CPE301 – SPRING 2019

MIDTERM 2

Student Name: Shaquille Regis
Student #: 2000686590
Student Email: regis@unlv.nevada.edu
Primary Github address: https://github.com/regis-shaquille/submissions-SR
Directory: https://github.com/regis-shaquille/submissions-SR/tree/master/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

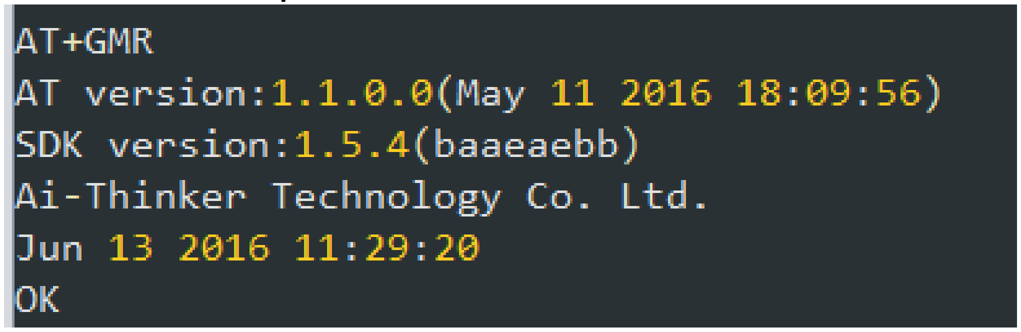
Atmega328P Xplained Mini Microcontroller
ESP 8266 Wifi Module
APDS9960 Ambient Light Sensor

