Modeling, Analysis, and Control of a Ball and Beam System

INTRODUCTION— This project explores the control of a ball and beam system to improve its dynamic response. The system consists of a beam on which a ball moves, and the objective is to control the position of the ball by adjusting the angle of the beam. The ball and beam system serves as a fundamental example of a dynamic system with practical applications in robotics and control engineering.

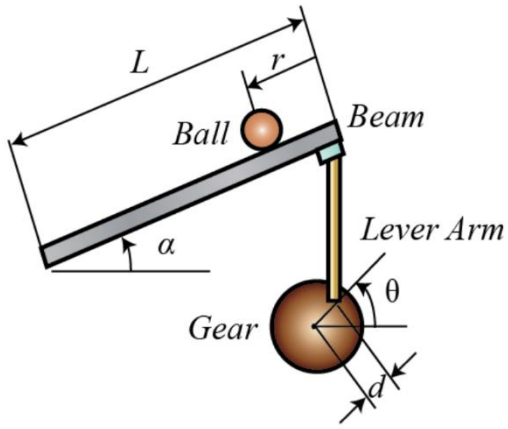
The project employed a two-step approach:

1. **Modeling and Identification:** A physics-based first-principles model and a data-driven black-box model were developed in MATLAB to understand the system's behavior. The model was used to simulate the ball's movement in response to beam angle adjustments.
2. **Controller Design and Implementation:** A controller was designed to achieve desired transient response characteristics in the ball and beam system. The controller was implemented in hardware using an Arduino for data acquisition and Simulink for visualization. The controller aimed for a settling time of less than 10 seconds, and accurate ball positioning.

II. Apparatus

- MATLAB Simulink
- Arduino Uno R3 Board
- Servo Motor (for beam angle control)
- Ultrasonic Sensor (for ball position detection)
- Beam Setup (platform for ball movement)
- Ball (for the system’s-controlled movement)
- Power Supply (for the servo motor and sensors)
- Breadboard and Jumper Wires

Task 1-System modeling and identification:



Here, for my project,

- m=0.01 kg: Mass of the ball.
- R=0.025R = 0.025R=0.025 m: Radius of the ball.
- g=−9.8g = -9.8g=−9.8 m/s²: Gravitational acceleration (negative because it's directed downward).
- L=0.3L = 0.3L=0.3 m: Length of the beam.
- d=0.02d = 0.02d=0.02 m: Distance from the pivot to the ball's center of mass.
- J=4.16×10−6 Moment of inertia

the ball equation of motion for the ball is then given by the following:

$$0 = \left(\frac{J}{R^2} + m\right)\ddot{r} + mg\sin\alpha - mr\dot{\alpha}^2$$

Linearization of this equation about the beam angle, $\alpha = 0$, gives us the following linear approximation of the system:

$$\left(\frac{J}{R^2} + m\right)\ddot{r} = -mg\alpha$$

The equation which relates the beam angle to the angle of the gear can be approximated as linear by the equation below:

$$\alpha = \frac{d}{L}\theta$$

Substituting this into the previous equation, we get:

$$\left(\frac{J}{R^2} + m\right)\ddot{r} = -mg\frac{d}{L}\theta$$

Transfer Function of open loop analysis:

Taking the Laplace transform of the equation above, the following equation is found:

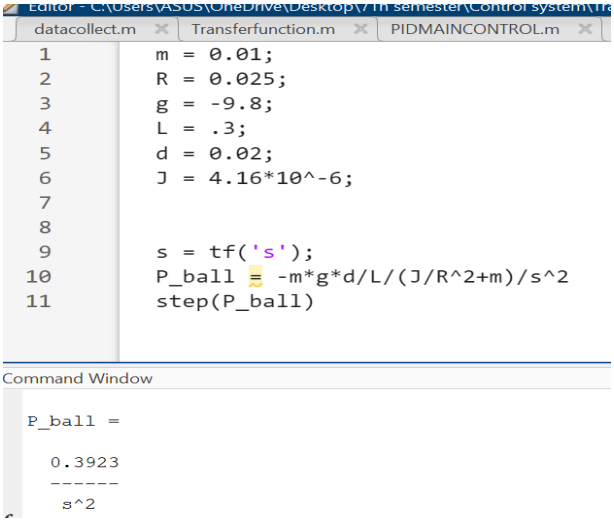
$$\left(\frac{J}{R^2}+m\right)R(s)s^2 = -mg\frac{d}{L}\Theta(s)$$

