CPE301 – SPRING 2019

MIDTERM 2

Student Name: James Skelly

Student #: 2000945485

Student Email: skellj1@unlv.nevada.edu

Primary Github address: <https://github.com/skellj1/submission_da>

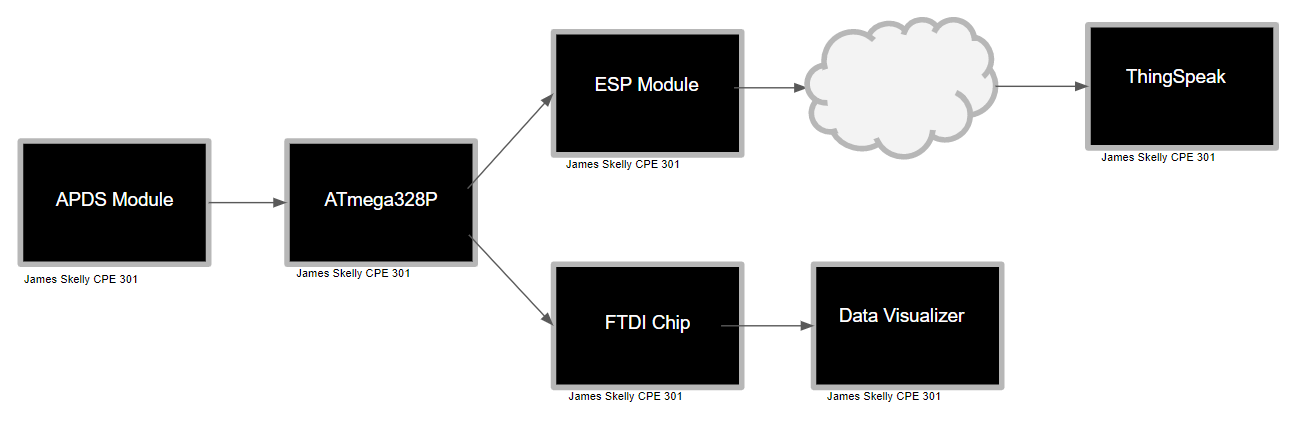
Directory: skellj1/submission\_da

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**

Components include : Xplained Mini, ESP Module, FTDI Chip, APDS9960, Breadboard.



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

/\*

\* Midterm 2

\*

\* Created: 5/8/2019 1:43:56 AM

\* Author : Skellj1

\*/

#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 ReadValues(void);

// Initialize variables for field readings of red, green, blue light

*uint16\_t* RED, GREEN, BLUE;

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

int main( void )

{

UART\_init(); // UART Initialization

APDS\_init(); // Sensor Module Initialization

i2c\_init(); // I2C Initialization

*stdout* = &str\_uart;

RED = 0; // Initialize RED read variable to 0

GREEN = 0; // Initialize GREEN read variable to 0

BLUE = 0; // Initialize BLUE read variable to 0

*\_delay\_ms*(2000);

*printf*("AT\r\n"); // AT initial communication confirmation

*\_delay\_ms*(5000);

*printf*("AT+CWMODE=3\r\n"); // AP + Station Mode

*\_delay\_ms*(5000);

*printf*("AT+CWJAP=\"[Network]\",\"[Password]\"\r\n");// WIFI information inserted here

while(1)

{

*\_delay\_ms*(5000);

*printf*("AT+CIPMUX=0\r\n"); // Initial connection

*\_delay\_ms*(5000);

*printf*("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n");// connect to cloud

*\_delay\_ms*(5000);

ReadValues(); // call function to read values from sensor

*printf*("AT+CIPSEND=104\r\n");

// channel key for thingspeak (API Key excluded)

*printf*("GET https://api.thingspeak.com/update?api\_key=[API\_KEY\_GOES\_HERE]&field1=0%05u&field2=%05u&field3=%05u\r\n", RED, GREEN, BLUE);

*\_delay\_ms*(3000);

}

}

void UART\_init(void)

{

// Set baud rate

*uint16\_t* BAUDRATE = UBRREQ;

UBRR0H = BAUDRATE >> 8;

UBRR0L = BAUDRATE & 0xFF;

// Enable receive/transmit

UCSR0B = ( 1 << RXEN0)|( 1 << TXEN0);

// 8 data bit, one stop bit

UCSR0C = (3 << UCSZ00);

}

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);

}

int uart\_putchar(char c, *FILE* \*stream)

{

// wait here

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

// insert data into buffer

UDR0 = c;

return 0;

}

void USART\_putstring(char \*StringPtr)