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

```
/*
 * Midterm2.c
 *
 * Created: 5/12/2019 4:17:39 PM
 * Author : regis
 */

#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include "i2c_master.h"
#include "APDS9960_def.h"
#define BAUD 9600
#define MYUBRR F_CPU/16/BAUD-1

void APDS9960_Init()/* Gyro initialization function */
{
    _delay_ms(150);/* Power up time >100ms */
    i2c_start(0x39);/* Start with device write address */
    i2c_write(APDS9960_ENABLE);/* Write to power on register */
    i2c_write(0x03);/* Device On, ALS Enable */
    i2c_start(0x39);/* Start with device write address */
    i2c_write(APDS9960_ATIME);/* Write to sample rate register */
    i2c_write(0xFF);/* 1KHz sample rate */
    i2c_stop();
    i2c_start(0x39);
    i2c_write(APDS9960_CONTROL);/* Write to gain control register */
    i2c_write(0x01);/* 4x gain */
    i2c_stop();
}

void APDS9960_writereg(uint8_t reg, uint8_t val)
{
    i2c_start(APDS9960_I2C_ADDR+i2c_write());
    i2c_write(reg); // go to register e.g. 106 user control
    i2c_write(val); // set value e.g. to 0100 0000 FIFO enable
    i2c_stop(); // set stop condition = release bus
}
uint16_t APDS9960_readreg(uint8_t reg)
{
    int raw;
    i2c_start(APDS9960_I2C_ADDR+i2c_write()); // set device address and write mode
    i2c_write(reg); // ACCEL_XOUT
    i2c_start(APDS9960_I2C_ADDR+i2c_readReg); // set device address and read mode
    raw = i2c_read_ack(); // read one intermediate byte
    raw = (raw<<8) | i2c_read_nack(); // read last byte
    i2c_stop();
    return raw;
}

void ADC_init(void) //initialize ADC
{

    ADMUX = (0<<REFS1)| // Reference Selection Bits

    (1<<REFS0)| // AVcc - external cap at AREF
    (0<<ADLAR)| // ADC Left Adjust Result
    (0<<MUX2)|  // ANalog Channel Selection Bits
    (1<<MUX1)|  // ADC2 (PC2 PIN25)
    (0<<MUX0);

    ADCSRA = (1<<ADEN)| // ADC ENable

    (0<<ADSC)|  // ADC Start Conversion
    (0<<ADATE)| // ADC Auto Trigger Enable
    (0<<ADIF)|  // ADC Interrupt Flag
    (0<<ADIE)|  // ADC Interrupt Enable
    (1<<ADPS2)| // ADC Prescaler Select Bits
    (0<<ADPS1)|
    (1<<ADPS0);

    // Timer/Counter1 Interrupt Mask Register

    TIMSK1 |= (1<<TOIE1);           // enable overflow interrupt
    TCCR1B |= (1<<CS12)|(1<<CS10);  // native clock
    TCNT1 = 49911;                  //((16MHz/1024)*1)-1 = 15624

}


void readADC(void) {
    unsigned char i =4;
    adc_temp = 0; //initialize
    while (i--) {
        ADCSRA |= (1<<ADSC);
        while(ADCSRA & (1<<ADSC));
        adc_temp+= ADC;
