CPE301 – SPRING 2019

Design Assignment 6

Student Name: Rocky Yasuaki Gonzalez
Student #: 5003229733
Student Email: gonzar14@unlv.nevada.edu
Primary Github address: https://github.com/rockyg1995/ihswppdar.git
Directory: C:\Users\rocky\Documents\CpE 301+L - Embedded Systems Design\CpE 301\Repository\DesignAssignments\DA6

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 FLOW DIAGRAMS

Atmega328PB Xplained Mini
Micro USB Cable (Power Supply)
Breadboard
MPU6050
Male-to-Male Wires
10k Ohm Resistor (x2)

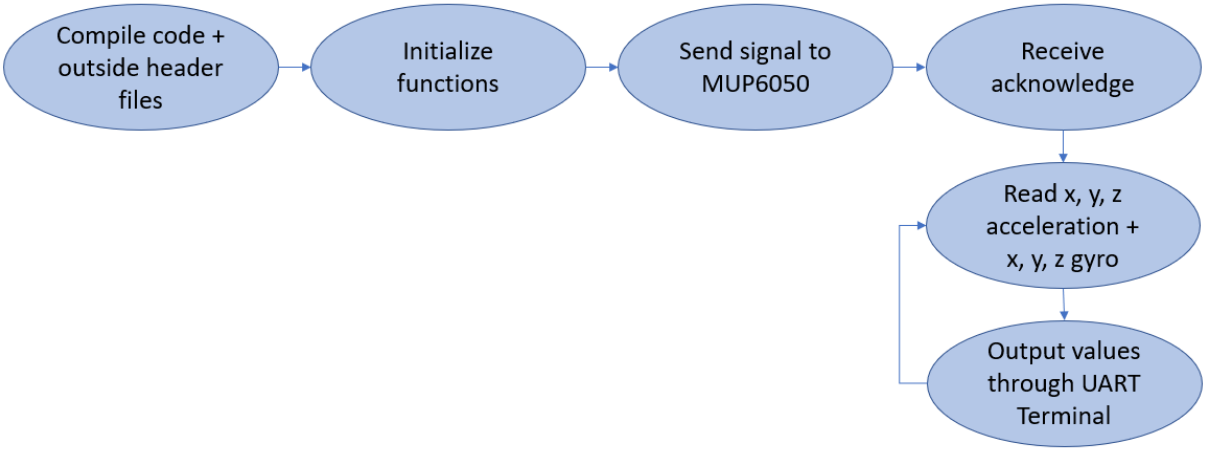

Figure 1 – Flow Chart for Coding Algorithm in Task

2. INITIAL/DEVELOPED CODE OF TASK

```
/*
 * DA6.c
 *
 * Created: 5/4/2019 9:58:14 AM
 * Author: RYG95
 */

#ifndef F_CPU
#define F_CPU 16000000UL
#endif

#include <avr/io.h>          /* Include AVR input/output file */
#include <util/delay.h>      /* Include delay functions file */
#include <math.h>            /* Include math functions file */
#include <stdlib.h>          /* Include standard library file */
#include <stdio.h>           /* Include standard input/output file */
#include "MPU6050_def.h"     /* Include MPU6050 register define file */
#include "i2c_master.h"      /* Include I2C Master header file */
#include "uart.h"            /* Include USART header file */

#define MPU6050_WRITE 0xD0
#define MPU6050_READ 0xD1

float Acc_x;
float Acc_y;
float Acc_z;
```

```
float Gyro_x;
float Gyro_y;
float Gyro_z;

void init_uart(uint16_t baudrate){

	uint16_t UBRR_val = (F_CPU/16)/(baudrate-1);

	UBRR0H = UBRR_val >> 8;
	UBRR0L = UBRR_val;

	UCSR0B |= (1<<TXEN0) | (1<<RXEN0) | (1<<RXCIE0); // UART TX (Transmit - senden)
	                                                 // einschalten
	UCSR0C |= (1<<USBS0) | (3<<UCSZ00); // Modus Asynch 8N1 (8 Datenbits, No Parity, 1
	                                    // Stopbit)
}

void uart_putc(unsigned char c){

	while(!(UCSR0A & (1<<UDRE0)));     // wait until sending is possible
	UDR0 = c;                          // output character saved in c
}

void uart_puts(char *s){
	while(*s){
		uart_putc(*s);
		s++;
	}
}

void init_MPU6050(void){
	_delay_ms(150);             /* Power up time >100ms */
	i2c_start(MPU6050_WRITE);   // Set Gyroscope Sample Rate = 1 KHz, Accelerometer
	                            // Sample Rate = 1 KHz (default)
	i2c_write(SMPLRT_DIV);      // Sample Rate is generated by dividing the gyroscope
	                            // output rate by SMPLRT_DIV
	i2c_write(0x07);            // Gyroscope Output Rate = 8kHz, Sample Rate =
	                            // Gyroscope Output Rate / (1 + SMPLRT_DIV)
	i2c_stop();

	i2c_start(MPU6050_WRITE);
