Ball and Plate Stabilizing System

Rowan University

Mechanical Engineering

Dynamic Systems and Control

Dr. Hong Zhang

Team 4: James Lee, Ian Janaszik, Zander Marks, and Vanessa Hutchinson

1.0 Abstract

This project focused on designing, building, and controlling a ball-and-plate system capable of balancing a ball at various assigned positions on a flat plate. Unlike a ball-and-beam mechanism, which is a one-dimensional system, the ball-and-plate introduced two-dimensional dynamics requiring simultaneous control of both the x and y axes.

The primary objective was to ensure that the ball could be positioned at multiple assigned coordinates on the plate and remain within 0.5 inches of the target location in both directions. In addition, whenever the ball was intentionally disturbed, the system needed to return the ball to the correct position within 20 seconds and maintain it in that position for at least 15 seconds.

To achieve these goals, the project was divided into three major phases. First, a complete SolidWorks model of the system was developed. Most importantly, the completion of the design allowed the team to decide between using two or three servos to actuate the plate. This mechanical design dictated the layout and wiring of the system's control electronics. The second phase involved constructing the electrical circuit, which would integrate the actuators and the resistive touchscreen sensor together. The final phase focused on system tuning and controller implementation, ensuring the mechanical and electrical systems worked together to accurately stabilize the ball within the project description.

1.1 Project Overview

Choosing the ball-and-plate system introduced several new challenges compared to the simpler ball-and-beam mechanism. Due to the plate requiring controlled motion in two independent directions, the system needed two actuators and a full set of dynamic equations governing both

the x and y axes. The design with the two servo actuators was completed in SolidWorks and can be seen below in Figure 1. This CAD design allowed for the team to make sure that the baseplate that the two servos would attach to would have enough additional room to hold the battery pack, breadboard, and Arduino Uno Board. Additionally this design incorporated links from the servo to the plate, which would have the touch screen resting on it.

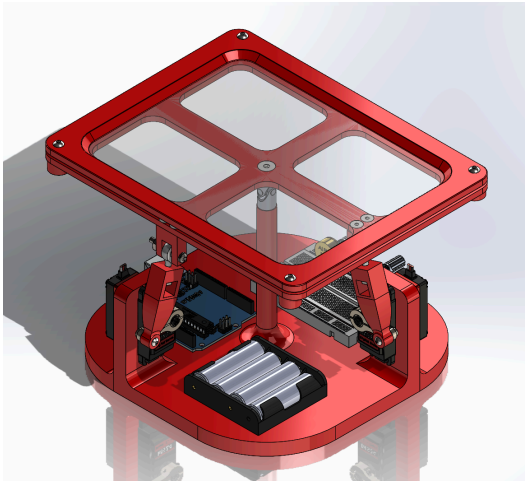


Figure 1: *CAD Design of Ball-and-Plate*

Once this design was complete the parts were printed and assembled. From there the arduino uno was then properly wired to the breadboard and battery pack to actuate the motors. Another important component that had to be integrated into the breadboard was two potentiometers that would allow the team to change the x and y coordinates of the desired position. A resistive touchscreen was incorporated to measure the real-time position of the ball.

After the system was properly wired the code for the project was written. The code uses the real time position subtracted from the desired position to calculate the position error, which then serves as the input of the PID controller algorithm. The controller would then adjust the servo angles to orient the plate in the desired position for the ball to return to the desired position. In

order to get the ball to stop at the desired position the PID had to be tuned for our system. For this we had to determine the proper proportional, integral, and derivative constants of the system for the x and y axes. If these values were not tuned properly, the plate ball would either overshoot off the plate or oscillate without ever stopping at the correct position.

Overall, the project required coordinated integration of mechanical design, electrical circuit construction, sensor interfacing, and control-system development to reliably balance the ball at multiple locations. Additionally, the system also had to recover from disturbances, and meet the specified accuracy and timing requirements.

2.0 System Architecture

The ball-and-plate system is a complex, two-dimensional dynamic system that consists of a combination of mechanical components, electrical hardware, sensing, and control software. The system is designed to balance a ball at various desired positions on a flat plate by simultaneously controlling both the x and y axes. The project was split into three different phases to achieve this: mechanical design, electrical construction, and control implementation and tuning.

2.1 System Components

To successfully achieve the desired goal, the system relied on specific hardware. The following table below details the key components used in the physical construction of the ball-and-plate system.

