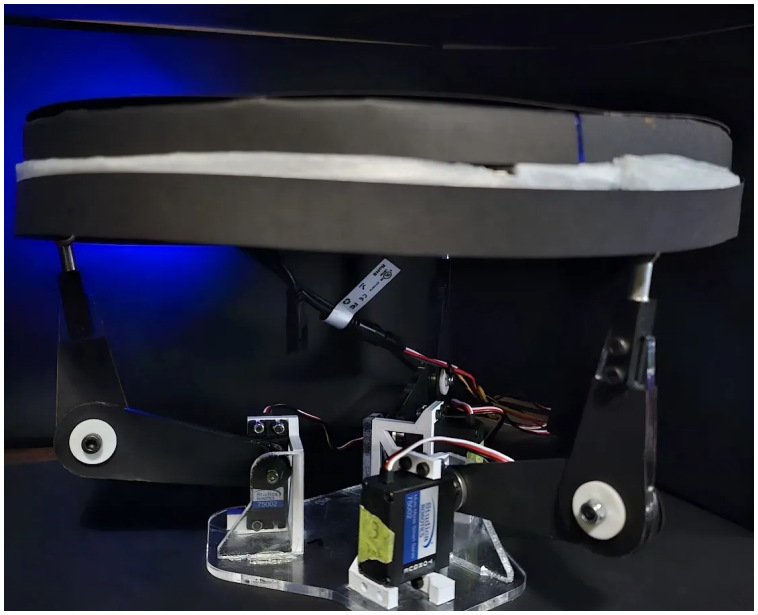
Design and Construction of a 3-DOF Stewart Platform for Ball Balancing

MTE 380 Final Design Report



Prepared for

Dr. Sanjeev Bedi
Department of Mechanical and Mechatronics Engineering,
University of Waterloo

Team 5: Richard Ding, Armaan Sengupta, Esha Haque, Jessica Liu, Jacob Lin

December 2, 2025

Acknowledgements

We want to thank Paul and the teaching assistants for their assistance during fabrication and testing. Their expertise enabled us to quickly iterate effectively under tight development constraints.

We gratefully acknowledge ForceN for providing access to their force–torque sensor, which provided the team with an great opportunity to deepen and diversify our learning through exploring alternative sensing methods.

Preface

This final report builds upon the earlier design proposal submitted at the beginning of the Stewart Platform project. Several sections—most notably the problem statements, constraints and criteria, and preliminary conceptual methods—are adapted and expanded from that initial submission in order to maintain continuity in system definition and design intent. As the project progressed, these foundational elements were refined through analytical modelling, simulation, and experimental validation.

The material presented here therefore represents both a continuation and a consolidation of the work initiated in the proposal stage, with all content updated to reflect the final design outcomes, empirical findings, and lessons learned throughout the term.

Furthermore, the team initially set out to answer question (1) and (6) of the suggested list of questions, which are trajectory planning comparison and removal of ball joint, in addition to the investigation of a new loadcell based sensing method. However, as the term progressed and the scope drastically increased beyond what was reasonably foreseen at the beginning of the project, it is determined that in the best interest of maximizing learning opportunity for the group members and ensure a fully functional presentation of the alternative sensing method, changes need to be made on the team's approach to the questions. Furthermore, increasing the ball mass to accommodate for the loadcell sensing significantly complicated the empirical implementation of trajectory planning, as the system's stability margin decreases due to an increase in the ball's inertia coupled with the slower motor dynamics (due to a much higher weight of the platform). Therefore, the following decisions were made:

1. The trajectory planning comparison would be limited to a Simulink implementation.
2. To replace question (1), the team will alternatively investigate question (10), modifying the design of the platform for a 100x heavier ball, which was done as part of the sensor replacement.

The rest report will be written in acknowledge to the above modifications made to the project objectives.

Contents

1 Introduction

The Stewart platform is a parallel manipulator architecture that provides six degrees of freedom (6-DoF) motion through six actuated legs connecting a fixed base to a moving platform [1]. The close-loop structure with good load capacity, precise positioning and high stiffness makes it it a benchmark mechanism in flight simulation, vibration isolation, and precision machining applications [2].

In the context of mechatronics engineering, the Stewart platform represents an attractive yet challenging benchmark due to its nonlinear kinematics, multi-axis coupling, and control complexity. The MTE 380 course adopts a scaled down version of this architecture, which is a 3-DoF platform designed to balance a rolling ping-pong ball. [3]. Some of the topics explored in this proeject includes forward and inverse kinematics, actuator coordination, sensing, control and feedback, and iterative electromechanical design (essentially the engineering design cycle).

Within this architecture, the present project focuses on two primary goals. The first (Question 6) examines the mechanical role of the spherical ball joints that interface the linkages with the top plate. These joints provide the rotational compliance necessary to prevent over-constraint and kinematic binding. A central objective is to characterize these functions and to propose a feasible redesign that eliminates the use of ball joints while preserving the required mobility, stiffness, and manufacturability of the system.

The second investigation (Question 10) arises naturally from the development of a loadcell–based sensing system. Because force and moment measurements require a significantly higher normal force to produce a usable signal, the platform must operate with a ball roughly one hundred times heavier than the standard ping-pong ball that's currently being tracked with the camera. This new ball mass therefore introduces substantial mechanical implications: increased loading on the linkages, higher moments at the joints, and greater torque demands on the the servos. Question 10 addresses how the Stewart platform must be mechanically redesigned, which includes linkage geometry, joint selection, and electrical redesign to operate reliably under these elevated loads. Structural analysis, simulation, and empirical evaluation of component feasibility form the basis of this redesign.

In parallel, trajectory planning strategies were explored through a dedicated Simulink study, using the reduced-order plant model to compare step based, time-parameterized, and adaptive trajectory generation methods. This analysis provides insight into control behavior and closed-loop performance but remains simulation only, as empirical implementation was not pursued on the heavier ball platform due to the altered dynamics and stability constraints introduced by the loadcell sensing architecture.

Together, these investigations link mechanical design, structural robustness, actuator capability, and control system behaviour. They provide a focused examination of how changing sensing requirements and load conditions reshape the feasibility and performance envelope of a reduced-DoF Stewart platform.

2 Redesign for Heavier Ball and Alternative Sensing Method

2.1 Motivation and Requirements

2.1.1 Limitations of the Default Camera-Based Sensing

The standard course provided Stewart platform relies on an overhead camera system to provide the ball's planar position. The camera offers an immediate and absolute measurement of (x, y), with minimal signal processing and negligible noise once calibrated. Calibration is however a manual process requiring re-mapping the camera pixels to the physical location on the 2-D plane each time the system is moved. While effective, this sensing method requires manual human input and a separate stand to mount the camera. In addition the sensing method does not require any substantial mechanical or algorithmic innovation beyond what was provided in class. Because the camera directly returns position, the control loop bypasses the problem of estimating ball state from physical interaction forces, leaving limited opportunity to explore more advanced sensing methodologies.

2.1.2 Motivation for a Load-cell-Based Alternative

To create a more challenging design problem and eliminate manual calibration, the team introduced an alternative sensing architecture based on a 3 axis force torque sensor (referred to from here on out for brevity as a loadcell) mounted beneath the plate. Unlike camera vision, the loadcell does not provide direct position information. Instead, it measures the distribution of normal force and tipping moments applied by the ball onto the platform, from which the in-plane position must be inferred. This sensing strategy requires consideration of platform stiffness, structural deformation, and the mapping from measured forces and moments to spatial coordinates. It also provides a richer integration of mechanical design, integration, calibration and controller integration.

Furthermore, a loadcell based approach aligns more closely with the principles of force and moment sensing used in robotic manipulation. Implementing this method introduces practical engineering challenges such as managing sensor noise, ensuring that forces are transmitted cleanly through the load path, accounting for structural compliance in the platform, and obtaining signals with a sufficiently high signal-to-noise ratio. These factors collectively motivate the introduction of a three-axis loadcell as a means to deepen the mechanical and controls focused aspects of the project beyond what the default camera based system provides.

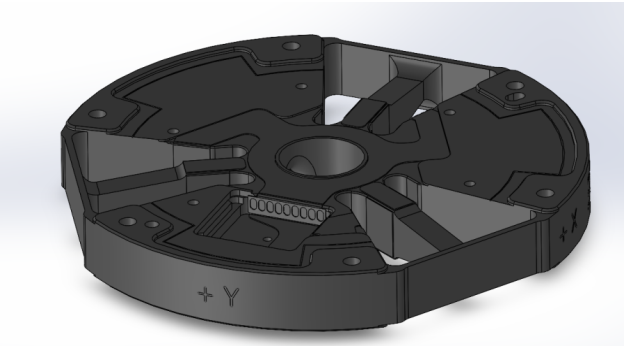


Figure 1: CAD of the proposed loadcell

2.1.3 Need for a Heavier Ball to Enable Loadcell Sensing

While the loadcell sensing method is conceptually straightforward, its practical performance is strongly dependent on the magnitude of the forces it experiences. A standard 2.7 g ping-pong ball produces normal forces on the order of only a few millinewtons and correspondingly an negligible pitch/roll moment (refereed from here on out collectively as tipping moment). These values lie far below the useful resolution of typical compact three-axis loadcell, where electrical noise, quantization error, platform vibration, and mechanical tolerances dominate the signal. As a result, the inferred (x, y) position becomes unreliable.

To obtain force and moment readings that exceed the noise floor and are sufficiently repeatable for closed-loop control, the ball mass must be increased substantially. A much heavier ball is therefore required for the loadcell to operate within a usable force range; the specific mass selection and its mechanical implications are detailed in the following sections.

2.1.4 Mechanical Implications of the Heavier Ball

The substantially heavier steel ball introduces forces and moments that are roughly two orders of magnitude larger than those generated by the original ping-pong ball. These increased loads affect every major subsystem: the linkages and joints experience higher axial, shear, and tipping-induced reaction forces; the plate must remain sufficiently stiff to avoid deflection that would corrupt loadcell measurements; and the actuators must supply greater torque to achieve comparable tilt dynamics.

The larger ball inertia also reduces the system's effective stability margin, since greater momentum must be dissipated during corrective motions. As a result, the redesigned platform must accommodate higher mechanical loads, maintain a rigid load path for accurate force–moment sensing, and operate within actuator limits that are substantially more demanding.

2.1.5 Resulting Design Requirements

These considerations lead to the following design requirements for the redesigned platform:

1. The linkage geometry and material choice must support the increased static and dynamic loads with an adequate safety factor.
2. Joints must withstand larger reaction moments without binding, fracturing, or introducing backlash.
3. Actuators must provide sufficient torque and speed under elevated loading conditions, which may require higher-torque servos or alternative motor technologies.
4. The mechanical interface between the top plate and the loadcell must remain rigid so that measured forces and moments map accurately to in-plane position.
5. The top plate must achieve a high stiffness-to-weight ratio: excessive mass overloads the linkages, whereas insufficient stiffness produces unacceptable deflection.
6. Structural deformation throughout the platform must be minimized to preserve sensing accuracy and predictable control response.
7. The linkages, joints, and actuators together must support the increased overall platform mass resulting from the heavier ball and strengthened structure.

2.2 Impact of Increased Mass on Platform Mechanics

The introduction of the 538 g, 2-inch 440C stainless-steel ball substantially alters the mechanical operation of the Stewart platform when compared to the original 2.7 g ping-pong ball. For one,

the new ball produces forces and moments that are two orders of magnitude larger, since the mass ratio between the steel ball and the original ping-pong ball is

$$m_{\text{pp}} \approx 2.7 \text{ g}, \tag{1}$$

$$m_{\text{steel}} \approx 538 \text{ g}, \tag{2}$$

$$\frac{m_{\text{steel}}}{m_{\text{pp}}} \approx \frac{538}{2.7} \approx 2.0 \times 10^2. \tag{3}$$

Because the normal force scales as $F = mg$, the corresponding contact forces and resulting tipping moments are also approximately two orders of magnitude large, especially when the ball is off center and near the edge of the platform. These forces and moments must be transferred through the linkages, joints, servo horns, and base structure. In practice, these loads exceeded the capabilities of many components in the original platform given, leading to repeated mechanical failures and necessitating several rapid iterations of the platform and linkage geometry.

2.2.1 Static and Dynamic Load Increases

The substantially heavier steel ball increases the static and dynamic loads on all platform components. The top plate experiences larger bending stresses and out-of-plane deflection, while the linkages carry significantly higher axial and shear forces arising from the increased normal load and from the lateral forces generated as the ball moves across the surface.

During initial testing, these elevated loads exceeded the capacity of the original acrylic linkage design. In the first platform iteration, the three-layer plate increased the overall mass of the assembly, and the resulting reaction forces caused cracking at the Link–1 to Link–2 joint, as shown in the following figure.

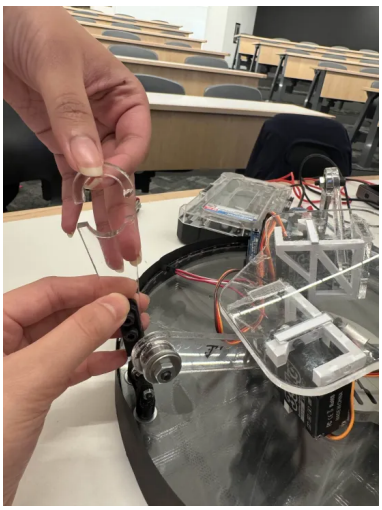


Figure 2: Fractured linkage joint from the first platform iteration.

Inspection showed that the failure occurred at the rotational joint between the two links. When the servos attempted to tilt the heavy platform, the heavy ball resisted the motion and generated a large lateral reaction force through the linkage chain. This force was carried primarily by the Link–1 to Link–2 joint, with the link oriented most perpendicular to the direction of tilt experiencing the highest shear load. The resulting lateral shear caused cracks to develop and propagate at the

thinnest acrylic section surrounding the bearing, leading to fracture. These observations motivated the redesign of the linkage geometry and reduction of overall platform mass.

2.2.2 Tipping Moments and Stability Margin

The heavier ball also generates substantially larger roll and pitch moments whenever it moves away from the platform center. These tipping moments must be reacted through the servo horns and structural interfaces, and even small compliance in these elements results in measurable platform misalignment and corrupted loadcell readings.

In addition, the larger mass reduces the system's stability margin. Because the ball carries greater momentum during lateral motion, larger corrective tilts are required to bring it to rest, especially near the platform edge where the restoring moment is minimal. Under these conditions, the original MG995 servos did not provide sufficient torque or speed to maintain stable operation, motivating the actuator upgrades described in Section 2.5.

2.2.3 Plate Stiffness and Deflection Constraints

For loadcell-based sensing to operate correctly, the top plate must behave as a rigid body relative to the sensor. Any significant deflection under the heavier ball alters the measured moments (M_x, M_y) and normal force F_z, which leads to systematic position estimation errors and introduces additional instability into the control loop.

The heavier ball therefore imposes a strict requirement on the plate's stiffness-to-weight ratio: the plate must remain sufficiently rigid under worst case loading while not adding excessive mass to the linkage assembly.

2.2.4 Linkage Stress, Shear Failure, and Mechanical Slop

The heavier ball also exposed several limitations in the original acrylic linkage assembly. The thin regions surrounding the bearings were susceptible to shear and lateral deformation, and the increased inertial forces during motion produced noticeable relative twisting between Link 1 and Link 2. This manifested as mechanical slop throughout the linkage chain and reduced the precision of the platform.

2.2.5 Summary of Mechanical Impact

The increased ball mass introduces substantially higher static and dynamic loads, larger tipping moments, and greater sensitivity to compliance within the linkage chain. These effects collectively necessitate a lighter yet stiffer top plate, strengthened linkage geometry, and actuators capable of providing higher torque.

2.3 Redesigned Top Plate Architecture

This subsection summarizes the three major plate concepts explored, the failure modes observed, and the rationale for selecting the final geometry.

2.3.1 Iteration 1: Three-Layer Acrylic Plate

The first design placed the loadcell between a bottom acrylic plate and a stack of three concentric 300 mm acrylic plates on top (each 3 mm thick, bonded with UV resin). Acrylic was selected

because it was laser-cuttable, and a thickness of 9 mm was chosen as a practical target for plate rigidity.

Although the triple-layer plate achieved sufficient stiffness, the combined mass of the three acrylic layers and the steel ball exceeded the load capacity of the linkage system. During initial assembly, the increased axial and bending forces caused the original linkages to shear and fracture. Root-cause analysis showed that the platform mass itself, rather than servo torque limitations was the dominant failure mode.

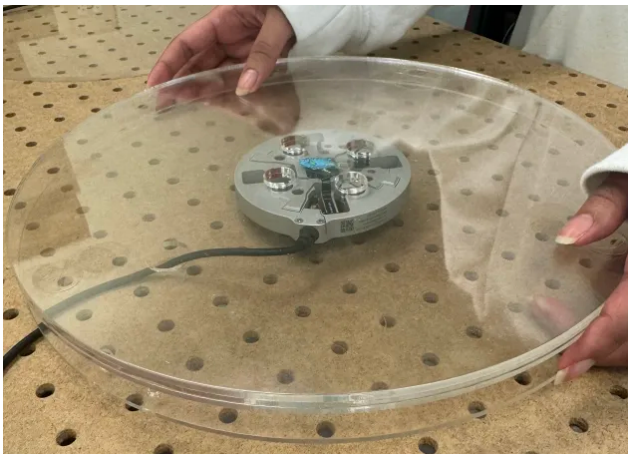


Figure 3: First iteration: triple-layer acrylic plate

This failure demonstrated that plate stiffness alone was insufficient; weight reduction was equally important to prevent linkage failure and increase servo response, as the current plate results in a massive mass the servo needs to move.

