OsloMet – storbyuniversitetet
Fakultet for teknologi, kunst og design
Institutt for maskin, elektronikk og kjemi
Postboks 4 St. Olavs plass, 0130 Oslo
Telefon: 67 23 50 00

PROSJEKT NR. 19

TILGJENGELIGHET:
PUBLIC

BACHELOROPPGAVE

BACHELOROPPGAVENS TITTEL	DATO
Five Link	23.5.2025
	ANTALL SIDER / BILAG 123
FORFATTERE Nökkvi Ágústsson, Jesper Moe, Jacob Solberg Ellingsen og Araz Ezzatkhah Bastani	INTERN VEILEDER Paraskevi Zouganeli

UTFØRT I SAMMARBEID MED	EKSTERN VEILEDER
Tronrud Engineering	Ming Kit Wong

SAMMENDRAG
This bachelor project is about improving an existing robotic arm, previously known as DuoArm. The report explains the methods used to develop and improve features on the robotic arm, with the main focus areas being to design a functioning gripper, making sure that the arms can move in 3 different axes and to implement intuitive controls for all movements of the robot with a wireless controller.

3 STIKKORD
Robotic arm
Gripper
Inverse kinematic

MATS3900 group 19 group members contribution

	Written contribution	Technical contribution
Nökkvi Ágústsson	12, 13, 15-17, 44, 45, 66-76, 99, 109, 107, 109, 110	Torque and RPM calculations, 2d machine drawings, 3D modeling, 3d printing
Jesper Moe	10-11, 32-66, 110	3D modelling, 3D assembly modelling, 2D assembly technical drawings, 3D printing
Jacob Solberg Ellingsen	13-14, 90-98, 105, 106, 109, 110	Software Design and Development, Laser-cutting, Hardware design
Araz Ezzatkhah Bastani	5-8, 11, 12, 17-31, 77-89, 101-105, 107-109	Hardware Design Software development Soldering

Acknowledgements

We would like to provide a very special thank you to Ming Kit Wong who has helped us tremendously with all the knowledge and insight he has given us. This project would not have been as awesome without him. We would also like to thank our internal supervisor Paraskevi Zouganeli for all the guidance we received, while we send our deepest gratitude to Steffen Hurum and his students at Hønefoss videregående skole for helping with production.

Abstract

This bachelor project focuses on the continued development of a Duopod robotic arm in collaboration with Tronrud Engineering (TE). The primary objective is to create an exhibition-ready robot arm to meet TE demands. Key improvements include the partial redesign of the gear assembly (Festehub), the replacement of weaker motors with better ones from Dynamixel, design of three fingered adaptive gripper, fixing all the electronics and making a fully working stabilizer. The system is wirelessly controlled through a PS4 controller connected to a Raspberry Pi that can control all three movement axes while also being able to operate a gripper. The robotic arms generally functioned well, but occasionally a gear broke and halted the arm's movement.

Contents

Nomenclature

Property	Symbol	Units
Force	F	N
Torque	τ	Nm
Mass	m	Kg
Distance	l	m
Gravity	g	m/s^2
Coefficient of friction	f	Not Applicable
Mean diameter	d_m	mm
Thread angle	γ	rad
Pitch	p	mm
Angle seen from active joint, end-effector and passive joint	α	rad
Angle seen from active joint, between origo and end-effector	β	rad
Active joint (servo) angle	θ	rad
End-effector x-coordinate	x	mm
End-effector y-coordinate	y	mm
Temperature	T	C°
Volt	V	J/C
Ampere	A	C/s
Watt	W	J/s

List of Tables

1. Introduction

In recent times, robotics has become a fundamental part of a wide range of industries and is rapidly advancing in terms of speed and accuracy. One specific area of development is in the field of parallel kinematic robotics, known as DuoPods. These systems offer exceptional speed and precision compared to conventional automated robots.

This bachelor project continues the development of a DuoPod inspired robotic system initiated by a previous bachelor group for Tronrud Engineering. The aim of this project is to analyze and improve the existing system based on updated requirements. These updates include implementing inverse kinematics, wireless control, and a gripper, while reusing as much of the previous robot as possible.

The following chapters of this report delve deeper into the background of the project and present the analysis, development, improvements, and the final result.

2. Background

2.1 State of the art

The pursuit of automation in industry is constantly driven by the need for greater precision, efficiency, and safety. Robotics is an essential technology that has evolved significantly from basic mechanical systems into intelligent machines capable of handling both complex and repetitive tasks. Today, robots are widely used across nearly every aspect of life, from everyday household applications to advanced space missions. They are transforming industries where speed, accuracy, and human safety are critical.

A typical robot in the industry is primarily focused on applications where the task is repetitive and stationary. These robots are commonly found in the automobile industry, where they are fixed in place along an assembly line and dedicated to performing a single task such as the KUKA KR Agilus (KUKA, n.d). However, the industry is slowly shifting towards robotic systems where speed, agility, and responsiveness are prioritized. Robots like the KUKA KR Delta operate using parallel kinematics (Finder, n.d), which typically outperform standard serial robots like the Agilus in terms of speed and agility.

A serial robot such as the Agilus is built up of several joints, similar to a human arm, where all joints contribute to moving a shared end effector. Due to the series of arm segments, the longer the arm becomes, the more mass the serial robot must carry, which in turn reduces its flexibility and speed. In contrast, the DuoPod operates using parallel kinematics, where multiple sliding joints act directly on a common end effector. This design enables high speed and agility (Finder, n.d). Additionally, alongside the growing popularity of DuoPod systems, technologies such as actuators used in parallel kinematic robots are continuously improving. Combined with the distributed mass across joints, the DuoPod stands out from traditional serial robots by having minimal error-loading characteristics (PANDILOV & DUKOVSKI, 2012).

The growing popularity of DuoPods since the early 1990s marks a shift from serial to parallel kinematics, signaling a change in the next generation of robotic systems. The industry is continuously searching for faster robots with higher accuracy, and systems like the DuoPod, offering reduced inertia, increased speed, and greater agility, are gradually becoming specialists in meeting those demands. The foundation laid by these robotic systems has the potential to unlock entirely new systems and applications, evolving the industry toward greater efficiency, reduced costs, and higher efficiency.

2.2 Project Assignment

This project is based on the further development of a DuoPod system named DuoArm created by a previous bachelor group in collaboration with Tronrud Engineering, a company based in Hønefoss, Norway, known for designing and developing its own machinery, primarily in the field of automation. The ultimate goal is for the company to showcase the final product at seminars, demonstrating their ongoing innovations and potentially attracting commercial interest and future customers. The existing model was handed over to the bachelor group of spring 2025 for further analysis, testing, development, and improvement.

The assignment given was to analyze and improve the robot's functionalities based on a strict set of requirements provided by the supervisor from Tronrud. These updates clearly focused on enhancing the mechanical and electrical systems, particularly the movement of both arm joints and this time, also included the integration of an end effector. In addition, the supervisor emphasized the importance of reusing as many existing components as possible while implementing new possible solutions.

For this there are some demands Tronrud have given that needs to be met:

1. The robot should be travel- and exhibition friendly.
2. The robot should be powered with 230 voltages from the power outlet.
3. The robot should not weigh more than 12 kg.
4. The robot should be able to move three axes, both X, Y and Z.
5. The robot should be controlled by a wireless controller.
6. The robot should have a gripper.
7. The user should be able to execute a simple pick and place task with the hand controller.

The project is by nature interdisciplinary, combining the fields of mechanics, hardware, and software. The collaboration between these areas was a crucial part of the project in order to achieve a functional and user-friendly robot that met the demands and expectations of Tronrud Engineering.

2.3 Limitations

While the assignment focused on further improving the project, important considerations shaping the task such as time, resources, and the system's foundation made it clear there will be certain limitations to work by.

- The direction of the project was limited by continues work on an existing DuoPod design and other robotic architecture were not thought of.
- Mechanical design freedom was limited due to several fixed dimensions and component placements from the previous group's hardware.

- Reusing as many components as possible restricted opportunities for other potential hardware solutions.

- Testing and evaluation were limited to unprofessional environments, meaning during testing, the system was never exposed in real world industrial settings.

2.4 Report Structure

The report is structured into several chapters covering the theory, methods, development, and results of testing. The structure within each chapter explores the mechanical aspects, followed by hardware, and finally the software components.

Chapter 3 presents the theoretical foundation, explaining the essential concepts related to each field in this interdisciplinary project.

Chapter 4 describes the methods used in each field and how each part of the system was analyzed and planned before entering the development phase. It also includes the tools and equipment later used during implementation.

Chapter 5 focuses on the design and development of the mechanical components, while chapter 6 focuses on the design and development of the hardware and software components of this project.

Chapter 7 focuses on the results after the testing performed during the end period of the project. The tests are primarily isolated, meaning most were carried out on individual components of the final product rather than the system as a whole.

Chapter 8 discusses the results that were presented in chapter 7. The chapter also includes a section on possible future work to further improve the product after this project is completed.

Chapter nine is the final part of the report and presents the overall conclusion of the project, including reflections and recommendations for future work.

2.5 System Overview

The final robotic system is built from several mechanical and electronic components, implemented together to function as a compact and interactive DuoPod-inspired setup. The robot itself, including its arm joints, is mounted and suspended from a sliding rail on a metallic frame, which enables movement across all three axes x, y, and z. The rail, responsible for movement along the Y-axis, is powered by a servo motor, and both arm joints are also controlled by two servo motors, enabling movement along the X- and Z-axes. At the end of the arms is a gripper mounted and designed to pick and drop objects with flexible fingers.

The system is wirelessly controlled using a controller connected to a single-board computer (Raspberry Pi), which manages all communication between the software and hardware. Attached to the front and top of the structure is a 7-inch display, serving as a human interface for interacting with the system.

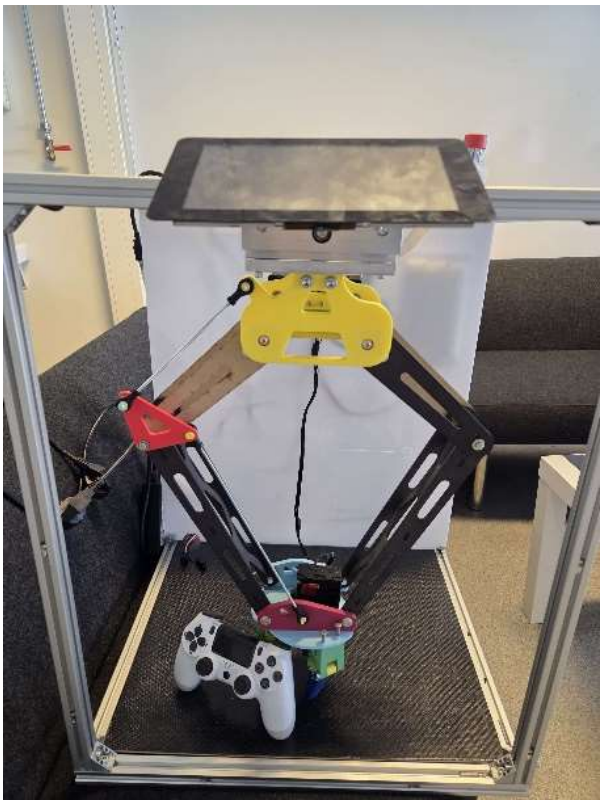


Figure 1 System overview

3. Theory

3.1 Soft robotics

Adaptive grippers

Adaptive grippers (**Figure 2**) are designed to more effectively grasp different types of objects by gradually conforming to their shapes (Sun et al., 2022). This versatility makes them suitable for a wide range of applications where the shape and size of the object varies. Unlike vacuum grippers, which are typically designed for very specific applications and are best suited for flat or uniform surfaces (Kumar, 2017), adaptive grippers can handle various shapes and, to some extent, different weights, all with improved grip stability (Sun et al., 2022). As this type of gripper is considered a form of soft gripper in the fields of soft robotics, it suffers from similar challenges based on design and application, such as robustness and speed (Shintake et al., 2018).

Figure 2 Two-fingered adaptive gripper from Festo.

Fin Ray Effect

Fish tail fins have a unique structural feature that improves the fish's ability to swim, and this is the way the tail bends when pressure is applied. For example, when pressure is applied to a piece of paper it bends away from the point of force. Fish fins, however, bend towards (**Figure 3**) exerted pressure rather than away (Crooks et al., 2016). This is what allows the fish to push water effectively when swinging their tails from side to side, an action known as the Fin Ray Effect (FRE).

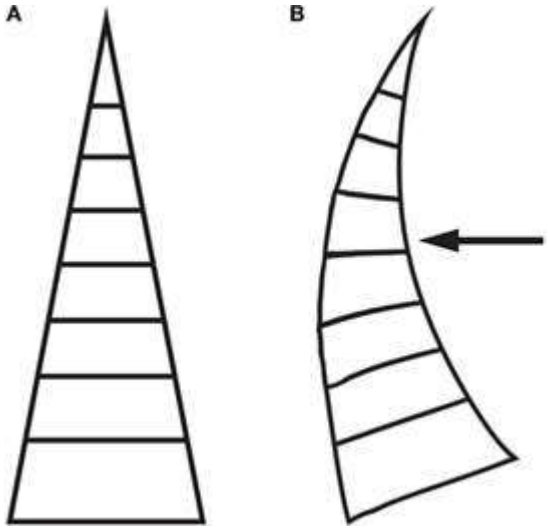


Figure 3 Two fingers where finger A is not applied pressure and finger B is applied pressure and bends towards the pressure point.(Crooks et al., 2016)

In robotics, this principle has been used to develop gripper fingers that automatically adapt to the object's shape by bending inwards upon contact. This makes Fin Ray-inspired fingers ideal for picking up any type of shaped object, especially delicate or fragile items (Elgeneidy et al., 2020). Today, the Fin Ray Effect is especially popular in soft robotics. While not used in all soft robotics applications, it has found its place in areas such as automotive assembly, battery production and electronics handling – industries that require safe, adaptive gripping solutions (Festo). In this project, the fin ray effect was utilized in the design of a three fingered adaptive gripper to provide a secure, shape-forming grip for the given application.

3.2 Servo Motor and closed loop system

The act of controlling a couple of actuators based on where an end effector is positioned with a wireless controller is heavily dependent on accuracy and control. This is especially important when using inverse kinematics, where programmed motors must calculate angles to rotate based on controller input. This is achievable with a **closed-loop system**, where built-in components inside the motor such as an encoder continuously update the angular position of the shaft (KOLLMORGEN, 2020). There are common electrical motors such as DC and stepper motors in the industry, both with certain advantages. However, the DC motor is typically used for systems with continuous movement, and the stepper motor for applications where positioning control is important where both motors typically does not have closed loop abilities to produce real time feedback of necessary parameters (Rosmo, 2023).

Servo motors, on the other hand, are motors designed to rotate to a certain angular position based on electrical input signals (Rosmo, 2023). This is typically accomplished with a built-in encoder that tracks the degrees of rotation. In the context of an end effector, where a controller dictates the position, the motors receive electrical inputs and react by sending feedback about the current position to the MCU, which further adjusts the desired angles.

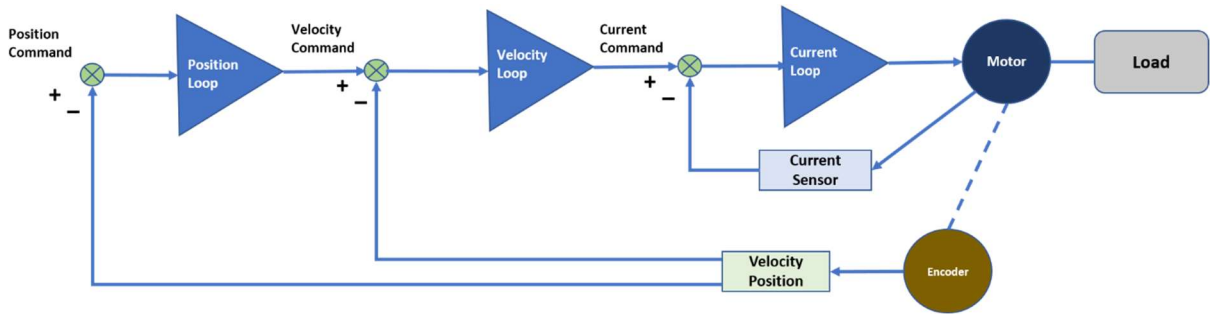


Figure 4 Representation of a closed loop with typically several other parameters, including feedback position from source (KOLLMORGEN, 2020)

3.3 Torque

"Torque is the quantitative measure of the tendency of a force to cause or change and objects rotational motion."(Young & Freedman, 2020) Torque τ is measured around a specific point by taking the length l from a point on a rigid object, where at the end of the length, there is a force F. The torque is directly proportional to F and l, making the torque formula:

$$\tau = F \cdot l \qquad [1]$$

To calculate the torque used by the motors to be able to move the arms of the robot, we have to find F in equation [1] by first simplifying looking at only one of the sides of the arms, since both sides are approximately mirrored versions of each other. Therefor a free body diagram can be made to help with the visualization of the calculations:

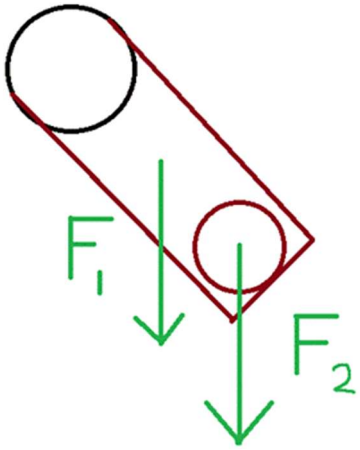


Figure 5 Free body diagram of one side of the arms, where the top circle is the rotary point and bottom circle is a freely rotating joint.

The force that the motor has to move is the mass m of the rigid part of the arm plus the mass from the freely rotating join times gravity g as shown in equation [2].

$$F = m \cdot g \qquad [2]$$

The torque equation can therefore be expanded as:

$$\tau = m_1 \cdot g \cdot l_1 + m_2 \cdot g \cdot l_2 \qquad [3]$$

Where $l_1$is the distance from the center mass of the rigid part to the motor and m_1 is the part rigid part of the arm, while l_2 is the distance from the rotating joint to the motor and m_2 is the mass that the rotating joint is holding.

The formula for the torque required to move the railing of the robot comes from the formula for the required torque to raise or lower a load using a power screw. Equation [2] below is the formula for torque when raising the load, while formula 3 is the formula for lowering the load. τ_r is the torque required to not only raise the load, but also overcome thread friction, F is the is the sum of the axial forces, while d_m is the mean diameter of the screw, p is the pitch of the screw and f is the coefficient of friction. (Budynas & Nisbett, 2020)

$$\tau_r = \frac{F \cdot d_m}{2}\left(\frac{p + \pi \cdot f \cdot d_m}{\pi \cdot d_m - f \cdot p}\right) \qquad [4]$$

While the formula for lowering the load is shown in equation 5 below.

$$\tau_l = \frac{F \cdot d_m}{2}\left(\frac{\pi \cdot f \cdot d_m - p}{\pi \cdot d_m + f \cdot p}\right) \qquad [5]$$

Note that the above formulas [3] and [4] are for square threads where the threads are parallel to the axis of the screw. When using other types of screws, the thread angle γ must be considered. The formula below shows the torque required to raise loads with thread types other than square threads.

$$\tau_r = \frac{F \cdot d_m}{2}\left(\frac{p + \pi \cdot f \cdot d_m \cdot \sec(\gamma)}{\pi \cdot d_m - f \cdot p \cdot \sec(\gamma)}\right) \qquad [6]$$

3.4 Inverse Kinematics

Given the scope of this project and its application, it's vital to have controls that do not require training; For the user to pick up the controller and be able to control the robot with ease. For that purpose, having the user control the point of the end effector would be intuitive. Inverse Kinematics will make this possible. This means we calculate from the desired point E(x,y) and work our way to what angle the active joints (θ_{a1} and θ_{a2}) should be. (Le et al., 2017)

The angles (θ_{a1} and θ_{a2}) are derived by splitting up the parallel arm into two triangles and calculating their angles α (for the outer triangle with l_1 as base) and β (for the inner triangle with l_0 as base)(Le et al., 2017)

Geometric Setup

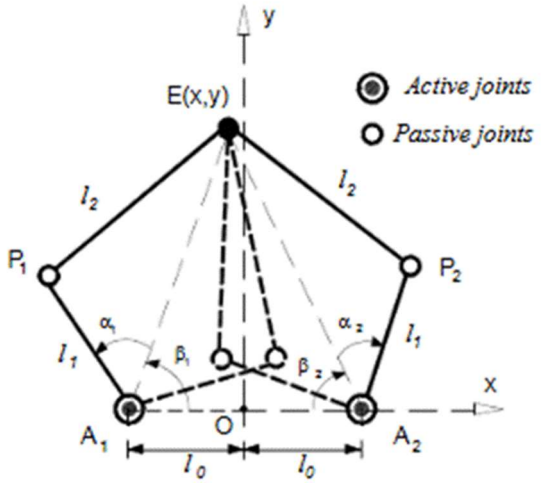


Figure 6 visual representation of geometric setup (Le et al., 2017)

The five-bar parallel manipulator consists of two actuated links (l_1) and two passive links (l_2), which connect the base joints and to the end-effector. The calculations rely on breaking down the problem into intermediate angles:

- α_1 and α_2: Angles between their respective **link** (**$P_{1,2}$**) through their **base (**$A_{1,2}$**) to** end-effector (**E**).
- β_1 and β_2: Angles between origo (**O**) through their **base (**$A_{1,2}$**) to** end-effector (**E**).

Using this setup, the desired joint angles are computed as:

$$\theta_{a_1} = \beta_1 \pm \alpha_1 \qquad [7]$$

$$\theta_{a_2} = \pi - \beta_2 \pm \alpha_2 \qquad [8]$$

For us we need to use (+) α_1 and (-) α_2 (fix when making the formula)

Calculation of α_1, α_2, β_1 and β_2

he values of these angles are derived using the law of cosines and basic trigonometric functions:

[9]

$$\alpha_2 = \cos^{-1}\left(\frac{l_1^2+\left((l_0-x)^2+y^2\right)-l_2^2}{2l_1\sqrt{(l_0-x)^2+y^2}}\right) \qquad [10]$$

$$\beta_1 = \tan^{-1}\left(\frac{y}{l_0+x}\right) \qquad 11]$$

$$\beta_2 = \tan^{-1}\left(\frac{y}{l_0-x}\right) \qquad [12]$$

4. Methods and Materials

4.1 Mechanical software usage

Autodesk Inventor

Autodesk Inventor (Autodesk, 2025) has been one of the two main software products used by the mechanical department. This software makes it possible to make almost any 3d model with exact dimensions, while also being able to assemble a 3d model of all the 3d parts together as shown in figure [7]. This makes it possible to virtually see how the system is going to look like when the final product is finished, making it possible to spot and fix lots of mistakes earlier in the design process before the actual product is even produced for the first time. Autodesk Inventor is also used to produce 2d technical drawings as shown in figure [8] which are later used by machinists to produce parts in material that we as students cannot make ourselves, such as parts made out of aluminum.