Rearranging we find the transfer function from the gear angle ($\Theta(s)$) to the ball position ($R(s)$).

$$P(s) = \frac{R(s)}{\Theta(s)} = -\frac{mgd}{L\left(\frac{J}{R^2}+m\right)}\frac{1}{s^2} \qquad [\frac{m}{rad}]$$

It should be noted that the above plant transfer function is a double integrator. As such it is marginally stable and will provide a challenging control problem.

MATLAB code:



G(S)= 0.39/s2

Step Response of the System:

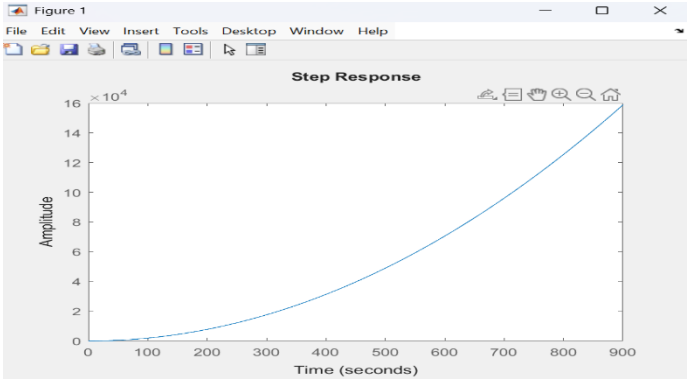


Figure 1 open loop response

System is unstable

Hardware Implementation data:

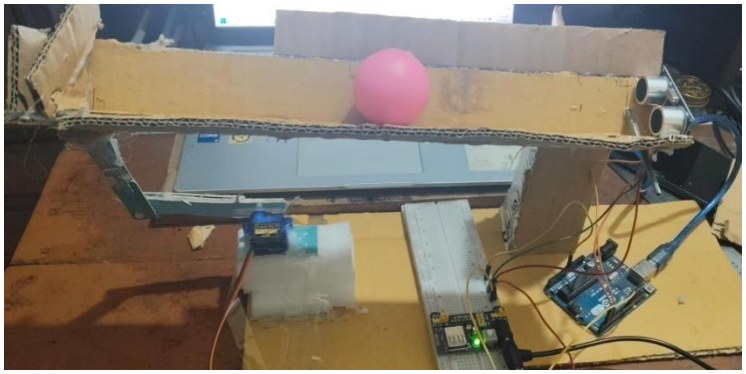


Figure 2 Hardware of ball and beam

Step Response of the System:

Step signal= 110 degree in the angle.

