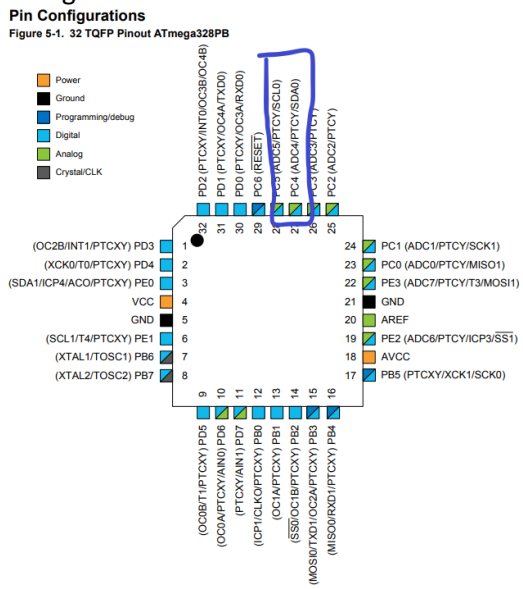
CPE301 – SPRING 2020

Design Assignment 6

Student Name: Minsung Cho
Student #: 2001446442
Student Email: chom3@unlv.nevada.edu
Primary Github address: https://github.com/cho-minsung
Directory: https://github.com/cho-minsung/assignment6

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

Atmega328PB



MPU-6050 6-DOF IMU Sensor

2. Task 1 code

```
#define F_CPU 16000000UL // 16MHz CPU Clock
#include <avr/io.h>
#include <util/delay.h>
#include <inttypes.h>
#include <stdlib.h>
#include <stdio.h>

//codes from Dr. Venki's repository
#include "MPU6050_def.h"
#include "i2c_master.h"
#include "uart.h"

// values from MPU6050 from Dr. Venki's repo
```

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

// function prototypes
void MPU6050_Init();
void MPU_Start_Loc();
void Read_RawValue();
void printForPlot();
void printForTerminal();

int main()
{
	//initialization codes
	I2C_Init();
	MPU6050_Init();
	USART_Init(9600);


	while(1)
	{
		Read_RawValue();
		display();
	}
}

// function to initialize MPU6050
void MPU6050_Init(){
	_delay_ms(150); // wait 150ms for power up
	I2C_Start_Wait(0xD0); // start the device with write address
	I2C_Write(SMPLRT_DIV); // write to the sample rate register
	I2C_Write(0x07); // use a 1KHz sample rate
	I2C_Stop(); // stop I2C

	I2C_Start_Wait(0xD0);
	I2C_Write(PWR_MGMT_1); // write to the power management register
	I2C_Write(0x01); // reference frequency for gyro x-axis
	I2C_Stop();

	I2C_Start_Wait(0xD0);
	I2C_Write(CONFIG); // write to config register
	I2C_Write(0x00); // Fs = 8KHz
	I2C_Stop();

	I2C_Start_Wait(0xD0);
	I2C_Write(GYRO_CONFIG); // write to gyro config register
	I2C_Write(0x18); // use the full-scale range
	I2C_Stop();

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

```
// start the MPU loc
void MPU_Start_Loc(){
	I2C_Start_Wait(0xD0); // start the device with 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
}

// read the gyroscope values
void Read_RawValue(){
	MPU_Start_Loc();
	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();
}

// function to send to terminal
void display(){
	char buffer[20], float_[10];
	float Xa = Acc_x/16384.0; // Divide raw value by sensitivity scale factor to get real
values
	float Ya = Acc_y/16384.0;
	float Za = Acc_z/16384.0;

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

	float t = (Temperature/340.00)+36.53; // convert temperature to C

	// print Xa value
	dtostrf( Xa, 3, 2, float_ );
	sprintf(buffer," Ax = %s g\t",float_);
	USART_SendString(buffer);
	// print Ya value
	dtostrf( Ya, 3, 2, float_ );
	sprintf(buffer," Ay = %s g\t",float_);
	USART_SendString(buffer);
	// print Za value
	dtostrf( Za, 3, 2, float_ );
	sprintf(buffer," Az = %s g\t",float_);
	USART_SendString(buffer);
	// print temperature value
	dtostrf( t, 3, 2, float_ );
	sprintf(buffer," T = %s%cC\r\n",float_,0xF8);
	USART_SendString(buffer);
	// print Xg value
```

