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

EEE 318 (January 2024)

Control Systems Laboratory

Final Project Report

Section: A1 Group: 03

Object Balancing on a Beam using PID Controller

Course Instructors:

Shafin Bin Hamid, Lecturer
Md. Meherab Hossain, 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: Orni **Full Name: Sayba Kamal Orni** **Student ID: 2006009**	**Signature:** Soumik **Full Name: Soumik Saha** **Student ID: 2006011**
Signature: Dipro **Full Name: Adith Saha Dipro** **Student ID: 2006018**	**Signature:** Adib **Full Name: Md. Adib Rahman** **Student ID: 2006024**
Signature: Nittya **Full Name: Nittya Ananda Biswas** **Student ID: 2006025**	**Signature:** Ashik **Full Name: Khondakar Ashik Shahriar** **Student ID: 2006026**

Table of Contents

1 Abstract 1

2 Introduction 1

3 Design 2

3.1 Problem Formulation 2

3.1.1 Identification of Scope 2

3.1.2 Literature Review 2

3.1.3 Formulation of Problem 2

3.1.4 Analysis 3

3.2 Design Method 4

4 Implementation 6

4.1 Description 6

I. Taking Input from Sonar Sensor: 6

II. PID Output and Angle Mapping: 7

III. PID Tuning (Ziegler-Nichols Method): 9

IV. Servo Angle Feedback and Ball Positioning: 10

VI. Car and Beam Modelling: 12

VII. Servo Motor Modelling: 13

VIII. Simulink Model: 14

4.2 Experiment and Data Collection 15

4.3 Data Analysis 16

4.4 Results 20

5 Design Analysis and Evaluation 20

5.1 Novelty 20

5.2 Design Considerations 21

5.2.1 Considerations to public health and safety 21

5.2.2 Considerations to environment 21

5.2.3 Considerations to cultural and societal needs 21

5.3 Investigations 21

5.3.1 Literature Review 21

5.3.2 Experiment Design 22

5.3.3 Data Analysis and Interpretation 23

5.4 Limitations of Tools 23

5.5 Impact Assessment 25

5.5.1 Assessment of Societal and Cultural Issues 25
5.5.2 Assessment of Health and Safety Issues 26
5.5.3 Assessment of Legal Issues 26
5.6 Sustainability and Environmental Impact Evaluation 27
5.7 Ethical Issues 28
6 Reflection on Individual and Team work 28
6.1 Individual Contribution of Each Member 28
6.2 Log Book of Project Implementation 29
7 Communication 29
7.1 Executive Summary 29
8 Project Management and Cost Analysis 29
8.1 Bill of Materials 29
9 Future Work 30
10 References 30
11 Appendix 31

List of Figures:

Figure 1: Overall System 3
Figure 2: System Setup and Description 4
Figure 3: Methodology 5
Figure 4: PID Output and Angle Conversion 7
Figure 5: PID Output and Angle Mapping 7
Figure 6: System's Physical Constraint 8
Figure 7: Mapped U(Angle) vs U(rad) 8
Figure 8: Sinusoidal Oscillatory Response for Kp=12 9
Figure 9: Car and Beam Model 12
Figure 10:MG996R Servo and its Specifications 13
Figure 11: Simulink Model 14
Figure 12: Response of the System (Test Case 1) 15
Figure 13: Response of the System (Test Case 2) 15
Figure 14: Response of the System (Test Case 3) 16
Figure 15: Model Validation (Test Case 1) 17
Figure 16: Model Validation (Test Case 2) 18
Figure 17: Model Validation (Test Case 3) 19
Figure 18: Sonar Sensor Noisy Data for Ball 23
Figure 19: Car with an attached Board for Improved Accuracy 24
Figure 20: Car and Beam System with Position-Limiting Bar 24

1 Abstract

This project presents the design and implementation of a control system for a car-and-beam balancing mechanism. Using a closed-loop feedback structure, the system maintains the car's position on a desired position on the beam by adjusting its tilt via a servo motor. The system incorporates a sonar sensor to measure the car's position, translating the data into actionable feedback. A PID controller minimizes deviations from the desired position and the servo motor is optimized for precise angular control. Also in this project, Beam and car dynamics are mathematically modelled, with the derived transfer functions capturing the system's physical behaviour. Simulink simulations were made to validate the system, showcasing a robust framework for adaptive balancing with applications in automation and robotics.

2 Introduction

Balancing mechanisms are fundamental in control systems, demonstrating real-time feedback and precision control principles. This project focuses on a car-and-beam setup where the objective is to maintain the car at a desired position on the beam using a PID controller. A PID (Proportional-Integral-Derivative) controller is a fundamental tool in control systems, offering an effective method to achieve stability and precision in dynamic environments. It combines three terms to address different aspects of control. The proportional term reacts to the current error, providing immediate corrections; the integral term accumulates past errors, eliminating steady-state deviations; and the derivative term predicts future errors, dampening oscillations and ensuring smooth responses. This versatility makes PID controllers applicable across a wide range of systems, from robotics to industrial automation.

A servo motor adjusts the beam's angle based on PID controller’s output which ensures accurate corrections. The system is modeled mathematically to capture its dynamics, and Simulink is used for simulation and performance evaluation.

This project highlights the application of control theory in practical scenarios, offering insights into adaptive and robust system design. The importance of a beam balancing system lies in its ability to provide a real-life example of control principles in action. It demonstrates how to achieve a balance between responsiveness and stability in dynamic systems. Through this system, students can learn about tuning PID parameters to achieve optimal system performance, minimizing overshoot, oscillations, and error. Additionally, the system is a useful model for more complex, real-world applications, such as robotic arm control, balancing systems, or even aerospace applications, where precise positioning and balance are critical. By experimenting with the PID controller on this system, users can gain practical experience in system design, feedback control, and disturbance rejection, ultimately making it an indispensable tool for learning and applying control theory in engineering.

3 Design

3.1 Problem Formulation

3.1.1 Identification of Scope

The primary goal of this project is to design a system that uses a PID controller to maintain the position of a car on a beam by adjusting the beam's angle in real time. The system needs to be able to adjust the beam dynamically based on feedback from sensors, ensuring that the car stays at a desired position, regardless of disturbances or system variations. This project serves as both an educational tool and a practical application of control theory.

The system's components include the physical beam setup, sensors to measure the car's position, servos to tilt the beam, and the PID controller itself. Key deliverables for this project include the design of the complete system, the implementation of the PID controller, and testing of the system under different conditions.

the project establishes a foundation for more advanced simulations and the exploration of nonlinearities or complex disturbances that might arise in more sophisticated, real-world systems. While the current project focuses on a relatively simple system, it lays the groundwork for developing more complex control systems that could be applied to industrial, robotic, or aerospace applications, where similar principles of position control and disturbance rejection are critical.

3.1.2 Literature Review

The object and beam system is a well-known example used to understand feedback and stability in control systems. The goal is to balance an object on a beam by adjusting its angle, which is tricky due to its nonlinear and unstable nature. To make it easier to control, the system is often simplified by focusing on its stable point. The PID controller is a popular choice because it's simple and effective. Its proportional, integral, and derivative parts handle position errors, steady-state errors, and sudden changes. Techniques like Ziegler-Nichols are used to fine-tune it (Ogata, 2010).

Recent improvements include adaptive PID controllers that adjust automatically for better performance and hybrid systems that mix PID with fuzzy logic or neural networks (Ali et al., 2020). Many setups use microcontrollers like Arduino, paired with sensors for precise control. While challenges like noise and actuator limits exist, PID controllers remain a reliable option, with research focused on making them even more adaptive and robust.

3.1.3 Formulation of Problem

The primary problem in this project is to design a system that can maintain the position of a car on a beam by adjusting the angle of the beam using a PID controller. The specific challenges involve ensuring that the car stays at a desired position, even when subjected to disturbances such as changes in the weight of the car, friction between the car

and beam, and external forces. The goal is to achieve precise control of the car's position by continually adjusting the angle of the beam in real-time based on feedback from sensors. Aspects we need to consider while designing the system:

1. **System Behavior**: The car needs to be positioned at a desired point along the beam. This requires dynamically adjusting the beam's angle to control the car's motion. The problem involves understanding the relationship between the beam angle, the car's position, and the forces acting on the car.
2. **Control Objective**: The main objective is to use a PID controller to control the beam's angle to maintain the car's position. The controller needs to minimize position errors, reject disturbances, and maintain stability.
3. **Dynamic Model of the System**: The system needs a mathematical model that represents the relationship between the angle of the beam and the position of the car.
4. **Feedback Mechanism**: Accurate sensors are necessary to measure the car's position along the beam. The feedback from these sensors will be used to adjust the beam's angle continuously. The formulation of the problem includes determining how to gather and process this sensor data in real time.
5. **PID Controller Design**: The problem involves designing and tuning a PID controller that responds appropriately to changes in the car's position.
6. **Constraints and Limitations**: Any limitations of the system, such as physical constraints (e.g., the range of motion of the servos, sensor accuracy).

3.1.4 Analysis

The provided diagram illustrates the control system of a Car and beam setup designed to balance a car at a desired position on the beam using a closed-loop feedback mechanism. The system consists of several key components, each playing a crucial role in achieving precise control.

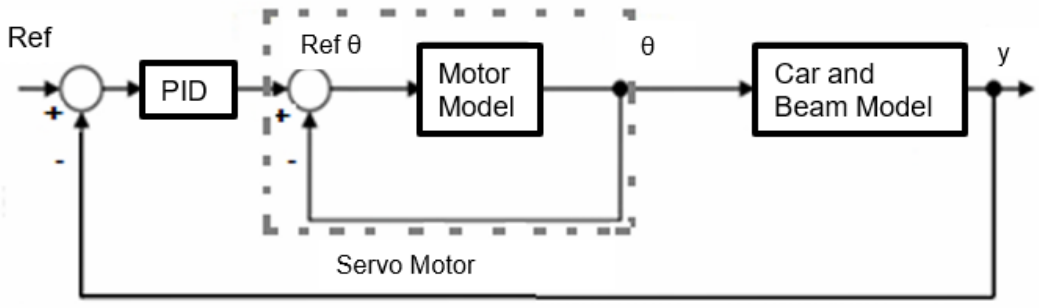

Figure 1: Overall System

The process begins with a **reference signal (Ref)**, representing the desired position of the car on the beam. This reference signal is compared to the ball's actual position (y) in the feedback loop, generating an **error signal**. This error signal quantifies the deviation of the car's current position from the desired position.

