University of Maryland

Department of Electrical and Computer Engineering

ENEE 461
Control Systems Laboratory
Spring 2025

ENEE 461 – Final Project: Ball and Beam PID Controller – Project Report

Gibril Turay and Ryan Gaffere

gatturay@terpmail.umd.edu and **rgaffere@terpmail.umd.edu**

Table of Contents

Introduction

Balancing a ball on a tilting beam represents a classic problem in control systems engineering, known for its inherent instability and real-time control demands. This project aims to explore the practical implementation of a feedback control system using hardware and software co-design principles. By leveraging the capabilities of an Arduino Mega 2560 microcontroller, a Time-of-Flight (ToF) sensor, and a standard servo motor, we created an experimental platform to study dynamic system behavior and closed-loop control.

The system is integrated and managed through Simulink, offering real-time responsiveness and a graphical interface for tuning and deployment. Through the design and execution of this project, we gained hands-on experience in sensor interfacing, control algorithm deployment, system modeling, and hardware integration, all within the constraints of mechanical feasibility and signal noise.

Goal

The primary goal of this project was to develop and implement a closed-loop control system capable of maintaining a rolling ball at a specified position along a horizontal beam by dynamically adjusting the beam's tilt. This required careful coordination between the mechanical setup, electronic hardware, and embedded control logic.

The aim was not only to achieve accurate and stable positioning but also to ensure the system could respond effectively to disturbances and variations in setpoint commands. Achieving this goal involved overcoming challenges such as sensor noise, mechanical backlash, and real-time processing limitations, all while maintaining safety and robustness during autonomous operation.

Design Overview

The system is built around a simple yet effective ball-and-beam mechanism, where an aluminum beam supports a plastic ball that can roll freely under the influence of gravity. The beam is mounted at two ends, one fixed and the other connected to a servo motor that allows it to tilt. A Time-of-Flight sensor is positioned at one end of the beam to continuously track the ball's horizontal position. This measurement is processed by a PID controller implemented in Simulink, which calculates the necessary correction and outputs a PWM signal to the servo motor.

The control model is deployed directly onto the Arduino Mega 2560, enabling the system to operate autonomously. Electrical power is supplied via a 12V input stepped down to 5V using a buck converter, distributed through a breadboard to the sensor and motor. This integrated setup facilitates real-time feedback control, enabling precise ball positioning while providing a platform for control system experimentation and validation.

Materials and Equipment

Hardware

- **Arduino Mega2560**
  A microcontroller reads sensor input and controls the servo motor via Simulink-deployed code.

- **VL53L0X Time-of-Flight Sensor**
  Used as a position sensor to measure the horizontal displacement of the ball along the beam.

- **3001HB Servo Motor (Standard, ~180° rotation)**
  Actuates the beam by adjusting its tilt angle based on the control signal from the PID controller.

- **Ball and Beam Mechanism**
  Custom-built mechanical assembly consisting of:

    - A straight, smooth beam (Aluminum)

    - A rolling ball (plastic)

    - Pivot mechanism to allow beam rotation by the servo (PLA Plastic)

- **Breadboard and Jumper Wires**
  For connecting the ToF sensor and servo to the Arduino and power distribution.

- **12V Power Supply / 5V Step-down Buck-Converter / USB Cable**
  USB connection for programming the Arduino and powering the system.

- **Mounting Components**
  Screws, brackets, and adhesives used to secure the beam, ToF Sensor, and motor.

Software

- **MATLAB**

- **Simulink**

- **Simulink Support Package for Arduino Hardware**
  Enables model deployment and real-time interfacing with Arduino.

Constraints

Design Constraints

The objective of our project was to design a closed-loop control system capable of maintaining a rolling ball at a specified position along a beam by adjusting the beam's angle. This required mechanical, electrical, and software subsystems to work in coordination, with considerations for real-time responsiveness, physical feasibility, and control stability.

System Constraints

Several constraints influenced the design and implementation of our system:

- **Mechanical Limitations:**

    - The beam had to be lightweight and rigid to allow responsive motion.

    - The servo motor's range of motion (~0° to 180°) limited the maximum tilt angle of the beam.

    - Physical backlash and looseness in the mechanical mounting introduced nonlinearity and delay.

