BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 318 (January 2023)

Control Systems Laboratory

Final Project Report

Section: C2 Group: 05

Self-Balancing Monopod: A Reaction Wheel Inverted Pendulum

Course Instructors:

Dr. Pran Kanai Saha, Professor
Azazul Islam Razon, Part-Time Lecturer

Signature of Instructor: ____

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. Type the student ID and name, and put your signature. ***You will not receive credit for this project experiment unless this statement is signed in the presence of your lab instructor.***

"In signing this statement, We hereby certify that the work on this project is our own and that we have not copied the work of any other students (past or present), and cited all relevant sources while completing this project. We understand that if we fail to honor this agreement, We will each receive a score of ZERO for this project and be subject to failure of this course."

Signature: ________________________
Full Name: Ameer Hamja Ibne Jamal
Student ID: 1906177

Signature: ________________________
Full Name: Fahim Shahriar Anim
Student ID: 1906178

Signature: ________________________
Full Name: Yeasir Arafat Prodhan
Student ID: 1906191

Signature: ________________________
Full Name: U Mong Sain Chak
Student ID: 1906195

Table of Contents

1 Abstract

A self-balancing monopod is a device designed to help individuals maintain stability and balance while using a single-leg support structure, typically for activities such as photography, videography, or even for personal mobility assistance. The basic goal of this project is to build such a user-friendly and most importantly cost-effective self-balancing monopod which works on the basis of a reaction wheel inverted pendulum. Basically, the motor at the top of the monopod spin-up and spin-down, creating a torque on the monopod. That torque is used to bring the monopod back to its most stable position, thereby preventing it from falling. The monopod is also designed in such a way that it can handle slight disturbances (for example: wind). The IMU combining the gyro and accelerometer, uses the current angle and change in angle with respect to time (angular speed) to create torque to the motor using a PID control algorithm

2 Introduction

While being on a hiking trip, to snap a group photo, certainly one either needs a drone, a tripod, or a selfie stick and certain problems may arise:

- Balancing a selfie stick to get the perfect snap might be difficult due to vibration of hand.
- Drones are expensive and complicated.
- Tripods are too bulky to carry

Therefore, a monopod that can balance itself can be a suitable solution having certain pros mentioned below:

❖ Immunity to Disturbances

  ➢ It can balance itself on rugged surfaces and under various disturbances like wind.

- Lightweight and User-friendly
  - Compared to a tripod, our proposed monopod is lighter and easier to setup: what the user needs to do is to turn on the power switch and put the monopod at the vertical balanced position.
- Offers Manual Interventions
  - The user can always manually change the shooting angle of the monopod by holding it at a different angle. The monopod would automatically go back to balancing position once the user releases the monopod.

The technical term for a self-balancing monopod is a reaction wheel inverted pendulum. A reaction wheel inverted pendulum is a dynamic control system that combines elements of a reaction wheel (used in spacecraft and robotics for attitude control) and an inverted pendulum (a classic control system problem). This setup is often used in educational settings, research, and robotics demonstrations to illustrate principles of control theory and demonstrate advanced control techniques.

The core of a reaction wheel inverted pendulum system is the control algorithm. This algorithm calculates the necessary torque to apply to the reaction wheel to maintain the pendulum in an upright position. Various control techniques can be used, including PID (Proportional-Integral-Derivative) control, LQR (Linear Quadratic Regulator) control, or more advanced techniques like nonlinear control or reinforcement learning.

3 Design

3.1 Problem Formulation (PO(b))

3.1.1 Identification of Scope

To identify the scope for a reaction wheel inverted pendulum system, several aspects were considered:

- Developing a mathematical model of the system by understanding various physical properties of the pendulum.
- Selecting an appropriate control strategy (for example PID control, LQR Control) for achieving the control objectives. In our project PID mechanism is implemented.
- Determining the sensors and actuators needed for our system. In this case, sensors to measure the pendulum's angle and angular velocity and actuators to control the reaction wheel's speed were needed.
- Noise and external disturbances may affect the system's performance. In this project control system was designed having an aim to handle these disturbances and maintain stability.
- To improve the system's performance, the system was continuously tuned and optimized by adjusting control gains, implementing anti-windup mechanisms, or using adaptive control techniques.