2.3.2 Iteration 2: Single Layer Plate Concept

The next concept attempted to remove mass by transitioning from a three plate assembly to a single 3 mm plate (with 3 countersink hole in the center to fix it to the sensor), which essentially mirrors the provided setup. While this design reduced weight significantly, testing showed that the plate deflected excessively under the 538 g ball. When the ball was placed near the perimeter, the top plate bent downward enough to contact the lower plate, exceeding the height of the loadcell and making the force/moment readings unusable.

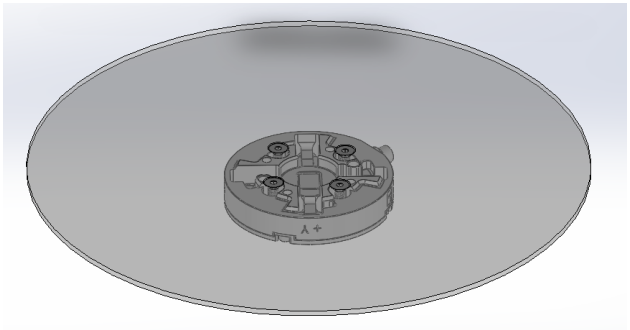


Figure 4: Second iteration: single-layer plate concept

This iteration again confirms the requirement: the top plate must be sufficiently stiff to prevent large deflections under worst case loading (when the ball is at edge of the plate), while remaining light enough to avoid overloading the linkages and increase servo responsiveness.

2.3.3 Iteration 3: Mass Optimized Plate (Cutout + Topology Optimization)

To balance rigidity and mass, two independent design strategies were pursued.

Standard Cutout Based Design This straightforward design introduced a symmetric pattern of rounded cutouts arranged in two concentric rings. The outer ring consists of twelve large curved pockets distributed evenly around the circumference, while a second inner ring of similarly shaped pockets provides additional mass removal closer to the plate center. This geometry preserves continuous material spokes across both the radial and circumferential directions, allowing load to be distributed efficiently while significantly reducing overall weight. The large perimeter pockets reduce bending stiffness where deflection is less critical, and the smaller interior pockets remove mass near the center without compromising structural connectivity to the load paths.

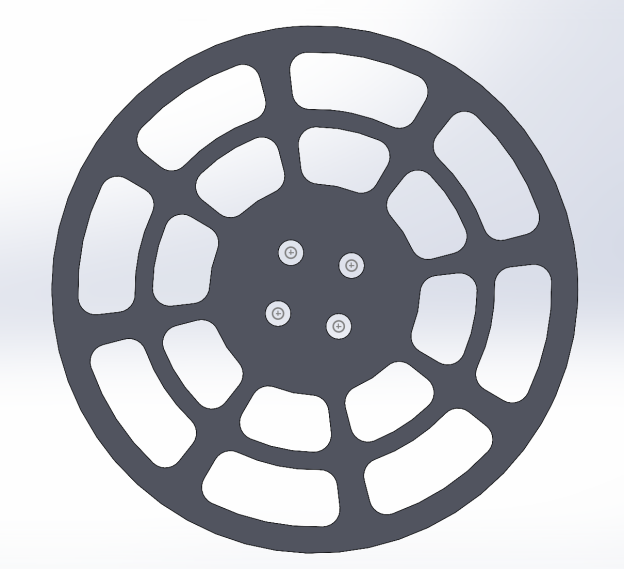


Figure 5: Concentric ring design with cutouts

Topology-Optimized Design In parallel, the team generated a plate geometry using topology optimization to minimize weight while maintaining stiffness under the expected load distribution from the heavy ball. The resulting design also featured material rotated by 45° and produced a structurally efficient network of load-bearing ribs. FEA results indicated that this design achieved the lowest mass among the eligible candidates while providing the greatest stiffness-to-weight ratio.

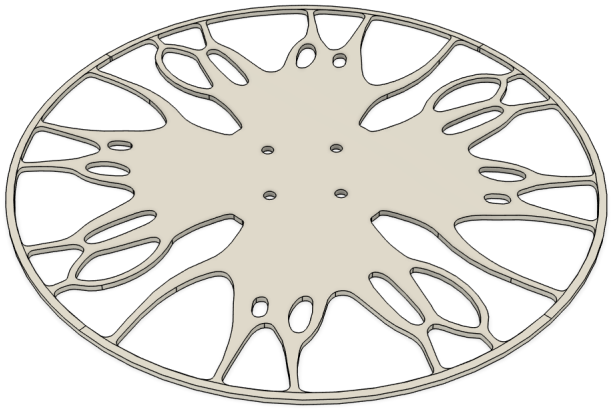


Figure 6: Topology optimized plate

To compare the two plate geometries, each design was evaluated using two metrics: (1) maximum displacement under the heavy-ball loading at edge scenario, and (2) total mass of the plate. A simple combined metric defined as the product of mass and maximum displacement was computed to provide a relative comparison. Since all plate designs are radially symmetric, analyzing a single plate accurately preserves the performance ranking and avoids the need to model a full three layer stack.

Design	**Max Displacement (mm)**	**Mass (g)**	**Mass $\times$ Disp.**
Topology-optimized	5.646	118.831	670.9
Standard cutout	6.417	137.990	885.5

Table 1: Comparison of plate performance under heavy-ball loading. Lower values of the combined metric indicate better stiffness-to-weight performance.

As a result, the topology plate was chosen. A visualization of the FEA done can be found in Appendix B.1.

2.3.4 Manufacturing Considerations

All designs were constrained by the available fabrication options at either E3, WatIMake or Rapid Prototyping Center. Laser cutting in 3 mm acrylic imposed limits on minimal feature size, kerf tolerance, and overall plate thickness.

2.3.5 Final Plate Selection and Performance

The final plate geometry (the topology optimized design show in the following figure) represented the best compromise between rigidity and weight. The plate maintained stiffness sufficient to not bottom-out under the heavy ball, while minimizing mass to reduce loading on the linkages. Physical testing confirmed that the design avoided the catastrophic failures of the first iteration and the excessive deflection of the second. CAD design of the plates can be found in appendix A.1.

The three acrylic plates were bonded using UV-curable resin, which proved to be a far superior choice compared to conventional superglue. The UV resin provided effectively unlimited working

time to tune alignment and tolerances before curing, while the bond could be instantaneously set by exposing the joint to a handheld UV nail lamp.

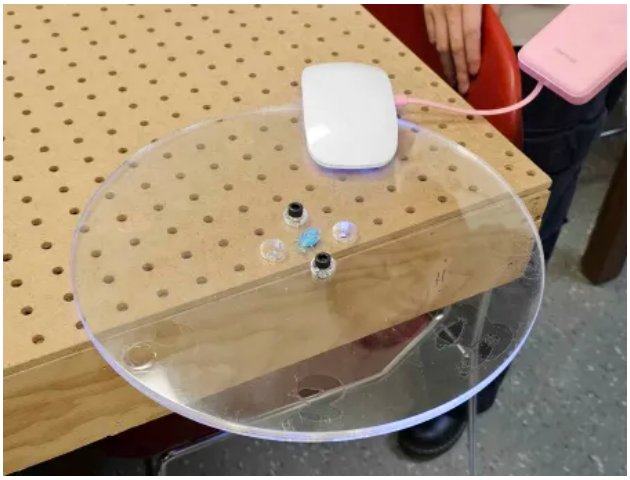


Figure 7: UV Bonding of the plates

Before committing to the final manufacturing acrylic, all plate geometries were first laser-cut using paper stock to verify dimensions. This step proved essential for validating clearances around the cutout patterns and confirming that the features aligned correctly after assembly. Once the paper iterations matched the expected tolerances, the final designs were laser-cut in 3 mm acrylic to ensure that the manufactured components fit together with minimal post processing required.

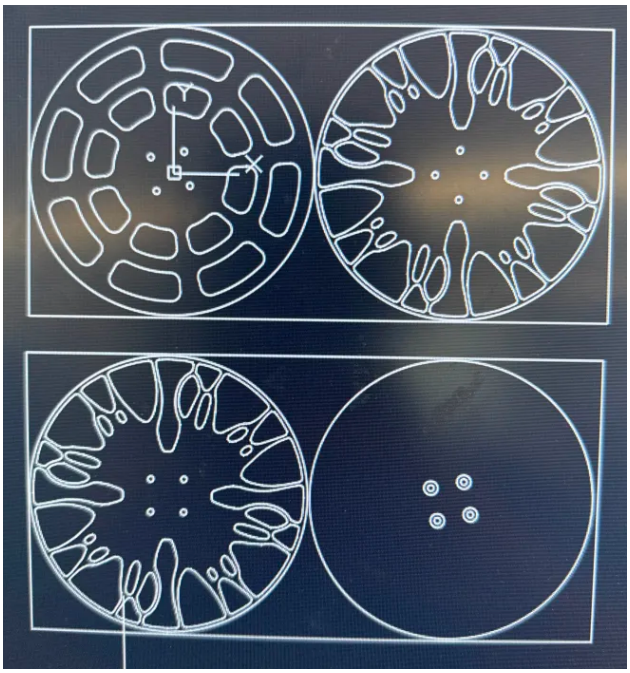


Figure 8: DXF for the plate prepared to laser-cut

2.4 Linkage and Joint Redesign

With the mechanical impacts of the heavier ball established, the linkage assembly was identified as the next major bottleneck in achieving reliable operation. Testing showed that the increased loads introduced excessive compliance at the bearing interfaces, making this region the primary source of mechanical slop. Accordingly, the redesign focused on strengthening the bearing housing and reducing free play between linkage components..

2.4.1 Geometric Reinforcement and Shear Mitigation

To improve resistance to shear failure, the linkages were redesigned to increase cross-sectional area in the regions surrounding the bearings. Material thickness was constrained by the available 3 mm acrylic sheets, but the links were widened wherever possible to increase shear capacity. This modification significantly reduced the likelihood of fracture under shear.

2.4.2 Slop Reduction Measures

Several iterative modifications were implemented to reduce slop between the links:

- 3D-printed end-stops: Small end-stop features were printed and bonded to prevent Link 2 from peeling or twisting away from Link 1 during aggressive tilting.
- Shims and washers: 0.1 mm shims were inserted between all bearing interfaces and outer faces of the links to minimize lateral play. The team did trial and repeats until the optimal number of shims were found to minimize slop while ensuring it does not generate an overly amount of friction.
- Lubrication: WD-40 was applied at the acrylic bolt interfaces to reduce friction induced torsion and to smooth the relative rotation between links.

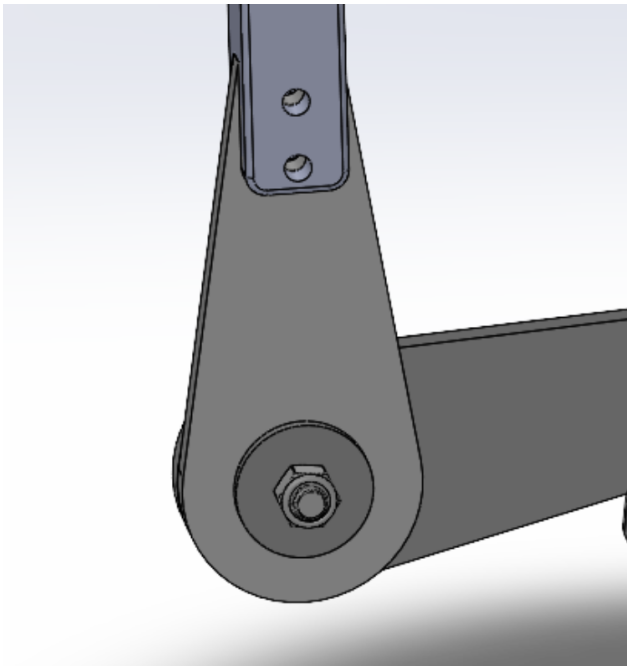


Figure 9: Design of the new joint with end stops

These measures reduced free play but did not fully eliminate compliance. The remaining slop

became a limiting factor for platform precision, especially when combined with the heavier ball's increased momentum. Under repeated motion, the compliance of acrylic joints introduced additional delay and variability into the control loop. But this became an addressable problem by tuning the parameters of the controller.

2.4.3 Remaining Limitations and Future Work

Despite the improvements, the acrylic linkages and joints retain a noticeable amount of flexibility. This residual amount of movement affects the effective platform dynamics, especially during rapid movements. The team thought of printing the links out of engineering filament to increase the allowable shear stress, but ultimately did not pursue it, as widening the acrylic links proved to be sufficient for the purpose of this demo. A future redesign should replace the acrylic links with machined aluminum and incorporate press-fit metal bushings to carry shear loads.

That being said manufacturing tolerances and better materials can only help so far. Several mitigation measures were implemented to reduce free play, which improved but did not fully eliminate residual slop. This remains a key limitation of acrylic-based linkages and motivated the other question being investigated, which is elimination of the ball joints with a parallel spherical manipulator. The improved design helps in practically with practically all the issues discussed in this section, it lowers the static flat holding torque required from the motors, it places lower stresses upon the connecting links, and it minimizes slop/play due to lateral loads imposed by the ball's movement.

2.5 Actuator and Motor Selection

The introduction of the much heavier steel ball and the significantly heavier platform structure (as a result of reducing deflection from the ball) created torque and dynamic loading conditions far beyond the capability of the MG995 servo motors. Even with the improved linkage geometry and reduced platform mass developed in the preceding subsections, actuator failure became a limiting factor during early testing. The actuator redesign therefore focused on two questions: (i) which servos are physically able to lift and accelerate the much heavier platform and ball, and (ii) how to prevent electrical failure caused by the interaction between platform inertia and servo back-driving, which was a surprise discovery during the implementation of the new servo.

2.5.1 Limitations of the Original Servos

The provided MG995 servos for the original platform were unable to lift the steel ball under any meaningful tilt command. The combination of platform mass and the inertia of the steel ball caused immediate stalling. Even after weight optimizing the plates, the original servos still lacked the torque necessary to tilt the platform in a controlled manner. This made actuator replacement unavoidable. Appendix E.1 gives the breakdown of specs for servos being discussed in the following sections.

2.5.2 First Replacement Attempt: High-Power Brushless Servos

The first upgraded actuator set consisted of high-power brushless servos (GOTECK "35 kg"). These motors provided sufficient torque but drew currents well beyond what any laboratory bench supply on campus could deliver. The instantaneous current draw were measured to reach 20A. Powering the system using Li-Po batteries was deemed impractical for extended testing, and the excessive current draw risked damaging the servo shield.

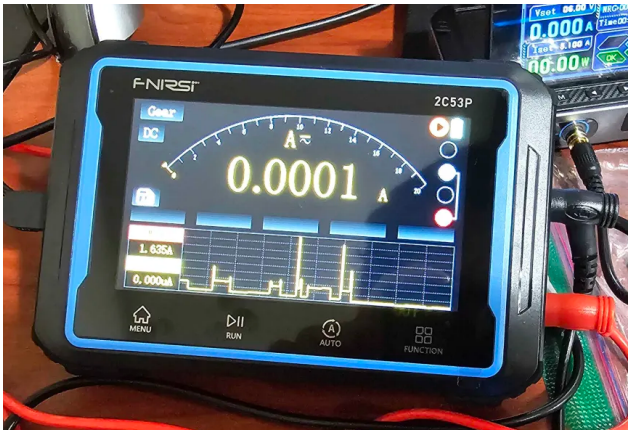


Figure 10: Current spike detected by the ammeter

As a result, despite meeting torque requirements and having excellent response (the behaviour almost matches balancing an ping-pong ball), these servos were not viable for the platform.

2.5.3 Second Replacement: Higher Gear Ratio Studica Servos

A second actuator redesign used Studica "Multi-Mode Smart Servo", which offered a substantially higher gear ratio and lower electrical power demand. These servos were mechanically capable of lifting the platform and steel ball; however, they introduced a new failure mode during downward tilting.

When some subset of the servos was commanded downward to tilt the platform, the mass of the ball accelerated the platform faster than the servo could back-drive under its own acceleration limits. The resulting relative angular velocity between commanded and actual shaft speed induced a back-EMF in the servo motor. This effectively caused the servo to behave as a electrical generator and injecting current back into the circuit.

The induced voltage propagated back through the servo shield into the bench power supply, which detected the event as an overvoltage condition [1] and shut down the system. This problem became reproducible during any rapid tilt.

2.5.4 Electrical Protection with Diode and Dissipation Path

To prevent the power supply from shutting down, a protection circuit was added between the servo shield and the bench supply. A flyback diode was installed to block reverse current flow, ensuring that the back-EMF generated by the servo did not propagate into the supply. However, isolating the power supply meant that the induced current required a dissipation path.

A capacitor and large power resistor were installed in parallel as an energy sink. The resistor served as a heatsink to convert the back-driven electrical energy into thermal dissipation.

Table 2 lists the specific parts used in the final design. The detailed electrical sizing calculations for the diode, capacitor, and resistor are provided in Appendix C.1.

[1] The team believes the power supply's reverse current protection kicked in based on what's seen by the ammeter, but this theory is not confirmed.

Component	Model	Rating
Dump Resistor	PATIKIL Aluminum Shell Resistor	10 Ω, 100 W
Schottky Diode	BOJACK 15SQ045 Diode	15 A, 45 V
Energy Buffer Capacitor	SING F 4700 μF Capacitor	4700 μF, 50 V

Table 2: Components installed to protect the servo power bus from back-EMF induced over-voltage.

The following figure shows the assembled circuit with the diode and dissipation soldered onto the shield.

