KEY LOCATED HERE
            _delay_ms(3000);

            //
================================================================================
============================
      }
}

void getreading(){

      uint8_t redH, redL;
      uint8_t greenH, greenL;
      uint8_t blueH, blueL;

      // RED
      i2c_readReg(APDS9960_WRITE, APDS9960_RDATAH, &redH, 1);
      i2c_readReg(APDS9960_WRITE, APDS9960_RDATAL, &redL, 1);

      // GREEN
      i2c_readReg(APDS9960_WRITE, APDS9960_GDATAH, &greenH, 1);
      i2c_readReg(APDS9960_WRITE, APDS9960_GDATAL, &greenL, 1);

      // BLUE
      i2c_readReg(APDS9960_WRITE, APDS9960_BDATAH, &blueH, 1);
      i2c_readReg(APDS9960_WRITE, APDS9960_BDATAL, &blueL, 1);

      red = (redH << 8) | redL;
      green = (greenH << 8) | greenL;
      blue = (blueH << 8) | blueL;

      // THRESHOLD
      if (red > 255)
            red = 255;
      if (green > 255)
```

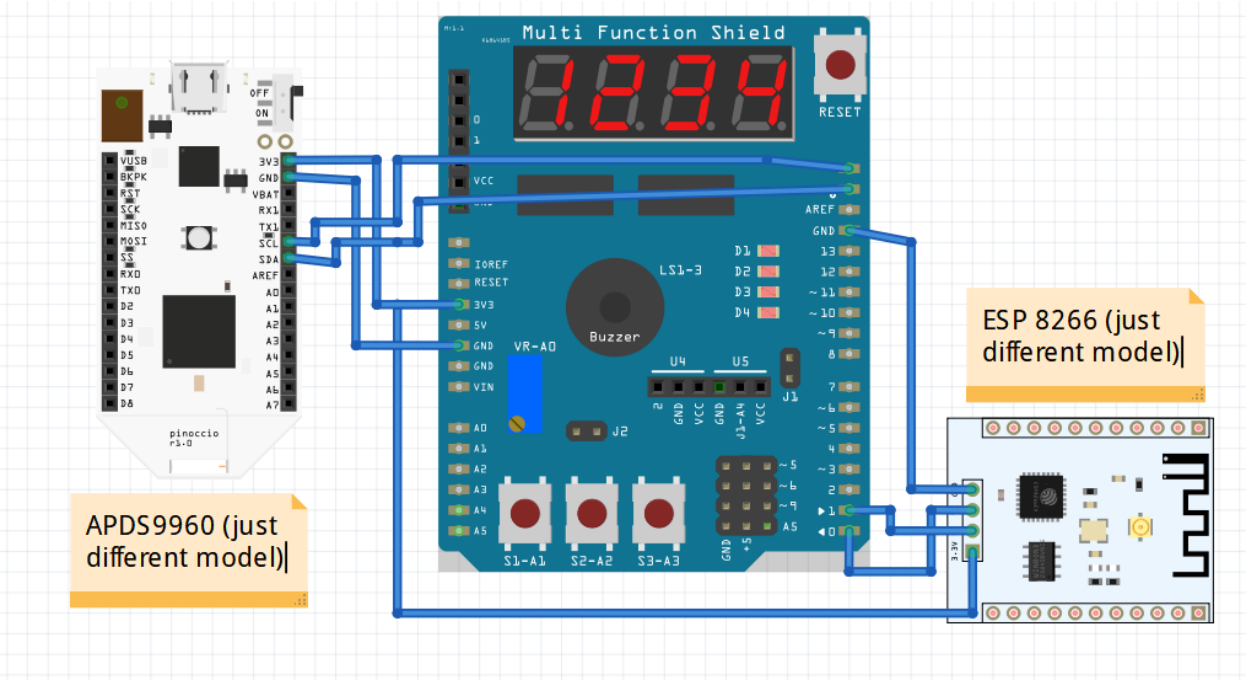
```
            green = 255;
      if (blue > 255)
            blue = 255;
}

void APDS_init(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 USART_putstring(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
      }
}

void UART_init(void)
{
      //Set baud rate
```

```
        uint16_t baud_rate = UBRREQ;
        UBRR0H = baud_rate >> 8;
        UBRR0L = baud_rate & 0xFF;