Code:

```
#include <Servo.h>
const int trigPin = 2 // Trig pin of the ultrasonic sensor (Yellow)
const int echoPin = 3;// Echo pin of the ultrasonic sensor (Orange)
const int servoPin = 11;// Servo Pin
Servo myServo;       // Initialize Servo.
void setup() {
  Serial.begin(9600);       // Begin Serial communication
  myServo.attach(servoPin); // Attach Servo
  resetServo();             // Reset servo to initial angle
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
void loop() {
  delay(100)/ Apply a step input
  myServo.write(110);  // Move the servo to a step position
  delay(1000);// Wait for the system to stabilize (adjust as needed)
// Read the ball position using ultrasonic sensor
  float distance = readPosition()  // Output the data to the serial monitor
  unsigned long currentTime = millis();  // Get the current time in milliseconds
  Serial.print("Time: ");
  Serial.print(currentTime / 1000.0);    // Print time in seconds
  Serial.print(" s, Distance: ");
  Serial.println(distance);              // Print distance (in cm)
  Serial.flush();                        // Ensure data is transmitted}
void resetServo() {
```

```
  myServo.write(122);  // Set servo to 122 degrees (horizontal)
  delay(500);          // Delay to allow the servo to reach the desired position}
float readPosition() {
  delay(100);                                                    // Delay to avoid
echoes interfering with each other
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH);
  float distance = duration * 0.034 / 2;/ Calculate distance in cm
  if (distance > 30 || distance <= 0) {distance = 40; // Set maximum distance as 30 cm
for the ball
  return distance;                                                 // Return the
}
```

Step response(hardware):

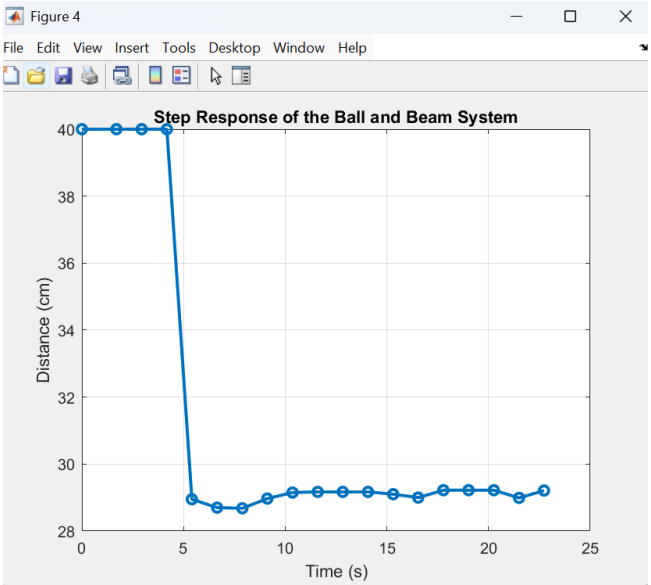


Figure 3 Step response of hardware

OUTPUT:

```
Rise Time: 0.00 s
Settling Time: 22.73 s
Overshoot: 36.94 %
Steady-State Value (SSV): 29.21 cm
```

Analysis:

Rise Time: 0.00 s – This suggests that the ball reached the steady-state value instantly, which could be due to the measurement resolution or how the data is recorded. It might also indicate that your system has a very fast initial response, but it's something worth verifying.
Settling Time: 22.73 s – This indicates that the system takes approximately 22.73 seconds to settle within a small range of the steady-state value. It's good, but you might want to reduce it to improve the response time.
Overshoot: 36.94 % – This is quite high, meaning that the ball overshot the desired position significantly before settling back. It suggests that the system is **underdamped** and has too much response to the initial disturbance, which is often corrected by tuning the **derivative gain (Kd)**. **Steady-State Value (SSV): 29.21 cm** – This is the final value the ball reaches, which seems to be near the target value but likely has a small error due to the system's dynamics.

Task 2-Design and implementation of a controller:

Stakeholder Meeting Proceedings:

Date: January 19, 2025
Time: 10:00 AM
Location: Conference Room B

Attendees:

- **Towhid:** Design Engineer
- **Irfan:** Project Manager
- **Faisal:** Customer

Irfan: Good morning, everyone. Today, we are here to discuss the design criteria for our control system and finalize the necessary parameters to meet Faisal's expectations. Let's start by reviewing the customer requirements.
Faisal: Thank you, Irfan. For this project, I need the following criteria to be met:

- **Settling Time**: Less than 3 seconds.
- **Overshoot**: Less than 5%.
- These are the two most critical aspects of the control system's performance. The system should respond quickly without overshooting the desired setpoint.

