CPE301 – SPRING 2023

Design Assignment 7

Student Name: David Lenzin
Student #: 2001654470
Student Email: lenzin@unlv.nevada.edu
Primary Github address: https://github.com/dlenzin15/submissions
Directory: submissions/DA7

Submit the following for all Labs:

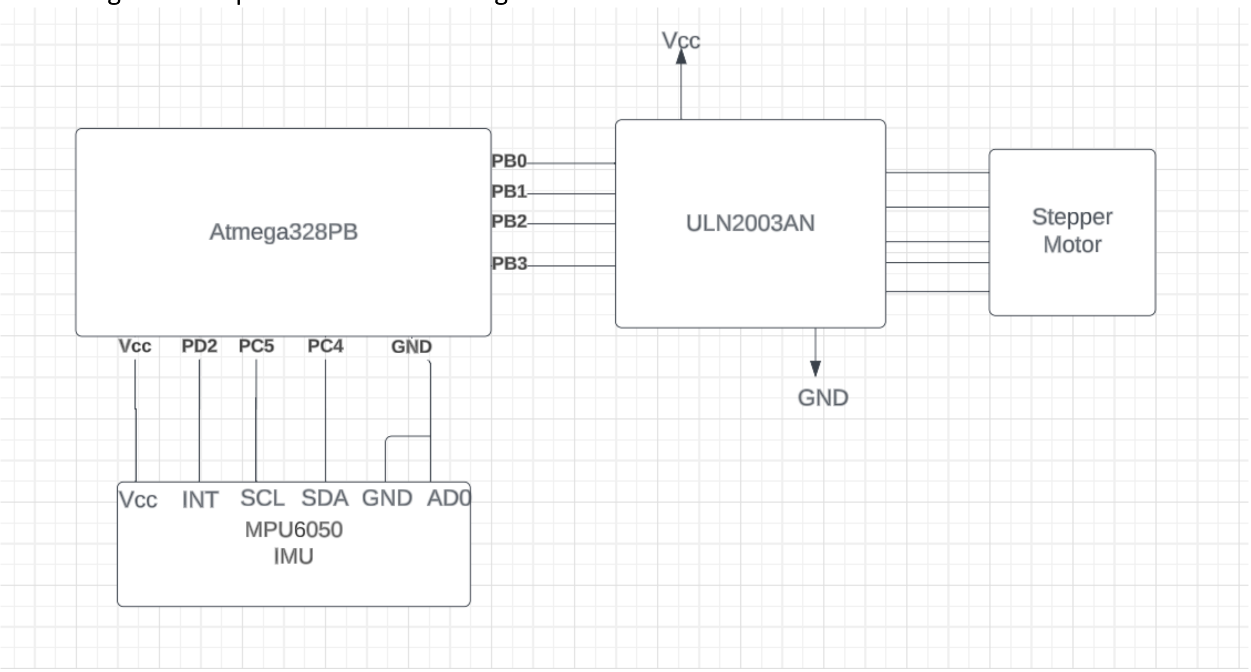
1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder 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:

- Atmega328PB Xplained Mini with Multi-Functional Shield
- MPU6050 Inertial Measurement Unit
- 28BYJ-48 Unipolar Stepper Motor
- ULN2003AN Integrated Circuit
- Breadboard

Block diagram with pins used in the Atmega328PB



2. Header and Secondary Codes Used in Tasks 1-3

UART.h

```
/*
 * USART_RS232_H_file.h
 * http://www.electronicwings.com
 *
 */


#ifndef USART_RS232_H_FILE_H_                          /* Define library H file if not defined */
#define USART_RS232_H_FILE_H_

#define F_CPU 16000000UL                                      /* Define CPU clock Frequency e.g. here its 8MHz */
#include <avr/io.h>                                           /* Include AVR std. library file */
#define BAUD_PRESCALE (((F_CPU / (BAUDRATE * 16UL))) - 1)     /* Define prescale value */

void USART_Init(unsigned long);                        /* USART initialize function */
char USART_RxChar();                                   /* Data receiving function */
void USART_TxChar(char);                               /* Data transmitting function */
void USART_SendString(char*);                          /* Send string of USART data function */
```

```
#endif                                                                     /* USART_RS232_H_FILE_H_ */
```

UART.c

```
/*
 * USART_RS232_C_file.c
 * http://www.electronicwings.com
 *
 */

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

void USART_Init(unsigned long BAUDRATE)                     /* USART initialize function */
{
	UCSR0B |= (1 << RXEN0) | (1 << TXEN0);              /* Enable USART transmitter and receiver */
	UCSR0C |= (1 << UCSZ00) | (1 << UCSZ01);  /* 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++;
	}
}
```

I2C_master.h

```
/*
 * I2C_Master_H_file.h
 * http://www.electronicwings.com
 *
 */


#ifndef I2C_MASTER_H_FILE_H_                                        /* Define library H file if not defined */
#define I2C_MASTER_H_FILE_H_

#define F_CPU 16000000UL                                                   /* Define CPU clock Frequency e.g. here its 
8MHz */
#include <avr/io.h>                                                        /* Include AVR std. library file */
```

```
#include <util/delay.h>                                              /* Include delay header file */
#include <math.h>                                                    /* Include math function */
#define SCL_CLK 100000L                                              /* Define SCL clock frequency */
#define BITRATE(TWSR)       ((F_CPU/SCL_CLK)-16)/(2*pow(4,(TWSR&((1<<TWPS0)|(1<<TWPS1))))) /* Define bit rate */

void I2C_Init();                                                     /* I2C initialize function */
uint8_t  I2C_Start(char);                                     /* I2C start function */
uint8_t  I2C_Repeated_Start(char);                     /* I2C repeated start function */
void I2C_Stop();                                                     /* I2C stop function */
void I2C_Start_Wait(char);                                    /* I2C start wait function */
uint8_t  I2C_Write(char);                                     /* I2C write function */
char I2C_Read_Ack();                                          /* I2C read ack function */
char I2C_Read_Nack();                                                /* I2C read nack function */


#endif
```