	i2c_write(PWR_MGMT_1);
	i2c_write(0x01); // PLL with X axis gyroscope reference
	i2c_stop();

	i2c_start(MPU6050_WRITE);
	i2c_write(CONFIG);   // Frame Synchronization & Digital Low Pass Filter (DLPF)
	                     // setting
	i2c_write(0x00);
	i2c_stop();

	i2c_start(MPU6050_WRITE);
	i2c_write(GYRO_CONFIG); //gyroscopes' scale range=FS_SEL selects = 11 = ± 2000 °/s
	i2c_write(0x18);        // accelerometer range = ± 2g (default)
	i2c_stop();

	i2c_start(MPU6050_WRITE);
	i2c_write(INT_ENABLE); // DATA_RDY_EN = 1
```

```
	i2c_write(0x01);
	i2c_stop();
}

void getreading(void){

	i2c_start(MPU6050_WRITE);
	i2c_write(ACCEL_XOUT_H); // set pointer
	i2c_stop();
	i2c_start(MPU6050_READ);
	Acc_x = (((int)i2c_read_ack()<<8) | (int)i2c_read_ack());
	i2c_stop();

	i2c_start(MPU6050_WRITE);
	i2c_write(ACCEL_YOUT_H); // set pointer
	i2c_stop();
	i2c_start(MPU6050_READ);
	Acc_y = (((int)i2c_read_ack()<<8) | (int)i2c_read_ack());
	i2c_stop();

	i2c_start(MPU6050_WRITE);
	i2c_write(ACCEL_ZOUT_H); // set pointer
	i2c_stop();
	i2c_start(MPU6050_READ);
	Acc_z = (((int)i2c_read_ack()<<8) | (int)i2c_read_ack());
	i2c_stop();

	i2c_start(MPU6050_WRITE);
	i2c_write(GYRO_XOUT_H); // set pointer
	i2c_stop();
	i2c_start(MPU6050_READ);
	Gyro_x = (((int)i2c_read_ack()<<8) | (int)i2c_read_ack());
	i2c_stop();

	i2c_start(MPU6050_WRITE);
	i2c_write(GYRO_YOUT_H); // set pointer
	i2c_stop();
	i2c_start(MPU6050_READ);
	Gyro_y = (((int)i2c_read_ack()<<8) | (int)i2c_read_ack());
	i2c_stop();

	i2c_start(MPU6050_WRITE);
	i2c_write(GYRO_ZOUT_H); // set pointer
	i2c_stop();
	i2c_start(MPU6050_READ);
	Gyro_z = (((int)i2c_read_ack()<<8) | (int)i2c_read_ack());
	i2c_stop();
}

int main(void){
	char buffer[20], float_[10];
	float Xa;
	float Ya;
	float Za;
	float Xg;
	float Yg;
	float Zg;
	init_uart(9600);
```

```
	i2c_init();
	init_MPU6050();

	while(1){
		getreading();
		Xa = Acc_x/16384.0;  /* Divide raw value by sensitivity scale factor to get
		                        real values */
		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;

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

		dtostrf( Ya, 3, 2, float_ );
		sprintf(buffer,"%s Ya, ",float_);
		USART_SendString(buffer);

		dtostrf( Za, 3, 2, float_ );
		sprintf(buffer,"%s Za, ",float_);
		USART_SendString(buffer);

		dtostrf( Xg, 3, 2, float_ );
		sprintf(buffer,"%s Xg, ",float_);
		USART_SendString(buffer);

		dtostrf( Yg, 3, 2, float_ );
		sprintf(buffer,"%s Yg, ",float_);
		USART_SendString(buffer);

		dtostrf( Zg, 3, 2, float_ );
		sprintf(buffer,"%s Zg, ",float_);
		USART_SendString(buffer);

		USART_SendString("\r\n");
		_delay_ms(1000);
	}

	return 0;
}
```