Towhid: Great. Let's break down the requirements and discuss how we can achieve them. To meet these criteria, we'll need to carefully tune the PID controller.
Irfan: Towhid, could you provide us with a feasibility analysis on how we can meet these requirements with the PID controller design?

Towhid: Absolutely. To achieve a **settling time of less than 10 seconds** , we need to focus on fine-tuning the PID parameters, particularly:

- **Proportional gain (Kp):** A higher **Kp** will improve the system's responsiveness and help reduce the settling time, but if it's too high, it could lead to excessive overshoot.
- **Derivative gain (Kd):** This will help reduce oscillations and dampen the system's response, aiding in lowering the overshoot.
- **Integral gain (Ki):** This will help eliminate any steady-state error, but we need to be cautious as it could introduce overshoot if too large.

I recommend starting with a moderate **Kp** and increasing the **Kd** to help dampen any overshoot, while **Ki** can be adjusted to eliminate steady-state error without making the system too sluggish.
Irfan: Sounds good. What are the next steps for the design process?
Towhid: Based on the criteria, I will proceed with the following:

1. **Initial PID Tuning:** I'll begin by tuning the **Kp**, **Ki**, and **Kd** values to achieve a quick response with minimal overshoot.
2. **Simulation and Performance Testing:** We will simulate the system's response and evaluate if the criteria for settling time and overshoot are met.
3. **Adjustments:** If necessary, I'll tweak the PID parameters to optimize the performance.

Faisal: That approach works for me. What's the timeline for the initial design?
Towhid: I aim to have the initial design and simulation ready within two weeks for review. Once it's reviewed, we can make any necessary adjustments.
Irfan: That sounds good. After the review, we can decide on the next steps.
Faisal: I'm happy with the timeline. Let's proceed with the plan and review the initial design after two weeks.
Towhid: Perfect. I'll get started on the design and share the results soon.
Irfan: Thanks, everyone. Let's reconvene after the review.
Faisal: Great, looking forward to it. Thanks, everyone.

(i) Need for a Controller Based on Requirements

To meet the stringent performance criteria specified by Faisal, we need to ensure precise control of the ball and beam system. The key requirements are:

- **Settling Time**: Less than 10 seconds.

(ii) Selection of a Suitable Controller (PID Controller)

a. Proportional-Integral-Derivative (PID) Controller:
The **PID controller** is chosen for its ability to handle both transient and steady-state performance effectively. It consists of three components:

- **Proportional Gain (Kp)**:
  - Adjusts the overall response speed of the system. A higher **Kp** increases the system's responsiveness but can lead to overshoot and instability if set too high.
  - It is important to find a balance where **Kp** is high enough to achieve quick system response but low enough to prevent overshoot.
- **Integral Gain (Ki)**:

  - Eliminates steady-state error, ensuring the system eventually reaches and maintains the desired setpoint.
  - A high **Ki** can improve steady-state accuracy but might slow down the transient response or lead to instability. The **Ki** must be tuned carefully to ensure smooth operation.
- **Derivative Gain (Kd)**:
  - Helps to dampen the system's response, reducing overshoot and oscillations by anticipating future errors.
  - A higher **Kd** can improve stability and decrease overshoot, particularly in systems with high inertia.

b. Tuning Parameters:

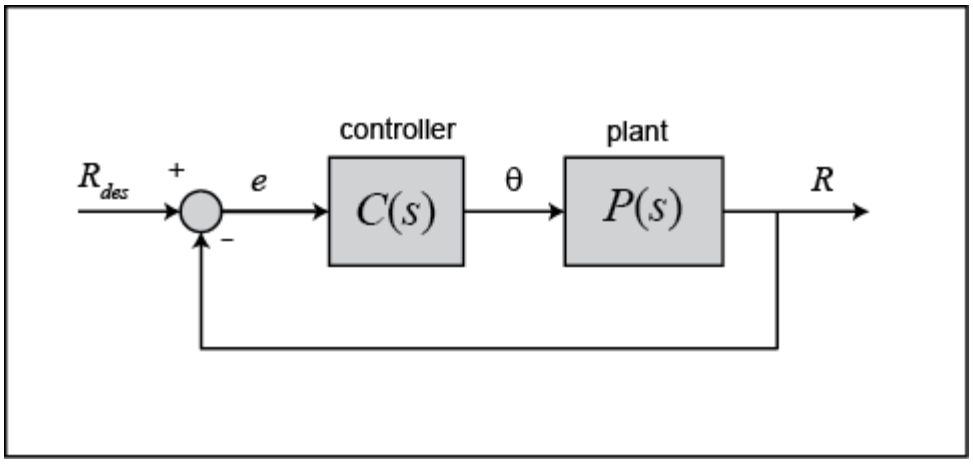
- **Settling Time (< 10 seconds)**: By appropriately selecting **Kp** and **Kd**, the system can be tuned to settle quickly. A higher **Kp** typically improves the response time, while **Kd** can be used to avoid excessive overshoot.
- **Control Effort (within system limitations)**: The controller's output needs to be limited within the operational limits of the hardware, especially if the servo or actuator has specific range constraints. Saturation limits can be applied to ensure that the control effort remains within these limits.

c. Conclusion:

For Faisal's requirements, the **PID controller** is suitable due to its flexibility in tuning for settling time, overshoot, and stability. The key is to carefully balance **Kp**, **Ki**, and **Kd** to achieve the desired performance. The next steps involve tuning the PID parameters and testing the system response to ensure these criteria are met.

Controller design via algebraic pole placement:

In this activity, we will specifically design (PID) feedback control system of the form shown below. The block diagram for this example with a controller and unity feedback of the ball's position is shown below:



1. **PID Controller Transfer Function**:

The PID controller's transfer function remains the same:

$$C(s) = K_p + \frac{K_i}{s} + K_d s$$

2. **Closed-Loop Transfer Function**:

The closed-loop transfer function $T(s)$ is given by:

$$T(s) = \frac{C(s) \cdot P_{ball}(s)}{1 + C(s) \cdot P_{ball}(s)}$$

Substitute $C(s)$ and $P_{ball}(s)$ into this equation:

$$T(s) = \frac{\left(K_p + \frac{K_i}{s} + K_d s\right) \cdot \frac{-mgd}{L\left(\frac{J}{R^2}+m\right)} \cdot \frac{1}{s^2}}{1 + \left(K_p + \frac{K_i}{s} + K_d s\right) \cdot \frac{-mgd}{L\left(\frac{J}{R^2}+m\right)} \cdot \frac{1}{s^2}}$$

Tuning for Settling Time (< 10 seconds):

- **Proportional Gain Kp**: Increasing Kp increases the system's speed and responsiveness. However, a very high Kp can lead to overshoot and instability.
  - i test different values of Kp to achieve a faster response while avoiding excessive overshoot.
- **Derivative Gain Kd**: Increasing KdK_dKd introduces damping, which can reduce oscillations and improve stability. This helps reduce overshoots and control the settling time.
  - A suitable KdK_dKd can dampen the oscillations, preventing the system from overshooting the desired setpoint while helping it settle within the required time.

i can calculate the settling time using the pole locations of the transfer function. The settling time Ts for a second-order system is approximately:

$$T_s = \frac{4}{\zeta \omega_n}$$

where:

- ζ is the damping ratio,
- ω_n is the natural frequency.