I2C_master.c

```
/*
 * i2c_master.c
 *
 * Created: 5/2/2023 9:10:33 PM
 *  Author: david
 */

#include "i2c_master.h"                                                     /* Include I2C header file */

void I2C_Init()                                                      /* I2C initialize function */
{
       TWBR0 = BITRATE(TWSR0 = 0x00);                                /* Get bit rate register value by formula */
}


uint8_t I2C_Start(char slave_write_address)                          /* I2C start function */
{
       uint8_t status;                                               /* Declare variable */
       TWCR0 = (1<<TWSTA)|(1<<TWEN)|(1<<TWINT); /* Enable TWI, generate start condition and clear interrupt flag
*/
       while (!(TWCR0 & (1<<TWINT)));           /* Wait until TWI finish its current job (start condition) */
       status = TWSR0 & 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 */
       TWDR0 = slave_write_address;      /* If yes then write SLA+W in TWI data register */
       TWCR0 = (1<<TWEN)|(1<<TWINT);     /* Enable TWI and clear interrupt flag */
       while (!(TWCR0 & (1<<TWINT)));   /* Wait until TWI finish its current job (Write operation) */
       status = TWSR0 & 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;                                               /* Declare variable */
       TWCR0 = (1<<TWSTA)|(1<<TWEN)|(1<<TWINT)/* Enable TWI, generate start condition and clear interrupt flag */
       while (!(TWCR0 & (1<<TWINT)));           /* Wait until TWI finish its current job (start condition) */
       status = TWSR0 & 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 */
       TWDR0 = slave_read_address;   /* If yes then write SLA+R in TWI data register */
       TWCR0 = (1<<TWEN)|(1<<TWINT); /* Enable TWI and clear interrupt flag */
       while (!(TWCR0 & (1<<TWINT)));  /* Wait until TWI finish its current job (Write operation) */
       status = TWSR0 & 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 */
{
       TWCR0=(1<<TWSTO)|(1<<TWINT)|(1<<TWEN);/* Enable TWI, generate stop condition and clear interrupt flag */
       while(TWCR0 & (1<<TWSTO));  /* Wait until stop condition execution */
}

void I2C_Start_Wait(char slave_write_address)    /* I2C start wait function */
{
       uint8_t status;             /* Declare variable */
       while (1)
       {
              TWCR0 = (1<<TWSTA)|(1<<TWEN)|(1<<TWINT);   /* Enable TWI, generate start condition and clear
interrupt flag */
              while (!(TWCR0 & (1<<TWINT))); /* Wait until TWI finish its current job (start condition) */
              status = TWSR0 & 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 */
              TWDR0 = slave_write_address;    /* If yes then write SLA+W in TWI data register */
              TWCR0 = (1<<TWEN)|(1<<TWINT);   /* Enable TWI and clear interrupt flag */
              while (!(TWCR0 & (1<<TWINT)));     /* Wait until TWI finish its current job (Write operation) */
              status = TWSR0 & 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;                    /* Declare variable */
       TWDR0 = data;                      /* Copy data in TWI data register */
       TWCR0 = (1<<TWEN)|(1<<TWINT);      /* Enable TWI and clear interrupt flag */
       while (!(TWCR0 & (1<<TWINT)));    /* Wait until TWI finish its current job (Write operation) */
       status = TWSR0 & 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 */
{
        TWCR0=(1<<TWEN)|(1<<TWINT)|(1<<TWEA);       /* Enable TWI, generation of ack and clear interrupt flag */
        while (!(TWCR0 & (1<<TWINT)));       /* Wait until TWI finish its current job (read operation) */
        return TWDR0;                        /* Return received data */
}

char I2C_Read_Nack()                  /* I2C read nack function */
{
        TWCR0=(1<<TWEN)|(1<<TWINT); /* Enable TWI and clear interrupt flag */
        while (!(TWCR0 & (1<<TWINT)));      /* Wait until TWI finish its current job (read operation) */
        return TWDR0;                        /* Return received data */
}
```

MPU6050_def.h

```
/*
 * MPU6050_def.h
 *
 * Created: 5/2/2023 4:31:48 PM
 *  Author: david
 */

#ifndef MPU6050_RES_DEFINE_H_
#define MPU6050_RES_DEFINE_H_

#include <avr/io.h>

#define XG_OFFS_TC 0x00
#define YG_OFFS_TC 0x01
#define ZG_OFFS_TC 0x02
#define X_FINE_GAIN 0x03
#define Y_FINE_GAIN 0x04
#define Z_FINE_GAIN 0x05
#define XA_OFFS_H 0x06
#define XA_OFFS_L_TC 0x07
#define YA_OFFS_H 0x08
#define YA_OFFS_L_TC 0x09
#define ZA_OFFS_H 0x0A
#define ZA_OFFS_L_TC 0x0B
#define XG_OFFS_USRH 0x13
#define XG_OFFS_USRL 0x14
#define YG_OFFS_USRH 0x15
#define YG_OFFS_USRL 0x16
#define ZG_OFFS_USRH 0x17
#define ZG_OFFS_USRL 0x18
#define SMPLRT_DIV 0x19
#define CONFIG 0x1A
#define GYRO_CONFIG 0x1B
#define ACCEL_CONFIG 0x1C
#define FF_THR 0x1D
#define FF_DUR 0x1E
#define MOT_THR 0x1F
#define MOT_DUR 0x20
#define ZRMOT_THR 0x21
#define ZRMOT_DUR 0x22
#define FIFO_EN 0x23
#define I2C_MST_CTRL 0x24
```

```
#define I2C_SLV0_ADDR 0x25
#define I2C_SLV0_REG 0x26
#define I2C_SLV0_CTRL 0x27
#define I2C_SLV1_ADDR 0x28
#define I2C_SLV1_REG 0x29
#define I2C_SLV1_CTRL 0x2A
#define I2C_SLV2_ADDR 0x2B
#define I2C_SLV2_REG 0x2C
#define I2C_SLV2_CTRL 0x2D
#define I2C_SLV3_ADDR 0x2E
#define I2C_SLV3_REG 0x2F
#define I2C_SLV3_CTRL 0x30
#define I2C_SLV4_ADDR 0x31
#define I2C_SLV4_REG 0x32
#define I2C_SLV4_DO 0x33
#define I2C_SLV4_CTRL 0x34
#define I2C_SLV4_DI 0x35
#define I2C_MST_STATUS 0x36
#define INT_PIN_CFG 0x37
#define INT_ENABLE 0x38
#define DMP_INT_STATUS 0x39
#define INT_STATUS 0x3A
#define ACCEL_XOUT_H 0x3B
#define ACCEL_XOUT_L 0x3C
#define ACCEL_YOUT_H 0x3D
#define ACCEL_YOUT_L 0x3E
#define ACCEL_ZOUT_H 0x3F
#define ACCEL_ZOUT_L 0x40
#define TEMP_OUT_H 0x41
#define TEMP_OUT_L 0x42
#define GYRO_XOUT_H 0x43
#define GYRO_XOUT_L 0x44
#define GYRO_YOUT_H 0x45
#define GYRO_YOUT_L 0x46
#define GYRO_ZOUT_H 0x47
#define GYRO_ZOUT_L 0x48
#define EXT_SENS_DATA_00 0x49
#define EXT_SENS_DATA_01 0x4A
#define EXT_SENS_DATA_02 0x4B
#define EXT_SENS_DATA_03 0x4C
#define EXT_SENS_DATA_04 0x4D
#define EXT_SENS_DATA_05 0x4E
#define EXT_SENS_DATA_06 0x4F
#define EXT_SENS_DATA_07 0x50
#define EXT_SENS_DATA_08 0x51
#define EXT_SENS_DATA_09 0x52
#define EXT_SENS_DATA_10 0x53
#define EXT_SENS_DATA_11 0x54
#define EXT_SENS_DATA_12 0x55
#define EXT_SENS_DATA_13 0x56
#define EXT_SENS_DATA_14 0x57
#define EXT_SENS_DATA_15 0x58
#define EXT_SENS_DATA_16 0x59
#define EXT_SENS_DATA_17 0x5A
#define EXT_SENS_DATA_18 0x5B
#define EXT_SENS_DATA_19 0x5C
#define EXT_SENS_DATA_20 0x5D
#define EXT_SENS_DATA_21 0x5E
```

```
#define EXT_SENS_DATA_22 0x5F
#define EXT_SENS_DATA_23 0x60
#define MOT_DETECT_STATUS 0x61
#define I2C_SLV0_DO 0x63
#define I2C_SLV1_DO 0x64
#define I2C_SLV2_DO 0x65
#define I2C_SLV3_DO 0x66
#define I2C_MST_DELAY_CTRL 0x67
#define SIGNAL_PATH_RESET 0x68
#define MOT_DETECT_CTRL 0x69
#define USER_CTRL 0x6A
#define PWR_MGMT_1 0x6B
#define PWR_MGMT_2 0x6C
#define BANK_SEL 0x6D
#define MEM_START_ADDR 0x6E
#define MEM_R_W 0x6F
#define DMP_CFG_1 0x70
#define DMP_CFG_2 0x71
#define FIFO_COUNTH 0x72
#define FIFO_COUNTL 0x73
#define FIFO_R_W 0x74
#define WHO_AM_I 0x75



#endif /* MPU6050_RES_DEFINE_H_ */
```

3. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1

Insert initial code here

```
/*
 * DA7_Task1.c
 *
 * Created: 5/1/2023 8:57:44 PM
 * Author : David Lenzin
 */

#define F_CPU 16000000UL                                    /* Define CPU clock Frequency e.g. here its 16MHz */
#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 library 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 */

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,t;
       float Xg=0,Yg=0,Zg=0;
       I2C_Init();                                                         /* Initialize I2C */
       MPU6050_Init();                                                     /* Initialize MPU6050 */
       USART_Init(9600);                                              /* Initialize USART with 9600 baud rate */

       while(1)
       {
              Read_RawValue();

              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, ",float_);
		USART_SendString(buffer);

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

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

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

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

		dtostrf( Zg, 3, 2, float_ );
		sprintf(buffer,"%s\r\n",float_);
		USART_SendString(buffer);
	}
}
```

4. DEVELOPED MODIFIED CODE OF TASK 2

Insert only the modified sections here

```
/*
 * DA7_Task2.c
 *
 * Created: 5/3/2023 5:27:27 PM
 * Author : david lenzin
 */

#define F_CPU 16000000UL                                /* Define CPU clock Frequency e.g. here its 16MHz */
#define ACCELEROMETER_SENSITIVITY 16384.0
#define GYROSCOPE_SENSITIVITY 16.4
#define dt 0.01             // 10 ms sample rate!

#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 library 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 */

float Acc_x,Acc_y,Acc_z,Temperature,Gyro_x,Gyro_y,Gyro_z;
float pitch, roll, yaw;

void ComplementaryFilter()
{
       float pitchAcc, rollAcc;

       // Integrate the gyroscope data -> int(angularSpeed) = angle
       pitch += (Gyro_x / GYROSCOPE_SENSITIVITY) * dt; // Angle around the X-axis, (float)gyrData[0]
       roll -= (Gyro_y / GYROSCOPE_SENSITIVITY) * dt;    // Angle around the Y-axis, (float)gyrData[1]
       yaw += (Gyro_y / GYROSCOPE_SENSITIVITY) * dt;

       // Compensate for drift with accelerometer data if !bullshit
       // Sensitivity = -2 to 2 G at 16Bit -> 2G = 32768 && 0.5G = 8192
       int forceMagnitudeApprox = abs(Acc_x) + abs(Acc_y) + abs(Acc_z);
       if (forceMagnitudeApprox > 8192 && forceMagnitudeApprox < 32768)
       {
              // Turning around the X axis results in a vector on the Y-axis
              pitchAcc = atan2f(Acc_y, Acc_z) * 180 / M_PI;
              pitch = pitch * 0.98 + pitchAcc * 0.02;

              // Turning around the Y axis results in a vector on the X-axis
              rollAcc = atan2f(Acc_x, Acc_z) * 180 / M_PI;
              roll = roll * 0.98 + rollAcc * 0.02;
       }
}

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];
       I2C_Init();                     /* Initialize I2C */
       MPU6050_Init();                 /* Initialize MPU6050 */
       USART_Init(9600);               /* Initialize USART with 9600 baud rate */

       while(1)
       {
              Read_RawValue();
              ComplementaryFilter();

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

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

              dtostrf(roll, 3, 2, float_);
              sprintf(buffer, "%s\r\n", float_);
              USART_SendString(buffer);
       }
}
```

