CPE301 – SPRING 2019

Design Assignment 6A

Student Name: Cody Hudson

Student #: 8000603642

Student Email: hudsoc1@unlv.nevada.edu

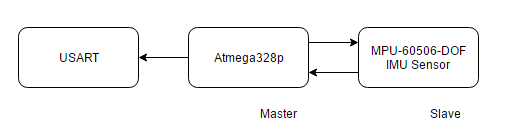
Primary Github address: https://github.com/eed911/class\_proj.git

Directory:

https://github.com/eed911/class\_proj/tree/master/DesignAssignments/DA6A/Project\_6A

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

* Task 1:
  + ATMEGA328p
  + MPU-6050 6-DOF IMU Sensor



1. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1**

**Interface the provided MPU-6050 6-DOF IMU Sensor to the ATmega328p using the I2C interface. Using the earlier developed code for UART, display the accelerometer and gyro data to the UART Terminal**

/\*

\* DA6.c

\*

\* Created: 5/4/2019 4:47:14 PM

\* Author : hudsoc1

\*/

#ifndef F\_CPU

#define F\_CPU 16000000UL

#endif

#include <avr/io.h>

#include <util/delay.h>

#include <math.h>

#include <stdlib.h> /\* Include standard library file \*/

#include <stdio.h> /\* Include standard library file \*/

#include "MPUlibraries/MPU6050\_def.h" /\* Include MPU6050 register define file \*/

#include "MPUlibraries/i2c\_master.h" /\* Include I2C Mater header file \*/

#include "MPUlibraries/uart.h" /\* Include USART header file \*/

#define MPU6050\_WRITE 0xD0

#define MPU6050\_READ 0xD1

/\*Creating Floating Variables for Gyro & Acceleration\*/

float Acc\_x;

float Acc\_y;

float Acc\_z;

float Gyro\_x;

float Gyro\_y;

float Gyro\_z;

/\*Setting up the Usart\*/

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 Asynchron 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++;

}

}

/\*Setting up the Gyro\*/

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){

// ACCEL X

i2c\_start(MPU6050\_WRITE);

i2c\_write(ACCEL\_XOUT\_H); // set pointer

i2c\_start(MPU6050\_READ);

Acc\_x = (((int)i2c\_read\_ack()<<8) | (int)i2c\_read\_ack());

// ACCEL Y

Acc\_y = (((int)i2c\_read\_ack()<<8) | (int)i2c\_read\_ack());

// ACCEL Z

Acc\_z = (((int)i2c\_read\_ack()<<8) | (int)i2c\_read\_ack());

// GYRO X

Gyro\_x = (((int)i2c\_read\_ack()<<8) | (int)i2c\_read\_ack());

// GYRO Y

Gyro\_y = (((int)i2c\_read\_ack()<<8) | (int)i2c\_read\_ack());

// GYRO Z

Gyro\_z = (((int)i2c\_read\_ack()<<8) | (int)i2c\_read\_ack());

i2c\_stop();

}

int main(void){

char buffer[20], float\_[10];

char\* PAX = "X Acceleration = ";

char\* PAY = "Y Acceleration = ";

char\* PAZ = "Z Acceleration = ";

char\* PGX = "X Gyro Position = ";

char\* PGY = "Y Gyro Position = ";

char\* PGZ = "Z Gyro Position = ";

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 for "X","Y", "Z" \*/

Ya = Acc\_y/16384.0;

Za = Acc\_z/16384.0;

Xg = Gyro\_x/131;

Yg = Gyro\_y/131;

Zg = Gyro\_z/131;

*dtostrf*( Xa, 3, 2, float\_ ); /\* Take values in buffer to send all parameters over USART \*/

*sprintf*(buffer,"%s, ",float\_);

USART\_SendString(PAX);

USART\_SendString(buffer);

*\_delay\_ms*(100);

*dtostrf*( Ya, 3, 2, float\_ ); /\* Take values in buffer to send all parameters over USART \*/

*sprintf*(buffer,"%s, ",float\_);

USART\_SendString(PAY);

USART\_SendString(buffer);

*\_delay\_ms*(100);

*dtostrf*( Za, 3, 2, float\_ ); /\* Take values in buffer to send all parameters over USART \*/

*sprintf*(buffer,"%s, ",float\_);

USART\_SendString(PAZ);

USART\_SendString(buffer);

USART\_SendString("\r\n");

*\_delay\_ms*(1000);

*dtostrf*( Xg, 3, 2, float\_ ); /\* Take values in buffer to send all parameters over USART \*/

*sprintf*(buffer,"%s, ",float\_);

USART\_SendString(PGX);

USART\_SendString(buffer);

*\_delay\_ms*(100);

*dtostrf*( Yg, 3, 2, float\_ ); /\* Take values in buffer to send all parameters over USART \*/

*sprintf*(buffer,"%s, ",float\_);

USART\_SendString(PGY);

USART\_SendString(buffer);

*\_delay\_ms*(100);

*dtostrf*( Zg, 3, 2, float\_ ); /\* Take values in buffer to send all parameters over USART \*/

*sprintf*(buffer,"%s, ",float\_);