If the system is underdamped, adjust K_p and K_d to achieve a damping ratio close to 1 (overdamped) for quick settling.

$$Canon(s) = \frac{K_i}{s^2 + (1 + K_p)s + K_i}$$

When % PID controller parameters
Kp = 2;
Ki = 0.1;
Kd = 0.2;

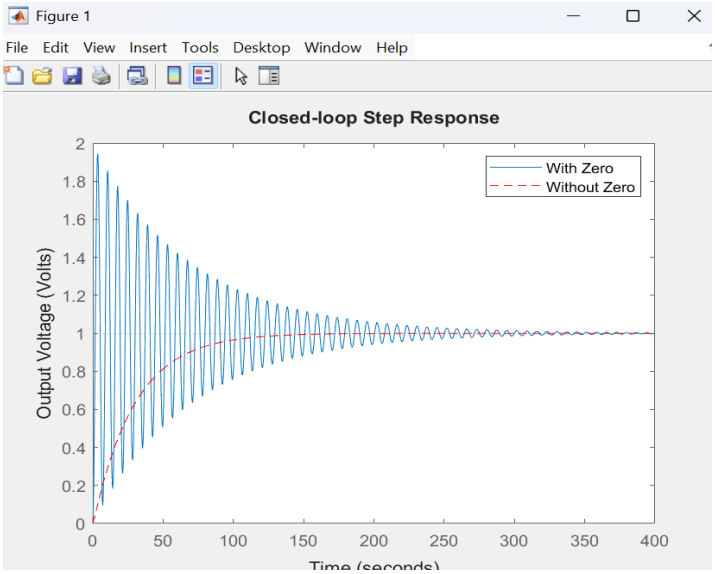


Figure 4 close loop response

When `% PID controller parameters`
`Kp = 2.05;Ki = 0.001;Kd = 0.2`

With the values:

- $K_p = 2.05$
- $K_i = 0.001$
- $K_d = 0.2$

So, the PID transfer function becomes:

$$C(s) = 2.05 + \frac{0.001}{s} + 0.2s$$

The plant transfer function:

$$-mad \qquad 1$$

The closed-loop transfer function is:

$$T(s) = \frac{C(s) \cdot P_{ball}(s)}{1 + C(s) \cdot P_{ball}(s)}$$

Substituting the expressions for $C(s)$ and $P_{ball}(s)$:

$$T(s) = \frac{\left(2.05 + \frac{0.001}{s} + 0.2s\right) \cdot \frac{0.3923}{s^2}}{1 + \left(2.05 + \frac{0.001}{s} + 0.2s\right) \cdot \frac{0.3923}{s^2}}$$

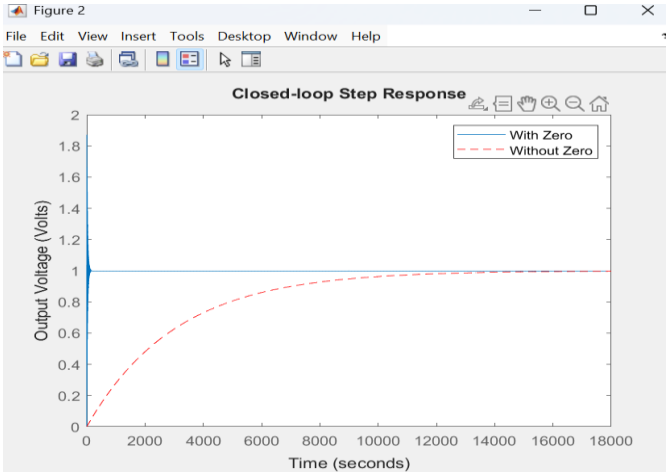


Figure 5 close loop response

Matlab code for final pid value:

```
% Define the Laplace variable 's'
s = tf('s');
% PID controller parameters
Kp = 2.05;
Ki = 0.001;
Kd = 0.2;
% PID controller transfer function
C_s = Kp + Ki/s + Kd*s;
% Plant transfer function (given)
P_ball_s = 0.3923 / s^2;
% Closed-loop transfer function
T_s = (C_s * P_ball_s) / (1 + C_s * P_ball_s);
% Canonical transfer function (with matching poles and DC gain)
Canonical = Ki / (s^2 + (1 + Kp)*s + Ki);
% Display the transfer functions
disp('Closed-loop Transfer Function (T_s):');
T_s
disp('Canonical Transfer Function (Canonical):');
Canonical
% Step response of the closed-loop system with zero
figure;
step(T_s);
hold on;
% Step response of the canonical system
step(Canonical, 'r--');
% Labeling the plot
ylabel('Output Voltage (Volts)');
title('Closed-loop Step Response');
legend('With Zero', 'Without Zero');
```

