Département de Génie Électrique & Informatique

Tutored research report

# Development of a ball balancing system

May 19, 2023

**Students :**

| | | |
|---|---|---|
| ABREU DE ANDRADE | Heitor | abreu-de-and@insa-toulouse.fr |
| BELAFKI | Ayoub | belafki@etud.insa-toulouse.fr |
| COELHO ROBL | Ana Carolina | coelho-robl@insa-toulouse.fr |
| EYCHENNE | Vincent | veychenn@etud.insa-toulouse.fr |
| GARNICA AZA | Jose Eduardo | garnica-aza@insa-toulouse.fr |
| SUAREZ SEPULVEDA | Johan Sebastian | suarez-sepul@insa-toulouse.fr |

**Tutors :**

| | |
|---|---|
| CHANTHERY | Elodie |
| LE CORRE | Gwendoline |

**Keywords:** Stewart platform; PID control system; 3dof; Position regulation; Camera recognition

## Abstract :

This study addresses the regulation of ball position on a 3-degree-of-freedom (3-DOF) Stewart platform using a Proportional Integral Derivative (PID) control system. Previous research has primarily focused on either 2-DOF or 6-DOF platforms, leaving a significant knowledge gap concerning the application of control techniques to 3-DOF Stewart platforms. This research aims to bridge this gap by investigating the feasibility of PID control for regulating ball position on the platform.

To achieve this, an Arduino-based system incorporating three servos is employed for platform control, with the PID control system implemented in MATLAB Simulink. The ball's position is acquired through a webcam and computer vision, and the data are transmitted serially for processing.

While the results in terms of ball position regulation were unsatisfactory within the scope of this preliminary study due to time constraints and model play, the detection system demonstrated exceptional performance, and the Arduino-based servo control proved highly effective.

As a next step, further research will focus on enhancing the accuracy and robustness of the platform by implementing advanced control techniques such as Linear Quadratic Regulator (LQR), Fuzzy Logic, or Sliding Mode controllers. These methods are expected to address the limitations encountered in this study and improve the regulation of the ball's position on the 3-DOF Stewart platform.

In conclusion, this study presents a preliminary investigation into the regulation of ball position on a 3-DOF Stewart platform using PID control. Although the results were limited by time constraints and model play, the findings indicate the potential for achieving accurate ball position detection. Future research efforts will explore advanced control techniques to overcome the limitations encountered and enhance the platform's regulation capabilities.

# Contents

## 1 Introduction

Our project consists in developing a ball stabilization model using a 3 degree of freedom (3dof) Steward platform. In previous years, other student groups have developed the physical model. This year, we have two main objectives: to stabilize the ball on the model and to make our model usable for the automatic laboratory at the Institut National des Sciences Appliquées (INSA) of Toulouse. In this report, we focus on exploring the regulation of ball position on a 3dof Stewart platform using a Proportional Integral Derivative (PID) control system.

Previous research has primarily concentrated on either 2dof [1, 2] or 6dof [3, 4] Stewart platforms, with limited investigations conducted on the 3dof variant. Addressing this knowledge gap, our research aims to assess the feasibility of employing a PID controller to achieve accurate ball position regulation on a 3dof Stewart platform.

Firstly, we will present the hardware employed and explore the integration of the Arduino microcontroller in the system, explaining its role and functionality in facilitating real-time control and communication with the platform.
Secondly, we will discuss the development and implementation of the ball detection algorithm, highlighting its effectiveness and reliability in accurately determining the position of the ball.
Furthermore, the mathematical model of the 3dof Stewart platform will be presented, elucidating the relationships and equations utilized for control and manipulation.
The report will finally examine the theoretical foundations and principles underlying the PID control system, exploring its potential capabilities and limitations in regulating the ball position. Although we couldn't perform practical experiments in this regard, we will present a comprehensive theoretical framework for understanding the PID control system's functioning and its potential application to the 3dof Stewart platform.

While practical experimentation with the PID control system was limited, this report aims to provide a comprehensive understanding of the various components involved in the regulation of ball position on the 3dof Stewart platform. By focusing on the theoretical calculations, algorithm development, platform modeling, and the integration of the Arduino, this report lays the foundation for future practical implementation and testing.