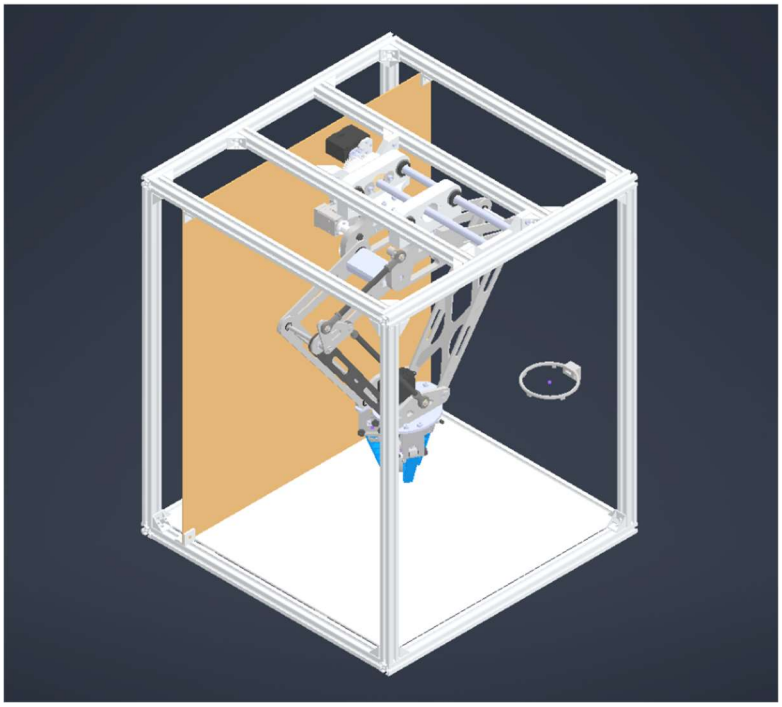


Figure 7 3D assembly of the team's final product

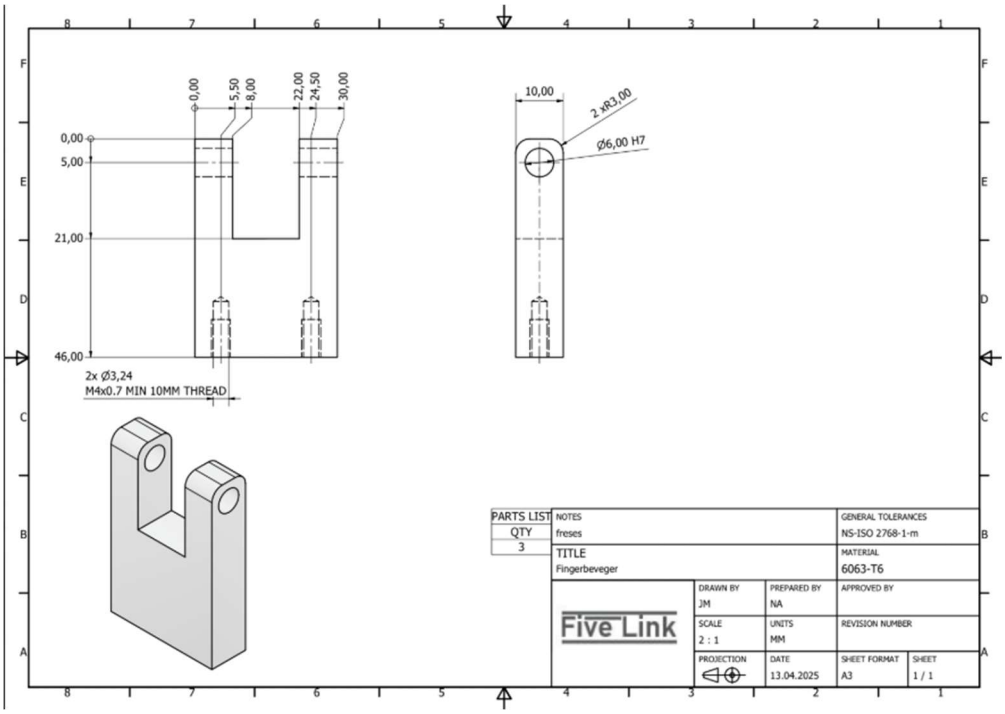


Figure 8 one of the teams 2D machine drawings, which machinists can use to produce the part in aluminum

3d printing

Another way to help identify and fix problems is by 3d printing parts and then assembling those parts in real life to test out if the design actually functions as it is supposed to. This is less expensive than making parts out of materials like aluminum, both because of manufacturing costs and the material being priced differently. 3d printing in this project was done with 2 types of printers. Ultimaker 2+ and Bambu Lab A1 printers. At the beginning of the project, the team mostly used Ultimaker 2+ printers, but after a fair bit of failed prints the team decided to test out Bambu printers which worked much better. The software used to prepare prints with the Bambu printer is called Bambu Studio. (Lab, 2025) With Bambu Studio it is possible to change important settings when preparing the 3d printing. For example, the layer height or quality of the print can be changed to make prints for accurate, or the infill percentage and pattern of 3d prints can be changed to increase the strength of the part you are printing. This is especially useful when making parts, because larger parts are often stronger, and therefore require less infill than smaller parts, while larger parts also take significantly longer to print if a larger infill percentage is chosen. Usually, the parts are to be printed with default settings, meaning 0.2mm layer height and 15% infill, then an evaluation is done to see if a higher quality or strength is needed for the part. An example of a print is shown in the figure [9] below.

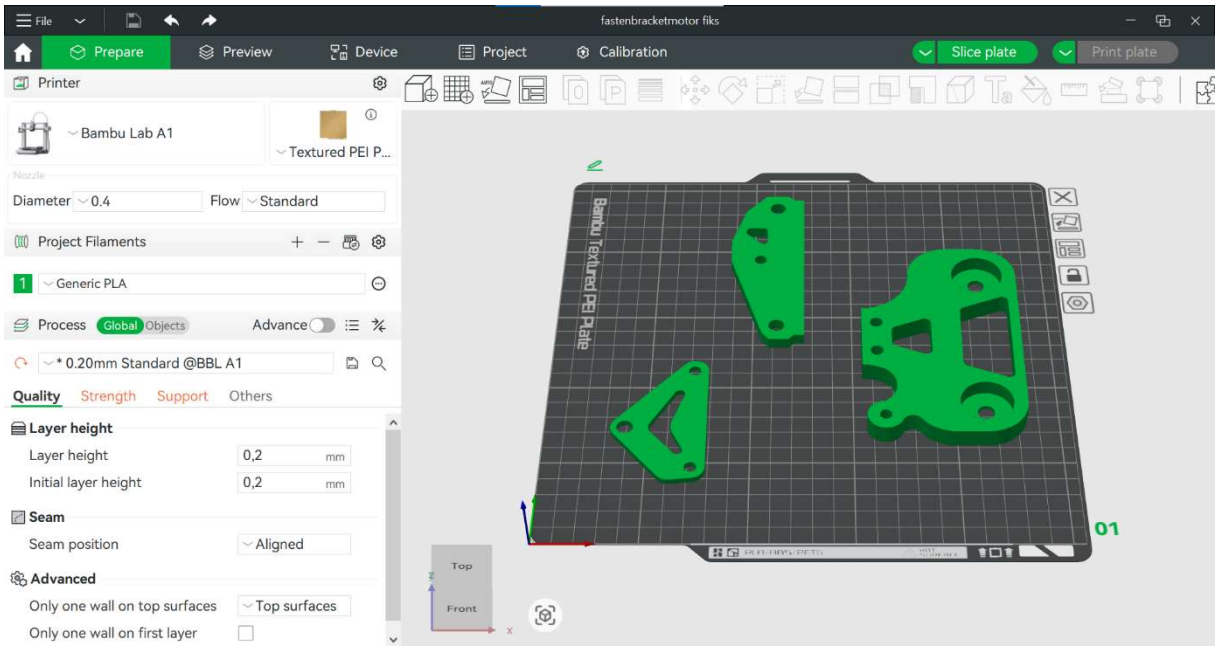


Figure 9 Example of 3D printing program Bambu studio

4.2 Hardware

4.2.1 Overview of existing hardware design

With knowledge gathered from the previous group's thesis (Magnor et al., 2024), the project started by observing both the previous group's thesis and the physical robot received from Tronrud. This inspection was crucial, as it gave an understanding of the current state of the hardware, what components were used, how they were interconnected, and which aspects were functional or could potentially be improved. Through inspection, identification of potential areas for improvement, while importantly, aligned with a new set of project requirements, specific to this year's continuation of the same project.

The hardware setup developed by the previous group included:

- Power supply (RD-85)
- Raspberry Pi 4 Model B
- 2x Arduino Mega
- Custom-built wired controller

These components were all integrated into a single controller unit. The controller itself consisted of self-made buttons and a joystick on a PCB board, which was connected to the Arduino Mega via male-to-female jumper wires. Data was then transferred from the Arduino to the Raspberry Pi using a USB cable, and the second Arduino Mega to control stripped LED lights placed around the structure. This integration enabled control over all three motors,

which operated the robot's rail system and both arms. The past hardware system is depicted below in figure 7.

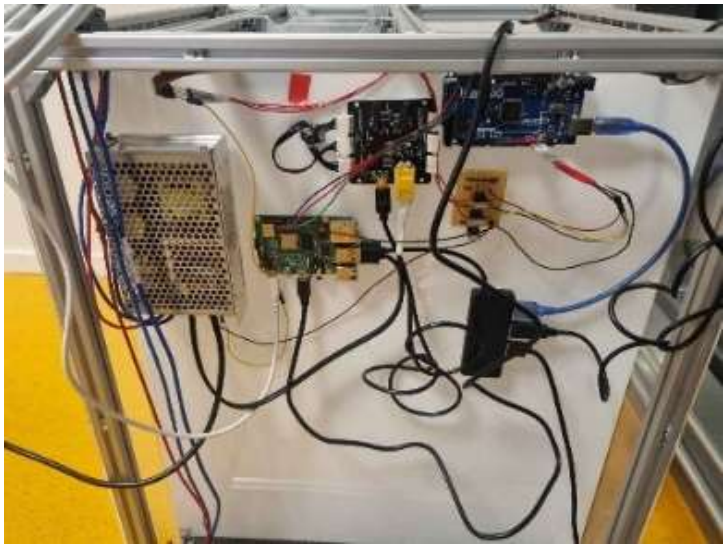


Figure 10 Previous group hardware design and implementation on the backside of the arm

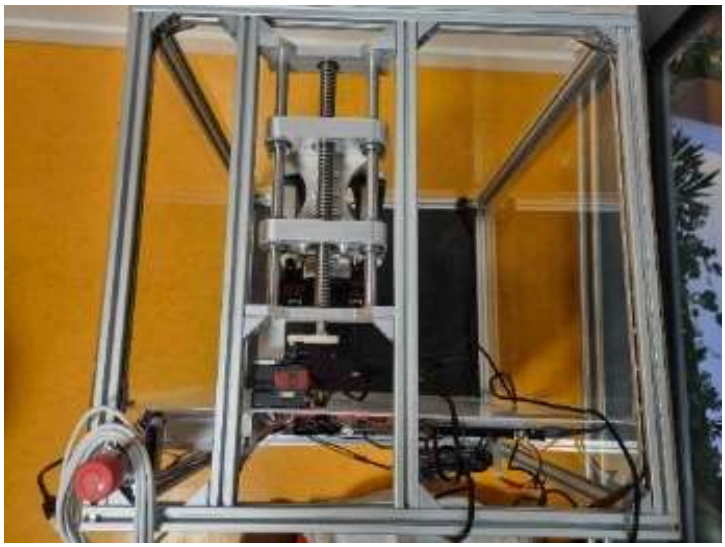


Figure 11 Previous group design from top view with a stop button on the left corner and a HS1 servo motor connected to a gear controlling the railing

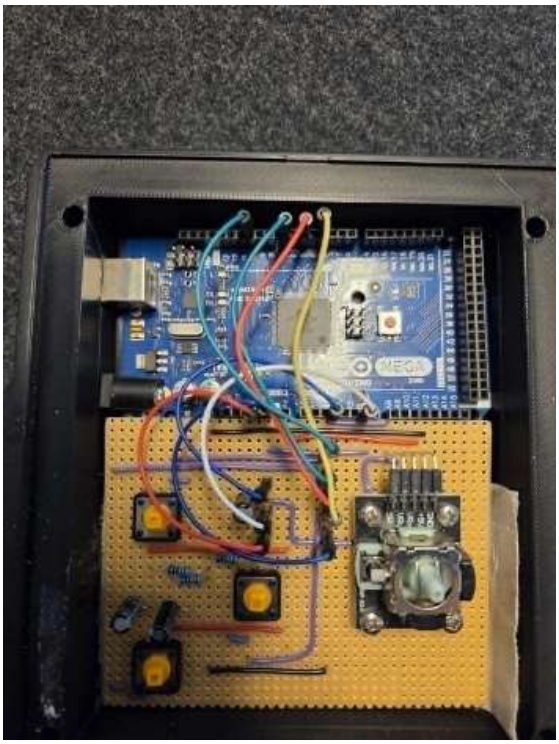


Figure 12 Inside the previous controller with an Arduino Mega board connected with a self-made PCB board with buttons and a joystick for input signals.

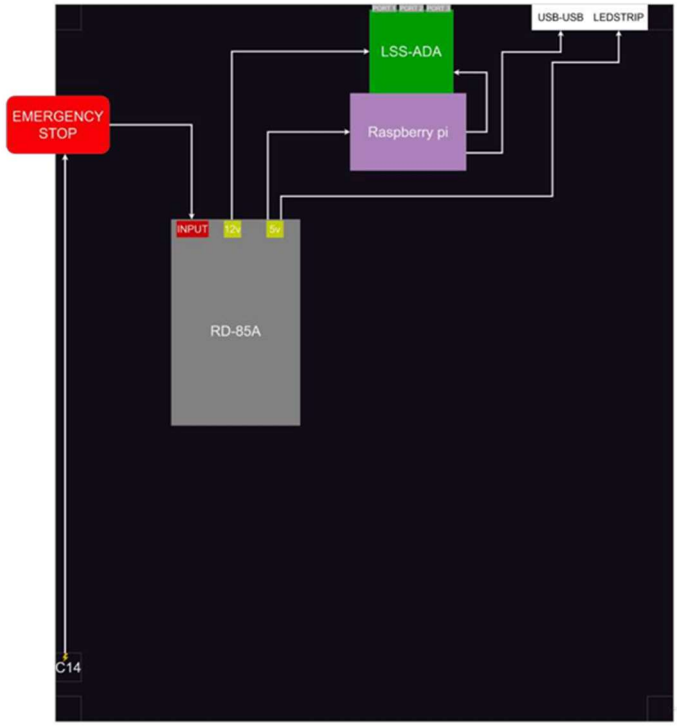


Figure 13 Previous group power distribution diagram from their thesis (Magnor et al., 2024)

4.2.2 Hardware Challenges

According to the previous group's thesis, several setbacks emerged during their semester related to the hardware, particularly in the controller and the arm movements driven by the HS1 servo motors. These challenges were not only described in their thesis but also confirmed through observations during the initial testing with the robot.

Controller and Movement

Initially the design of the controller was to use the same PCB board, connected to an Arduino Nano in figure 14. The main reason for this implementation is due to the size difference of a Nano compared to a Mega. In this way the controller could potentially have been designed in a smaller size, making the controller cleaner, attractive and practical as explained in the thesis. However, when the Nano was set up with the PCB of the controller and tested by their software department, they experienced the Nano to not live up to their desired outputs. Their reasoning was that the Nano was most likely a cheap knock of the original, and when borrowing a Nano which was produced by Arduino, they still experienced problems with the software not performing the wished outputs for the motors. Due to time creeping, ordering a new Arduino Nano would not arrive in time, therefore they decided the design of the controller would consist of an Arduino Mega communicating with their self-made PCB board with joystick and buttons input.

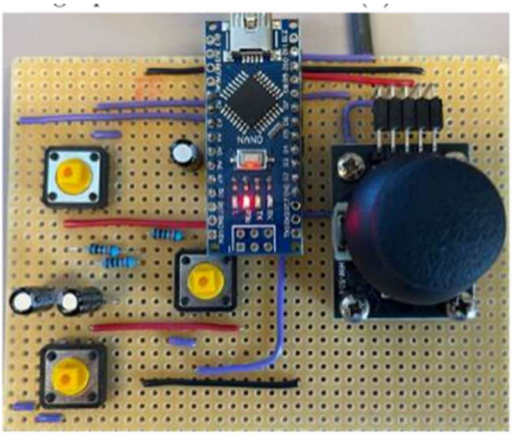


Figure 14 The potential design of the controller with a Nano board instead of a Mega.

In addition to the controller issues, the previous group encountered issues related to the movement of the robotic arm's joints. Their initial objective was to implement inverse kinematics for both motors that controlled the two arm joints. The goal was to allow the user, via controller input, to specify a desired position for the gripper. The motors would then receive coordinate values and calculate the required rotation angles to reach that position, as depicted in figure 15. The key advantage of this method was that it would allow the arm to move freely and accurately in any direction the user desired, maximizing its range and flexibility.

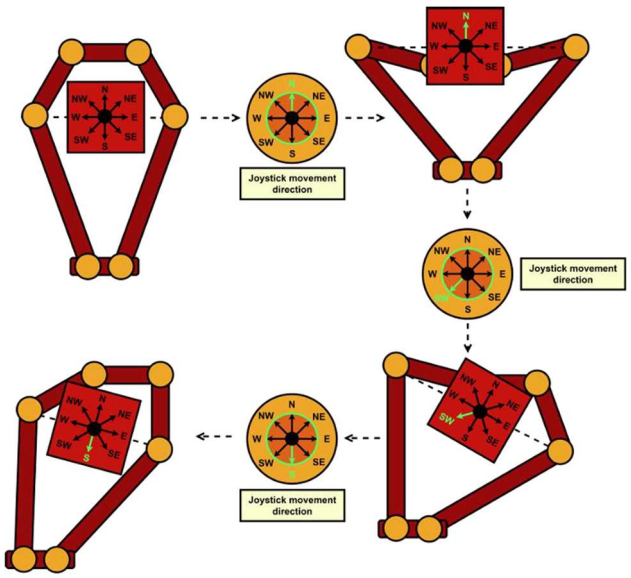


Figure 15 Representation of the desired movement with the joystick on the controller by the previous group from their thesis (Magnor et al., 2024).

However, the group was not able to implement inverse kinematics. According to the thesis, their explanation was a fault in the inverse kinematics code. To overcome this, they attempted to use a tool inside ROS called Mapper Node, which allowed them to visualize the reachable positions of the robot by creating a virtual space for the arm as in figure 16. The software would help the user to validate controller inputs by determining whether the requested movement was physically achievable. While they mentioned the Mapper Node did improve the motion, the system still failed to perform as they wished, and the group decided to not use inverse kinematics, but rather what they described as a simpler method, mainly due to not having enough time.

Based on observations, when testing the simpler method referenced by the previous group appeared to involve symmetrical motor rotations. When the gripper was moved vertically up or down, both motors rotated in opposite directions, enabling the two arms to bend symmetrically. This allowed for basic vertical movement of the gripper without any deviations from the location of the gripper. However, this method proved to have its limitations when moving the gripper horizontally from side to side.

The main issue was that the gripper could not extend to its full potential range. This was because the motors did not compute precise rotation angles based on controller input. Instead, they were programmed to rotate in opposite directions up to a fixed angle, likely set manually as a limit. As a result, the movement lacked the dynamic adjustment needed for accurate positioning and full reach of the arm.

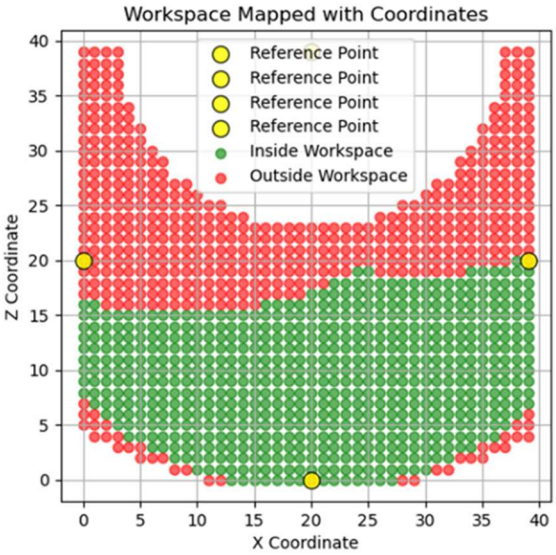


Figure N.13: Mapped workspace visualized with matplotlib

Figure 16 The Mapper Node, a software to visualize the reachable limits of the arm from their thiesis (Magnor et al., 2024).

4.2.3 Reuse of Previous Hardware Components

Last year's project, developed by the previous group, was guided by a different set of goals and requirements. On the other hand, this year's focus is on building upon their foundation and taking the robot a step further based on the new objectives described in the project assignment section. One key requirement made clear by our supervisor from Tronrud Engineering during multiple meetings prior to project start was to reuse as many existing hardware components as possible. The idea behind this plan was to increase our efficiency and save time and money without starting the same project all over again, especially parts which were already fulfilling the requirements of this year's project.

Power supply and Stop button

It became clear when analyzing the robot that both the power supply and the emergency stop button would be kept. These two components, already physically integrated, also fulfilled one of the requirements outlined by Tronrud Engineering which were a 230 (Well, n.d) volt from a power outlet and a stop button to shut down the system in case of emergency.

The stop button in figure 17, designed as a normally closed (NC) contact, was wired to the AC input of the power supply (Electric, n.d). It functions as a safety mechanism, and when active it allows AC current from the wall outlet to flow into the power supply, which then converts the current to a suitable DC voltage for other components connected to the system. If an emergency occurs and the button is pressed, the current flowing through the wall to the

power supply is stopped immediately for all components connected to the outputs of the power supply.

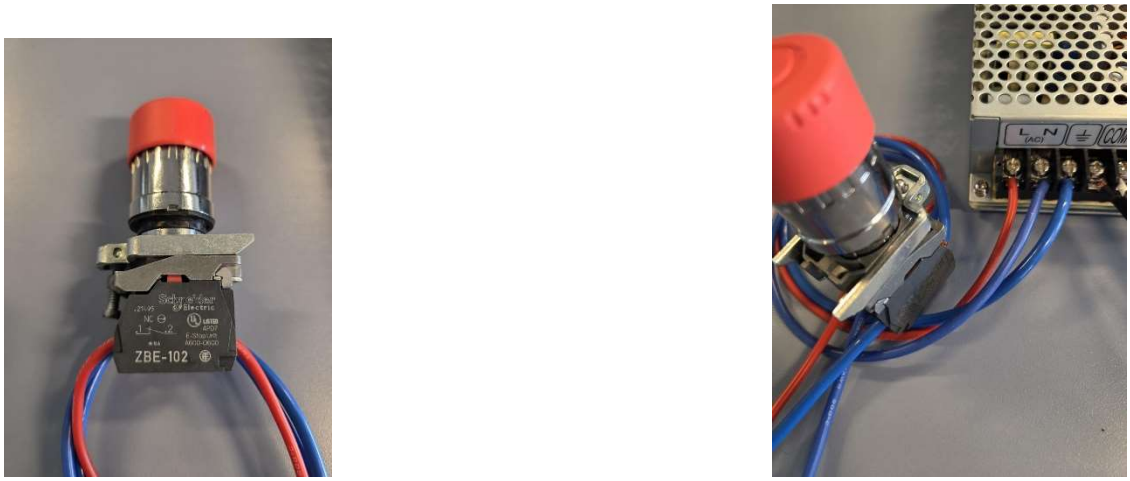


Figure 17 Leftmost picture shows a closed stop Button while the rightmost picture shows the stop Button connected to AC power suply

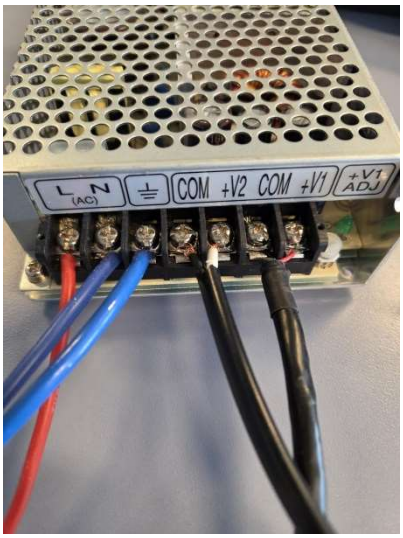


Figure 18 Input and output of the power supply with 12- and 5-volt DC output for +V2 and + V1

LSS ADA Driver and HS1 Servo Motors

The previous groups actuation system consisted of three identical HS1 servo motors,

connected to recommended 12 volts (Nantel, 2024a) provided from the power supply and further connected to the Raspberry Pi with a USB cable. Two of the HS1 motors were connected to the arm joints controlling the arms, while the third HS1 motor controlled the railing system, enabling movement across all three axes. While these HS1 servos are in general greatly suited for most small robotic applications, their torque of 0.78 Nm, RPM of 100 RPM and weight of 58 grams each motor (Nantel, 2024a), did not satisfy the mechanical department and demanded servo motors with higher power requirements for this year's objectives. However, the HS1 servo motors were allowed to be reused for the railing and the gripper because they have the necessary specification such as a closed loop system explained in section 3.3, providing feedback on their rotational angle.

