Design Assignment 7

Student Name: Samuel McCormick
Student #: 1014303276
Student Email: samuel.mccormick@gmail.com
Primary Github address: https://github.com/brokenboredom/tech-muffin.git
Directory: DesignAssignments/DA7

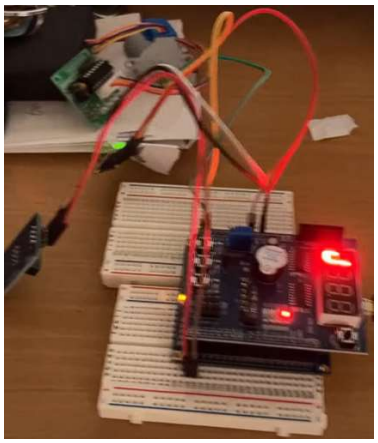
The goal of the assignment is to develop the above code to do the following:

1. *Interface the provided MPU6050 6-DOF IMU Sensor to the ATmega328pb using the I2C interface. Using the earlier developed code for UART, display the accelerometer and gyro data to the UART Terminal and Serial Plotter application.*

For task 1 I simply used the provided example code and connected the UART and MPU6050 to the board. UART is configured the same as it was previously and I2C is connected to GND, 5V, PC5 (SCL) and PC4 (SDA). The UART output strings were altered to be just the data buffer with trailing comma to interface with SerialPlot.

Code of altered UART output for task 1.

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

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

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

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

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

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



SerialPlot of raw sensor values (left). Demo circuit (right).

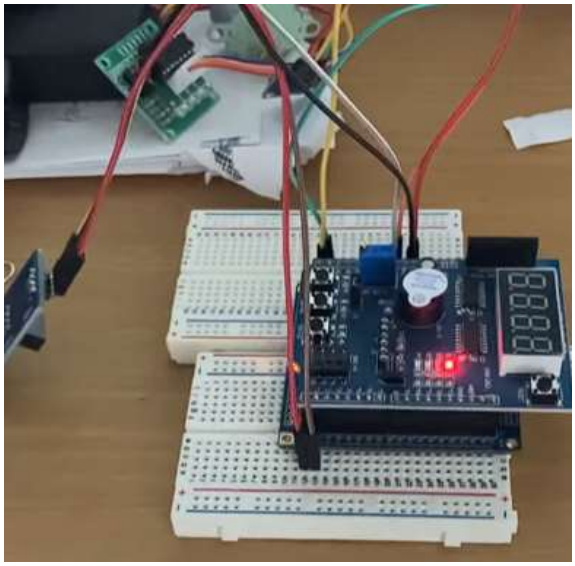
2. *Apply Complementary to the accelerometer and gyro data to determine the roll, pitch, and yaw of the sensor orientation. Plot the above six values as graphs.*

Using the provided pitch, roll, and complementary filter calculations I took the raw sensor data and calculated the Euler angles for our sensor orientation.

Code for complementary filter.

```
void getAngle(int Ax,int Ay,int Az)
{
        double x = Ax;
        double y = Ay;
        double z = Az;
        pitch = atan(x/sqrt((y*y) + (z*z)));        // pitch calculation
        roll = atan(y/sqrt((x*x) + (z*z)));         // roll calculation
        //converting radians into degrees
        pitch = pitch * (180.0/3.14);
        roll = roll * (180.0/3.14) ;
}

void ComplementaryFilter()
{
        float pitchAcc, rollAcc;

        // Integrate the gyroscope data -> int(angularSpeed) = angle
        pitch += ((float)Gyro_x / GYROSCOPE_SENSITIVITY) * dt; // Angle around the X-axis
        roll -= ((float)Gyro_y / GYROSCOPE_SENSITIVITY) * dt; // Angle around the Y-axis
        //yaw += ((float)Gyro_z / 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((float)Acc_y, (float)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((float)Acc_x, (float)Acc_z) * 180 / M_PI;
                roll = roll * 0.98 + rollAcc * 0.02;
        }
}
```



