CPE301 – SPRING 2019

Design Assignment 6

Student Name: Rimon Sawa
Student #: 5001529281
Student Email: sawar1@unlv.nevada.edu
Primary Github address: https://github.com/sawar1/UNLV301
Directory: DA6

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

Xplained mini
USB
Male/female wires
MPU-6050

2. DEVELOPED C CODE

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <inttypes.h>
#include <stdlib.h>
#include <stdio.h>
#include "MPU6050_res_define.h"
#include "I2C_Master_H_file.h"
#include "USART_RS232_H_file.h"
 float
Acc_x,Acc_y,Acc_z,Temperature,Gyro_x,Gyro_y,Gyro_z;

void MPU6050_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;     float Xg=0,Yg=0,Zg=0;
	I2C_Init();          //Initialize I2C
	MPU6050_Init();             //Initialize MPU6050
	USART_Init(9600);    //Initialize USART                        while(1)
	{
		Read_RawValue();
	//Divide raw value by sensitivity scale factor to get real values
		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;

	//Output values
		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( 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);
                _delay_ms(1000);
        }
}



#include "USART_RS232_H_file.h"
 void USART_Init(unsigned long
BAUDRATE)
{
        UCSR0B = (1<<RXEN0)|(1<<TXEN0);// Enable USART transmitter and receiver
                // Write USCRC for 8 bit data and 1 stop bit
        UBRR0L = BAUD_PRESCALE;     // Load UBRRL with lower 8 bit of prescale value
        UBRR0H = (BAUD_PRESCALE >> 8);// Load UBRRH with upper 8 bit of prescale value
}
 char USART_RxChar()                // Data receiving function
{
         while (!(UCSR0A & (1 << RXC0))); // Wait until new data receive
        return(UDR0);                       // Get and return received data
}

void USART_TxChar(char data) {     // Data transmitting function

        UDR0 = data;                           //Write data to be transmitting in UDR
        while (!(UCSR0A & (1<<UDRE0)));// 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++;
```

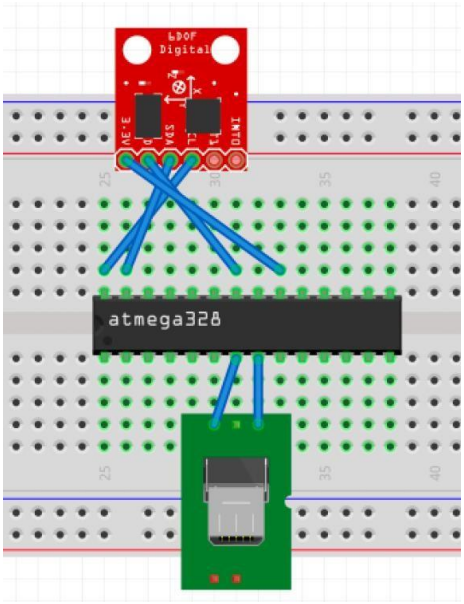
```
	}
}
```

```
#include "I2C_Master_H_file.h"

void I2C_Init()
	// I2C initialize function
{
	TWBR = BITRATE(TWSR = 0x00);// Get bit rate register value by formula
}

uint8_t I2C_Start(char slave_write_address)		// I2C start function
{
	uint8_t status;

 TWCR = (1<<TWSTA)|(1<<TWEN)|(1<<TWINT);//Enable TWI, generate start condition and clear
interrupt flag
 while (!(TWCR & (1<<TWINT)));// Wait until TWI finish its current job (start condition)
	status = TWSR & 0xF8;
	// Read TWI status register with masking lower three bits
	if (status != 0x08)
	// Check weather start condition transmitted successfully or not?
	return 0;
	// If not then return 0 to indicate start condition fail
	TWDR = slave_write_address; // If yes then write SLA+W in TWI data register
TWCR = (1<<TWEN)|(1<<TWINT);// Enable TWI and clear interrupt flag		while
(!(TWCR & (1<<TWINT)));// Wait until TWI finish its current job (Write operation)
	status = TWSR & 0xF8;
	// Read TWI status register with masking lower three bits
	if (status == 0x18)
	// Check weather SLA+W transmitted & ack received or not?
	return 1;
	// If yes then return 1 to indicate ack received i.e. ready to accept data byte
	if (status == 0x20)
	// Check weather SLA+W transmitted & nack received or not?
	return 2;
	// If yes then return 2 to indicate nack received i.e. device is busy
	else
	return 3;
	// Else return 3 to indicate SLA+W failed
}
uint8_t I2C_Repeated_Start(char slave_read_address)// I2C repeated start function
{
	uint8_t status;