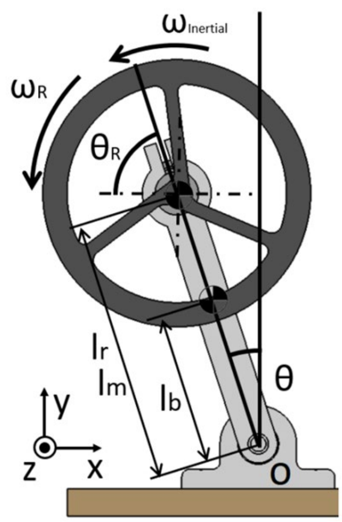
3.1.2 Literature Review

In the paper titled "The Modeling and Control of the Flywheel Inverted Pendulum System" by Ruan et al. (2010), the dynamic model of the flywheel inverted pendulum, as the object to be controlled, has been established. The mathematical model, once established, was certified, and a system performance analysis was conducted. Subsequently, a fuzzy controller based on packet control was designed as the linear model. They achieved this by making a linear approximation of the nonlinear model around the unstable equilibrium point in the upper position.

To achieve global stability, the paper "Linear Control of the Flywheel Inverted Pendulum" by Olivares et al. (2014) presents two options: firstly, by introducing an internal stabilizing controller, and secondly, by replacing the PID controller with an observer-based state feedback control. In most cases within the literature, the PID controller is commonly used. However, in "LQR and MPC Controller Design and Comparison for a Stationary Self-Balancing Bicycle Robot with a Reaction Wheel" by Kanjanawanishkul (2015), three control strategies are proposed: Linear Quadratic Regulator (LQR), Linear Model Predictive Control (LMPC), and Nonlinear Model Predictive Control (NMPC). The problem of the reaction wheel is almost similar to that of the flywheel inverted pendulum, making the ideas transferrable.

3.1.3 Formulation of Problem

Assuming a structure like below:



A. Notation

- M_m : Motor Mass.
- M_R : Reaction Wheel Mass.
- M_b : Pendulum Arm Mass.
- b_b : Pendulum Arm axis viscous friction.
- l_b : Pendulum Arm center of mass distance to axis.
- l_m : Motor center of mass distance to axis.
- l_R : Reaction wheel center of mass distance to axis.
- g : Gravitational acceleration.
- τ_c : Effective Motor Control torque.

Inertia:

- I_{bo} : Pendulum with respect to point o.
- I_{ro} : Reaction wheel with respect to point o.
- I_{mo} : Motor with respect to point o.
- I_R : Reaction wheel with respect to its center of mass.

Angular Speed:

- ω_R : wheel speed with respect to pendulum arm.
- $\omega_{Inertial}$: wheel inertial speed.

It is practically a non-linear system hence has to be linearized.

The state space can be designed as shown below:

$$\dot{X} = AX + Bu(t)$$

$$y = CX + Du(t)$$

with the matrices being:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{k_{mgl}}{I_{so}} & -\frac{b_b}{I_{so}} & \frac{k_t k_e}{RI_{so}} + \frac{b_R}{I_{so}} \\ -\frac{k_{mgl}}{I_{so}} & \frac{b_b}{I_{so}} & -\frac{I_{so}+I_R}{I_{so}I_R}(b_R + \frac{k_e k_t}{R}) \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ -\frac{k_t}{RI_{so}} \\ \frac{I_{so}+I_R}{I_{so}I_R}\frac{k_t}{R} \end{bmatrix}$$

C. *Model Linearization*

The system is linearized around the steady state point:

$$\begin{cases} \theta^* = 0 & \text{(10a)} \\ \omega_R^* = 0 & \text{(10b)} \end{cases}$$

In equation (1), $\sin\theta$ is linearized around θ^* as θ. No deviation variables need to be defined because the linearization point is at $\theta = 0$.

$$\dot{X} = AX + Bu(t)$$

$$y = CX + Du(t)$$

with the matrices being:

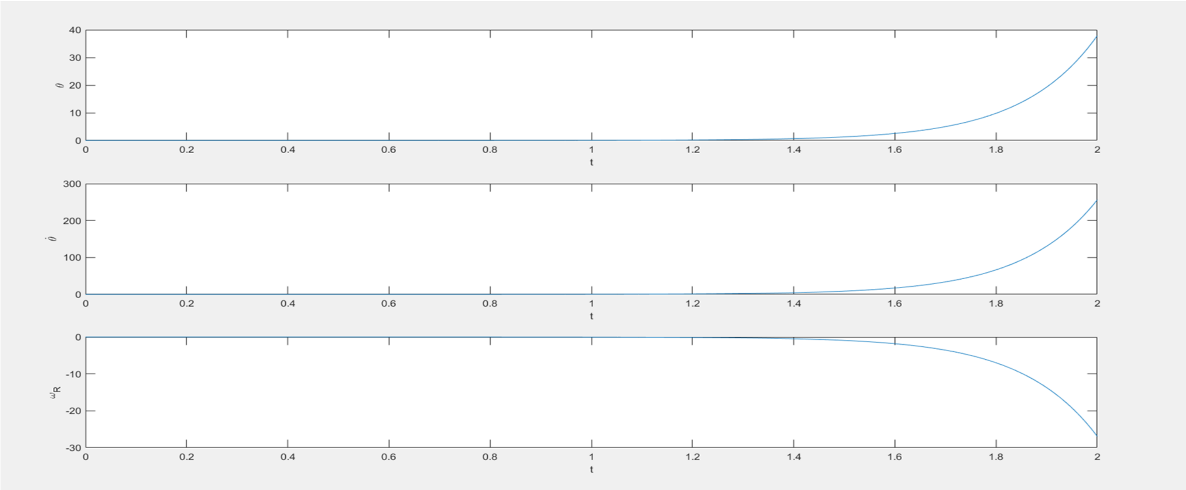
$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{k_{mgl}}{I_{so}} & -\frac{b_b}{I_{so}} & \frac{k_t k_e}{RI_{so}} + \frac{b_R}{I_{so}} \\ -\frac{k_{mgl}}{I_{so}} & \frac{b_b}{I_{so}} & -\frac{I_{so}+I_R}{I_{so}I_R}(b_R + \frac{k_e k_t}{R}) \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ -\frac{k_t}{RI_{so}} \\ \frac{I_{so}+I_R}{I_{so}I_R}\frac{k_t}{R} \end{bmatrix}$$

