CPE301 – SPRING 2019

MIDTERM 2

Student Name: Armon Latifi
Student #: 2000698173
Student Email: latifa1@unlv.nevada.edu
Primary Github address: https://github.com/armonlatifi
Directory: https://github.com/armonlatifi/sub_da/tree/master/MIDTERM%202

Submit the following for all Labs:

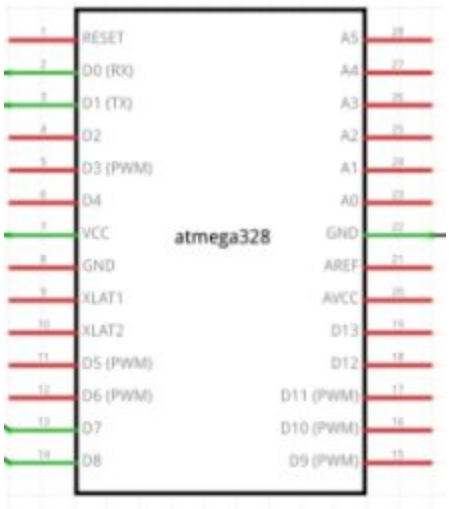
1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder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

- Assembler
- Simulator
- Debugger
- Breadboard
- Atmega328P
- Wires
- Esp32
- APDS
- Microusb
- Atmel Studio
- Thinkspeak

Block diagram with pins used in the Atmega328P



Green represents pins used.

2. INITIAL/MODIFIED/DEVELOPED CODE

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"

char results[256];
FILE str_uart = FDEV_SETUP_STREAM(uart_putchar, NULL , _FDEV_SETUP_WRITE);
```

```
int main(void)
{
	//need to set up our variables
	uint16_t red = 0, green = 0, blue = 0;
	//call i2c init
	i2c_init();
	//call uart init
	init_UART();
	stdout = &str_uart;
	//initialize apds function
	apds_init();
	//hit with 2000ms delay
	_delay_ms(2000);
	printf("AT\r\n");
	//hit with 5000ms delay
	_delay_ms(5000);
	printf("AT+CWMODE=1\r\n");
	//hit with 5000ms delay
	_delay_ms(5000);
	printf("AT+CWJAP=\"XXXXX\",\"XXXXX\"\r\n");

   while (1)
   {
			//hit with 5000ms delay
			_delay_ms(5000);
			printf("AT+CIPMUX=0\r\n");
			//hit with 5000ms delay
			_delay_ms(5000);
			printf("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n");
			//hit with 5000ms delay
			_delay_ms(5000);
			readColor(&red, &green, &blue);
			printf("AT+CIPSEND=104\r\n");
			printf("GET
https://api.thingspeak.com/update?api_key=1V8WUUJNEHZGA9L7&field1=%05u&field2=%05u&field3=%05u\r
\n", red, green, blue); //send to thinkspeak
			//hit with 5000ms delay
			_delay_ms(3000);
   }
}
```

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 readColor(uint16_t *red, uint16_t *green, uint16_t *blue){
	uint8_t redl, redh;
	//time to declare variables
	uint8_t greenl, greenh;
	uint8_t bluel, blueh;
	//also time to read the i2c variables
	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;
}
```

APDS.h

```
#ifndef APDS_H
#define APDS_H

#include <avr/io.h>
#include "i2c_MASTER.h"
#include "APDS.h"

#define APDS_WRITE     (0x39 << 1) | 0
#define APDS_READ      (0x39 << 1) | 1
// APDS-9960 I2C address
#define APDS9960_I2C_ADDR       0x39
// Acceptable device IDs
#define APDS9960_ID_1           0xAB
#define APDS9960_ID_2           0x9C
//Misc parameters
//wait period
#define FIFO_PAUSE_TIME         30
//APDS
#define APDS9960_ENABLE         0x80
#define APDS9960_ATIME          0x81
#define APDS9960_WTIME          0x83
```

```
#define APDS9960_CONFIG1        0x8D
#define APDS9960_CONFIG2        0x90
#define APDS9960_ID             0x92
#define APDS9960_RDATAL         0x96
#define APDS9960_RDATAH         0x97
#define APDS9960_GDATAL         0x98
#define APDS9960_GDATAH         0x99
#define APDS9960_BDATAL         0x9A
#define APDS9960_BDATAH         0x9B
#define APDS9960_CONFIG3        0x9F
//defaults
#define DEFAULT_ATIME           219     // 103ms
#define DEFAULT_WTIME           246     // 27ms
#define DEFAULT_GESTURE_PPULSE  0x89    // 16us, 10 pulses
#define DEFAULT_POFFSET_UR      0       // 0 offset
#define DEFAULT_POFFSET_DL      0       // 0 offset
#define DEFAULT_PERS            0x11    // 2 consecutive prox or ALS for int.
#define DEFAULT_CONFIG2         0x01    // No saturation interrupts or LED boost
#define DEFAULT_CONFIG3         0       // Enable all photodiodes, no SAI

void apds_init();
void readColor();
#endif
```