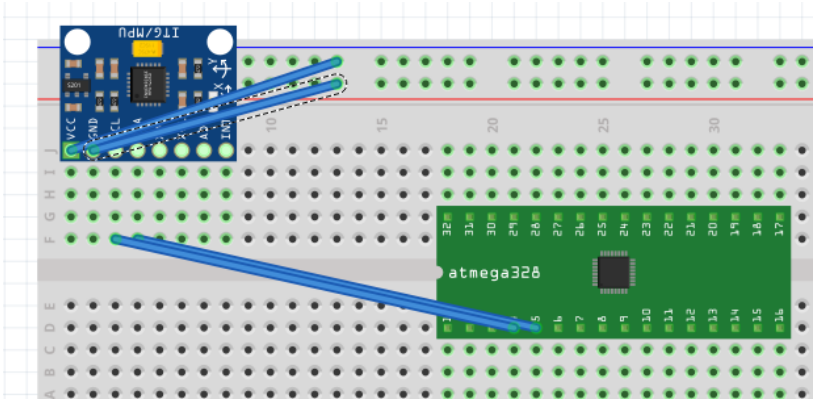
```
        dtostrf( Xg, 3, 2, float_ );
        sprintf(buffer," Gx = %s%c/s\t",float_,0xF8);
        USART_SendString(buffer);
        // print Yg value
        dtostrf( Yg, 3, 2, float_ );
        sprintf(buffer," Gy = %s%c/s\t",float_,0xF8);
        USART_SendString(buffer);
        // print Zg value
        dtostrf( Zg, 3, 2, float_ );
        sprintf(buffer," Gz = %s%c/s\r\n",float_,0xF8);
        USART_SendString(buffer);
        _delay_ms(1000);
}
```

- All the rest of the codes included were from Dr. Venki's repo.

3. Task 2 code

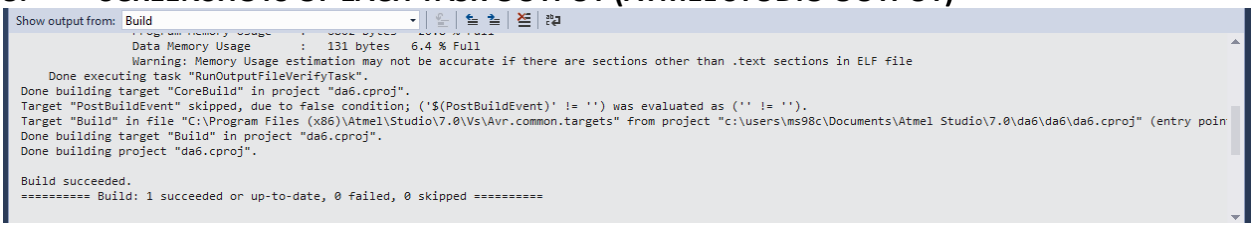
```
#define ACCELEROMETER_SENSITIVITY 16384.0
#define GYROSCOPE_SENSITIVITY 16.4
#define dt 0.01

void ComplementaryFilter()
{
        folat pitchAcc, rollAcc;

        pitch += ((float)gyrData[0] / GYROSCOPE_SENSITIVITY) * dt;
        roll -= ((float)gyrData[1] / GYROSCOPE_SENSITIVITY) * dt;

        int forceMagnitudeApprox = \ abs(accData[0]) + abs(accData[1]) + abs(accData[2]);

        if (forceMagnitudeApprox > 8192 && forceMagnitudeApprox < 32768)
        {
                pitchAcc = atan2f((float)accData[1], (float)accData[2]) * 180 / M_PI;
                pitch = pitch * 0.98 + pitchAcc * 0.02;

                rollAcc = atan2f((float)accData[0], (float)accData[2]) * 180 /M_PI;
                roll = roll * 0.98 + rollAcc * 0.02;
        }
}
```

4. SCHEMATICS

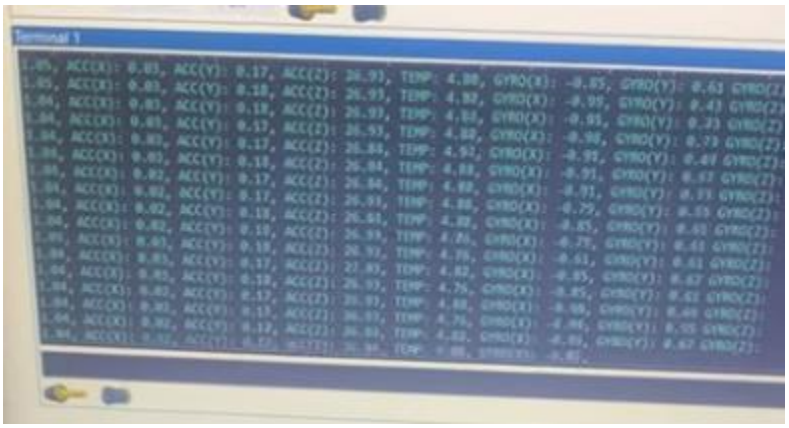


Pin C4&5 used.
VCC and GND used.