# 2 Materials and Methods

## 2.1 Hardware

### 2.1.1 Arduino

The Arduino Uno served as a programmed interface between the computer-based control system and the mechanical components of our 3dof Stewart platform, which was at the center of our experimental setup. An Arduino Uno, a board built on the ATmega328P microprocessor, was employed in this study. It includes a 16 MHz quartz crystal, 6 analog inputs, 14 digital input/output pins (of which 6 can be used as PWM outputs), a USB port, a power jack, an In-Circuit Serial Programming (ICSP) header, and a reset button.[5]

The Arduino's ability to translate MATLAB Simulink signals into servo-actionable commands is what makes the experimental setup successful. The Uno was configured to execute these instructions on the 3dof Stewart platform in real-time using the Arduino Integrated Development Environment (IDE). This allowed the PID controller to manage the ball's position.

The maximum current output of the Arduino Uno is one of its drawbacks because it cannot power numerous servos. The Uno's digital or analog pins have a cap on the amount of current they can deliver at roughly 20 mA per pin, with a total output of 200 mA across all pins[5]. This is insufficient for the current-hungry servos utilized in this configuration.

### 2.1.2 External Power Supply

The servos were powered by an external power source in order to get around this restriction. This configuration prevented the Arduino, which would have most certainly led to system failure due to the Uno's power limitations, from supplying the servos with the necessary current.

To guarantee constant voltage levels and avoid any ground loop issues, the external power supply and the Arduino shared a common ground. This allowed the Arduino to control the servos without also having to supply power for them, ensuring that the Arduino Uno's capabilities were fully utilized within its operational parameters.

### 2.1.3 Servos

Servo motors, a class of electric motor devices sometimes referred to as "servos," are used in robotics and other sectors due to their accuracy and control skills. A motor, a control circuit, a potentiometer (for position feedback), and gears are the components of a servo. They may be controlled by providing them a particular electrical pulse signal, and they are made to rotate and retain the output shaft in a given position. They are perfect for applications needing precision positioning since they can hold their position even when outside forces try to shift them.

We used Tower Pro MG995R servo motors for this build. These are metal-geared, high-torque servos that provide greater stability and longevity than their plastic-geared equivalents. They are a great option for demanding applications like ours because of their voltage range of 4.8 - 7.2V and their stall torque of roughly 10 kg-cm (at 6V)[6].

RC servo pulses with a range of roughly 500 to 2500 microseconds and a refresh rate of about 50 Hz are accepted by the Tower Pro MG995R. The servo can rotate its output shaft to a precise angle within a range of 180 degrees in response to these input pulses.

The Arduino in our setup converted the control directives from the MATLAB Simulink model into the pulse-width modulation (PWM) signals that the servo motors could interpret. The Stewart platform's position was modified by the servos in response to these signals, allowing for fine control of the ball's placement on the platform.

The Tower Pro MG995R servos' sturdy design and good torque performance were crucial in preserving accurate and stable control of the 3dof Stewart platform, highlighting their efficiency in high-precision control systems.

### 2.1.4 Serial communication

Serial communication is a form of data transfer where information is transmitted sequentially, one bit at a time, over a single communication channel. It is a fundamental method employed in digital communication systems for the exchange of data between two devices. Serial communication offers several advantages, including simplicity, reliability, and compatibility with a wide range of devices. It is extensively utilized in various applications, such as computer peripherals, embedded systems, and long-distance communication between devices.
Serial communication was employed in our setup to ensure the reliable and real-time transfer of data between the computer and the Arduino. We used the Arduino's onboard hardware serial ports to communicate with the computer. Serial communication must be configured on the transmitter(MATLAB Simulink) and on the receiver(Arduino).

**MATLAB Simulink side.** Serial communication requires perfect synchronization between the transmitter and the receiver. The processing time in our algorithm being variable, the use of a Zero-Order Hold (ZOH)[1] is necessary to guarantee a perfect synchronization. The Simulink program concatenates the desired angle for each of the servos and transmits them at fixed intervals to the Arduino.

**Arduino Uno side.** The Arduino board is configured to read at fixed intervals its serial reception buffer. The reading frequency is the same as the transmission frequency configured in the ZOH. The received angles are decomposed and are finally transmitted to their respective servo.

## 2.2 Ball detection

### 2.2.1 Image acquisition

The camera used is a "Microsoft LifeCam HD-3000", presented in Figure 1. It is a generic webcam supporting a resolution of up to 1280x720 pixels[7]. Even though faster technologies exist such as touchscreens[8], we decided to keep the camera since it is a cheaper alternative and has frames per second (FPS) and latency properties that are enough for a simple ball detection system.