The error is fed into a **PID controller**, which generates a control signal to minimize the error. The PID controller uses three terms to achieve this:

- The **proportional (P)** term provides a correction based on the current error.
- The **integral (I)** term accounts for past accumulated errors.
- The **derivative (D)** term anticipates future errors based on the error's rate of change.

The output of the PID controller serves as the input to the **motor model**, representing a digital servo motor. The motor converts the PID signal into a physical tilt angle (θ) for the beam. This tilt causes the car to move in the desired direction.
The **Car and beam model** describes the behavior of the car in response to the beam's tilt. The relationship between the beam angle (θ) and the car's movement is governed by the system's physical dynamics, including factors like gravity, inertia, and friction.
A feedback loop continuously monitors the car's actual position (y) and feeds it back to the system for comparison with the reference position. This real-time feedback allows the system to correct any discrepancies by adjusting the beam angle, ensuring the ball stabilizes at the desired position over time.

3.2 Design Method

We used the following Setup for this project. Here we have a beam and small car is placed on the beam. The object is to balance the car on a desired setpoint, which set by the object placed at the bottom rail which lies on base of this structure.

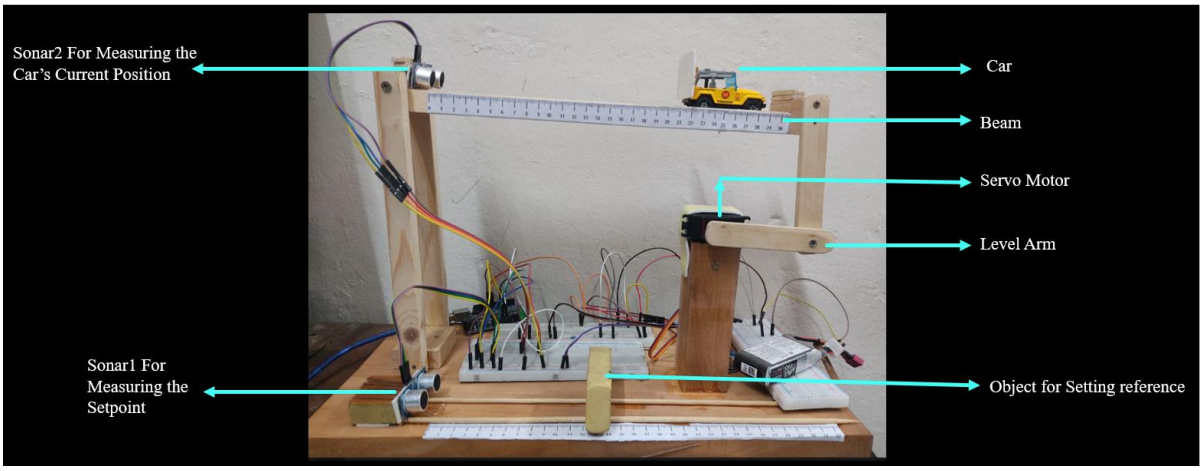

Figure 2: System Setup and Description

Key components of the setup are described below:

- **Sonar1 (Setpoint Sensor)**: Measures the desired position (setpoint) for the car on the beam. It also Provides feedback to the system about where the car should be positioned.

- **Sonar2 (Position Sensor)**: Measures the car's current position on the beam. It supplies feedback to determine the deviation from the setpoint.

- **Car**: The object being balanced or controlled on the beam.Its position is adjusted by tilting the beam.

- **Beam**: Acts as the platform for the car.Tilts to control the movement of the car based on feedback from the sensors.

- **Servo Motor**: Controls the angle of the beam. Adjusts the beam's tilt to move the car toward the desired position.

- **Level Arm**: Connects the servo motor to the beam. Transfers the servo motor's motion to tilt the beam.

- **Object for Setting Reference**: Used to manually set or indicate the setpoint position.

The flowchart illustrates the methodology for the car-on-beam balancing system.

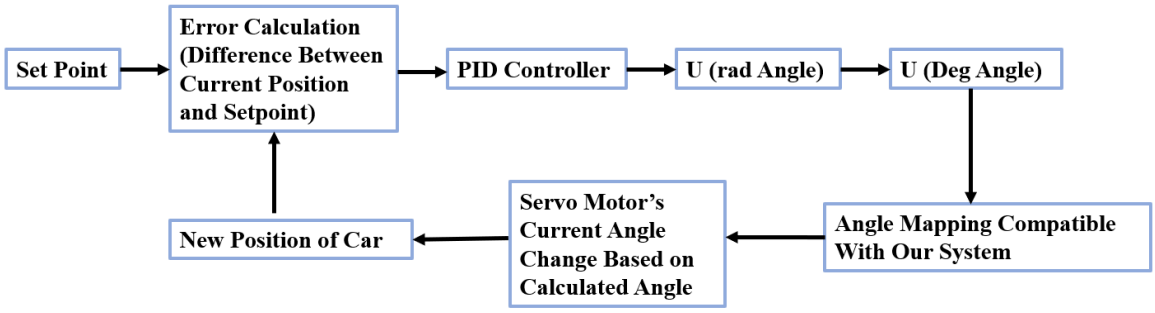

Figure 3: Methodology

It describes how the system calculates the necessary adjustments to keep the car at the setpoint. A detailed explanation of each step is given below:

1. **Setpoint**: The desired position of the car on the beam, as defined by the user or system. This acts as the target input for the control system.

2. Error Calculation: The difference between the **current position** (from Sonar2) and the **setpoint** (from Sonar1). This value indicates how far the car is from the target position.

3. PID Controller: Processes the error using Proportional (P), Integral (I), and Derivative (D) terms: **Proportional**: Reacts to the current error.
Integral: Accounts for the accumulation of past errors.
Derivative: Predicts future errors based on the rate of change.
Produces an output (U) which is the desired correction angle in radians.

4. Angle Conversion: Converts (U) from radians to degrees for compatibility with the servo motor. It ensures the calculated correction is in a usable format for the system.

5. Angle Mapping: Maps the degree angle to the servo motor's operational range. Ensures the servo motor's movement corresponds to the required beam tilt for correction.

6. Servo Motor Adjustment: Adjusts the servo motor's angle based on the mapped value. The beam tilts accordingly, influencing the car's motion to reduce the error.

7. **New Position of Car**: The car moves to a new position on the beam as a result of the adjustment. The system loops back to measure the new position and recalculate the error.

8. **Feedback Loop**: This entire process repeats continuously, creating a closed-loop system where the beam's angle is dynamically adjusted to maintain the car at the setpoint.

4 Implementation

4.1 Description

I. Taking Input from Sonar Sensor:

A sonar sensor works by emitting ultrasonic waves from a transmitter. When the waves hit an object, they bounce back and are received by the sensor. Arduino measures the time taken for the waves to return and calculates the distance using the speed of sound. This is done through the trigger and echo pins, and the results can be used in various applications like obstacle detection or automation

The Arduino sends a high signal to the sensor's **trigger pin** for a short duration (usually 10 microseconds). This activates the sensor to emit an ultrasonic pulse (sound wave) from its transmitter. The ultrasonic transmitter sends out sound waves at a frequency higher than the human hearing range (e.g., 40 kHz). These waves travel through the air until they encounter an object. When the sound waves hit an object, they bounce back toward the sensor. The sensor's **echo pin** detects the reflected sound waves using the ultrasonic receiver. The Arduino measures the time it takes for the sound waves to travel to the object and back. This time is recorded as the **pulse width** on the echo pin.

The main equations used in a sonar sensor system with Arduino are as follows:

1. **Time Measurement**:
   Measure the time ‘t’ taken for the ultrasonic wave to travel to the object and back

   t=time of echo pulse in µs

2. **Distance Calculation**:
   The distance to the object is calculated using the speed of sound in air, approximately 343 m/s or 0.034 cm/µs.

Distance, d= $\frac{v \times t}{2}$ Here, the division by 2 accounts for the round trip of the ultrasonic wave.

So, simplified Equation (in cm) for distance, d(cm)= $\frac{t(\mu s)}{58.2}$

Also, we took 50 samples to reduce noisy data. By averaging these samples, it improves the accuracy of the distance measurement.

II. PID Output and Angle Mapping:

The error(m) is inputted into the PID controller. The PID controller gives us a radian angle 'U' as the output which relate the error to an angle. Then The U(rad) is converted to U(deg).

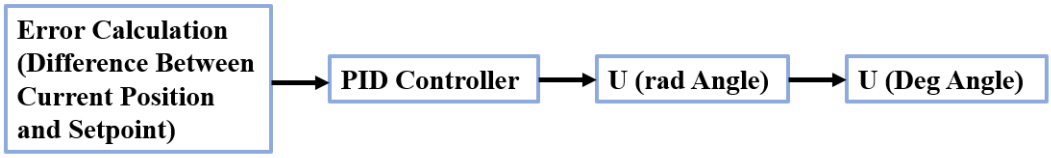

Figure 4: PID Output and Angle Conversion

The saturation check in the PID control algorithm is necessary to ensure that the control action (U) stays within the physical limits of the system. The servo motor cannot rotate beyond its maximum and minimum allowed angles, which are defined as Umax and Umin in radians (or degrees).
So, for the PID Output, we have to set a range for the value of U(rad). For our system, we have considered the range of -0.524 (-30°) to +0.524 (+30°). This max and min values were calculated by observing the values of 'U' for limiting cases, where error was the maximum.

If U goes below Umin, it is set to Umin, and the Saturation flag is enabled. Similarly, if U exceeds Umax, it is set to Umax, and the Saturation flag is enabled.This prevents instability or break the setup.
Also, by enabling the Saturation flag when the output reaches its limit, the system prevents further accumulation of the integral term until the control output returns within bounds. Because if the integral term (I) continues to grow unchecked, it may cause integral windup and might unstable the system.

Now, U(deg) is mapped to a new range of values that are appropriate for the system's physical constraints.

Figure 5: PID Output and Angle Mapping

The equilibrium position of the servo corresponds to an angle of 90°. To move the car to the leftmost position, the servo angle needs to adjust to 130°, while to move the car to the rightmost position, the servo angle must adjust to 60°.