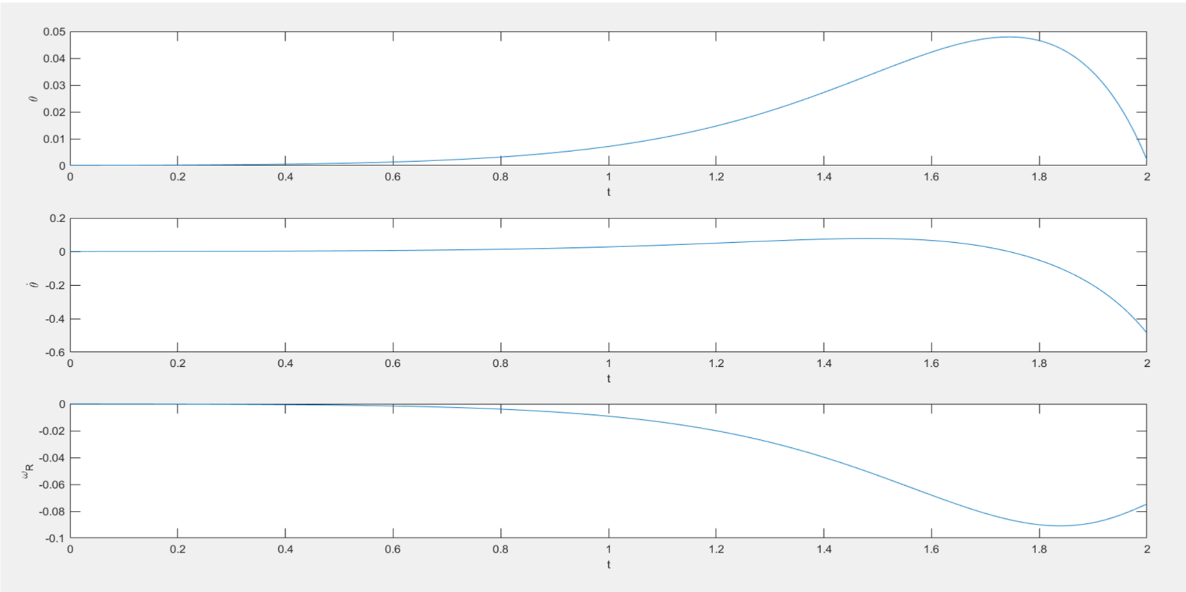
3.1.4 Analysis

A matlab program was simulated using the "TWIDDLE" algorithm for PID tuning.

Open Loop System Ouput:



Closed Loop System:



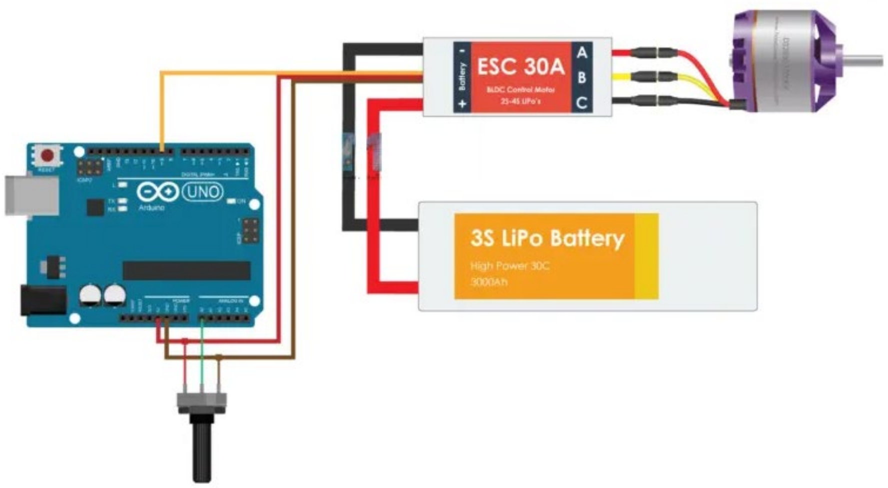
Here, theta being the position, theta prime being the change of position with respect to time and omeca being the reaction speed of motor.