[1]a Zero-Order Hold (ZOH) approximates a continuous-time signal by maintaining a constant value during each sampling period.

Figure 1: Microsoft LifeCam HD-3000 Webcam [7]

#### 2.2.2 Description of our image recognition system

As our model will be used for practical work, we decided to use exclusively MATLAB Simulink for the whole project. Therefore we have to use it also for the ball detection.
The camera provides images of the ball on the platform to the MATLAB algorithm and the images are processed using the "Blob Analysis" function[9]. In order to keep a simple algorithm and to increase the processing speed, no AI was involved.
For this reason, we had to pick an orange ball whose color is very different from the white plate. This allowed the algorithm to detect the outline of the ball, and from this outline, to compute the x and y coordinates of the center of the ball.

#### 2.2.3 The Blob Analysis function

In MATLAB, the blob analysis function is used to identify and analyze connected regions or "blobs" in an image. Blobs typically represent objects or regions of interest that share similar characteristics, such as color, intensity, or texture.
The blob analysis function in MATLAB offers several operations to extract useful information from blobs, such as their centroid, area, perimeter, bounding box, and more.
The following parameters must be chosen to make the algorithm work properly :

1. Allowable color range: In order to recognize only the orange ball, an appropriate color interval must be used. This range must be wide enough to take into account the different brightness conditions. We have chosen an interval from hsv(5,40,50) to hsv(15,100,100), shown in Figure 2.

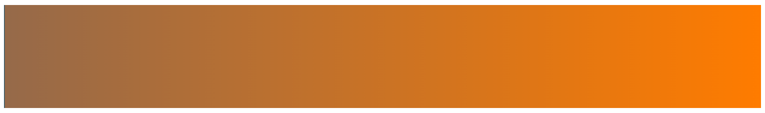

Figure 2: Allowable color range : from hsv(5,40,50) to hsv(15,100,100)

2. Image preprocessing: A x5 numeric zoom is applied in order to only analyze the platform
3. Bloc Detection: The allowable area range for the object is set [100, 10000] square pixels. The defined interval prevents the system from confusing the ball with very small orange objects (wire, digital noise)
4. Post-processing: Of all the orange objects that can be identified, only the largest is considered. The function then returns the center of this object.

### 2.2.4 User interface

In order to allow an easy debugging to the user, an interface allowing to visualize in live the images taken by the camera as well as the position of the ball has been added.
A first function named concatenate is in charge of transforming the position (x,y) and the surface of the ball into a shape
The "Draw circle" block allows to overlay the original image with the shape.
The whole set is then displayed to the user thanks to the "To video display" block.
The Simulink blocks used are shown in Figure 3.

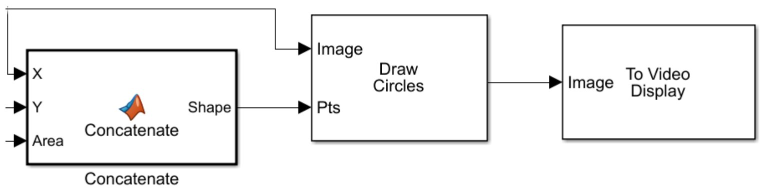

Figure 3: Simulink diagram of the interface

## 2.3 System modeling

### 2.3.1 Mathematics modeling

**Illustrative graph**

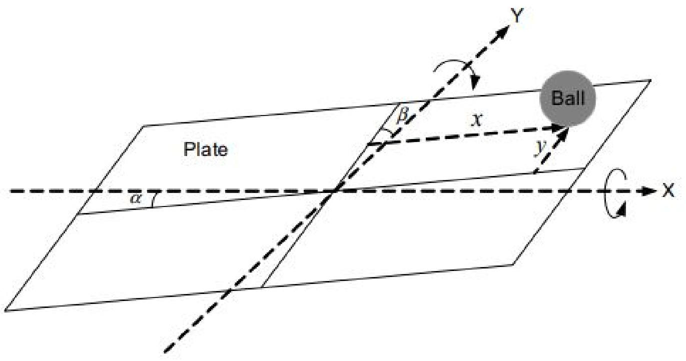

Figure 4: Free body diagram

**Mathematical modeling** In this case, we use Newton's second law to the free body diagram presented in Figure 4, which states that the sum of forces is equal to mass times acceleration. Applying it to angular acceleration, we obtain the following two differential equations that provide the modeling of the system

$$(m + \frac{I}{r^2})\ddot{x} - m(x\dot{\alpha}^2 + y\dot{\alpha}\dot{\beta}) + mg\sin(\alpha) = 0 \qquad \text{(eqt. 1)}$$

$$(m + \frac{I}{r^2})\ddot{y} - m(y\dot{\beta}^2 + x\dot{\alpha}\dot{\beta}) + mg\sin(\beta) = 0 \qquad \text{(eqt. 2)}$$