Table 1: *Ball-and-Plate Component List*

Subsystem	Component	Quantity
Mechanical	MS24 20KG Servo Motor	2
	60645K61 Ball Joint Rod End	2
	Universal Joint	1
	6-32 Screws	14
	6-32 Nuts	14
	3D Printed (PLA) System Base	1
	3D Printed (PLA) Servo Arm	2
	3D Printed (PLA) Touch Pad Base	1
	3D Printed (PLA) Touch Pad Top	1
	Battery Pack	1
	Steel Ball	1
Electrical	Half Breadboard	1
	Arduino Uno	1
	AA Battery	4
	Wires	20
Control	Resistive Touch Pad Senor	1
	B10K Potentiometer	2

2.1.A Mechanical Subsystem

The finalized CAD model created on SolidWorks, which can be seen in Figure 1, was manufactured using 3D printed PLA which produced the parts seen above in Table 1 . The base

served as the mounting platform for the two servo motors, the Arduino Uno, the breadboard half, and the battery pack. The base extended up to interface with the universal joint and the touch pad base. The touch pad base and top served as a casing for the resistive touch pad sensor, holding it level in place, while ultimately connecting to each servo motor through the 3D printed arms, a ball joint rod end, screws, and nuts.

2.1.B Electrical Subsystem

The electrical system integrates all the electrical components which enables the entire system to function. The electrical components include an Arduino Uno, half a breadboard, four AA batteries, two MS24 servo motors, and two B10K potentiometers. The wiring diagram can be seen in the figure below.

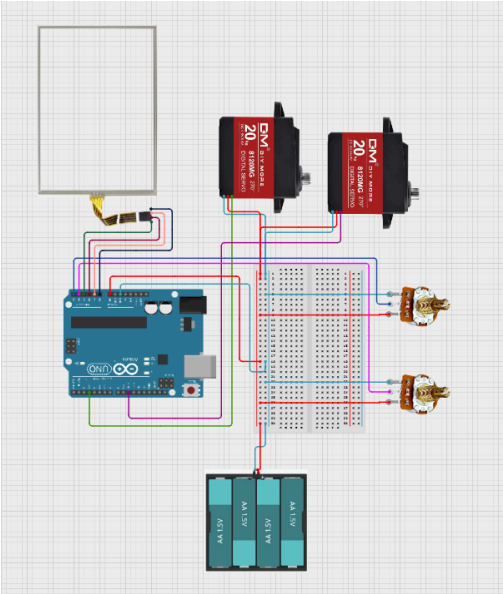


Figure 2: *Electrical Schematic Diagram*

Figure 2 displays the wiring of the circuit. The circuit is powered by a pack of four AA batteries which supplies the necessary current and 6V to power the two high-torque servos and the Arduino Uno. The battery pack also powers the rest of the components that are connected on the

common ground and power rail of the breadboard. The core of the electrical circuit is the Arduino Uno that receives and stores code from the computer. The analog pins, A0-A3, of the Arduino are connected to the resistive touch pad sensor, along with A4 and A5 being wired to each potentiometer respectively.

2.1.C Control Subsystem

The system's control logic is implemented as a feedback loop centered around a PID algorithm which is further explained from Figure 5 in section 3.2 of this report. The PID algorithm works with the resistive touch pad sensor reading the position of the steel ball when placed on top. This allows for a target point to be established to stabilize the ball. The potentiometers allow for the target point of the ball to be adjusted from 0 to 2 in both the x and y directions.

2.2 System Dynamics

The dynamics of the ball-and-plate system with the ball rolling on the tilted plate were modeled to predict the ball's position relative to the plate angle. The dynamic model is derived by applying Newton's Second Law to the ball assuming there is no slip, the steel ball is a uniform solid sphere [$I=0.4mr^2$], air resistance is ignored, and the plate will not reach high angles, therefore $\sin\theta=\theta$ and $\cos\theta=1$. Below Figure 3 shows the free body diagram (FBD) of the ball on a plate. The FBD is modeled along the x-axis and would look the same along the y-axis.

