CPE301 – SPRING 2019

MIDTERM 2

Student Name: David Flores

Student #: 2000881655

Student Email: flored6@unlv.nevada.edu

Primary Github address: https://github.com/David-Floress/submission\_da.git

Directory: <https://github.com/David-Floress/submission_da/tree/master/Midterm2/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**

**Components Used**

Atmega328p

Breadboard

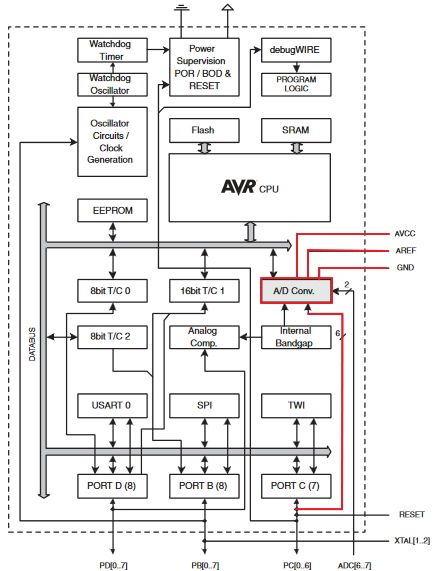
FTDI Basic

ESP8266

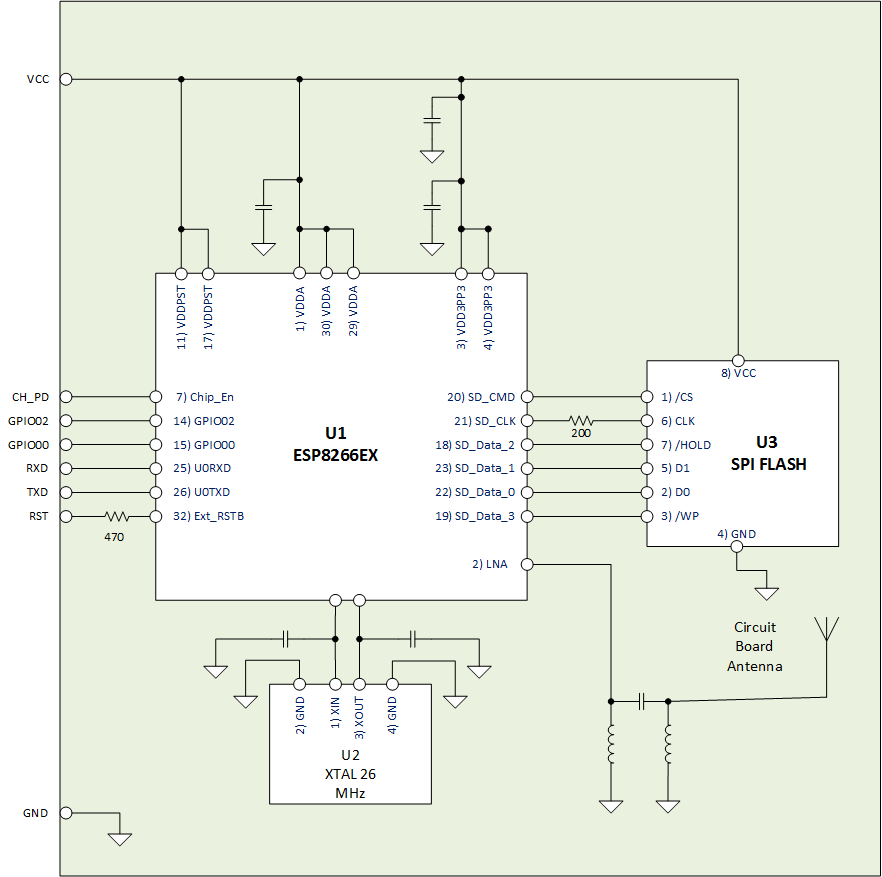
APDS9960

Wires

Block diagram with pins used in the Atmega328P



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



I used Tx and Rx to transmit and receive data

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

/\*

\* Midterm2.c

\*

\* Created: 5/11/2019 8:29:33 PM

\* Author : David Flores

\*/

#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 startreading(void);

*uint16\_t* red;

*uint16\_t* green;

*uint16\_t* blue;

int main( void )

