University of Colombo

Department of Physics

PH-3032 - EMBEDDED SYSTEMS LABORATORY

A low-cost self-stabilizing smart spoon for people with Parkinson's disease

Name: L.G.Kalani Keshila

Index: s16041

Admission: 2021s18888

Date of submission: 30.10.2024

1 Abstract

The increasing use of technology to assist individuals with functional impairments has made significant progress in recent years. This project focuses on developing a low-cost, self-stabilizing spoon designed to help people with Parkinson's disease during the eating process. Human limbs lost due to accidents can be replaced with bionic limbs and with help from smartphones, blind people can by audio be informed what kind of object appears in front of them. These are a few examples where technology has eased everyday life for people with impaired functionality.

Utilizing an ATmega328P microcontroller and an MPU6050 sensor, the spoon is designed to counteract hand tremors by maintaining the spoon's position using servo motors. The MPU6050 sensor, which combines a gyroscope and accelerometer, detects movements in the roll, pitch, and yaw axes. The data is processed by the microcontroller, which then adjusts the position of the spoon through two orthogonally placed servomotors to maintain stability. 4.2° roll and 2.8° pitch yaw showing promising results with minimal errors: The smart spoon provides an affordable and effective solution for individuals with impaired motor skills, improving their ability to perform everyday tasks like eating. This project demonstrates the potential for technology to improve the quality of life for those with functional challenges.

Table of Contents

Table of Figures

2 Introduction

Parkinson's disease (PD) is known as older person's disease because it is most often found in individuals over 60. It is found that only 4 percent of all cases are diagnosed before age 50. Since Alzheimer's, Parkinson's is the second most age-related nerve that degenerates. (Parkinson's Disease in the Elderly) Parkinson's Disease is a neurodegenerative disease that occurs in the aging brain. Although Parkinson's disease occurs with movements, it affects people with Parkinson's disease while doing daily activities such as eating, drinking, or writing. Parkinson's disease usually happens when the hands are stretched as the muscles are opposed to gravity.

People who are affected by this disability can execute certain tasks that occur in everyday life with the help of compensating devices. With the help of the stabilizing spoon, an individual with these problems can eat independently and does not need any help from another. The use of stabilizing mechanisms appears in many different fields. There are a few examples that can be found in aircraft, industrial robotics, and video stabilization. Many of these mechanisms have a preference for a body's constant and unchangeable location, irrespective of the motion involved. This motion is very important in supporting people with tremors.

This project aims to produce a stabilizing spoon that will compensate for unintended motions, such as tremors. With a low budget, the goal is to make a highly efficient prototype that consists mainly of a microcontroller and servo motors. There are several different technological assisting spoons on the market today, but the products are unfortunately quite expensive. With the results from this project, further research could be able to continue developing cheaper and better-stabilizing spoons.

3 Theory

3.1 Stabilizing spoons

If the handle of the spoon is tilting an angle with α degrees, see Figure 1, actuators in the spoon's construction will compensate with the same angle α and put the spoon bowl in its initial horizontal position.

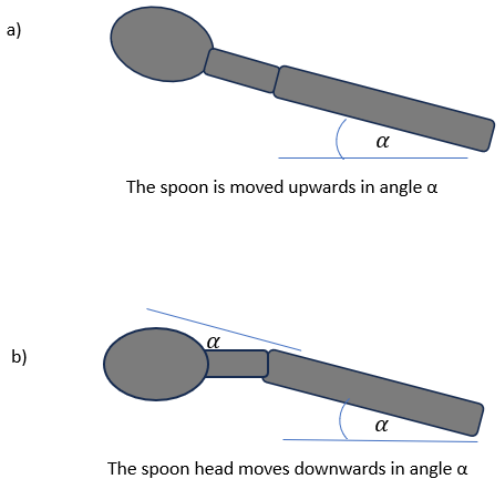

Figure 1: Spoon's movement illustrated

3.2 Stabilization

Once the gyroscope and accelerometer values have been converted to angular readings, they have to be assessed to determine the movement received by the user towards the rear end of the spoon. As the accelerometer works in a cylindrical coordinate system, it is important to convert our values to the cartesian system before applying them to the stabilization process

$$Change\ in\ X = \ tan^{-1}\left(\frac{accY}{accZ)}\right) \times 180 \times \frac{\pi}{2}$$

$$Change\ in\ Y = \ tan^{-1}\left(\frac{accZ}{accX)}\right) \times 180 \times \frac{\pi}{2}$$