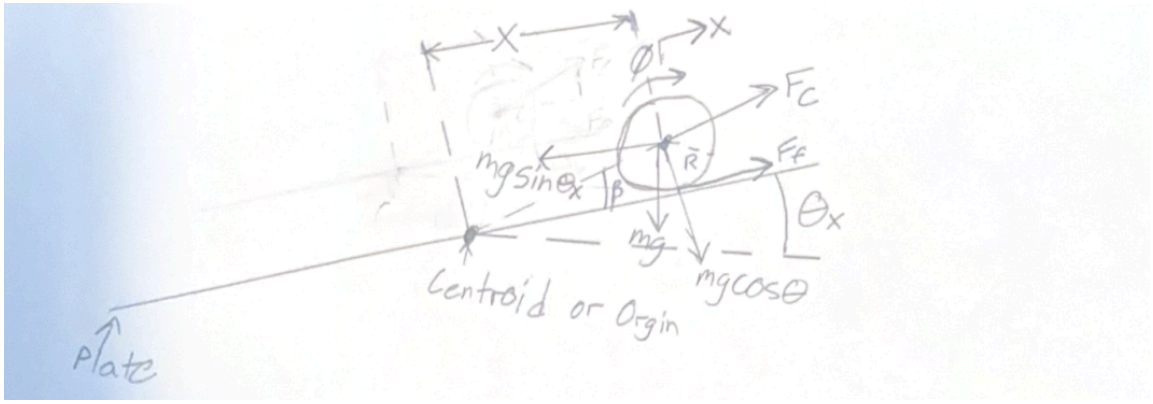


Figure 3: *Free Body Diagram of the Ball on a Tilted Plate*

The governing equation of motion for the ball is derived by inspecting the forces and moments acting on the ball. The sum of all forces in the x direction is:

$$\Sigma F_x = mx'' = F_f + F_{cx} - mg\sin\theta_x.$$

From there, F_f and F_{cx} were solved for and substituted back into the equation for ΣF_x. The calculations for the governing equation can be seen in Figure 4 below.

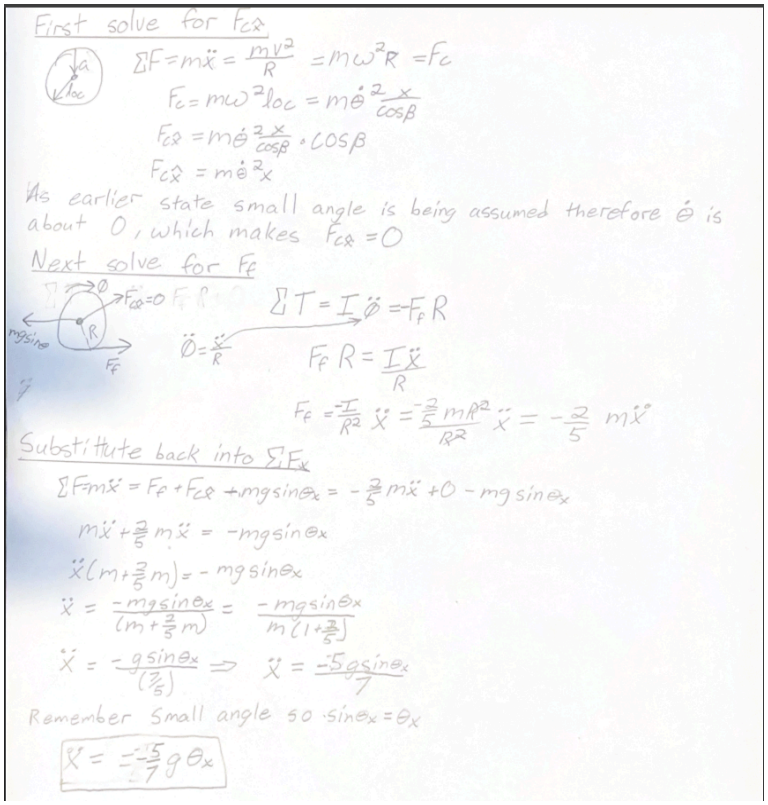


Figure 4: *Governing Equation Calculations*

Most importantly, it was determined that $F_{cx} = 0$, and $F_f = -(⅖)mx''$. The governing equations were determined to be $x'' = -(5/7)g\theta_x$ in the x direction and $y'' = -(5/7)g\theta_y$ in the y direction.

3.0 Control System Design

The control system for the ball-and-plate project was designed to integrate both hardware and software elements into a unified feedback loop capable of stabilizing the ball in two dimensions. At its core, the system employed a PID controller that continuously compared the desired ball position, set by user input through potentiometers, with the actual position measured by the resistive touchscreen sensor. The resulting error signal was processed by the PID algorithm, which generated corrective commands to the two servo motors responsible for tilting the plate along the X and Y axes. To ensure realistic behavior, the Simulink model incorporated saturation limits reflecting the mechanical constraints of the servos, as well as governing equations that translated plate tilt into ball acceleration and position. This architecture allowed the team to simulate, tune, and implement a control strategy that reliably balanced the ball at designated coordinates while rejecting disturbances.

3.1 Control Strategy

After constructing the physical system, the next step in the process was to determine how the hardware would be implemented to actually balance the ball. This was accomplished through a PID feedback algorithm. First, a control block diagram was created as a digital prototype to explain how the system was expected to behave. Then, an arduino script was created that allowed the user to tune the P, I, and D values to match the expected behavior of the system.