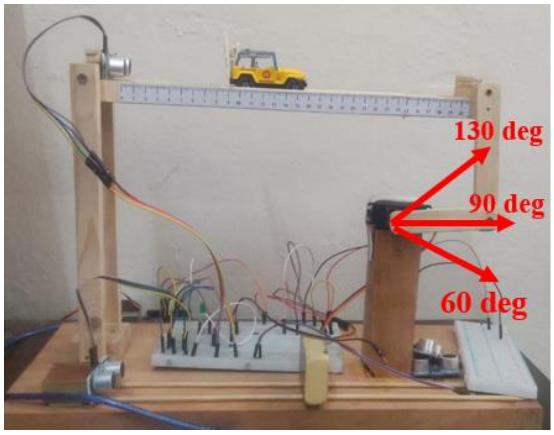

Figure 6: System's Physical Constraint

So, we need to map U(deg) to this range. angle = map(U, Umax, Umin, 130, 60); is used for the mapping purpose. The map() function performs a linear transformation. The general form of a linear transfer function is:
angle(U)=a·U+b, Where:

- U is the input in degrees.
- angle(U) is the output in degrees (the angle of the servo).
- a and b are constants.

The equation for mapping is, angle = ($\frac{U-U_{min}}{U_{max}-U_{min}}$) × (130-60) +130

Where, Umin=−30, Umax=30 So, angle = ($\frac{U+30}{60}$) × (70) + 130

Hence, angle= $\frac{7}{6}$ × U + 95

Which is the linear transform between U and angle.

The following graph shows the relation between U(rad) and Angle.

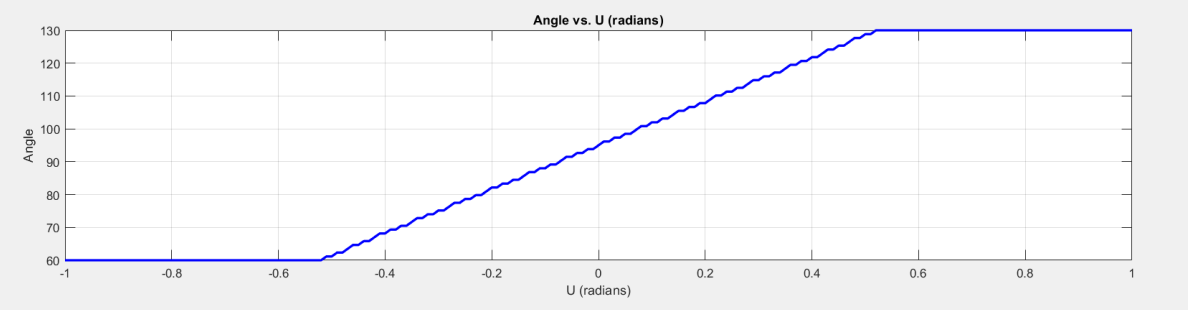

Figure 7: Mapped U(Angle) vs U(rad)

As, U exceeds the max or min range (+0.524 or -0.524) saturations occurs, and servo angle is set to max (130°) or min (60°) respectively.

III. PID Tuning (Ziegler-Nichols Method):

PID tuning involves adjusting the proportional (P), integral (I), and derivative (D) gains to achieve desired system performance. Proper tuning ensures stability, minimizes overshoot, and achieves fast settling times.

We used Ziegler Nichols Method for tuning the PID controller. First, we set K_d and K_I to 0 and gradually increased Kp until the system showed a near-sinusoidal response.

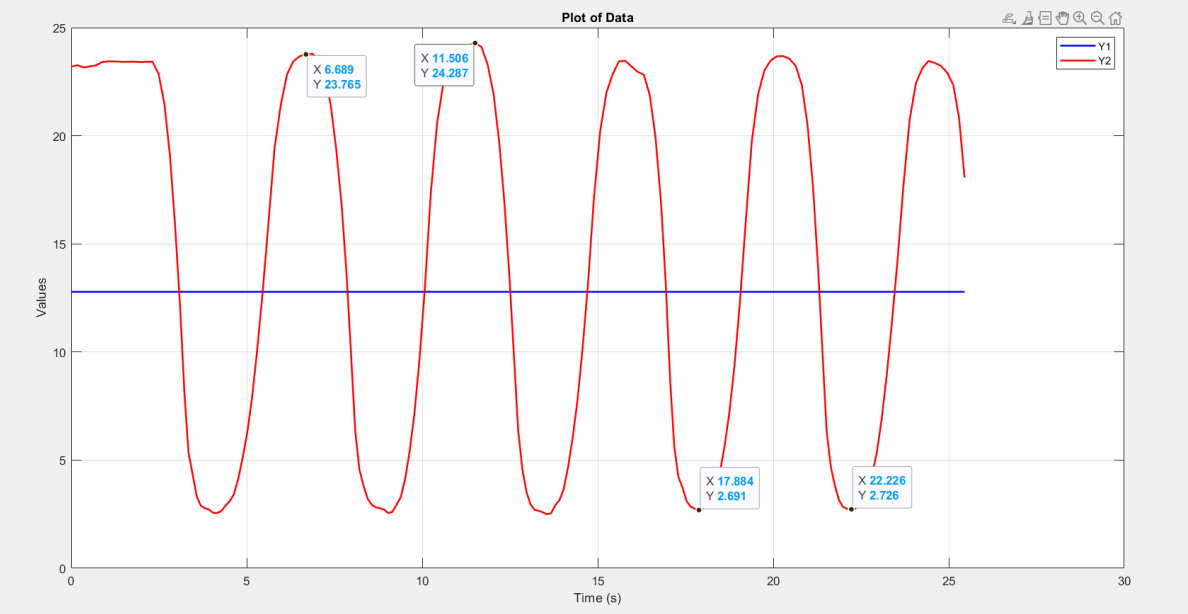

Figure 8: Sinusoidal Oscillatory Response for Kp=12

As shown in the graph, this occurred at K_U = 12,referred to as the ultimate gain. the period is 4.5 s. Using the following equations, we calculated the coefficients:

K_U = 12;

t_U= 4.5s

$K_p = 0.6 \times K_U = 8.4$

$T_i = \frac{t_u}{2 \times K_p} = 0.267$

Ki=3.73;

$K_d = T_d = \frac{t_u K_p}{8} = 4.725$

However, we got better response for K_D=8. So, we used that in our PID controller.

IV. Servo Angle Feedback and Ball Positioning:

We need to adjust the servo's angle based on the magnitude of the error between the car's current position and the setpoint. The adjustment mechanism is designed to provide smooth and adaptive control by varying the step size of the servo movement according to the magnitude of the error.

The servo's current angle (current_angle) is incrementally adjusted towards a desired target angle (angle) based on the calculated error (error). The magnitude of the error determines the size of the step by which the servo angle is updated. The logic is divided into two main cases:

1. Moderate Error (0.0035 < |error| < 0.03)

When the error is moderate, the servo angle is adjusted in small increments of **3 degrees** to ensure smooth and precise corrections. This minimizes overshooting and helps stabilize the system as it approaches the setpoint.

- **If the current angle is less than the target angle (current_angle < angle)**, the servo angle is increased by 3 degrees.
- **If the current angle is greater than the target angle (current_angle > angle)**, the servo angle is decreased by 3 degrees.

2. Large Error (|error| ≥ 0.03)

When the error is large, the servo angle is adjusted more aggressively with steps of **5 degrees**. This allows the system to respond quickly to significant deviations from the setpoint, reducing the time required to correct the position.

- **If the current angle is less than the target angle (current_angle < angle)**, the servo angle is increased by 5 degrees.
- **If the current angle is greater than the target angle (current_angle > angle)**, the servo angle is decreased by 5 degrees.

Steady-State Condition

If the current servo angle matches the desired target angle (current_angle == angle), the servo holds its position without further adjustment. This prevents unnecessary movement and maintains stability.

This approach is designed to provide adaptive control by dynamically adjusting the servo's movement based on the error magnitude. Smaller steps are used for precise corrections when the error is moderate, while larger steps allow for faster responses to significant deviations. The logic balances stability and responsiveness, reducing oscillations near the setpoint and enabling quick recovery from large disturbances.

V. Handling Stuck Scenarios

In the car-and-beam balancing system, a specific logic was designed to address situations where the car becomes "stuck" outside the acceptable range around the setpoint for an extended period. This mechanism ensures the system can recover and maintain stability, even when the car fails to move toward the setpoint due to insufficient control action or Friction.

Stuck Condition Detection
The system identifies a "stuck" condition by monitoring the car's current position (y) and the desired position (setpoint). If the current position is outside of within ±10% of the setpoint, a timer is initiated. This timer tracks the duration for which the ball remains stuck.

- If the car stays beyond the ±10% threshold for more than 3 seconds, the system considers the car as "stuck."
- Conversely, if the car returns to within the acceptable range before the 3-second threshold, the timer is reset, and the stuck condition is cleared.

Corrective Action

When the system determines the car is stuck, it takes corrective action by adjusting the servo to extreme angles. These adjustments aim to provide enough force to dislodge the car and direct it back toward the setpoint.

- **Stuck Below the Setpoint:**
  If the servo angle indicates the beam is tilted downward (servo angle < 90°), the servo is adjusted to 40°. This extreme downward tilt encourages the car to roll toward the setpoint.
- **Stuck Above the Setpoint:**
  If the servo angle indicates the beam is tilted upward (servo angle > 90°), the servo is adjusted to 140°. This extreme upward tilt encourages the car to roll toward the setpoint.

These angles represent significant tilts of the beam, providing sufficient force to overcome friction or other forces keeping the ball stationary. Once the corrective action is applied, the system resumes normal control logic to fine-tune the ball's position around the setpoint.This mechanism ensures robustness in the ball-and-beam control system by enabling recovery from scenarios where the car fails to respond adequately to standard control inputs. Such scenarios may arise due to mechanical constraints, friction, or external disturbances. By incorporating a stuck-detection and recovery mechanism, the system can maintain performance and stability even under challenging conditions.