- **Sensor Accuracy and Resolution:**

    - The ToF voltage output was subject to noise and had to be put through threshold detection logic and low-pass filtering. Spikes would occur as a result of noise and had to be filtered out.

    - Mechanical misalignment could result in non-linear position readings.

- **Controller Tuning:**

    - The PID controller needed to be finely tuned to balance responsiveness and stability.

    - Aggressive tuning could lead to oscillation or overshoot, while conservative tuning could cause sluggish performance.

- **Real-Time Constraints:**

    - The sampling rate in Simulink had to be fast enough to capture the ball's motion without overloading the Arduino.

    - Latency in signal processing or communication could destabilize the control loop.

- **Software and Hardware Integration:**

    - The Simulink model had to be compatible with the Arduino Support Package and correctly map I/O pins.

    - Debugging is required using External Mode before fully deploying the controller.

- **Power and Safety:**

    - The Arduino and servo had to be powered through an external source within safe operating limits.

    - Safety measures were implemented to prevent hardware damage from excessive servo motion or ball ejection.

These constraints guided our design choices and iterative improvements throughout the project, ensuring the final system was both functional and robust under realistic operating conditions.

Design Overview

Our system consists of a rigid beam mounted on a pivot point, actuated by a servo motor to tilt the beam. A ball placed on the beam rolls under the influence of gravity. A ToF sensor mounted along the beam measures the ball's position, which is fed back into a PID controller implemented in Simulink. The controller processes this feedback and adjusts the servo motor's angle to bring the ball back to the target position.

The Simulink model was designed to:

- Read the ToF sensor value.
- Convert this voltage to a position value.
- Apply a PID control algorithm to compute the required correction.
- Output a PWM signal to control the servo motor angle.

This model was then deployed to the Arduino, allowing it to run the control loop autonomously.

Procedure and Results

Experiment Setup

The experimental setup consists of a ball-on-beam balancing system designed using a combination of 3D-printed components, wood, metal, and electronic control elements. The base of the setup is made from a piece of lumber measuring approximately 65 cm in length and 9.5 cm in width. A vertical wooden support, 10.5 cm high, is attached to one end of the base using screws to provide a rigid mounting point for the time-of-flight (ToF) sensor housing. Opposite this support, a servo motor is mounted on the base approximately 52 cm away, with its own

3D-printed housing. The beam itself is an aluminum beam that is fitted at both ends, one into the sensor housing and the other into the servo housing, allowing it to tilt in response to the servo motor's movement.

To power the system, a 12V input is supplied to a buck converter, which steps the voltage down to 5V and distributes it through a breadboard. The 5V output powers the ToF sensor, the Arduino, and the servo motor. An Arduino Mega 2560 microcontroller is used to read distance measurements from the ToF sensor and control the servo motor accordingly. All connections between the components, the sensor, motor, buck converter, and Arduino are made through jumper wires on the breadboard, forming a compact and functional control circuit. This setup allows the system to measure the position of a ping-pong ball on the beam and automatically adjust the beam angle to maintain balance, forming the basis for experiments in real-time control and feedback systems.