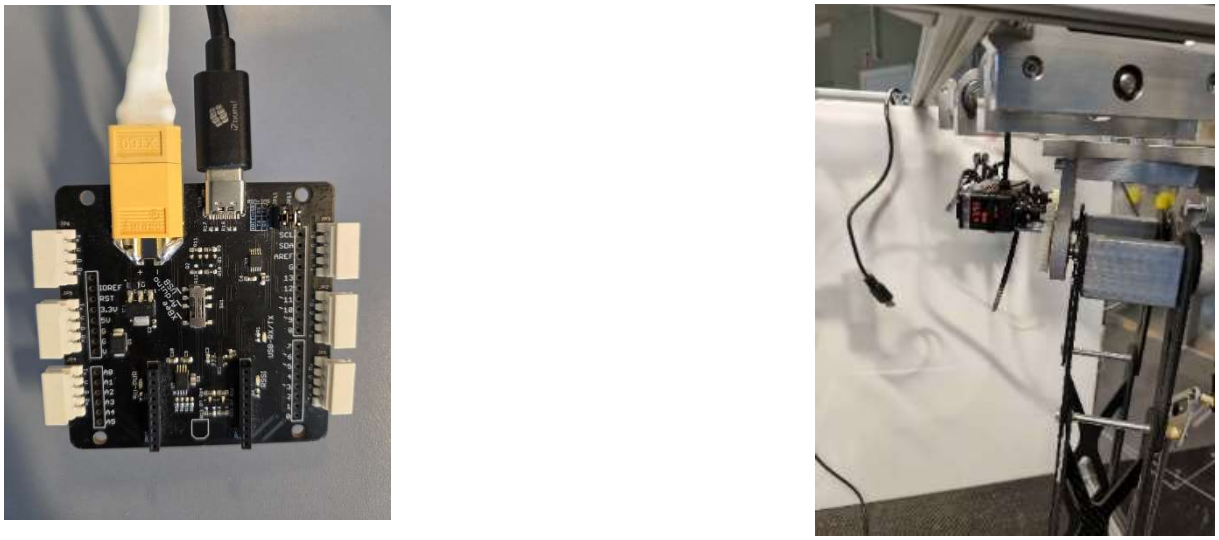


Figure 19 The rightmost picture shows two HS1 motors placed to control the arm from previous group while the leftmost picture shows HS1 driver connected to the 12-volt output of the power supply with a USV - C cable

Raspberry Pi 4 Model B

Through research into the components we already possessed, we found the Raspberry dominates all relevant specifications needed for this project compared to what the Arduino Mega boards could provide (Arduino, 2025a). The second ability possessed by the Pi, which we considered to be the most important factor motivating us to reuse the Raspberry, was because of the built-in Bluetooth capabilities and enabling wireless connection with a controller (Ltd, 2024). In addition, Raspberry Pi has the programming language python as its

native speaker (Ltd, n.d). Therefore, downloading necessary packages to interact with the servo motors would most likely not become a major issue.

Figure 20 Raspberry Pi 4 model B

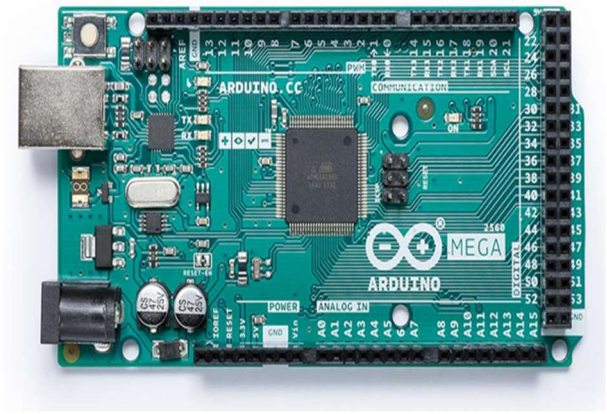


Figure 21 Arduino Mega from the previous design (Arduino, 2025a).

Table 1 The comparison between the two available boards from the previous group.

	Rasperry Pi 4 Model B	Arduino Mega
Clock speed	1.5 GHz	16 MHz
Architecture	64 – Bit	8 – Bit
CPU cores	4	1
SRAM	8 kb	8 GB
Bluetooth	Yes	None

4.2.4 New Hardware Requirements and Changes

The main improvements in the hardware part of the project were the controller design and the implementation of inverse kinematics, both of which are together the key to achieving full range movement of the robot arm are all based on the analysis of the previous robot. While it was clear for us on how to plan and develop improved solutions, we still had to align ourselves with specific criteria's provided by Tronrud Engineering. These requirements and the previous group's robot together helped us decide how to approach the hardware part of the project.

Based on the required functionalities and previous group design:

Previous group requirements:

- 2 HS1 servo motors for joints of the arm
- 1 HS1 servo motor for railing
- 1 self-made wired controller

This year's requirements:

- Gripper
- Inverse kinematics for joint of both arms
- Wireless controller
- Display for user interaction
- Sensor to capture the score

Our own software department decided to implement a PS4 controller, two new Servo motors and a Beam Break sensor to fulfill the criteria.

Dynamixel Motors (XC430-W150) and Diver (Power Hub & U2D2)

Both Dynamixel motors have a torque of 1.6 Nm and RPM of 106 and only wights 65 gram each individually (Robotis, n.d-c), which compared to the RPM and the weight of the HS1 motors is not a major difference, but the torque factor is a substantial improvement, satisfying the mechanical department.

Both are servo motors designed as a closed loop system explained in the theory section and are programmable with python which is a requirement from the software department and would allow the motors to achieve inverse kinematics by real time monitoring their own position.

Both Dynamixel motors are supported by a Power hub and a U2D2. The power hub is the main driver between the motors with an additional adapter, providing a 12-volt output. (Robotis, n.d-b). The U2D2 is a USB communication converter providing data to the servo motors (Robotis, n.d-a).

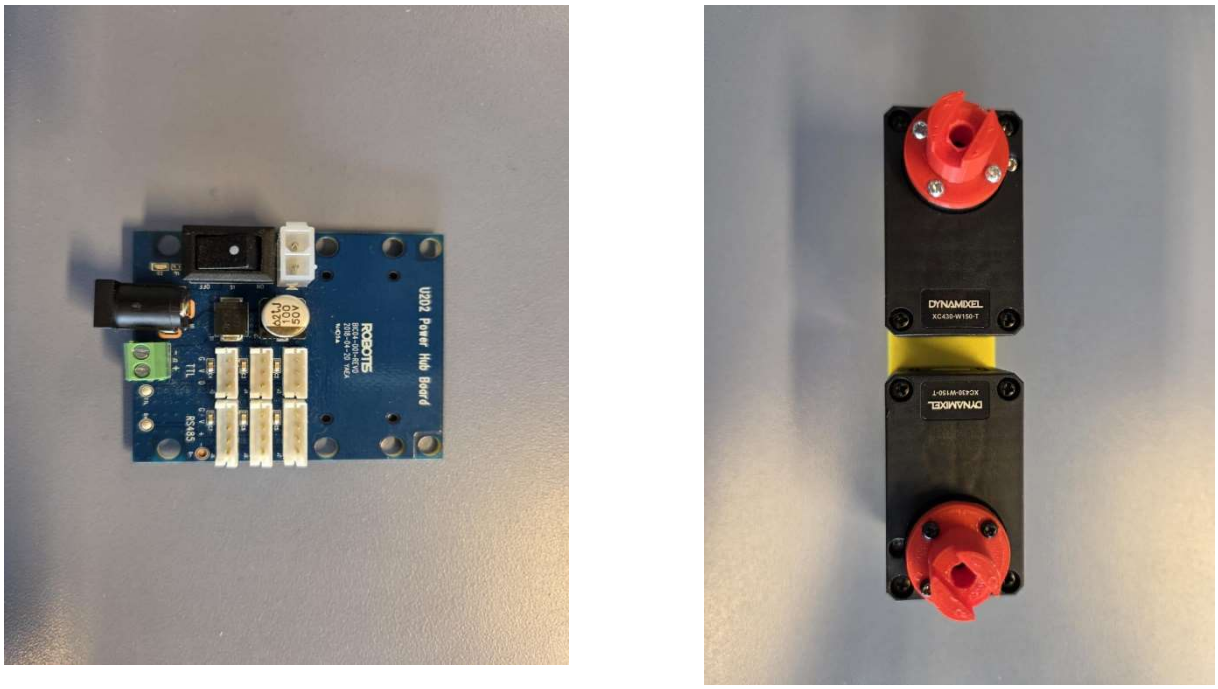


Figure 22 rightmost picture shows the Dynamixel driver(powerhub), while the leftmost picture shows Both Dynamixel servo motors for both arm joints replacing both previous HS1 servo motors at the same locations.The red 3D printed parts are gear supporters.

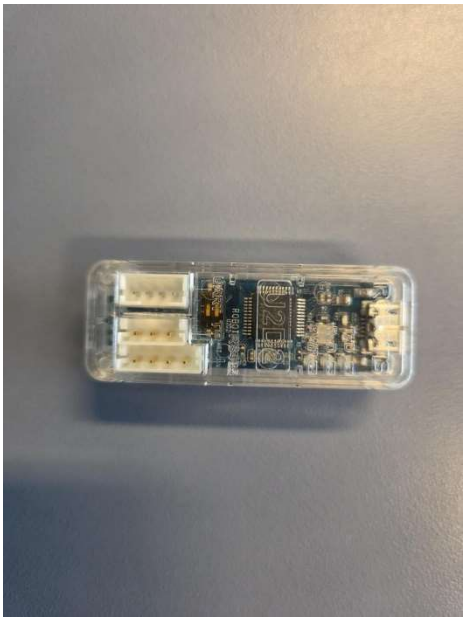


Figure 23 U2D2, responsible for data transfer between the Raspberry and the driver, connected to both driver and the Pi

Figure 24 Adapter to be connected to a wall and the driver supplying 12 volts

Table 2 Both motors have the same capabilities, but the servo motors from Dynamixel had a greater torque ability than the HS1 servo motors, which was the parameter satisfying the mechanical department.

	Torque (Nm)	RPM	Feedback loop	Python	Weight
Dynamixel	1.6	106	YES	YES	65 g
HS1	0.78	100	YES	YES	58 g

PS4 – Controller

The controller designed by the previous group, whether they had used the Arduino Mega or the Nano, needed to be replaced regardless of the requirements from Tronrud Engineering.

One of the key criteria was that the controller must be wireless, which the existing solution did not fulfill. To meet the wireless requirement by Tronrud Engineering, we replaced the previous group's controller with a PlayStation 4 controller with built in Bluetooth (SONY, n.d). This implies a wireless connection to the raspberry compared to the previous groups wired solution.

Figure 25 PS4 Controller

Display

An interactive user experience was a desired outcome from Tronrud engineering. The idea is to have as many customers as possible to interact with robots, showcasing what the company is capable of producing in different fields of engineering and attracting potential customers. The best solution for an interactive experience was to integrate a display designed to be compatible with the Raspberry Pi. It is a 7 inches screen with a DSI display wire connector, including touch functionalities (Ltd, n.d-a).

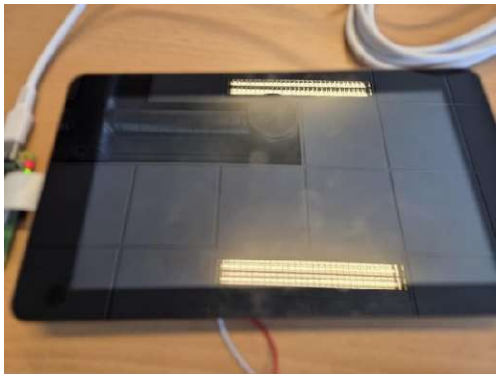


Figure 26 Front side of the display

Figure 27 Backside of the display with the DSI connector on the left side of the image

IR Beam Break Sensors

In an interactive event with a scoreboard system, the interface has to record certain accomplishments such as standard points. To keep track of the score through a game developed by the software department, a beam break sensor was implemented to detect incoming balls.

The sensor is designed as two parts where one is the emitter, and the other one is the receiver. An infrared light is sent from the emitter and when pointed at the receiver a stable infrared light is produced. Both parts operate at 5 volt and common ground, and the receiver has in addition a third wire which is connected to a GIOP of the PI (RoboShop). When a separation occurs by an object, the receiver communicates with the Pi and updates the score based on the number of light separations, further displayed on the screen.

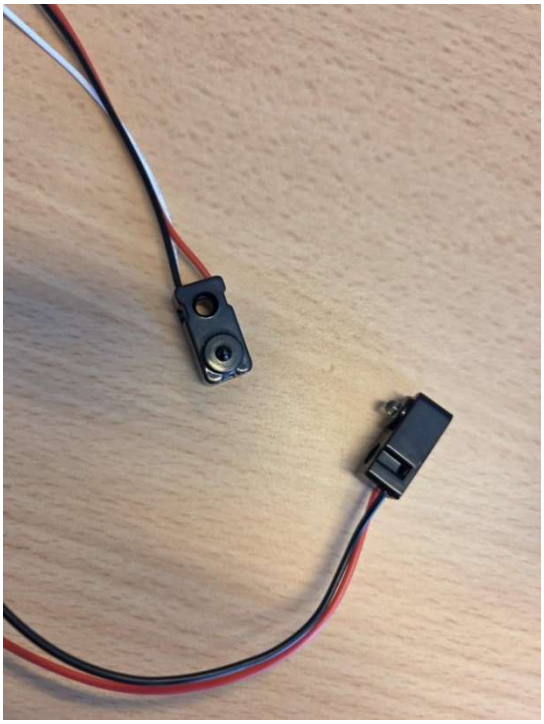


Figure 28 Emitter to the right and the receiver on the left.

4.2.5 Firmware

Firmware is an essential tool for programming each servo motor with a specific ID, which is later utilized by the software department. In addition to its programming function, the firmware is also used by both the hardware and mechanical departments to test motor performance. In this project, both types of servo motors, the HS1 and the Dynamixel both have their own dedicated firmware installed on a laptop. Each motor can be accessed by inserting the data transfer cable from its corresponding driver into the laptop.

Dynamixel Firmware

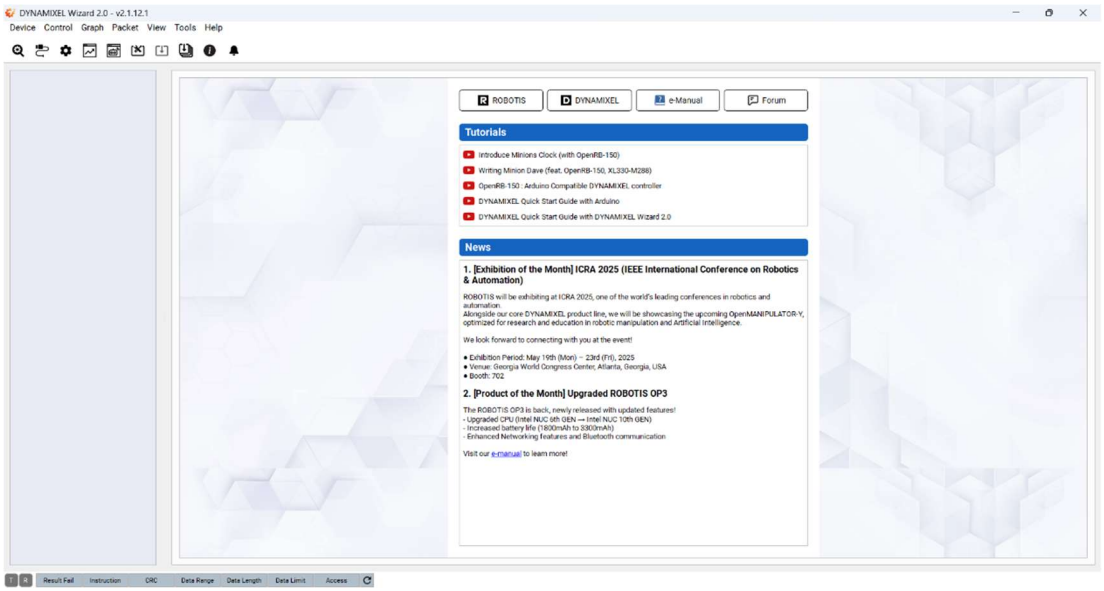


Figure 29 The Dynamixel servo motors use the firmware called Dynamixel Wizard 2.0. Image illustrates the firmware without a communicating motor.

LSS HS1 Firmware

Figure 30 The HS1 servo motors use the firmware called LSS Config

4.3 Software

4.3.1 ROS

Robot Operating System (ROS) is an open-source software specialized in the field of robotics, offering tools and libraries for handling typical robotic tasks such as communication between sensors, controllers, and actuators (ROS, n.d).

4.3.2 Matlab

Matlab is a programming platform to analyze data, develop algorithms and create models and simulations (Matlab, n.d).

4.3.3 Python

Python is a high-level programming language widely used in robotics and automated systems. Due to its wide adoption, a large selection of libraries is highly available, many of which are compatible with common actuators and sensors used in the industry. This makes Python a flexible and powerful tool for developing robotic control systems and handling various communication protocols across different hardware platforms (Python, n.d).

4.3.4 Python Libraries

NumPy: For numerical computing in Python (GeeksforGeeks, 2025c).

PyQt5: Graphical user interfaces (GeeksforGeeks, 2022).

PyQtGraph: Real time visualization of simulations (GeeksforGeeks, 2020).

RPi.GPIO: Generates communications for general purpose pins of the Raspberry Pi (GeeksforGeeks, 2025a).

pygame: Platform for reading inputs of a controller such as a PS4 controller (GeeksforGeeks, 2025b).

dynamixel_sdk: Software provided by Dynamixel servo motors to communicate with python (Robotis, n.d-c).

LSS-Python: Library for HS1 servo motors to communicate with python (Nantel, n.d-b).

5. Mechanical Design and Development

This chapter presents the full mechanical design and development process of the robotic arm. It is structured around three subchapters: Robotic Arm Design and Changes, Robotic Gripper Design and Changes and Motor torque and RPM requirements. Due to limited manufacturing capacity at Tronrud Engineering (TE), the easier components in the final designs were produced in collaboration with Hønefoss videregående skole. These parts were designed to be manufactured, although the parts won't be completed before the submission of this report.

5.1 Robotic Arm Design and Changes

This section goes through the mechanical changes made to the arm itself. It includes updates to parts such as the Festehub, the stabilizer, and other general arm components. The goal was to generally improve the arm functionality and to fix problems with the previous versions.

5.1.1 Festehub Assembly Designs and Changes

One of the requests from TE regarding the robotic arm was to address the existing gear problem. The original design relied on the small gear to “rest” in the larger gear, as shown in **Figure 31**. This design often caused the small gear to slip out of position, resulting in a complete loss of movement in both the X and Z axes. During testing, the only way to maintain movement was for one person to manually hold the small gear in place to keep the gears in contact. This was not an optimal solution for further testing and a redesign was required to ensure contact without human help

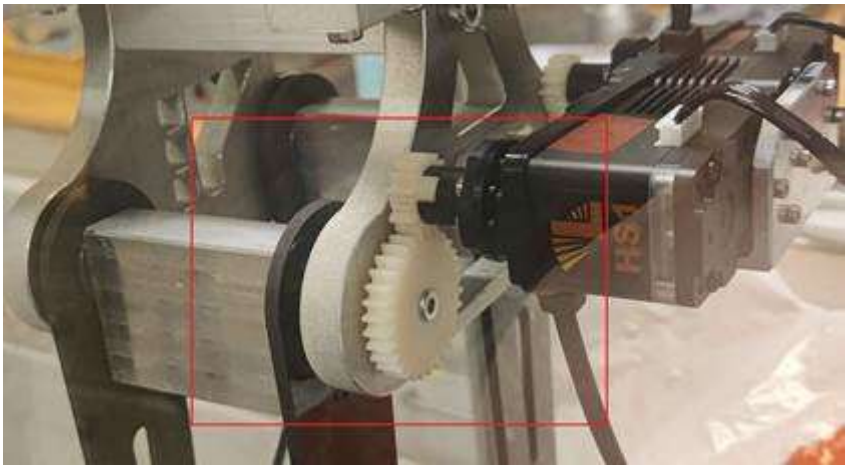


Figure 31 Photograph of the Festehub Assembly. Red square shows the smaller gear and larger gear in question.

Festehub Assembly 1.0:

The initial approach to fixing this problem was to create a connection on both sides of the gear. This involved not only support from the motor’s side, but also a secondary connection through the backplate, shown in **Figure 32**. This would make a connection on both ends of the gear shaft, ensuring the gear remained in position, allowing for smooth and accurate movement.

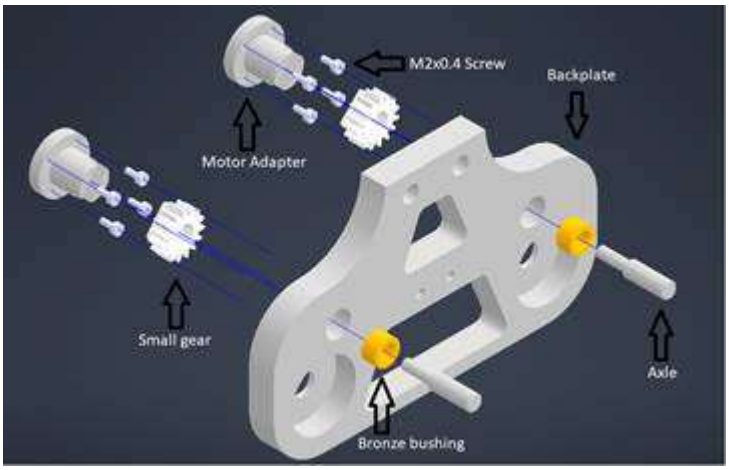


Figure 32 Exploded-view drawing of the Festehub Assembly 1.0, annotated with part names.

Backplate design 1.0:

To provide support with a secure secondary connection, a modified backplate was designed. A hole with a diameter of 10mm, H7 tolerance, and a depth of 6mm was added to align with the gear shaft, allowing a 6mm bronze bushing to be press-fitted into the plate, as shown in **Figure 32.** The backplate had a thickness of 10mm, and the central area had weight-saving cutouts as shown in **Figure 33** which is kept from the original design to meet TE requirement of keeping the total weight of the arm below 12kg. During testing the backplate itself functioned correctly and provided enough support to make the gear stay in place. However, there were other parts in the assembly that failed.

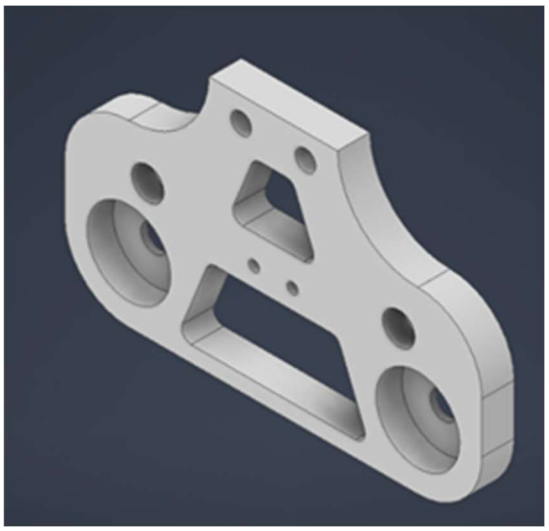


Figure 33 3D render of Backplate design 1.0.

Axle design 1.0:

To hold the small gear in place, a custom axle (**Figure 34**) was designed to replace the original screw that connected the gear to the motor adapter. By adding another point of contact through the bronze bushing the axle simultaneously provided support. The axle consisted of a 6mm diameter shaft with g6 tolerance, allowing for a

close fit with the 6mm inner diameter bronze bushing while still enabling free rotation. To secure the axle in place, a 4mm diameter threaded section (M4x0.7) was included to fasten it into the motor adapter and gear. As TE no longer had the capacity to produce custom metal parts, the axle was 3D-printed in PLA. During testing, the printed axle frequency broke or slipped out of place due to 3D-printers not being able to print small threaded sections. As a result, the axle was replaced with a shoulder screw in *Festehub Assembly 2.0.*

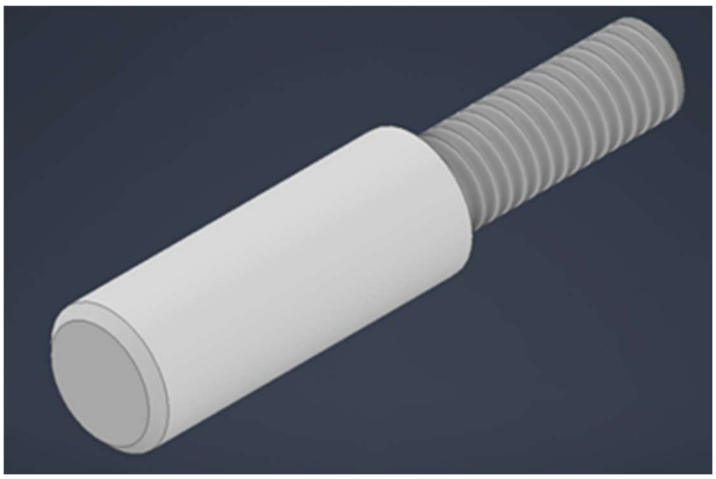


Figure 34 3D render of Axle design 1.0, with its 6mm shaft and 4mm threaded sections.