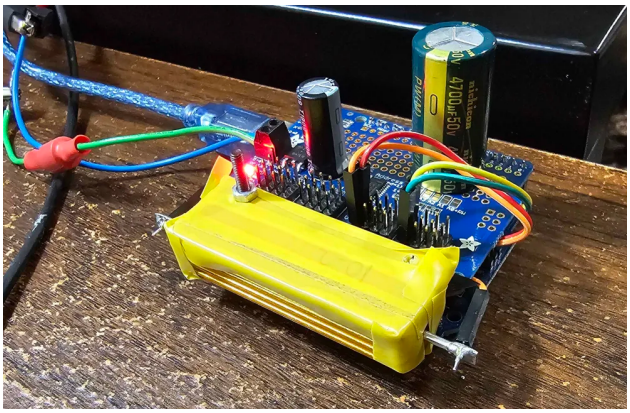


Figure 11: Protection circuit added between servo shield and power supply (flyback diode, resistor, capacitor).

2.5.5 Dynamic Performance Under Heavy Load

Even with sufficient torque and electrical protection to tilt the platform, platform dynamics still changed significantly because the heavy ball increased system inertia, causing the platform to overshoot and oscillate unless strong derivative action was used in the controller.

This actuator–sensor interaction dictated the achievable closed-loop performance: aggressive servo action risked exciting noise, while weak servo action failed to counteract ball inertia. This challenge is addressed further in the control section.

Future improvements would involve selecting servos with integrated regenerative braking management or switching to actuators designed for back-driving, such as certain DC motor/gearbox configurations with external encoders. Steppers would also provide a very significant edge over the servos, because the motor dynamics are know and can be exactly modelled as part of the controller. In the servo's case, it essentially acts as a black-box with non-predictable dynamics.

2.6 Control Modifications for Heavy Ball

Although the overall controller structure remains a PID loop acting on the plate tilt, the introduction of the much heavier steel ball required several practical modifications to the control tuning and operating limits of the system.

The most significant effect comes from the ball's substantially increased momentum rather than its acceleration. Although the rolling acceleration on a tilted plate is independent of mass, the heavier ball generates much larger forces and carries significantly more momentum at any

given velocity. With the original gain values, this caused the ball to overshoot the desired region and oscillate, since the servos were unable to generate sufficient opposing tilt quickly enough to counteract the increased momentum.

As the ball accelerates towards edge of the platform and quickly gains momentum, a purely proportional controller can no longer compensate enough to contain the ball within the boundaries of the plate. Therefore, a sufficiently high derivative term were deemed necessary to counteract the velocity of the ball. This will be further discussed in the controls section of the report.

The heavier loading also slows the response of the actuators. Under the increased platform mass, the Studica servos cannot achieve the same acceleration or tracking accuracy as in the baseline configuration withe the MG995 and ping-pong ball. Commands demanding rapid tilt changes resulted in noticeable lag between the requested and actual platform angle. To accommodate this, the controller gains were tuned more conservatively and the allowable tilt rate was effectively limited by selecting gains that remained within the servos' acceleration and torque capacity.

These modifications ensured that the PID controller remained stable and predictable under the new ball configuration. A full discussion of the final control strategy, simulation analysis, and trajectory-based methods is provided in Section 4.

2.7 Summary of Mechanical, Electrical, and Control Redesign As a Result of Heavier Ball

The 538 g steel ball required major changes to the platform's mechanical, sensing, and actuation design. The substantially larger forces and moments drove a full redesign of the top plate, linkages, and joints, and exceeded the capability of the original servos, which led to the replacement of existing MG955 servos with the higher-torque Studica servos, together with a regenerative protection circuit. These mechanical changes also constrained the achievable control performance: the heavier ball's increased momentum required stronger derivative damping, reduced proportional action, and more conservative bandwidth to remain stable under the servos' loaded dynamics (much slower than the ping-pong ball). Finally, the use of a force and moment sensing method imposed strict stiffness and load-path requirements that guided the optimized plate geometry. These redesign outcomes form the basis for the control methods presented in Section 4.

3 Ball Position Data Acquisition via Camera and ForceN

3.1 Camera Based Location Tracking

As a result of switching to a metallic ball with a reflective surface, the corresponding computer vision detection algorithm from the original ball tracking software must be modified to accommodate this change.

The most significant problem with switching to a reflective ball is that attempting to pick out the ball object from its colour no longer works. As the ball's surface is mirror-like, it reflects the outside world environment, meaning that the colour on the ball's surface changes with environmental change. This change made colour based detection unrealistic. The objective is to obtain a stable estimate of the steel ball's center under strong specular reflections and varying illumination.

Ball position is extracted from each camera frame using a constrained Hough Circle Transform combined with platform based geometric masking. The captured RGB frame is first converted to grayscale and smoothed using a Gaussian filter. This reduces high-frequency noise and suppresses small reflection artifacts that commonly appear on metallic surfaces. Let I_g denote the resulting grayscale image.

To prevent false detections outside the platform region, a binary mask is generated based on the previously calibrated platform center and radius. The mask $M(x, y)$ is defined as a filled circle corresponding to the platform interior. The pre-processed frame is then spatially constrained by computing

$$I_m(x, y) = I_g(x, y)\, M(x, y),$$

which ensures that only pixels inside the platform contribute to subsequent circle detection. It is assmed that the platform is clear of any other objects except for the rolling ball.

After pre-processing of each frame, circle detection is performed using the Hough Gradient method implemented in `cv2.HoughCircles`. The parameters are tuned specifically for the geometry of the steel ball and the expected imaging conditions. Significant attention to tuning is focused on the strictness of circular detection(Param 1 and Param 2). Too strict of circular detection results in circles being snapped to the reflection on the steel ball, which also are a circular arc. Such parameters only need slight adjustment for each environmental change. The distance threshold `minDist` is set to 40 pixels, as it is unreasonable for the ball to have travelled more than 40 pixels in distance in between 2 consecutive frames.

The Hough transform typically returns several candidate circles; these are sorted in descending order of radius, and only the largest physically reasonable circle is retained. Detections outside the allowable radius range (5–40 pixels) are rejected to avoid false positives. Once a valid circle is selected, its pixel coordinates (x, y) are converted into physical displacement from the platform center. This is performed using the calibrated pixel-to-meter scaling factor k and the known platform center (c_x, c_y):

$$\Delta x = (x - c_x)k, \qquad \Delta y = -(y - c_y)k.$$

The negative sign in the y-direction is an adjustment made to accommodate the physical setup, compensating for the camera's set up position. The algorithm outputs the ball position in meters, the corresponding pixel coordinates, and the detected radius. These values can subsequently be used by the control system for position estimation and positional intake.

3.2 Force and Moment Based Location Tracking

The new implemented sensing mode estimates the ball's in-plane position from the vertical reaction force and tipping moments measured by the Forcen ASY-00035 three-axis loadcell rigidly mounted beneath the top plate. The device streams real-time measurements at 100 Hz in the format

$$\langle M_x,\ M_y,\ F_z,\ W,\ \theta_{\text{pitch}} \rangle,$$

where M_x and M_y are moments (Nmm), F_z the normal force (mN), W an internal weight estimate (g), and θ_{pitch} a diagnostic tilt angle. The companion class `ForcenInterface` handles serial configuration, frame parsing, filtering, and coordinate transforms. The full implementation code for this feature can be found in Appendix F.1.

3.2.1 Quasi-Static Position Model

When the ball of weight W rests statically at planar position (x, y) on the plate, it applies a vertical reaction $F_z \approx Wg$ at that location. Because the new mental ball used by the team is a bearing ball, it have a extremely low surface roughness and perfectly spherical, which allows it to behave like a point load when sitting on the platform. Therefore, neglecting horizontal acceleration, the load path is modeled as a single vertical force applied at (x, y), giving the moment balance

$$\boldsymbol{M} = \boldsymbol{r} \times \boldsymbol{F} = (x, y, 0) \times (0, 0, F_z) = (yF_z,\ -xF_z,\ 0).$$

Thus,

$$x_{\text{raw}} = \frac{M_y}{F_z}, \qquad y_{\text{raw}} = \frac{M_x}{F_z}. \tag{4}$$

In software, readings with $|F_z| < 10$ mN are rejected to avoid division by small force when the ball loses contact.

3.2.2 Filtering and Validity Checks

Raw single-frame estimates from (4) are highly sensitive to sensor noise and to dynamic platform accelerations. This becomes a significant problem when parsed to the controller, as the derivative term will interpret such noises as error and enforce an enormous correction force. The processing pipeline therefore applies several checks:

- **Radial bound:** Readings with $\sqrt{x_{\text{raw}}^2 + y_{\text{raw}}^2} > 150$ mm (nominal plate radius) are discarded and replaced with the last valid position.
- **Force gating:** Only frames with $F_{z,\min} < F_z < F_{z,\max}$ are accepted. In the final implementation, $F_{z,\min} \approx 4000$ mN and $F_{z,\max} \approx 8000$ mN.
- **Exponential filter:** Valid samples are smoothed using

$$x_f[k] = \alpha x_{\text{raw}}[k] + (1 - \alpha) x_f[k-1], \tag{5}$$
$$y_f[k] = \alpha y_{\text{raw}}[k] + (1 - \alpha) y_f[k-1], \tag{6}$$

  with $\alpha \approx 0.9$. The last valid (x_f, y_f) is stored for use when a sample is rejected. This effectively behaved like a first order low pass filter on the sensor noise.

Additionally, the state is considered valid only when the internal weight estimate exceeds a threshold ($W > 200$ g) to ensure the ball is present on the plate. Therefore, an series of additional calibration are added.

3.2.3 Offset, Scaling, and Coordinate Alignment

Even after filtering, (x_f, y_f) does not directly match the platform coordinate frame due to mounting offsets, scale mismatch, and sensor-to-plate rotation.

Offset Removal. Before computing ball position, the raw Forcen readings have a non-zero baseline due to the weight of the plate, preload in the mounting screws, and internal bias of the sensor. To remove this bias, the platform is left empty and a batch of samples is collected (this is noted as the "calibration phase" during each time the platform boots up). Averaging these readings yields

$$(\texttt{offset\_x},\ \texttt{offset\_y}),$$

which define the recentered coordinates

$$x_c = x_f - \texttt{offset\_x}, \qquad y_c = y_f - \texttt{offset\_y}.$$

Axis Scaling. Scale factors `scale_x` and `scale_y` are chosen by collecting measurements of the ball sitting at the edge, so that the measured radius at the plate edge matches the known physical value:

$$x_s = \texttt{scale\_x}\, x_c, \qquad y_s = \texttt{scale\_y}\, y_c.$$

Rotation and Axis Flip. Because the sensor axes were mounted at 135° relative to the platform frame, the aligned coordinates are

$$x_r = x_s \cos\theta - y_s \sin\theta, \tag{7}$$

$$y_r = x_s \sin\theta + y_s \cos\theta, \tag{8}$$

with $\theta = 135°$. Finally, signs are flipped as needed so that the loadcell based coordinates match the camera frame:

$$x = x_r, \qquad y = -y_r.$$

3.2.4 Runtime Integration and Visualization

At each control cycle, the Forcen interface provides the latest position estimate (x, y) together with diagnostic quantities. The controller reads this state directly and uses it as the ball position measurement whenever the camera feed is unavailable.

For visualization and logging, each update writes a compact state vector to a shared binary file:

$$\texttt{ball\_state.dat} = \{x,\ y,\ W,\ \theta_{\text{pitch}},\ \theta_{\text{roll}},\ e_x,\ e_y,\ t\},$$

consisting of eight 64-bit floating-point values. A dedicated visualization script (`viz_forcen.py`) continuously reads and plots this state in real time. The separation between the controller and visualization process ensures that the control loop remains lightweight and deterministic.

4 Control Strategy and Trajectory

4.1 Problem Definition

The control objective is to regulate the ball at rest from an arbitrary initial position on the plate to a desired target location while maintaining stability over the full operating range of the Stewart platform. Early in the term, the team planned to experimentally compare three reference–generation strategies:

1. a direct step command from the initial position to the target;
2. a fixed, time–parameterized trajectory that the ball should follow;
3. a trajectory with periodic replanning that updates the desired path based on the current ball state.

As the design evolved to include loadcell based position estimation and a steel ball with nearly two orders of magnitude greater inertia than the original ping-pong ball, the closed-loop dynamics became significantly slower and more sensitive to disturbances. The combination of higher inertia, slower servo response, and sensing latency reduced the effective stability margin of the platform. Under these conditions, aggressive reference tracking (particularly trajectory generators and periodic replanning) becomes difficult to stabilize without a high-fidelity dynamic model and faster servo dynamics.

For this reason, the three trajectory strategies were evaluated primarily in a reduced-order Simulink environment, where plant dynamics and sensing latency could be idealized. On the physical system, the real hardware implementation focused on a robust stabilizing controller capable of bringing the ball to the plate center (or any other set locations) when the ball receives a moderate disturbance.

4.2 Reduced-Order Simulink Plant Model

To study the three reference–generation strategies, the team referenced and adapted an online model posted by Nassim Khaled (`ball_plate_interaction`) [4] into a plant for simulation. In this model, the Stewart platform is abstracted as a rigid circular plate supported by a two-axis universal joint (pitch and roll) driven by second order actuators, with a rigid spherical ball rolling on the plate surface. The abstraction removes the full Stewart kinematics, leg dynamics, and servo electronics, but retains the essential ball–plate interaction needed to compare controllers.

The core of the plant is the `ballplateforces` S-function [4], which computes normal and friction forces at the contact patch based on the instantaneous ball position, translational and angular velocities, and simulation time. The function models:

- a linear spring–damper penalty for ball penetration through the plate surface using stiffness K_{pen} and damping D_{pen};
- a circular support region of radius r_{pla} and a spherical ball of radius r_{bal};
- Coulomb-like static friction with coefficient μ_{stat} and a slip-threshold speed v_{thr}, producing tangential friction forces F_{fx}, F_{fy} proportional to the normal load F_z;
- rolling torques T_x, T_y about the ball center generated by the tangential forces.

The S-function outputs $(F_{fx}, F_{fy}, F_z, T_x, T_y)$, which are applied to the ball body in the Simscape 6-DOF joint block. The resulting state evolution yields ball position and velocity in x and y, which are fed back to the controller blocks.

Each plate axis is actuated independently. The Simulink diagram implements two identical single-input–single-output loops: one for x and one for y. For each axis, the controller block computes a desired plate angle from the position error (and optionally velocity), and this command

is passed through a second-order servo model before reaching the plate. The servo dynamics are approximated by the transfer function [5]

$$G_{\text{studica}}(s) = \frac{5118}{s^2 + 103.6\,s + 4903}.$$

representing a actuator with finite bandwidth and rise time. This is a far simpler model than the actual underlying logic behind a servo in real life, as it contains internal controller loops to control the actuation to target angles. However, as the response is not apparent and can't be simply determined, the team can only estimate the response based on known parameters, so the tilt of the plate is not instantaneous. A full write out of this calculation can be found in Appendix E.2.

A disturbance block injects additional planar forces $F_x(t), F_y(t)$ into the ball model, so the team can observe the system response under controlled perturbations.

4.3 Trajectory Generation in Simulink

On top of the plant, three reference profiles were implemented at the Simulink level. The full implementation can be found in Appendix F.3

- **Direct step to target:** The reference block outputs an instantaneous step in x and y from the initial ball position to the target position set. This excites the closed loop most strongly and exposes overshoot and damping characteristics of the inner PID design.
- **Fixed, time–parameterized trajectory:** A pre-computed position trajectory is defined as a function of time (a trapezoidal velocity profile if distance is sufficient, or a triangular if not). The plant sees a continuously evolving reference, and the PID must track this path without excessive lag or phase error.
- **Periodic–replanning trajectory:** At intervals, the reference generator recomputes a new trajectory segment from the current ball state to the target.

To compare the behaviour of the three reference profiles, the following conditions were held constant across all simulations:

- The ball begins at rest at the origin, $(x, y) = (0, 0)$.
- At $t = 1$ s, a command is issued for the ball to move to the point (0.08, 0.08) m.
- The same PID gains are used for all three cases; no retuning is performed at the trajectory level.
- For each method, the ball's position, velocity, acceleration, radial distance, tracking error, and commanded plate tilt are logged over the full simulation interval.

Table 3 summarizes the physical parameters used in all simulations. These values match the measured ball properties and the CAD dimensions of the redesigned plate.

Table 3: Physical Parameters Used in Simulation

Parameter	**Value**	**Unit**
Ball radius	0.0254	m
Ball mass	531	g
Plate thickness	0.003	m
Plate radius	0.15	m
Spring constant k	10 000	N/m
Damping constant c	100	N·s/m
Static friction coefficient μ_s	0.6	–
Friction threshold speed	1×10^{-4}	m/s

In the PID control layer, the proportional, integral, and derivative gains used for the y-axis during simulation were:

$$K_P = 10, \qquad K_I = 0.3, \qquad K_D = 1.0.$$

Because the setpoint is symmetrical in the x and y direction, only one axis was analyzed in the following section.

4.3.1 Simulation Results

The first case uses a direct step input. Because the reference jumps instantaneously at 1 second, the controller observes a large position error at the moment of the command change and responds with a large control effort. This produces a sharp tilt of the plate, causing the ball to accelerate aggressively, overshoot the target, and ultimately fall off the platform.