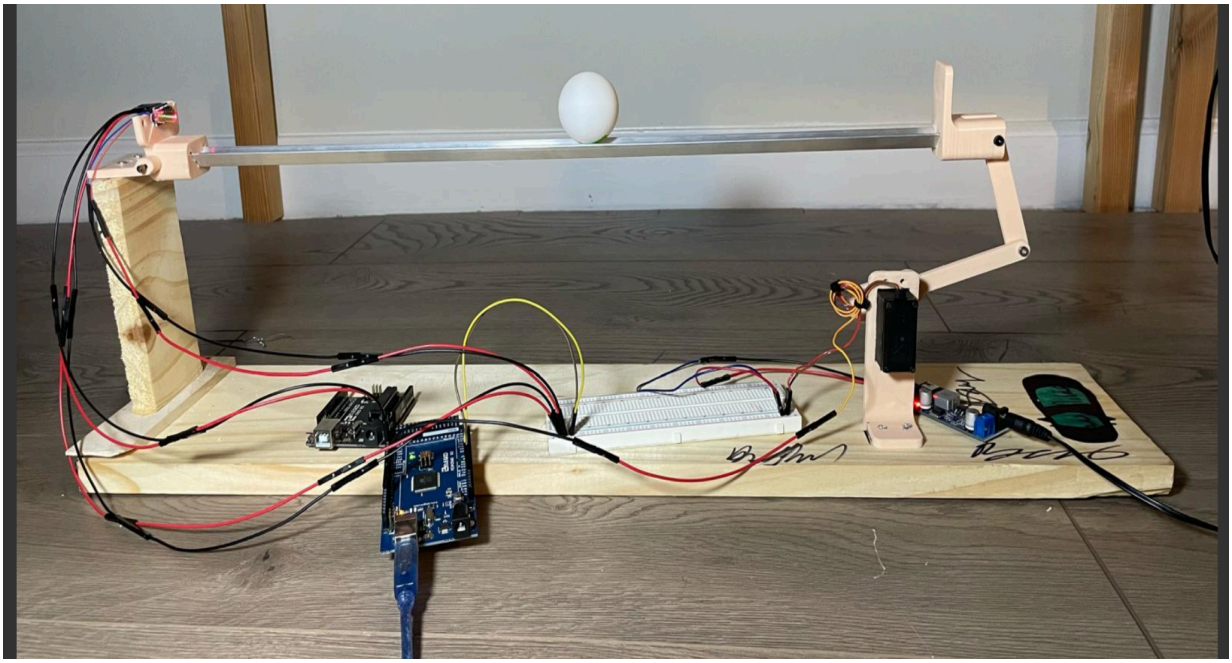

Fig 1: Front View of Experimental setup.

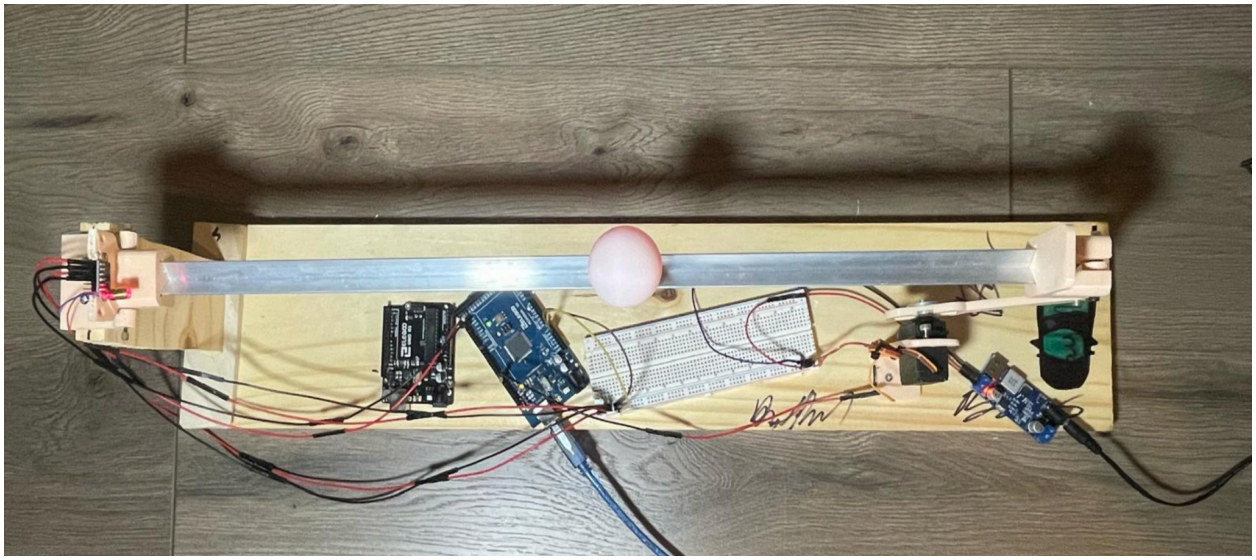

Fig 2: Top View of the Experimental Setup

Implementing the PID Controller

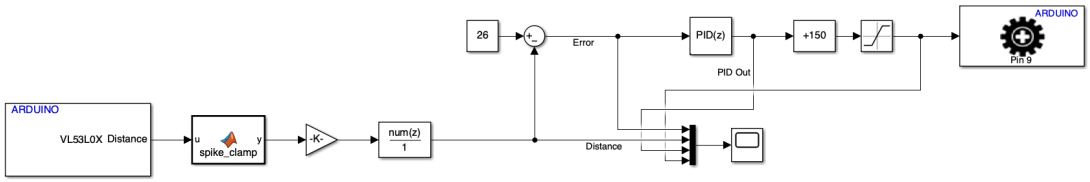

Figure 3: Final Simulink Schematic

After evaluating multiple sensor options, we determined that a Time-of-Flight (ToF) sensor was the most reliable for detecting a matte ping-pong ball. Integrating it with Simulink was straightforward using the Arduino Support Package, which includes an I2C-compatible block for the VL53L0X sensor. We set the I2C address to `0x29` and selected a sample time of **0.03 s**, which provided stable and responsive readings during testing.