5. DEVELOPED MODIFIED CODE OF TASK 3

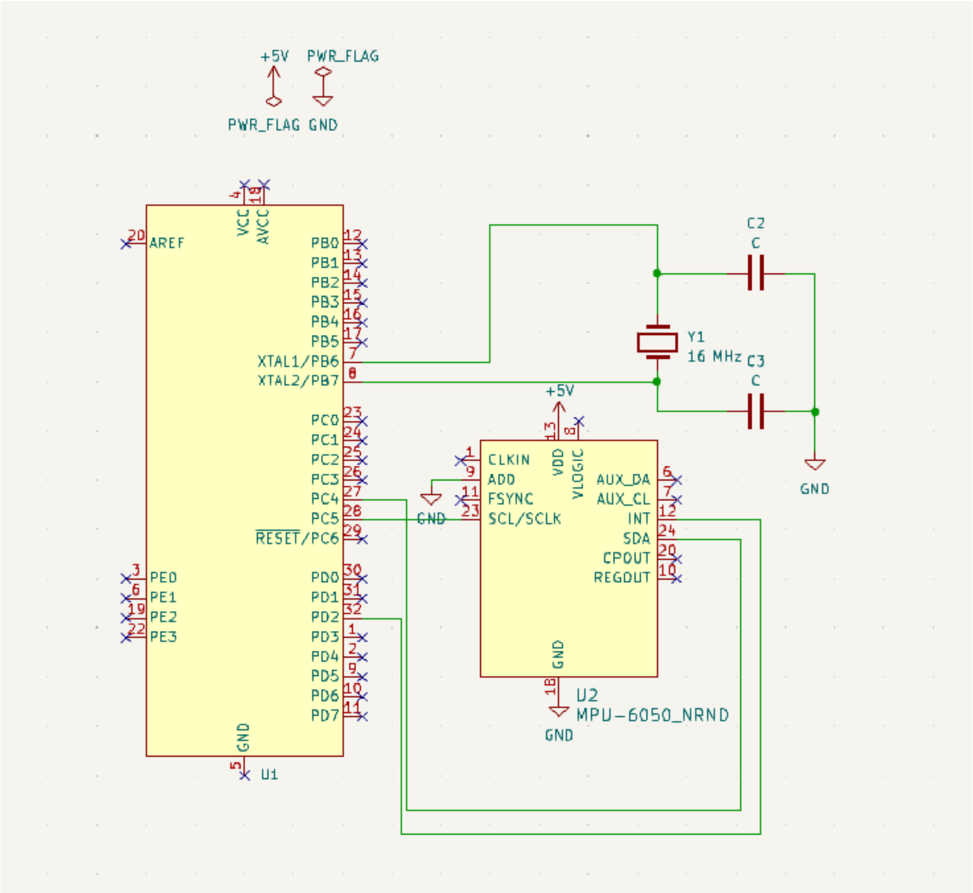
Insert only the modified sections here

```
/*
 * DA7_Task3.c
 *
 * Created: 5/5/2023 2:17:28 PM
 * Author : david lenzin
 */

#define F_CPU 16000000UL                               /* Define CPU clock Frequency e.g. here its 16MHz */
#define ACCELEROMETER_SENSITIVITY 16384.0
#define GYROSCOPE_SENSITIVITY 16.4
#define dt 0.01             // 10 ms sample rate!

#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 library file */
#include "MPU6050_def.h"                               /* Include MPU6050 register define file */
#include "i2c_master.h"                                       /* Include I2C Master header file */

float Acc_x,Acc_y,Acc_z,Temperature,Gyro_x,Gyro_y,Gyro_z;
float pitch, roll, yaw;

void ComplementaryFilter()
{
       float pitchAcc, rollAcc;

       // Integrate the gyroscope data -> int(angularSpeed) = angle
       pitch += (Gyro_x / GYROSCOPE_SENSITIVITY) * dt; // Angle around the X-axis, (float)gyrData[0]
       roll -= (Gyro_y / GYROSCOPE_SENSITIVITY) * dt;    // Angle around the Y-axis, (float)gyrData[1]
       yaw += (Gyro_y / GYROSCOPE_SENSITIVITY) * dt;

       // Compensate for drift with accelerometer data if !bullshit
       // Sensitivity = -2 to 2 G at 16Bit -> 2G = 32768 && 0.5G = 8192
       int forceMagnitudeApprox = abs(Acc_x) + abs(Acc_y) + abs(Acc_z);
       if (forceMagnitudeApprox > 8192 && forceMagnitudeApprox < 32768)
       {
              // Turning around the X axis results in a vector on the Y-axis
              pitchAcc = atan2f(Acc_y, Acc_z) * 180 / M_PI;
              pitch = pitch * 0.98 + pitchAcc * 0.02;

              // Turning around the Y axis results in a vector on the X-axis
              rollAcc = atan2f(Acc_x, Acc_z) * 180 / M_PI;
              roll = roll * 0.98 + rollAcc * 0.02;
       }
}

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 */
```

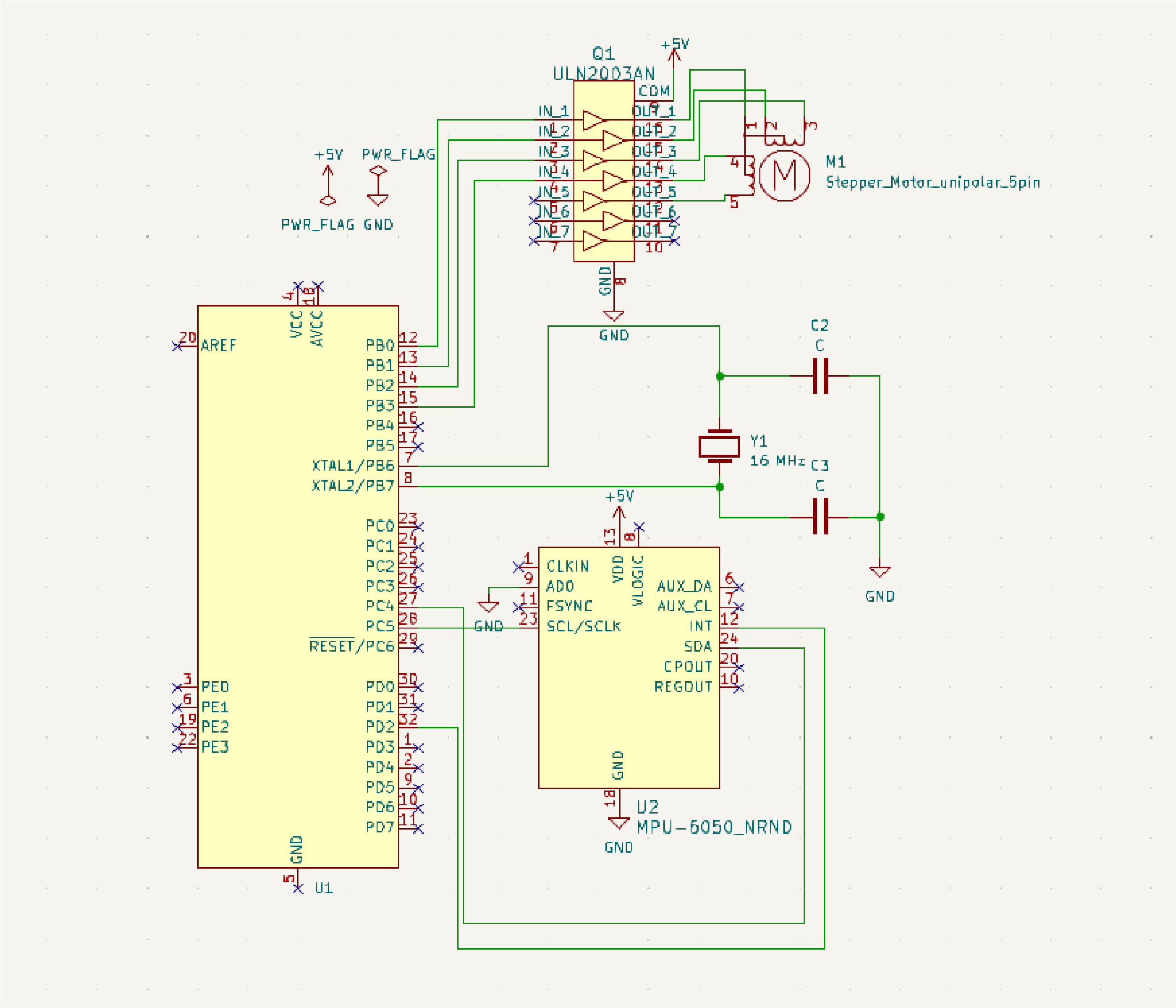
```c
       I2C_Write(0x00);                                        /* Fs = 8KHz */
       I2C_Stop();

       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(void)
{
       I2C_Init();                                      /* Initialize I2C */
       MPU6050_Init();                                  /* Initialize MPU6050 */

       int period;
       DDRB = 0x0F;                              /* Make PORTD lower pins as output */
       period = 5000;                                   /* Set period in between two steps of Stepper Motor */

    while (1)
    {
              Read_RawValue();
              ComplementaryFilter();

              if (roll > 20.0)
              {
                     for(int i=0;i<100;i++)    /* Rotate Stepper Motor counter-clockwise with Full step sequence; 
Full step angle 7.5 */
                     {
                            PORTB = 0x09;
                            _delay_us(period);
                            PORTB = 0x0C;
                            _delay_us(period);
                            PORTB = 0x06;
                            _delay_us(period);
                            PORTB = 0x03;
                            _delay_us(period);
                            Read_RawValue();
                            ComplementaryFilter();
                     }
                     PORTB = 0x09;              /* last one step to acquire initial position */
                     _delay_us(period);
              }

              if (roll < -20.0)
              {
                     for(int i=0;i<100;i++)    /* Rotate Stepper Motor clockwise with Full step sequence; Full
step angle 7.5 */
                     {
                            PORTB = 0x09;
                            _delay_us(period);
                            PORTB = 0x03;
                            _delay_us(period);
                            PORTB = 0x06;
                            _delay_us(period);
                            PORTB = 0x0C;
                            _delay_us(period);
                            Read_RawValue();
                            ComplementaryFilter();
                     }
                     PORTB = 0x09;              /* last one step to acquire initial position */
                     _delay_us(period);
              }
              _delay_ms(100);      // Small delay to read values
    }
}
```