**Linearization** In general, the dynamics of the ball and plate system can be linearized under the following assumptions:

- The ball is considered as a point object with concentrated mass.
- The plate is considered rigid and its deformation is negligible.
- Contact forces between the ball and plate are modeled as normal and tangential forces.
- Angular accelerations of the ball and plate are small.
- The angle of rotation of the plate is small so that we can suppose $\sin\alpha \approx \alpha$ and $\sin\beta \approx \beta$
- The centrifugal force on the ball can be ignored as it is very small compared to gravity.

From these assumptions, the linearized model can be expressed as following:

$$\ddot{x} + \frac{5}{7}g\alpha = 0 \qquad \text{(eqt. 3)}$$

$$\ddot{y} + \frac{5}{7}g\beta = 0 \qquad \text{(eqt. 4)}$$

Equations (3) and (4) are decoupled so that we can consider as 2 different single-input and single-output systems. We can also design a controller by considering a single system and then apply it to both systems because of their similarity. The input and output for each system is$\alpha$, x and $\beta$, y.

From this, Python code[10] uses alpha and beta to generate the angle that each servo motor has to have in order to get the ball in the center of the plate.

### 2.3.2 Digital twin modeling

For the successful development of the project, it was crucial to create a digital twin of the physical system. This modeling enables precise comprehension of the mechanics of the movement, as well as the system's response to desired inputs. Moreover, it provides the advantage of manipulating the system without the need of direct contact with the physical model. By implementing a digital twin, a deeper understanding of the system's behavior is achieved, ensuring more accurate analysis and facilitating the optimization process.

**3D Assembly.** The first step to develop the digital twin was to make an assembly of the previously modeled parts [2], for this it was necessary to convert these armature parts into solid parts, then using the Autodesk Inventor software, we were able to modify joint conditions such as extreme conditions, insertion angles and motion geometry between the solid bodies.

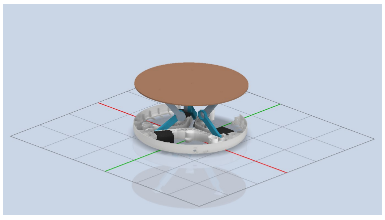

Figure 5: Assembly 3D made by Inventor Software with CAD pieces.

[2]CAD parts modeled in 3D by the last project team

This software allowed us to precisely manipulate the position and joint conditions between components and then export them as a block diagram to the MATLAB Simulink environment using the Simscape[11] library.

**Simulink model.** Due to the nature of the project its export and development in the MATLAB Simulink environment was imperative, thanks to the use of the Simscape export library it was possible to transform the CAD model into a block diagram that allowed its numerical and precise manipulation to later concatenate the mechanical subsystem obtained with the ball detection system and processing of the input angles.

In the Figure 6,it is evident that the input to the joints [0 to 2] corresponds to the angles to be taken by the servomotor shaft, in this case they are constant but it is there where the connection of this subsystem to the outputs of the ball detection subsystem will be made, The rotational joints [3 ... 5] correspond to the connection between the base arm and the arm connected to the plate, likewise there are 3 spherical joints that join the plate to the second arm allowing the movement of 3 degrees of freedom, shown in the Figure 7 an image of the execution of this subsystem.