USART\_SendString(PGZ);

USART\_SendString(buffer);

USART\_SendString("\r\n");

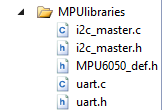
*\_delay\_ms*(1000);

}

return 0;

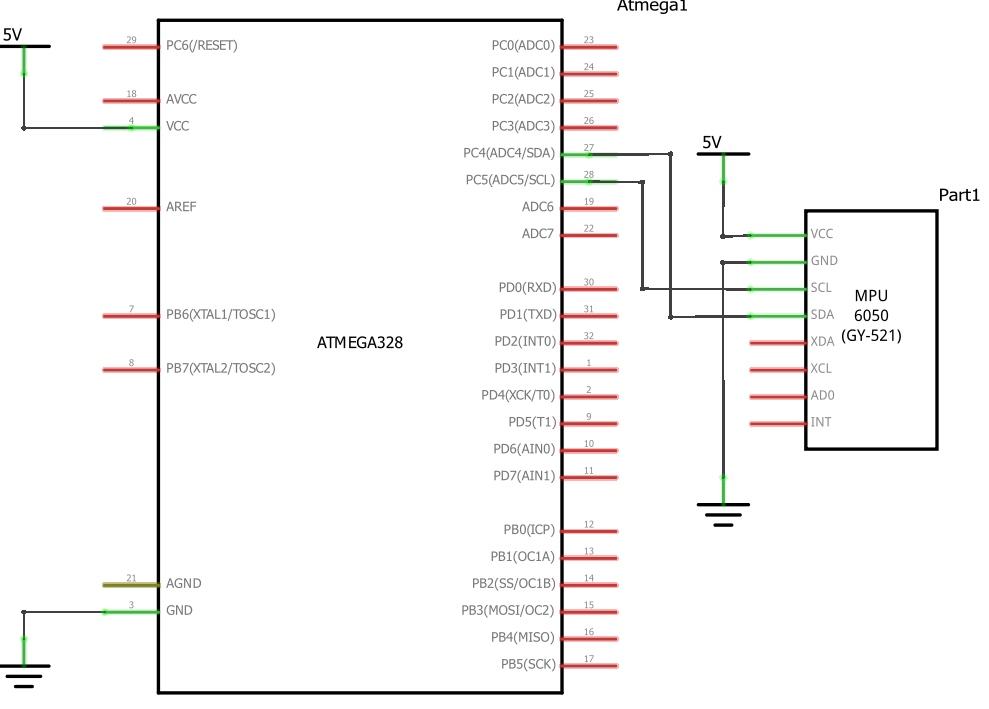
}

Included MPU libraries from given github



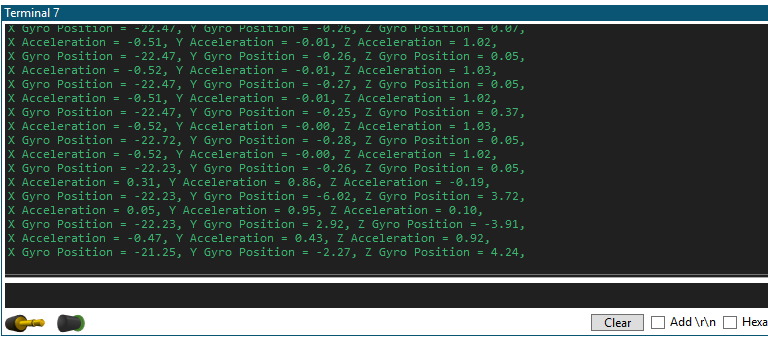
1. **SCHEMATICS**

Task 1:



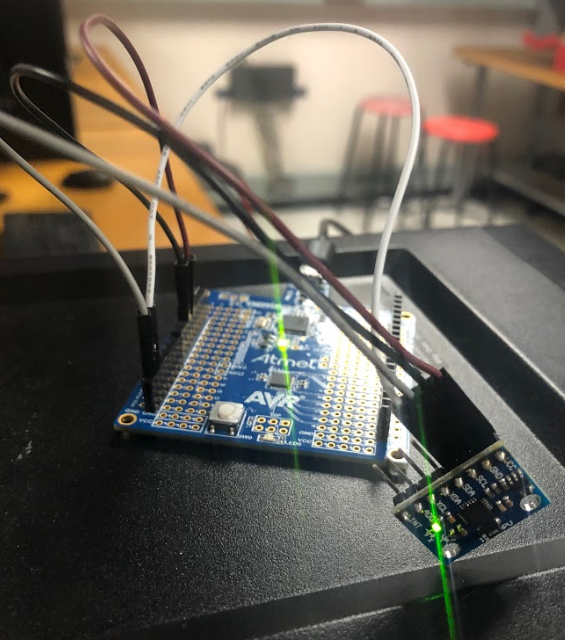
1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

Task1:



1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**

Task 1:



1. **VIDEO LINKS OF EACH DEMO**

Task 1:

https://youtu.be/lr\_vLXtpAHw

1. **GITHUB LINK OF THIS DA**

https://github.com/eed911/class\_proj.git

“This assignment submission is my own, original work”.

Cody Hudson