6. SCHEMATICS

Task 1 and 2:

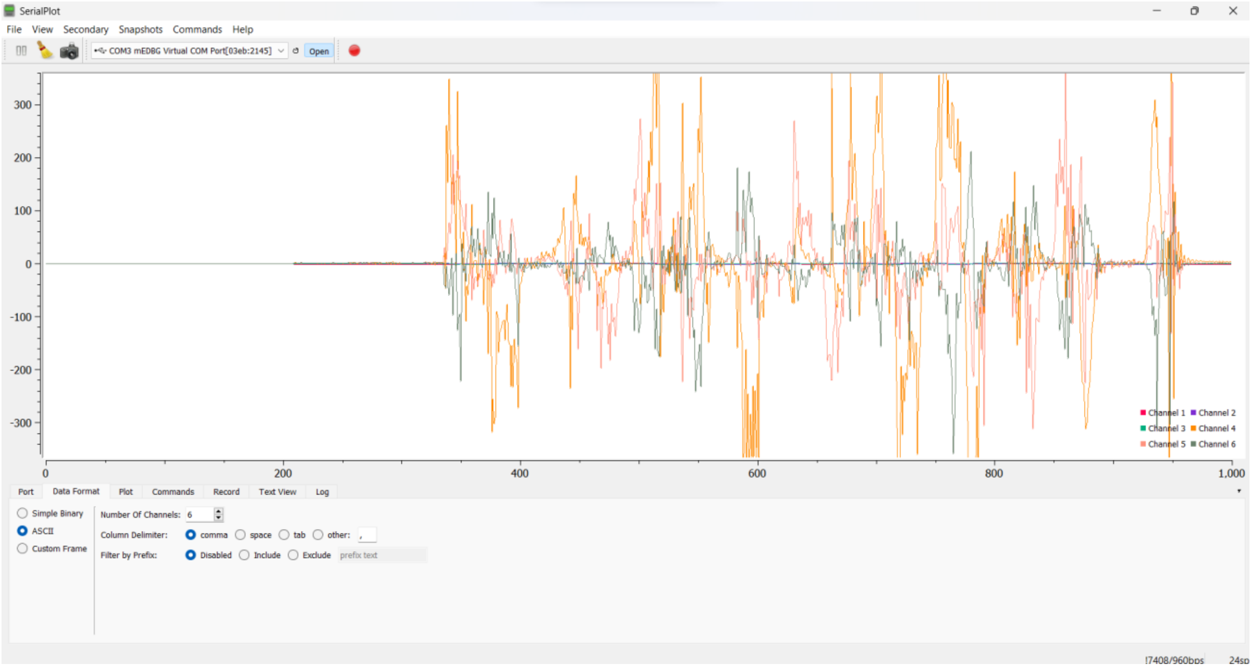

+5V
PWR_FLAG
PWR_FLAG
GND
VCC
AVCC
4
18
20
AREF
PB0
PB1
PB2
PB3
PB4
PB5
XTAL1/PB6
XTAL2/PB7
12
13
14
15
16
17
7
8
PC0
PC1
PC2
PC3
PC4
PC5
RESET/PC6
23
24
25
26
27
28
29
PE0
PE1
PE2
PE3
3
6
19
22
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
30
31
32
1
2
9
10
11
GND
5
U1
C2
C
Y1
16 MHz
C3
C
GND
+5V
13
8
VDD
VLOGIC
1
CLKIN
9
ADD
11
FSYNC
23
SCL/SCLK
GND
AUX_DA
AUX_CL
INT
SDA
CPOUT
REGOUT
6
7
12
24
20
10
GND
18
U2
MPU-6050_NRND
GND

Task 3:



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

Task 1 Serial Plot:


SerialPlot
File View Secondary Snapshots Commands Help
COM3 mEDBG Virtual COM Port[03eb:2145]
Open
300
200
100
0
-100
-200
-300
0
200
400
600
800
1,000
Channel 1
Channel 2
Channel 3
Channel 4
Channel 5
Channel 6
Port Data Format Plot Commands Record Text View Log
!7408/960bps
24sp