Where accX, accY, and accZ are raw angular values obtained after conversion, from the accelerometer. Once we obtain our final angular readings, we can set our servo motor 'X' and 'Y' accordingly with the respective value + 90° (initial). This will calibrate exactly to its base position at any given instant. We can do this by the given formula.

$$Servo\ 'X' = 90 + Change\ in\ angle\ on\ the\ X\ axis$$

$$Servo\ 'Y' = 90 + Change\ in\ angle\ on\ the\ Y\ axis$$

3.3 PWM- Pulse Width Modulation

To control electronics through a microcontroller, PWM is commonly used. PWM signals are square waves, switching on and off with a certain voltage. The unit that is being controlled by the PWM signals behaves accordingly after each duty cycle. A duty cycle is the percentage of time when the voltage is "on" which means that if a motor is being controlled, the duty cycle will determine indirectly its velocity.

3.4 Complementary filter

Complementary filter An MPU6050 presents occasionally values that are incoherent and vary unreasonably to one another. To get proper values, clear values, a filter is needed. A complementary filter consists of one low pass filter and one high pass filter. The implementation of this setup of a complementary filter in software is shown in the equation,

$$Filtered\ signal = x(1 - \alpha) + y\alpha$$

where the fraction, α is generally 0.98 and the parameters x and y represent values from the accelerometer respectively the gyroscope from the IMU(MPU6050).

4 Methodology

4.1 Overall Block Diagram

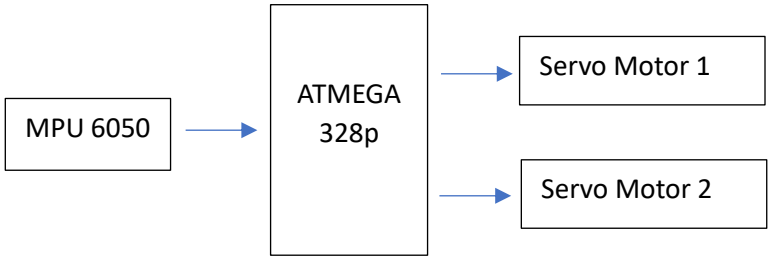

Figure 2: Block diagram

The system consists of an ATmega328p microcontroller, 2 servo motors, and an MPU6050. MPU 6050 has an accelerometer and gyroscope inside the sensor to detect the movement of the Parkinson's patient's arm and follow orientation rather than gravity. ATmega328p is a microcontroller for data processing and decisions, It sends the data to the 2 servo motors. As shown in the figure above, the working principle for this project started with input from the MPU6050, which reads data when tilted or shaken. Then, the data is sent and processed by the ATmega328p which acts as the microcontroller and drives the servomotor as the output to stabilize the auxiliary device.