{

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

{

while (!(UCSR0A & (1 << UDRE0))); // loop until UDRE0 is high

UDR0 = \*StringPtr; // grab value from pointer address

StringPtr++; // Increment string pointer character by character

}

}

// Function to read values from sensor

void ReadValues()

{

*uint8\_t* redHigh, redLow;

*uint8\_t* greenHigh, greenLow;

*uint8\_t* blueHigh, blueLow;

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

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

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

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

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

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

// bit manipulation to format values read in from sensor

RED = (redHigh << 8) | redLow;

GREEN = (greenHigh << 8) | greenLow;

BLUE = (blueHigh << 8) | blueLow;

// If statements to cap value at 255 to keep an 8-bit value

if (RED > 255)

{

RED = 255;

}

if (GREEN > 255)

{

GREEN = 255;

}

if (BLUE > 255)

{

BLUE = 255;

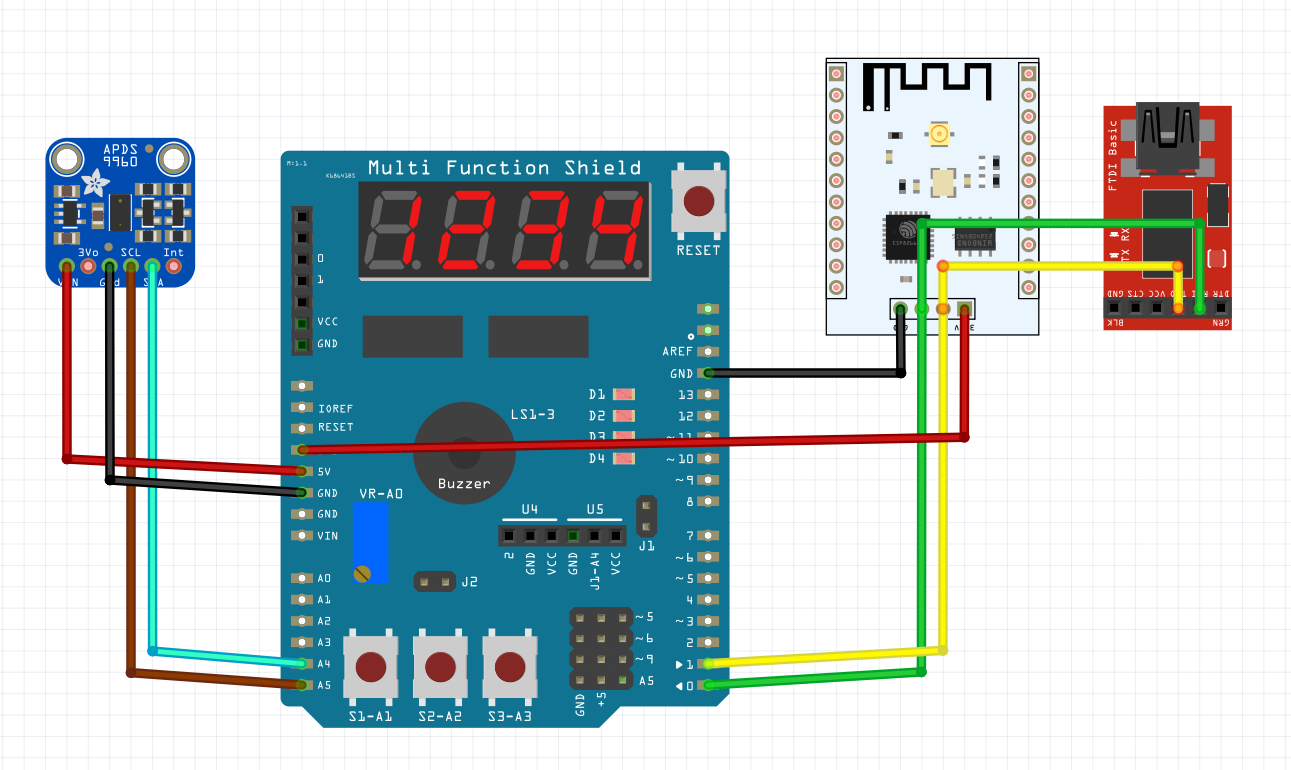
}

}

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

**Not applicable for this assignment.**

1. **SCHEMATICS**

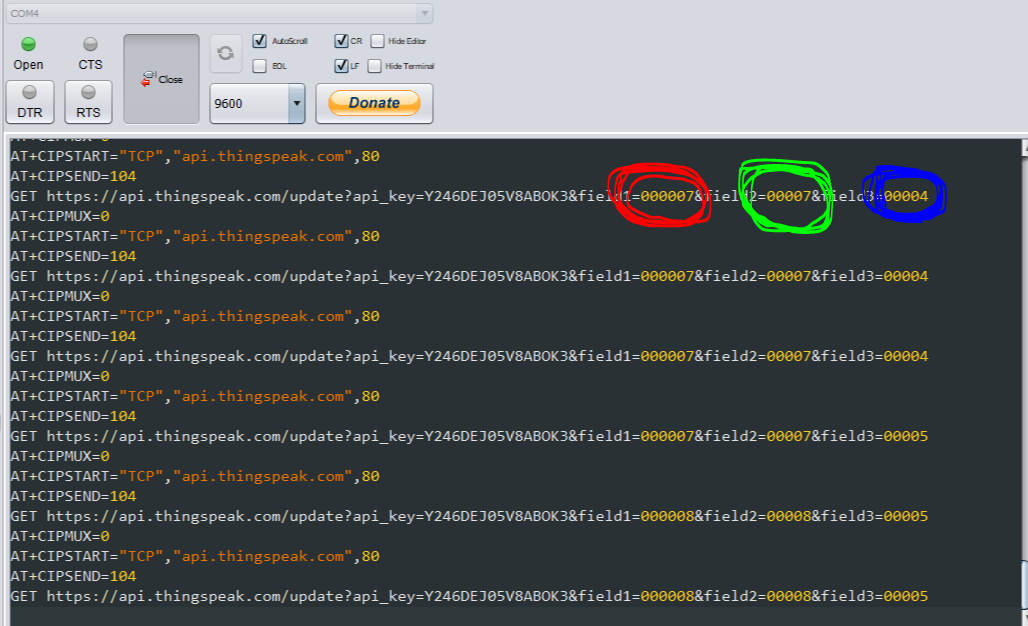


The schematic above was drafted using fritzing. The shield is shown here simply for access to the Xplained Mini pins, which correspond to the pins on the shield. The ESP module was wired up, powered with the on board 3.3V power supply, grounded to the board, and its transmit and receive pins were connected to the receive and transmit pins of the mini.

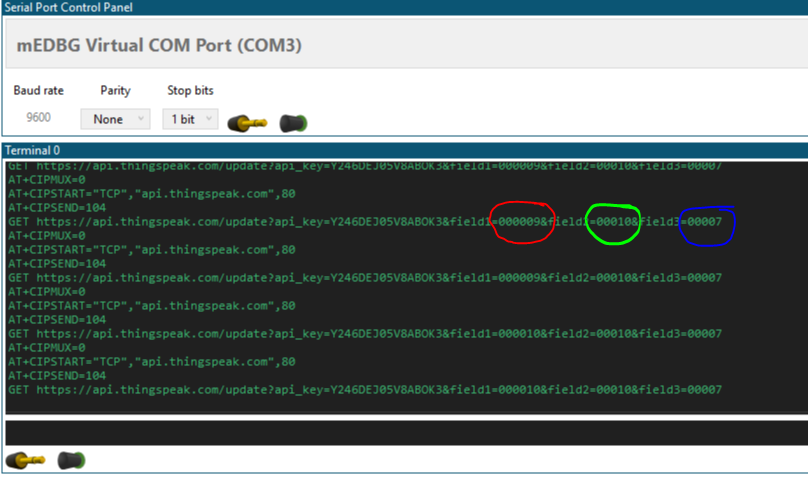
The FTDI module was also connected by Tx/Rx to both the WiFi module and the mini.

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

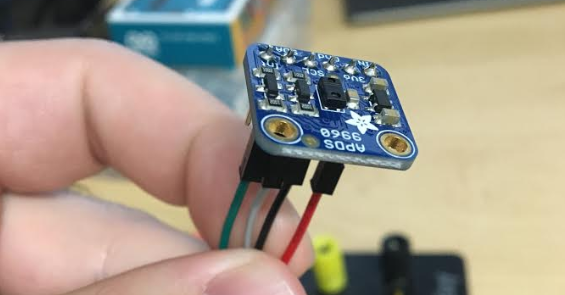
**Outputting Values from Sensor in ESPlorer**