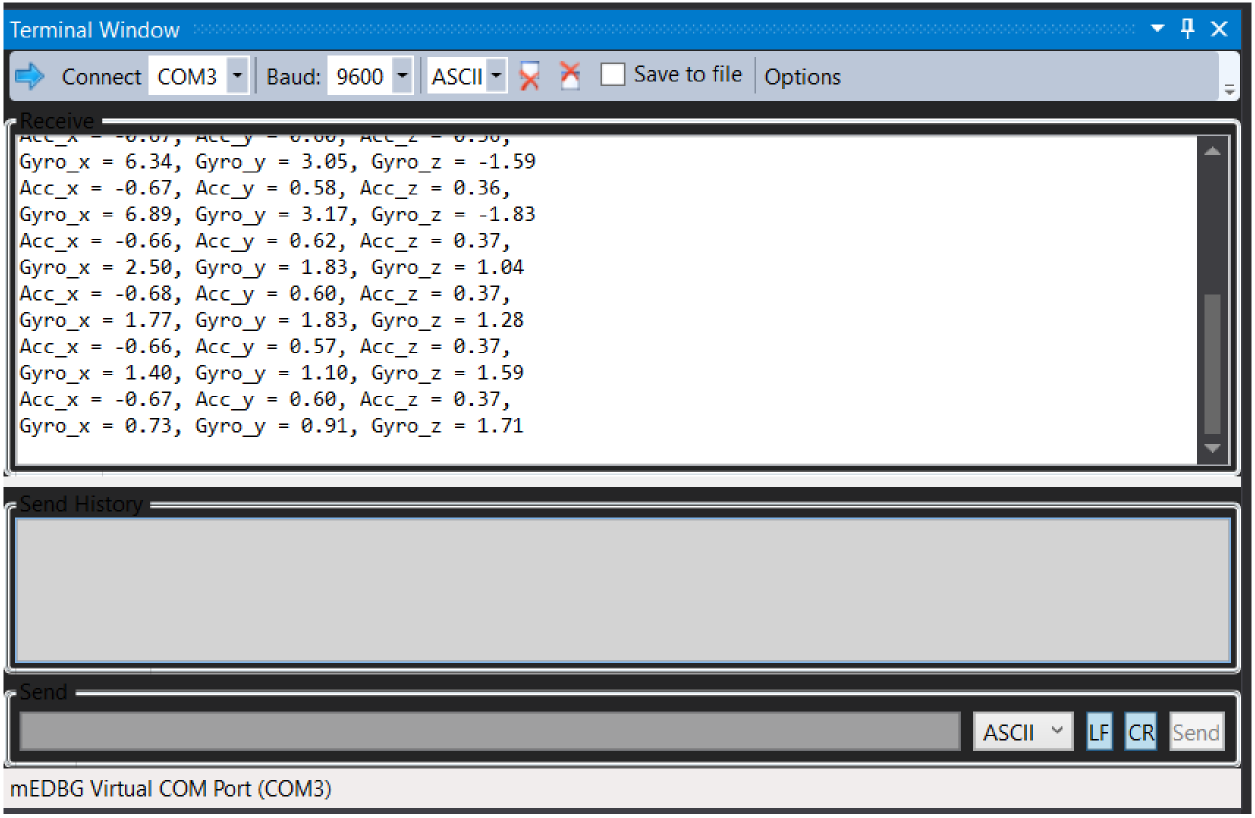
○ Simple Binary
● ASCII
○ Custom Frame

Number Of Channels: 6

Column Delimiter: ● comma ○ space ○ tab ○ other: ,

Filter by Prefix: ● Disabled ○ Include ○ Exclude prefix text

Task 1 Terminal Output:



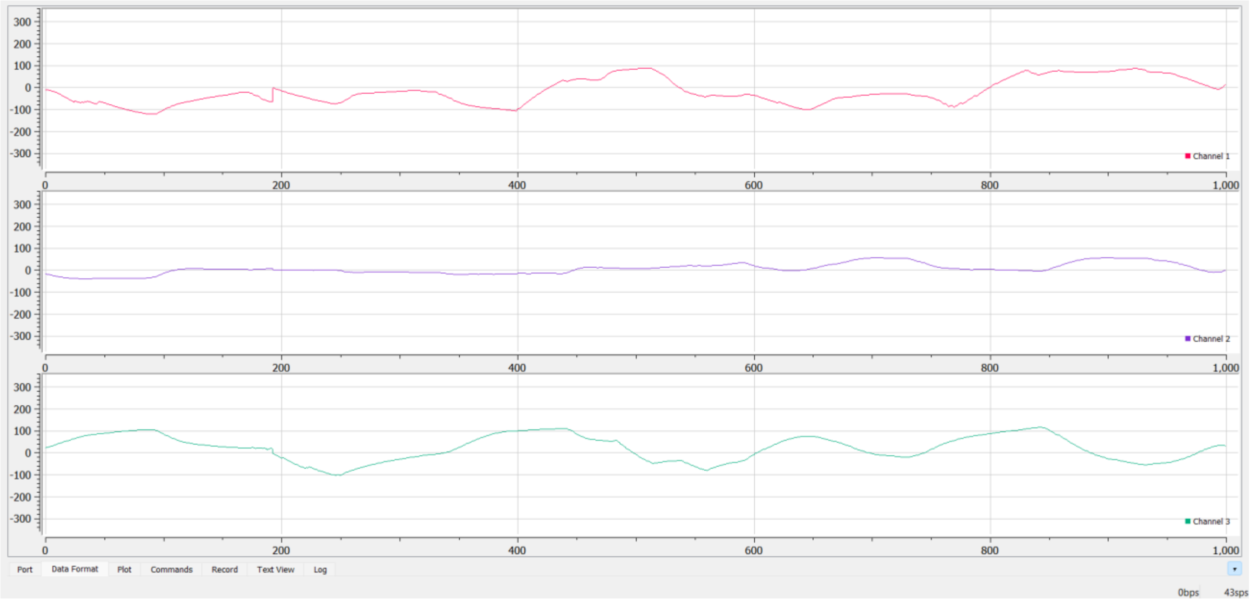
COM3 Baud: 9600 ASCII □ Save to file

```
Acc_x = -0.67, Acc_y = 0.60, Acc_z = 0.36,
Gyro_x = 6.34, Gyro_y = 3.05, Gyro_z = -1.59
Acc_x = -0.67, Acc_y = 0.58, Acc_z = 0.36,
Gyro_x = 6.89, Gyro_y = 3.17, Gyro_z = -1.83
Acc_x = -0.66, Acc_y = 0.62, Acc_z = 0.37,
Gyro_x = 2.50, Gyro_y = 1.83, Gyro_z = 1.04
Acc_x = -0.68, Acc_y = 0.60, Acc_z = 0.37,
Gyro_x = 1.77, Gyro_y = 1.83, Gyro_z = 1.28
Acc_x = -0.66, Acc_y = 0.57, Acc_z = 0.37,
Gyro_x = 1.40, Gyro_y = 1.10, Gyro_z = 1.59
Acc_x = -0.67, Acc_y = 0.60, Acc_z = 0.37,
Gyro_x = 0.73, Gyro_y = 0.91, Gyro_z = 1.71
```

ASCII

mEDBG Virtual COM Port (COM3)

Task 2 Serial Plot:



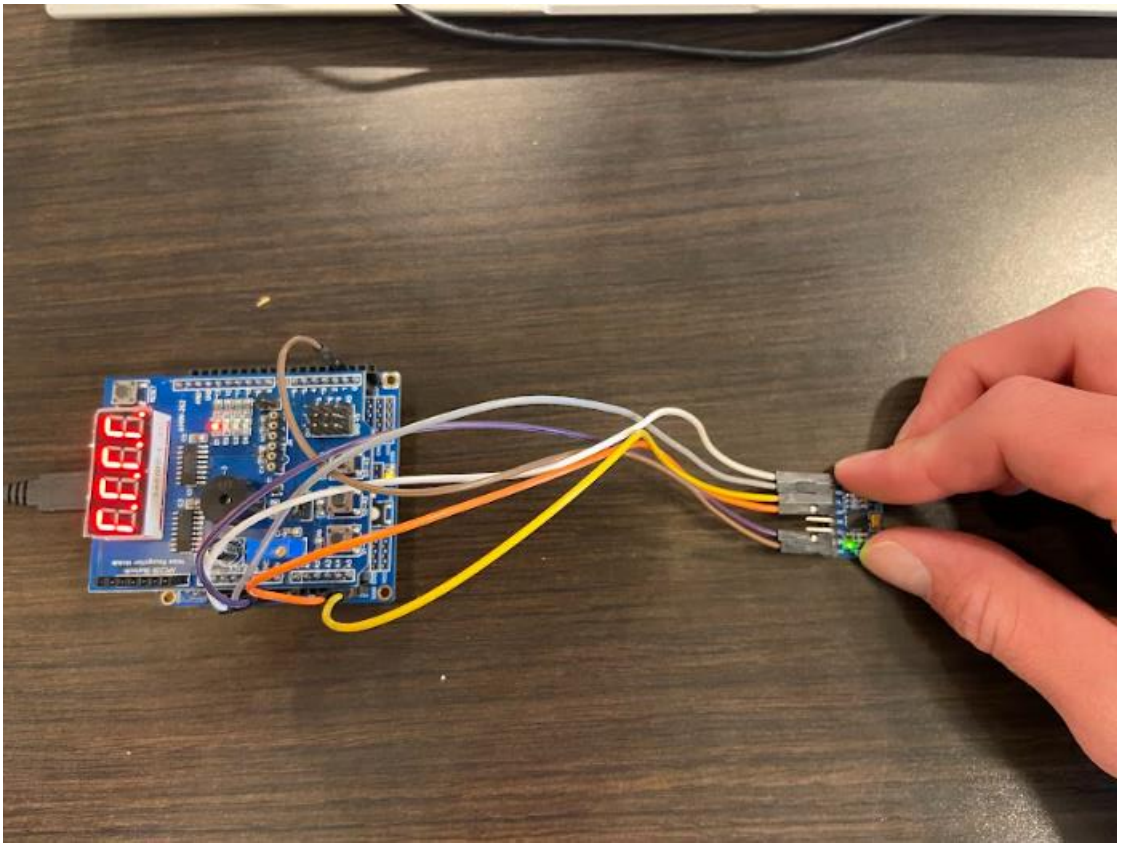
- **Channel 1 = Pitch, Channel 2 = Yaw, Channel 3 = Roll**

Task 3:

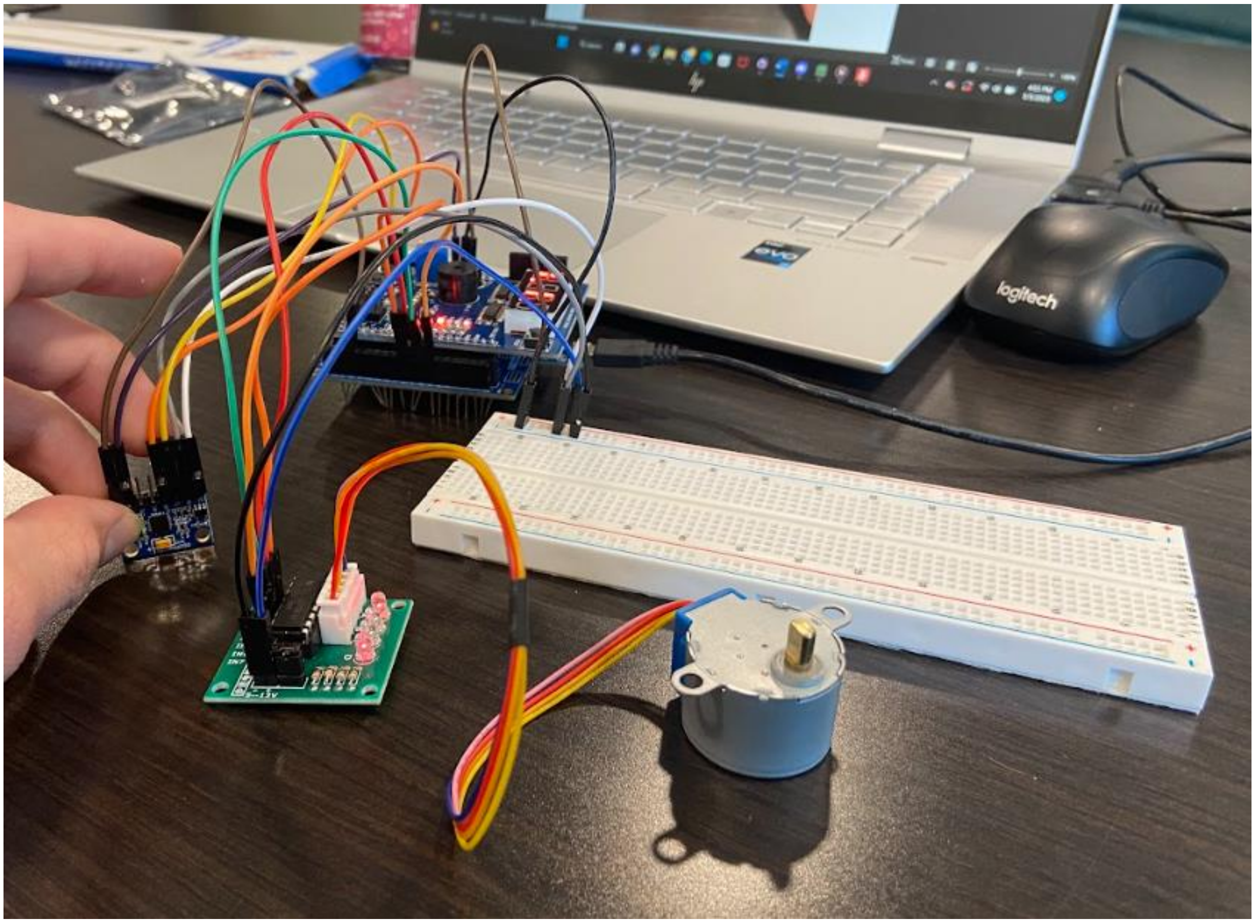
- No Microchip Studio 7 output

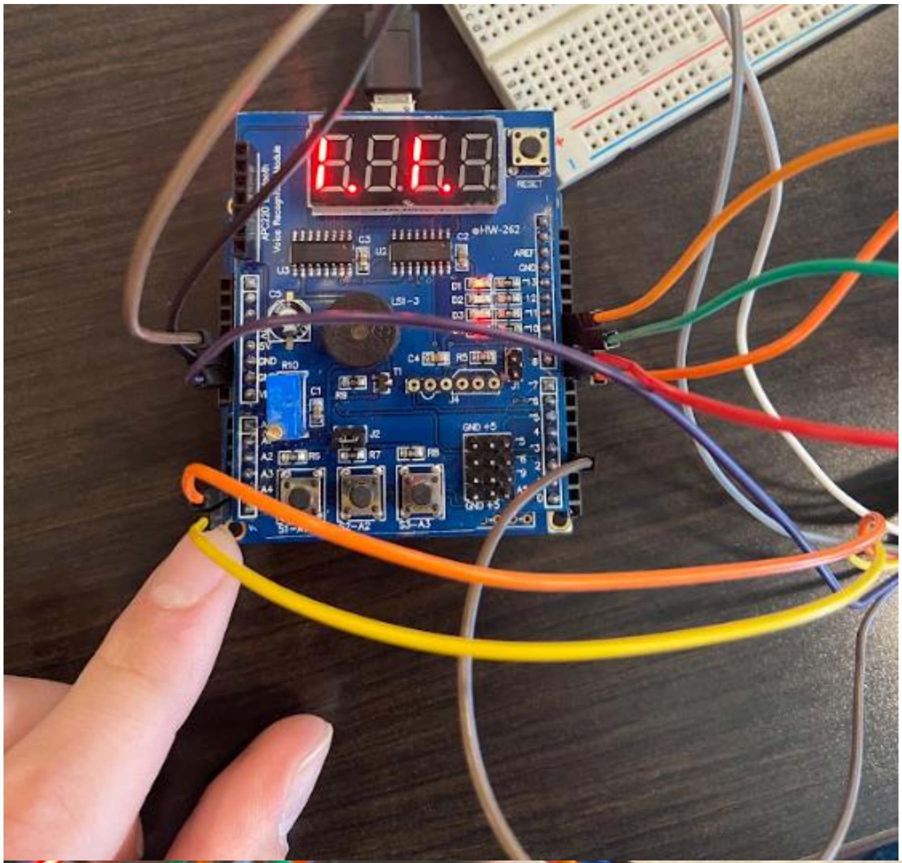
8. SCREENSHOT OF EACH DEMO (BOARD SETUP)

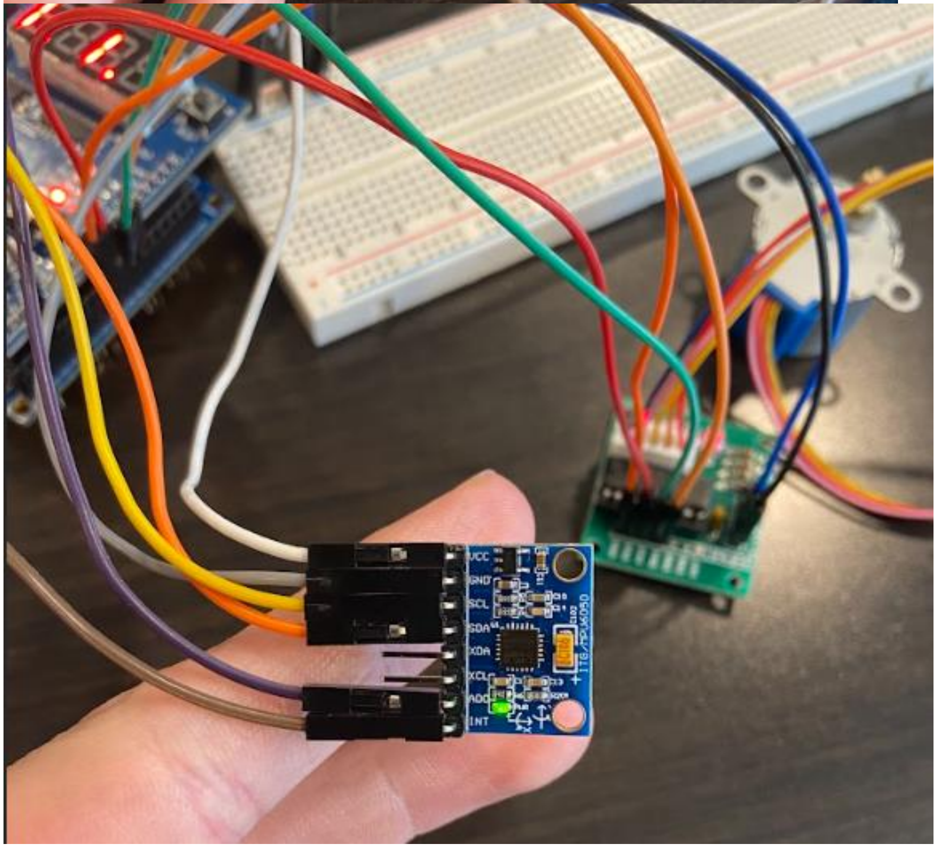
Task 1 and 2:

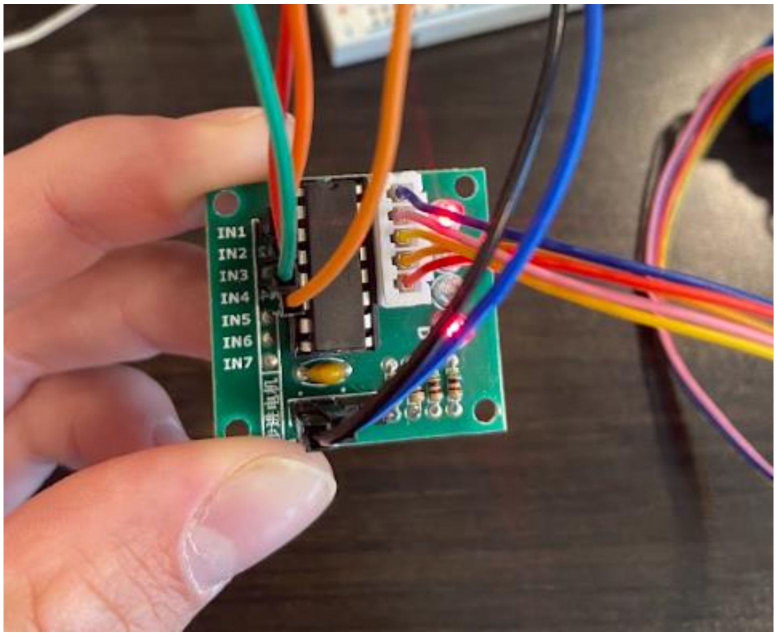


Task 3:




RESET
HW-262
APC220 Bluetooth Voice Recognition Module
AREF
GND
U3
U2
C3
C2
D1
D2
D3
LS1-3
R10
C1
C4
R5
T1
J4
J2
R7
R8
GND +5
S1-A1
S2-A2
S3-A3


VCC
GND
SCL
SDA
XDA
XCL
ADO
INT
ITG/MPU6050



9. VIDEO LINKS OF EACH DEMO

Playlist: https://www.youtube.com/playlist?list=PLlHKEZIJ23uA1LJy5tFlQrYuEsQnqW8g1
Task 1: https://youtu.be/ZfhhCy-dIDY
Task 2: https://youtu.be/RAG6Kjhbfgw
Task 3: https://youtu.be/ECl3ZZVMW_w

10. GITHUB LINK OF THIS DA

https://github.com/dlenzin15/submissions/tree/main/DA7

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

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