VI. Car and Beam Modelling:

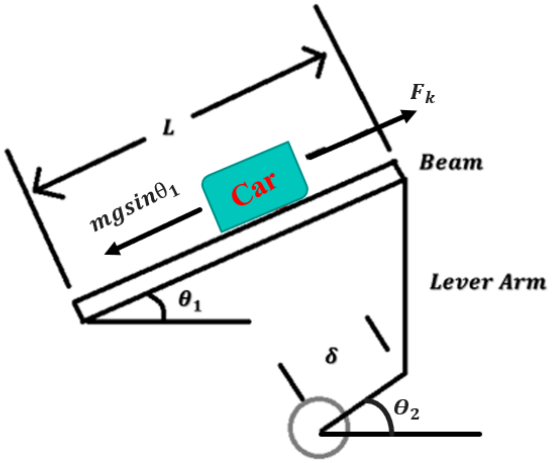

Figure 9: Car and Beam Model

Here,

m= mass of the car	δ=offset
L= length of the beam	x= car position
θ_1 = Beam angle	θ_2= Gear angle

For modeling this part of the system, we have calculated the net force that's being applied on the car and equated it to the acting force on the car. Here clearly the gravitational force tries to take the car downwards and the friction tries to resist. So, from the concept we can say that,

$Ma = Mgsin\theta_1 - F_K = Mgsin\theta_1 - f_x v$

Now, taking it into consideration that $a = \frac{d^2x}{dt^2}$ and $v = \frac{dx}{dt}$

$M\frac{d^2x}{dt^2} = Mgsin\theta_1 - f_x\frac{dx}{dt}$

In laplace domain this equation can be written as –

$(Ms^2 + f_v s)\, X(s) = F(s)$

$(Ms^2 + f_v s)\, X(s) = Mgsin\theta_1(s)$

Now, this angle θ_1 is the inclination of the beam, it is not the same as the angle the motor motor angle. We can approximate that the vertical displacement of the lever arm is half the vertical displacement of the beam. From there, we can write –

Since, $Lsin\theta_1 = 2\times\delta sin\theta_2$

Replacing this value in the equation we get-

$(Ms^2 + f_v s)\, X(s) = \frac{2Mg\delta sin\theta_2(s)}{L}$

$G(s) = \frac{X(s)}{sin\theta_{2(s)}} = \frac{2Mg\delta}{L(Ms^2+f_v s)}$

Now,

m=mass of the car = 0.02kg

δ=offset=0.08m

L= length of the beam=0.3m

Fv= 0.02 (typical value for wood)

Applying These Values,

We get the transfer function of the Beam and Car system,

$$G_B(s)=\frac{5.22}{s^2+s}$$

VII. Servo Motor Modelling:

A Dc Motor Can be Modelled as a first order system. $G_M(s) = \frac{K}{\tau s+1}$

The MG996R has a typical speed specification:

- 0.16 s/60° at 6V, which gives insight into its time constant

Specifications

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 8.5 kgf·cm (4.8 V), 10 kgf·cm (6 V)
- Rotation Angle: 120deg. (+- 60 from center)
- Operating speed: 0.2 s/60º (4.8 V), 0.16 s/60º (6 V)
- Operating voltage: 4.8 V to 7.2 V
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Metal Gears for longer life
- Temperature range: 0 ºC – 55 ºC

Figure 10:MG996R Servo and its Specifications

The time constant (T) is approximately one-fourth of the rise time to 63.2% of the final value. For the MG996R:

T ≈ 0.16 s/4 ≈ 0.04 s

Steady-State Gain (K):

The gain (K) depends on how input PWM values map to angular displacement.

The servo's input is a PWM signal:

- A 1ms pulse typically corresponds to 0°.
- A 2ms pulse typically corresponds to 180°.

Our Servo Motor Moves from 60° to 130°. For simplicity we will consider 60-120° range.

So, 60° maps to 1.33ms Pulse and 120° maps to 1.66ms Pulse.

So, K= $\frac{d\theta}{du} = \frac{60}{0.33}$ = 212.12 deg/s

Servo Motor Transfer Function, $G_M(s) = \frac{181.82}{0.04s+1}$

VIII. Simulink Model:

This is how the whole system looks and it works as-
The user sets a desired position (setpoint) in centimeters, which is compared with the car's current position to calculate the error. This error is fed into a PID controller, which adjusts the motor angle by combining proportional, integral, and derivative components. The PID

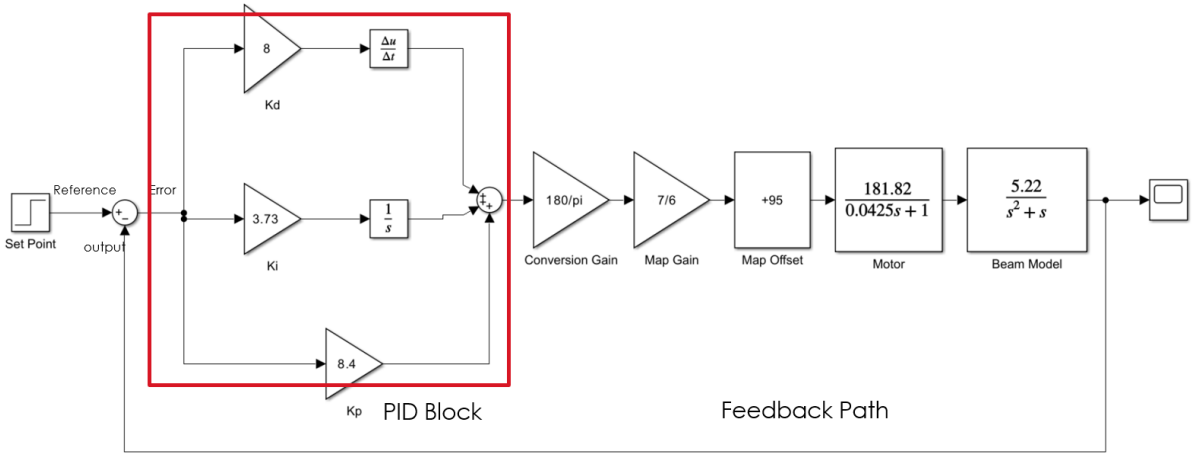

Figure 11: Simulink Model

output, an angle in radians (-0.5236 to 0.5236 rad), is converted to degrees and mapped to the motor's range (60° to 130°). The motor transfer function uses this angle to adjust its position, causing the car to move on the beam. The car's updated position is fed back into the system to recalculate the error, ensuring the car continuously balances at the desired position through this feedback loop.

4.2 Experiment and Data Collection

Here, we will experiment the systems performance for some of the test cases. The response of the system was recorded and plotted using Matlab.

Test Case 1: Here the setpoint = 12.44cm

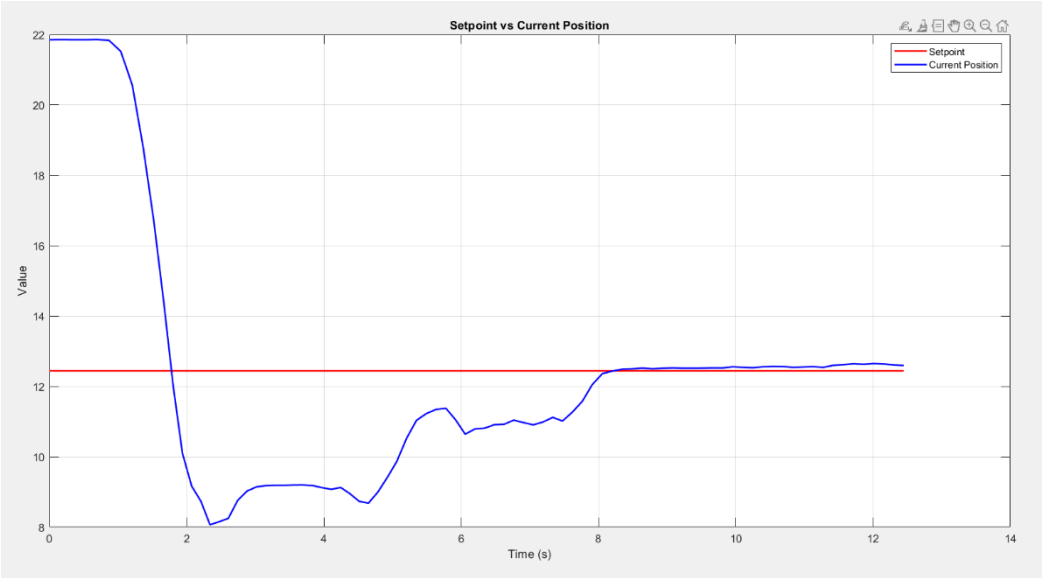

Figure 12: Response of the System (Test Case 1)

The final position of the ball was 12.593cm

Test Case 2: Here the setpoint = 20.292 cm

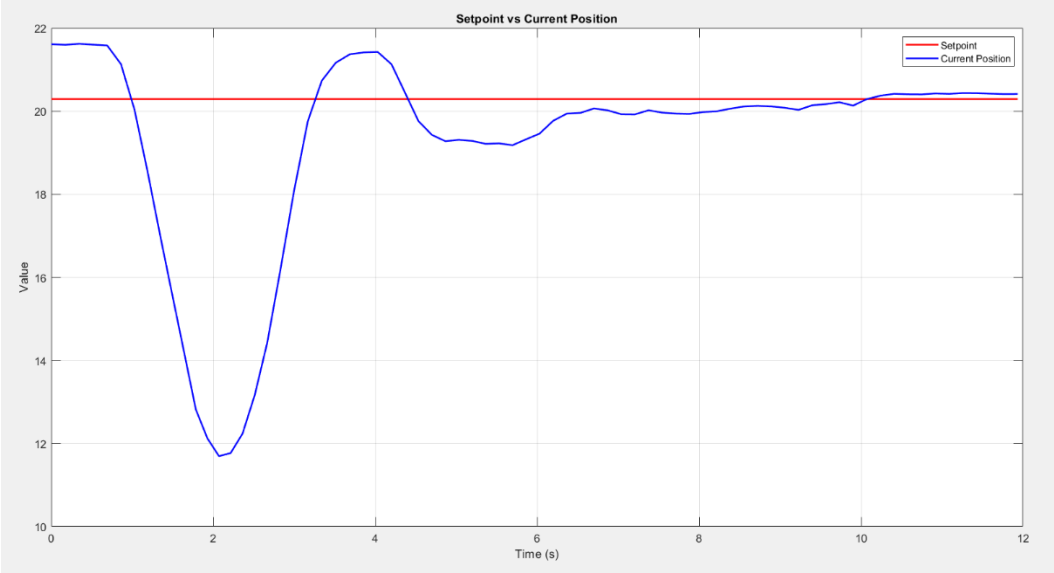

Figure 13: Response of the System (Test Case 2)

The final position of the ball was 20.413cm.