        _delay_ms(50);
    }
    adc_temp = adc_temp / 4; // Average a few samples

}

// INIT USART (RS-232)
void USART_init( unsigned int ubrr ) {
    UBRR0H = (unsigned char)(ubrr>>8);
    UBRR0L = (unsigned char)ubrr;
    UCSR0B |= (1 << TXEN0) | (1 << RXEN0)| ( 1 << RXCIE0); // Enable receiver,
transmitter & RX interrupt
    UCSR0C |= (1<<UCSZ01) | (1 << UCSZ00);
}

void USART_tx_string( char *data ) {
    while ((*data != '\0')) {
        while (!(UCSR0A & (1 <<UDRE0)));
        UDR0 = *data;
        data++;
    }
}
```

```
ISR(TIMER1_OVF_vect) //timer overflow interrupt to delay for 1 second
{
    char TEMP[256];
    unsigned char AT[] = "AT\r\n"; //AT Commands
    unsigned char CWMODE[] = "AT+CWMODE=1\r\n"; //Set operation MODE
    unsigned char CWJAP[] = "AT+CWJAP=\"SSID\",\"PASSWORD\"\r\n"; // Do not turn in
with personal wifi/password
    unsigned char CIPMUX[] = "AT+CIPMUX=0\r\n";
    unsigned char CIPSTART[] = "AT+CIPSTART=\"TCP\",\"184.106.153.149\",80\r\n";
    unsigned char CIPSEND[] = "AT+CIPSEND=100\r\n";

    _delay_ms(200);
    USART_tx_string(AT); //send commands
    _delay_ms(5000);
    USART_tx_string(CWMODE); //set operation mode

    _delay_ms(5000);
    USART_tx_string(CWJAP); //connect to WIFI

    _delay_ms(15000);
    USART_tx_string(CIPMUX); //select MUX

    _delay_ms(10000);
    USART_tx_string(CIPSTART);//connect TCP

    _delay_ms(10000);
    USART_tx_string(CIPSEND);//send size

    _delay_ms(5000);

    PORTC^=(1<<5);
    readADC(); //read ADC
    snprintf(out,sizeof(out),"GET
https://api.thingspeak.com/update?api_key=9HD0YXSMDWBFG6Q7&field2=%3d\r\n",
adc_temp);// print
    USART_tx_string(out);//send data
    _delay_ms(10000);
    TCNT1 = 49911; //reset

}

int main(void)
{
    /* Replace with your application code */
    char buffer[20], float_[10];
    float Xa;
    i2c_init();/* Initialize I2C */
    APDS9960_Init();/* Initialize MPU6050 */
    USART_init(9600);/* Initialize USART with 9600 baud rate */
    while (1)
    {
        //Read_RawValue();
        /* Divide raw value by sensitivity scale factor to get real values */
        Xa = Acc_x/16384.0;
        /* Take values in buffer to send all parameters over USART */
        dtostrf( Xa, 3, 2, float_ );
        sprintf(buffer," Ax = %s g\t",float_);
        USART_tx_string(buffer);

    }
}
```

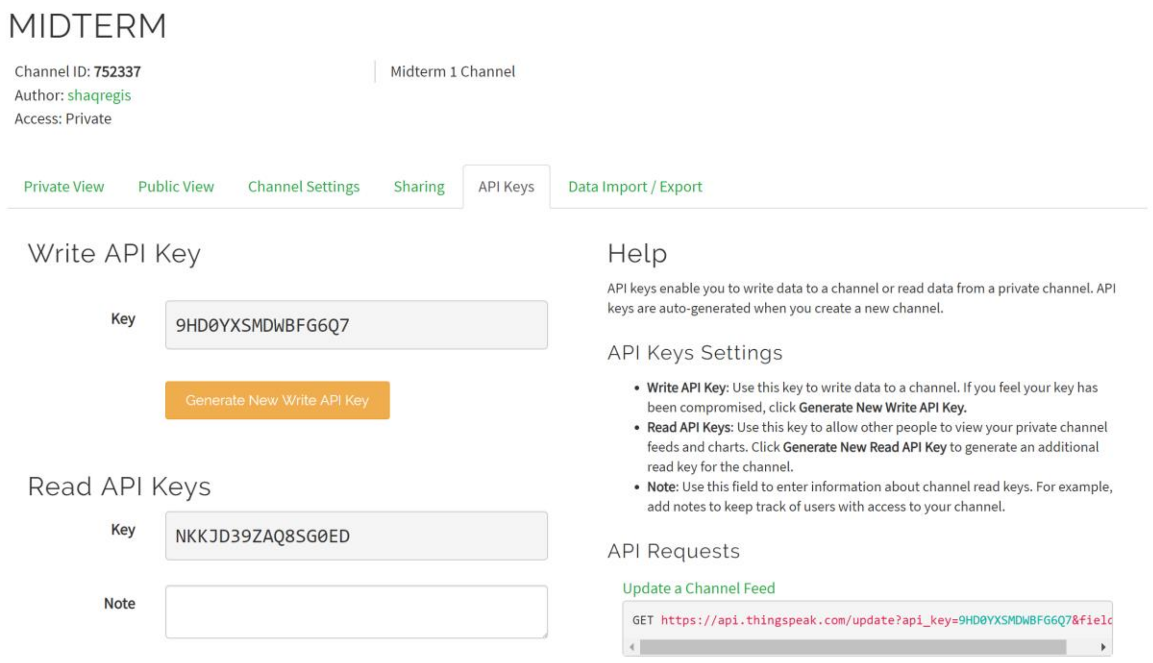
3. ESP8266 Setup



```
AT+GMR
AT version:1.1.0.0(May 11 2016 18:09:56)
SDK version:1.5.4(baaeaebb)
Ai-Thinker Technology Co. Ltd.
Jun 13 2016 11:29:20
OK
```