        //Enable receiver and transmitter
        UCSR0B = ( 1 <<RXEN0)|( 1 <<TXEN0);

        // Set frame format: 8data, 1stop bit
        UCSR0C = (3 <<UCSZ00);
}

int uart_putchar(char c, FILE *stream)
{
        //wait until buffer empty
        while ( !( UCSR0A & ( 1 <<UDRE0)) );

        //Put data into buffer
        UDR0 = c;
        return 0;
}
```

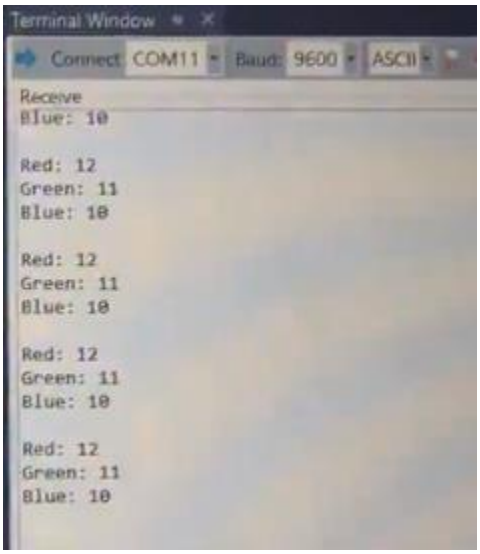
3. DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A

n/a

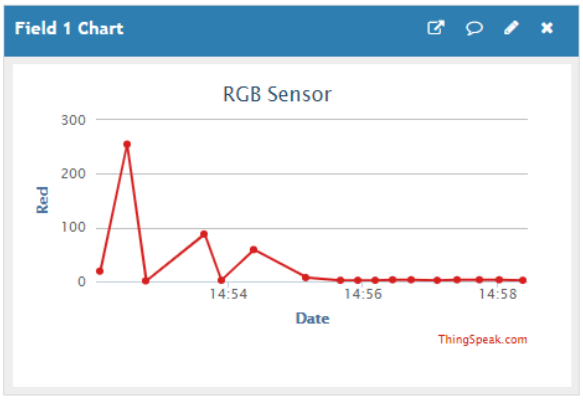
4. SCHEMATICS

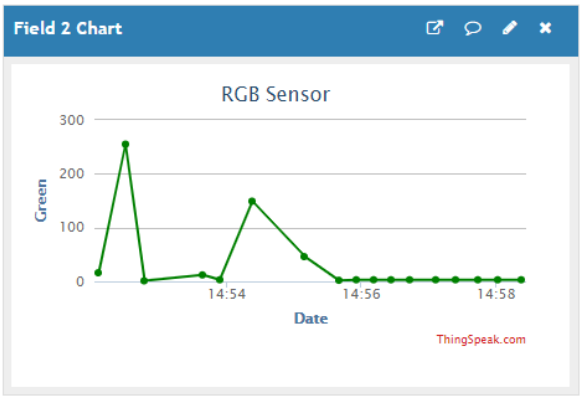


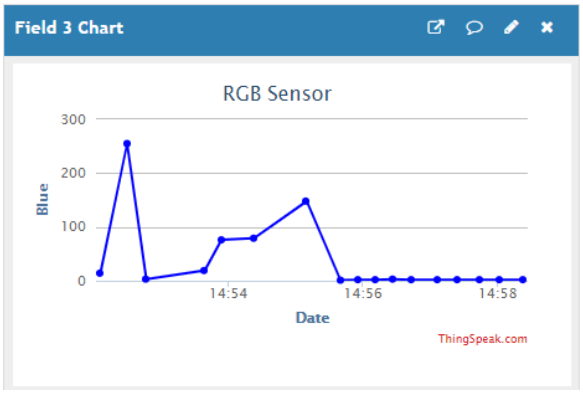
5. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)