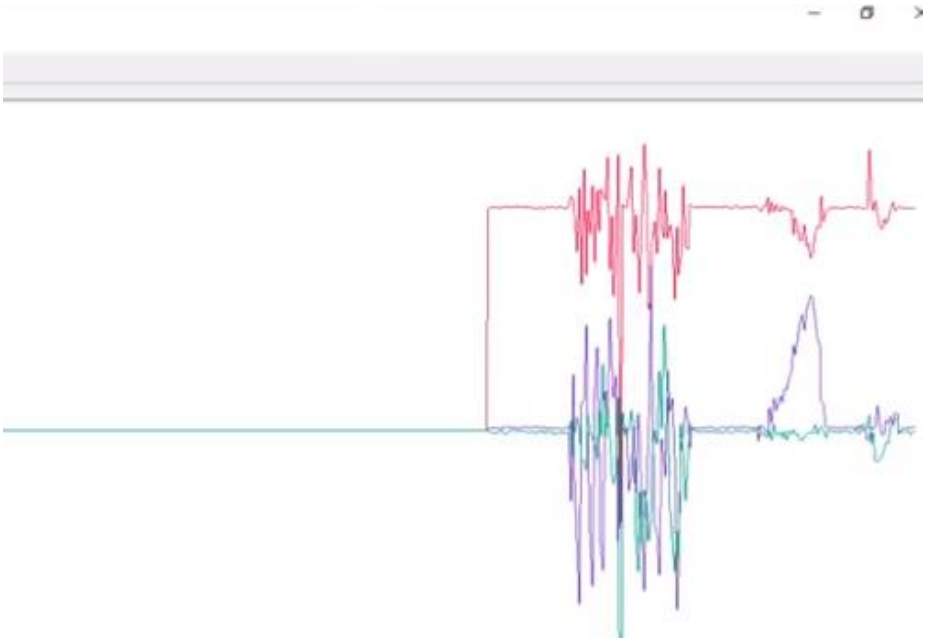
5. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)



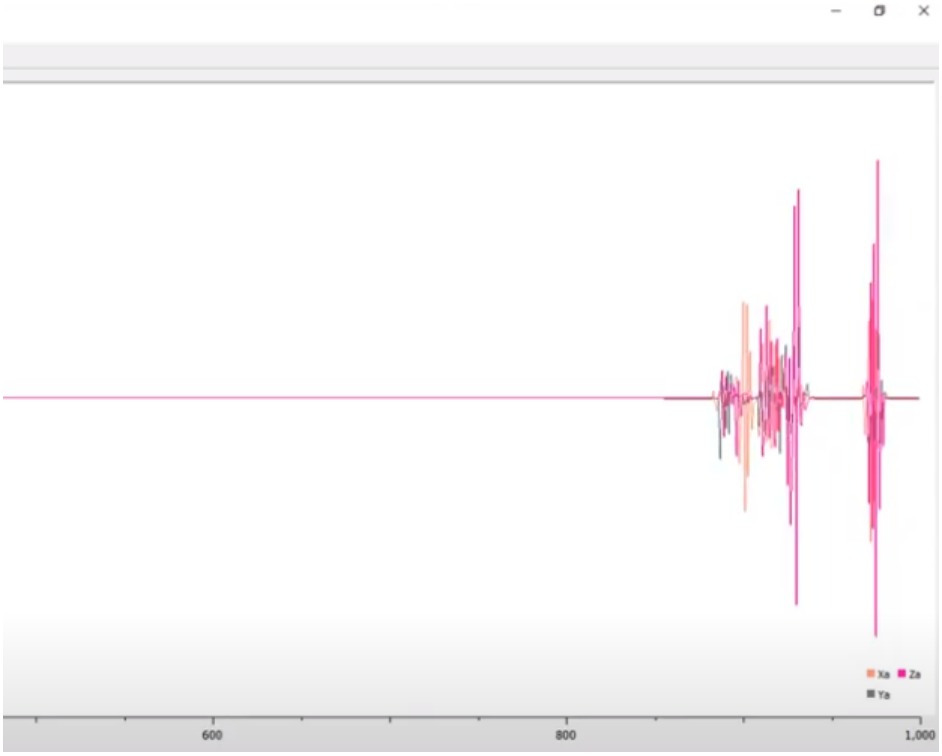
Task 1



Terminal info

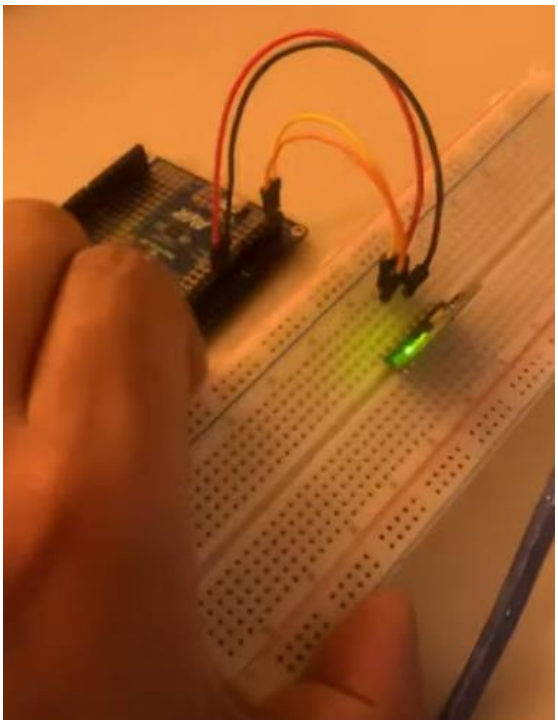


Gyro Serial plotter



Acceleration serial plotter

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)



7. VIDEO LINKS OF EACH DEMO

https://www.youtube.com/watch?v=vW8vL76GRbA

8. GITHUB LINK OF THIS DA

https://github.com/cho-minsung/assignment6

Student Academic Misconduct Policy

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

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

Minsung Cho