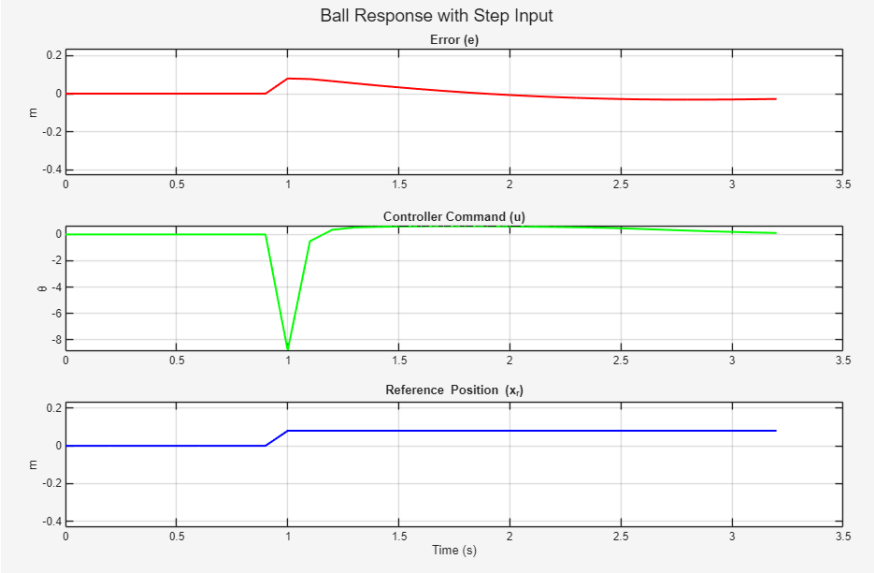


Figure 12: Ball response to a step input to (0.8, 0.8)

Under a fixed, pre-computed trajectory, the ball tracks the reference very closely, resulting in near zero error throughout the motion. The smoothness of the trapezoidal profile prevents the large control spikes observed in the step-input case. However, as the ball nears the target, a small

overshoot still occurs. The controller subsequently produces a series of corrective tilt and jitters to settle the ball at the final position.

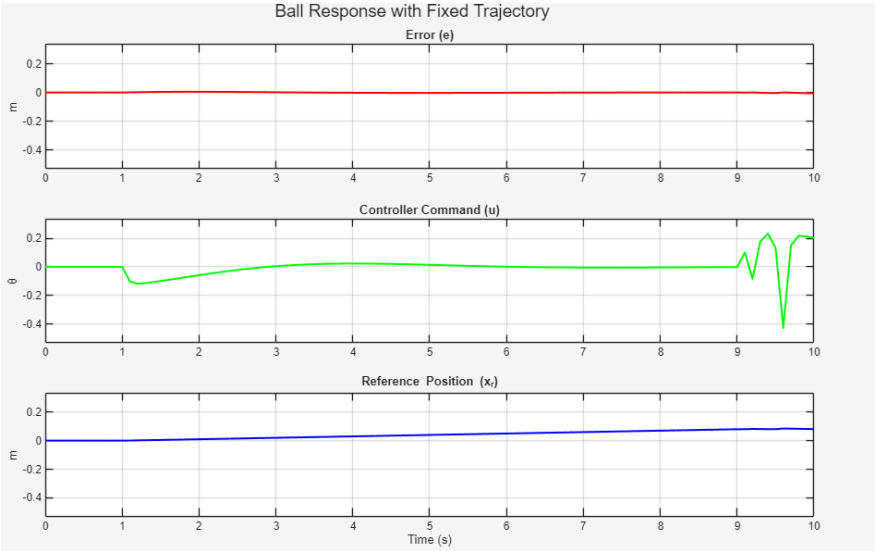


Figure 13: Ball response to a fixed trajectory

The time updated trajectory also had the ball follow very closely. Initially, the periodic replan happens regardless of whether the ball is correctly tracking the reference position profile, which caused each trajectory update to introduce a discontinuity in the reference velocity and acceleration. As a result, every replan effectively "reset" the motion plan from the ball's instantaneous measured state, producing abrupt changes in the reference and manifesting as sharp spikes in the controller output. These spikes occur because the controller suddenly observes a mismatch between its previously planned deceleration phase and the newly generated trajectory, forcing an opposite tilt command to realign the ball with the updated reference.

Once the replanning logic was modified to trigger only when the ball was genuinely diverging from the desired path, the trajectory generator no longer overwrote the deceleration phase.

Because the simulink setup has a deterministic profile (there's no random fricional/backlash/sensor noise disturbances during simulation). When disturbances are not injected into the system, the ball's response to the time updated trajectory essentially always matches fixed trajectory.

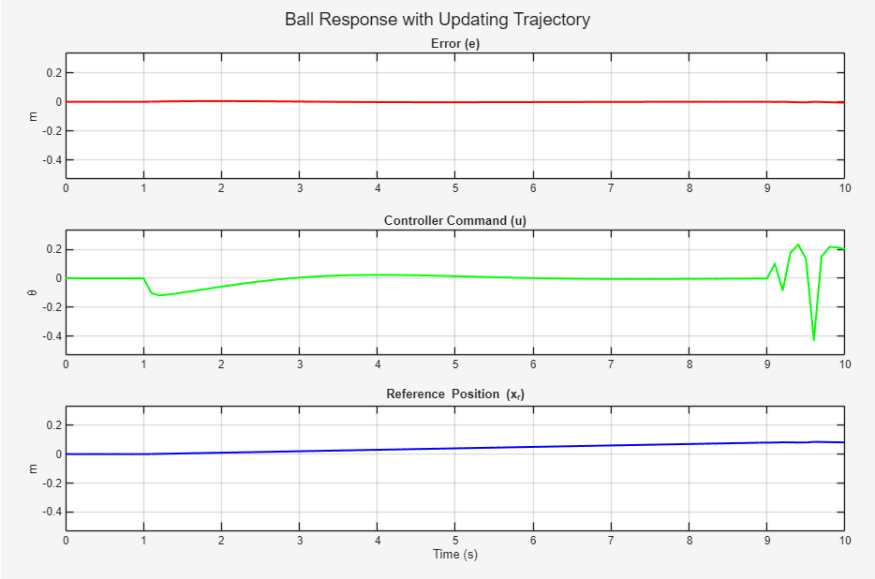


Figure 14: Ball response to a periodically updated trajectory

The difference comes from when disturbances exist in the system. From the following two figures, it is observed that with a disturbance injected during the ball's movement at 4 seconds, the fixed trajectory input ignores this disturbance, which resulted in an higher control effort to try "catching up" the ball with the newest reference location. In contrast, the time parametrized trajectory catches this change in the next update, which plans a new trajectory and results in the ball following this more "optimal route" to the target. As a result, a 1 second time improvement can be seen here when periodically updated trajectory is followed.

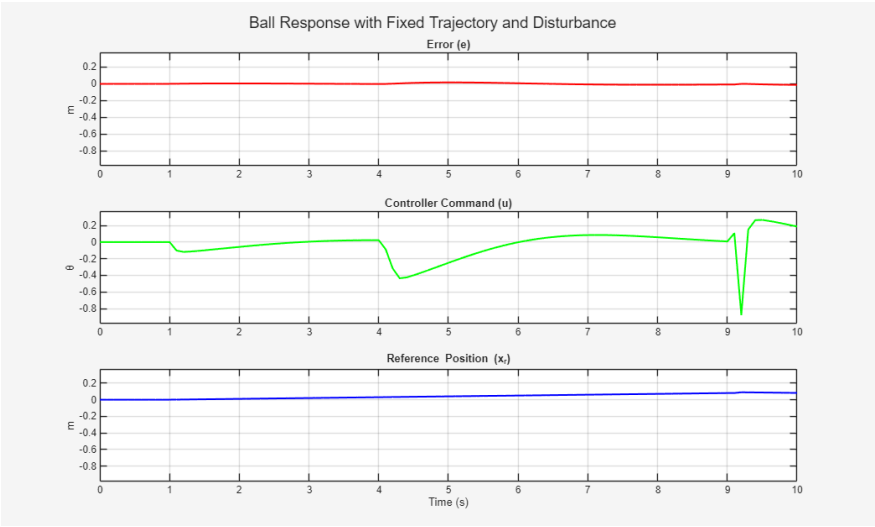


Figure 15: Ball response to a disturbance while following a fixed trajectory

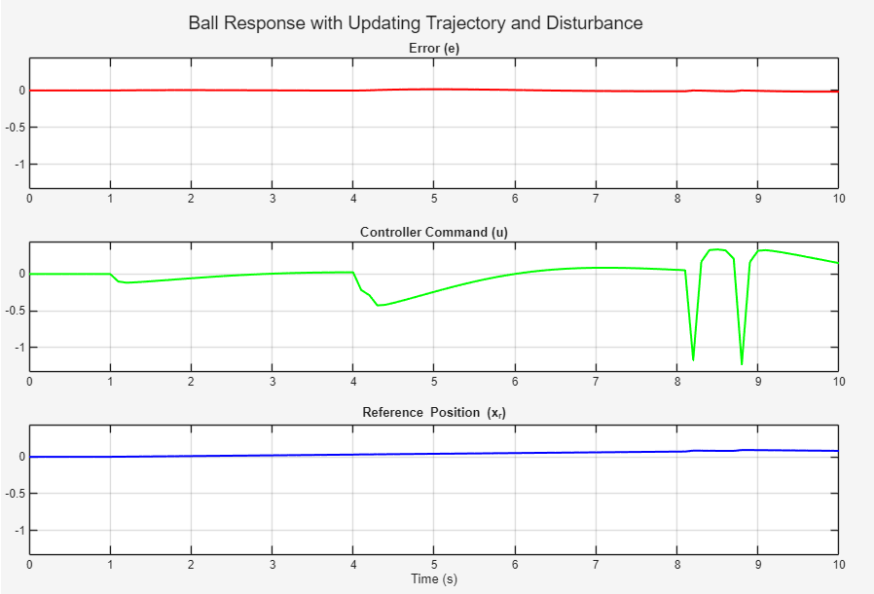


Figure 16: Ball response to a disturbance while following a periodically updated trajectory

For both cases, the ball was able to closely track the generated path. Feed forward was applied for both acceleration and velocity, to the PID feedback controller only needed to care about the tiny errors caused by disturbance.

4.3.2 Simulation Analysis

The three simulation scenarios had quite distinct behaviours depending on how the reference position is generated. A direct step input produces the most aggressive dynamics, as the controller must instantaneously counter a large position error. Therefore, this approach only seems suitable for applications with high damping or when rapid repositioning is required and overshoot is acceptable. In contrast, a fixed trajectory provides a smooth, time-parameterized motion that the ball can follow with minimal error. This method is most appropriate for point-to-point moves with no disturbance and where minimizing controller effort is the goal. The periodically updated trajectory works the best in environments with disturbances or drift, as it constantly re-evaluates the new trajectory based on error, which makes sure the ball will always be following the most optimal path to target.

It is worth noting that a fixed, pre-generated trajectory only makes sense when the ball begins from a known stationary state. If the ball is disturbed and its initial position is not incorporated into the trajectory generation, the reference will no longer correspond to the actual motion, causing the controller to misinterpret the error. In contrast, the periodically updated trajectory avoids this limitation by regenerating the trajectory from the current measured state.

Another observation is that the specific controller gains play a diminished role when trajectory planning is present. Because the reference itself is already smooth and dynamically feasible, the controller primarily performs small corrective actions rather than driving the full acceleration profile. Thus, even with imperfect gain tuning, the trajectory cases maintain stable behaviour and accurate tracking. In contrast, when applying the same gains for the step input, the controller exhibited strong sensitivity to the values, as it must generate the entire motion from the instantaneous error alone. Therefore, trajectory planning was able to create more predictive performance, by simply setting the limit of velocity and acceleration to match the limitations of the real system.

The trajectory based methods also demonstrated a strong advantage at minimizing the system's effort to control the ball. By bounding the reference velocity and acceleration, trajectory planning essentially eliminates the large and wasteful control spikes associated with instantaneous reference jumps. As a result, both the fixed and time-updated trajectories outputs smoother plate motions, which translated to lower actuator demand if this were implemented in the actual ball balancing system.

4.4 Final Real-Time Controller Implementation

While the Simulink study informed high-level design choices, the real hardware implementation ultimately adopted a single robust control strategy: decoupled PID regulation of x and y with velocity feedforward and derivative filtering, operating on loadcell-based position feedback. Trajectory planning was developed and tested on the real hardware, however, it is suspected that due to the significant noise in ball position sensing, this causes an un-reliable velocity reading, making the techniques used for trajectory planning cause the system to go unstable regardless of the PID gains used or the acceleration profile used.

4.4.1 Feedback Architecture and Error Definition

In the final platform, the ball position is estimated primarily from the loadcell. The companion `ForcenInterface` class converts the raw (M_x, M_y, F_z) signals into an estimated planar position (x, y) through calibration and filtering (Appendix G.13). The controller receives (x, y) and compares it against a fixed setpoint $(x_{\text{sp}}, y_{\text{sp}})$, which is default set to the plate center (but can be adjusted) and generates the error function:

$$e_x(t) = x_{\text{sp}} - x(t), \qquad e_y(t) = y_{\text{sp}} - y(t).$$

A small deadband is applied around the target (again in this case, the center) so that when the radial error $\sqrt{e_x^2 + e_y^2}$ falls below a few millimetres, the error is effectively treated as zero. This prevents unnecessary chattering once the ball is visually at rest near the center.

4.4.2 Decoupled X–Y PID with Velocity Feedforward

The chosen inner-loop structure treats the ball motion as two approximately independent axes. Two PID controllers are implemented, one for each direction, and their outputs are interpreted as desired platform pitch and roll:

$$\theta_{\text{pitch}}(t) = f_{\text{PID},y}\big(e_y(t)\big), \qquad \theta_{\text{roll}}(t) = f_{\text{PID},x}\big(e_x(t)\big).$$

These are implemented as two instances of a custom PID class in software, `pid_x` and `pid_y`, which share the same gain set and are tuned symmetrically:

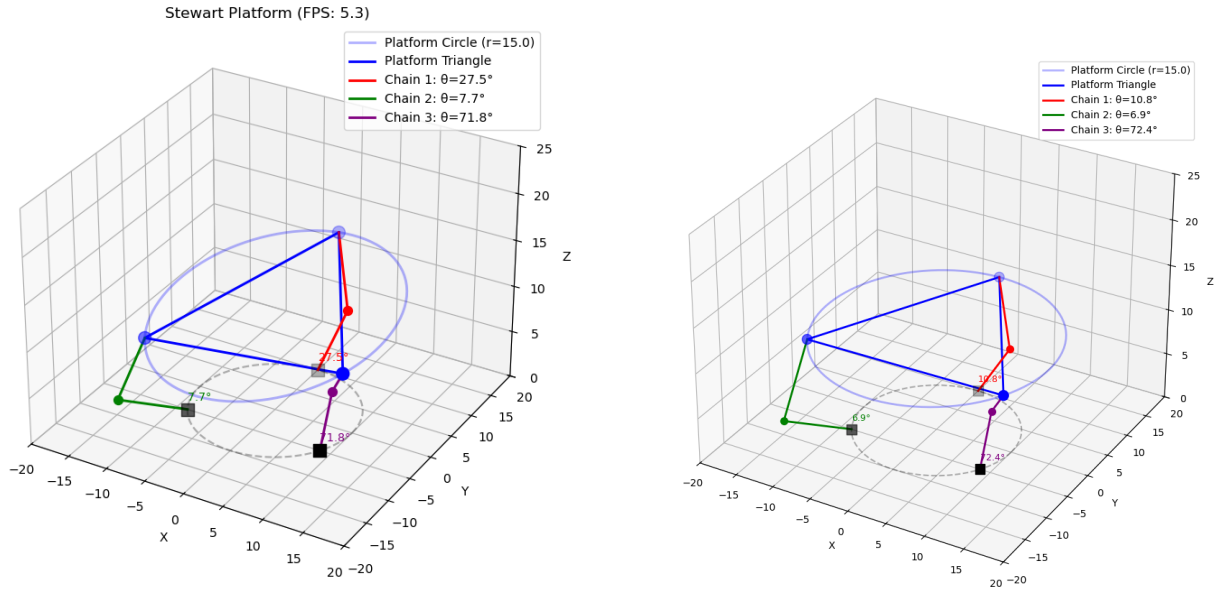
- Proportional, integral, and derivative gains (K_p, K_i, K_d) are applied separately in x and y.
- A velocity feedforward term is included to anticipate ball motion and counteract inertia. However, because trajectory planning was never really implemented, the feedforward velocity is always set to 0 , which just in turn acted like a secondary derivative block.
- The derivative term is low-pass filtered to reduce amplification of sensor noise. (This was previously discussed, but because how significant the D term becomes when the ball rolls near the edge, the gain is set to be very high. Therefore, this low-pass filter plays an critical role of preventing random noises from being interpreted as massive velocity errors the controller must counter-react and rapidly destablizes the system.)

Ball velocities $(\dot{x}, \dot{y})$ are estimated in real time by fitting a line to a short window of recent position samples. These velocity estimates are passed into the PID objects and used in a predictive feedforward term that "leans into" the ball motion, which provides additional damping for the steel ball.

The final control law implemented in software for each axis has the form

$$u_\alpha(t) = K_p e_\alpha(t) + K_i \int e_\alpha(\tau)\,\mathrm{d}\tau + K_d\,\dot{e}_{\alpha,\mathrm{filt}}(t) - K_{\mathrm{ff}}\,\dot{x}_\alpha(t),$$

where $\alpha \in \{x, y\}$, $\dot{e}_{\alpha,\mathrm{filt}}$ is the low-pass–filtered derivative of the error, and $\dot{x}_\alpha$ is the estimated ball velocity. The outputs u_x, u_y are then mapped to platform pitch and roll, combined into an equivalent plate orientation (rotation matrix and normal vector), and converted to three servo angles via the Stewart platform inverse kinematics before being sent to the arduino (servo controller).



(a) unbalanced (b) balanced

Figure 17: Live kinematic mapping of links and servo angle

To support tuning on the physical system, a dedicated `viz_tuner.py` GUI writes gain values $(K_p, K_i, K_d, K_{\mathrm{ff}})$ to a shared file, and the main controller periodically reloads these gains at runtime. This allowed the team to prioritize stability first (minimum gains required to move the ball) and then increase damping through derivative and feedforward terms until overshoot was acceptable and keeps the ball within the boundary of the platform.