3.2 Control Block Diagram

To simulate the ball and plate control system, Simulink was utilized. The model takes a given input position (in both X and Y coordinates), then uses a feedback control loop that incorporates our PID values, a saturation function to account for the mechanical limit that the servos are allowed to rotate, and a subsystem block that incorporates our governing equations and initial position. The overall control block diagram is captured below in Figure 5.

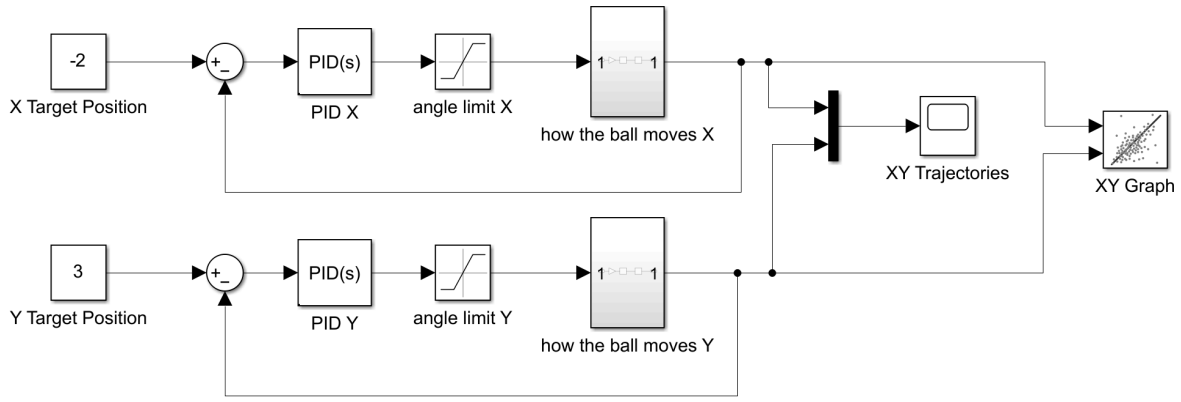


Figure 5: *Simulink Control Block Diagram of Ball and Plate PID Control System*

The PID Control blocks, which were present in both the X and Y branches of the Simulink model, capture the tuning behavior which we want the plate to control the ball with. Each block was set to "Transfer Function Based (PID Tuner App)." This accurately reflects the code, as P, I, and D constants were implemented. The values of P, I, and D within our system, as shown in Figure U, are the same ones used in the physical model and code. This was done in an attempt to make the Simulink model behave as closely to the real model as possible. Something else that should be noted is that the P, I, and D values for the blocks within the X and Y branches were both the same. This was an intentional simplification placed upon the system in an attempt to

make things easier to tune (as independently controlling and tuning the ball's behavior on the X and Y axes independently introduced further complexities that the team did not consider worth the effort of pursuing).