	TWCR = (1<<TWSTA)|(1<<TWEN)|(1<<TWINT); // Enable TWI, generate start condition
and clear interrupt flag
 while (!(TWCR & (1<<TWINT)));// Wait until TWI finish its current job (start condition)
	status = TWSR & 0xF8;
	// Read TWI status register with masking lower three bits
```

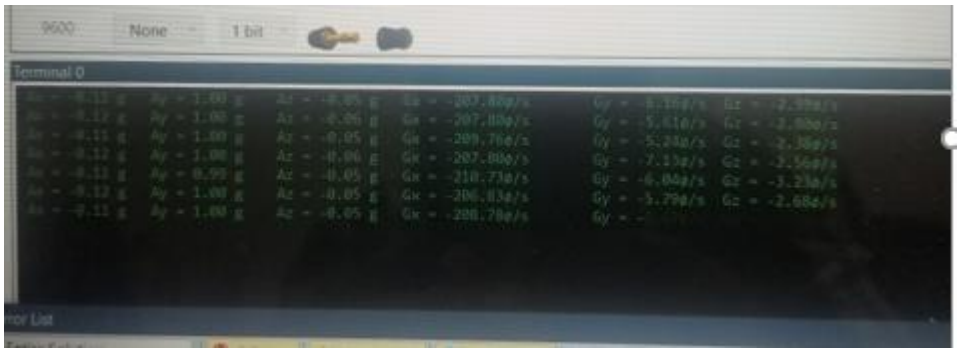
```
	if (status != 0x10)
	// Check weather repeated start condition transmitted successfully or not?
return 0;
	// If no then return 0 to indicate repeated start condition fail
	TWDR = slave_read_address; // If yes then write SLA+R in TWI data register
TWCR = (1<<TWEN)|(1<<TWINT);// Enable TWI and clear interrupt flag   while (!(TWCR
& (1<<TWINT)));// Wait until TWI finish its current job (Write operation)
	status = TWSR & 0xF8;
	// Read TWI status register with masking lower three bits
	if (status == 0x40)
	// Check weather SLA+R transmitted & ack received or not?
	return 1;
	// If yes then return 1 to indicate ack received
	if (status == 0x20)
	// Check weather SLA+R transmitted & nack received or not?
	return 2;
	// If yes then return 2 to indicate nack received i.e. device is busy
	else
	return 3;
	// Else return 3 to indicate SLA+W failed
}

void I2C_Stop()
	// I2C stop function
{
	TWCR=(1<<TWSTO)|(1<<TWINT)|(1<<TWEN);// Enable TWI, generate stop condition and
clear interrupt flag	while(TWCR & (1<<TWSTO)); // Wait until stop condition
execution
}
void I2C_Start_Wait(char slave_write_address)// I2C start wait function
{
	uint8_t status;

	while (1)
	{
		TWCR = (1<<TWSTA)|(1<<TWEN)|(1<<TWINT); // Enable TWI, generate start
condition and clear interrupt flag		while (!(TWCR & (1<<TWINT)));//
Wait until TWI finish its current job
(start condition)
		status = TWSR & 0xF8;
	// Read TWI status register with masking lower three bits
		if (status != 0x08)
	// Check weather start condition transmitted successfully or not?
		continue;
	// If no then continue with start loop again
		TWDR = slave_write_address; // If yes then write SLA+W in TWI data
register		TWCR = (1<<TWEN)|(1<<TWINT);// Enable TWI and clear interrupt flag
	while (!(TWCR & (1<<TWINT)));// Wait until TWI finish its current job
		status = TWSR & 0xF8;
	// Read TWI status register with masking lower three bits		if (status
!= 0x18 ) // Check weather SLA+W transmitted & ack received or not?
		{
			I2C_Stop();
	// If not then generate stop condition
```