Motor adapter design 1.0 and small gear:

The motor adapter's function (**Figure 35**) was to connect the small gear and the axle to the motor shaft. As TE required these parts to be 3D-printed, the original design of the motor adapters was kept, with minor dimensional adjustments made to adapt to the new XC430-W150-T Dynamixel motors. One of the reasons for keeping the original adapter design was to maintain compatibility with the existing gears, which had been printed in ABS filament using a Bumbu Lab X1 Carbon FDM printer. These gears were of higher quality than what could be produced at Makerspace.

The original gears were not manufactured to industrial standards, as shown in **Figure 36**. To implement industrial grade gears, the motor adapter would have needed to support features such as a keyed shaft connection, as shown in **Figure 37**. This would have required the adapter to be produced in-house, which was not possible at the time. As a result, the original gears were reused for further testing. After several tests, the original gears failed. To continue further testing on the robotic arm, new gears had to be printed. In Festehub Assembly 2.0, the decision was made to merge the small gear and the motor adapter into a single component to simplify the design and reduce part count.

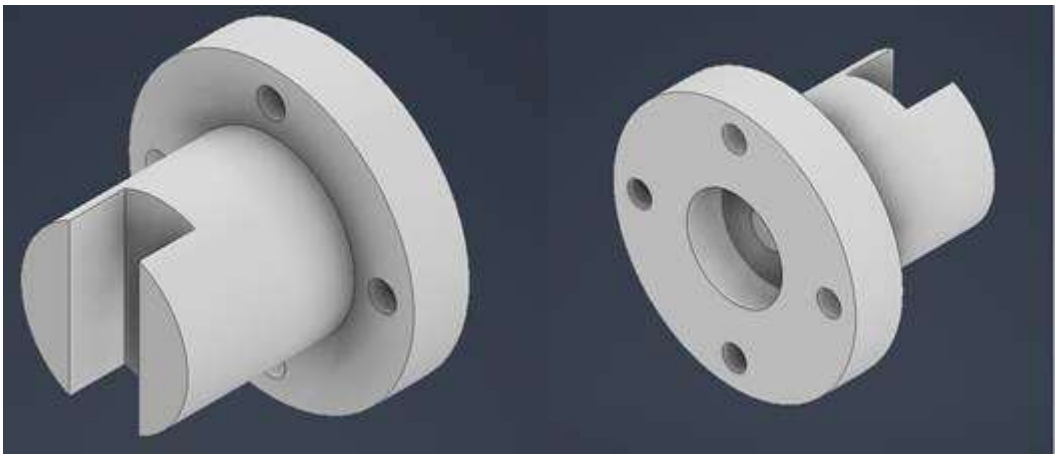


Figure 35 3D renders of Motor adapter 1.0, from two different angles.

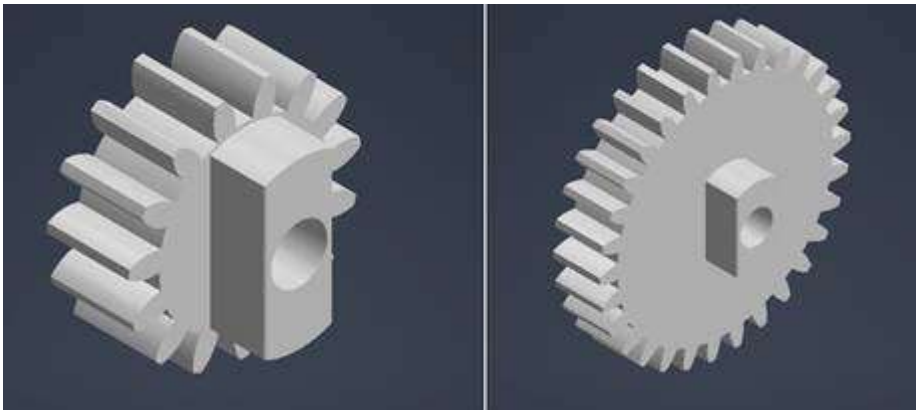


Figure 36 3D renders of the small gear (left) and big gear (right). Showing the non-industrial design which is the extruded part in the center.

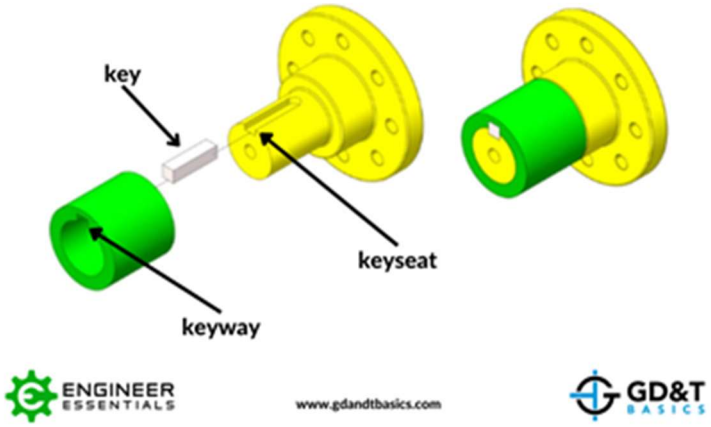


Figure 37 Diagram showing an example of a keyed connection design, (From Keyseats and Keyways, by GD&T Basics, 2022, https://www.gdandtbasics.com/keyseats-and-keyways/. Used under fair use for educational purposes.)

Festehub Assembly 2.0:

Festehub Assembly 2.0 (**Figure 38**) represents the final design. Several modifications were made to fix the problems in Festehub Assembly 1.0. As previously mentioned, the custom axle was replaced with a shoulder screw to improve durability. Additionally, the motor adapter and small gear were merged into one single component, with minor dimensional adjustments for easier assembly.

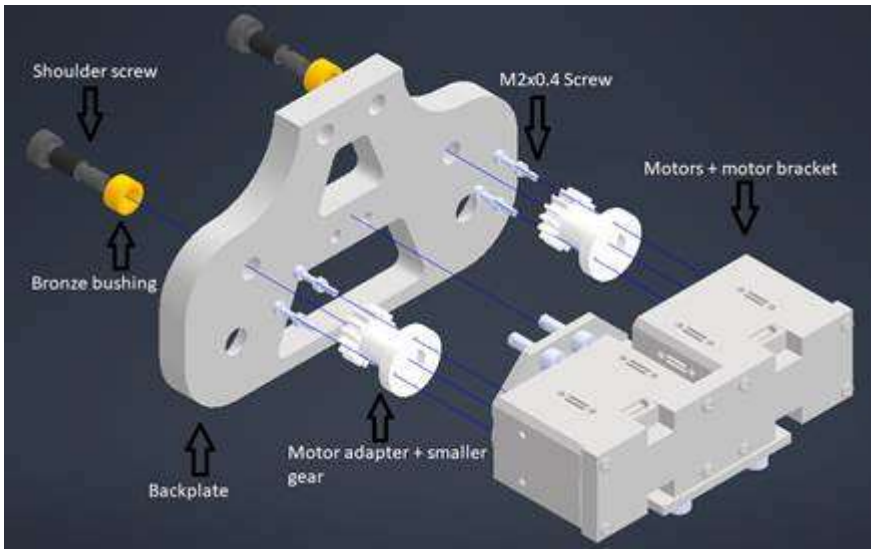


Figure 38 Exploded-view drawing of the Festehub Assembly 2.0, with the motors and motor brackets, with part names annotated.

Replacement of Axle with Shoulder Screw:

As described in Axle Design 1.0, the axle was replaced with a shoulder screw after it, made from 3D-printed PLA failed under load. Since TE preferred to reuse available components from inventory, the decision was made to use a shoulder screw as it had all the needed properties, and it was in stock at TE.

The shoulder screw offered several advantages that addressed the failures of the original axle. As shown in **Figure 39**, the screw features a head for fastening, an M5x0.8 threaded section for connection to the motor adapter and gear, and a smooth 6mm diameter shoulder section with f9 tolerance, which functions as a shaft. This tolerance provides a suitable fit with the 6mm H7 bronze bushing, ensuring smooth rotation while maintaining axial stability. This together with robustness makes it a great replacement.

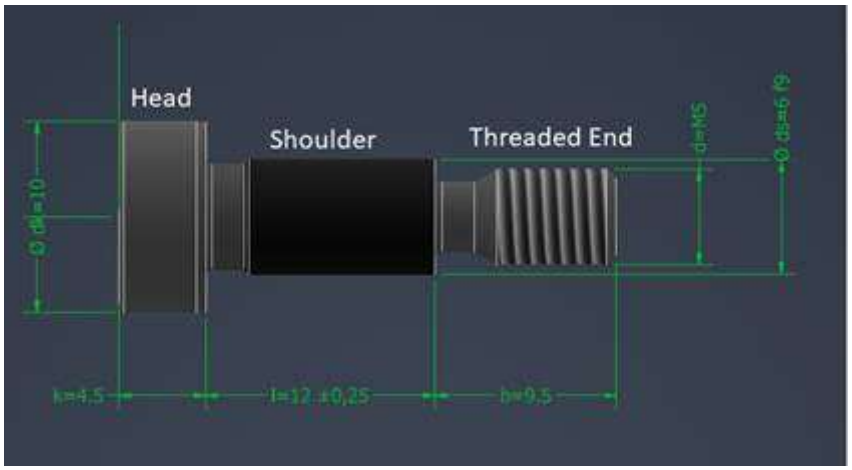


Figure 39 Diagram of the shoulder screw with dimensions, names and tolerances.

Backplate design 2.0:

The initial plan was to use the first backplate design, as it functioned as intended. However, when the axle was replaced with a shoulder screw, some details were overlooked, such as the screw head height. While the original axle sat flush with the surface, the head of the shoulder screw sticked out from the backplate. This caused the arm to collide with the screw head during movement, as shown in **Figure 40**.

To fix this, a 12mm diameter and 2mm deep clearance hole was added to the screw hole in the backplate, creating clearance for the screw head, as shown in **Figure 42**. This also made the bronze bushing be bushed deeper into the backplate. This adjustment was made easily in the 3D model and did not affect anything. As shown in **Figure 41** starting from right, there is the clearance hole of 12mm diameter, then the 10mm diameter H7 hole for the bushing and last a 7mm diameter hole at the end to prevent the bushing from falling out.

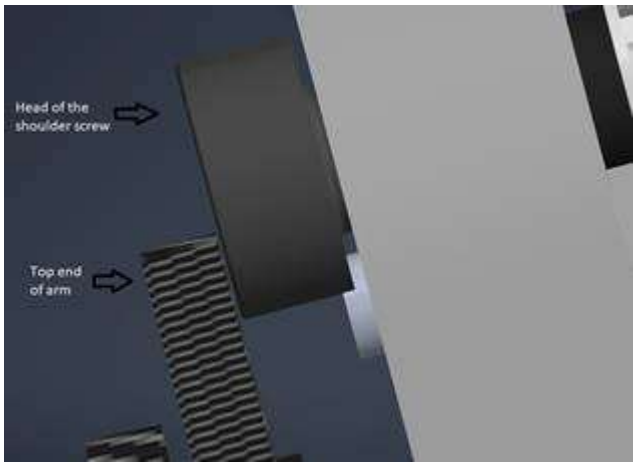


Figure 40 Close-up 3D render of the arm assembly. Note the particularly narrow clearance between the shoulder screw head and the top of the arm.

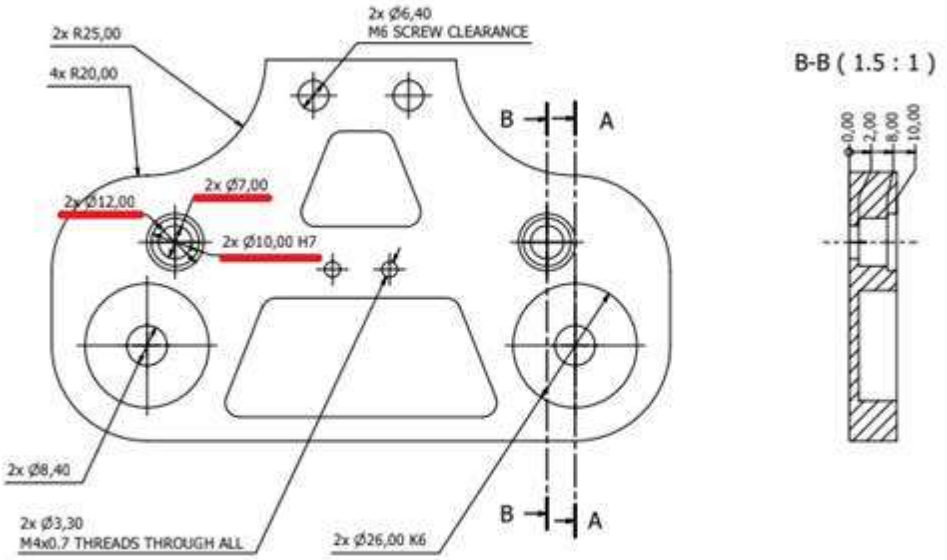


Figure 41 Technical drawing of Backplate design 2.0. The left view is from the front, and hole dimensions referenced in the main text are underlined in red. The right view is a cutaway view that shows the holes from the side.

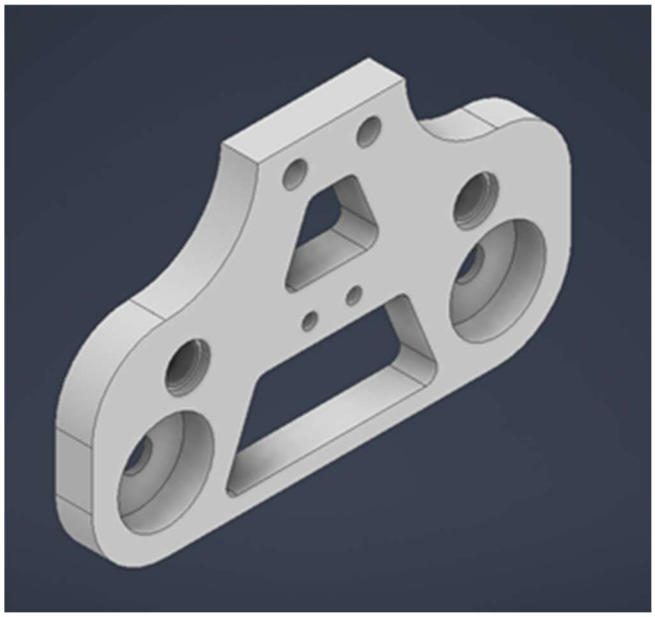


Figure 42 3D render of Backplate design 2.0

Motor adapter with small gear design:

Since both the small gear and the motor adapter were 3D-printed, there was no reason to keep them as separate components. To reduce the number of parts and

simplify assembly, the two were merged into one single component, as shown in **Figure 43**. One modification made for this component was to increase the diameter of the central hole to 4.2mm, allowing the M5x0.8 threaded section of the shoulder screw to fasten directly into the plastic. Although the hole was intended to be threaded, threads produced by 3D-printer are often weak and fail. As this was the case, it would be better to manually thread the hole with an M5x0.8 threading tap.

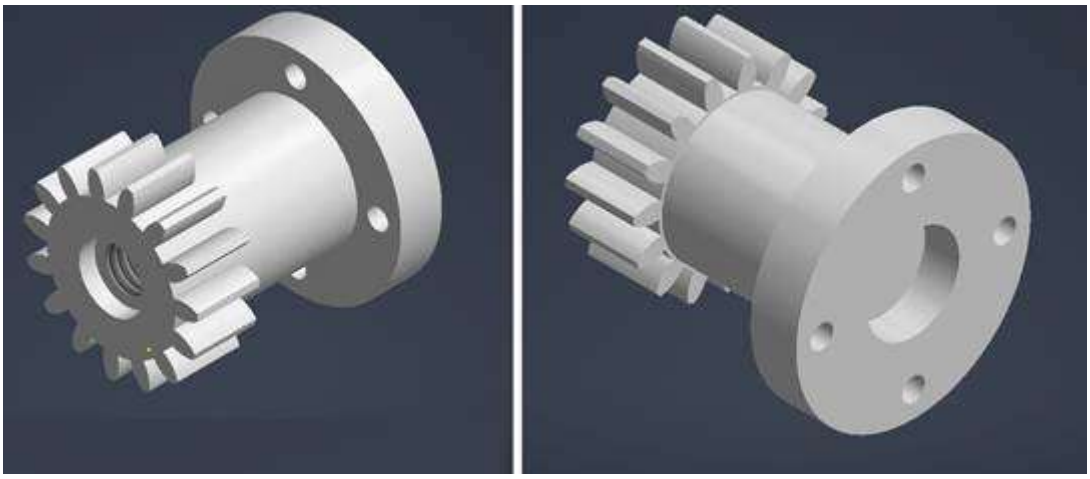


Figure 43 3D renders of the new motor merged motor adapter with small gear design. Left frame shows the front view and the right shows the back view.

Motor bracket assembly:

The Motor bracket assembly (**Figure 44**) consists of two components: *bracket part 1* and *bracket part 2*. The function of these parts is to securely hold the motor in place and attach the entire assembly to the Festehub. The parts were designed as separate components rather than a single unit to simplify manufacturing. Bracket part 1, shown in **Figure 45**, includes M4x0.7 threaded holes (A), which align with corresponding holes in the backplate. The two additional holes (C) connect to bracket part 2, shown in **Figure 46**, and are also M4x0.7 threaded holes (E). To secure the motor, it is mounted to bracket part 2 using eight M2.5 bolts, which were supplied with the motors. These bolts fasten through 2.7 mm clearance holes (D), allowing for easy assembly since the threads are already present in the motor casing itself.

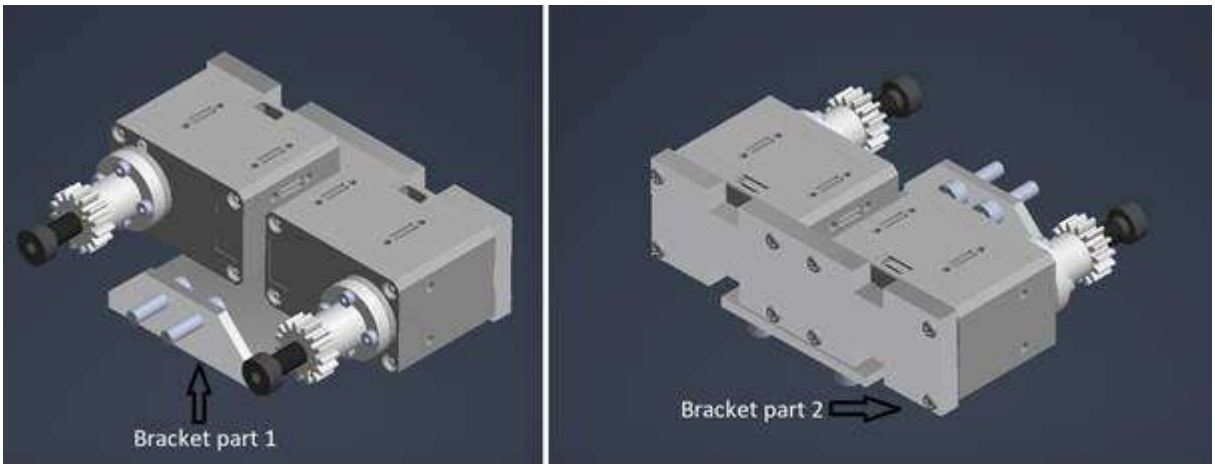


Figure 44 3D render of the motor bracket assembly, with annotations with part names.

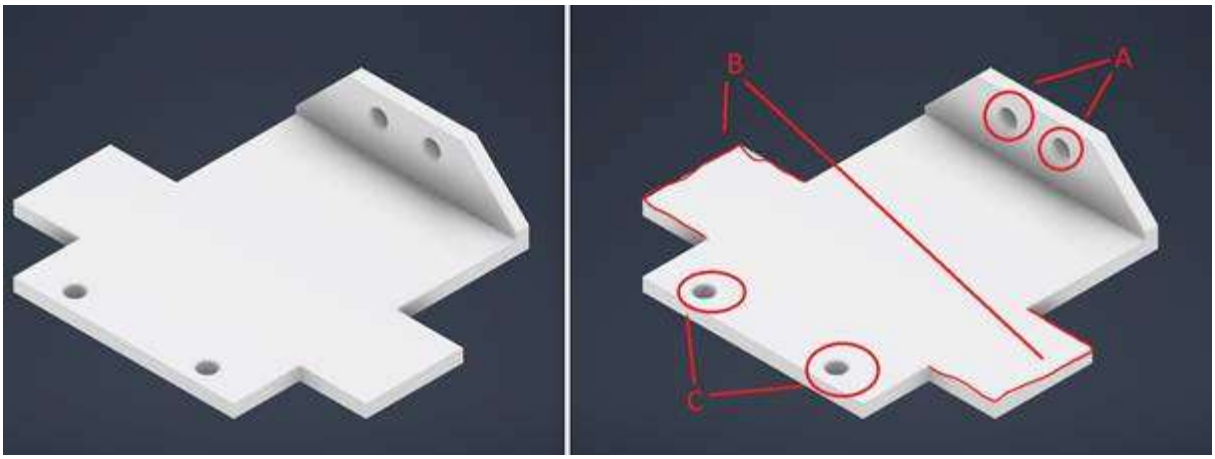


Figure 45 3D render of bracket part 1. On the right, red markings are added for clarity in the text.

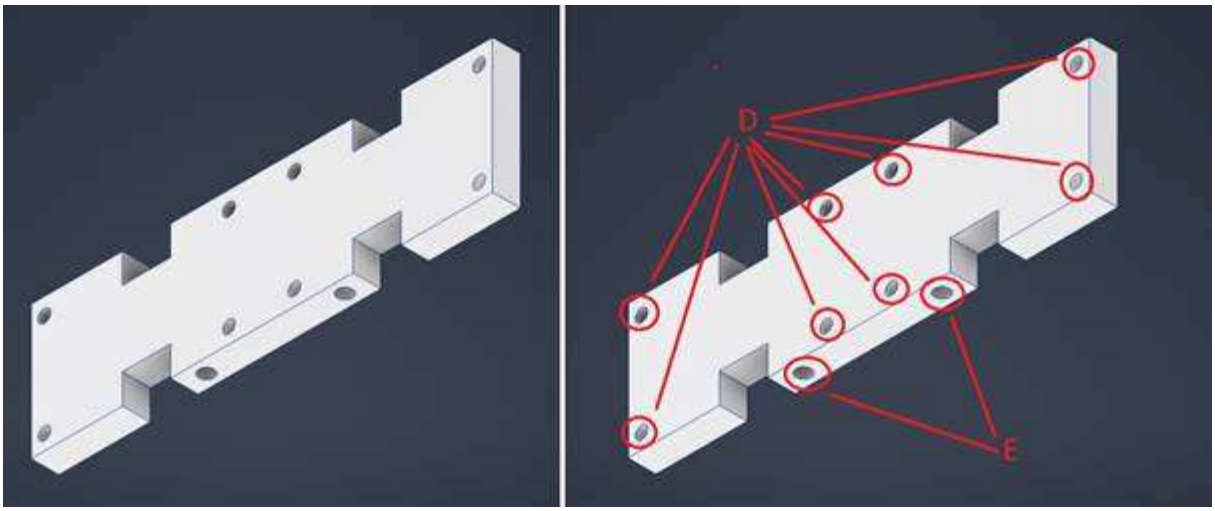


Figure 46 3D render of bracket part 2. On the right, red markings are added for clarity in the text.