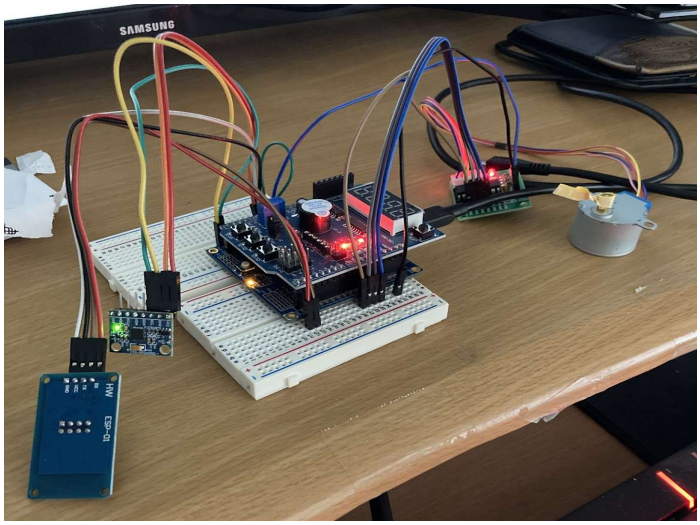
SerialPlot for pitch and roll with filter (left). Demo circuit (right).

3. *Assuming the sensors controls the roll of a platform that is operated by a stepper motor, control the movement of the stepper motor to mimic the motion of th roll value of the sensor.*

Using the roll angle calculated in task 2 I iterated through full steps for every integer angle difference between the stepper motor's last position and the current roll angle by subtracting the difference. A positive/negative delta determined the direction and number of steps.

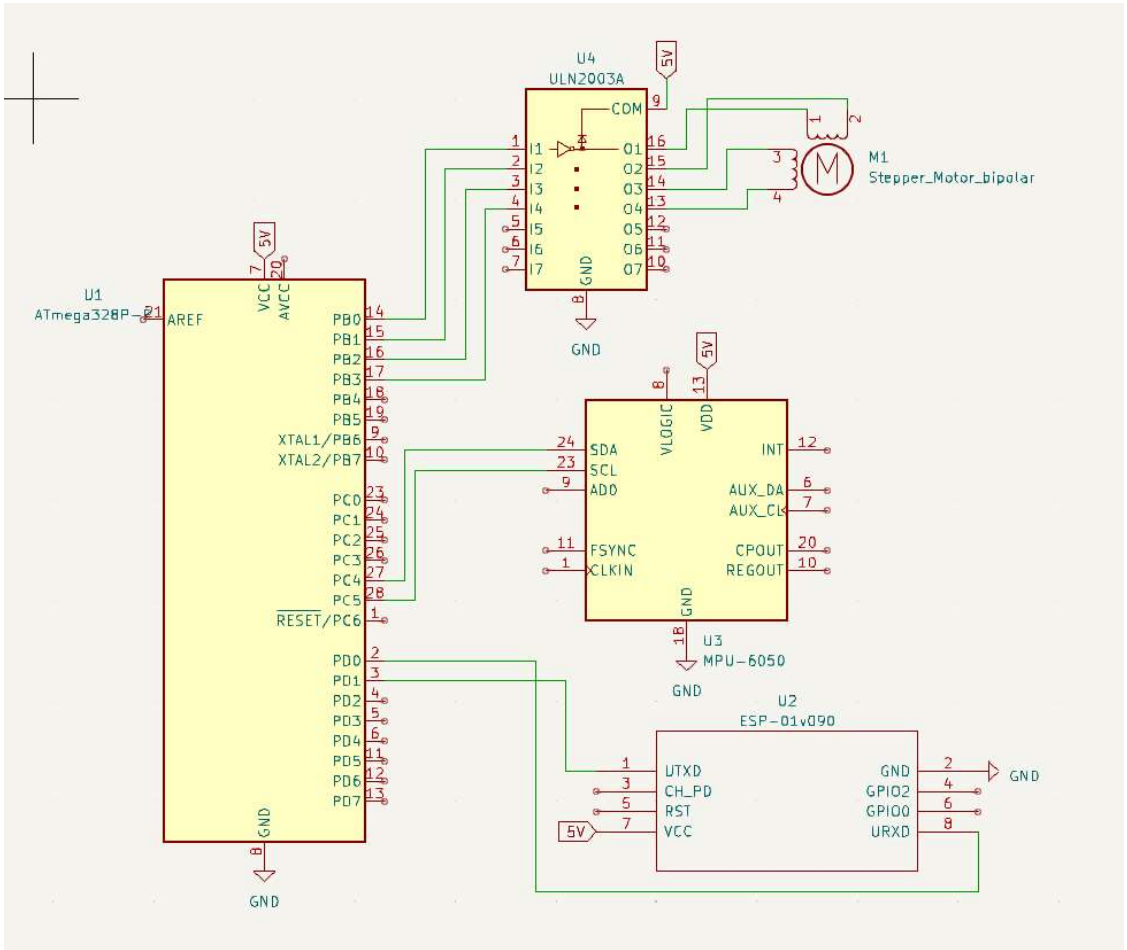
Code for stepper control using roll.

```
void getStepperAngle() {
        int currentAngle = (int)(roll);                 // get current roll angle
        int deltaAngle = lastAngle - currentAngle;      // calc delta angle
        lastAngle -= deltaAngle;                        // update stepper angle
        // Rotate Stepper Motor Clockwise
        if (deltaAngle < 0) {
                for(int i=0;i<(-deltaAngle);i++)
                {
                        PORTB = 0x09;
                        _delay_us(period);
                        PORTB = 0x0C;
                        _delay_us(period);
                        PORTB = 0x06;
                        _delay_us(period);
                        PORTB = 0x03;
                        _delay_us(period);
                }
        }
        // Rotate Stepper Motor Anticlockwise
        else if (deltaAngle > 0) {
                for(int i=0;i<(deltaAngle);i++)
                {
                        PORTB = 0x09;
                        _delay_us(period);
                        PORTB = 0x03;
                        _delay_us(period);
                        PORTB = 0x06;
                        _delay_us(period);
                        PORTB = 0x0C;
                        _delay_us(period);
                }
        }
        // Do nothing at angle 0
        else {}

        PORTB = 0x09;                   /* last one step to acquire initial position */
        _delay_us(period);
        _delay_ms(100);
}
```



