CPE301 – SPRING 2019

MIDTERM 2

Student Name: Benjamin Johnson

Student #: 5003284489

Student Email: johnsb18@unlv.nevada.edu

Primary Github address: <https://github.com/johnsb18/ClassRepository>

Directory: Midterms

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**

Atmega328P

ESP01 Module

APDS device

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

// main.c //

#define F\_CPU 16000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <stdio.h>

#include "i2c\_master.h"

#include "uart.h"

#include "apds.h"

// This is used to set up a filestream to use UART\_string

*FILE* UART\_string = *FDEV\_SETUP\_STREAM*(uart\_putchar, *NULL* , *\_FDEV\_SETUP\_WRITE*);

// This array of chars is where we will store our result

char result[256];

int main(void)

{

// red, green, and blue are the rgb components that we want.

*uint16\_t* red = 0, green = 0, blue = 0;

// Initialize I2C communication protocol

i2c\_init();

// Initialize UART communication protocol

init\_UART();

// variable used for UART string streaming

*stdout* = &UART\_string;

// Initialize APDS device

APDS\_init();

*\_delay\_ms*(2000);

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

*\_delay\_ms*(5000);

*printf*("AT+CWMODE=1\r\n"); // enables wifi

*\_delay\_ms*(5000);

*printf*("AT+CWJAP=\"AndroidAP\",\"easymode\"\r\n"); // join network

while (1)

{

*\_delay\_ms*(5000);

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

*\_delay\_ms*(5000);

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

*\_delay\_ms*(5000);

RGB\_reader(&red, &green, &blue);

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

*printf*("GET https://api.thingspeak.com/update?api\_key=WGRC31FBM6LOHT0D&field1=0%05u&field2=%05u&field3=%05u\r\n", red, green, blue);

*\_delay\_ms*(3000);

}

}

// i2c\_master.c //

#ifndef F\_CPU

#define F\_CPU 16000000UL

#endif

#include <avr/io.h>

#include <util/twi.h>

#include "i2c\_master.h"

#define F\_SCL 100000UL // SCL frequency

#define Prescaler 1

#define TWBR\_val ((((F\_CPU / F\_SCL) / Prescaler) - 16 ) / 2)

void i2c\_init(void)

{

TWBR = (*uint8\_t*)TWBR\_val;

}

*uint8\_t* i2c\_start(*uint8\_t* address)

{

// reset TWI control register

TWCR = 0;

// transmit START condition

TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);

// wait for end of transmission

while( !(TWCR & (1<<TWINT)) );

// check if the start condition was successfully transmitted

if((TWSR & 0xF8) != *TW\_START*){ return 1; }

// load slave address into data register

TWDR = address;

// start transmission of address

TWCR = (1<<TWINT) | (1<<TWEN);

// wait for end of transmission

while( !(TWCR & (1<<TWINT)) );

// check if the device has acknowledged the READ / WRITE mode

*uint8\_t* twst = *TW\_STATUS* & 0xF8;

if ( (twst != *TW\_MT\_SLA\_ACK*) && (twst != *TW\_MR\_SLA\_ACK*) ) return 1;

return 0;

}

*uint8\_t* i2c\_write(*uint8\_t* data)

{

// load data into data register

TWDR = data;

// start transmission of data

TWCR = (1<<TWINT) | (1<<TWEN);

// wait for end of transmission

while( !(TWCR & (1<<TWINT)) );

if( (TWSR & 0xF8) != *TW\_MT\_DATA\_ACK* ){ return 1; }

return 0;

}

*uint8\_t* i2c\_read\_ack(void)

{

// start TWI module and acknowledge data after reception

TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);

// wait for end of transmission

while( !(TWCR & (1<<TWINT)) );

// return received data from TWDR

return TWDR;

}

*uint8\_t* i2c\_read\_nack(void)

{

// start receiving without acknowledging reception

TWCR = (1<<TWINT) | (1<<TWEN);

// wait for end of transmission

while( !(TWCR & (1<<TWINT)) );

// return received data from TWDR

return TWDR;

}

*uint8\_t* i2c\_transmit(*uint8\_t* address, *uint8\_t*\* data, *uint16\_t* length)

{

if (i2c\_start(address | I2C\_WRITE)) return 1;

for (*uint16\_t* i = 0; i < length; i++)

{

if (i2c\_write(data[i])) return 1;

}

i2c\_stop();

return 0;

}

*uint8\_t* i2c\_receive(*uint8\_t* address, *uint8\_t*\* data, *uint16\_t* length)

{

if (i2c\_start(address | I2C\_READ)) return 1;

for (*uint16\_t* i = 0; i < (length-1); i++)

{

data[i] = i2c\_read\_ack();

}

data[(length-1)] = i2c\_read\_nack();

i2c\_stop();

return 0;

}

*uint8\_t* i2c\_writeReg(*uint8\_t* devaddr, *uint8\_t* regaddr, *uint8\_t*\* data, *uint16\_t* length)

{

if (i2c\_start(devaddr | 0x00)) return 1;

i2c\_write(regaddr);

for (*uint16\_t* i = 0; i < length; i++)

{

if (i2c\_write(data[i])) return 1;

}

i2c\_stop();

return 0;

}

*uint8\_t* i2c\_readReg(*uint8\_t* devaddr, *uint8\_t* regaddr, *uint8\_t*\* data, *uint16\_t* length)

{

if (i2c\_start(devaddr)) return 1;

i2c\_write(regaddr);

if (i2c\_start(devaddr | 0x01)) return 1;

for (*uint16\_t* i = 0; i < (length-1); i++)

{

data[i] = i2c\_read\_ack();

}

data[(length-1)] = i2c\_read\_nack();

i2c\_stop();

return 0;