Despite this, the raw sensor data exhibited high-frequency noise and occasional spikes, particularly when the ball was at the far end of the beam. To mitigate these anomalies, we implemented a custom MATLAB function that clamps values exceeding a defined threshold, replacing them with the last valid reading:

```
function y = spike_clamp(u)
persistent last_good
threshold = 4000;
if isempty(last_good)
   last_good = u;
end
if abs(u) > threshold
   y = last_good;
else
   y = u;
   last_good = u;
end
```

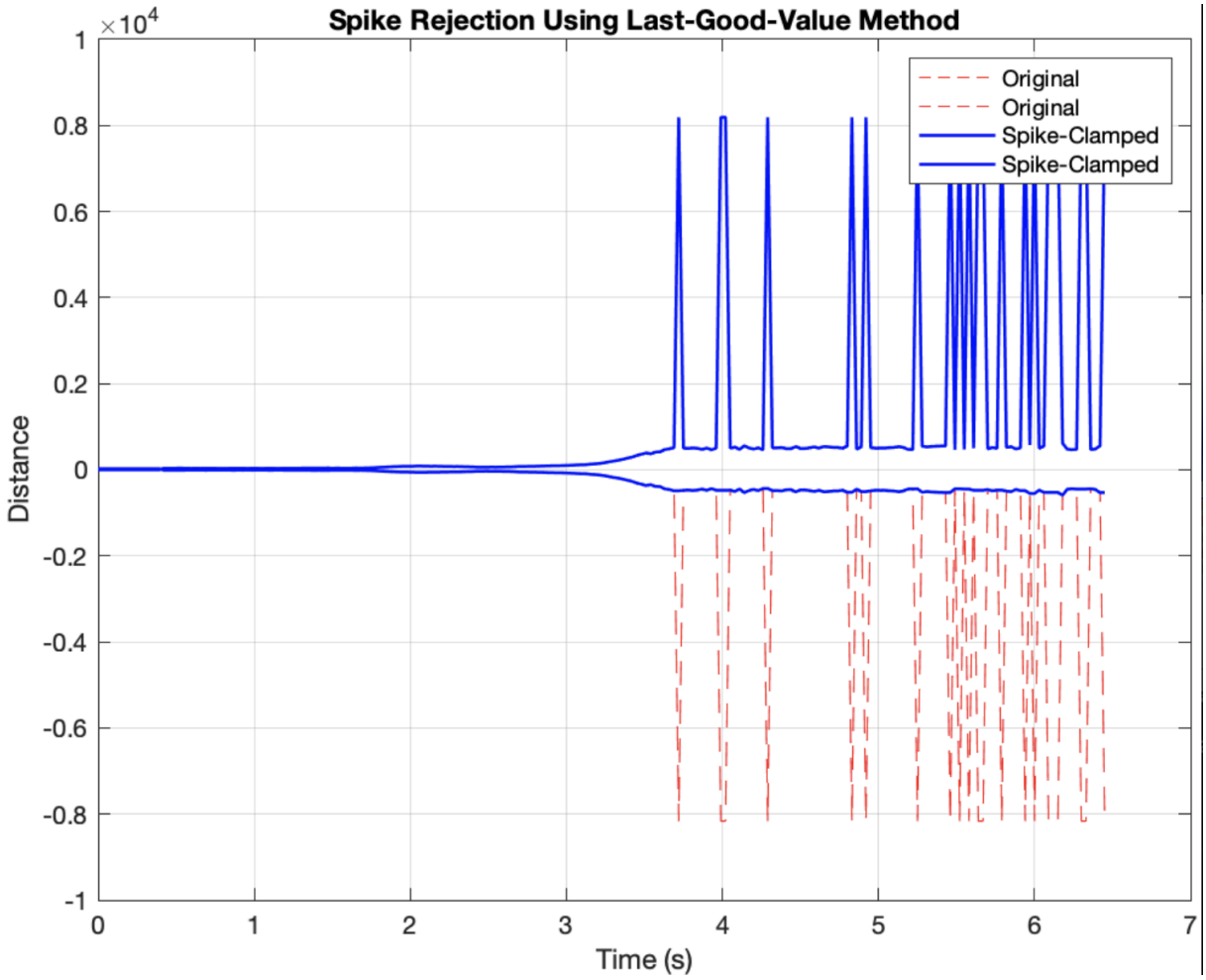

Figure 4: Plot showing original sensor data (top line) and filtered data using the spike clamp logic (bottom line).

(Note: Plot formatting is preliminary as it was not originally intended for the final report. However, the top blue line is the original data and the bottom blue line is the augmented data but upside down. Apologies.)