3.2 Design Method (PO(a))

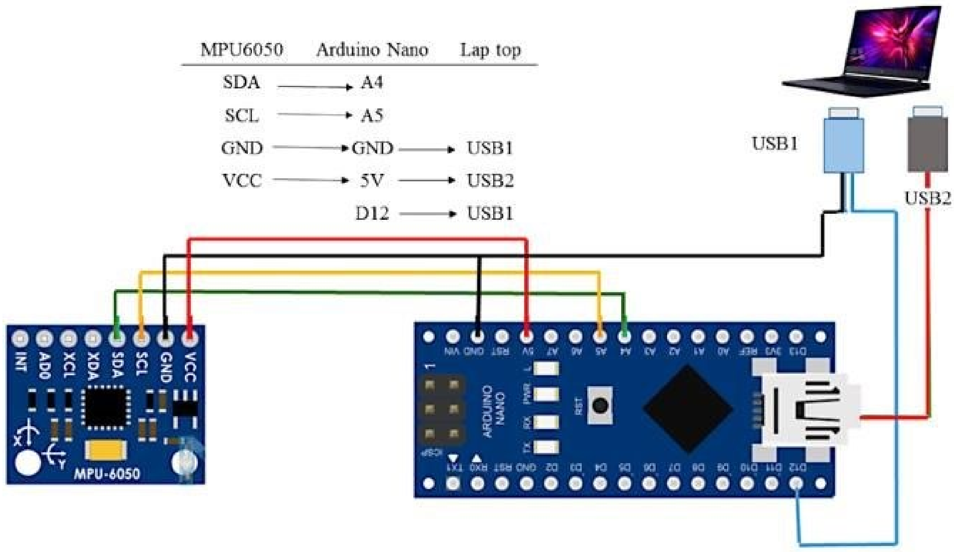
◦ Components Used:

Name	Quantity
6 Degrees of Freedom IMU Breakout – MPU6050	1
Arduino nano	1
ESC 30A Bidirectional BLDC motor speed controller	1
1400 rpm/V Brushless DC motor	1
Rechargeable Battery	1

3.3 Circuit Diagram



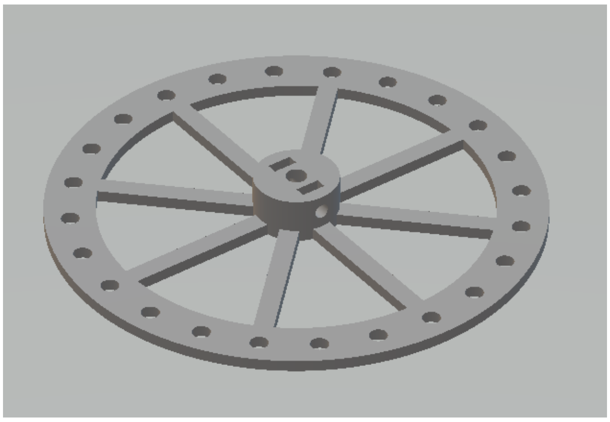
Connection with speed controller and Battery