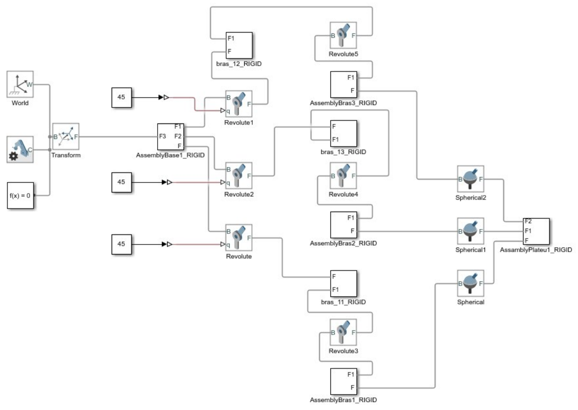

Figure 6: Block diagram obtained by Simscape

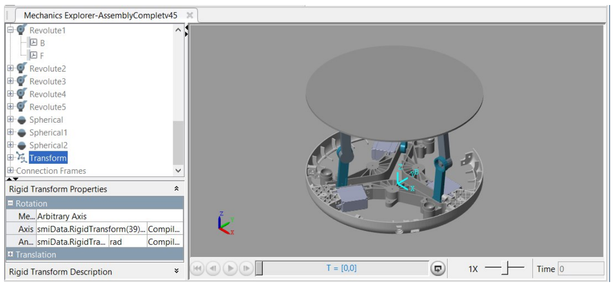
Mechanics Explorer-AssemblyCompletv45
Revolute1
B
F
Revolute2
Revolute3
Revolute4
Revolute5
Spherical
Spherical1
Spherical2
Transform
Connection Frames
Rigid Transform Properties
Rotation
Me... Arbitrary Axis
Axis smiData.RigidTransform(39)... Compil...
An... smiData.RigidTra... rad Compil...
Translation
Rigid Transform Description
T = [0,0]
1X
Time 0

Figure 7: 3D animation generated by Simulink run

## 2.4 Controller

**PID Controller.** The PID controller is known for its simplicity and effectiveness in regulating system behavior by continuously adjusting the control input based on error, integral, and derivative terms.The control of dynamic systems plays a crucial role in numerous scientific and industrial processes. Among various control algorithms, the PID controller has gained significant popularity due to its versatility and easy of implementation.

The PID controller comprises three main components: the proportional (P) term, the integral (I) term, and the derivative (D) term. The proportional (P) term in a PID controller contributes to the control action based on the current error between the desired setpoint and the actual system output. It directly multiplies the error by a gain factor, which determines the proportionality between the error and the control output.The integral (I) term addresses steady-state errors by considering the cumulative sum of past errors.The I term acts as a corrective measure to eliminate residual errors that may persist even when the proportional term is applied. The derivative (D) term anticipates the future behavior of the system by measuring the rate of change of the error. It calculates the derivative of the error and multiplies it by a gain factor, the derivative gain.

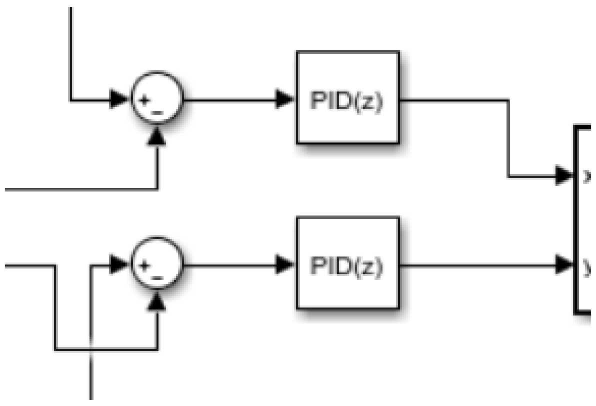

Figure 8: Discrete PID utilized in simulink

**Controller implementation.** The PID controller for the ball and plate system was implemented using MATLAB Simulink[12] and is presented in Figure 8. The control algorithm was designed to regulate the position of the ball by adjusting the plate's tilt angles. The input of the PID controller is the difference between the position of the ball and the position of the center of the plate, both in pixels, normalized, and its output is used to define the angles $\alpha$ and $\beta$ of the plate. We use a PID in discrete time[13] since the frequency of the camera and consequently of our system is 30 Hz. The Simulink block used is shown in Figure 8,. And finally, PID gains were estimated with simulink's sisotool[14] tool.

# 3 Results

## 3.1 Hardware

The results of our hardware setup demonstrated a successful interplay between the Arduino Uno, the Tower Pro MG995R servos, and the serial communication protocol. However, some operational challenges were observed that affected the overall performance.

### 3.1.1 Arduino Uno and External Power Supply

The Arduino Uno, coupled with an external power supply, effectively managed the high-power requirements of the servo motors. It successfully converted control commands from the MATLAB Simulink model into signals for the servos, despite the power limitations of its own microcontroller. This successful integration demonstrated the feasibility of using an Arduino as a central control hub even when the power demands of the system exceed its native capabilities.

### 3.1.2 Servo control

In terms of positional control, the Tower Pro MG995R servos performed commendably within the 3dof Stewart platform setup. They provided a significant level of precision and responsiveness to the Arduino's control signals, consistently rotating and maintaining their output shaft at the required positions.