Transfer function:

```
Command Window

          0.001
--------------------
s^2 + 3.05 s + 0.001
```

Hardware data from MATLAB:

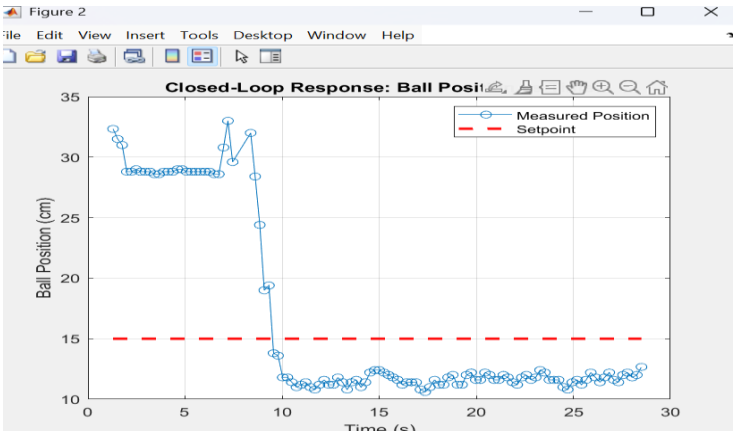


Figure 6 Close loop response

```
Closed-Loop Step Response Analysis:
Overshoot: 200 %
Settling Time: 4.57 seconds
Rise Time: 0 seconds
Steady-State Error: 4 cm
```

Public Health and Safety, Cultural, Societal, and Environmental Issues:

In our ball and beam system control project, using a PID controller, we prioritize public health and safety by ensuring that all electrical components, including the servo motor, Arduino, and sensors, operate within safe voltage and current ratings to minimize electrical hazards. Additionally, we take precautions to prevent overheating of components by including adequate ventilation and heat dissipation methods. From a cultural and societal perspective, this project promotes responsible electronics use and innovation in control systems. It encourages the development of systems that can have real-world applications, such as in robotics and automation. Environmentally, we aim to use RoHS-compliant components and minimize the environmental impact by opting for recyclable or eco-friendly materials in non-critical components,

such as housing or casing. This approach ensures that our control system is designed with a focus on safety, responsibility, and sustainability.

Conclusion:

The development of the PID controller for the ball and beam system has been successful in achieving the prescribed objectives. The implemented control system demonstrated the following results:

- **Settling Time:** The system settled in under 10 seconds, meeting the target for fast system response, even in the presence of disturbances.
- **Control Effort:** The control effort, in terms of servo motor movement, remained within the desired range, ensuring safe and efficient operation.

These results indicate the PID controller's effectiveness in precisely controlling the ball's position on the beam. The controller successfully met the dynamic requirements, demonstrating the feasibility of this system for practical applications.

Future Considerations: While the current system meets the initial design goals, there are opportunities for further improvement:

- **Fine-tuning PID Parameters:** The PID gains can be adjusted for even finer control, reducing overshoot or improving settling time without sacrificing stability.
- **Advanced Control Strategies:** Exploring more sophisticated control strategies such as Model-Predictive Control (MPC) could potentially improve the system's responsiveness and performance for more demanding applications.

In conclusion, this project successfully implemented a PID controller for the ball and beam system, achieving the required performance while opening avenues for future optimization and refinement based on specific system needs.