CPE301 – SPRING 2019

Design Assignment 6

Student Name: Gabriela Cuicas
Student #: 5002960022
Student Email: cuicas@unlv.nevada.edu
Primary Github address: https://github.com/cuicattack/cat1
Directory: https://github.com/cuicattack/cat/Cat1Assn12_DA_6

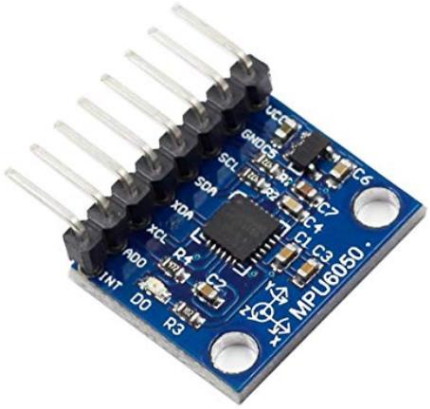
Submit the following for all Labs:

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/DA, 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).

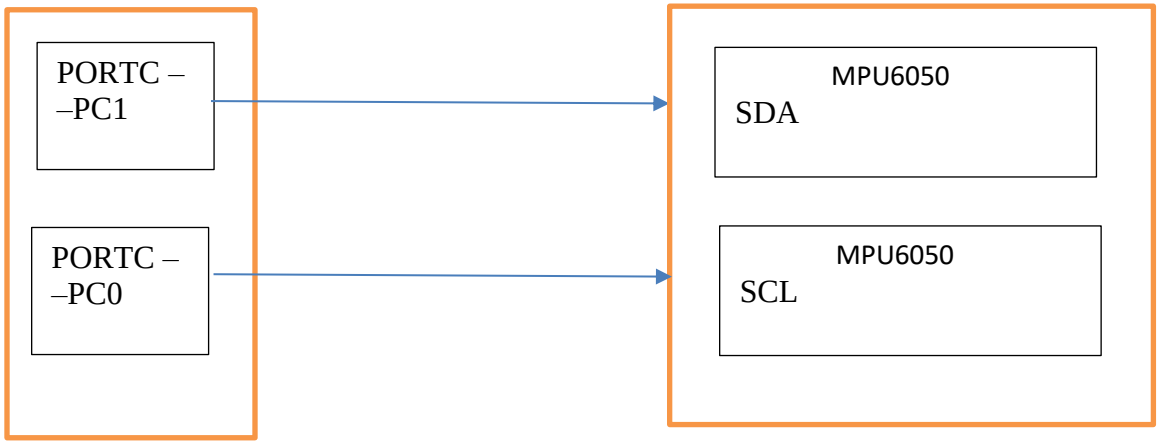
1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