However, we encountered an issue related to the speed of the servos' response. Despite receiving the correct control signals, the servos showed a delay of about a second before responding, which negatively impacted the real-time control of the platform. This delay was unexpected given the servos' known specifications and introduced additional complexities to the system's performance.

### 3.1.3 Serial Communication

The serial communication protocol facilitated reliable data transmission between the Arduino Uno and the computer running the MATLAB Simulink model. Despite potential latency risks associated with the protocol, the positional data and control commands were consistently transmitted with minimal delay.

However, in relation to the servo response delay, further investigation into the communication block suggested that the issue might lie in the communication between the Arduino and the servos. Despite receiving correct signal frequency at the communication block input, the delay issue persists, indicating that this could be a communication problem between the Arduino and the servos.

## 3.2 Ball detection

The ball detection algorithm has given good results under various brightness conditions, thanks to the auto-brightness adjustment in MATLAB Simulink. We verified the algorithm's effectiveness by testing it with a different webcam, which demonstrated the flexibility of our program.
During the utilization of the algorithm, no noticeable lag was detected, ensuring a real-time regulation of the ball's position.

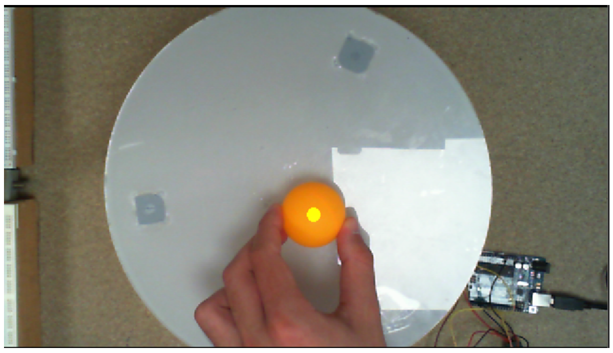

Figure 9: Calculated ball position, marked by a yellow circle on the user interface

However, we encountered a minor drawback where the wood support occasionally got misidentified as the ball. Nevertheless, using a x5 zoom on the camera has mitigated much of the issue. Finally, other solutions are evoked in 4.2.

### 3.3 Simscape model

After creating a virtual twin of the physical system, a key step was to determine the initial servo angles. It was observed that the angles of the *Revolute Joint* [15] (servos) were reversed compared to the physical system. Therefore, an important adjustment was made by introducing a negative unit gain to establish the correct angle relationship. Following this adjustment, the digital model exhibited perfect response.

As depicted in Figure 10, with all setpoint angles set to 45°, the plate appeared perfectly straight.

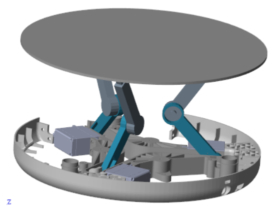

Figure 10: 3D animation of the digital twin model for setpoint angles all equal to 45°

As illustrated in Figure 11, for different setpoint angles (60°, 40°, and 0°), the plate exhibited a tilted position similar to that observed in the real system.

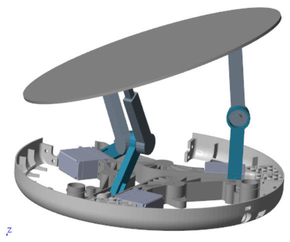

Figure 11: 3D animation of the digital twin model for 60°, 40°, and 0° setpoint angles

## 3.4 Controller

To design the controller, we utilized Simulink to implement the system model described in section 2.3.1 By leveraging the sisotool, we obtained insightful results pertaining to the PID constants and the system's behavior. Simulink served as a valuable platform for constructing and simulating the model derived from the theoretical framework outlined in 2.3.1. The model incorporated the relevant system dynamics and allowed us to explore various control strategies. By defining the appropriate transfer functions and input-output relationships, we were able to represent the system's behavior accurately within the Simulink environment.

Through the sisotool's graphical interface, we could visualize the system's response to different PID gains and observe the resulting effects on stability, settling time, overshoot, and other relevant performance metrics. By adjusting the PID constants and observing the system's response in real-time, we were able to iteratively fine-tune the controller to achieve the desired behavior.

The sisotool allowed us to gain a deep understanding of the interplay between the PID constants and the system's dynamic response. We carefully analyzed the trade-offs between stability and performance, seeking an optimal balance that minimized overshoot, ensured fast response times, and maintained robustness in the face of disturbances or uncertainties.