{

UART\_init(); // This initializes UART values

APDS\_init(); // This initializes APDS9960

i2c\_init(); // This initializes I2C

*stdout* = &str\_uart;

red = 0;

green = 0;

blue = 0;

//

*\_delay\_ms*(2000);

*printf*("AT\r\n");

// Set AP’s info connected by ESP8266. (AP + Station Mode)

*\_delay\_ms*(5000);

*printf*("AT+CWMODE=3\r\n");

// Internet Connection

*\_delay\_ms*(5000);

*printf*("AT+CWJAP=\"\",\"\"\r\n"); // Wifi password and SSID

while(1) // This constantly sends values to the cloud

{

// Enables Single Connection

*\_delay\_ms*(5000);

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

// Start the connection to cloud

*\_delay\_ms*(5000);

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

// Send values red, green, blue to field 1, field 2, and field 3

*\_delay\_ms*(5000);

startreading();

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

*printf*("GET https://api.thingspeak.com/update?api\_key=B6EV1IA6ED34HM1D&field1=0%05u&field2=%05u&field3=%05u\r\n", red, green, blue);

*\_delay\_ms*(3000);

//

}

}

void startreading(){

*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;

// The Threshold limit

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')){ // Keep looping until the end of the line

while (!(UCSR0A & (1 << UDRE0))); // Keep looping until UDRE0 goes high

UDR0 = \*StringPtr;

StringPtr++;

}

}

void UART\_init(void)

{

//Sets the baud rate

*uint16\_t* baud\_rate = UBRREQ;

UBRR0H = baud\_rate >> 8;

UBRR0L = baud\_rate & 0xFF;

//This enable receiver and the transmitter

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

// Set frame format: 8data, 1stop bit

UCSR0C = (3 <<UCSZ00);

}

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

{

//waits until buffer is empty

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

//Put data into the buffer

UDR0 = c;

return 0;

}

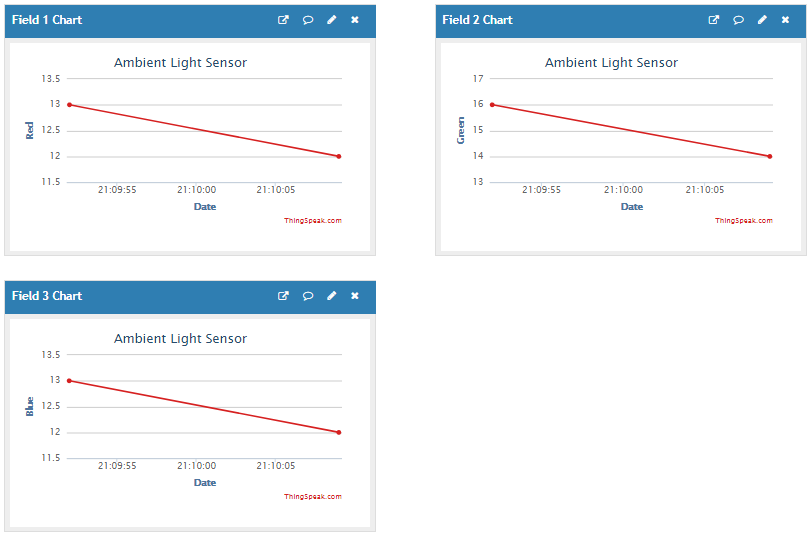
1. **DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A**

n/a

1. **SCHEMATICS**

N/A

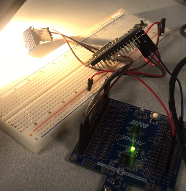
1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**



ThingSpeak Graphs

For the graphs I got a good output at first that showed the Threshold Voltage and both Rise and Fall. But then when I went to record it crashed. So, I had to do it over again and I couldn’t get the same output anymore. The Threshold was 255 which was the maximum point the graphs got to. But I ran out of time to get the correct simulations.

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



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

1. **VIDEO LINKS OF EACH DEMO**

<https://www.youtube.com/watch?v=fImrgXOBCxU>

1. **GITHUB LINK OF THIS DA**

<https://github.com/David-Floress/submission_da/tree/master/Midterm2/Midterm2>

**Student Academic Misconduct Policy**

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

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

David Flores