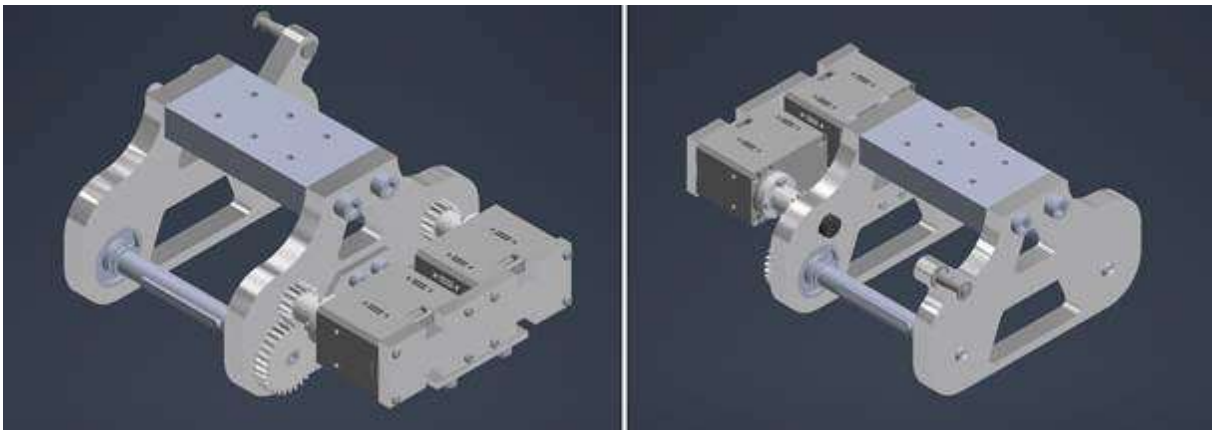


Figure 47 3D-render of the whole Festehub assembly with front view (right) and back view (left).

5.1.2 Stabilizer and Arm Designs and Changes

In the previous group's report (Magnor et al., 2024), it was explained that achieving a stable and horizontal tool position relies heavily on maintaining equal lengths between the stabilizing rods and the arm. This also includes the triangle, Festehub and Toolhub. The supervisor from TE also stated early one that this was one of the weak points of the robot and would like to see it function as intended. There were also other duo pod robots like this one with working stabilizers which were used for inspiration, as shown in **Figure 48**.

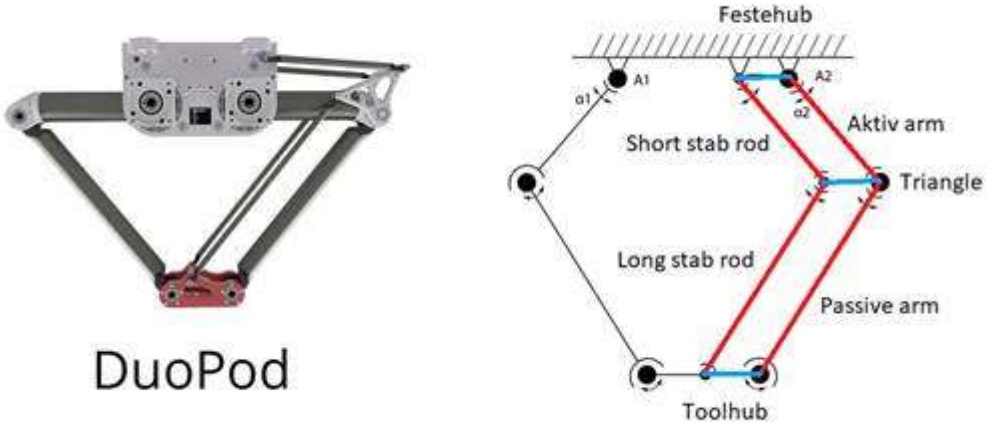


Figure 48 An adjusted picture. To the right is the schematic with added colors and names for easier understanding through the text, with names corresponding to this bachelor's robotic arm parts. (From Comparison of Kinematic Models, by Autonox Robotics GmbH, n.d., https://autonoxfinder.com/en/Kinematics-comparison/. Used under fair use for educational purposes.)

Troubleshooting:

The initial approach to diagnosing the faulty stabilizer was to examine all the components lengths, as this was most likely the cause of the issue. This was examined in two ways, first by measuring every part needed in the assembly 3D-drawing, and secondly by physically measuring the physical robot by hand. Shown in **Figure 49** some of the lengths are not similar, especially the length of the *long stabilizer rod* and the *passive arm*, which they should be. The other deviating lengths are negligible and won't affect the arm to any significant degree. When measuring the same lengths by hand, it confirmed our results from Autodesk Inventor 2025, indicating that this was probably the cause of the faulty stabilizer. This was something that would be fixed for this year's design.

Another cause of the general instability of the stabilizer may have been caused by its many long and thin aluminum rods. As seen in **Figure 50**, the stabilizer rods go far out from the arm, which a combination of being thin and long will cause deflection. After some troubleshooting, it was noticed that the cause of the far-out design was to eliminate any collisions with other parts. If the hinge pins were any shorter, the stabilizer rods would have collided with the Festehub when moving upwards. As the group wanted a specific height, this was not possible for them. For this year's design, the stabilizer would be moved closer to the arm. Before any changes to the stabilizer, first the arms had to be adjusted to fit this year's project.

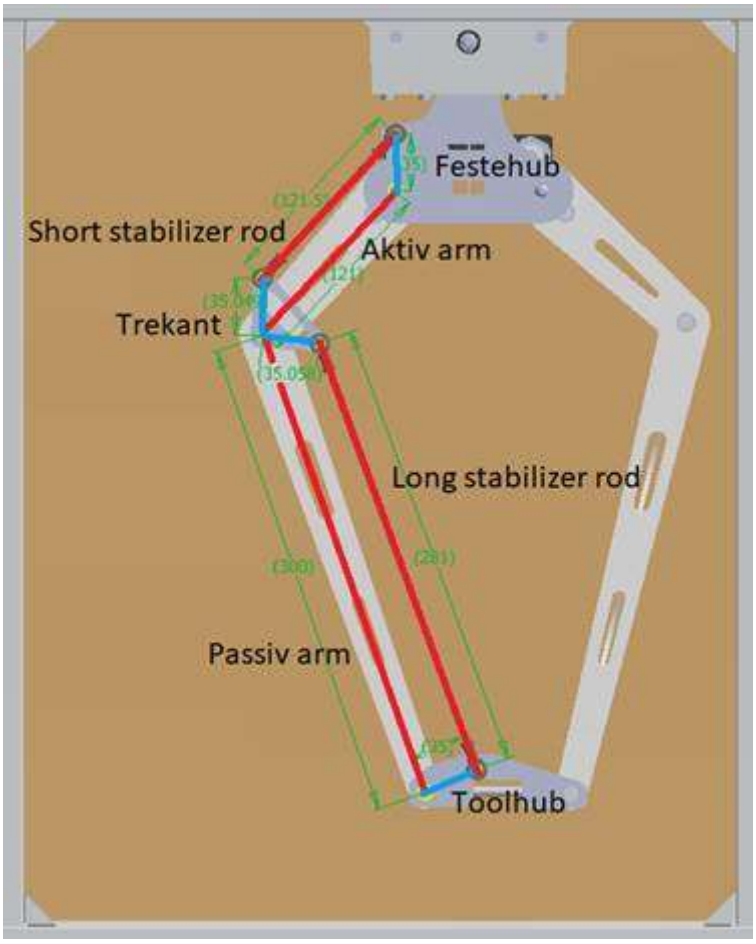


Figure 49 3D-render of the stabilizer on the final product of the last group. Dimensions are annotated in green, with joint centers length in blue and red.

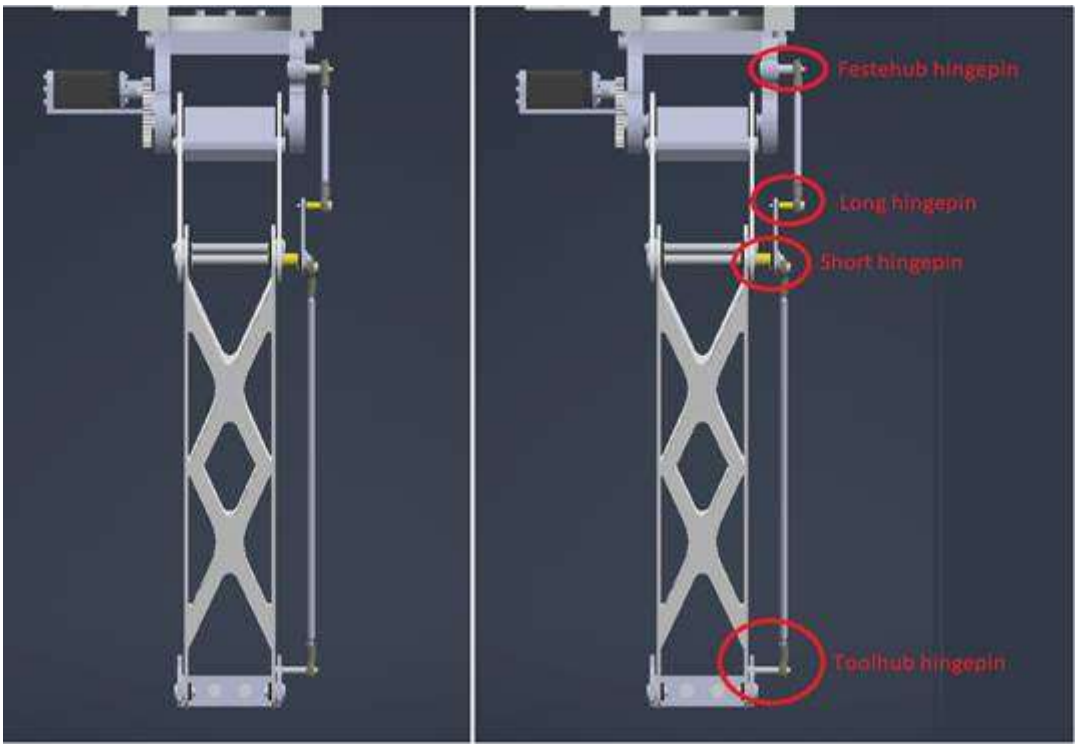


Figure 50 3D-render of side view of last group's robotic arm. Red circles highlighting components.

Arm adjustments and mechanical stopper design:

To accommodate the implementation of the gripper into the design of the arm, adjustments to the length of the arms were needed. This is because the gripper takes more space, while it also had to be able to reach as much surface area as possible to be able to pick up a ball from the interactive game. After some experimental testing by simply moving the arms of the original robot around while measuring the height from the bottom part of the arms, the team decided on a new total arm length of 401mm, reduced from the original 421mm. The initial approach for changing the total length of the arm was to simply extend each link of the arms by how proportionally long they were in relation to the total original arm length. However, after laser-cutting the new arms, it was apparent that the new arm design could not reach the desired gripper-height required to place a ball into the basketball-like hoop.

To address this, the lengths of each link was. This was done by making the top part, also known as the active part of the arm, longer, while making the bottom part, also known as the passive part, shorter. The logic behind this is that the topmost part of the arm can now pull the end of the arm further up because it is longer. This, however, comes with the disadvantage that the arm will need more torque for its movement. To see if the new torque value to use the arm is a noticeable problem, some mathematical work is required, and with the help of Equation 3 the two torque values can be determined. The results show 67.2Ncm for the older version, compared to 85.2Ncm for the newer version. As shown later in this report in section 5.2.1 about Motor torque and RPM requirements, a value of 85.2Ncm is not a cause of concern. Note the difference in the lengths of the arms between the two designs (**Figure 51**).

After spending some time with the robot, it was evident that when the arms were fully extended, the joints of the arm would occasionally over-rotate, which causes the arms to fold inward on themselves. To mitigate this issue, the team decided to incorporate a mechanical stopper into the arm design to limit the joints range of motion. To this into the arm, some minor design adjustments had to be made. Firstly, the design to limit the range of degrees (A) (**Figure 54**) was made into the passiv arm and a hole (B) into aktiv arm to hold the mechanical stopper in place (**Figure 54**). After 3D-printing the mechanical stopper attachment, which is labeled as A in **Figure 53**, the attachment was glued to the surface of the arms to secure it. This simple design limits the joint's freedom of rotation, as shown in **Figure 52**.

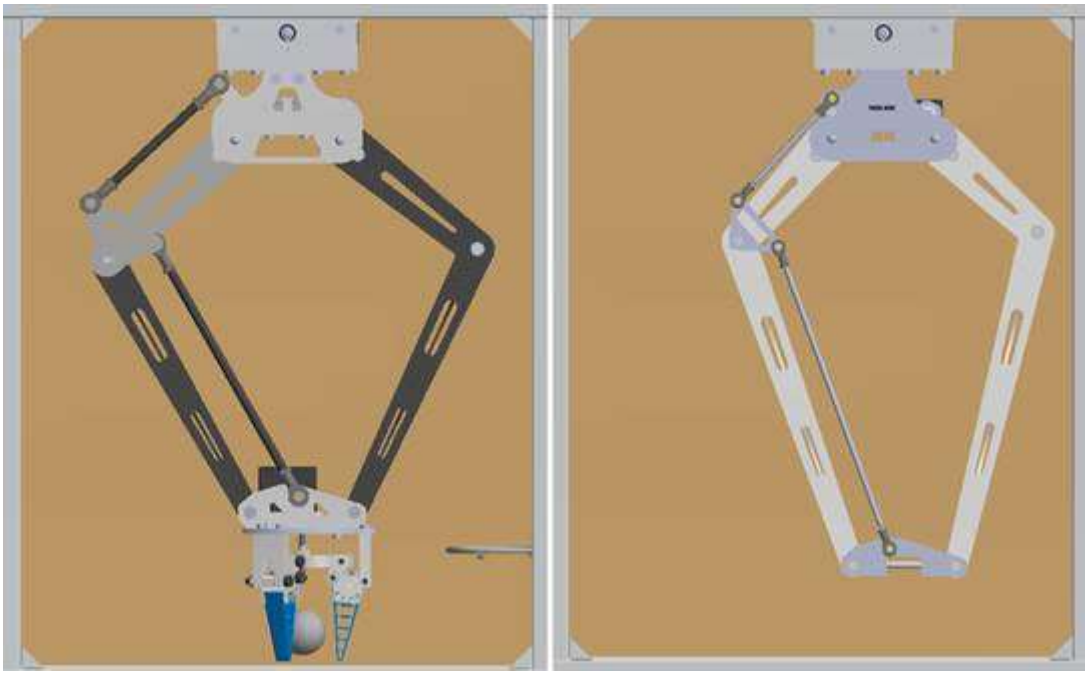


Figure 51 3D render of both versions of the arms. The left view is the final version, while the right view is the initial version

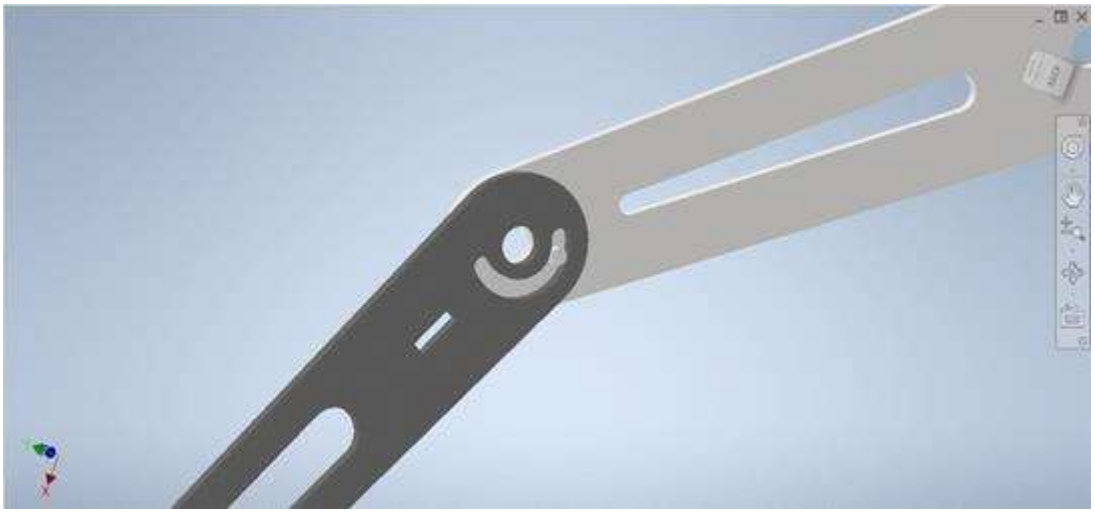


Figure 52 3D render showing the stopper mechanism.

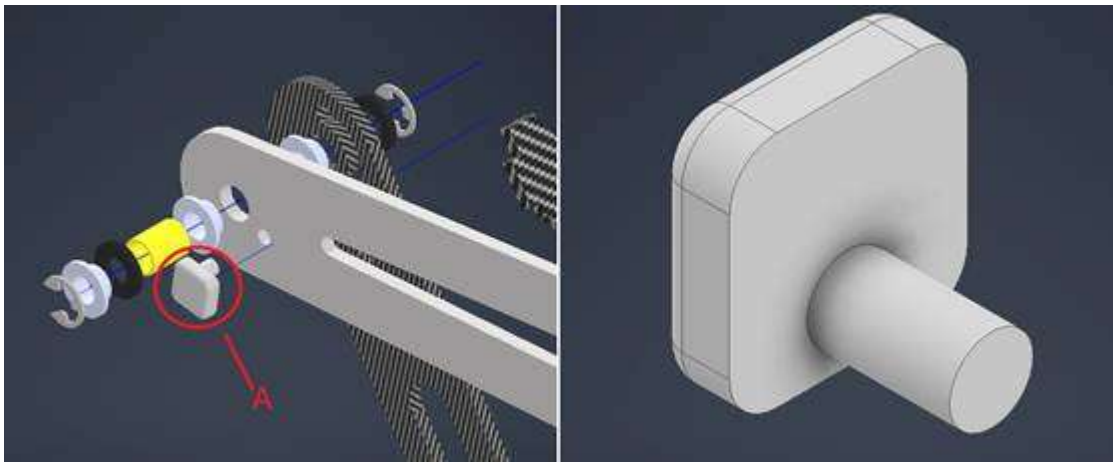


Figure 53 3D render of the mechanical stopper both exploded view (left) and part view (right).

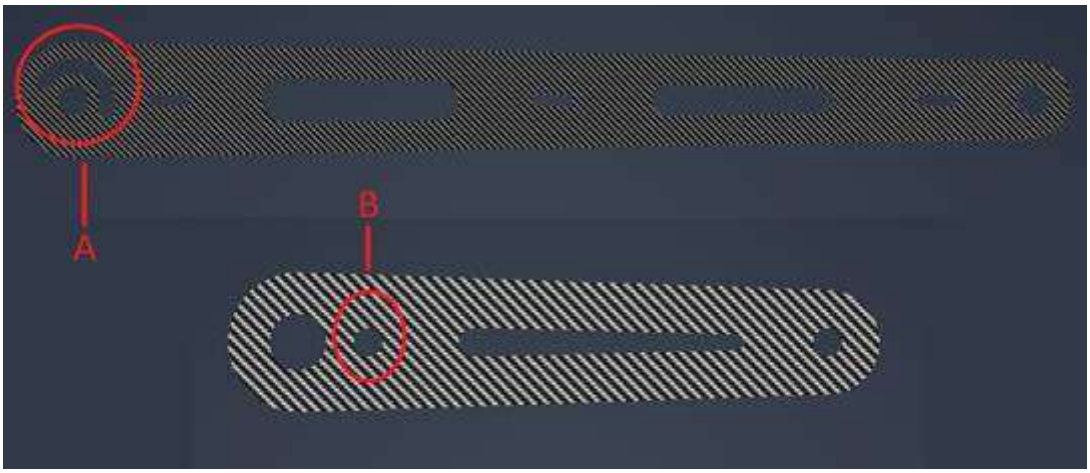


Figure 54 3D-render of the aktiv and passiv arm. Both labeled for clarity in the main text.

Stabilizer design:

As mentioned above, the first step for a working stabilizer is to ensure equal lengths between the stabilizer rods and the arms. By changing the Festehub and Toolhub stabilizer holes to correspond to their respective holes on the triangle (**Figure 55**, Figure **56**), the relation between the stabilizer rods and the arms automatically happens if these lengths are correct (**Figure 57**). This worked, and the stabilizer worked as intended.

Some small adjustments from the original design were changing the rod end bearings from an M3 threading to M5 threading, allowing the use of M5 threaded rods as stabilizer rods. The original bushings were swapped out for larger bushings (**Figure 58**), with an inner diameter of 5 mm which was an excellent fit for the 5 mm diameter hinge pins.

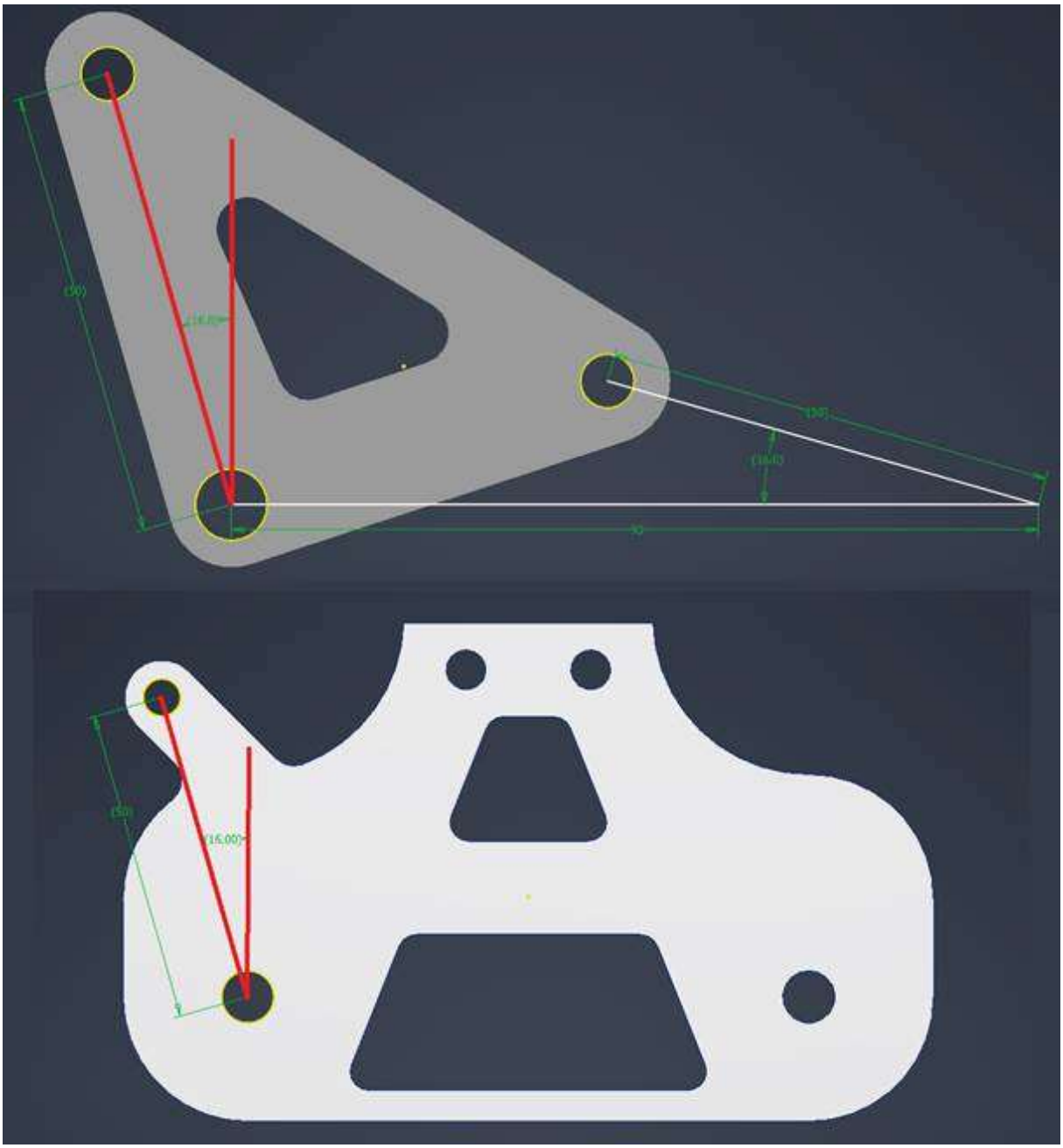


Figure 55 3D render with visible 2D sketch in red, and dimensions in green to illustrate the equal lengths between Festehub and triangle.