4.2 Electronic Connections of whole System

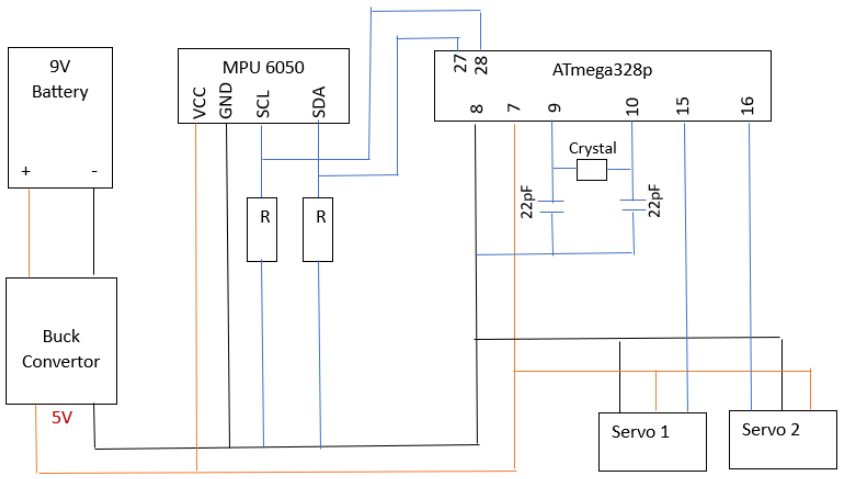

Figure 3: Electronic Connections of whole System

As shown in the figure above, the wires for the prototype should be connected and should look like this. A 9V battery with a buck converter is used to supply 5V power to an ATmega328p microcontroller and 2 servomotors. Here, the buck converter reduces the potential of 9V from the battery to 5V and gives it to the circuit. If a potential of 9V is applied directly to the circuit, the microcontroller may burn. In the circuit diagram above, the wires to be connected to the ground are shown in black and the wires to be connected to the power supply are shown in red. For the MPU6050, there are 4 wires to connect to the ATmega328p. As mentioned earlier, the red wires should be connected to the power supply and the black wires should be connected to the ground. For SCL and SDA, the wire should be connected to PINs 27 and 27 of the microcontroller. For servomotors, three colored wires connect the servomotor to the microcontroller. The yellow wire represents PWM. They should be connected to pins 15, and 16 of the microcontroller, the red wire should be connected to the power supply and finally, the brown wire should be connected to the ground. All the pins must be defined in the AVR programming code according to the wire connected to the pins of the microcontroller for the prototype to work very well.

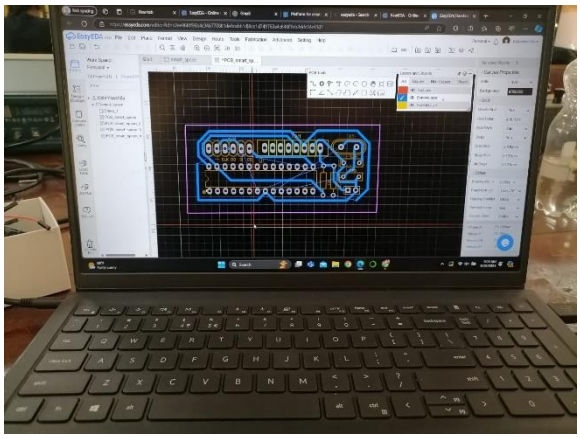

Figure 4: PCB design using EasyEDA web site.

Now the MPU6050 sensor can detect the roll, pitch, and yaw angle of Parkinson's hand tremors. Then, a microcontroller takes readings from the angle via I2C. From a microcontroller, a PWM signal is given to the servo motor through the motor driver so that the spoon can be stabilized in its original position. The servo motor includes feedback so that a microcontroller can provide information to the servo motor according to Parkinson's disease. This helps to set the angle for spoon preparation. Here, the development of the code was divided into three parts. The first part is to declare each parameter that will be used later in the code. This is also important when

wiring the prototype because the pin declaration already declared in the code must follow the code set to execute it. The second part is the basic setup, establishing communication between the microcontroller, servomotor, and IMU (MPU 6050). The third part is to take the output value from the IMU and pass it through a complementary filter to command the servomotor to rotate accordingly. Here, before preparing the PCB and completing the prototype, the MPU 6050 was placed on a flat surface and the offset values of the x and y axes were calibrated. The calibration was done by Arduino. The performance of the stabilizing scoop can be analyzed.