Overall, the combined use of Simulink and the sisotool provided us with a systematic approach to design and optimize the PID controller. The simulation environment enabled us to accurately represent the system's behavior, while the sisotool empowered us to fine-tune the PID constants and achieve a control strategy that met the desired performance criteria.

The simulation results yielded insightful findings. However, due to an issue encountered with the physical prototype, its physical implementation was not feasible. Additionally, it is crucial for the controlled system's response to exhibit a restrained nature, avoiding excessive aggressiveness. This consideration is essential as an overly aggressive response could lead to extraordinarily high velocities and accelerations, rendering the system uncontrollable.The system response is as follows:

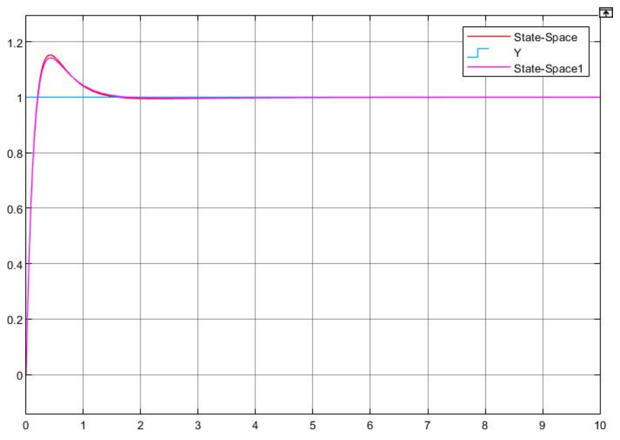

Figure 12: Modeled system response

Considering the requirement for a restrained system response, it is essential to strike a balance between stability and performance. A highly aggressive response may result in rapid and drastic changes in velocity and acceleration, making it difficult to maintain control over the system. By aiming for a more moderate and controlled response, we can ensure that the system remains within manageable limits as we can see in Figure 12, allowing for effective control and ensuring the safety and stability of the overall operation. The optimal gains obtained are as follows: $k_p = 3.467$, $k_i = 1.595$, $k_d = 1.414$.[3]

[3]The gains $k_p$, $k_i$, and $k_d$ correspond respectively to the parameters $(P)$, $(I)$, and $(D)$ mentioned in Section 2.4 .

# 4 Discussion

## 4.1 Hardware operation

The results obtained from our experimental hardware setup highlight both the capabilities and limitations of using an Arduino Uno, Tower Pro MG995R servos, and a serial communication protocol to control a 3dof Stewart platform.

**External power supply.** Despite the inherent power limitations of the Arduino Uno, our setup demonstrated that it could serve as a highly functional control hub when coupled with an external power supply. This finding underscores the Arduino's flexibility and adaptability in complex control systems that demand more power than it can natively provide.

**Servo.** Our servos, while demonstrating their precision and accuracy in positional control, encountered issues related to response speed. A noticeable delay of approximately a second was observed in their reaction time, which had an adverse impact on the real-time control of the Stewart platform. Given the Tower Pro MG995R servos' known specifications and previous performances, this delay was unexpected and, thus, suggests an area requiring further investigation.

**Serial communication.** Serial communication proved to be a reliable medium for data transmission between the computer running the MATLAB Simulink model and the Arduino Uno. However, we suspect the communication between the Arduino and the servos might be implicated in the observed servo response delay. While the input signal frequency at the communication block was confirmed as correct, the persistent response delay indicates a potential issue in the communication pathway to the servos.

Taken together, these results suggest that while our current hardware configuration is a promising starting point for controlling a 3dof Stewart platform, improvements are necessary for better real-time control and responsiveness. Specifically, resolving the servo response delay is crucial for enhancing the system's overall performance. This challenge provides a direction for future work, where more advanced control strategies and possibly different hardware configurations should be explored to increase the platform's accuracy and robustness. In doing so, the findings from this research will serve as a foundation for more sophisticated Stewart platform control systems.

## 4.2 Ball detection algorithm

As seen in 3.2, the ball detection algorithm has given good results under various brightness conditions. The user interface implemented for the project is really helpful, providing real-time display of the ball's position. Throughout its utilization, no noticeable lag was detected, ensuring a real-time regulation of the ball's position.
As said in 3.2, we encountered a minor drawback where the wood support occasionally got misidentified as the ball. This issue can be resolved by repainting the wood surface in white or by using a green ball, which would provide a distinct color contrast for accurate detection and prevent false identification.
Overall, the ball detection algorithm proved to be reliable and efficient, exhibiting strong performance in different conditions. With the suggested remedies for the detection issue, the algorithm can further enhance its accuracy and usability.