List of Components used

MPU6050



Block diagram



2. DEVELOPED C CODE OF TASK 1

```
#define F_CPU 8000000UL          /* Define CPU clock Frequency 8MHz */
#include <avr/io.h>              /* Include AVR std. library file */
#include <util/delay.h>          /* Include delay header file */
#include <inttypes.h>            /* Include integer type header file */
#include <stdlib.h>              /* Include standard library file */
#include <stdio.h>               /* Include standard I/O library file */
#include "MPU6050_res_define.h" /* Include MPU6050 register define file */
#include "I2C_H_file.h" /* Include I header file */
#include "USART_RS232_H_file.h"

float Acc_x,Acc_y,Acc_z,Temperature,Gyro_x,Gyro_y,Gyro_z;

void Gyro_Init()                 /* Gyro initialization function */
{
        _delay_ms(150);          /* Power up time >100ms */
        I2C_Start_Wait(0xD0);    /* Start with device write address */
        I2C_Write(SMPLRT_DIV);   /* Write to sample rate register */
        I2C_Write(0x07);         /* 1KHz sample rate */
        I2C_Stop();

        I2C_Start_Wait(0xD0);
        I2C_Write(PWR_MGMT_1);   /* Write to power management register */
        I2C_Write(0x01);         /* X axis gyroscope reference frequency */
        I2C_Stop();

        I2C_Start_Wait(0xD0);
        I2C_Write(CONFIG);       /* Write to Configuration register */
        I2C_Write(0x00);         /* Fs = 8KHz */
        I2C_Stop();

        I2C_Start_Wait(0xD0);
        I2C_Write(GYRO_CONFIG); /* Write to Gyro configuration register */
        I2C_Write(0x18);         /* Full scale range +/- 2000 degree/C */
        I2C_Stop();

        I2C_Start_Wait(0xD0);
        I2C_Write(INT_ENABLE);   /* Write to interrupt enable register */
        I2C_Write(0x01);
        I2C_Stop();
}
```

```
void MPU_Start_Loc()
{
        I2C_Start_Wait(0xD0);   /* I2C start with device write address */
        I2C_Write(ACCEL_XOUT_H);/* Write start location address from where to read */
        I2C_Repeated_Start(0xD1);/* I2C start with device read address */
}

void Read_RawValue()
{
        MPU_Start_Loc(); /* Read Gyro values */
        Acc_x = (((int)I2C_Read_Ack()<<8) | (int)I2C_Read_Ack());
        Acc_y = (((int)I2C_Read_Ack()<<8) | (int)I2C_Read_Ack());
        Acc_z = (((int)I2C_Read_Ack()<<8) | (int)I2C_Read_Ack());
        Temperature = (((int)I2C_Read_Ack()<<8) | (int)I2C_Read_Ack());
        Gyro_x = (((int)I2C_Read_Ack()<<8) | (int)I2C_Read_Ack());
        Gyro_y = (((int)I2C_Read_Ack()<<8) | (int)I2C_Read_Ack());
        Gyro_z = (((int)I2C_Read_Ack()<<8) | (int)I2C_Read_Nack());
        I2C_Stop();
}

int main()
{
        char buffer[20], float_[10];
        float Xa,Ya,Za,t;
        float Xg=0,Yg=0,Zg=0;
        I2C_Init();             /* Initialize I2C */
        Gyro_Init();            /* Initialize Gyro */
        USART_Init(9600);       /* Initialize USART with 9600 baud rate */

        while(1)
        {
                Read_RawValue();

                /* Divide raw value by sensitivity scale factor */
                Xa = Acc_x/16384.0;
                Ya = Acc_y/16384.0;
                Za = Acc_z/16384.0;

                Xg = Gyro_x/16.4;
                Yg = Gyro_y/16.4;
                Zg = Gyro_z/16.4;

                /* Convert temperature in /c using formula */
```

```
                /* Divide raw value by sensitivity scale factor */
                Xa = Acc_x/16384.0;
                Ya = Acc_y/16384.0;
                Za = Acc_z/16384.0;

                Xg = Gyro_x/16.4;
                Yg = Gyro_y/16.4;
                Zg = Gyro_z/16.4;

                /* Convert temperature in /c using formula */
                t = (Temperature/340.00)+36.53;

                /* Take values in buffer to send all parameters over USART */
                dtostrf( Xa, 3, 2, float_ );
                sprintf(buffer," Ax = %s g\t",float_);
                USART_SendString(buffer);

                dtostrf( Ya, 3, 2, float_ );
                sprintf(buffer," Ay = %s g\t",float_);
                USART_SendString(buffer);

                dtostrf( Za, 3, 2, float_ );
                sprintf(buffer," Az = %s g\t",float_);
                USART_SendString(buffer);

                dtostrf( t, 3, 2, float_ );
                /* 0xF8 Ascii value of degree on serial */
                sprintf(buffer," T = %s%cC\t",float_,0xF8);
                USART_SendString(buffer);

                dtostrf( Xg, 3, 2, float_ );
                sprintf(buffer," Gx = %s%c/s\t",float_,0xF8);
                USART_SendString(buffer);

                dtostrf( Yg, 3, 2, float_ );
                sprintf(buffer," Gy = %s%c/s\t",float_,0xF8);
                USART_SendString(buffer);

                dtostrf( Zg, 3, 2, float_ );
                sprintf(buffer," Gz = %s%c/s\r\n",float_,0xF8);
                USART_SendString(buffer);
        }
}
```

C files.
I2C_c_File.c

```
#include "I2C_H_file.h"
void I2C_Init()
{
        TWBR = BITRATE(TWSR = 0x00);
}
uint8_t I2C_Start(char slave_write_address)
{
        uint8_t status;
        TWCR = (1<<TWSTA)|(1<<TWEN)|(1<<TWINT);
        while (!(TWCR & (1<<TWINT)));
        status = TWSR & 0xF8;
        if (status != 0x08)
        return 0;
        TWDR = slave_write_address;
        TWCR = (1<<TWEN)|(1<<TWINT);
        while (!(TWCR & (1<<TWINT)));
        status = TWSR & 0xF8;
        if (status == 0x18)
        return 1;
        if (status == 0x20)
        return 2;
        else
        return 3;
}

uint8_t I2C_Repeated_Start(char slave_read_address)
{
        uint8_t status;
        TWCR = (1<<TWSTA)|(1<<TWEN)|(1<<TWINT);
        while (!(TWCR & (1<<TWINT)));
        status = TWSR & 0xF8;
        if (status != 0x10)
        return 0;
        TWDR = slave_read_address;
        TWCR = (1<<TWEN)|(1<<TWINT);
        while (!(TWCR & (1<<TWINT)));
        status = TWSR & 0xF8;
        if (status == 0x40)
        return 1;
        if (status == 0x20)
        return 2;
        else
        return 3;
```