4.3 MPU 6050 calibration by Arduino

A simple circuit was set up to test the functionality of the MPU6050 and understand how the Arduino can talk to the MPU6050. The MPU6050 programming code needs to calibrate the position so the normal fit does not match the MPU6050 fixed position already set on the breadboard. Therefore, keeping the MPU6050 in a fixed position, at that time about 1000 values related to the x and y axis were read separately by the gyro sensor, and the two average values were obtained on the serial printer. (Using the given setup and codes in Arduino, the raw values can be seen on the serial monitor as the filters are automatically applied to the gyroscope and MPU6050 to get the output.) The two values received by the MPU 6050 for the X, and Y axis when the MPU 6050 is stationary are GyX in the AVR code. and assigned as offset values of GyY. The MPU 6050 was repeatedly calibrated by the Arduino in many cases while running the AVR code for the spoon application.

4.4 The building up of the prototype

It would have been more appropriate to use 3D printing to build the prototype, but instead, plastic parts in the shape of a tube, easily found at home, were used. They were cut into parts as needed and separated. Such parts are also easily available in the market. The main objective of the project is to build the prototype using cost-effective, low-cost, and sustainable materials. After designing and finalizing the PCB, according to the final design of the prototype, the glue gun was used to attach the two servo motors as shown in the image, and assemble. A switch was used to "on" the device and it was set to it externally.

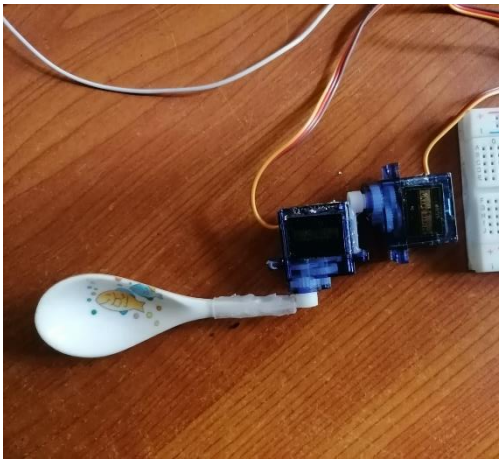

Figure 5:One servo motor is attached perpendicular to the other servo motor

Figure 6: Final design of the prototype

5 Results And Analysis

5.1 Servo Motor Testing

This testing is carried out to find out whether the servomotor can be moved according to the turning angle as an input. After the test is performed, a table is made to identify whether the servomotor functions properly or not.

Table 1: THE RESULT OF ROLL ANGLE TESTING

No	Angle (Degree)	Data of sensor reading (degree)	Error (Degree)
1	0	1	1
2	30	26	4
3	45	41	4
4	60	54	6
5	90	84	6
Average			4.2

Based on the result above, it is shown that the servomotor can be moved according to the turning angle. Based on the result, it can be seen that the servomotor can do the angle movement in line with the set point, with the average error for the pitch is 4.2 degrees. This value is still in the normal range and can be used. Another main finding is that the servomotor returns to its original position when it is being rested after being turned to a certain angle which might cause a higher value in the average error for the pitch.

Table 2: THE RESULT OF PITCH ANGLE TESTING

No	Angle (Degree)	Data of sensor reading (degree)	Error (Degree)
1	0	2	2
2	30	28	2
3	45	43	2
4	60	57	3
5	90	85	5
Average			2.8

Based on the result above, it was shown that the servomotor can be moved according to turning angle in line with the set point, with the average error for the roll being 2.8 degrees. This value is still in the normal range and can be used. The servomotor for the pitch is more stable compared to the roll servomotor.

6 Discussion And Conclusion

To develop the self-stabilizing smart spoon, the prototype is designed and the program is successfully programmed using the atmega328p microcontroller. The prototype, also known as a stabilizing spoon or "smart" spoon, is quite heavy compared to the existing device called Liftware, which may be difficult for people with Parkinson's disease.