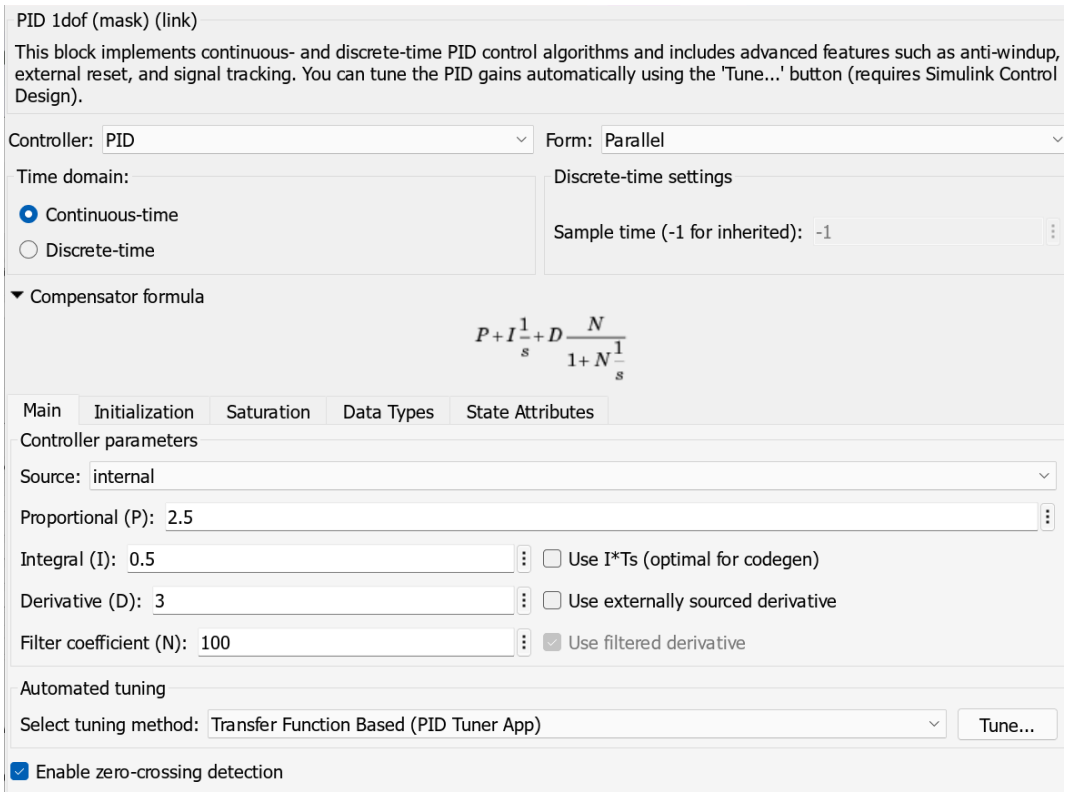


Figure 6: *PID Control Block*

The next block in the system is a saturation block, used to control the amount that the servo can travel within each axis. It places boundaries on the amount that each axis is able to travel at a given time. Through both research (and then physical testing later on), it was decided that 10 degrees was a safe boundary to set for the maximum rotation of the servos. This value, captured in radians, can be seen in Figure 7 below.

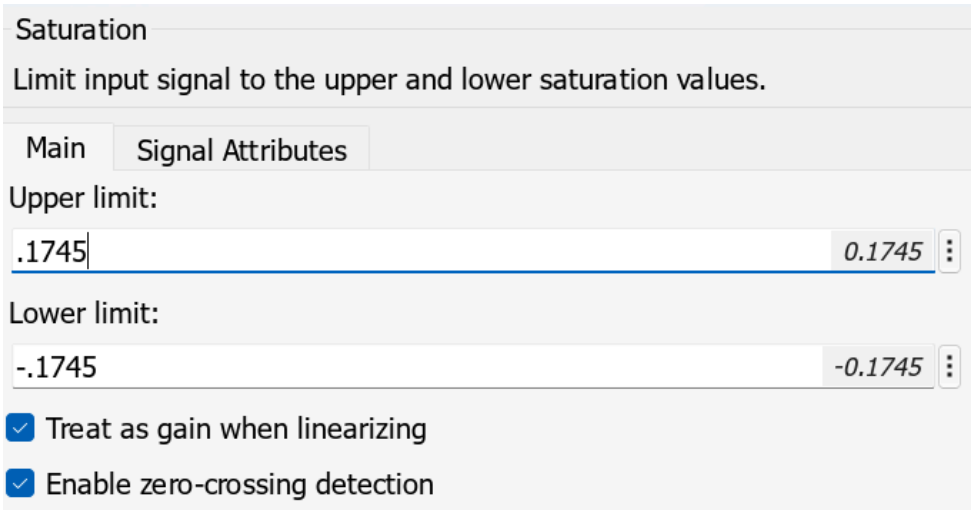


Figure 7: *Upper and Lower Limits of Servo Rotation Inside the System*

Next, the subsystem block can be broken down as follows: The governing equation utilized by the simulation was (-5/7)*g (acceleration of gravity). Then, two integrating blocks are able to convert the acceleration of the ball into velocity, and then into position. The position is required to analyze where the ball is in a given time, relative to the target value. Inside the integrating block, an initial condition was used that represents where the ball was initially "placed" at the beginning of the simulation. The innards of the subsystem block are demonstrated in Figure Y. What must be noted is that these subsystem blocks are the same for both the X and Y branches. All that differs between them is the initial condition that was arbitrarily decided for each.