To further smooth the signal, we applied a discrete Finite Impulse Response (FIR) filter with coefficients `[0.2 0.2 0.2 0.2 0.2]`.

The filtered sensor readings were converted to centimeters by dividing by **8.25**, which was determined empirically to yield reasonable accuracy. We then computed the control error as the difference between the target position (**26 cm**) and the measured value. This error was input to a PID controller with the following tuned parameters:

- **Proportional Gain (Kp)** = 1.05
- **Integral Gain (Ki)** = 0.0095
- **Derivative Gain (Kd)** = 0.15

These values were selected to minimize overshoot, dampen oscillations, and allow the system to reach steady state quickly.

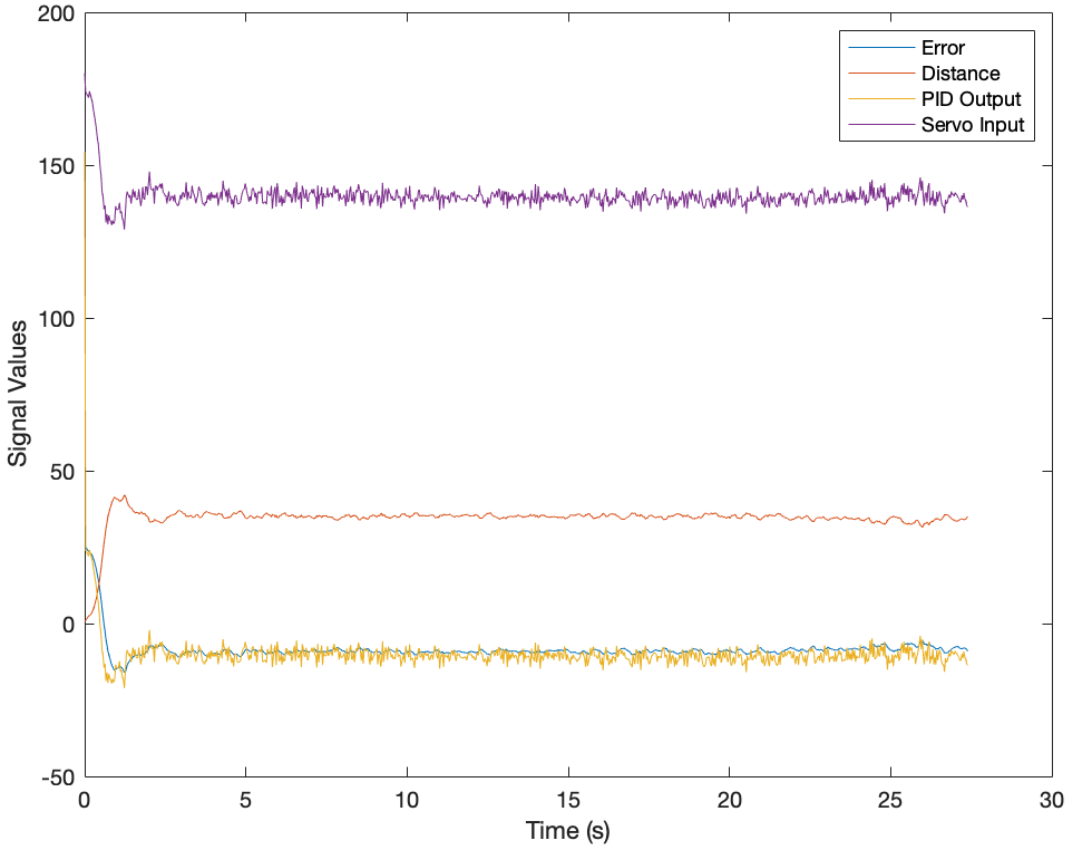

Figure 5: System response showing fast settling, low overshoot, and minimal oscillations.