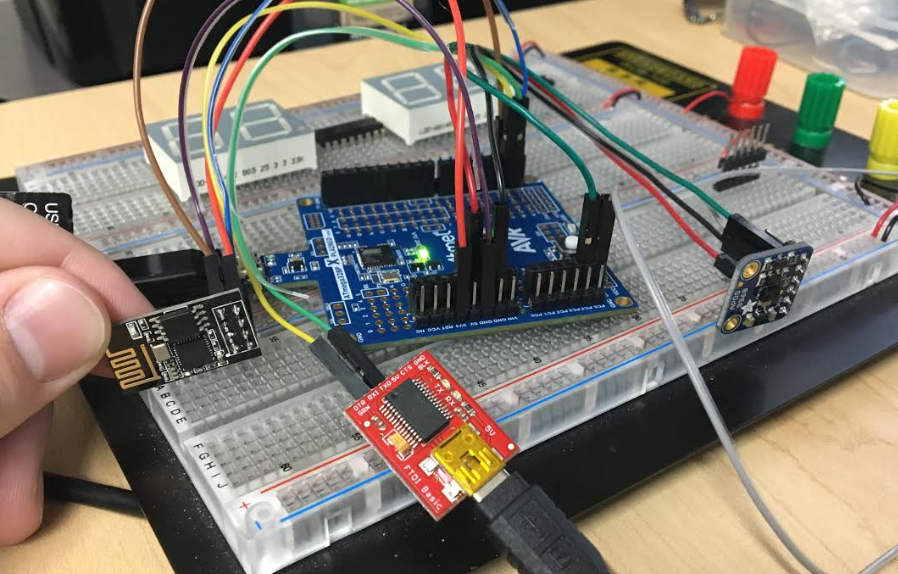


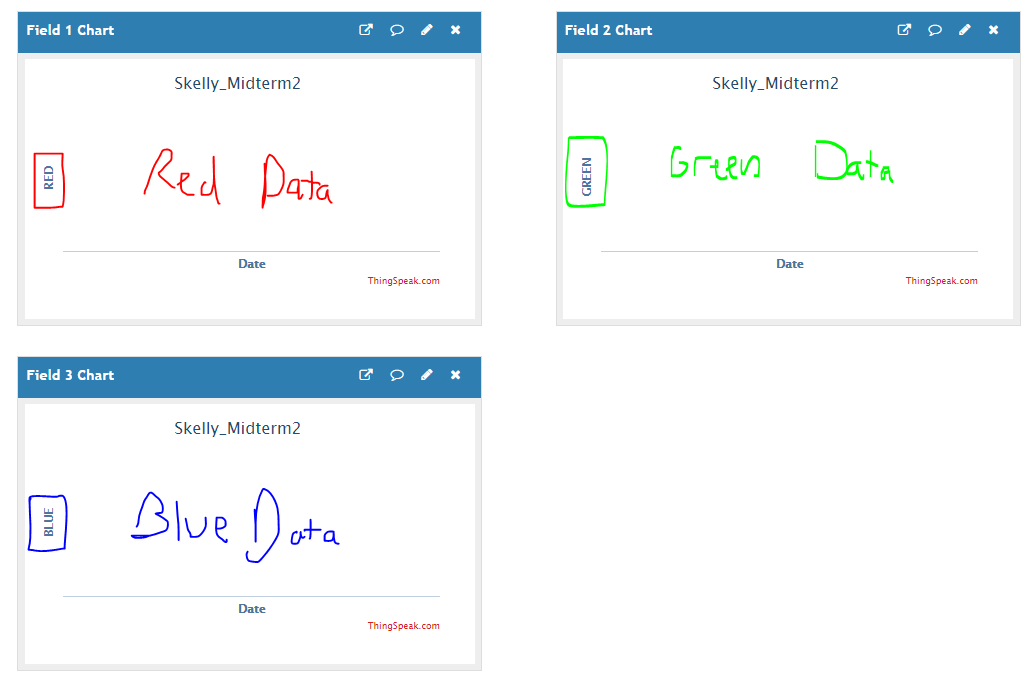
**Outputting Values from Sensor in Data Visualizer in Atmel Studio**



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







**Encountered Problems**

**Unfortunately, I was not able to get ThingSpeak properly working for this assignment. I used the same ESP Module that I used for midterm 1, and my ESP Module was working just fine for that midterm. However, for some reason, even when connected to the wifi, and even with the channel key entered, the light on my ESP module would still flash indicating that receiving and transmitting of data was occurring. Regardless the data never showed up in ThingSpeak.**

**Being finals week, time constraints were tough, and I was unable to get ThingSpeak working. If I had been able to get it working, my plan was, as is seen above, to have three separate fields all in the same channel that would output the RED, GREEN, and BLUE values as a function of time, with the Y-axis ranging from 0 to 255. If I had more time, I would have gotten a new ESP Module and tried to flash the firmware onto that newer module and reattempt to upload data to ThingSpeak.**

1. **VIDEO LINKS OF EACH DEMO**

<https://www.youtube.com/watch?v=yr3-6CnEsH4>

1. **GITHUB LINK OF THIS DA**

<https://github.com/skellj1/submission_da>

**Student Academic Misconduct Policy**

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

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

James W. Skelly