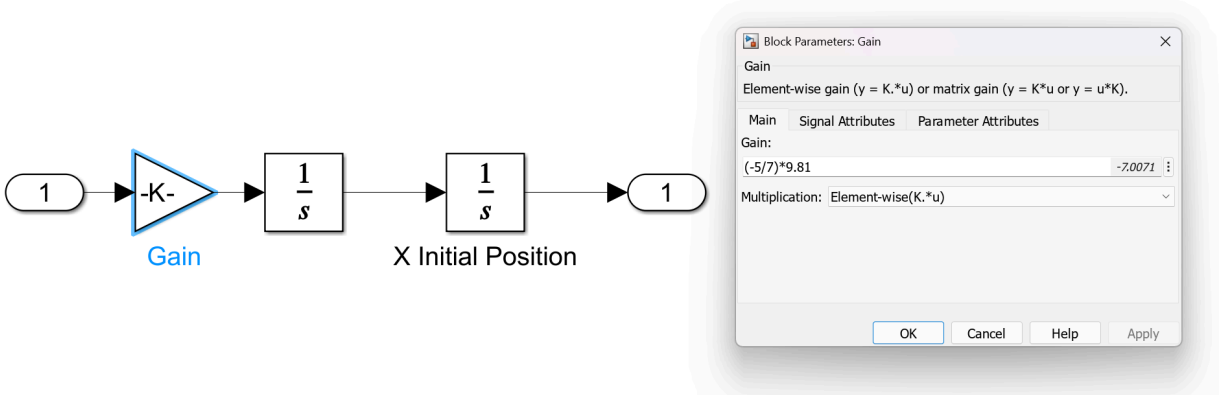


Figure 8: *Subsystem Block for Governing Equation and Initial Position*

To show the position of the ball in the X and Y directions over time (relative to both the initial and target positions), two different figures were created. First, a scope block was created to visualize the X and Y positions of the ball over time. This is captured in Figure Z. In the Figure, the X position is illustrated with the yellow line, and the Y position is illustrated in the blue line. Because our simulation used the same PID values as our actual model, this graph gave the team a general idea of the amount of time that the ball would need to stabilize on each independent axis.