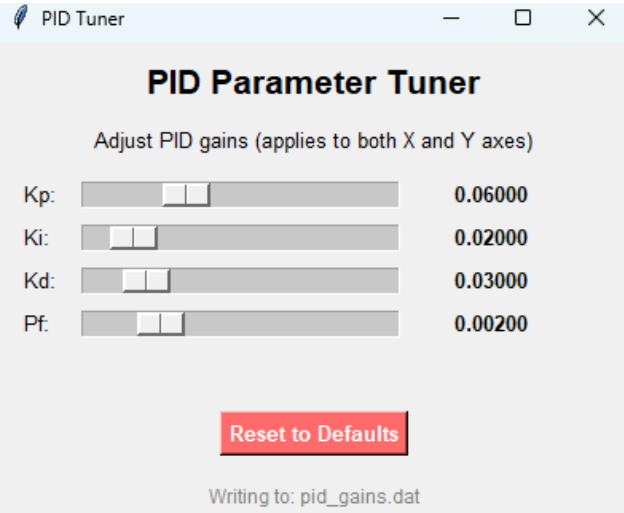


Figure 18: The PID tuner GUI

4.4.3 Synchronization

The software architecture separates the real-time control loop from all auxiliary tools by running each component in its own process and synchronizing them through small binary files. The main controller executes the PID law at a fixed update period, while visualization scripts (`viz_forcen.py`, `viz_spv4.py`, `viz_platform_cal.py`) read continuously updated state snapshots from `ball_state.dat` and `platform_state.dat`. This lets the team use responsive graphical displays without impeding the performance of the control loop. Likewise, the tuning interface `viz_tuner.py` writes updated gain values to `pid_gains.dat`, which the controller reloads asynchronously during runtime. This file based communication between the processes allows each component to run at its own natural rate—control at hundreds of hertz, visualization at up to 60 Hz, and tuning on demand. At the same time, the real-time loop remains deterministic. Full implementation can be found in the Appendix G.

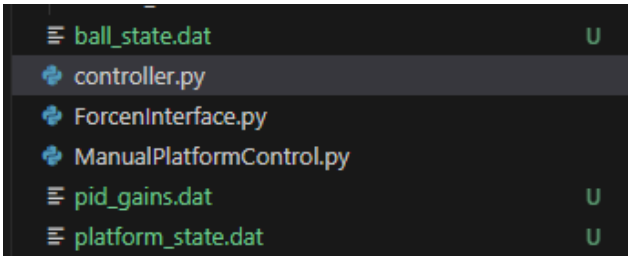
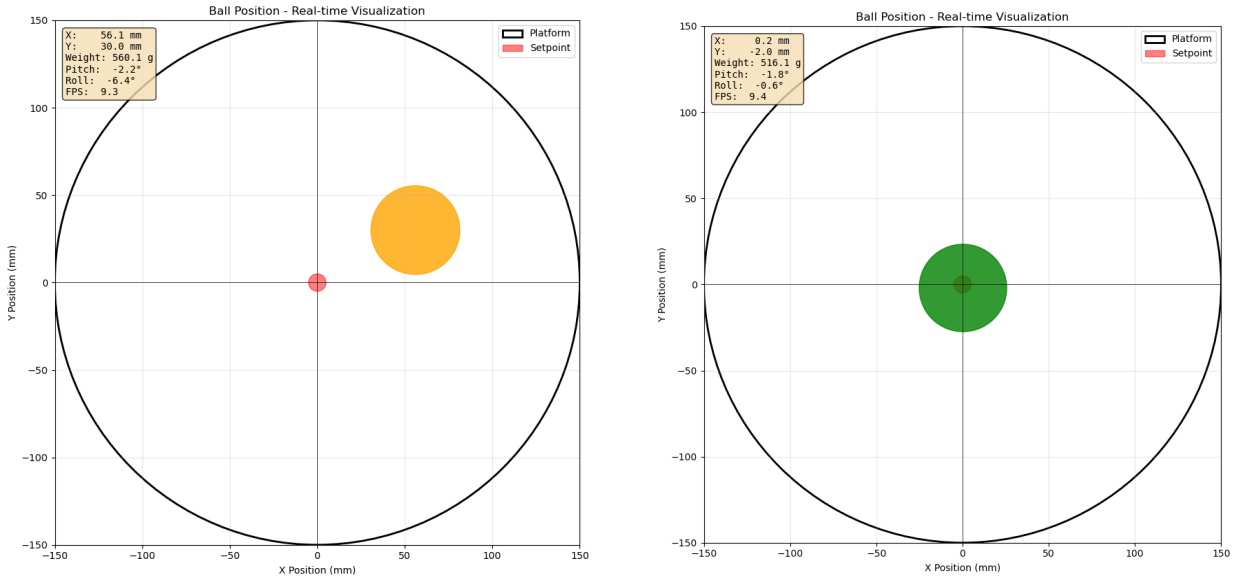


Figure 19: Binary files created during runtime



(a) unbalanced (b) balanced

Figure 20: Visualization interface for the ball's current position

5 Eliminating Ball Joints in the Stewart Platform

5.1 Problem Definition

The objective of the second investigation is to study the mechanical role of ball joints in the Stewart platform, and to design an alternative mechanism that eliminates the need for ball joints while still enabling controlled multi–degree-of-freedom motion. In the current Stewart platform being provided by the course, ball joints provide a passive spherical three axis rotational degree of freedom between each actuator link and the top plate, which prevents kinematic binding on the platform.

Removing ball joints from the existing design introduces geometric and kinematic constraints that can lead to internal stresses, actuator binding, and reduced motion range. The problem therefore requires developing an alternative joint or compliant mechanism capable of transmitting force and motion while resolving 3-DoF alignment requirements. The analysis will focus on how joint design influences workspace (range of motion the platform have without mechanical interference), kinematic accuracy, drivability, structural rigidity, assembly tolerance, and manufacturing complexity.

5.2 Constraints and Specifications

The redesigned mechanism must replace the ball–joint interface without restricting the platform's required rotational degrees of freedom. It must allow smooth, multi-axis tilting without actuator binding, maintain controllability, and preserve kinematic accuracy with minimal unintended compliance or backlash. The mechanism must withstand repeated loading from the platform and ball, tolerate small fabrication misalignments, integrate cleanly with the existing servo geometry, and remain manufacturable methods such as 3D printing and laser cutting.

Functional Requirements

- Preserve the rotational degrees of freedom required to avoid actuator binding.
- Provide a functionally equivalent replacement for the spherical joint interface.
- Maintain controllability over the tilt range needed for balln balancing.

Structural Requirements

- Withstand repeated platform and ball loading without structural degradation.
- Tolerate small geometric misalignments and fabrication tolerance stackup.

Fabrication and Integration Requirements

- Be manufacturable using 3D printing, laser-cut parts, and standard hardware.
- Install with minimal changes to existing servos or platform geometry.

Performance Requirements

- Achieve stable balancing and target-tracking using the existing control strategy.
- Maintain a maximum tilt range comparable to the original ball joint system.
- Avoid locking, interference, or jamming across the full motion range.

5.3 Conceptual Designs

This section will investigate different concepts for an alternative Stewart Platform that were designed by the team. A brief description of each idea will be expanded upon as well as the decision process to determine the most appropriate option.

5.3.1 Pulley System

One proposed idea is to replace the ball joints with a pulley and cable system. Three cables are attached to the Stewart Platform equally spaced apart. These cables are fed through pulleys that are mounted on frames that suspend the platform. The three servo motors are at the base of the frame corresponding to each pulley. They move the platform by pulling or releasing the cables to raise and lower the sides of the platform as needed.

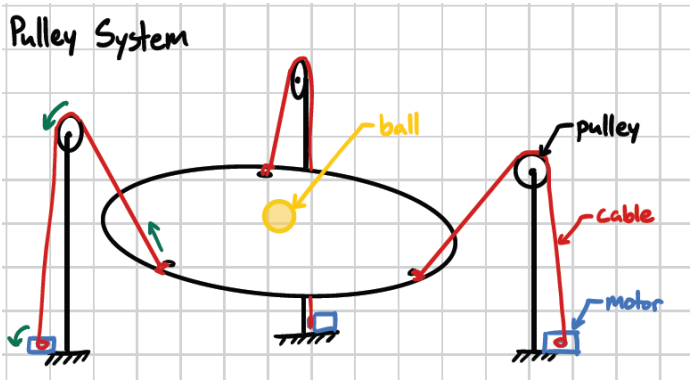


Figure 21: Concept of Pulley Based Stewart Platform

- Pros:
  - Ease of manufacturing
  - Simple
- Cons:
  - Not as Robust
  - Slower response as longer distance motor needs to travel for equivalent tilt

5.3.2 Magnetic Levitation

The proposed idea is to replace mechanical linkages and servo motors entirely with magnets. The bottom of the Stewart Platform would contain permanent magnets while the base would have electromagnets that align with the permanent magnets. The arrangement could be three magnets up to an entire ring of magnets around the perimeter. The platform would be controlled by adjusting the strength of the magnetic field at different locations along the perimeter.

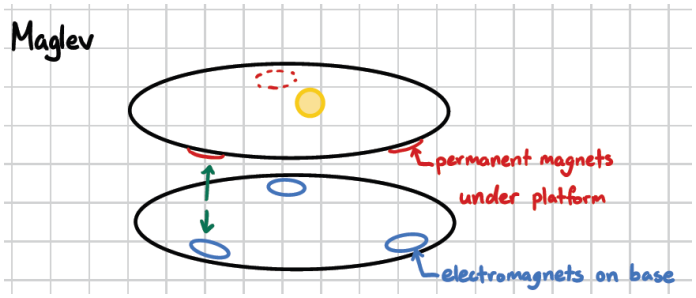


Figure 22: Concept of Magnet Based Stewart Platform

- Pros:
  - Mechanically simple
  - No friction
- Cons:
  - Incredibly complex to implement even balancing the plate itself, much less the ball
  - Precise control and balance would be very difficult

5.3.3 2 Axis with Frame Mounted Motor and Base Mounted Motor

This design has an outer ring that rotates about one axis. The platform is concentric inside the outer ring and supported by a motor and a pin joint mounted on the outer ring, allowing it to rotate about the perpendicular axis.

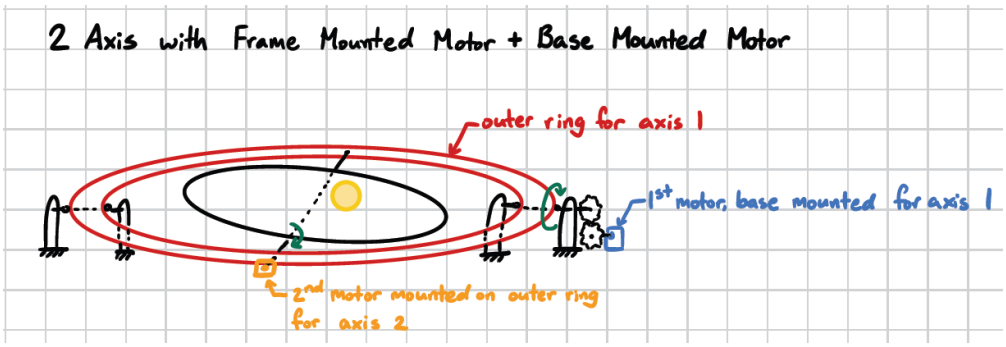


Figure 23: Concept of 2 Axis with Frame Mounted Motor and Base Mounted Motor Stewart Platform

- Pros:
  - Simple to design and understand
  - Control algorithm would be simpler
- Cons:
  - Weight of Motor mounted on outer ring would throw balance off
  -

5.3.4 2 Axis with Both Motors Base Mounted

This proposal builds off the previous 2 Axis concept, addressing the concerns that the weight of the servo motor would offset the balancing of the platform due to the misaligned weight distribution. This idea has two variations. The first is stacking two platforms on top of each other, one for each

perpendicular axis. The second one has a similar outer ring and nested platform. Both motors in these designs are base mounted. This can be achieved by a system of shafts and gears to apply the rotation at the second axis hinge whose position varies depending on the input angle about the first axis. This implementation may also require nested concentric shafts to translate the motion from both of the motors along the same path to rotation about the second axis.

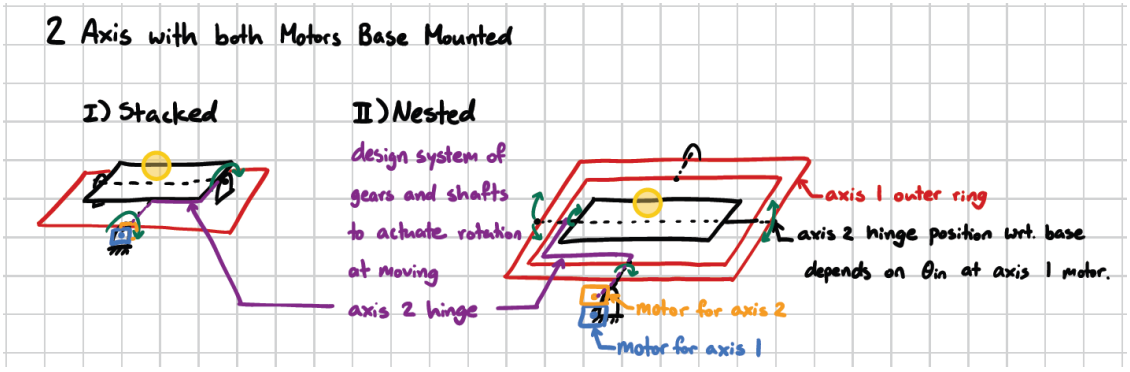


Figure 24: Concept of 2 Axis with Both Motors Base Mounted Stewart Platform

- Pros:
  - Location of motors is convenient and accessible.
  - Weight of motors does not affect balance of platform.
- Cons:
  - Very complex mechanism required to implement

5.3.5 Parallel Spherical Manipulator

This idea uses multiple independent linkages that are connected in a parallel axis to control the precise rotation of a single moving platform along a common and fixed central point. All components are designed in a way that any movement results in the end point pivoting around a specific point in space. This point acts as the origin for roll, pitch and yaw.

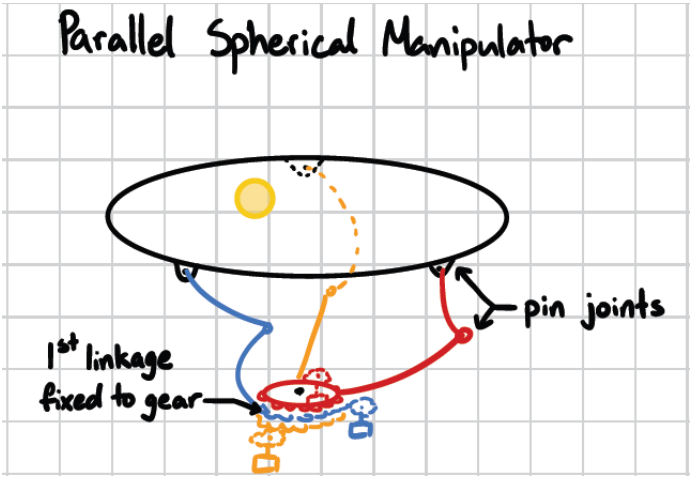


Figure 25: Concept of Parallel Spherical Manipulator Stewart Platform

- Pros:

  - Compact Size
  - Very Precise control and movement
  - Locations of motors are accessible.
  - Motors do not need to provide torque when platform is flat
- Cons:
  - Complicated Kinematics.
  - Geometry of linkages needs to fit certain mathematical parameters.

5.3.6 Slotted Connection

This idea involves four slotted connections that replaced the three ball joints connected to the platform. Each axis would have one servo motor and one pin joint. Both the motor and the pin joint connect to linkage that slides along the slot and has a stopper above and below the platform to keep it at a certain position along the linkage. The angle of the linkage controls the height of the contact point between the platform and the linkage. This system is implemented along two perpendicular axis for two dimensional control of the platform. This allows it to tilt about each axis

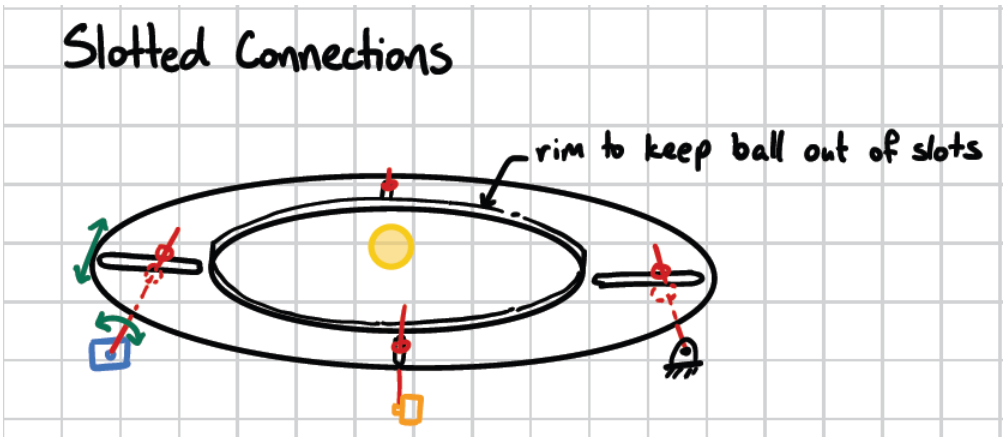


Figure 26: Concept of Slotted Connector Stewart Platform

- Pros:
  - Ease of manufacturing.
  -
- Cons:
  - Friction resulting in wear of the parts.
  - Oversized Mechanism.
  - Lack of precision as by necessity there are large clearances in the parts.