## 4.3 Digital twin model

Due to time constraints, only the plate was modeled in the project. As observed in 3.3, the plate demonstrates a perfect response to the input angle. However, in order to make the model more comprehensive, incorporating the ball into the Simulink/Simscape model would be a valuable addition. One possible approach could be to consider the ball as a separate rigid body with its own dynamics and interaction with the plate. This would involve defining the ball's mass, size, and properties such as friction and elasticity. Additionally, the forces and torques exerted on the ball by the plate could be modeled based on their contact and interaction. By integrating the ball into the existing model, a more accurate representation of the entire system could be achieved, allowing for a more comprehensive analysis and control design.

## 5 Conclusion

In conclusion, our project aimed to develop a 3DOF Stewart platform with ball position regulation. Throughout the course of our work, we made several key findings.

Firstly, the ball detection algorithm using a camera proved to be highly effective. It successfully detected and tracked the ball's position, providing reliable data for our control system.
Secondly, the Simscape digital twin model of the system worked well, accurately representing the dynamics of the platform. However, it is important to note that the model does not include the ball, which limits its comprehensive representation of the entire system.
Lastly, we encountered challenges in setting up the PID controller for the real model. Due to various difficulties, we were unable to fully implement and optimize the controller in practice. This serves as an area for future improvement and further development.

Overall, while we achieved success in certain aspects, such as the ball detection algorithm and the performance of the Simscape digital twin model, our project faced limitations in the practical implementation of the PID controller. Moving forward, addressing these challenges and refining the control system would be crucial to fully realize the potential of the 3DOF Stewart platform with ball position regulation using a PID controller.

## References

[1] J. Kumar, N. Showme, M. Aravind, and R. Akshay, “Design and control of ball on plate system,” vol. 9, pp. 765–778, Jan. 2016.

[2] A. Adiprasetya and A. Wibowo, “Implementation of PID controller and pre-filter to control non-linear ball and plate system,” Sep. 2016, pp. 174–178.

[3] S.-H. Lee, J.-B. Song, W.-C. Choi, and D. Hong, “Position control of a Stewart platform using inverse dynamics control with approximate dynamics,” *Mechatronics*, vol. 13, no. 6, pp. 605–619, Jul. 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957415802000338

[4] A. Koszewnik, K. Troc, and M. Słowik, “PID controllers design applied to positioning of ball on the Stewart platform,” *Acta Mechanica et Automatica*, vol. 8, no. 4, pp. 214–218, 2014. [Online]. Available: https://bibliotekanauki.pl/articles/387652

[5] Atmel, “ATmega328P 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash.” [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[6] Tower Pro, “MG995 High Speed Metal Gear Dual Ball Bearing Servo.” [Online]. Available: https://www.electronicoscaldas.com/datasheet/MG995_Tower-Pro.pdf

[7] “Microsoft Webcam: LifeCam HD-3000.” [Online]. Available: https://www.microsoft.com/en/accessories/products/webcams/lifecam-hd-3000

[8] K. B. Adak, A. N. Singh, and A. B. Kamble, “Touchscreen using web camera,” *Compusoft*, vol. 4, no. 5, p. 1717, 2015.

[9] “Statistics for labeled regions - Simulink - MathWorks France.” [Online]. Available: https://fr.mathworks.com/help/vision/ref/blobanalysis.html

[10] J. Link, “Système de stabilisation PID,” Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, La Tour-de-Peilz, Feb. 2020.

[11] “Simscape.” [Online]. Available: https://fr.mathworks.com/products/simscape.html

[12] “Simulink : simulation et approche Model-Based Design.” [Online]. Available: https://fr.mathworks.com/products/simulink.html

[13] “Continuous-time or discrete-time PID controller - Simulink - MathWorks France.” [Online]. Available: https://fr.mathworks.com/help/simulink/slref/pidcontroller.html

[14] “Design single-input, single-output (SISO) controllers - MATLAB - MathWorks France.” [Online]. Available: https://fr.mathworks.com/help/control/ref/controlsystemdesigner-app.html

[15] “Joint with one revolute primitive - MATLAB - MathWorks France.” [Online]. Available: https://fr.mathworks.com/help/sm/ref/revolutejoint.html