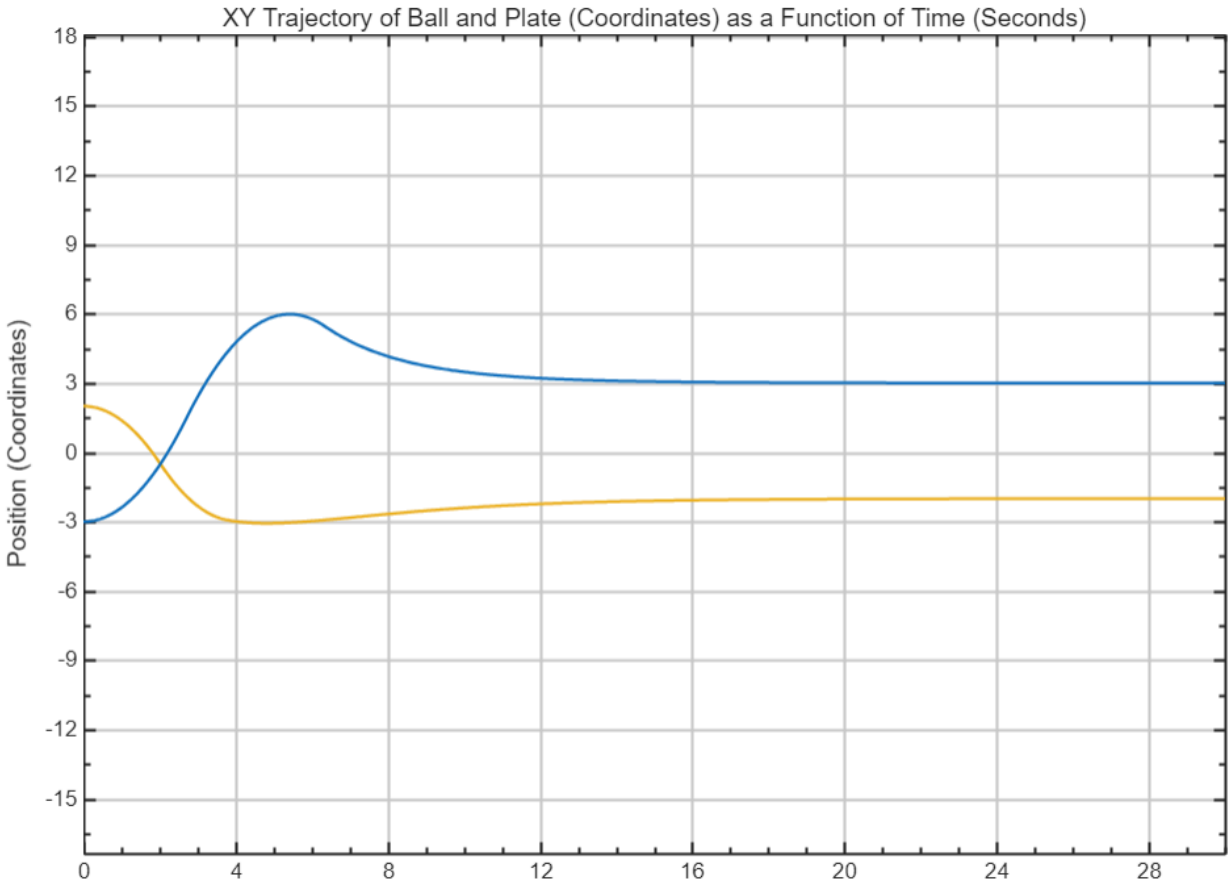


***Figure 9**: XY Trajectories of Ball and Plate (Coordinates) as a Function of Time (Seconds)*

As a supplementary piece of information, an XY graph block was also used. This block creates a graph that creates a mark where the ball is located at given time stamps. What results is a "top down" view of the ball over time. This helped give the team a general sense of how the ball should behave on the plate when physically constructed. The ball's oscillations are captured here

as well, demonstrated through the “cluster” of points at the top left of the graph. The results of the XY graph block can be seen in Figure 10.

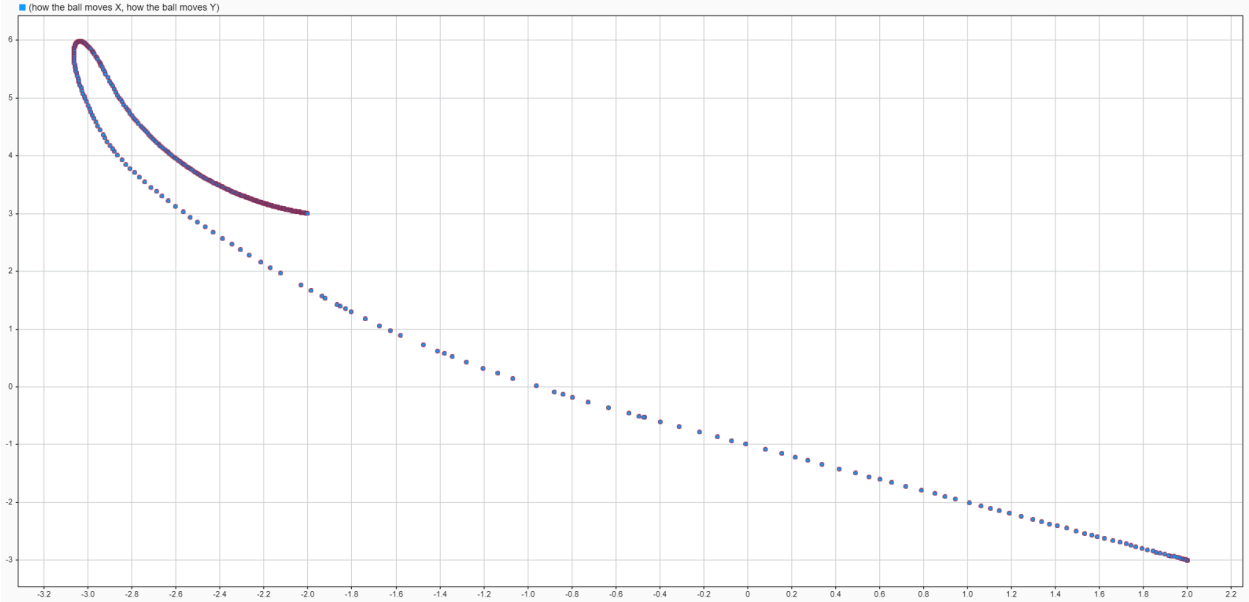


Figure 10: *XY Position of Ball and Plate System over Time*

3.3 PID Algorithm

The control law implemented for the ball-and-plate system was based on the classical Proportional–Integral–Derivative (PID) algorithm. The PID controller continuously calculates the error signal as the difference between the desired ball position (set by the potentiometers) and the measured ball position (from the resistive touchscreen sensor). The controller then applies corrective action to the servo motors according to the following relationship:

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

where:

- $e(t)$= position error (target – measured)

- K_p = proportional gain, which scales the immediate error
- K_i = integral gain, which accumulates past error to eliminate steady-state offset
- K_d = derivative gain, which predicts future error by responding to the rate of change

In practice, the proportional term provided the primary corrective action, the integral term ensured the ball settled exactly at the target position, and the derivative term reduced oscillations by damping rapid changes in error. The PID algorithm was implemented identically for both the X and Y axes, allowing the system to simultaneously stabilize the ball in two dimensions.

3.4 PID Tuning & Gains

Tuning the PID controller was a critical step in achieving stable performance. The process was iterative, beginning with the proportional term to establish basic responsiveness, followed by the integral term to correct steady-state error, and finally the derivative term to reduce overshoot and oscillations.