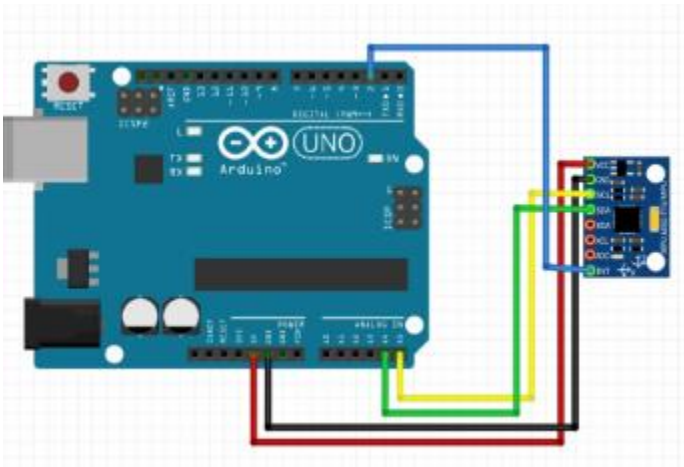
```
uint8_t I2C_Repeated_Start(char slave_read_address)
{
        uint8_t status;
        TWCR = (1<<TWSTA)|(1<<TWEN)|(1<<TWINT);
        while (!(TWCR & (1<<TWINT)));
        status = TWSR & 0xF8;
        if (status != 0x10)
        return 0;
        TWDR = slave_read_address;
        TWCR = (1<<TWEN)|(1<<TWINT);
        while (!(TWCR & (1<<TWINT)));
        status = TWSR & 0xF8;
        if (status == 0x40)
        return 1;
        if (status == 0x20)
        return 2;
        else
        return 3;
}

void I2C_Stop()
{
        TWCR=(1<<TWSTO)|(1<<TWINT)|(1<<TWEN);
        while(TWCR & (1<<TWSTO));
}
void I2C_Start_Wait(char slave_write_address)
{
        uint8_t status;
        while (1)
        {
                TWCR = (1<<TWSTA)|(1<<TWEN)|(1<<TWINT);
                while (!(TWCR & (1<<TWINT)));
                status = TWSR & 0xF8;
                if (status != 0x08)
                continue;
                TWDR = slave_write_address;
                TWCR = (1<<TWEN)|(1<<TWINT);
                while (!(TWCR & (1<<TWINT)));
                status = TWSR & 0xF8;
                if (status != 0x18 )
                {
                        I2C_Stop();
                        continue;
```

```
                continue;
                TWDR = slave_write_address;
                TWCR = (1<<TWEN)|(1<<TWINT);
                while (!(TWCR & (1<<TWINT)));
                status = TWSR & 0xF8;
                if (status != 0x18 )
                {
                        I2C_Stop();
                        continue;
                }
                break;
        }
}

uint8_t I2C_Write(char data)
{
        uint8_t status;
        TWDR = data;
        TWCR = (1<<TWEN)|(1<<TWINT);
        while (!(TWCR & (1<<TWINT)));
        status = TWSR & 0xF8;
        if (status == 0x28)
        return 0;
        if (status == 0x30)
        return 1;
        else
        return 2;
}

char I2C_Read_Ack()
{
        TWCR=(1<<TWEN)|(1<<TWINT)|(1<<TWEA);
        while (!(TWCR & (1<<TWINT)));
        return TWDR;
}

char I2C_Read_Nack()
{
        TWCR=(1<<TWEN)|(1<<TWINT);
        while (!(TWCR & (1<<TWINT)));
        return TWDR;
}
```