5.4 Designing a Spherical Parallel Manipulator

The spherical parallel manipulator (SPM) is chosen as the final solution for a design that does not involve ball joints. This solution presents a concept that is the most viable for design and assembly using 3D printed parts and basic servo motors. It also presented the most robust design out of all other solutions, as it purely consists of mechanical pin joints that significantly reduces the slop seen on the Stewart platform. Finally, all of the SPM designs have motors placed in a manner such that during times when the platform is not tilting, the motor would experience less gravity loading

from the ball and platform, which means motors are not required to constantly hold a large torque. This significantly reduces the power requirement of the motor, and reduces consideration for back voltage generated by the motor, therefore leading to a simpler electrical design.

5.4.1 First Iteration

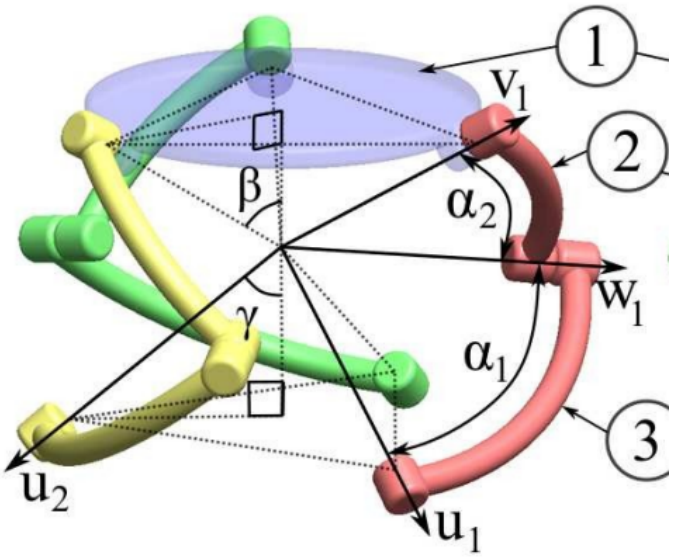


Figure 27: Concentric Sphere SPM Design [6]

The first iteration consists of linkage designs that are based on concentric spheres. As a requirement for an SPM, if a straight axis were to pass through each pin joint, the 3 pin joints of each linkage must meet at the same point. Since the 3 motors are located at varying positions of the assembly, it created a challenge to adjust link geometry to align the joint axis to the same point. The curved link also created a challenge for assembling the platform.

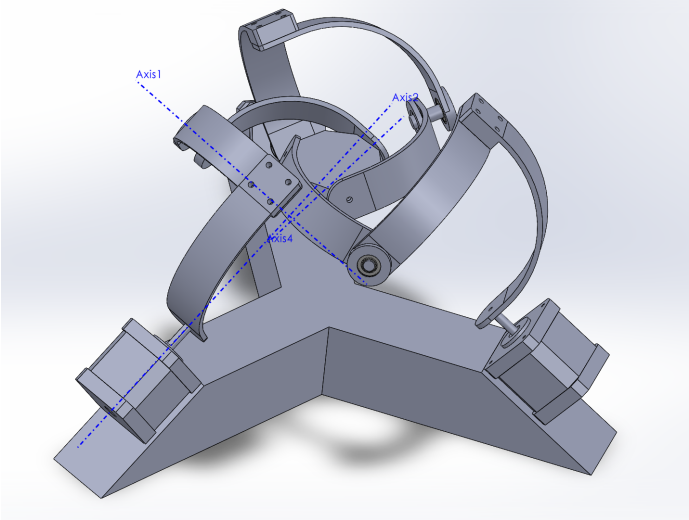


Figure 28: SPM Iteration 1 SolidWorks Model Inspired by [7]

5.4.2 Second Iteration

The second iteration of SPM features a co-axial input joint axis design ($\gamma = 0$). Inspired by [8], such design enables easier alignment of all pin joint axes. Mounting inputs has also been simplified due to the coaxial input link.

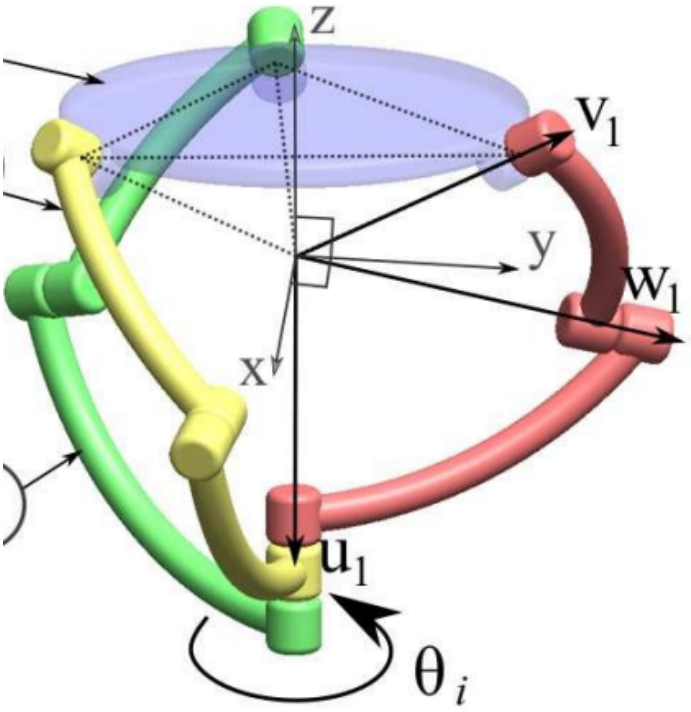


Figure 29: Co-axial input joint SPM Design [6]

In the group's original design, all base links extend upwards to reach the same height at the location of the pin joint. This enables three identical copies to be used for the second link, since all 3 base links will now have the same distance to the platform. All base links are angled 45 degrees to point towards the centre axis. This way, the pin joint axes for joints on the base link intersect at one point with the input centre axis.

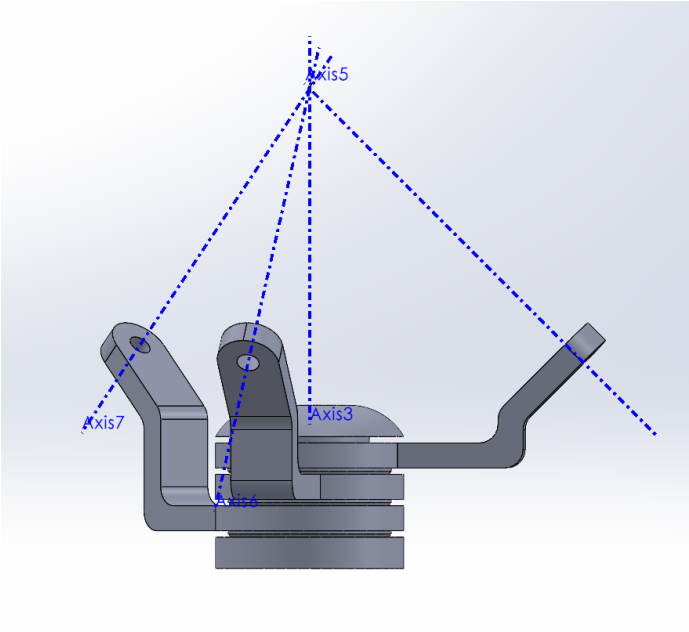


Figure 30: Intersecting Axis for Iteration 2 SPM Design

Link 2 of the SPM is a 90 degree L shaped link. Its lengths must be specifically selected based on the geometry of the rest of the platform. The horizontal length of the L link is dependent on the radius of the circular platform. The length of the horizontal part should allow all 3 instances of L link to connect to both the base link joint, as well as the joint on the bottom of the platform. If such distance is not calibrated correctly, assembly would be impossible as joint connections on the platform are compromised

Vertical length of L link must be selected as the exact length that allows the revolve axis of all pin joints between link 2 and platform to intersect at the same point. This intersection point must also be the same point as where the 3 pin joint axes at the base link intersect. If the link length is correctly adjusted, all axes on a single link could intersect.

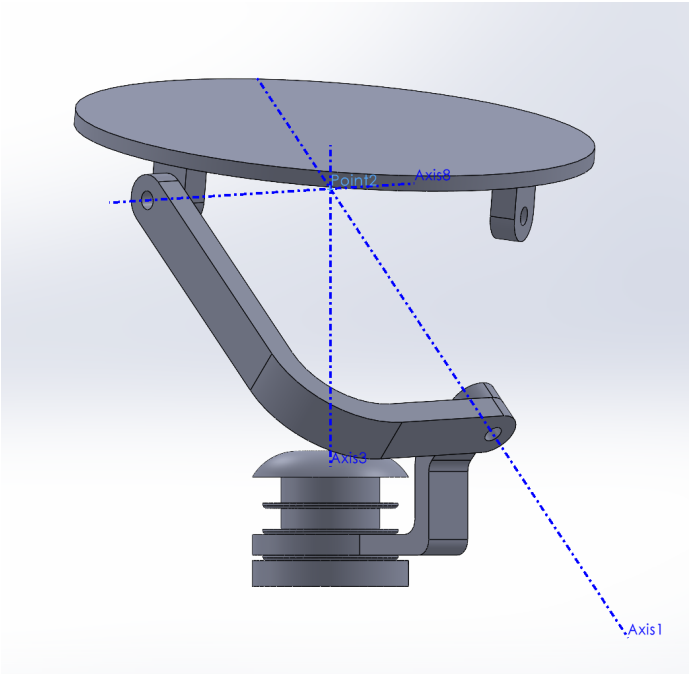


Figure 31: All Axis on a Single Link Intersecting

The intersection point of all pin joint axes will always be the same with respect to the base. The platform rotates about this point in space, and the centre of the platform will always be a fixed distance from this point. Figure below demonstrates the design and current dimensions of the L link.

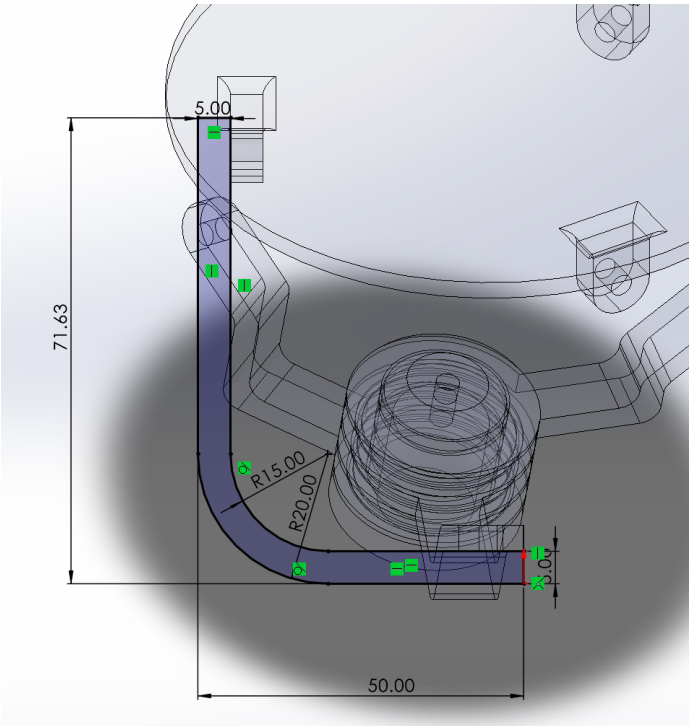


Figure 32: Dimension of Link L

Figure 33 below shows the interaction of all links.

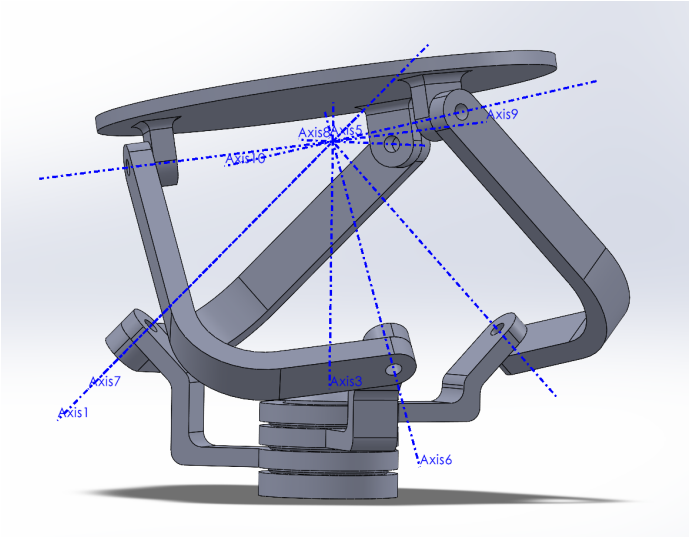


Figure 33: Intersection of All Links in Final Assembly

If the length of the vertical part of link is not chosen correctly, assembly could be possible, but movement of the platform will be restricted to only one DOF (rotating about the centre axis), as the rotation of the base input links with respect to each other will be constrained. Typically this results in the base links rotating to very small angles with each other. In the CAD model, the constraints cause the joint to overlap with each other, which would mean part interference in real life. Figure 34 below demonstrates the interaction.

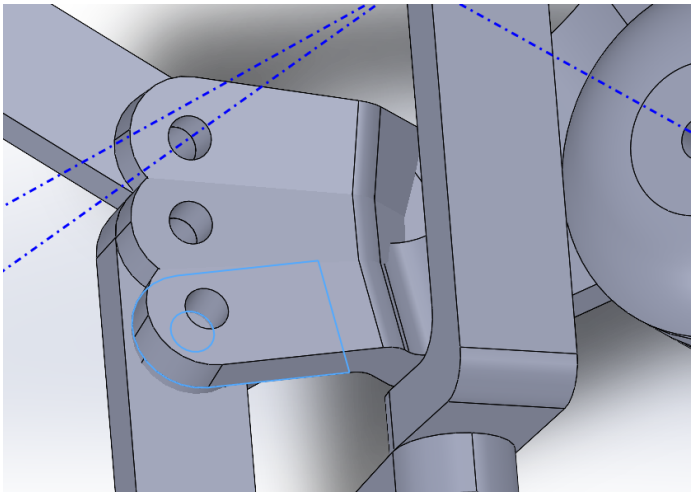


Figure 34: Base Linkages Intersecting

5.5 Takeaways and Final Prototype

The main take away of SPM design is that all pin joint's rotating axes for a platform must converge at the same point. This feature is present in all designs found in current research; the only difference is the number of the axes required to meet at the same point, depending on whether coaxial input or separate axis input is used. If this feature is not met, assembly would either be impossible, or the movement of the platform will be severely constrained.

Figure 35 below displays the final assembly of the SPM. The platform has the advantage of being able to support a load even when in a neutral position with no motor power input. The current prototype is able to perform smooth movements even without the use of bearings, which demonstrates the robustness of the platform design.

Figure 35: Final 3D-Printed and Assembled Spherical Parallel Manipulator

6 Summary

6.1 Conclusion

In the development of this Stewart platform, questions (6): Eliminating Ball Joints and question (10): Effect of a 100 times heavier ball were answered. The platform was successfully redesigned to accommodate the more than 100 times heavier ball. New servo motors were selected and the linkage was adjusted to accommodate the platform modifications. The alternative sensing method using the ForceN force-torque sensor integrated into the platform allowed the ball to balance on the platform. Overall, the project was a success and the team successfully demonstrated and presented the finalized platform on "Game Day" November 26th, 2025.

6.2 Future Improvements

Several opportunities exist to strengthen the sensing, estimation, and reference generation components of the platform beyond the current implementation.

At present, the platform relies on a steel ball nearly two orders of magnitude heavier than the original ping-pong ball to generate sufficiently large normal forces and tipping moments for reliable position inference. This constraint arises because the loadcell alone cannot distinguish between true changes in ball position and those induced by platform's own inertial acceleration.

By incorporating a accelerometer into the estimation framework, the inertial forces produced during platform motion can be measured directly and removed from the loadcell moments prior to computing the position estimate. As the signal-to-noise ratio of the corrected moment measurements increases, the platform no longer requires a heavy ball to overcome sensor noise and structural dynamics. In principle, this would allow the system to operate with much smaller or lighter balls, which would allow for far easier control dynamics without compromising the position estimation.

Next, the trajectory planning framework may be extended beyond the Simulink study. As demonstrated in simulation, fixed and periodically updated trajectories produce smoother control effort, reduced overshoot, and significantly improved disturbance rejection compared to direct step commands. Implementing a real time trajectory generator on the physical platform would allow the controller to command dynamically feasible plate motions with bounded acceleration and jerk, thereby reducing the burden on the servos under the heavier-ball dynamics. Coupling this with replanning would provide a principled strategy for recovering the ball from large disturbances or from positions near the platform edge where the ball's momentum is the highest. This is reliant on the acceleration compensation because otherwise as mentioned the derivative of position readings (velocity) is too noisy to be useful to the trajectory planning.

References

[1] D. Stewart, "A platform with six degrees of freedom," *Proceedings of the Institution of Mechanical Engineers*, vol. 180, no. 1, 1965.

[2] L.-W. Tsai, *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*. Wiley, 1999.

[3] S. Bedi, *Mte 380: Mechatronics engineering design workshop – course introduction slides*, Lecture slides, Course introduction lecture slides, 2025. [Online]. Available: https://learn.uwaterloo.ca/d2l/le/content/1173425/viewContent/6134947.

[4] N. Khaled, *Ball on plate simscape model: Control and diagnostics*, https://www.mathworks.com/matlabcentral/fileexchange/74331-ball-on-plate-simscape-model-control-and-diagnostics, 2020.

[5] G. di Pasquo, "Sg90 servo characterization," Aug. 2021. DOI: 10.13140/RG.2.2.15715.89127.

[6] I. Tursynbek, A. Niyetkaliye, and A. Shintemirov, "Computation of unique kinematic solutions of a spherical parallel manipulator with coaxial input shafts," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, Aug. 2019, pp. 1524–1531. DOI: 10.1109/COASE.2019.8843090.