- **Proportional Gain (P):** Initially set low to prevent instability, then gradually increased until the ball responded quickly to changes in target position.
- **Integral Gain (I):** Introduced after proportional tuning to eliminate small offsets. Care was taken to avoid excessive integral action, which could cause slow oscillations.
- **Derivative Gain (D):** Adjusted last to provide damping. This term was particularly important in the ball-and-plate system, as the two-dimensional dynamics made the ball prone to overshoot.

The final tuned values used in both the simulation and physical system were:

- $K_p = 2.5$

- $K_i = 0.5$
- $K_d = 3.0$

These values were chosen based on the Simulink PID Tuner app and validated experimentally. The filter coefficient N=100 was applied to the derivative term to reduce noise sensitivity.

During testing, these gains allowed the ball to reach steady state within 5–10 seconds depending on the target position, with maximum steady-state errors under 0.4 inches. Disturbance rejection was also effective, with recovery times consistently under 9 seconds.

4.0 Experimental Results

The experimental phase of the project validated the ball-and-plate system's ability to stabilize the steel ball at designated target positions under both nominal and disturbed conditions. Following the completion of the mechanical assembly, electrical wiring, and PID tuning, the team conducted a series of demonstrations to evaluate the accuracy, response time, and robustness of the control system. The results were compared against the performance requirements outlined in the project description, which specified that the ball must reach the designated position within 20 seconds, remain within ±0.25 inches of the target for at least 15 seconds, and recover from disturbances of at least 1 inch within 20 seconds.

The experiments were carried out using three distinct target positions on the plate: (0,0), (1,–1), and (–1,1). At each position, the system's steady-state accuracy, time to stabilization, and

disturbance rejection were recorded. The results demonstrated that the system consistently met the required performance criteria, with minor steady-state errors and acceptable recovery times.

4.1 Experimental Setup

The physical setup consisted of the following components:

- **Actuation system:** Two servo motors mounted beneath the plate, each responsible for tilting the plate along one axis (X and Y).
- **Input control:** Two rotary potentiometers connected to the Arduino Uno, allowing the user to set the desired X and Y coordinates of the ball.
- **Sensing system:** A resistive touchscreen mounted on the plate surface, providing real-time feedback of the ball's position in both axes.
- **Controller:** An Arduino Uno programmed with a PID feedback algorithm. The error signal was computed as the difference between the desired and measured ball positions, and the PID controller adjusted the servo angles accordingly.
- **Power and wiring:** A breadboard and battery pack supplied power to the servos and microcontroller, while jumper wires connected the potentiometers and touchscreen sensor.

During testing, the ball was initially placed at the center of the plate. The potentiometers were then adjusted to set the target coordinates. A stopwatch was used to measure the time required for the ball to reach steady state. Once stabilized, disturbances were introduced by manually displacing the ball by 2–4 inches, after which the system's recovery was observed and recorded.

4.2 Stabilizing Performance

The system's performance was evaluated at three designated positions. Table 2 summarizes the steady-state accuracy, time to stabilization, and disturbance rejection results.

Table 2: *Experimental Results of Ball-and-Plate Stabilization*

Target Position	Steady-State Ball Position (in)	Max Error (in)	Time to Steady State (s)	Disturbance Applied (in)	Recovered Position (in)	Max Error After Disturbance (in)	Recovery Time (s)	Duration at Steady State (s)
(0,0)	(0.24, 0.25)	(0.24, 0.25)	10	2	(1.16, 1.17)	(0.16, 0.17)	4.77	15
(1,–1)	(–1.07, 1.39)	(0.07, 0.39)	5	4	(0.79, –1.33)	(0.21, 0.33)	8.95	15
(–1,1)	(–1.31, 1.16)	(0.31, 0.16)	5	3	(–1.38, 0.83)	(0.38, 0.17)	7.2	15

At all three target positions, the ball reached steady state within 10 seconds, well below the 20-second requirement. The maximum steady-state error across all trials was less than 0.39 inches, satisfying the ±0.5-inch tolerance. Disturbance rejection was effective: the ball consistently returned to the designated position within 9 seconds after being displaced by 2–4 inches. The system maintained stability at each target position for the required 15 seconds, demonstrating robustness against small oscillations and noise.

Overall, the ball-and-plate system successfully met the project's performance criteria. The PID controller provided reliable stabilization in both axes, and the integration of the resistive touchscreen sensor ensured accurate position feedback. Minor steady-state errors were observed, primarily due to mechanical tolerances in the servo linkages and nonlinearities in the sensor readings. Nonetheless, the system demonstrated strong disturbance rejection and consistent accuracy, validating both the Simulink model and the physical implementation