Test Case 3:
Here the setpoint = 7.968cm

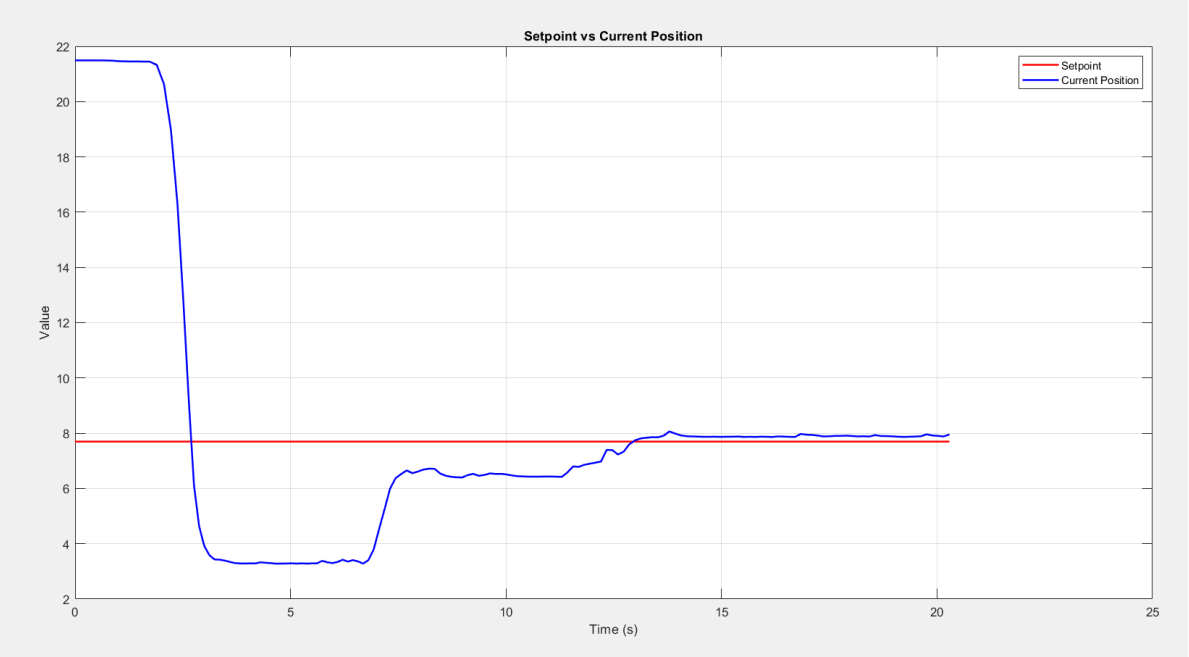

Figure 14: Response of the System (Test Case 3)

The final position of the ball was 7.959cm.

4.3 Data Analysis

Our error threshold was |0.35cm|

Test Case 1:

Steady State Error = (12.593 cm - 12.44 cm) =0.153cm which is less than error threshold.
Percentage Overshoot (%OS) = 75.71%
Peak Time (tp) = 0.69 seconds
Settling Time (ts) = 8 seconds

Test Case 2:

Steady State Error = (20.413 cm – 20.292 cm) =0.121 cm which is less than error threshold.
Percentage Overshoot (%OS) = 6.55%
Peak Time (tp) = 0.35 seconds
Settling Time (ts) = 9.1 seconds

Test Case 3:

Steady State Error = (7.959 cm – 7.968 cm) = -0.009 cm which is less than error threshold.
Percentage Overshoot (%OS): 79.20%
Peak Time (tp): 0.56 seconds
Settling Time (ts): 13.26 seconds

We have also analyzed the step responses to identify total transfer function, using Matlab System toolbox:

For better analysis we normalized the positions with respect to the final position. For
Test case -1:
Model validation:

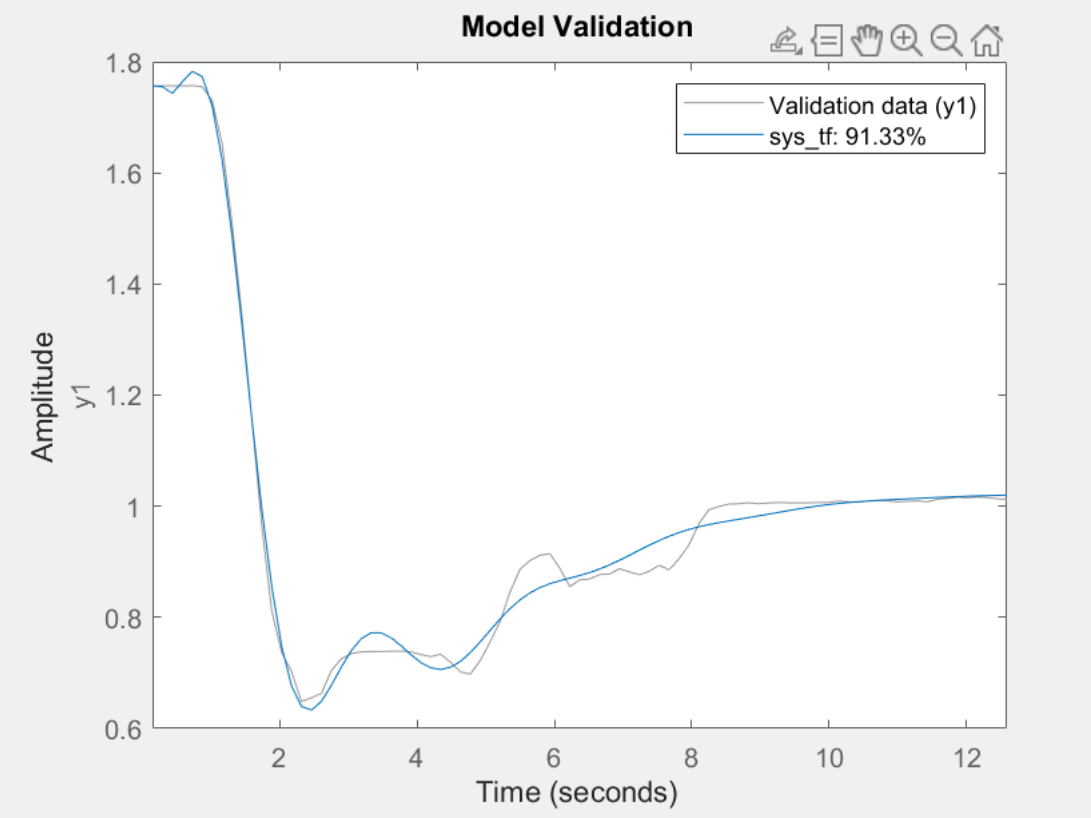

Figure 15: Model Validation (Test Case 1)

Transfer Function found from MATLAB System Identification toolbox:

$$\frac{-215.1s^4-395.6s^3-2472s^2-875s+331.8}{s^5+121.4s^4+291s^3+1174s^2+1055s+324.8}$$

Fit to estimation data: 91.33%
Number of poles: 5 Number of zeros: 4
FPE(Final Prediction Error): 0.00084, MSE(Mean Square Error): 0.0005929

Test Case 2:
Model Validation:

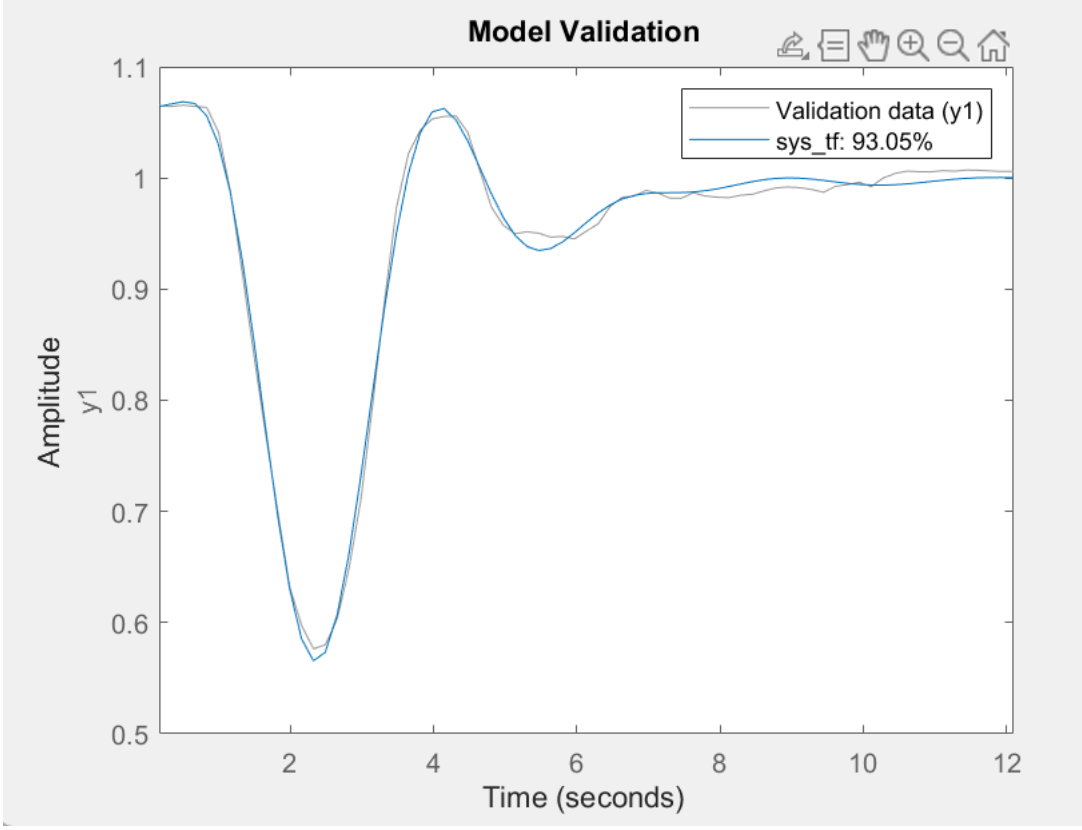

Figure 16: Model Validation (Test Case 2)