The PID output was not sent directly to the servo, as the servo accepts input angles between 0° and 180°. We added a **+150 bias** to center the control range, then applied a saturation block to constrain the signal between **90** and **180** degrees before sending it to the servo motor. This mapping was found to produce the most consistent control over the beam's tilt.

Results and Discussion

After designing and implementing the system, we conducted multiple test runs to evaluate the controller's performance and robustness. The main performance criteria included system stability, response time, accuracy in tracking the desired setpoint, and ability to recover from disturbances.

System Behavior and Performance

- **Stability:**
  Once tuned, the PID controller was able to stabilize the ball at various setpoints along the beam. The ball exhibited minimal oscillation and consistently returned to the desired position after minor disturbances.

- **Response Time:**
  The average settling time across test cases was approximately 2–3 seconds, with quicker recovery observed when the setpoint was closer to the ball's initial position. The system demonstrated an adequate rise time without overshooting significantly.

- **Steady-State Error:**
  In most cases, the steady-state error was less than ±1 cm, which was acceptable given the mechanical constraints and sensor resolution. Some minor drift was observed due to ToF sensor noise and servo backlash.

- **Setpoint Tracking:**
  The system accurately tracked changes in setpoint, with smooth transitions and no noticeable instability. When step inputs were applied (manually), the controller adapted with a controlled response.

Controller Tuning and Iteration

Tuning the PID controller was one of the most critical and time-consuming aspects of the project. We began with manual tuning using heuristic methods and then adjusted gains based on observed behavior:

- Increasing proportional gain reduced rise time but introduced overshoot.

- Adding derivative control helped reduce oscillation but made the system sensitive to noise.

- Integral gain was kept low to avoid instability and wind-up.

Eventually, we achieved a balance that prioritized responsiveness and reliability over aggressive control.

Mechanical and Sensor Challenges

While the control logic functioned effectively in simulation, real-world implementation introduced imperfections:

- **Sensor Noise:** Analog readings had small fluctuations and spikes, which occasionally caused jitter in the control signal. We used a threshold detector and FIR Low-Pass filter to mitigate this.

- **Servo Backlash:** Slight mechanical slack in the servo's arm mechanism introduced lag and minor nonlinearity in beam positioning.

- **Friction and Surface Imperfections:** Variations in the ball surfaces caused the system to behave differently at certain positions. At some points, the ball would not roll despite the beam moving. To resolve this we had to buy a better ping pong ball. I can't say much about the difference between the old ball and the new one as it was not clear, but the new one rolled better during testing.

These issues emphasized the importance of accounting for non-idealities when deploying control systems in hardware.

Simulink Deployment and Observability

Using Simulink's External Mode proved invaluable during tuning. It allowed us to:

- Monitor the ball's position and control signal in real time.

- Adjust PID gains on the fly.

- Visualize performance using scopes and dashboard blocks.

Once optimized, we deployed the model to the Arduino to run in standalone mode. The autonomous controller maintained performance without the need for a host computer.

Summary of Observed Metrics

Metric	Result
Average Settling Time	~2–3 seconds
Steady-State Error	< ±1 cm
Overshoot	Low to moderate (tunable)
Setpoint Tracking	Smooth, with minimal delay
Power and Thermal Issues	None observed

Conclusion

In conclusion, this project successfully demonstrated the design and implementation of a real-time closed-loop control system capable of stabilizing an inherently unstable mechanical system a ball on a tilting beam.

By integrating Simulink-based PID control with Arduino hardware, we created an autonomous system that accurately tracked and maintained a target ball position in the presence of sensor noise, mechanical imperfections, and real-world disturbances. Through iterative tuning, sensor filtering, and thoughtful system integration, we achieved fast settling times, minimal steady-state error, and smooth setpoint tracking.

Beyond meeting the project's technical goals, the process provided invaluable hands-on experience with hardware-software co-design, control theory application, and practical problem-solving under physical constraints. The project not only validated key control concepts but also deepened our understanding of embedded systems and the challenges of bringing theoretical models into functioning reality.