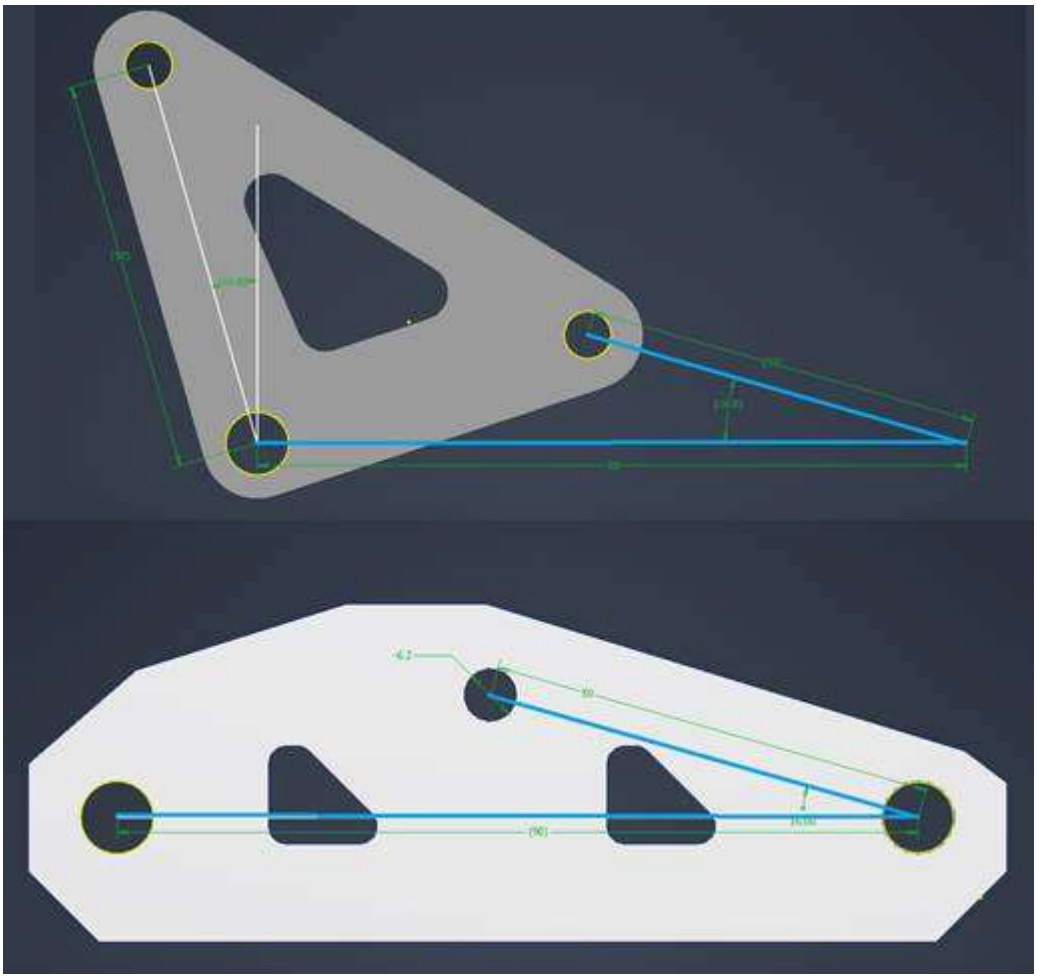


Figure 56 3D render with visible 2D sketch in blue, and dimensions in green, to illustrate the equal lengths between Toolhub and triangle.

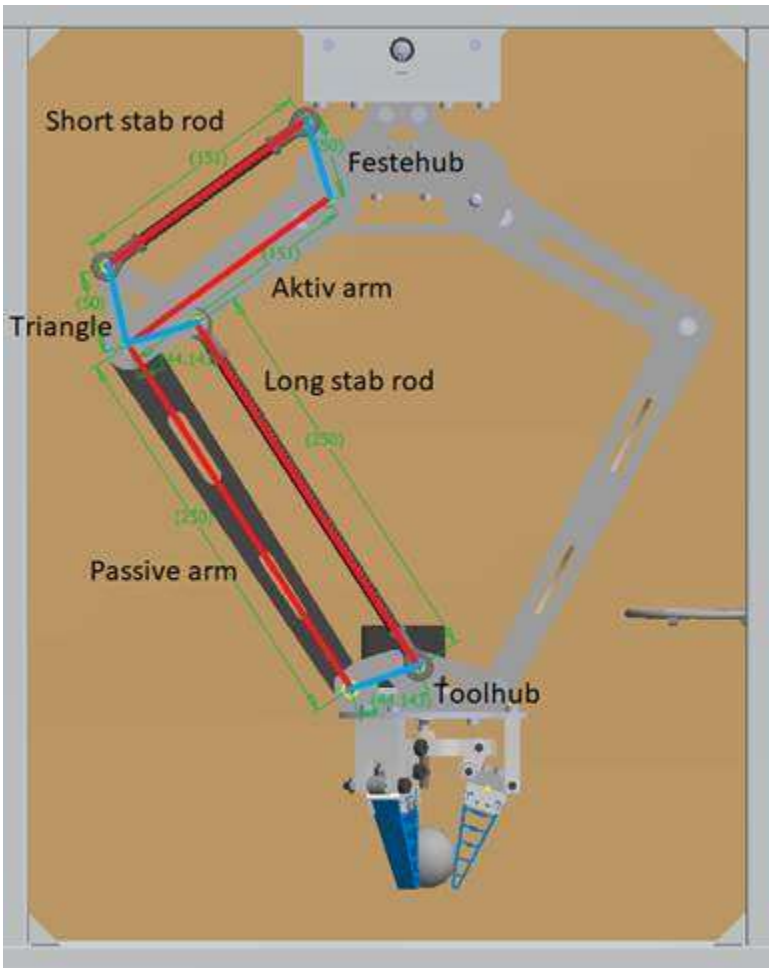


Figure 57 3D-render of the new stabilizer solution in the final product of the robotic arm. In green text is the lengths of the components. Color coding is for easier understanding.

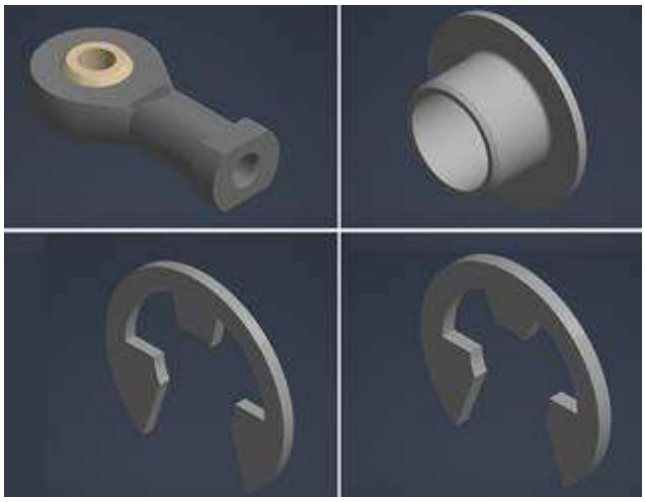


Figure 58 3D render of externally procured parts for general improvements such as fastening and managing friction. Top left: igubal® rod end bearing, top right: flanged bushing, bottom right and bottom left: E-rings.

Triangle design and long and short hingepin design:

One way of improving the stabilizer is to change the size of the triangle (**Figure 59**). As the arms apply a certain amount of force, the triangle must be able to support it. One way of increasing the amount of force the triangle can manage is to increase the distance between the 6 mm holes (B) and the 8 mm hole (A). The logic behind this is seen in Equation [1], torque increases the further away the force is from a rotational motion. The triangle resists rotational motion from the arms, which means by increasing the distance between the 6 mm holes (B) and the 8 mm hole (A) in the triangle. The triangle can now resist more torque received from the rotational motion of the arms.

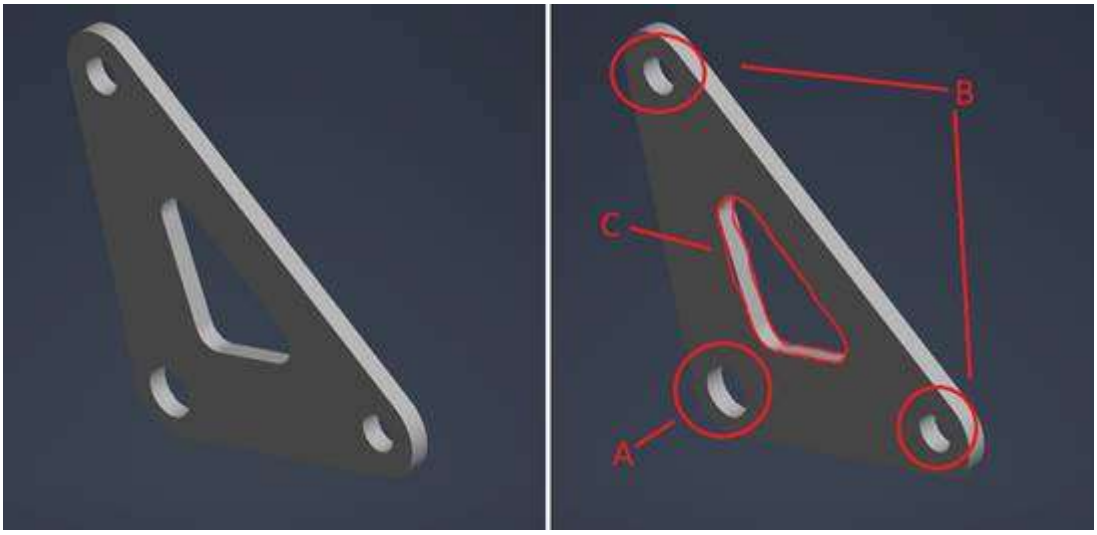


Figure 59 3D render of the stabilizer triangle. On the right, red markings are added for clarity. Refer to the main text for details.

Frontplate design:

The frontplate (**Figure 60**) was a direct copy of the backplate as it was symmetric, with some minor adjustments, such as the stabilizer hole (A). Other than that, it had the same cutouts (C) and holes for ball bearings (B) as in the original design from the previous group.

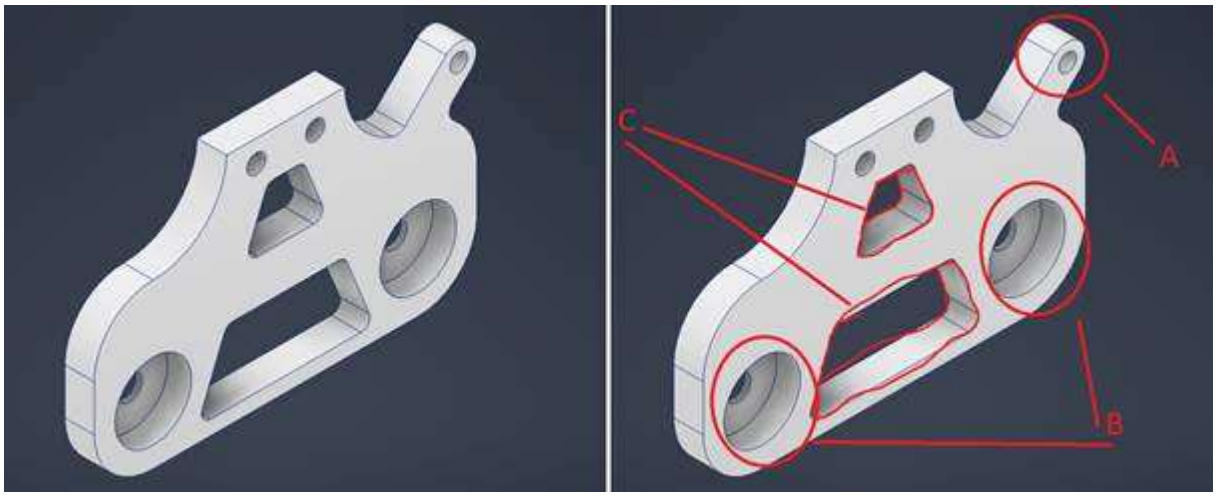


Figure 60 3D render of the frontplate. On the right, red markings are added for clarity. Refer to the main text for details

Toolhubfestning design:

Toolhubfestning (**Figure 61**) was mainly affected by the placement of the stabilizer hole (A). The holes (C) which connect the arm to the Toolhub were not changed from the original design as they remain 8 mm diameter with a H7 tolerance. The only other change was to remove some material from the cutouts (B).

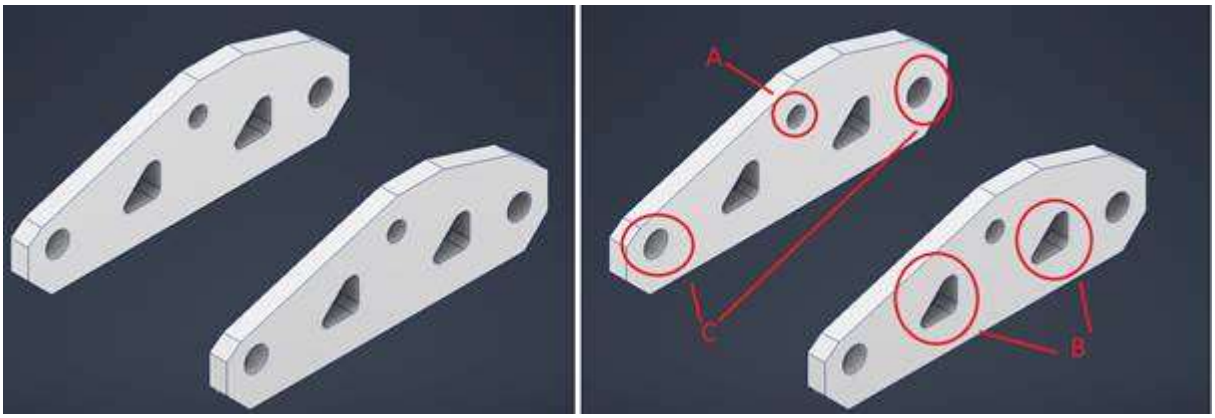


Figure 61 3D render of the toolhubfestning. On the right, red markings are added for clarity. Refer to the main text for details.

Hingepin design:

The hingepins (**Figure 62**) only underwent dimensional changes, as their original design functioned as intended and required no redesign. In this stabilizer version, the standard hingepins were produced with a diameter of 5 mm, while those used in the thicker sections of the Festehub and Toolhub were made with a diameter of 6 mm. All hingepins featured an indentation to accommodate E-rings, which holds the parts in place. For the 5 mm sections, the indent diameter was 4 mm with a width of 0.7 mm, while for the 6 mm sections, the indent was 5 mm in diameter, with the same width. All hingepins were changed in length to shorten the length from stabilizer to arm as much as possible.

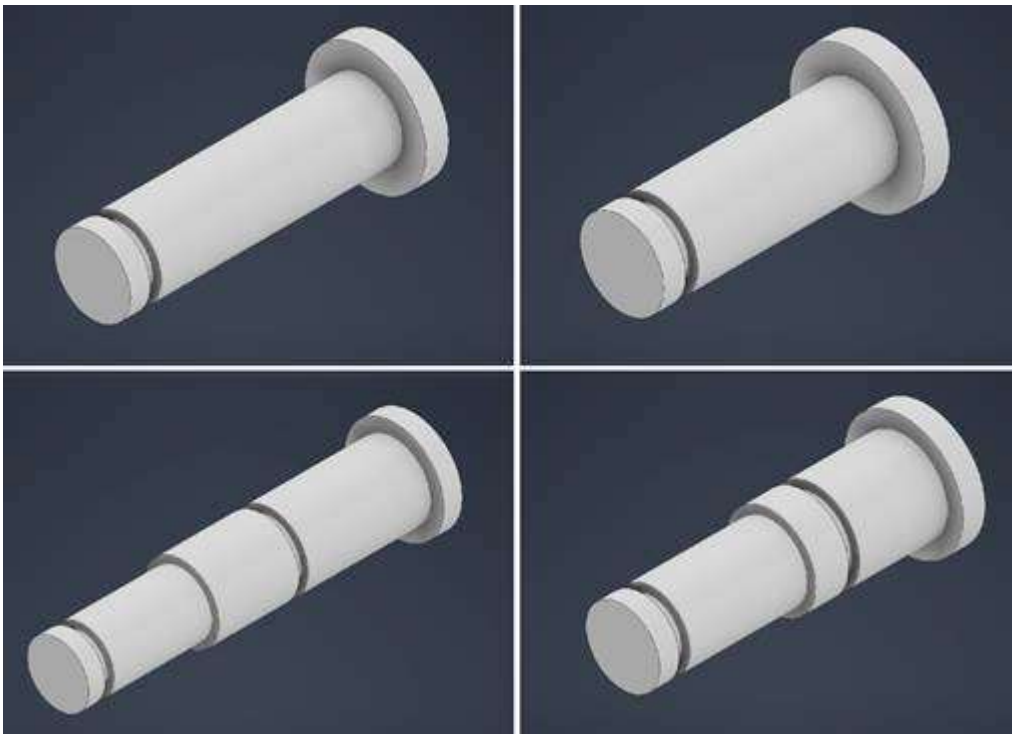


Figure 62 3D render of all hingepins. Top left: long hingepin, top right: small hingeping, bottom left: Festehub hingepin and bottom right: Toolhub hingepin.

Result from stabilizer to arm length:

The length changes of the hingepins gave good length changes, specifically for the Toolhub (**Figure 63**). The triangle length (**Figure 64**) got increased because the triangle had to be bit further out to be able to get the long stabilizer rod at the back side of the triangle, as shown in **Figure 66**. Another small change in length for the Festehub (**Figure 65**), however, it's visually visible. This worked as intended as its clear length change of the old stabilizer design and the new, as shown in **Figure 66**.

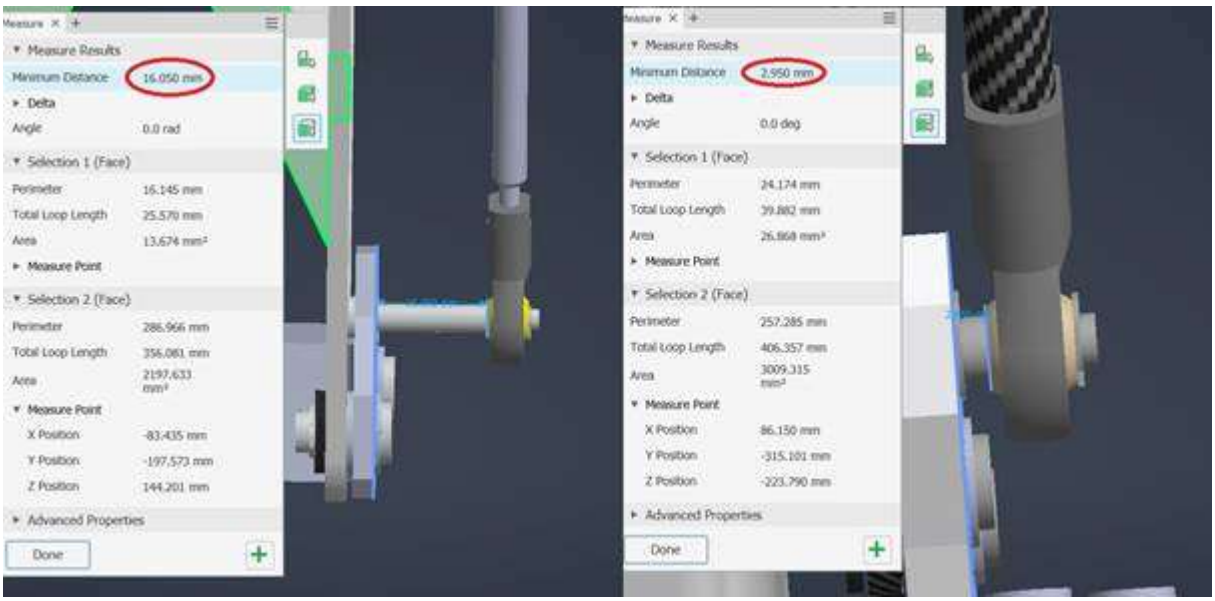


Figure 63 Toolhub length change. Old length to the left and new length to the right.

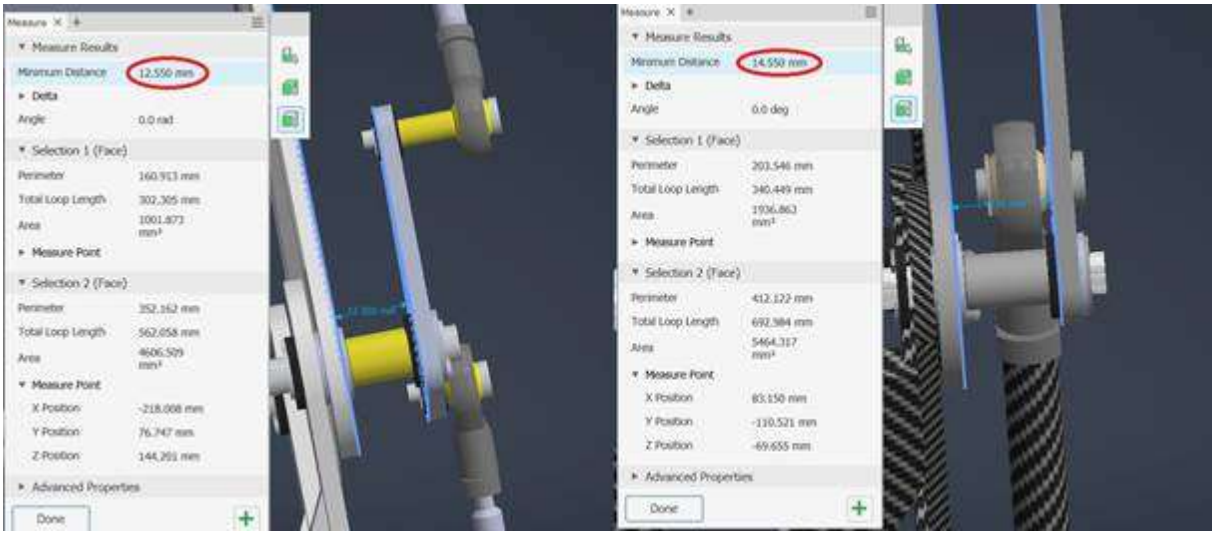


Figure 64 Triangle length change. Old length to the left and new length to the right.

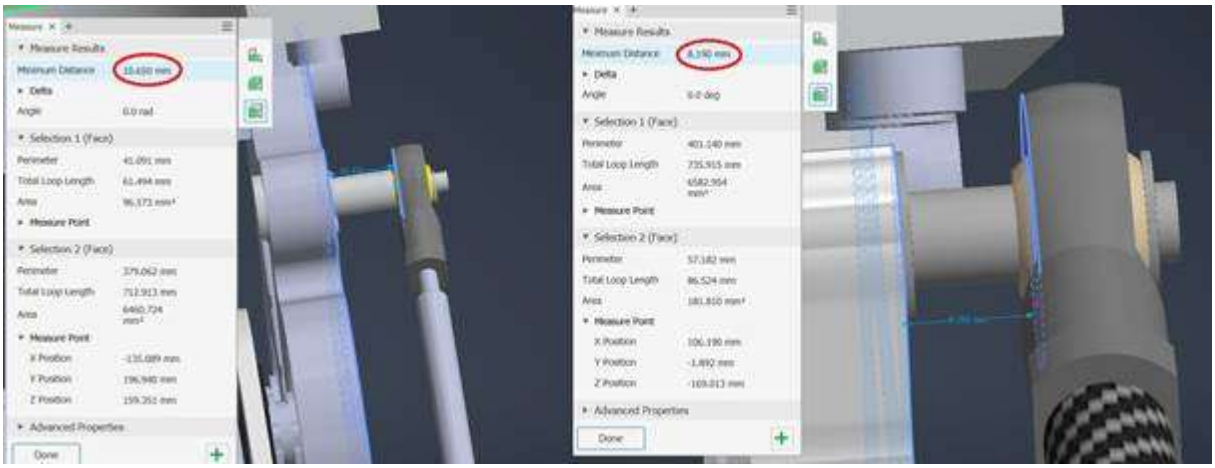


Figure 65 Festehub length change. Old length to the left and new length to the right.