i2c_MASTER.c

```
#ifndef  F_CPU
#define F_CPU 16000000UL
#endif



#include <avr/io.h>
#include <util/twi.h>
#include "i2c_MASTER.h"



//determine SCL frequency
#define F_SCL 100000UL
#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)
{
        TWCR = 0;
        // transmit start condition
        TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
        //wait
        //did transmission end, check
```

```
	while( !(TWCR & (1<<TWINT)) );
	//successful transmission check
	if((TWSR & 0xF8) != TW_START){ return 1; }
	TWDR = address;
	//transmit...
	TWCR = (1<<TWINT) | (1<<TWEN);
	//wait
	//did transmission end, check
	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;
	TWCR = (1<<TWINT) | (1<<TWEN);
	//wait
	//did transmission end, check
	while( !(TWCR & (1<<TWINT)) );
	if( (TWSR & 0xF8) != TW_MT_DATA_ACK ){ return 1; }
	return 0;
}

uint8_t i2c_read_ack(void)
{
	//TWI time
	TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);
	//wait
	//did transmission end, check
	while( !(TWCR & (1<<TWINT)) );
	return TWDR;
}

uint8_t i2c_read_nack(void)
{

	//start reception
	TWCR = (1<<TWINT) | (1<<TWEN);
	//wait
	//did transmission end, check
	while( !(TWCR & (1<<TWINT)) );
	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)
{
//	stop
	TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);
}
```

i2c_MASTER.h

```
#ifndef I2C_MASTER_H
#define I2C_MASTER_H


#define I2C_WRITE 0x00
#define I2C_READ 0x01
```

```
void i2c_init(void);
uint8_t i2c_start(uint8_t address);
uint8_t i2c_write(uint8_t data);
uint8_t i2c_read_ack(void);
uint8_t i2c_read_nack(void);
uint8_t i2c_transmit(uint8_t address, uint8_t* data, uint16_t length);
uint8_t i2c_receive(uint8_t address, uint8_t* data, uint16_t length);
uint8_t i2c_writeReg(uint8_t devaddr, uint8_t regaddr, uint8_t* data, uint16_t length);
uint8_t i2c_readReg(uint8_t devaddr, uint8_t regaddr, uint8_t* data, uint16_t length);
void i2c_stop(void);
#endif
```

UART.c

```
#include <stdio.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include "UART.h"

void init_UART(void){
        uint16_t baud_rate = BRGVAL; //baud rate needs to be set
        UBRR0H = baud_rate >> 8;
        UBRR0L = baud_rate & 0xFF;


        //enable transmitter
        //enable receiver
        UCSR0B = ( 1 <<RXEN0)|( 1 <<TXEN0);
        UCSR0C = (3 <<UCSZ00);
}

int uart_putchar(char c, FILE *stream){
        while ( !( UCSR0A & ( 1 <<UDRE0)) ); //is buffer empty?
        UDR0 = c;
        return 0;
}
```