[7] e. a. Djennane, "Design, analysis, and implementation of a 3-dof spherical parallel manipulator," *IEEE Access*, 2024.

[8] PLUSALPHADESIGNS, *Spherical parallel base*, https://cults3d.com/:859232, 2022.

A CAD Models

A.1 CAD Image

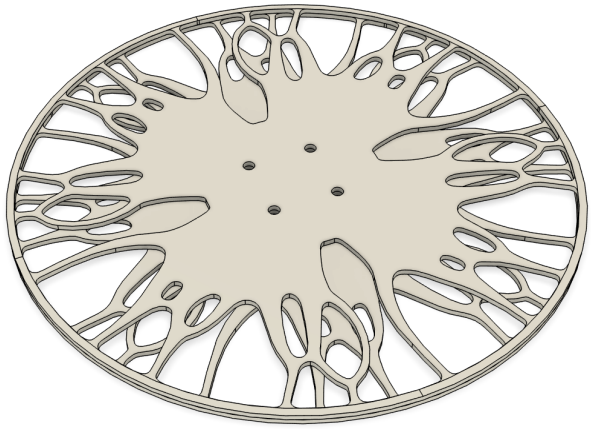


Figure 36: The three topology optimized plates stacked

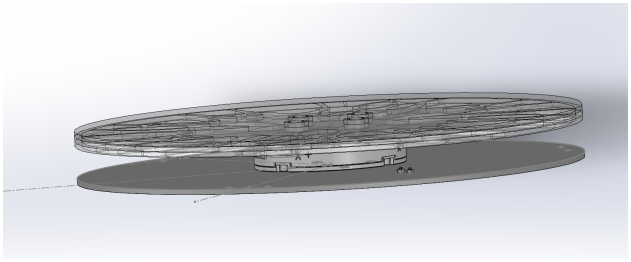


Figure 37: The three topology optimized plates with sensor

The redesigned platform show below should be treated as a graphical representation of the as-built platform, as it still uses stepper motors rather than servos. The team deemed it unnecessary to port the new servo mount/base design to CAD, as swapping the motors does not affect the inverse kinematics, due to Link 1 being mounted at the same position in either case.

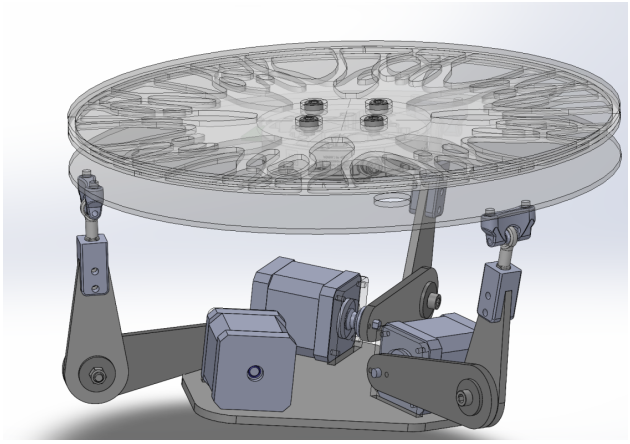


Figure 38: The redesigned platform

B Mechanical Analysis and FEA

B.1 Comparative FEA Between Plate Iterations

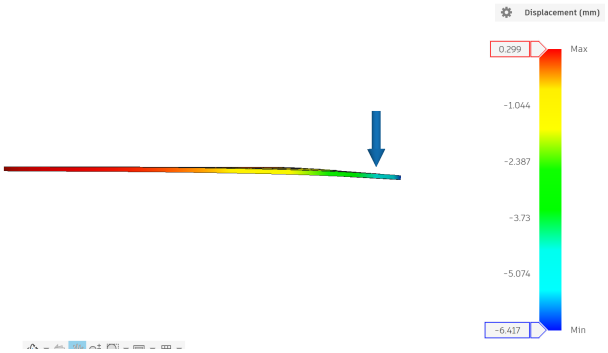


Figure 39: View 1: Standard Optimization Deflection

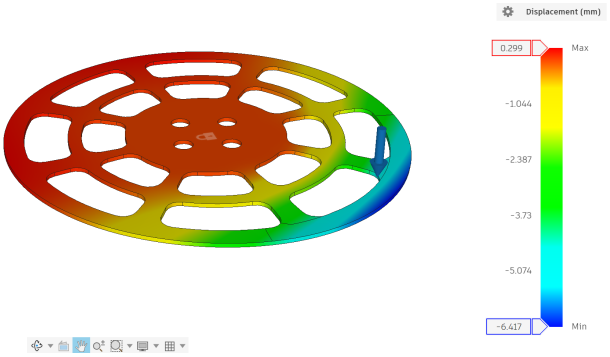


Figure 40: View 2: Standard Optimization Deflection

Figure 41: View 1: Topology Optimized Plate

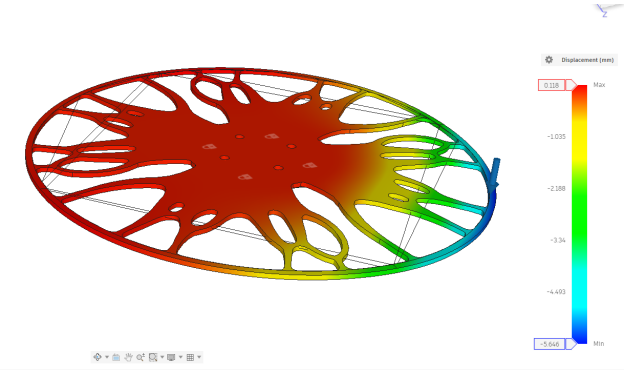


Figure 42: View 2: Topology Optimized Plate

C Servo Power Disspation Network Calculations

This appendix summarizes the calculations used to size the capacitor, dump resistor, and isolation diode used in the regenerative protection circuit for the Studica servos. The objective is to absorb short regenerative current pulses without allowing the 6 V bus to experience excessive over-voltage, and to dissipate the stored energy safely between events.

C.1 Servo Operating Conditions

Each Studica servo (that operates at 6V) has a stall current on the order of 1.8 A, with typical running currents in the 0.15 to 1 A range depending on load. With three servos installed, a conservative estimate for short regenerative events is a combined current pulse of approximately

$$I_{\text{regen}} \approx 3 \text{ A}$$

over a duration of roughly

$$\Delta t \approx 10 \text{ ms} = 0.01 \text{ s}.$$

The nominal supply voltage is

$$V_{\text{bus}} \approx 6 \text{ V}.$$

C.2 Capacitor Sizing

When the platform is back-driven by the heavy ball, the servos act as generators and inject current into the DC bus. The bus capacitor must absorb this current without allowing the voltage to rise excessively. For a current pulse of magnitude I and duration Δt, the bus voltage change ΔV across a capacitor C is approximated by

$$\Delta V = \frac{I\,\Delta t}{C}. \tag{9}$$

Using a worst-case regenerative current of $I_{\text{regen}} = 3$ A over $\Delta t = 0.01$ s and an original 680 μF capacitor yields

$$C_{\text{old}} = 680\ \mu\text{F} = 680 \times 10^{-6}\ \text{F}, \tag{10}$$

$$\Delta V_{\text{old}} = \frac{3 \cdot 0.01}{680 \times 10^{-6}} \approx 44\ \text{V}, \tag{11}$$

which is clearly unacceptable and would immediately trip the bench supply.

For the upgraded capacitor

$$C_{\text{new}} = 4700\ \mu\text{F} = 4.7 \times 10^{-3}\ \text{F},$$

the same worst-case pulse gives

$$\Delta V_{\text{new}} = \frac{3 \cdot 0.01}{4.7 \times 10^{-3}} \approx 6.4\ \text{V}. \tag{12}$$

This represents a substantial improvement in energy buffering. In practice the regenerative current is not a perfect square pulse and the diode isolates the bench supply so that most of the energy is absorbed by the capacitor and subsequently dissipated in the dump resistor.

The corresponding energy stored in the bus capacitor at 6 V is

$$E = \frac{1}{2}CV^2. \tag{13}$$

For the 4700 μF capacitor:

$$E_{\text{new}} = \frac{1}{2} \cdot 4.7 \times 10^{-3} \cdot 6^2 \approx 0.0846 \text{ J}, \tag{14}$$

compared to

$$E_{\text{old}} = \frac{1}{2} \cdot 680 \times 10^{-6} \cdot 6^2 \approx 0.0122 \text{ J} \tag{15}$$

for the original 680 μF capacitor. Thus, the new capacitor provides roughly a factor of seven increase in energy storage capacity.

C.3 Dump Resistor Sizing

The energy absorbed by the capacitor must be dissipated between regenerative events to prevent the bus voltage seen by the power supply from drifting upwards. This is accomplished using a 10 Ω dump resistor in parallel with the capacitor. The instantaneous power dissipated in the resistor at voltage V is

$$P = \frac{V^2}{R}. \tag{16}$$

At the nominal 6 V bus voltage and $R = 10\ \Omega$,

$$P_{\text{avg}} = \frac{6^2}{10} = 3.6 \text{ W}. \tag{17}$$

Regenerative events are short, so peak power during a pulse can be significantly higher than this average. To provide adequate thermal margin for repeated pulses and continuous operation, a 100 W aluminum-housed power resistor was selected. This allows the resistor to tolerate momentary overloads without overheating while still operating at modest temperature rise under typical conditions.

C.4 Discharge Time Constant

The resistor–capacitor pair forms a first-order discharge network with time constant

$$\tau = RC. \tag{18}$$

Using $R = 10\ \Omega$ and $C = 4700\ \mu\text{F} = 4.7 \times 10^{-3}$ F,

$$\tau = 10 \cdot 4.7 \times 10^{-3} = 0.047 \text{ s}. \tag{19}$$

A standard rule of thumb is that after approximately 3τ the capacitor is largely discharged:

$$3\tau \approx 0.14 \text{ s}. \tag{20}$$

Thus, the bus voltage settles back toward its nominal value within roughly 150 ms after a regenerative event, which is fast enough for repeated servo movements without imposing a continuous heavy load on the power supply.

C.5 Diode Selection

A Schottky diode is placed between the bench supply and the servo bus to block reverse current from the servos into the supply. The diode must tolerate the maximum expected forward current of all three servos combined, as well as the peak regenerative current. A 15 A, 45 V part (BOJACK 15SQ045) was therefore selected, providing a large margin over the nominal servo currents and the 6 V operating voltage. The low forward voltage of the Schottky minimizes additional voltage drop seen by the servos as well.

D Simulink Models and Controller Block Diagrams

D.1 Simulink Model

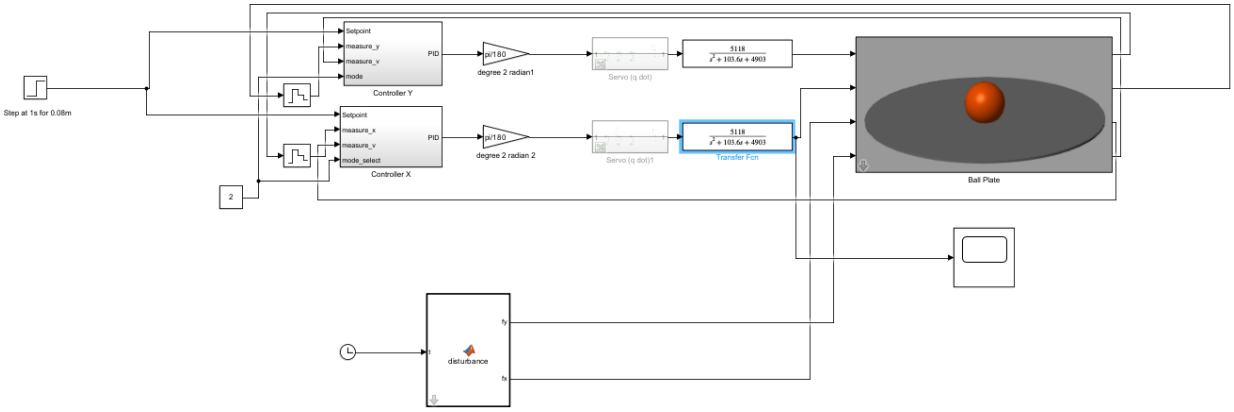


Figure 43: The outermost layer of simulink model, with all subsystems masked

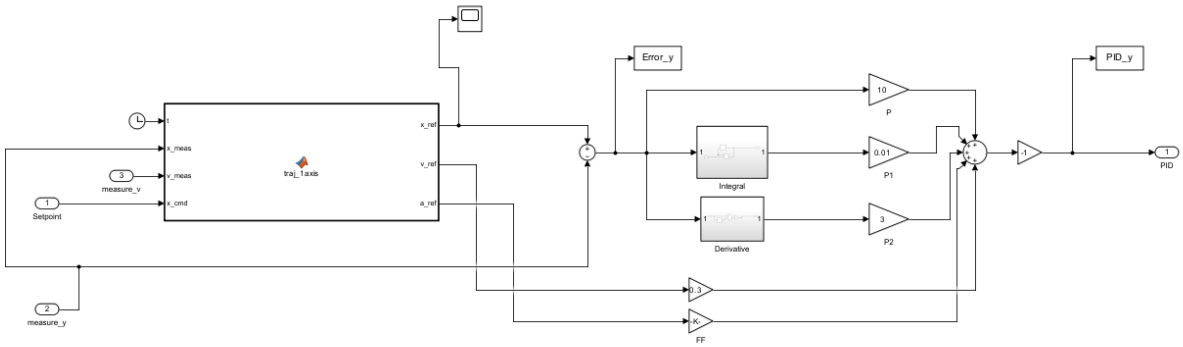


Figure 44: Controller block diagram with trajectory generation

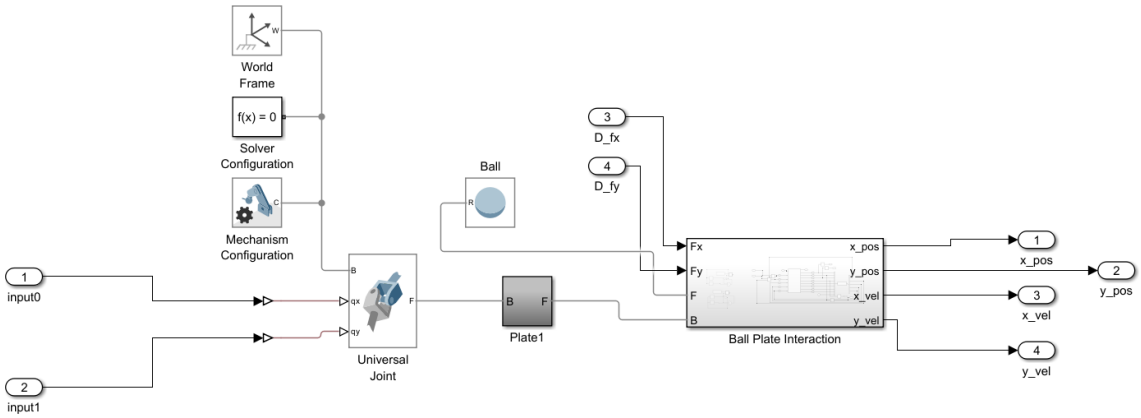


Figure 45: Plant model with plate tilt model

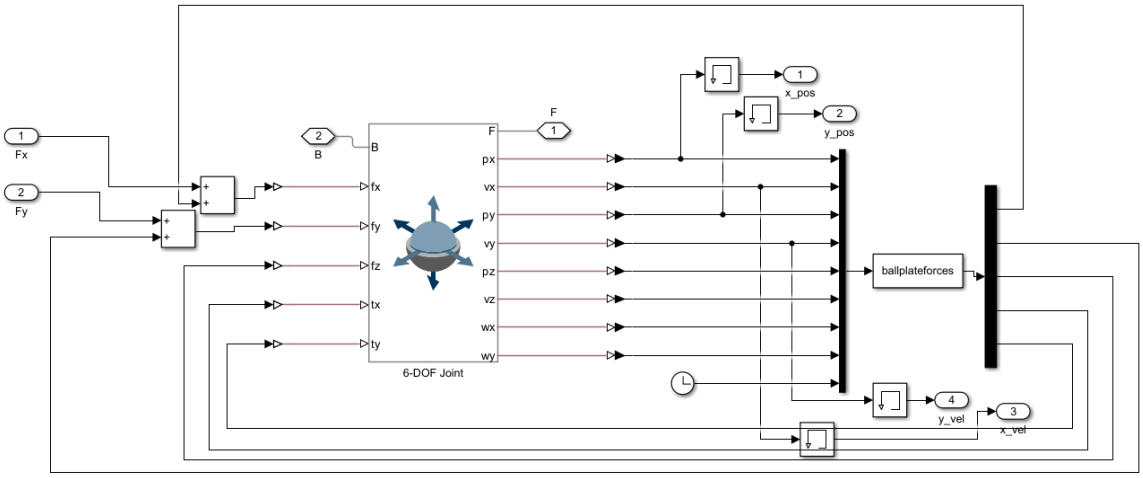


Figure 46: Ball-plate interaction layer

E Additional Figures and Extended Results

E.1 Actuator Specs

Specification	**Studica High-Torque Servo**	**Goteck GB3506MG (35 kg)**
Motor Type	Metal Brush Motor	Brushless Digital Motor
Gear Ratio	1:67	Not specified
Gear Type	Steel	Stainless Steel Geartrain
Bearing	Dual Ball Bearings	Dual Ball Bearings
Case	Nylon & Fiberglass	Metal/Composite
Spline	25T	25T
Dimensions	40 × 20.1 × 38.3 mm	40 × 20 × 40.5 mm
Weight	64 g	65 g
Voltage Range	4.8–6.0 V	6.0–8.4 V
Idle Current	5 mA (4.8 V) / 7 mA (6.0 V)	Not listed
Running Current	130 mA (4.8 V) / 150 mA (6.0 V)	Not listed
Stall Torque	180.8 oz-in (4.8 V) 300 oz-in (6.0 V)	30 kg·cm (6.0 V) 35 kg·cm (8.4 V)
Stall Current	1.5 A (4.8 V) / 1.8 A (6.0 V)	Very high (not specified)
No-Load Speed	0.25 s/60° (4.8 V) 0.20 s/60° (6.0 V)	0.09 s/60° (6.0 V) 0.06 s/60° (8.4 V)
Rotation Range	300° ± 5°	180°
Wire Gauge / Length	22 AWG / 70 cm	Standard / Not listed
Environmental	Operates -15°C to 70°C	Not listed

Table 4: Comparison of actuator specifications between the Studica servo and the Goteck 35 kg brushless servo.