The servomotor does not respond as quickly as it should. The code used for that and the calibration values are also affected. A "smart" spoon project can resist motion but fails to eliminate jitter due to the mass of hardware and the signal processing algorithm used. And some changes are needed if it is to be considered usable for people with high-frequency tremors. Also, if MPU6050 calibration is done after the prototype is finished, the project may be more successful. Also, if a gyro sensor is used for the spoon bowl and another one for the handle to test it after the prototype is finished, the accuracy of the final product can be measured well. However, with more time, changes in hardware, and more research into the "smart" spoon project, it can be further improved and produce a quality product.

7 Appendix

The project's developed code

```
#include <avr/io.h>
#include <util/delay.h>
#include <math.h>

#define MPU_ADDR 0x68  // I2C address of the MPU6050
#define F_CPU 16000000UL // Define CPU frequency for delay functions

// I2C Functions for AVR
void I2C_init(void);
void I2C_start(void);
void I2C_stop(void);
void I2C_write(uint8_t data);
uint8_t I2C_read_ack(void);
uint8_t I2C_read_nack(void);

// MPU6050 Functions
void MPU6050_init(void);
void MPU6050_read(int16_t* AcX, int16_t* AcY, int16_t* AcZ, int16_t* GyX, int16_t* GyY);

void servo_init();
void servo_write(uint8_t angle);
uint16_t map(int value, int fromLow, int fromHigh, int toLow, int toHigh);

// Define I2C functions (simplified for ATmega328P)
void I2C_init() {
   TWSR = 0; // Set prescaler to 1
   TWBR = 72; // Set bit rate register (for 100kHz clock)
```

```
        TWCR = (1 << TWEN); // Enable I2C
}
void I2C_start() {
   TWCR = (1<<TWSTA) | (1<<TWEN) | (1<<TWINT);  // Send START condition
   while (!(TWCR & (1<<TWINT)));  // Wait for transmission to complete
}
void I2C_stop() {
   TWCR = (1<<TWSTO) | (1<<TWINT) | (1<<TWEN);  // Send STOP condition
   while (TWCR & (1<<TWSTO));
}
void I2C_write(uint8_t data) {
   TWDR = data; // Load data into TWDR register
   TWCR = (1 << TWINT) | (1 << TWEN); // Start transmission
   while (!(TWCR & (1 << TWINT))); // Wait for transmission to complete
}
uint8_t I2C_read_ack() {
   TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWEA); // Start receiving with ACK
   while (!(TWCR & (1 << TWINT))); // Wait for reception to complete
   return TWDR; // Return received data
}
uint8_t I2C_read_nack() {
   TWCR = (1 << TWINT) | (1 << TWEN); // Start receiving without ACK
   while (!(TWCR & (1 << TWINT))); // Wait for reception to complete
   return TWDR; // Return received data
}
void MPU6050_init() {
   I2C_start(); // Start I2C communication
   I2C_write(MPU6050_ADDR << 1); // Write to the MPU6050 (0x68)
   I2C_write(0x6B); // Power management register
   I2C_write(0x00); // Wake up the MPU6050 (clear sleep bit)
   I2C_stop(); // Stop I2C communication
```

```
}
void MPU6050_read(int16_t* AcX, int16_t* AcY, int16_t* AcZ, int16_t* GyX, int16_t* GyY) {
   I2C_start();
   I2C_write(MPU_ADDR << 1); // Write to MPU6050
   I2C_write(0x3B); // Start with ACCEL_XOUT_H
   I2C_start();  // Restart and switch to reading
   I2C_write((MPU_ADDR << 1) | 1); // Read from MPU6050

   // Read accelerometer data
   *AcX = (I2C_read_ack() << 8) | I2C_read_ack();