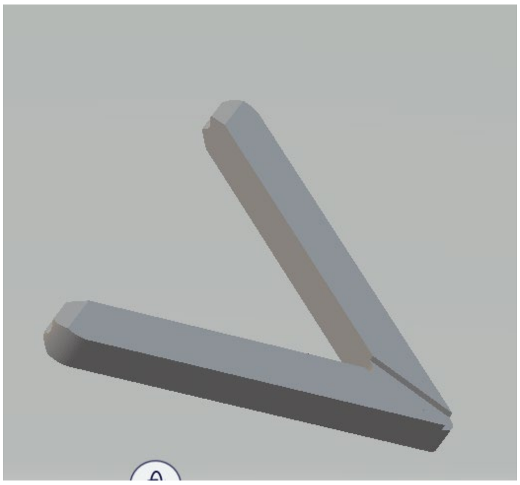
Connection with IMU

3.4 CAD/Hardware Design

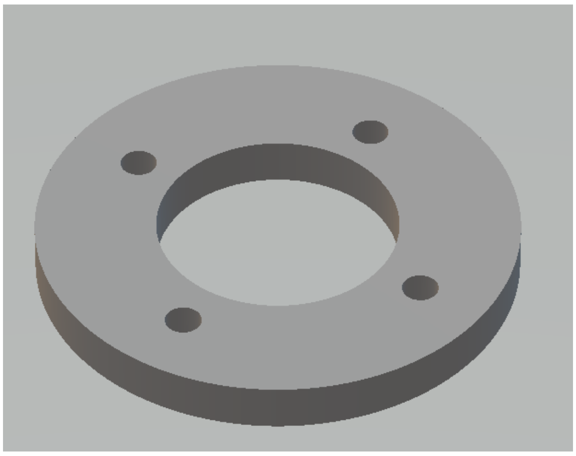
Main Axis



Reaction Wheel



Axis Holder



Motor Support

3.5 Full Source Code of Firmware

```
#include <Wire.h>
#include <Servo.h>

#define KP 3
#define KI 0.5
#define KD 0.0
#define Q 1
#define RANGE 200
#define FIX 5
#define MPU6050_AXOFFSET 158
#define MPU6050_AYOFFSET 9
#define MPU6050_AZOFFSET -91
#define MPU6050_GXOFFSET 19
#define MPU6050_GYOFFSET -42
#define MPU6050_GZOFFSET -26

long sampling_timer;
const int MPU_addr=0x68;  // I2C address of the MPU-6050

int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ; // Raw data of MPU6050
float GAcX, GAcY, GAcZ; // Convert accelerometer to gravity value
float gyro_theta, acc_theta, theta, error_diff, error_int, past_error, past_theta, error, target_angle;
float alpha = 0.96; // Complementary constant
long pulse, pulse_counter;
Servo myservo;

void setup(){
  myservo.attach(9, 1480 - RANGE, 1480 + RANGE);
  myservo.write(90);
  delay(2000);
  Wire.begin();
  init_MPU6050();
  Serial.begin(115200);
  target_angle = 0;
  error_int = 0;
  past_error = 0;
  past_theta = 0;
  pulse_counter = 0;
}

void loop(){
  // Read raw data of MPU6050
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr,14,true);  // request a total of 14 registers
  AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp=Wire.read()<<8|Wire.read();  // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  GyX=Wire.read()<<8|Wire.read();  // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY=Wire.read()<<8|Wire.read();  // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ=Wire.read()<<8|Wire.read();  // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)

  // Raw data of accelerometer corrected by offset value
//  AcX -= MPU6050_AXOFFSET;
//  AcY -= MPU6050_AYOFFSET;
//  AcZ -= MPU6050_AZOFFSET;

  // Convert accelerometer to gravity value
  GAcX = (float) AcX / 4096.0;
  GAcY = (float) AcY / 4096.0;
  GAcZ = (float) AcZ / 4096.0;

  // Calculate Pitch, Roll & Yaw from Accelerometer value
  // Reference are
  // https://engineering.stackexchange.com/questions/3348/calculating-pitch-yaw-and-roll-from-mag-acc-and-gyro-data
  // https://www.dfrobot.com/wiki/index.php/How_to_Use_a_Three-Axis_Accelerometer_for_Tilt_Sensing
  acc_theta = atan ((GAcY - (float)MPU6050_AYOFFSET/4096.0) / sqrt(GAcX * GAcX + GAcZ * GAcZ)) * 57.29577951; //
180 / PI = 57.29577951
  gyro_theta = (float)(GyX - MPU6050_GXOFFSET) * 0.000244140625;


  // Calculate Pitch, Roll & Yaw by Complementary Filter
  // Reference is http://www.geekmomprojects.com/gyroscopes-and-accelerometers-on-a-chip/
  // Filtered Angle = α × (Gyroscope Angle) + (1 − α) × (Accelerometer Angle)
  // where α = τ/(τ + Δt)   and   (Gyroscope Angle) = (Last Measured Filtered Angle) + ω×Δt
  // Δt = sampling rate, τ = time constant greater than timescale of typical accelerometer noise
  theta = alpha * (gyro_theta + theta) + (1 - alpha) * acc_theta;
  error = target_angle + theta;

  if (pulse_counter < 30) pulse = -1;
```