Terminal Output for testing the sensor



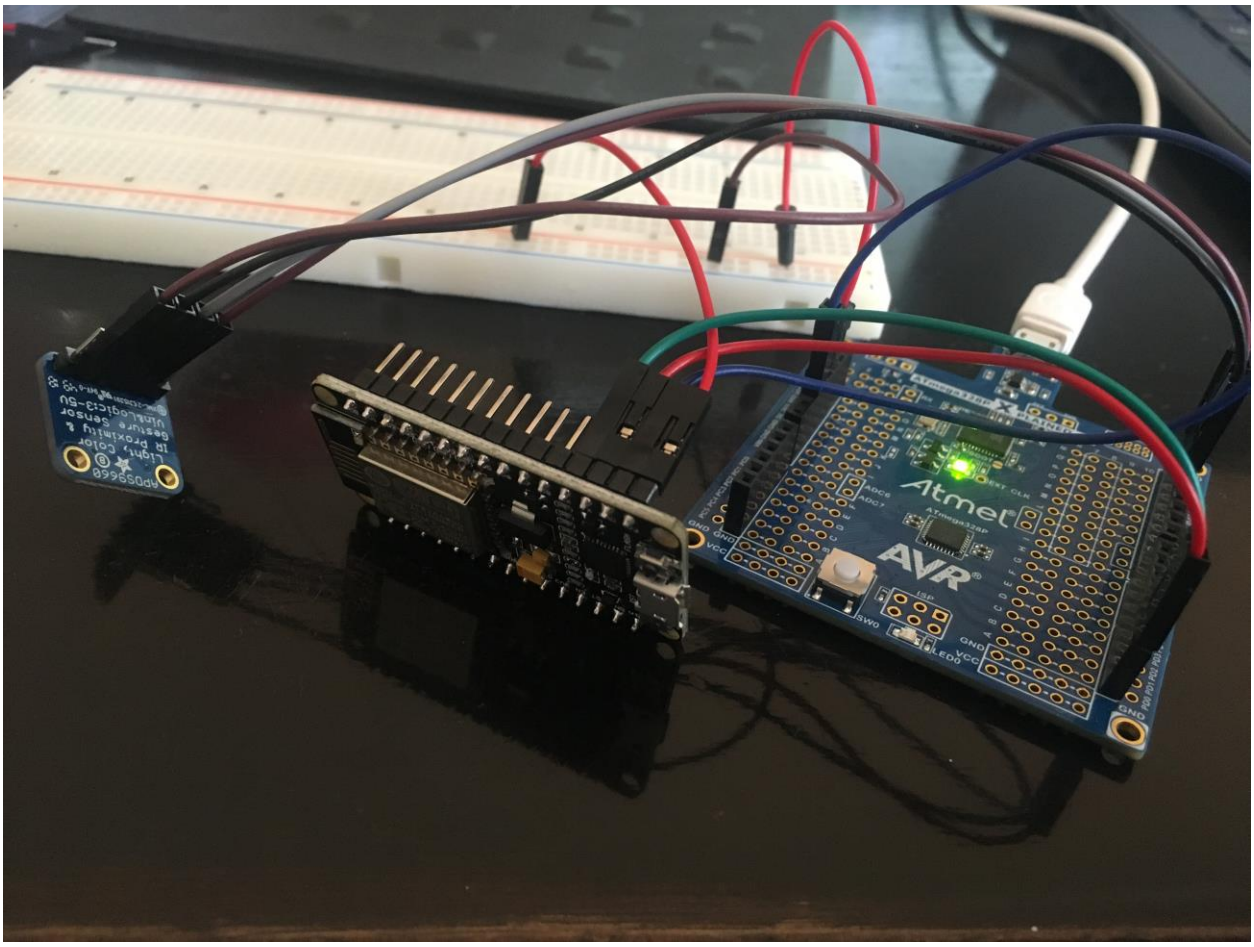




ThingSpeak output

Note: Tops off at 255 for each RGB which shows at the beginning of each graph. A white light was shined into it which gave 255 for all three. **255 is the <u>Threshold</u>** for the device.

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)



Board setup with APDS (blue) on the left, ESP in the center, and the Atmega328P on the right

7. VIDEO LINKS OF EACH DEMO

https://www.youtube.com/watch?v=Ky9lEbHhfRw

8. GITHUB LINK OF THIS DA

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"*This assignment submission is my own, original work*".
Chris Barr