Transfer Function found from MATLAB System Identification toolbox:

$$\frac{0.1561s^4+5.112s^3+13.7s^2+15.8s+8.769}{s^5+2.726s^4+11.47s^3+15.65s^2+23.08s+8.765}$$

Fit to estimation data: 93.05%
Number of poles: 5 Number of zeros: 4
FPE(Final Prediction Error): 0.0001115, MSE(Mean Square Error): 7.35e-05

Test Case-3:

Model Validation:

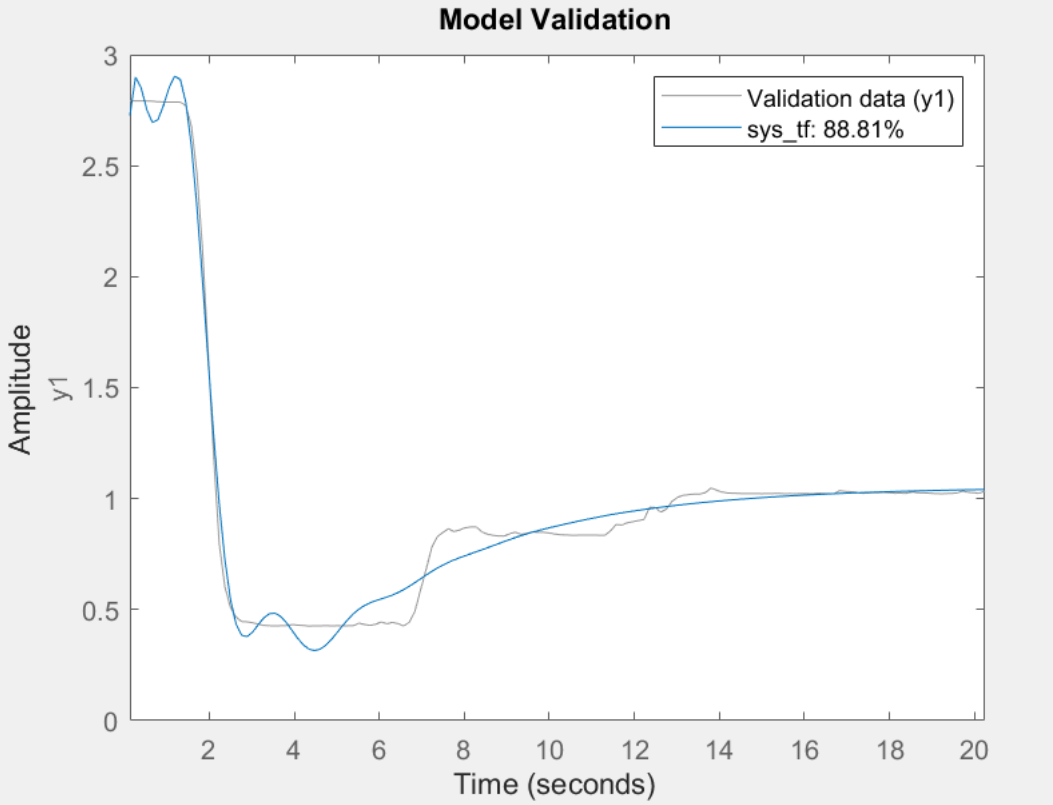

Figure 17: Model Validation (Test Case 3)

Transfer Function found from MATLAB System Identification toolbox:

$$\frac{39.86s^4+30.46s^3+441.9s^2+72.16s+6.643}{s^5+4.896s^4+18.75s^3+40.58s^2+33.23s+6.307}$$

Number of poles: 5 Number of zeros: 4

Fit to estimation data: 88.81%

FPE(Final Prediction Error): 0.005342, MSE(Mean Square Error): 0.004394

4.4 Results

Our system can balance the car at the desired setpoint with certain error threshold. Although, the response time is not same for all the cases.

Response time and other criteria highly depends upon the non-linearities of the system, such as friction, inertia of the car, drag force of the air, response time of the motor etc. Also, these factors depend upon the initial position of the car and the setpoint of the car.
By eliminating or mitigating those factors, we can make sure the system response is quite similar for all the test cases.

Also, these responses were analyzed with MATLAB system identification toolbox
From the analysis we can see that for all the test cases we found that all the transfer functions have 5 poles and 4 zeros. So, we can surely say that our total system has definitely 5 poles and 4 zeros. But the transfer functions have different coefficients for different cases of step responses. These dissimilarities occurred because we couldn't model the nonlinear dynamics and disturbances like friction, inertia of the car, drag force of the air, response time of the motor etc. But despite these errors we can say surely that the system has 5 poles and 4 zeros.

5 Design Analysis and Evaluation

5.1 Novelty

The novelty of this project lies in its innovative adaptations and improvements to the traditional car-and-beam balancing system. To address challenges, the project replaces the conventional ball with a car featuring a flat reflective surface, significantly improving sensor accuracy and feedback reliability.
Design modifications, such as adding a position-limiting bar, further enhance the system's robustness by preventing operational errors caused by sensor limitations. The project also incorporates a stuck-detection mechanism, which identifies and rectifies scenarios where the car fails to return to the desired position by dynamically adjusting the servo angle to dislodge the car. Additionally, a servo angle feedback system adapts the control response based on the magnitude of the error, ensuring smooth corrections for minor deviations and aggressive adjustments for larger disturbances.
These enhancements not only refine the system's functionality but also increase its educational value by providing a hands-on platform for learning and applying control theory. The project's emphasis on systematic problem-solving and its adaptability for real-world applications, such as robotics and automation, further underscore its originality and contribution to the advancement of control system technologies.

5.2 Design Considerations

5.2.1 Considerations to public health and safety

Ensuring the reliability and safety of the control system is crucial to prevent potential hazards during operation. Malfunctions in components such as the servo motor, PID controller, or feedback loop could lead to excessive beam tilting or system unresponsiveness, causing the car to fall and posing risks in demonstration settings. To mitigate these risks, the system should include redundant safety features like physical barriers or stops to limit the beam's tilt angle and emergency shutoff mechanisms to deactivate the motor in case of instability.
Proper electrical and mechanical safety measures are essential, such as using insulated wiring, robust connections, and protective covers to prevent accidental contact with moving parts. Safe testing protocols, including controlled environments and strict parameter tuning, help minimize unpredictable behavior during trials. Additionally, noise and ergonomic factors must be considered to enhance user interaction and ensure prolonged safety. Comprehensive training and documentation are vital to educate users on operational and maintenance practices, while ethical responsibility requires adherence to transparent and safe design principles. When presenting the system in public demonstrations, precautions like clear signage, physical barriers, and secure setups further reduce risks, ensuring the system remains safe and reliable for all use cases.

5.2.2 Considerations to environment

5.2.3 Considerations to cultural and societal needs

The environmental impact of the Car and Beam Control System spans several areas, including material selection, energy consumption, e-waste management, testing, and overall carbon footprint. To minimize resource depletion and pollution, the use of recyclable and sustainable materials like aluminum and steel is encouraged, along with efficient design to reduce waste.
Energy consumption can be mitigated by selecting energy-efficient motors and incorporating renewable energy sources, such as solar panels. Proper e-waste management is essential, with an emphasis on reusability, easy disassembly, and partnering with certified recycling facilities. Testing phases should minimize noise and vibrations, and be conducted in eco-friendly environments to reduce disturbances. A focus on local sourcing of materials helps lower transportation emissions, while carbon offset programs can balance emissions. Additionally, conducting a Life Cycle Assessment (LCA) ensures that the system is optimized for minimal environmental impact at each stage, from raw material extraction to disposal.

5.3 Investigations

5.3.1 Literature Review

The Car and Beam system, also referred to as the "inverted pendulum on a beam," is a classic example of a control system widely studied in engineering and control theory. Literature on this topic spans decades, focusing on the development of controllers, system modeling, and simulation techniques. The system demonstrates the application of Proportional-Integral-Derivative (PID) controllers and advanced control strategies such as Linear Quadratic Regulators (LQR) and Model Predictive Control (MPC).

Early studies highlight the Car and Beam system as a benchmark problem for stability and control design, emphasizing the importance of closed-loop feedback mechanisms. Researchers have extensively modeled the system dynamics using mathematical techniques, accounting for factors such as gravity, friction, and inertia. Simulation platforms like MATLAB/Simulink are frequently employed to design and validate control algorithms.

Recent literature explores the integration of machine learning and adaptive control methods to enhance system robustness under varying conditions. Studies also focus on practical applications, such as robotics and automation, where similar control principles are used to maintain stability.

Overall, the literature underscores the Car and Beam system as a pivotal educational tool and a foundation for advancing control system technologies, with ongoing research enhancing its applicability and efficiency.

5.3.2 Experiment Design

The experiment design for the Car and Beam system aims to test and validate the system's ability to balance a car at a desired position on the beam using a closed-loop control mechanism. The following steps outline the design:

1. **Objective Definition**: The experiment seeks to evaluate the effectiveness of the PID controller in minimizing the error between the reference position and the car's actual position. It also examines the system's response to disturbances and varying reference inputs.
2. **System Setup**: The experimental setup includes the Car and Beam assembly, a PID controller, a motor model, and sensors for measuring the car's position (y) and the beam's tilt angle (θ). The system is connected to a data acquisition system to monitor performance.
3. **Input Parameters**: The reference position (Ref) serves as the input to the system. Different reference positions and disturbance scenarios will be tested to evaluate system adaptability.
4. **Data Collection**: Sensors continuously measure the car's position, beam angle, and error signals. The system's response time, stability, and overshoot are recorded.
5. **Performance Metrics**: The experiment measures steady-state error, settling time, and oscillations to assess system performance. Adjustments to PID parameters (P, I, and D) will optimize performance.
6. **Analysis**: Results are analyzed to confirm the system's reliability and robustness in achieving balance under various conditions.

5.3.3 Data Analysis and Interpretation

Our system successfully balances the car at the desired setpoint within a specific error threshold. However, the response time varies across different scenarios. This variability in response time and other performance criteria is largely influenced by the system's nonlinearities, including factors such as friction, the car's inertia, air drag, and the motor's response time. Additionally, these factors are impacted by the car's initial position and its target setpoint.

By reducing or compensating for these nonlinearities, we can achieve more consistent system responses across all test cases.