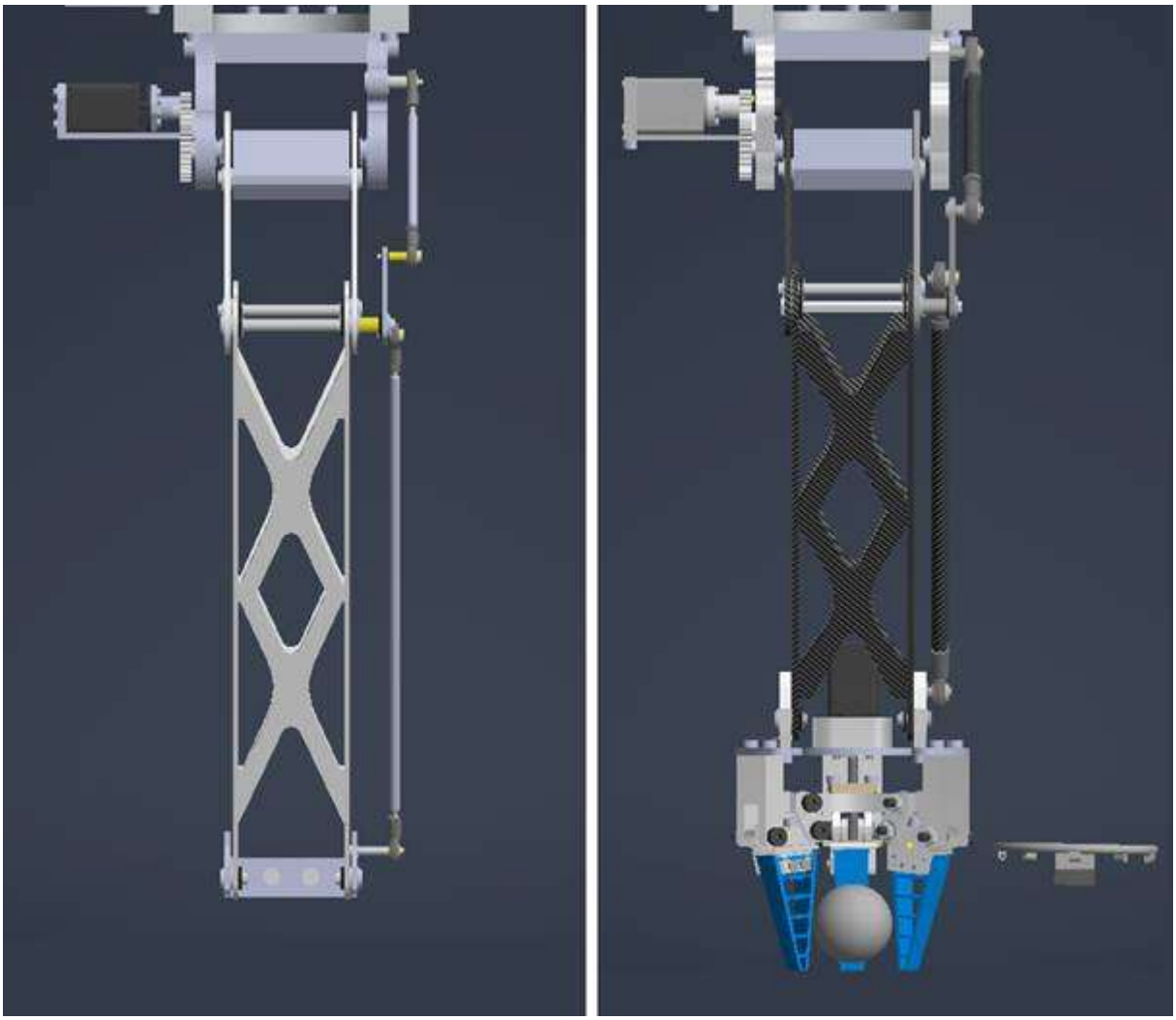


Figure 66 3D render of the arm from the side. Old to the left and new to the right.

5.2 Robotic Gripper Designs and Changes

One of Tronrud Engineering's (TE) requirements for this year's project was to develop an interactive pick-and-place game using the robotic arm. To achieve this, a gripper was needed, both to enable the game functionality and to fulfill one of the project's key objectives, which was to develop a gripper. TE also requested the gripper to be visually appealing to attract attention during demonstrations.

The supervisor from TE provided two recommendations for gripper development:

1. Design the gripper around the chosen game.
2. Search YouTube for state of the art and inspiration,

The chosen game concept was basketball, which required a gripper capable of reliably picking up and placing a small ball.

A wide range of gripper types were evaluated, including vacuum-based and soft robotic grippers. However, these designs typically require either a vacuum pump or a

compressor, which would cause the robotic arm to exceed the 12kg weight limit. A mechanical gripper was therefore chosen for the current application, offering both reduced weight and complying with the aesthetic requirement.

During the research phase, adaptive gripper designs such as those developed by Festo were also reviewed. Some of those designs used Festo's adaptive finger based off on the fin ray effect, and as mentioned in the theory section, make the finger bend around the applied pressure. The final concept combined elements from Festo's adaptive finger with a three-fingered mechanical gripper, presenting a gripper that can handle objects of varying shapes and sizes effectively.

5.2.1 Gripper design 1.0

Gripper design 1.0 was primarily created to evaluate the concept using Autodesk Inventor 2025, and to observe the movement of the individual joints and overall gripper mechanism, as shown in **Figure 67**. This early version was also meant to be presented to the TE supervisor for evaluation and feedback, which guided further development. The parts were also 3D-printed to manually check the grippers movement, as real world functionality does not always reflect simulations in Autodesk Inventor 2025.

The first design-adjustment included increasing the size of the holes from M3 to M4, as TE advised against using bolts with a diameter smaller than M4. In addition, the thickness of several components were increased, as most were too thin and therefore likely to fail. The minimum allowed thickness from TE was 4 mm if the part was going to be made from aluminum. To clarify the sequence of events, it is important to note that the feedback from TE was provided after presenting Gripper design 1.0.

It is important to note that tolerances were not considered during the 3D modelling phase, as the purpose was limited to prototyping. Tolerances were only to be applied once the parts were production-ready and transferred to 2D technical drawings. For this reason, no tolerances were included in the earlier designs.

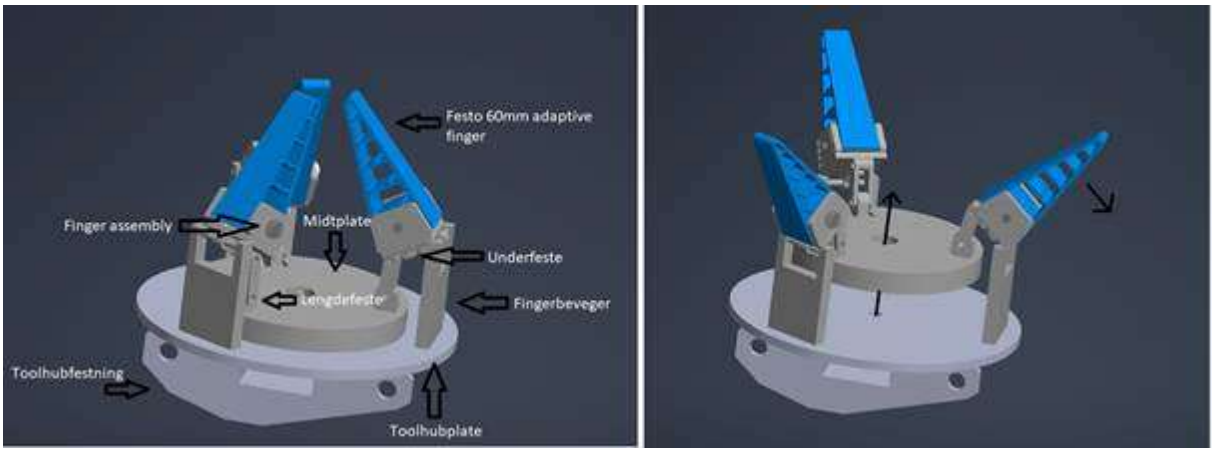


Figure 67 Assembly view model of the Gripper design 1.0, annotated with part names to the left. Showing the opening mechanism, with black arrows showing direction of movement in the rightmost view.

Underfeste design 1.0:

Underfeste (**Figure 68**) was designed to connect the finger assembly to the rest of the gripper. The M3 center holes (A) were used to mount the finger assembly, while the M3 axle holes (B) connected it to the rest of the gripper. While the part functioned as intended, it was considered too thin for reliable use. This was designed before receiving feedback from TE, who later advised against using bolts smaller than M4. This feedback was consistent with all the components in Gripper design 1.0.

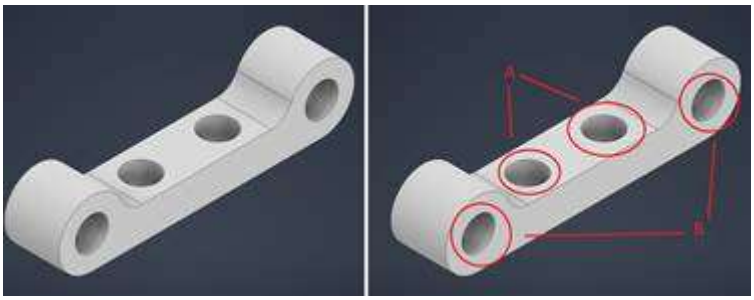


Figure 68 3D render of Underfeste design 1.0, with holes labeled for clarity for the main text.

Lengdefeste design 1.0:

Lengdefeste (**Figure 69**) was designed to make the connection between the finger assembly and the midtplate. One early idea was to directly connect the *finger assembly* to the *midtplate*, without the use of the lengdefeste. However, due to size and geometry of the finger components, a direct connection could potentially lead to collisions during movement. By designing this part, the assembly was given space to operate without interference. The part was too thin for reliable use, as it was only 3mm thick and designed for M3 bolts. These issues were addressed in the updated *Gripper design 2.0*.

Figure 69 3D render of Lengdefeste design 1.0.

Midtplate design 1.0:

The main function of the 88mm diameter midtplate was to control the opening and closing mechanism of the gripper. When the lead screw rotates, the midtplate moves vertically, either up or down, depending on the direction of the screw. Since the plate is fixed at three mounting points, as shown in **Figure 67**, it remains rotationally constrained, allowing only linear movement. This vertical motion drives the grippers open and close functionality.

For testing purposes a 12mm diameter, 6mm pitch trapezoidal lead screw was designed, as shown in **Figure 70**, along with a matching 12mm threaded hole in the midtplate. The gripper was operated manually during early tests, which was the

reason for designing the lead screw. At the time, the plan for TE to manufacture these parts using a CNC milling machine. However, producing the three extruded features on the top side of the plate, as shown in **Figure 70**, this would have required removing a fair bit of material, resulting in unnecessary material waste. For this reason, the part was completely redesigned in Gripper design 2.0.

Figure 70 3D render of Midtplate design 1.0 to the left, and the lead screw to the right.

Fingerbeveger design 1.0:

The primary function of the *fingerbeveger* (**Figure 71**) is to raise the mounting point of the finger assembly in the gripper. When the finger assembly is attached to this fixed component, it enables the fingers to open and close in response to the midtplate's vertical motion. This design remained unchanged between Gripper design 1.0 and Gripper design 2.0, with the only modification being an increase in axle hole diameter (A) from 3mm to 4mm.

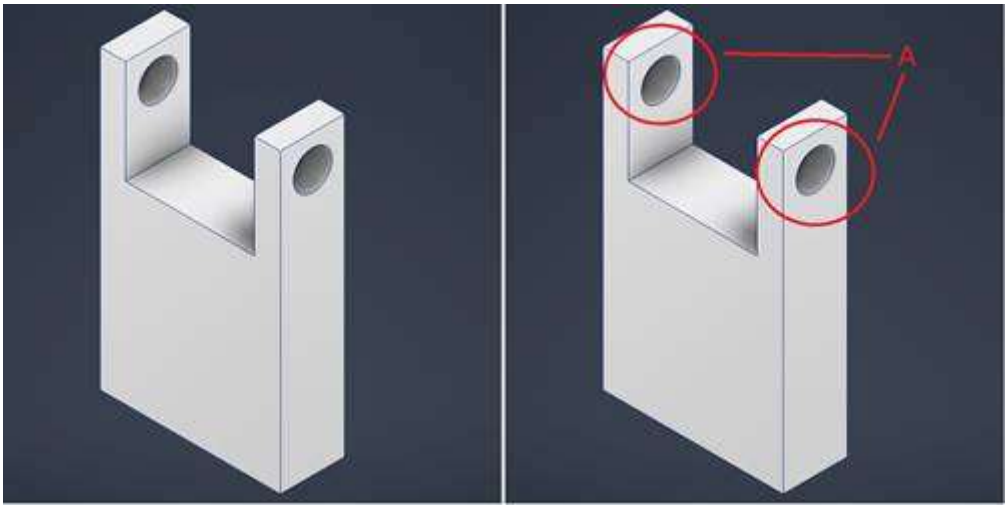


Figure 71 3D render of Fingerbeveger design 1.0, with axle holes labeled for clarity.

Finger assembly design 1.0:

The finger assembly consists of three parts: the *bakfingerfeste* the *frontfingerfeste*, as shown in **Figure 72**, and the *adaptive finger* from Festo, as shown in **Figure 73**. The frontfingerfeste includes two extruded rods (A) that pass through the bottom center section of the adaptive finger (D) and into the corresponding holes (B) in the bakfingerfeste. An M4 bolt is inserted through the mounting hole (C) to fasten the entire assembly, as shown in **Figure 67**.

While the design was functional, the frontfingerfeste would have been difficult to manufacture, and CNC milling the bakfingerfeste would have resulted in significant material waste. For this reason, the assembly was redesigned in Gripper design 2.0. Although the 1.0 version was 3D-printed for testing, the parts broke due to being to thin for 3D-printing.

The adaptive finger from Festo was purchased and performed as intended. When pressure is applied, the finger bends around the contact point, in line with the fin ray effect, as shown in **Figure 74**. However, it was observed that the closer the pressure point is to the fingertip, the less the finger bends, which may reduce its ability to grip in certain cases.

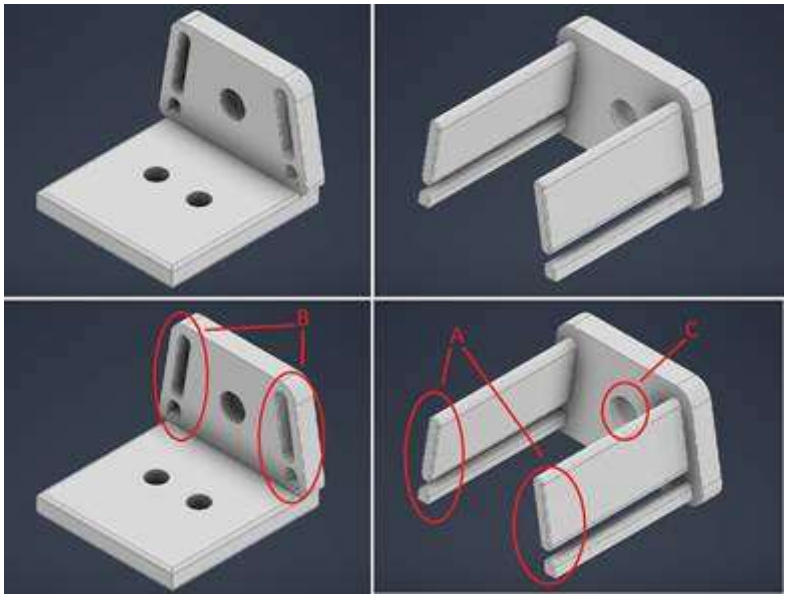


Figure 72 3D render of the parts needed for the finger assembly. To the left is bakfingerfeste and to the right is frontfingerfeste. The labeled elements are described in the main text.

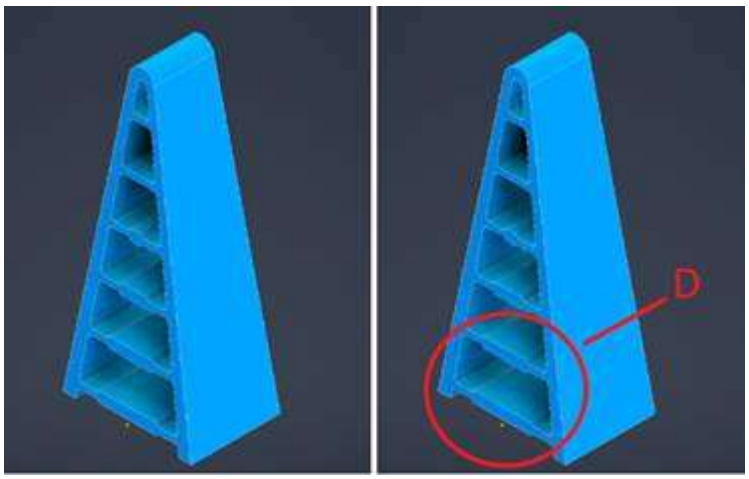


Figure 73 3D render of a Festo 60mm adaptive finger needed for the finger assembly. The labeled element is described in the main text.

Figure 74 Picture shows Festo's own gripper with 120mm adaptive fingers. When pressure is applied with the index finger, the adaptive finger bends around the pressure point.

3D-printed axle:

Since most of the holes in the Gripper design 1.0 components were sized at 3mm, it was difficult to find axles in that diameter of the correct length. As a temporary solution for testing, custom axles (**Figure 75**) were 3D-printed from PLA, shaped into simple cylinders with a 3mm diameter, in various lengths. Some of these were later reused in Gripper design 2.0, alongside standard M4 bolts.

Figure 75 3D render of the printed rods.

Toolhub design 1.0:

The *toolhub*, which is referenced frequently in the mechanical design and development chapter, serves as a connection between the mechanical gripper and robotic arm. It consists of three components: the *toolhubplate* and two *toolhubfestninger*, as shown in **Figure 76**. This design is used for both Gripper assembly 1.0 and Gripper assembly 2.0, which kept the toolhubfestning from the previous group, mounted onto the 130mm diameter toolhubplate.

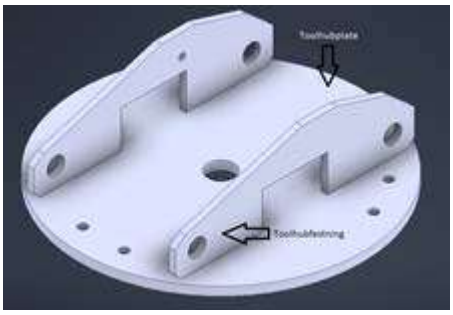


Figure 76 3D render of the first and original toolhub design, labeled with part name.

5.2.2 Gripper design 2.0

For the gripper design 2.0, some parts were altered more than others. As the concept from Gripper design 1.0 functioned as intended, the newer design was intended as a refinement of the same concept. The most common changes, as was mentioned in Gripper design 1.0, were the thickening of some parts as well as changing all the M3 holes to M4 or higher to accommodate the larger bolts. Only the parts significant changes are discussed below.

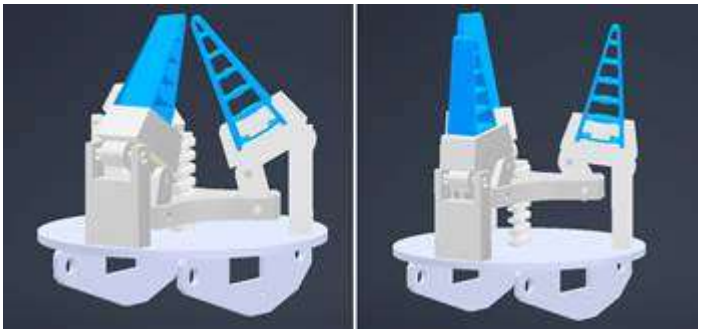


Figure 77 3D renders that show the gripper 2.0 assembly. It is closed in the left view, and open in the right.

Finger assembly design 2.0:

The finger assembly had a significant transformation in Gripper design 2.0. The previous version was both difficult to manufacture and structurally weak, so the decision was made to redesign the assembly entirely and 3D-print the new component. In the updated design, the adaptive finger slides into a newly developed *fingerfeste* as its design is shaped to match the profile of the finger (B). Where it is secured by a 5mm thick extrusion (A), as shown in **Figure 77**.

As the force is applied to the side of the fingers (**Figure 74**), when items get picked up, the risk of fingers slipping out from the top or the side of the fingerfeste is minimal. However, due to the flexible material of the finger, the finger itself occasionally slipped out. This extruded piece (A) in **Figure 78** was specifically added to prevent this and ensure the finger remains in place. This concept worked as intended during testing, however, the 5mm extruded piece (A) broke easily due to low infill on the print. As mentioned in the Methods section, this issue was resolved by increasing the infill percentage, which improved the strength of the component.

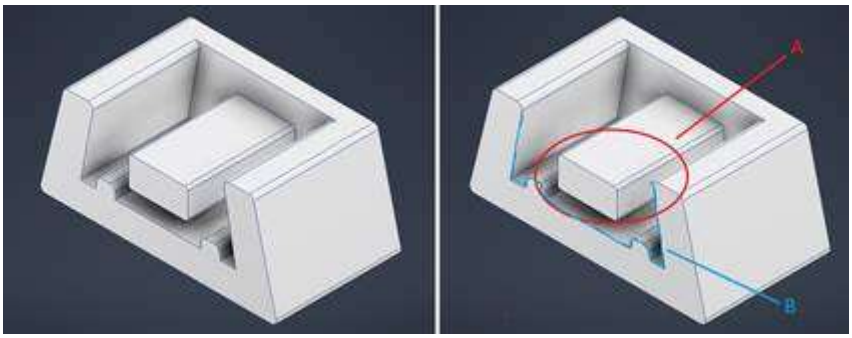


Figure 78 3D render of fingerfeste. On the right, red and blue markings are added for clarity in the text.

Midtplate design 2.0:

The midtplate had a major redesign compared to its previous version. In this iteration, the connection points were integrated into the plate itself, rather than relying on extruded features at the top. Additionally, a new fidget-spinner-style design was implemented to reduce overall weight, as shown in **Figure 79**.

The central 12mm threaded hole was kept allowing for manual testing using the same lead screw. In addition, new 4mm diameter holes were added on each side to fit an axle. The original plan was to create custom axles specifically for the gripper, but this would increase production time if they were to be machined. Alternatively, 3D-printing the axles would have resulted in lack of strength. Since the axles were initially intended to be press-fitted, this approach was not suitable for printed parts due to tolerance and overall strength issues. To resolve this, all parts requiring an axle were modified for *Gripper design 3.0* to fit a shoulder screw, like the one used in *Festehub Assembly 2.0*. These screws were already in TE's inventory, removing the need for manufacturing and improving overall robustness.

Figure 79 3D render of midtplate design 2.0.

5.2.3 Gripper design 3.0

As mentioned in the chapter Mechanical Design and Development, Hønefoss videregående skole would produce some of the final iterations of the project. As Gripper design 3.0 is the final iteration of this assembly, most of the parts were selected for production by the school. The school offered to manufacture several of the simpler components. This design focused on final adjustments, including changing hole dimensions to fit bushings, shoulder screws, and bolts. Another small adjustment was the geometric change to prevent collisions between moving parts. As

shown in **Figure 80**, all the previous axles were swapped out with the shoulder screws shown in **Figure 81**.

In the final design, the motor was ready for use, and the previously used 3D-printed lead screw was replaced with a standard M6 metric screw. In the original design, the lead screw was meant to be a 5 mm diameter high helix screw from Igus. However, due to time constraints and the relatively high cost of the component, it was replaced with a 50 mm long M6 screw, which was viable for testing and for final presentation.

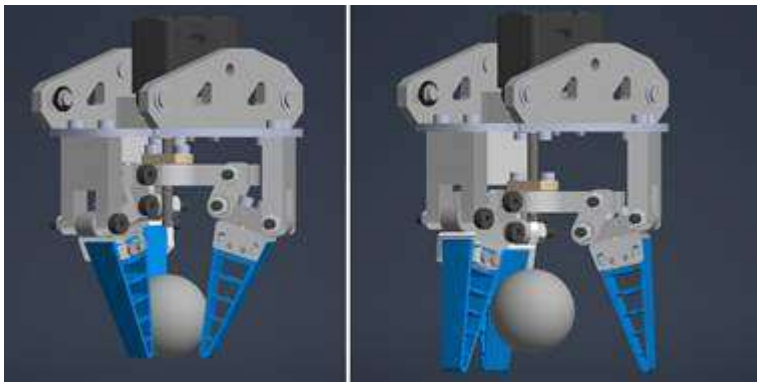


Figure 80 Picture which shows assembly view of the gripper design 3.0 closed to the left, and open to the right with a ball to simulate the game.