E.2 Servo Modelling

A second-order approximation is commonly used to model hobby servos. di Pasquo's experimental characterization of the SG90 servo states that "the servo is ideally a second order system" and provides identified transfer functions of the form [5]

$$G(s) = \frac{K}{s^2 + \alpha s + \beta}.$$

From the simplified SG90 model for a 10° step,

$$G_{\text{SG90}}(s) = \frac{5118}{s^2 + 103.6s + 4903}.$$

The natural frequency and damping ratio are obtained directly from the denominator:

$$\omega_{n,\mathrm{SG90}} = \sqrt{4903} \approx 70 \text{ rad/s}, \qquad \zeta_{\mathrm{SG90}} = \frac{103.6}{2\,\omega_n} \approx 0.74.$$

the damping ratio of 0.7 were a fair representation for the Studica servo, as it represent a well damped system. Therefore, for this project, the SG90-derived dynamics are used directly as a generic second-order servo approximation. [5]

$$G_{\mathrm{studica}}(s) = \frac{5118}{s^2 + 103.6\,s + 4903}.$$

However, this proved to be quite unrealisitic. Compared to the Studica servo, the SG90 exhibits a substantially higher no load speed (approximately $0.10\,\mathrm{s}/60^\circ$ versus $0.20\,\mathrm{s}/60^\circ$), hence the plate were able to tilt much faster than the physical platform, which significantly degrades the necessity of trajectory planning. Therefore, to emulate a more real life scenario, each servo TF output passes through a rate limter to constrain how fast the platform can tilt.

F Source Code Excerpts

F.1 Forcen Interface: Ball Position Estimation

Listing 1: Key routines for force and moment based ball position estimation in ‘ForcenInterface‘

```
import math

class ForcenInterface:
    def __init__(self, platform_radius=150.0, fz_min=4000.0, fz_max=8000.0,
                 filter_alpha=0.9, rotation_angle=135.0,
                 scale_x=1.1, scale_y=1.1, flip_x=False, flip_y=True):
        # Transformation parameters
        self.platform_radius = platform_radius
        self.fz_min = fz_min
        self.fz_max = fz_max
        self.filter_alpha = filter_alpha

        self.offset_x = 0.0
        self.offset_y = 0.0
        self.scale_x = scale_x
        self.scale_y = scale_y

        self.rotation_angle = rotation_angle
        self.rotation_rad = rotation_angle * math.pi / 180.0
        self.flip_x = flip_x
        self.flip_y = flip_y

        # State for filtering
        self.last_valid_position = (None, None)
        self.filtered_x = None
        self.filtered_y = None

    # ------------------------------------------------------------------ #
    # 1) Raw position from moments and normal force
    # ------------------------------------------------------------------ #
    def _estimate_ball_position_raw(self, data):
        if not data or 'Mx' not in data or 'My' not in data or 'Fz' not in data:
            return (None, None)

        Mx = data['Mx'] # Nmm
        My = data['My'] # Nmm
        Fz_mN = data['Fz'] # mN

        if abs(Fz_mN) < 10.0:
            return (None, None)

        Fz_N = Fz_mN / 1000.0 # convert to N

        # Mx = Fz * y, My = Fz * x
        y = Mx / Fz_N # mm
        x = My / Fz_N # mm
        return (x, y)

    # ------------------------------------------------------------------ #
    # 2) Filtering and validity checks (radial bound + Fz band)
    # ------------------------------------------------------------------ #
    def _estimate_ball_position_filtered(self, data):
```

```
53          x_raw, y_raw = self._estimate_ball_position_raw(data)
54          if x_raw is None or y_raw is None:
55              return self.last_valid_position
56
57          current_fz = data.get('Fz')
58          if current_fz is None:
59              return self.last_valid_position
60
61          distance = math.sqrt(x_raw**2 + y_raw**2)
62          if distance > self.platform_radius:
63              return self.last_valid_position
64
65          if current_fz < self.fz_min or current_fz > self.fz_max:
66              return self.last_valid_position
67
68          # Exponential moving average
69          if self.filtered_x is None or self.filtered_y is None:
70              self.filtered_x = x_raw
71              self.filtered_y = y_raw
72          else:
73              self.filtered_x = (self.filter_alpha * x_raw +
74                                 (1.0 - self.filter_alpha) * self.filtered_x)
75              self.filtered_y = (self.filter_alpha * y_raw +
76                                 (1.0 - self.filter_alpha) * self.filtered_y)
77
78          self.last_valid_position = (self.filtered_x, self.filtered_y)
79          return (self.filtered_x, self.filtered_y)
80
81      # ------------------------------------------------------------------ #
82      # 3) Apply offsets, scaling, rotation and axis flips
83      # ------------------------------------------------------------------ #
84      def estimate_ball_position(self, data):
85          x_raw, y_raw = self._estimate_ball_position_filtered(data)
86          if x_raw is None or y_raw is None:
87              return (None, None)
88
89          # Center offset in sensor frame
90          x_centered = x_raw - self.offset_x
91          y_centered = y_raw - self.offset_y
92
93          # Radial scale factors
94          x_scaled = x_centered * self.scale_x
95          y_scaled = y_centered * self.scale_y
96
97          # Rotation from sensor frame to platform frame
98          cos_theta = math.cos(self.rotation_rad)
99          sin_theta = math.sin(self.rotation_rad)
100         x_rot = x_scaled * cos_theta - y_scaled * sin_theta
101         y_rot = x_scaled * sin_theta + y_scaled * cos_theta
102
103         # Optional axis flips
104         if self.flip_x:
105             x_rot = -x_rot
106         if self.flip_y:
107             y_rot = -y_rot
108
109         return (x_rot, y_rot)
110
111     def get_weight(self, data):
```

```
112         return data.get('Weight', 0.0)
```

F.2 Controller Usage of Force-Based Position

Listing 2: Excerpt from main control loop using the Forcen-based position estimate

```
class BallBalancingController:
    def run(self, update_rate=0.05):
        dt = update_rate
        last_time = time.time()

        while True:
            loop_start = time.time()

            # 1. Read Forcen frame
            data = self.sensor.read_data()
            if data:
                # 2. Estimate ball position from forces and moments
                x, y = self.sensor.estimate_ball_position(data)
                weight_g = self.sensor.get_weight(data)

                # (Optional) use CV instead:
                # x, y, weight_g = self.read_cv_ball_position()

                if x is not None and y is not None and weight_g > 200.0:
                    # 3. Position error relative to setpoint
                    error_x = self.setpoint_x - x
                    error_y = self.setpoint_y - y

                    # Deadband: treat small errors as zero
                    error_mag = np.sqrt(error_x**2 + error_y**2)
                    if error_mag < 5.0:
                        error_x = 0.0
                        error_y = 0.0

                    # 4. PID control in each axis (not shown)
                    tilt_x = self.pid_x.update(error_x, dt)
                    tilt_y = self.pid_y.update(error_y, dt)

                    # 5. Convert to platform leg lengths and servo commands
                    servo_angles = self.platform.inverse_kinematics(
                        tilt_x, tilt_y, self.platform_height
                    )
                    self.servos.set_angles(servo_angles)

            # Enforce loop timing
            sleep_time = update_rate - (time.time() - loop_start)
            if sleep_time > 0:
                time.sleep(sleep_time)
```

F.3 Trajectory Generation

The trajectory generator `traj_1axis.m` produces reference position, velocity, and acceleration profiles for a single axis of the ball. It supports three operating modes that differ in how they form the reference signal:

- **Mode 1 — Direct Step:** The reference is set directly to the commanded position, with no trajectory shaping.
- **Mode 2 — One-Shot Trajectory:** A dynamically feasible trapezoidal or triangular trajectory is generated whenever the command changes or when the planner is idle and the ball has drifted away from the target.
- **Mode 3 — Limited Replanning:** A trajectory is regenerated only when the system is genuinely diverging from the command and a minimum time has elapsed.

The generator uses persistent variables to store the start and end states, segment distances, peak velocity, and the computed switching times for each phase. A complete motion consists of an accelerate–cruise–decelerate sequence for a trapezoidal profile or accelerate–decelerate for a triangular one.

Replanning Logic

Replanning is triggered when there is a significant change in the command, when the system is idle and off-target, or under Mode 3: when the ball is observed to be moving away from the desired position and enough time has elapsed since the previous plan:

```
cmd_changed = abs(x_cmd - last_x_cmd) > 1e-6;
disturbance = (mode == 0) && (abs(x_meas - x_cmd) > dist_eps);

moving_away = ((x_meas - x_cmd) * v_meas) > 0;
far_enough = abs(x_meas - x_cmd) > replan_dist_factor * dist_eps;

if mode_select == 3
    periodic_replan = (t - last_replan_time >= replan_T) ...
                      && moving_away && far_enough;
else
    periodic_replan = false;
end

do_replan = cmd_changed || disturbance || periodic_replan;
```

When replanning is triggered, the new trajectory is constructed from the current measured state. The generator normalizes motion direction, defines the start state (x_b, v_b), sets the desired end state (x_d, v_d), computes the maximum allowable velocity, and determines whether a trapezoidal or triangular trajectory is required.

Trajectory Classification

```
S1 = (v_eff_max^2 - vb_s^2) / (2 * a_max);
S3 = (vd_s^2 - v_eff_max^2) / (2 * a_min);

if dist > (S1 + S3)
    mode = 1; % Trapezoid
    S2 = dist - S1 - S3;
else
    mode = 2; % Triangle
    V = sqrt((2*a_min*a_max*dist + a_max*vb_s^2 - a_min*vd_s^2) ...
             /(a_max - a_min));
end
```

The switching times t_1, t_2, t_3 follow directly from the phase distances and known accelerations.

Trajectory Evaluation

At every call, the trajectory is evaluated at the current time to determine the instantaneous reference:

```
if mode == 1
    % Trapezoid: accel -> cruise -> decel
    if t < t1
        a = a_max;
        v = vb_s + a_max * dt;
        s = xb_s + vb_s*dt + 0.5*a_max*dt^2;
    elseif t < t2
        a = 0;
        v = V;
        s = xb_s + S1 + V*(t - t1);
    elseif t < t3
        a = a_min;
        v = V + a_min*(t - t2);
        s = xb_s + S1 + S2 + ...QD
            V*(t - t2) + 0.5*a_min*(t - t2)^2;
    else
        s = xd_s; v = vd_s; a = 0; mode = 0;
    end
end
```

The triangular case follows the same structure but omits the constant velocity phase. After evaluating the motion in a sign normalized coordinate system, the original direction is restored:

$$x_{\text{ref}} = sgn \cdot s, \qquad v_{\text{ref}} = sgn \cdot v, \qquad a_{\text{ref}} = sgn \cdot a.$$

Direct Step Mode

Mode 1 bypasses trajectory generation entirely and exposes the raw command as the reference:

```
if mode_select == 1
    mode = 0;
    x_ref = x_cmd;
    v_ref = 0;
    a_ref = 0;
    return;
end
```

G Real-Time Control Software Implementation

This appendix documents the Python implementation of the real-time controller used on the Stewart platform. The codebase is organised into the following modules:

- `controller.py`: main application entry point, containing the PID implementation and the high-level `BallBalancingController` class that runs the control loop.
- `ForcenInterface.py`: serial interface to the Forcen ASY-00035 loadcell, including packet parsing, calibration utilities, and conversion of (M_x, M_y, F_z) into an estimated planar ball position.
- `SPV4.py`: geometric model of the three-legged Stewart platform, providing `StewartPlatform.inverse_kinem` to map a desired plate normal vector and height to three actuator angles.

- `ServoController.py`: low-level interface that formats the desired servo angles and sends them over serial to the microcontroller.
- `viz_tuner.py`: Tkinter-based GUI for online tuning; writes updated PID gains to a binary file read periodically by `controller.py`.

G.1 PID Controller Class

The core feedback law is implemented by the `PIDController` class in `controller.py`. The constructor exposes the gains and key implementation parameters:

- K_p, K_i, K_d: proportional, integral, and derivative gains.
- K_{ff} (named `pf` in the code): velocity feedforward / feedback gain.
- `output_limit`: absolute saturation limit on the controller output (deg).
- `derivative_filter_coeff`: first-order low-pass filter coefficient for the derivative
- `max_kp_tilt`: cap on the proportional component alone so that $K_p e$ cannot demand an excessive tilt.

On each control step, the `update()` method is called with the current error $e_\alpha(t)$, timestamp, and (optionally) a pre-computed velocity estimate:

Listing 3: Structure of the PID update routine

```
class PIDController:
    def update(self, error, current_time=None,
               current_position=None, velocity_estimate=None):
        # Time step and initialisation
        if current_time is None:
            current_time = time.time()
        if self.last_time is None:
            self.last_time = current_time
            self.last_error = error
            self.last_position = current_position
            return 0.0
        dt = current_time - self.last_time
        if dt <= 0:
            return 0.0

        # Proportional term (capped)
        p_term = min(self.kp * error, self.max_kp_tilt)

        # Integral term with simple anti-windup
        if abs(self.integral * self.ki) < self.output_limit * 0.8:
            self.integral += error * dt
        i_term = self.ki * self.integral

        # Low-pass filtered derivative of the error
        raw_derivative = (error - self.last_error) / dt
        self.filtered_derivative = (
            self.derivative_filter_coeff * raw_derivative
            + (1.0 - self.derivative_filter_coeff) * self.filtered_derivative
        )
        d_term = self.kd * self.filtered_derivative

        # Velocity term: counter-steer against ball velocity
        ff_term = 0.0
        if self.use_velocity_feedforward and velocity_estimate is not None:
            ff_term = -self.pf * velocity_estimate

```

```
        # Final output with saturation
        output = p_term + i_term + d_term + ff_term
        output = np.clip(output, -self.output_limit, self.output_limit)

        # State update
        self.last_error = error
        self.last_time = current_time
        self.last_position = current_position
        return output
```

G.2 BallBalancingController and Main Loop

The high-level orchestration is handled by the `BallBalancingController` class. Its constructor wires together the sensor, kinematics, servo driver, and PID instances:

- Creates a `ForcenInterface` object using the configured serial port and sensor registers.
- Creates a `StewartPlatform` instance with the measured geometry and nominal plate height.
- Creates a `ServoController` to send actuator angles over serial.
- Instantiates two `PIDController` objects, `pid_x` and `pid_y`, with identical gains for the x and y axes.
- Allocates circular buffers to store the most recent position and timestamp samples for velocity estimation.
- Launches separate visualization processes and writes initial PID gains to the shared tuning file read by `viz_tuner.py`.

The `run()` method implements the real-time loop that is used in experiments:

Listing 4: Simplified structure of the control loop

```
def run(self, update_rate=0.05, verbose=True):
    next_update = time.time()
    while True:
        loop_start = time.time()

        # 1) Read sensor and estimate ball position (mm)
        data = self.forcen.read_data()
        ball_position = self.forcen.get_ball_position(data)
        if ball_position is None:
            continue
        x, y = ball_position

        # 2) Update position history and estimate velocity via linear fit
        self._update_ball_history(loop_start, x, y)
        vel_x, vel_y = self._estimate_velocity()

        # 3) Compute errors w.r.t. center setpoint
        error_x = self.setpoint_x - x
        error_y = self.setpoint_y - y

        # Deadband: ignore small errors near center
        error_mag = np.sqrt(error_x**2 + error_y**2)
        if error_mag < 5.0: # mm
            error_x = 0.0
            error_y = 0.0

        # 4) Per-axis PID with derivative filtering and velocity term
        roll_correction = self.pid_x.update(
```

```
            error_x, loop_start, x, vel_x
        )
        pitch_correction = self.pid_y.update(
            error_y, loop_start, y, vel_y
        )

        # 5) Map corrections to plate normal and servo angles
        roll_rad = np.deg2rad(roll_correction)
        pitch_rad = np.deg2rad(pitch_correction)
        normal = [
            np.sin(roll_rad),
            np.sin(pitch_rad),
            np.cos(pitch_rad) * np.cos(roll_rad)
        ]
        ik = self.platform.inverse_kinematics(
            normal, [0, 0, self.platform_height]
        )
        angles = [ik['theta_11'], ik['theta_21'], ik['theta_31']]
        self.controller.send_angles(angles)

        # 6) Periodically reload PID gains from viz_tuner file
        self._read_pid_gains()

        # 7) Sleep to enforce the desired loop rate
        next_update += update_rate
        sleep_time = next_update - time.time()
        if sleep_time > 0:
            time.sleep(sleep_time)
```

The helper methods `_update_ball_history()` and `_estimate_velocity()` perform the linear regression over a sliding window of recent (t, x, y) samples to obtain $\dot{x}$ and $\dot{y}$, which are then supplied to the feedforward term in Listing 3. Additional features in the full implementation (not shown here) include LED status indication to show when the platform is balancing, standby, calibrating and when the ball is balance, logging to a binary file, and streaming data to visualization.

Figure 47: A snippt of the terminal during startup