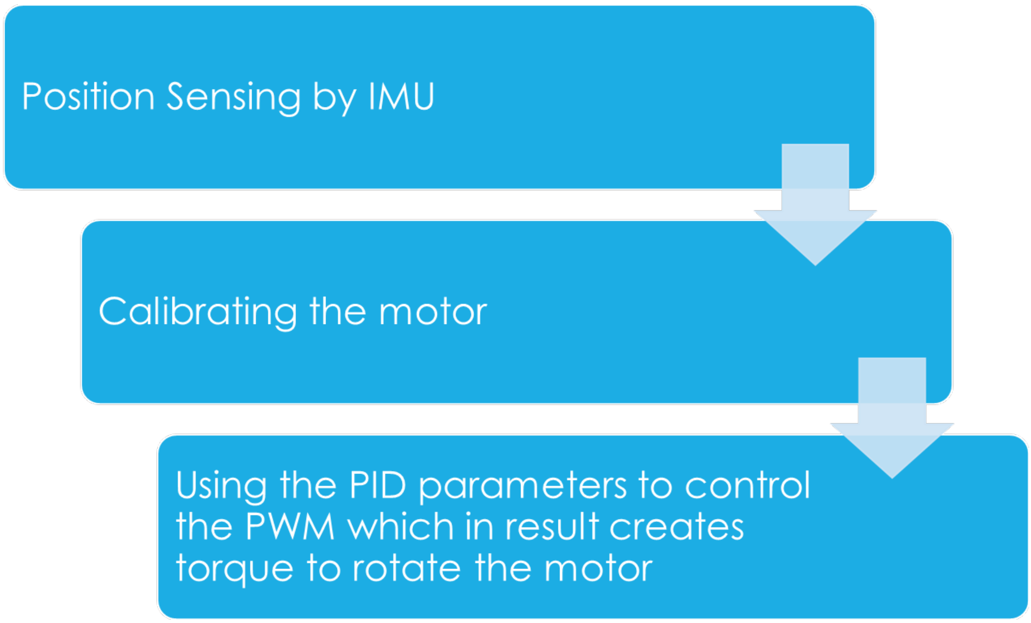
```
  else if (pulse_counter < 50) pulse = -1;
  else if (pulse_counter < 70) pulse = 2;
  else pulse_counter = pulse = 0;

  pulse_counter++;

  // if (theta < FIX && theta > -FIX)
  //   error = (theta - past_theta) ? -FIX : FIX;

  error = (int) error / Q;
  error *= Q;
  Serial.println(error);
  error_int += error * 4e-4;
  error_diff = (error - past_error) / 4e-4;
  past_error = error;
  past_theta = theta;

  float output = KP * error + KI * error_int + KD * error_diff;

  output = output * pulse;

  output = constrain(output + 90, 0, 180);
  output = map(output, 0, 180, 1480 - RANGE, 1480 + RANGE);
  myservo.writeMicroseconds(output);

  // Sampling Timer
  while(micros() - sampling_timer < 4000); //
  sampling_timer = micros(); //Reset the sampling timer
}

void init_MPU6050(){
  //MPU6050 Initializing & Reset
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);  // PWR_MGMT_1 register
  Wire.write(0);     // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);

  //MPU6050 Clock Type
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);  // PWR_MGMT_1 register
  Wire.write(0x03);     // Selection Clock 'PLL with Z axis gyroscope reference'
  Wire.endTransmission(true);

  //MPU6050 Gyroscope Configuration Setting
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x1B);  // Gyroscope Configuration register
  //Wire.write(0x00);     // FS_SEL=0, Full Scale Range = +/- 250 [degree/sec]
  //Wire.write(0x08);     // FS_SEL=1, Full Scale Range = +/- 500 [degree/sec]
  //Wire.write(0x10);     // FS_SEL=2, Full Scale Range = +/- 1000 [degree/sec]
  Wire.write(0x18);     // FS_SEL=3, Full Scale Range = +/- 2000 [degree/sec]
  Wire.endTransmission(true);

  //MPU6050 Accelerometer Configuration Setting
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x1C);  // Accelerometer Configuration register
  //Wire.write(0x00);     // AFS_SEL=0, Full Scale Range = +/- 2 [g]
  //Wire.write(0x08);     // AFS_SEL=1, Full Scale Range = +/- 4 [g]
  Wire.write(0x10);     // AFS_SEL=2, Full Scale Range = +/- 8 [g]
  //Wire.write(0x18);     // AFS_SEL=3, Full Scale Range = +/- 10 [g]
  Wire.endTransmission(true);

  //MPU6050 DLPF(Digital Low Pass Filter)
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x1A);  // DLPF_CFG register
  Wire.write(0x00);     // Accel BW 260Hz, Delay 0ms / Gyro BW 256Hz, Delay 0.98ms, Fs 8KHz
  Wire.endTransmission(true);
}
```