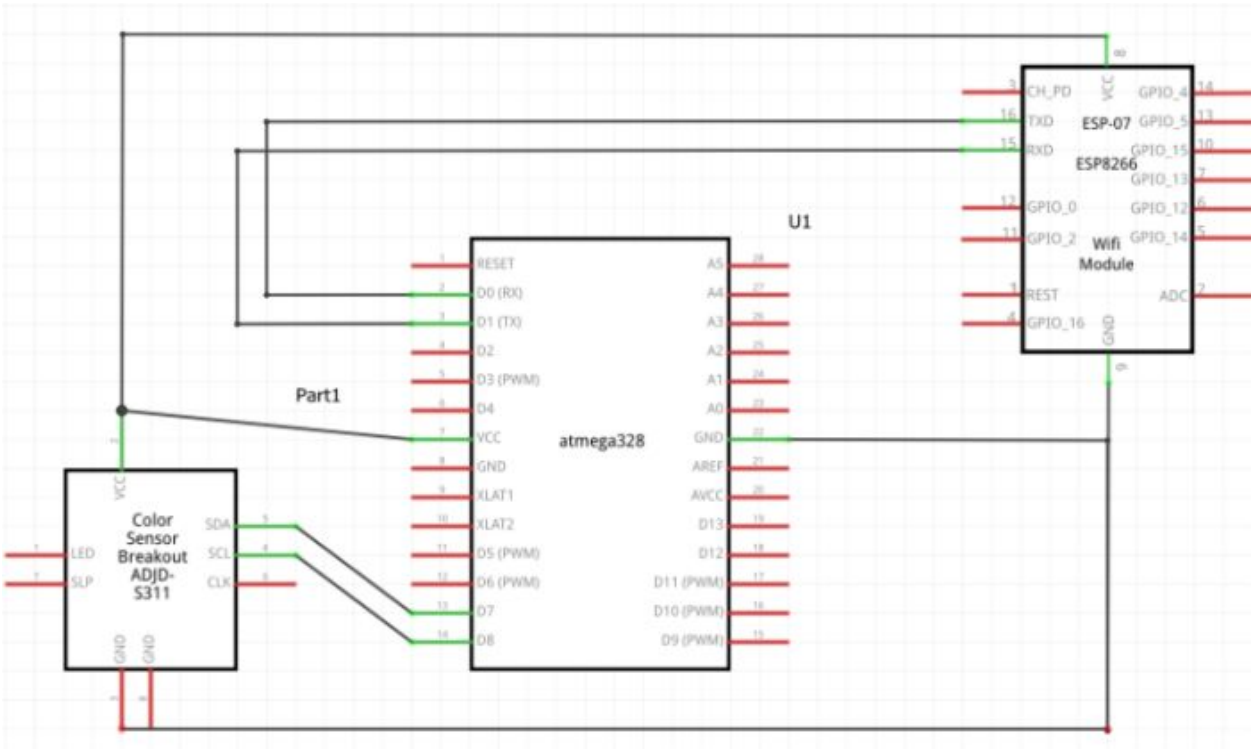
UART.h

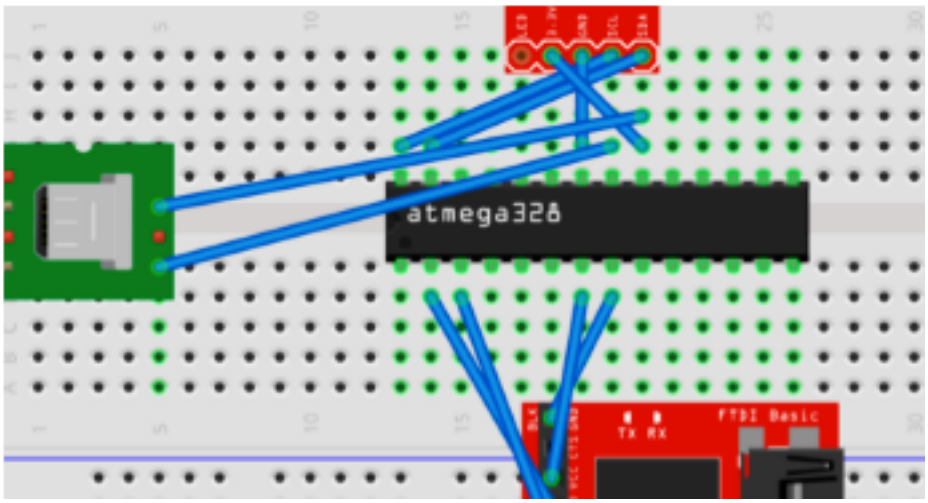
```
#ifndef UART_328P_H
#define UART_328P_H

#ifndef F_CPU
#define F_CPU 16000000UL
#endif



#include <stdio.h>
#include <avr/interrupt.h>
#include <avr/io.h>



#define BRGVAL (F_CPU/16/BAUD) - 1
#define BAUD 9600
```

```
void init_UART();
int uart_putchar( char c, FILE *stream);
#endif
```

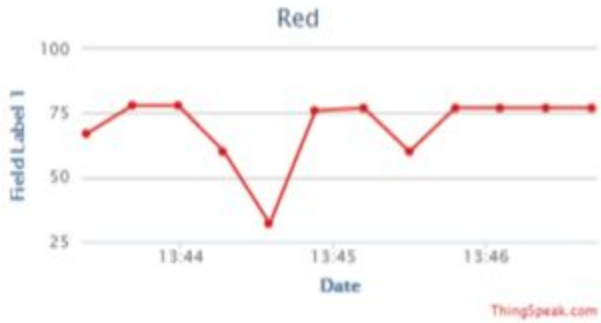
3. SCHEMATICS

Use fritzing.org

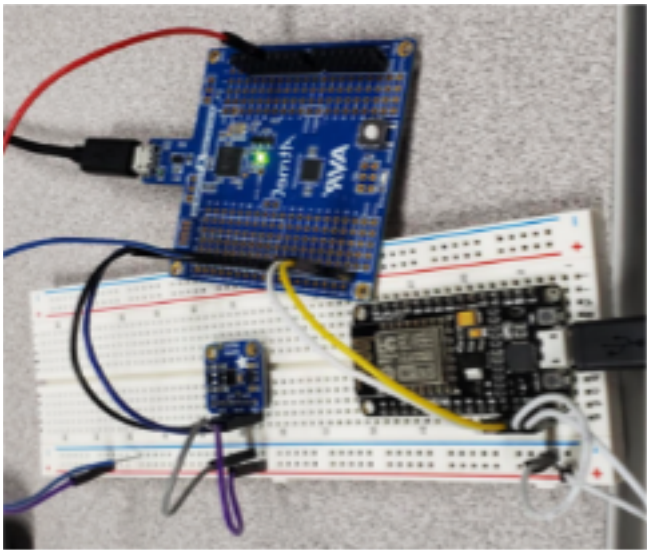




4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)



5. SCREENSHOT OF EACH DEMO (BOARD SETUP)



6. VIDEO LINKS OF EACH DEMO

7. GITHUB LINK OF THIS DA

https://github.com/armonlatifi/sub_da/tree/master/MIDTERM%202

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

"*This assignment submission is my own, original work*".
Armon Latifi