   *AcY = (I2C_read_ack() << 8) | I2C_read_ack();
   *AcZ = (I2C_read_ack() << 8) | I2C_read_ack();

   // Read gyroscope data
   *GyX = (I2C_read_ack() << 8) | I2C_read_ack();
   *GyY = (I2C_read_nack() << 8) | I2C_read_nack();
   I2C_stop();
}

// Servo control functions (basic PWM implementation)
void servo_init() {
   // Set PWM pin as output
   DDRB |= (1<<DDB1); // For example, OC1A on pin 15 (PB1)
   TCCR1A = (1<<COM1A1) | (1<<WGM11); // Fast PWM mode, non-inverted
   TCCR1B = (1<<WGM12) |(1<<WGM13) | (1<<CS11);   // Fast PWM mode, prescaler 8
   ICR1 = 19999; // Set the top value for 50Hz frequency
}

void servo_write(uint8_t angle) {
   uint16_t pulse = map(angle, 0, 180, 1000, 2000); // Map angle to pulse width
   OCR1A = pulse; // Set the pulse width
```

```
}
uint16_t map(int value, int fromLow, int fromHigh, int toLow, int toHigh) {
    return (value - fromLow) * (toHigh - toLow) / (fromHigh - fromLow) + toLow;

}
int main() {
    int16_t AcX, AcY, AcZ, GyX, GyY;
    float angleEstimateX = 0, angleEstimateY = 0;
    float alpha = 0.98, dt = 0.1;
    int gyroXOffset = -239, gyroYOffset = 230;

    I2C_init();
    MPU6050_init();
    servo_init();

            // Enable global interrupts
    //sei();

    while (1) {
        MPU6050_read(&AcX, &AcY, &AcZ, &GyX, &GyY);

        // Apply the gyro offsets
        GyX -= gyroXOffset;
        GyY -= gyroYOffset;

        // Calculate gyro rates
        float gyroRateX = GyX / 131.0;
        float gyroRateY = GyY / 131.0;

        // Calculate angles from accelerometer data
        float accelAngleX = atan2(AcY, AcZ) * 180 / M_PI;
```

```
        float accelAngleY = atan2(AcX, AcZ) * 180 / M_PI;

        // Update the estimated angles using the complementary filter
        angleEstimateX = alpha * (angleEstimateX + gyroRateX * dt) + (1 - alpha) * accelAngleX;
        angleEstimateY = alpha * (angleEstimateY + gyroRateY * dt) + (1 - alpha) * accelAngleY;

                    // Map the angle estimates to servo positions (0-180 degrees)
        int mappedAngleX = (int)fmin(fmax(angleEstimateX + 90, 0), 180); // Map and constrain X-axis
angle
        int mappedAngleY = (int)fmin(fmax(angleEstimateY + 90, 0), 180); // Map and constrain Y-axis
angle

        // Move the servos to the updated angles
        servo_write(mappedAngleX); // Set servo X angle
        servo_write(mappedAngleY); // Set servo Y angle

        _delay_ms(100); // Small delay to allow smooth movement

    }

    return 0;
}
```

8 References

1. Chaturvedi, C., Hingorani, V.V. and Gudipalli, A. (2024) 'An internet of things-enabled self-stabilizing spoon for patients with parkinson's disease', *RAiSE-2023*, 10, p. 150. doi:10.3390/engproc2023059150.

2. Nico J. Diederich, M. (2003) *Parkinson disease with old-age onset*, *Archives of Neurology*. Available at: https://jamanetwork.com/journals/jamaneurology/fullarticle/784012 (Accessed: 26 August 2024).
3. Sharma, A. (2019) 'Adjustable spoon for parkinsons sufferer', *International Journal for Research in Applied Science and Engineering Technology*, 7(5), pp. 1328–1330. doi:10.22214/ijraset.2019.5222.
4. Sharma, A. (2019) 'Adjustable spoon for parkinsons sufferer', *International Journal for Research in Applied Science and Engineering Technology*, 7(5), pp. 1328–1330. doi:10.22214/ijraset.2019.5222.