Table: Source Code for the main program

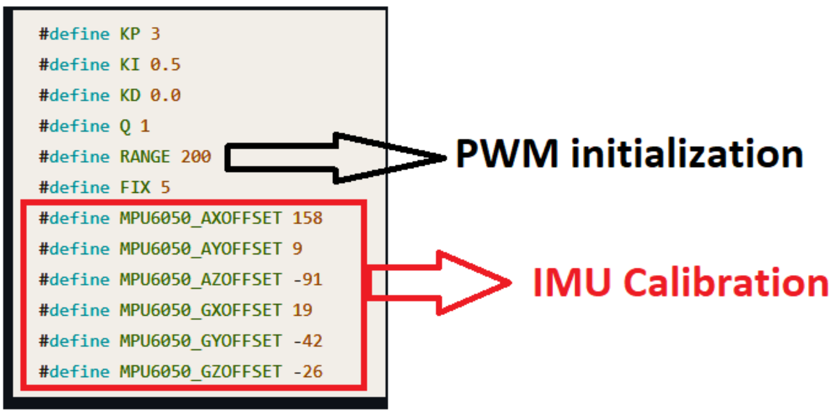
4 Implementation

4.1 Description

Steps Followed:



Calibration of IMU:



Initializing and Updating Parameters:

```
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ; // Raw data of MPU6050
float GAcX, GAcY, GAcZ; // Convert accelerometer to gravity value
float gyro_theta, acc_theta, theta, error_diff, error_int, past_error, past_theta, error, target_angle;
float alpha = 0.96; // Complementary constant
long pulse, pulse_counter;
```

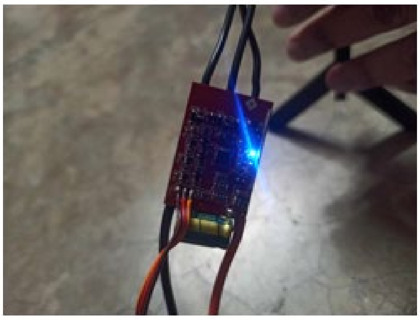
Noise Reduction:

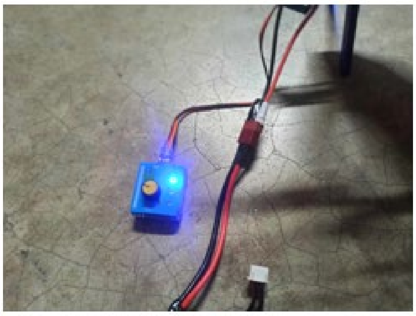
Accelerometer – The accelerometer shows sudden drop in readings due to high frequency noise. So sudden jerk is seen.

Gyro – Low frequency noise.

Solution- Complementary filter. Sensor fusion technique.

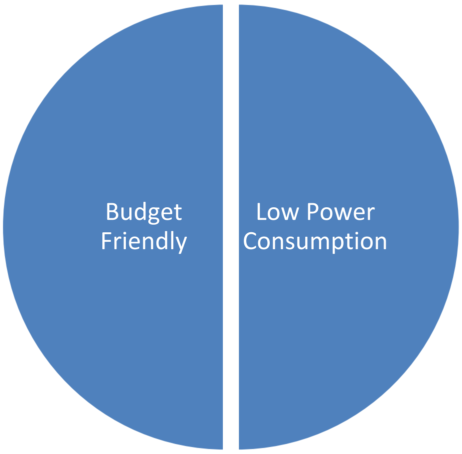
```
// Calculate Pitch, Roll & Yaw from Accelerometer value
// Reference are
// https://engineering.stackexchange.com/questions/3348/calculating-pitch-yaw-and-roll-from-mag-acc-and-gyro-data
// https://www.dfrobot.com/wiki/index.php/How_to_Use_a_Three-Axis_Accelerometer_for_Tilt_Sensing
acc_theta = atan ((GAcY - (float)MPU6050_AYOFFSET/4096.0) / sqrt(GAcX * GAcX + GAcZ * GAcZ)) * 57.29577951; // 180 / PI = 57.29577951
gyro_theta = (float)(GyX - MPU6050_GXOFFSET) * 0.000244140625;


// Calculate Pitch, Roll & Yaw by Complementary Filter
// Reference is http://www.geekmomprojects.com/gyroscopes-and-accelerometers-on-a-chip/
// Filtered Angle = α × (Gyroscope Angle) + (1 − α) × (Accelerometer Angle)
// where α = τ/(τ + Δt)   and   (Gyroscope Angle) = (Last Measured Filtered Angle) + ω×Δt
// Δt = sampling rate, τ = time constant greater than timescale of typical accelerometer noise
theta = alpha * (gyro_theta + theta) + (1 - alpha) * acc_theta;
error = target_angle + theta;
```





5 Design Analysis and Evaluation

5.1 Novelty



5.2 Design Considerations (PO(c))

- Considerations to public health and safety
    - This project was implemented Lithium Ion Battery which has to be handled carefully while using, especially during charging. Reckless use and carelessness may result in damage, although not very significant.
- Considerations to environment
    - Lithium-ion (Li-ion) batteries and devices containing these batteries should not go in household garbage or recycling bins. They can cause fires during transport or at landfills and recyclers. Instead, Li-ion batteries should be taken to separate recycling or household hazardous waste collection points
- Considerations to cultural and societal needs
    - Our Project focuses on enhancing personal experience in photography. Hence it has a significant impact on cultural and social occasions.