}

void i2c\_stop(void)

{

// transmit STOP condition

TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);

}

// uart.c //

#include <stdio.h>

#include <avr/io.h>

#include <avr/interrupt.h>

#include "uart.h"

void init\_UART(void){

//Set baud rate

*uint16\_t* baud\_rate = BRGVAL;

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;

}

// apds.c //

#include <avr/io.h>

#include "i2c\_master.h"

#include "apds.h"

void APDS\_init(){

*uint8\_t* setup;

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

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

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

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

setup = DEFAULT\_ATIME;

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

setup = DEFAULT\_WTIME;

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

setup = DEFAULT\_PROX\_PPULSE;

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

setup = DEFAULT\_POFFSET\_UR;

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

setup = DEFAULT\_POFFSET\_DL;

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

setup = DEFAULT\_CONFIG1;

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

setup = DEFAULT\_PERS;

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

setup = DEFAULT\_CONFIG2;

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

setup = DEFAULT\_CONFIG3;

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

}

void RGB\_reader(*uint16\_t* \*red, *uint16\_t* \*green, *uint16\_t* \*blue){

*uint8\_t* redl, redh;

*uint8\_t* greenl, greenh;

*uint8\_t* bluel, blueh;

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

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

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

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

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

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

\*red = redh << 8 | redl;

\*green = greenh << 8 | greenl;

\*blue = blueh << 8 | bluel;

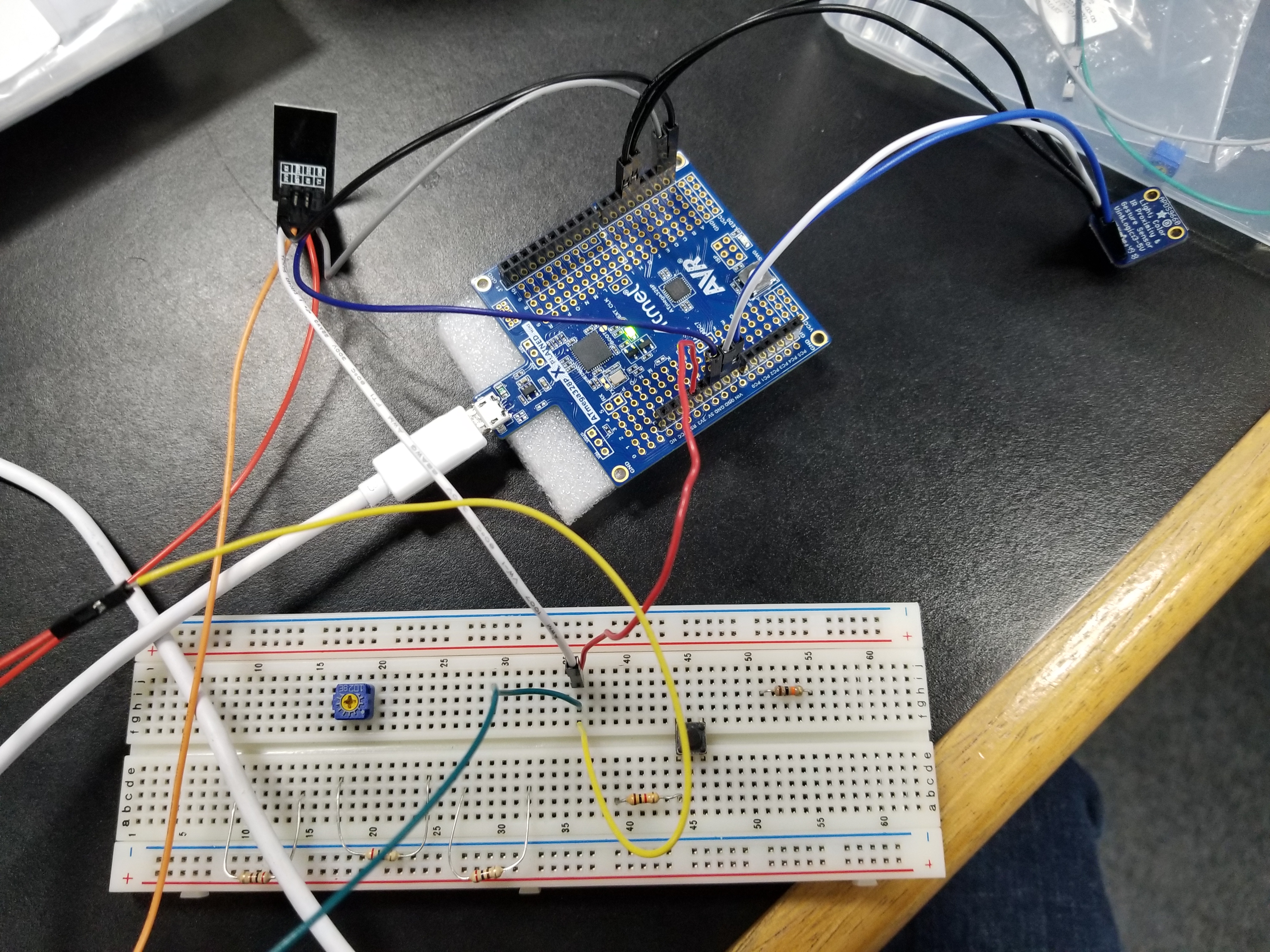
}

1. **DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A**
2. **SCHEMATICS**
3. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

A screenshot of a social media post

Description automatically generated

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



1. **VIDEO LINKS OF EACH DEMO**
2. **GITHUB LINK OF THIS DA**

<https://github.com/johnsb18/ClassRepository/tree/master/Midterms/Midterm%202>

**Student Academic Misconduct Policy**

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

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

Benjamin Johnson