```
                    continue;
        // continue with start loop again
                }
                break;
        // If yes then break loop
        }
}
uint8_t I2C_Write(char data)// I2C write function
{
        uint8_t status;

        TWDR = data;
        // Copy data in TWI data register
        TWCR = (1<<TWEN)|(1<<TWINT);// Enable TWI and clear interrupt flag
while (!(TWCR & (1<<TWINT)));// Wait until TWI finish its current job (Write
operation)
        status = TWSR & 0xF8;
        // Read TWI status register with masking lower three bits
        if (status == 0x28)
        // Check weather data transmitted & ack received or not?
        return 0;
        // If yes then return 0 to indicate ack received
        if (status == 0x30)
        // Check weather data transmitted & nack received or not?
        return 1;
        // If yes then return 1 to indicate nack received
        else
        return 2;
        // Else return 2 to indicate data transmission failed
}
char I2C_Read_Ack()         // I2C read ack function
{
 TWCR=(1<<TWEN)|(1<<TWINT)|(1<<TWEA);//  Enable  TWI,  generation  of  ack  and  clear
interrupt flag   while (!(TWCR & (1<<TWINT)));// Wait until TWI finish its current job
(read operation)
        return TWDR;
        // Return received data
}

char I2C_Read_Nack()        // I2C read nack function  {
        TWCR=(1<<TWEN)|(1<<TWINT);         // Enable TWI and clear interrupt flag  while
        (!(TWCR & (1<<TWINT)));// Wait until TWI finish its current job (read
operation)
        return TWDR;
        // Return received data
}
```

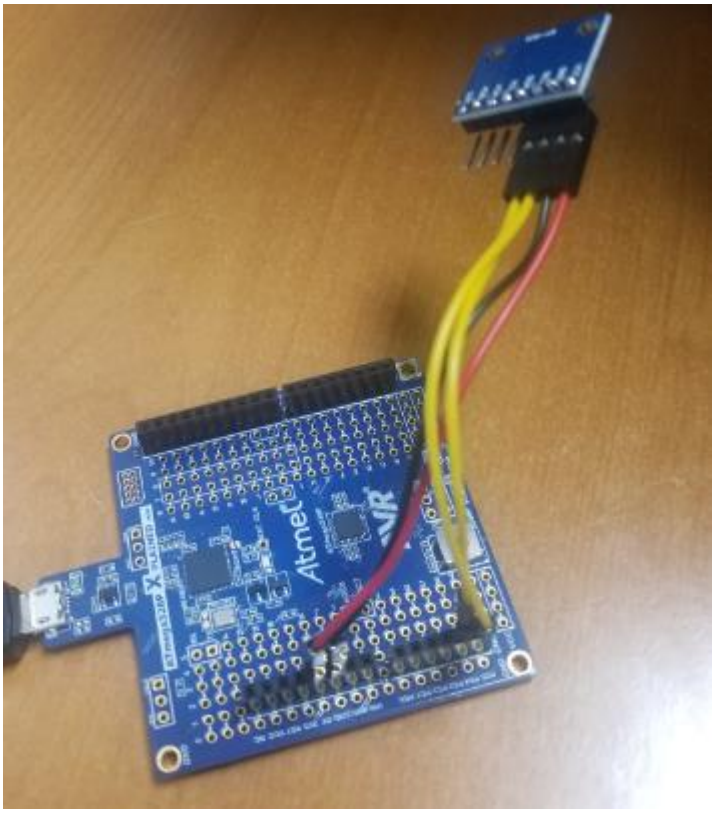
3. SCHEMATICS



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



5. SCREENSHOT OF EACH DEMO (BOARD SETUP)



6. VIDEO LINKS OF EACH DEMO

https://youtu.be/-tCmJTOKGSI

7. GITHUB LINK OF THIS DA

https://github.com/sawar1/UNLV301

Student Academic Misconduct Policy

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

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

RIMON SAWA