3. SCHEMATICS

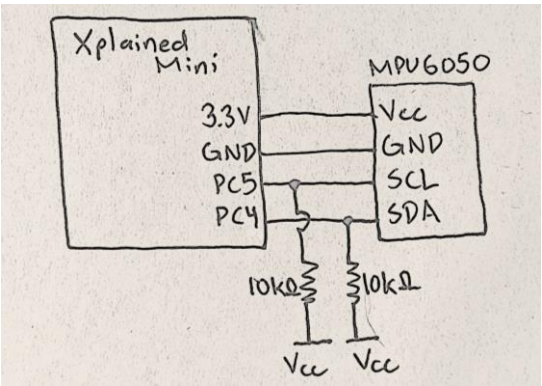

Figure 2 – Atmega328P/PB Xplained Minis + MPU6050 I2C Module

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

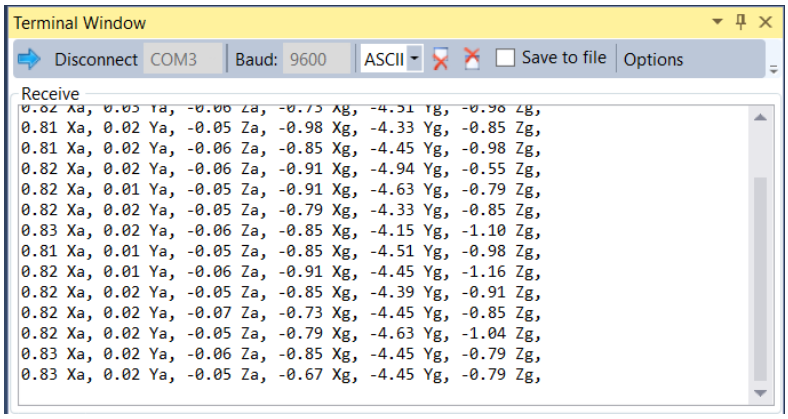

Figure 3 – Output Terminal for Configuration of SPI Module

There had been problems communicating data when the MPU6050 moved greatly, the terminal would stop reading values. MPU6050 could only change due to small changes in movement.

5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

6.

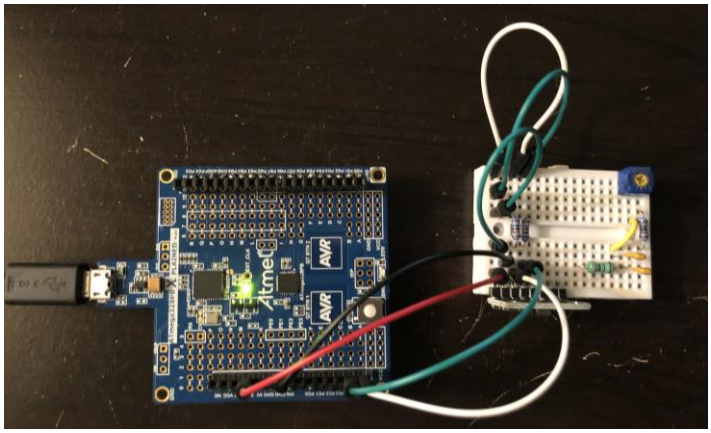

Figure 4 – Connecting the Xplained Mini + MPU6050

7. VIDEO LINKS OF EACH DEMO

N/A

8. GITHUB LINK OF THIS DA

https://github.com/rockyg1995/ihswppdar/tree/master/DesignAssignments/DA6

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

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