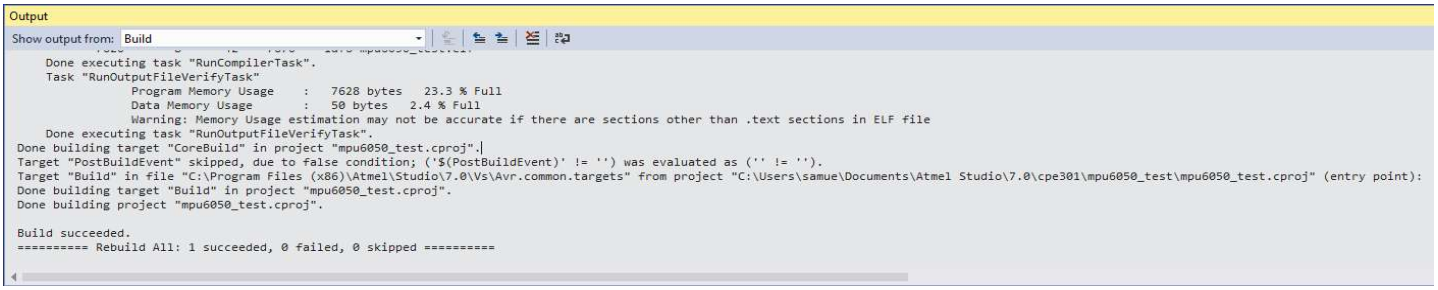
Demo circuit with stepper motor.

Schematics



Full assignment schematic. Task1 and Task2 are the same minus stepper connections. All powered through 5V output from board.

Captures



Successful compile.

Demo Videos

Task1: https://youtube.com/shorts/YFAPsuVs5Sg?feature=share
Task2: https://youtube.com/shorts/bGC1ZPDFac0?feature=share
Task3: https://youtube.com/shorts/WI1-vASqdUA?feature=share