5.3 Investigations (PO(d))

- **Obstacles:**
    - Due to unavailability of appropriate motor, we could not incorporate instant braking feature. Hence due to inertia effect, the main axis could not balance itself.
    - Additionally, the motor we used was a BLDC motor and a tachometer to measure angular speed could not be mounted.
    - Due to geometrical complexity, we could not simulate the matlab file to find appropriate tuned PID parameters. We used trial and error method to optimize Kp, Ki and Kd values.
    - The surface needs to have enough friction to handle the rotational speed of motor which is also a reason for additional disturbances.

5.3.1 Design of Experiment

5.3.2 Data Collection

Before Sensor Fusion:

After Sensor Fusion:

5.3.3 Results and Analysis

Although the monopod could balance itself from a certain disturbance angle, due to inertial effect it slided down to other side easily. This is due to the fact that we could not manage the NIDEC 24H servo which has instantaneous braking facility. Also, friction of the surface is an important factor.

5.4 Limitations of Tools (PO(e))

- Desired Motor: Nidec 24H
- Desired Characteristics:
  - **Instantaneous Brake feature**

Output

output		~11 W
torque		~25 mN・m
Stall torque		70~ mN・m
Rpm	4200 min-1	3800~4600 min-1
No-load rpm	5800 min-1	5200~6400 min-1
Momentary maximum torque		70~ mN・m
Instantaneous maximum allowable rpm	5800 min-1	5800~5800 min-1

5.5 Sustainability Evaluation (PO(g))

- **Energy Efficiency**
  - The prototype consumes low electrical power and the bidirectional motor speed controller efficiency is higher than existing unidirectional speed controller.
- **Power Source**:
  - Although the prototype uses Lithium polymer battery, we can use renewable energy sources (i,e, solar cells) to generate electrical power and associate with it.
- **End-of-Life Management**:
  - Lithium-ion (Li-ion) batteries and devices containing these batteries should not go in household garbage or recycling bins. They can cause fires during transport or at landfills and recyclers. Instead, Li-ion batteries should be taken to separate recycling or household hazardous waste collection points.

6 Reflection on Individual and Team work

Individual Contribution of Each Member

ID	Task
1906177	Calibration of IMU and Hardware Assembly
1906178	Modelling the characteristics curve in MATLAB and Hardware assembly
1906191	Modelling the CAD files and Hardware Assembly
1906195	Calibration of Motor and Trial-Error check

7 Communication to External Stakeholders (PO(j))

7.1 Executive Summary

The Flywheel Inverted Pendulum project represents a fascinating intersection of mechanical engineering and control systems. It revolves around achieving self-balancing capabilities using a PID controller. At its core, it integrates key components such as an Arduino Nano for intelligent decision-making, a BLDC motor for precise control and actuation, and an MPU6050 IMU for real-time sensing and feedback. This combination of brain, muscle, and nerve mimics human balance.

7.2 User Manual

- Keep the monopod at any suitable angle and initialize.
- Connect the Battery with the T-connector to start.

7.3 Github Link

[GitHub - sifat53/Reaction-Wheel-Inverted-Pendulum]

8 Project Management and Cost Analysis (PO(k))

8.1 Bill of Materials

Components	No of Units	Cost Per Unit	Total Cost
Arduino Nano R3	1	850	850
A2212 Brushless Motor 1400kV	1	850	850
Bidirectional Speed Controller	1	1500	1500
IMU 6 D.O.F	1	200	200
Lithium Polymer Battery 2200 MAH	1	2500	2500
Breadboard, Jumpers and Extras		200	200
		Subtotal	6100

8.2 Calculation of Per Unit Cost of Prototype

Cost Type	Amount
Raw materials	6100
Connection, Soldering and other costs	200
Labour	Depends on No of Units Produced

8.3 Timeline of Project Implementation

Task	Start Date	End Date	Duration
Finalizing the Proposal	3-Jul	10-Jul	8 days
Group Discussion	21-Jul	25-Jul	5 days
Purchasing the components	26-Jul	28-Jul	3 days
Testing the components	1-Aug	10-Aug	11 days
Purchasing extra/complimentary components	10-Aug	11-Aug	2 days
Prototyping	17-Aug	2-Sep	17 days
Testing the Prototype	2-Sep	11-Sep	10 days

9 References

- GitHub - Tomtom93/Reaction-Wheel-Inverted-Pendulum: Two-Axis Reaction Wheel Inverted Pendulum
- GitHub - remrc/One-Axis-Self-Balancing-Stick-DC