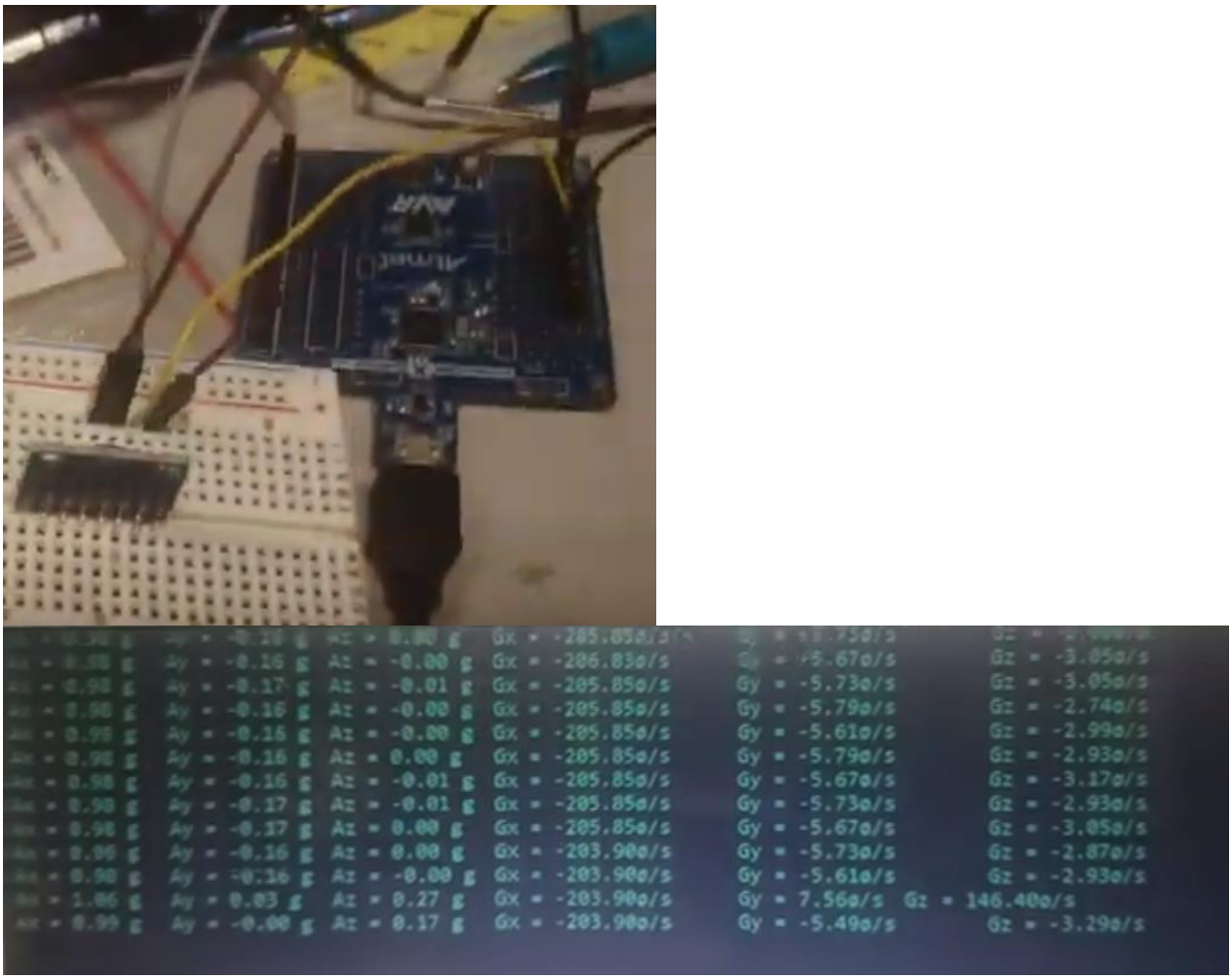
USART.c

```
#include "USART_RS232_H_file.h" /* Include USART header file */
void USART_Init(unsigned long BAUDRATE) /* USART initialize function */
{
        UCSRB |= (1 << RXEN) | (1 << TXEN);     /* Enable USART transmitter and receiver */
        UCSRC |= (1 << URSEL)| (1 << UCSZ0) | (1 << UCSZ1);     /* Write USCRC for 8 bit data and 1 stop bit */
        UBRRL = BAUD_PRESCALE;  /* Load UBRRL with lower 8 bit of prescale value */
        UBRRH = (BAUD_PRESCALE >> 8);   /* Load UBRRH with upper 8 bit of prescale value */
}

char USART_RxChar()     /* Data receiving function */
{
        while (!(UCSRA & (1 << RXC)));  /* Wait until new data receive */
        return(UDR);    /* Get and return received data */
}

void USART_TxChar(char data)    /* Data transmitting function */
{
        UDR = data;     /* Write data to be transmitting in UDR */
        while (!(UCSRA & (1<<UDRE)));   /* Wait until data transmit and buffer get empty */
}

void USART_SendString(char *str)        /* Send string of USART data function */
{
        int i=0;
        while (str[i]!=0)
        {
                USART_TxChar(str[i]);   /* Send each char of string till the NULL */
                i++;
        }
}
```

3. SCHEMATICS



4. SCREENSHOT OF EACH DEMO (BOARD SETUP)



5. VIDEO LINKS OF EACH DEMO

https://www.youtube.com/watch?v=ee5WfpwIwk4&list=PL5KkHOalGXZ71ilBIr32ij3e1gJDyAUsu&index=11&t=0s

6. GITHUB LINK OF THIS DA

https://github.com/cuicattack/cat/ Cat1Assn12_DA_6

Student Academic Misconduct Policy

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

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

NAME OF THE STUDENT