4. ThingSpeak Account

Insert only the modified sections here



MIDTERM

Channel ID: **752337**
Author: shaqregis
Access: Private

Midterm 1 Channel

Write API Key

Key: 9HD0YXSMDWBFG6Q7

Read API Keys

Key: NKKJD39ZAQ8SG0ED

Note:

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key.**
- **Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

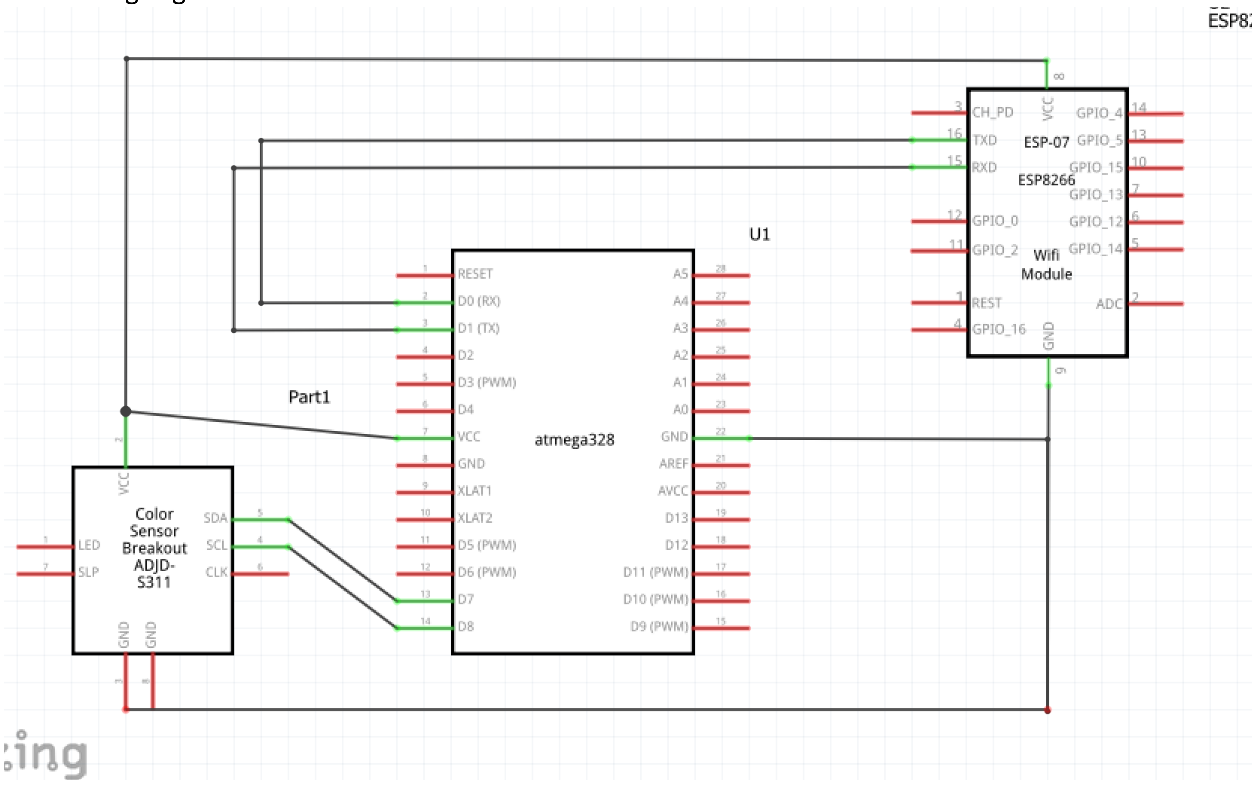
API Requests

Update a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=9HD0YXSMDWBFG6Q7&fielc
```

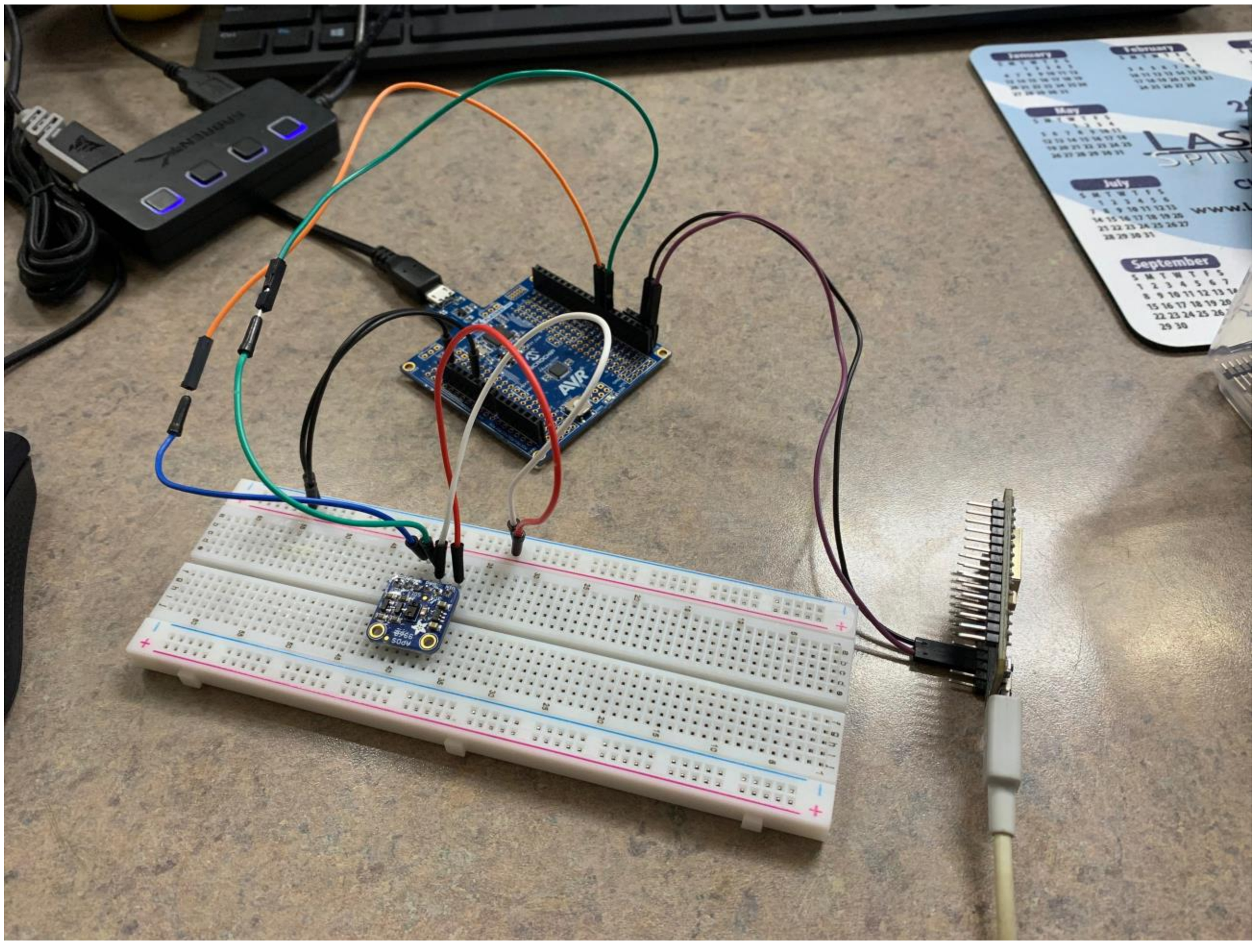
5. SCHEMATICS

Use fritzing.org



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

7. SCREENSHOT OF EACH DEMO (BOARD SETUP)



8. VIDEO LINKS OF EACH DEMO

9. GITHUB LINK OF THIS DA

https://github.com/regis-shaquille/submissions-SR/tree/master/Midterms/Midterm%202

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

"This assignment submission is my own, original work".
Shaquille Regis