The system's dynamics were analyzed using MATLAB's System Identification Toolbox. From this analysis, it was observed that, for all test cases, the transfer functions consistently exhibit five poles and four zeros. Therefore, it is confirmed that the overall system possesses five poles and four zeros. However, the coefficients of these transfer functions differ depending on the specific step response case. These variations arise due to challenges in accurately modeling nonlinear dynamics and disturbances, such as friction, inertia, air drag, and motor response time. Despite these discrepancies, the fundamental structure of the system remains consistent, and we can confidently conclude that the system has five poles and four zeros.

5.4 Limitations of Tools

A. <u>Sonar Sensor:</u>

Initially, we planned to balance a ball on a beam as part of our project. However, due to the ball's curved surface, the sonar sensor struggled to detect its position accurately. The curved geometry caused the reflected ultrasonic waves to scatter unpredictably, leading to fluctuating and unreliable position data. These inaccuracies severely affected the performance of our PID controller, as precise feedback is critical for maintaining balance.

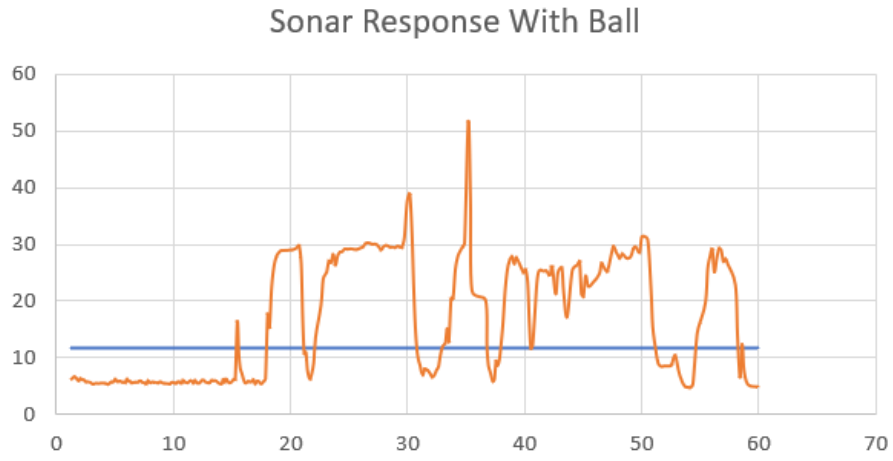

Figure 18: Sonar Sensor Noisy Data for Ball

To address this issue, we adapted the design by replacing the ball with a car. Unlike the

ball, the car has a flat surface that reflects the sonar waves more consistently back to the sensor. This change improved the reliability of the sensor data, allowing the system to receive more stable and precise feedback about the car's position on the beam.

Additionally, we added a board in front of the car to further enhance the sonar sensor's accuracy. The board provides a flat and uniform reflective surface, reducing the chances of signal scattering. This design tweak ensures that the sonar sensor receives clear and consistent echoes, resulting in highly accurate positional data.

Figure 19: Car with an attached Board for Improved Accuracy

With this new setup, the system's overall performance improved significantly as it reduced noise in the feedback loop. The PID controller now adjusts the beam's tilt more precisely to maintain balance. However, this modification introduced new challenges, such as adjusting the controller parameters (e.g., gain tuning) to accommodate the differences in dynamics between the ball and the car.

B. Sonar Sensor's Dead Zone:

To further address the challenges with the sonar sensor, we noticed that when the object gets too close to the sensor, it produces incorrect values due to the limitations of the sensor's minimum range. To mitigate this, we added a physical bar to limit the car's movement near the sonar sensor.

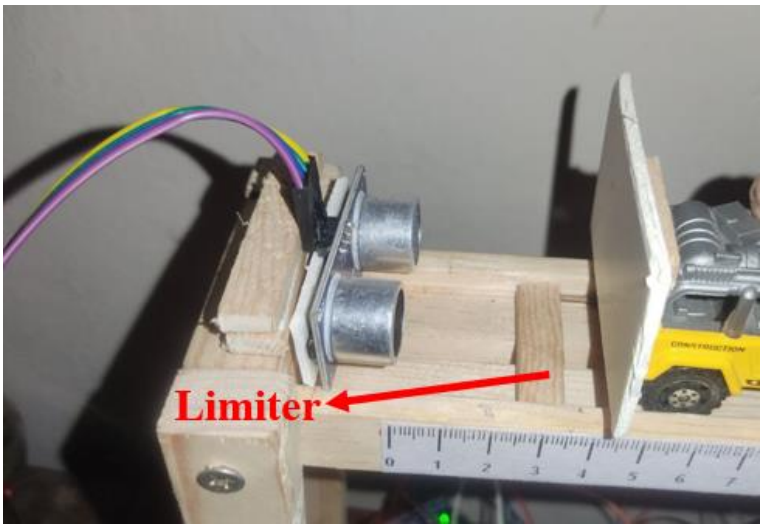

Figure 20: Car and Beam System with Position-Limiting Bar

This ensures the car stays within the effective sensing range of the sonar, preventing erroneous readings and maintaining the accuracy of position feedback. By doing so, we improved the reliability of the system while maintaining the car's ability to balance on the beam effectively.

C. Car's Motion:

we observed that the car occasionally bumps into the ending bar. This problem stems from the servo motor's slower response time compared to the time it takes for the car to travel from one end of the beam to the other. Essentially, the servo motor cannot adjust the beam's tilt quickly enough to correct the car's position, especially when the car gains speed near the ends.

To prevent this issue, we can consider the following solutions:

1. **Using a servo with better response time**:
   A faster servo motor would reduce the delay between the system's detection of the car's position and the corresponding beam adjustment. By improving the response time, the servo can react more quickly to the car's movement, keeping it balanced and preventing it from hitting the ending bar.
2. **Making the beam longer (about 40 cm)**:
   Extending the beam would provide the car with more travel distance. This effectively gives the servo more time to react before the car reaches the ends of the beam. A longer beam could also improve the overall stability of the system, as the car's movements would be more gradual, allowing the servo to make finer adjustments. However, this solution may require structural modifications and recalibration of the PID controller to accommodate the new beam length.

5.5 Impact Assessment

5.5.1 Assessment of Societal and Cultural Issues

The Car and Beam Control System project demonstrates societal relevance by addressing educational gaps, promoting technological literacy, and fostering innovation. As a learning tool, it equips students and engineers with practical skills in control systems, enhancing access to quality STEM education. Its adaptability to various cultural contexts, through localized learning materials and language support, ensures inclusivity and relevance across diverse regions.

From a societal perspective, the project exemplifies the application of advanced technologies to solve real-world problems, inspiring innovation in robotics, automation, and assistive devices. The principles demonstrated in this project can be applied to broader challenges, such as improving mobility for individuals with disabilities or advancing automation in industries, which can benefit society at large.

Cultural considerations include the ethical sourcing of materials and respect for local values during implementation. Additionally, ensuring affordability and accessibility to underserved communities can bridge educational inequities and promote global collaboration.

Potential societal concerns, such as over-reliance on technology or the exclusion of non-digital learning methods, must be balanced by integrating the project with traditional educational approaches. Overall, the Car and Beam project addresses societal and cultural needs effectively, with the potential to inspire innovation and equitable access to technological education worldwide.

5.5.2 Assessment of Health and Safety Issues

The Car and Beam Control System project involves several components and operations that may present health and safety concerns if not properly addressed. Understanding these risks is essential to ensure safe implementation and operation.

1. **Electrical Hazards:** The system typically includes electronic components, such as a digital servo motor and controller circuits, which require careful handling. Improper wiring or exposure to electrical parts could lead to shocks or short circuits. Ensuring proper insulation and adhering to safety standards during construction can mitigate these risks.
2. **Mechanical Hazards:** The beam's tilting motion and the car's movement pose a potential risk of injury, particularly if the system operates at high speeds or is scaled to larger models. Proper containment, speed limits, and securing moving parts can reduce the likelihood of mechanical accidents.
3. **Overheating of Components:** Continuous operation of the motor or PID controller may cause overheating, leading to system failure or potential burns. Integrating thermal protection and using appropriate materials can help manage heat dissipation.
4. **Environment-Specific Risks:** The system should be used in controlled environments to avoid interference or accidents involving unintended objects or people.

By implementing safety guidelines, proper training, and rigorous testing, these health and safety risks can be effectively minimized, ensuring safe project operation.

5.5.3 Assessment of Legal Issues

The Car and Beam Control System project, while primarily educational, must address several legal considerations to ensure compliance with regulations and avoid potential liabilities. Below are the key legal aspects:

1. Intellectual Property (IP) Rights

The project may involve the use of proprietary components, software, or design methodologies. Ensuring that all third-party hardware, software, or algorithms are appropriately licensed is crucial. If the project uses open-source materials, compliance with licensing terms, such as GNU GPL or MIT licenses, must be maintained to avoid infringement.

2. Product Liability and Safety

If the system is deployed outside academic settings or used as a prototype for commercial applications, ensuring compliance with safety standards is critical. Any malfunctions in the motor or feedback loop that lead to injuries or property damage could expose the developers to legal claims. Adequate warnings and proper documentation for safe operation must accompany the system.

3. Regulatory Compliance

For projects involving servo motors and electrical systems, adherence to regulations such as the Restriction of Hazardous Substances (RoHS) and electromagnetic compatibility (EMC) standards may be required. If deployed in public spaces, compliance with local laws governing equipment safety is essential.

Addressing these legal considerations helps mitigate risks and ensures ethical and lawful project execution.

5.6 Sustainability and Environmental Impact Evaluation

The Car and Beam control system is primarily an educational and experimental tool, but its development and usage have sustainability and environmental implications. From a sustainability perspective, the project can be optimized by using eco-friendly materials and energy-efficient components. For instance, the choice of materials for the beam and car could prioritize recyclability and minimal environmental footprint. Lightweight, durable, and recyclable materials like aluminum or composite plastics could reduce waste and energy consumption during manufacturing.

Energy efficiency is another critical factor. The motor and power supply should be chosen for low energy consumption without compromising performance. Integrating renewable energy sources, such as solar panels, into the system's operation or using rechargeable batteries, can further enhance its sustainability.

The project's environmental impact is minimal when scaled appropriately, as it produces negligible emissions and waste. However, care must be taken in proper disposal or recycling of electronic components, such as sensors and motors, to avoid contributing to electronic waste. Additionally, open-source designs can minimize environmental costs by reducing the need for large-scale manufacturing.

By focusing on energy efficiency, material sustainability, and proper end-of-life management, the Car and Beam system can align with broader environmental and sustainability goals, ensuring a minimal ecological footprint.

5.7Ethical Issues

The Car and Beam Control System, while primarily an educational and experimental model, raises several ethical considerations that must be addressed to ensure its development and application align with ethical principles.

1. Accessibility and Equity

One key issue is ensuring the system is accessible to all learners, regardless of socioeconomic status. If the design and implementation rely on expensive components or proprietary technologies, it risks excluding students and institutions with limited resources. Promoting open-source designs and cost-effective alternatives can help mitigate this inequity.

2. Environmental Responsibility

The materials and energy used in constructing and operating the system should be environmentally sustainable. Ethical concerns arise if non-recyclable materials or energy-intensive components are used without consideration for their environmental impact.

3. Misuse of Technology

While the project is designed for education, the principles of control systems could be misapplied in ways that may harm society, such as the development of surveillance drones or automated weaponry. Ensuring that its use is confined to ethical applications is crucial.

4. Intellectual Property

Sharing and replicating such projects can lead to disputes over intellectual property. Developers must ensure appropriate credit is given and that sharing knowledge benefits the broader community.

6 Reflection on Individual and Team work

6.1 Individual Contribution of Each Member

2006009	Hardware Implementation, Report Writing
2006011	Coding, Hardware Implementation.
2006018	Modelling, Report Writing.
2006024	Simulation, Debugging
2006025	Modelling, Debugging
2006026	Coding, Simulation

6.2 Log Book of Project Implementation

Date	Milestone achieved
09/11/2024	1st hardware model implemented
18/11/2024	Improved hardware model implemented
20/11/2024	Tested with Arduino code for open loop system
25/11/2024	Arduino code debugged
5/12/2024	Tested Arduino code for close loop system
7/12/2024	Simulation analysis done
27/12/2024	Report submission

7 Communication

7.1 Executive Summary

This project aimed to balance a car on a beam using a PID controller, highlighting control system principles. The system employs a sonar sensor for position measurement and a servo motor to adjust the beam's tilt dynamically. Key elements include mathematical modeling of beam dynamics, transfer functions, and simulations using Simulink. The PID controller was tuned using the Ziegler-Nichols method to ensure stability and precision. Test results demonstrated effective balancing with minimal steady-state error. Challenges like sensor noise and system non-linearities were mitigated through design improvements. The project serves as a practical application of feedback control in robotics and automation.

8 Project Management and Cost Analysis

8.1 Bill of Materials

Description	Units	Total Cost
Structure	1	1500
Car	1	100
Sonar Sensor	2	150
Breadboard	3	300
Buck Converter	1	70
Arduino UNO	1	600
Servo Motor	1	700
Jumpers	-	100
	Total =	3520

9 Future Work

For future work on the car and beam balancing system, several areas can be explored to enhance its performance and functionality. One key area is optimizing the PID controller, which could involve fine-tuning its parameters or implementing an adaptive version that adjusts in real-time based on the system's dynamics. Improving sensor integration is another priority, such as adding redundancy with multiple sensors or regularly calibrating the sonar to maintain accurate readings.

Enhancing the mechanical design could involve upgrading the servo motor for faster response times or experimenting with lighter, more responsive beam materials. Additionally, incorporating a motorized drive system to control the car's position could improve precision

10 References

1. Maalini, P. V. M., Prabhakar, G., & Selvaperumal, S. (2016). Modelling and control of ball and beam system using PID controller. *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, 322–326. https://doi.org/10.1109/ICACCCT.2016.7831655

2. Taifour Ali, A., A. M., A., H. A., A., A. Taha, O., & Naseraldeen A., A. (2017). Design and Implementation of Ball and Beam System Using PID Controller. *Automatic Control and Information Sciences*, *3*(1), 1–4. https://doi.org/10.12691/acis-3-1-1

3. Ahmad, B., & Hussain, I. (2017). Design and hardware implementation of ball & beam setup. *2017 Fifth International Conference on Aerospace Science & Engineering (ICASE)*, 1–6. https://doi.org/10.1109/ICASE.2017.8374271

11 Appendix

Arduino Code is given here:

```
#include <Servo.h>
#define Umax 30 // degrees (max angle)
#define Umin -30 // degrees (min angle)
#define Umax_rad 0.524 // radians (max angle in radians)
#define Umin_rad -0.524 // radians (min angle in radians)

const int echoPin2 = 11;
const int trigPin2 = 10;
const int echoPin1 = 5;
const int trigPin1 = 6;

Servo servo;

double setpoint, setpoint_prev;  // In Meters
double y, y_prev; // Meter
double error;
double P, I, D, U;
double I_prev = 0, U_prev = 0, D_prev = 0;
double dt, last_time;
int angle, current_angle;
boolean Saturation = false;

double Kp = 8.4; //
double Ki = 3.73; //
double Kd = 8; //
float measure_1(void);
float measure_2(void);
void move_servo(int);

unsigned long stuck_start_time = 0; // Timer for stuck detection
bool is_stuck = false;

void setup() {
    Serial.begin(9600);
    pinMode(trigPin2, OUTPUT);
    pinMode(echoPin2, INPUT);
    pinMode(trigPin1, OUTPUT);
    pinMode(echoPin1, INPUT);
    servo.attach(9);

    last_time = 0;
    angle = 90;
    current_angle = 90;
    delay(1000);
    move_servo(90);  // Set the servo to 90 degrees initially
    delay(2);
    setpoint_prev = measure_1(); // Initial setpoint (Sonar1)
    delay(2);
    y_prev = measure_2();  // Initial ball position (Sonar2)
    delay(1);
}

void loop() {
    double now = millis();
    dt = (now - last_time) / 1000.00;
    last_time = now;

    setpoint = setpoint_prev; // Setpoint in meters
    setpoint = 0.5 * setpoint + 0.5 * setpoint_prev;  // Digital filter

    y = measure_2();  // Current ball position
    y = 0.53 * y + 0.47 * y_prev;  // Digital filter

    error = (y - setpoint);  // Calculate error
```

```
// Check if the ball is outside ±1% of the setpoint
if (abs(error) > 0.1 * setpoint) {
    if (stuck_start_time == 0)
        stuck_start_time = now;  // Start timing if not already started

    if (now - stuck_start_time > 3000)
        is_stuck = true;  // Mark as stuck after 5 seconds
} else {
    stuck_start_time = 0;  // Reset timer if error is within range
    is_stuck = false;  // Ball is within acceptable range
}

// Adjust servo angle if stuck
if (is_stuck) {
    if (current_angle < 90) {  // Stuck below the setpoint
        current_angle = 40;
        servo.write(current_angle);  // Decrease angle further to push ball
    } else if (current_angle > 90) {
        current_angle = 140; // Stuck above the setpoint
        servo.write(current_angle);  // Increase angle to move ball towards setpoint
    }
}



    // PID Control
    P = Kp * error;
    if (!Saturation) I = I_prev + dt * Ki * error;
    D = (Kd / dt) * (y - y_prev);
    D = 0.56 * D + 0.44 * D_prev;  // Filter D
    U = P + I + round(100 * D) * 0.01;

    // Saturate control action
    if (U < Umin_rad) {
        U = Umin_rad;
        Saturation = true;
    } else if (U > Umax_rad) {
        U = Umax_rad;
        Saturation = true;
    } else {
        Saturation = false;
    }


    U = round(U * (180 / M_PI));  // Convert U to degrees
    angle = map(U, Umax, Umin, 130, 60);

        if (abs(error) > 0.0035 && abs(error) < 0.03) {
          if (current_angle < angle)
          {
            current_angle=current_angle+3;
            servo.write(current_angle);
          }

          else if (current_angle > angle)
          {
            current_angle=current_angle-3;
              servo.write(current_angle);
          }
        }
        else if (abs(error) >= 0.03) {
          if (current_angle < angle)
          {
            current_angle=current_angle+5;
              servo.write(current_angle);
          }

          else if (current_angle > angle)
          {
            current_angle=current_angle-5;
              servo.write(current_angle);
          }

          else if (current_angle == angle)
```

```
          {
            current_angle=current_angle;
              servo.write(current_angle);
          }
        }

    // Print the setpoint and current position to Serial Monitor in CSV format
   unsigned long timestamp = millis();  // Current time in milliseconds
    Serial.print(timestamp);             // Timestamp
    Serial.print(",");
    Serial.print(setpoint * 100, 3);     // Setpoint in cm
    Serial.print(",");
    Serial.println(y * 100, 3);

    // Update previous values
    I_prev = I;
    y_prev = y;
    D_prev = D;
    setpoint_prev = setpoint;

    delay(30);
}

float measure_2(void) { // Consistent function name
    const int numSamples = 50; // Number of samples
    float samples[numSamples];
    float sum = 0.0; // Variable to calculate the sum of samples

    // Collect 50 samples
    for (int i = 0; i < numSamples; i++) {
        long elapsed_time = 0;
        float distance = 0;

        // Trigger the sensor
        digitalWrite(trigPin2, LOW);
        delayMicroseconds(2);
        digitalWrite(trigPin2, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin2, LOW);
        // Measure echo time
        elapsed_time = pulseIn(echoPin2, HIGH, 60000); // Timeout after 30ms
        distance = (float)elapsed_time / 58.2;         // Convert to cm
        samples[i] = distance * 0.01;                  // Convert to meters

        delay(1); // Reduced delay to 1ms between samples
    }

    // Calculate the average distance
    for (int i = 0; i < numSamples; i++) {
        sum += samples[i];
    }
    float averageDistance = sum / numSamples;

    // Return the average distance
    return averageDistance;
}

// Function to measure distance for Sonar 2 with multiple samples for averaging
float measure_1() {
    // Trigger the sensor
    digitalWrite(trigPin1, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin1, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin1, LOW);

    // Measure echo time
    long elapsed_time = pulseIn(echoPin1, HIGH, 60000); // Timeout after 60ms

    // Convert to cm and then to meters
    float distance = (float)elapsed_time / 58.2; // Convert to cm
    return distance * 0.01; // Return distance in meters
}
```

```
void move_servo(int u) {
    servo.write(u);  // Set servo position
}
```