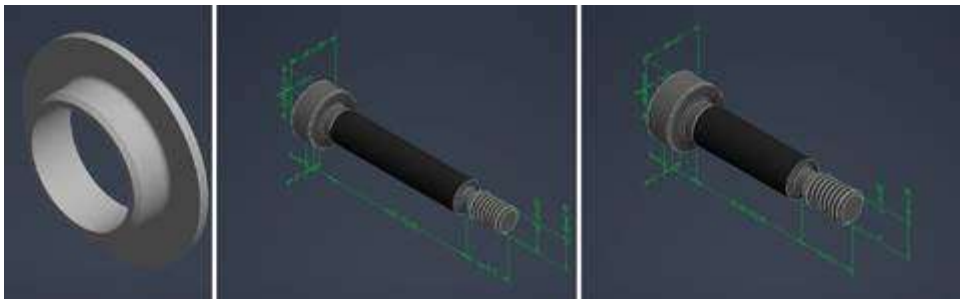


Figure 81 3D renders that shows, from left to right: the flanged bushing used in the whole gripper, a 30mm long shoulder screw, and a 20mm long shoulder screw

Fingerbeveger design 3.0:

In this final version of the fingerbeveger (**Figure 82**), only minor hole and geometric adjustments were made to make the component ready for production. The axle hole (A) was adjusted to 6mm in diameter with a H7 tolerance, providing a close fit for the 30 mm long shoulder screw with f9 fit tolerance. The bottom 4 mm holes which were designed for M4 bolts, as shown in **Figure 80**, remained unchanged.

During testing of Gripper design 2.0, the fingerbeveger collided with the underfeste during motion. To resolve this, an additional 5 mm of material was removed downwards from the center section (B), creating clearance when the gripper closed. Another collision issue was observed between the fingerfeste and the corners of the fingerbeveger, which was fixed by applying a 4 mm radius to the corners. This allowed the parts to move without colliding. Although this corner adjustment would require an additional setup step if the part was to be CNC machined, it was a

necessary modification, as the gripper would otherwise experience difficulties when moving.

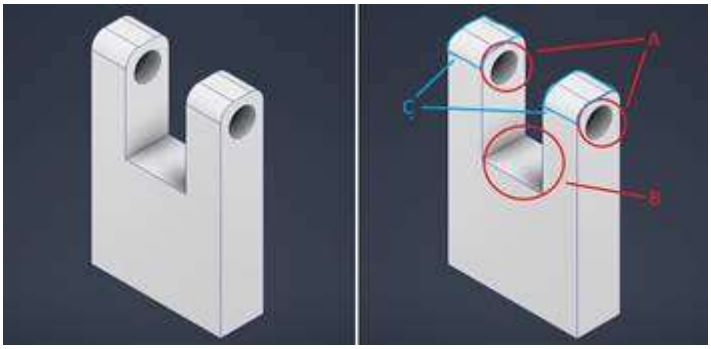


Figure 82 3D render of fingerbeveger design 3.0. On the right, red and blue markings are added for clarity. Refer to the main text for details.

Underfeste design 3.0:

Several adjustments were made to the underfeste for the final design. As shown in **Figure 83**, the two axle holes, marked as (A) and (B), were designed in different dimensions. This was intentional, as the only rotational movement for this part occurs around the shoulder screw connected to the fingerbeveger (A), highlighted with a red box and arrow shown in **Figure 84** to show the direction of rotation.

Hole (A) was made with a diameter of 7 mm and H7 fit tolerance, ensuring that the flanged bushings with an outer diameter of 7 mm and an inner diameter of 6 mm, shown in **Figure 81**, would fit. These bushings were planned to be press-fitted into the parts. This hole (A) together with a bushing on each side would ensure little to no friction when rotating around the shoulder screw. Hole (B) was made with a 6mm diameter and H7 tolerance, ensuring close fit around the shoulder screw. To prevent the underfeste from rotating around the point (B), a threaded M4 hole (D) was added for a set screw. The set screw would serve to secure the shoulder screw in place. Additionally, the holes at the top of the part (C), were threaded for M4 screws, allowing secure connection with the fingerfeste.

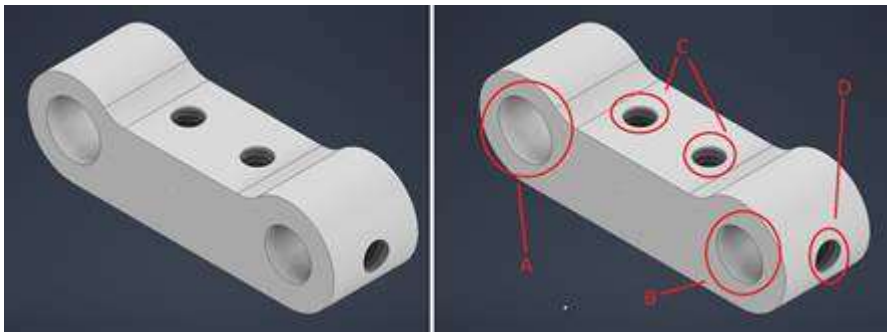


Figure 83 3D-render of underfeste design 3.0. On the right, red markings are added for clarity. Refer to the main text for details.

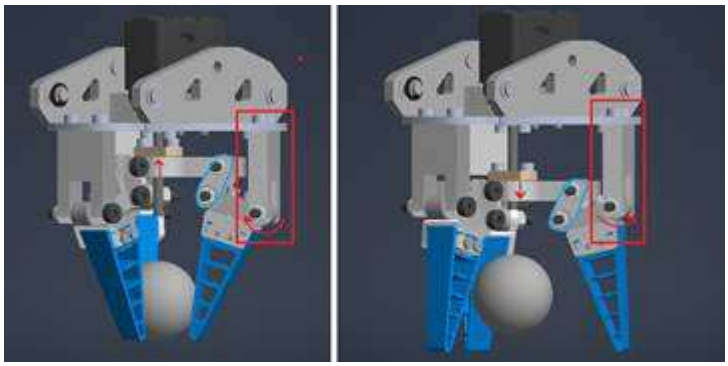


Figure 84 3D render assembly view of the gripper design 3.0 opening and closing. Both have red and blue markings for clarity. Refer to the main text for details.

Lengdefeste design 3.0:

The lengdefeste remained largely unchanged throughout the gripper design process, with only minor dimensional adjustments. One change was made to the holes (A), which were updated to 7 mm diameter with H7 tolerance, matching the larger hole of the underfeste. Since the lengdefeste rotates around both joints, flanged bushings were required to ensure smooth, low friction-rotation. The component, marked with a blue outline in **Figure 84**, rotates at both connection points visible in the figure. As mentioned earlier, the thickness (B) was updated to 4mm, and with the updated hole dimension, the heigh (C) was increased to 12mm to improve strength.

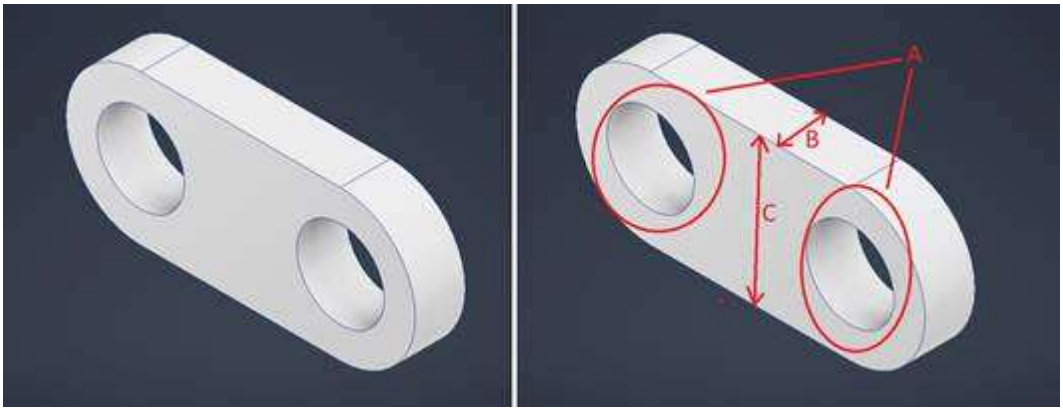


Figure 85 3D render of lengdefeste design 3.0. On the right, red markings are added for clarity. Refer to the main text for details.

Toolhub design 3.0:

In toolhub design 3.0, several adjustments were made, and two adjustments were introduced, as shown in **Figure 86**. One modification involved the toolhubplate, where the original circular shape was replaced with a more geometric design (C), as shown in **Figure 87**. This change was made to reduce the overall part weight by adding cutouts in the design. Additional updates included the 2 mm diameter holes (B) for mounting the newly introduced *brikke* (**Figure 86**) and the motor. Finally, the center hole (A) was resized to 26mm in diameter to fit the new motor adapter component.

The brikke was added to increase the height of the motor to ensure the gripper could fully close. Without this spacer, the motor adapter extended too far downward, causing interference with the midtplate as it moved. The brikke is a spacer block made from PLA, designed with the same 26 mm outer diameter and 2 mm mounting holes as the toolhubplate, allowing it to be assembled directly above the plate.The motor adapter used in the toolhub was based on a similar design to the one used in the *festehub*. This version featured an extruded section (A), as shown in **Figure 88**, with the outer diameter of 12 mm, along with a 6 mm threaded hole (B) used to fasten the M6 screw. The adapter's function is to connect directly to the motor, with the screw inserted and secured into the central threaded hole. The toolhubfesting is discussed in the stabilizer chapter as its more relevant there.

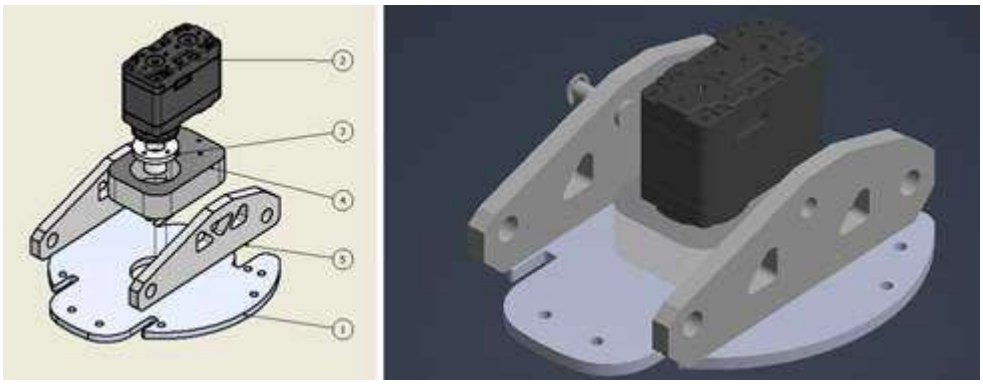


Figure 86 Left picture is the 2d drawing which show the parts numberized, 1 – toolhubplate, 2 – motor, 3 – motor adapter, 4 – brikke and 5 – toolhubfestning. The right picture is the assembly view of the model.

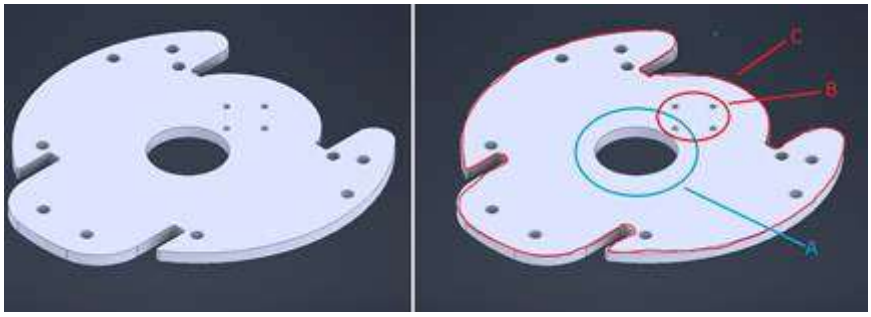


Figure 87 3D render of toolhubplate design 3.0. On the right, red and blue markings are added for clarity. Refer to the main text for details

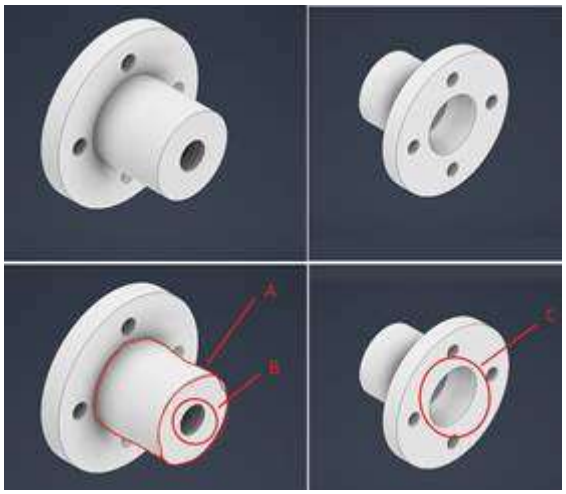


Figure 88 3D render of motor adapter design. On the bottom, red markings are added for clarity. Refer to the main text for details.

Finger assembly design 3.0:

The finger assembly design incorporated additional components ordered from Festo. Beside the adaptive fingers themselves, such as the frontfingerfeste. This decision was made after repeated failures of the fingerfeste used in Finger assembly design 2.0 during testing. Since producing these parts was not viable, the best solution was to use components designed to fit the existing adaptive fingers. To support this setup, the bakfingerfeste was designed to match the frontfingerfeste from Festo, as shown in **Figure 90**. Certain sections of the bakfingerfeste, marked B and C, were directly based on Festo's own bakfingerfeste-equivalent design to ensure compatibility.

As the frontfingerfeste had not yet been ordered and the bakfingerfeste had not been produced, both parts were temporarily 3D-printed. Despite the limitations of the PLA material, the parts performed as intended during testing and did not fail. These printed versions served as functional prototypes until the ordered part could be delivered. This assembly, shown in **Figure 89**, is mounted directly to the underfeste using M4 threaded holes (A) as part of the Gripper Assembly 3.0.

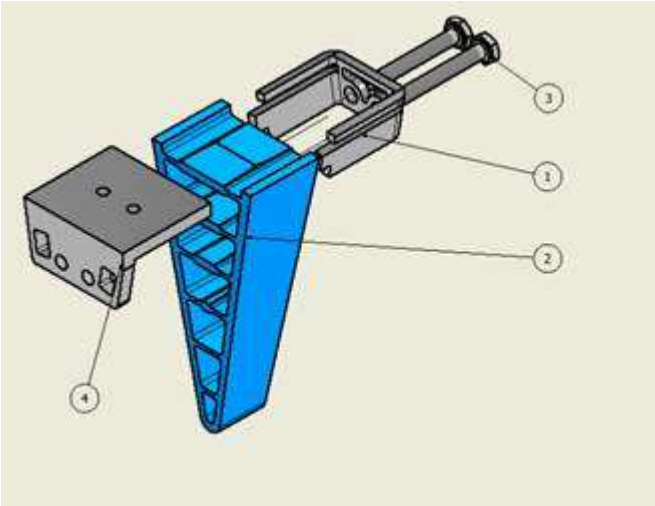


Figure 89 Technical drawing of the exploded view of finger assembly design 3.0 to show how the parts are assembled. 1 – frontfingerfeste, 2 – adaptive finger, 3 – M3 screws, 4 – bakfingerfeste.

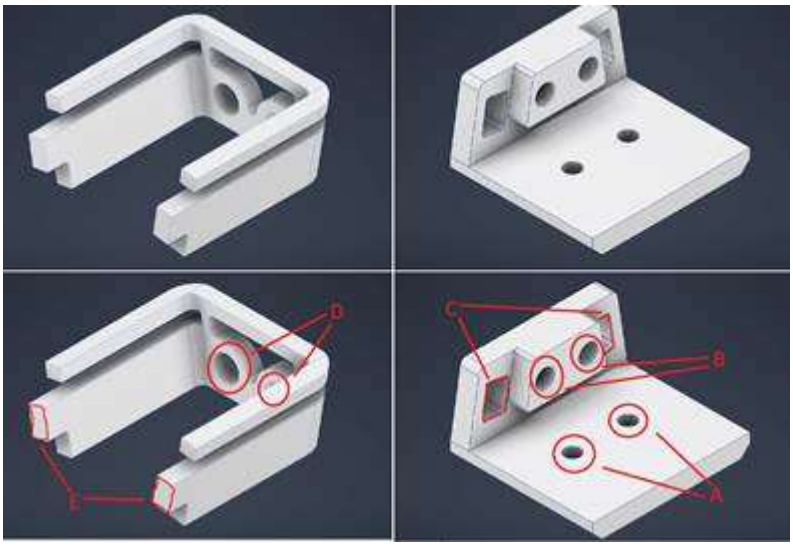


Figure 90 3D-render of the new frontfingerfeste (left) and new bakfingerfeste (right). The frontfingerfeste is Festo's own design and is ordered. On the bottom, red markings are added for clarity. Refer to the main text for details.

5.3 Motor torque and RPM requirements

It is important to get motors with the right requirements when buying motors. If the motors are too weak, the robot could end up not moving as fluidly as it should be. The project might also cost more if it ends up spending more money on motors that could do the same job as cheaper motors.

To get the robot moving, appropriate motors are needed. Motors without enough torque cannot create the desired movement for the mechanical arms and therefore, some calculations and testing is needed to get the desired output. To know how much torque and RPM is needed, an evaluation of the different parts of the robotic arms are needed.

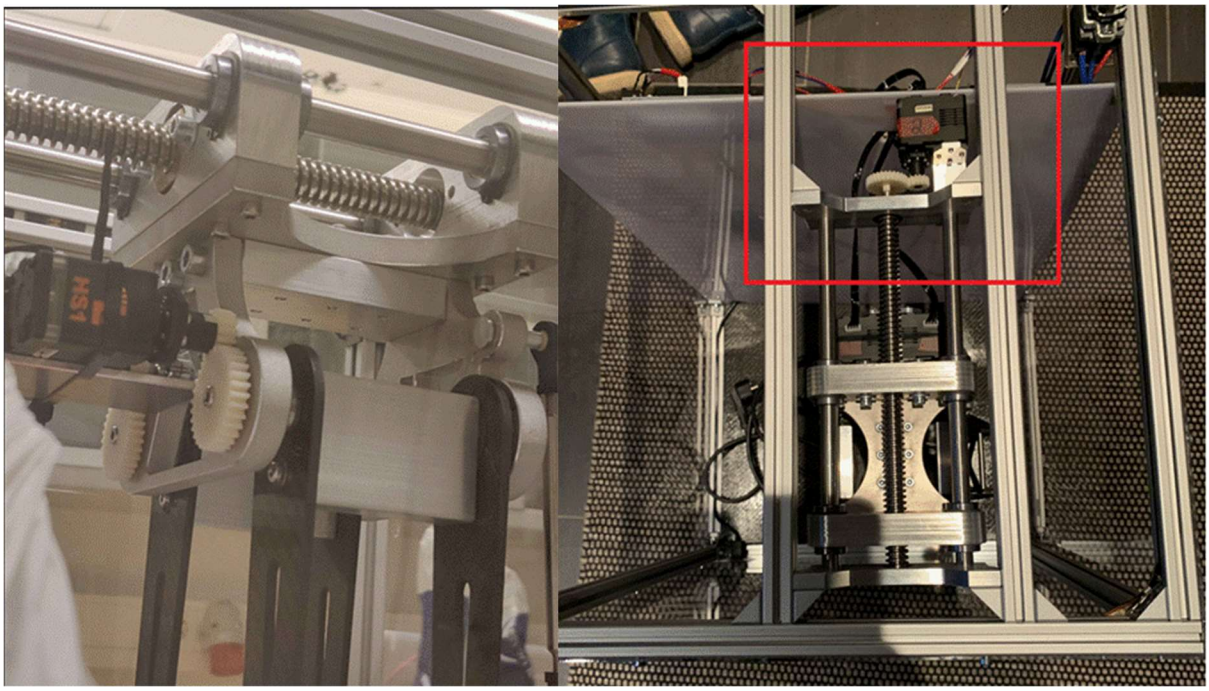


Figure 91 THE FIGURE TO THE LEFT SHOWS THE TWO MOTORS RESPONSIBLE FOR CONTROLLING THE MOVEMENT OF THE X AND Z AXIS WITH A GEAR RATIO OF 2 : 1, WHILE THE RIGHTMOST FIGURE SHOWS THE MOTOR RESPONSIBLE FOR MOVING THE Y AXIS WITH A GEAR RATIO OF 2 : 1 WITH THE HELP OF

Motor Overview

There are 4 motors required to move the robotic arms. Two for the X and Z movement of the arm, one for the railing which moves the arm in the Y direction and one to open and close the gripper. When calculating the torque required for the X and Z movement, equation [3] can be applied to give a result of 85.2Ncm. At this point in the design process, some parts of the robot had not yet been produced. Therefore, the assumed weight of the robot had to be taken from a combination of Autodesk Inventor 3d drawings and real-life weight of certain parts of the robot.

Arm motors

Dynamixel XC430-W150-T motors were chosen for the X and Z movement of the arms. The motors have a stall torque of up to 1.6 Nm and a max RPM of 106 depending on the Voltage and Ampere. The motors must rotate a relatively small amount for the full movement of the X and Z axes which is why a low RPM should not be a cause of concern when it comes to slow movements of the arms. The torque to RPM of the Dynamixel motors is shown in figure [92]

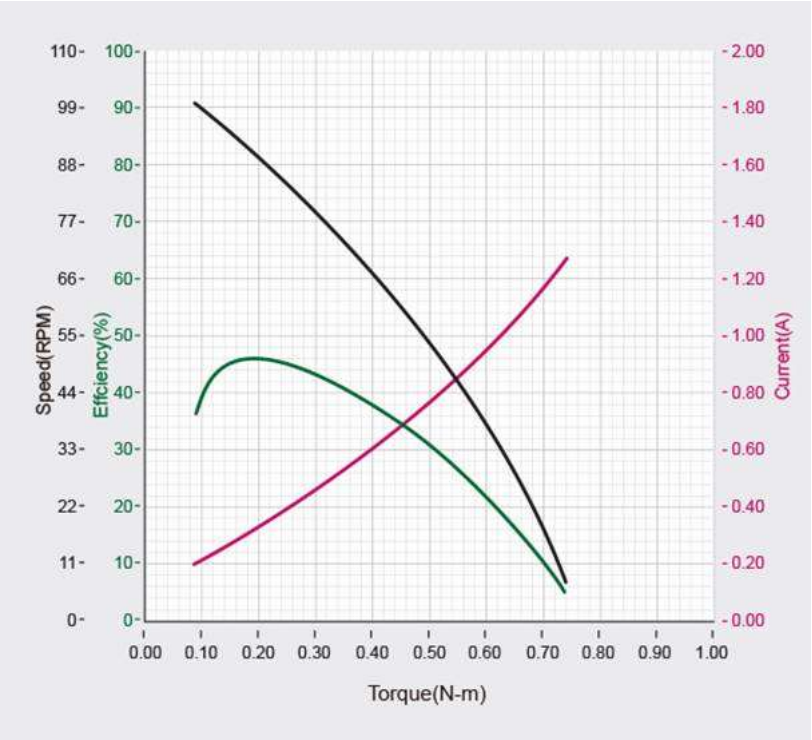


Figure 92 PERFORMANCE GRAPH OF THE XC430-W150-T. THE BLACK GRAPH SHOWS THE RPM AT DIFFERENT TORQUE VALUES. (Robotis, n.d-a)

As you can see with the black line figure [92], the Torque to RPM graph is exponential for the Dynamixel motor. For this reason, we want to keep the torque as low as possible, so that the velocity of the movement of the robot does not differ too much over time. This can be solved by adding a gear ratio to the motors, which changes the RPM of the output relative to the torque input of the motors. The original version of the robot used a gear ratio of 2:1 for the arms and the decision was made to keep that ratio. That means that the required torque of the motors would be halved, but in return the output RPM from the gears would also be halved.

There are two other motors to be placed on the robot, one motor is used for the Y axis movement while the other motor is used to open and close the gripper. Both motors use a lead screw system to move their respective parts of the system. To figure out the amount of torque needed from the motors, equation [6] can be applied.

Railing motor

The motor moving the arm in the Y axis had a produced and assembled design when this project was first started. To figure out how much torque was needed from the motor, equation [6] is applied to get a result of 5.46Ncm. At the start of the project, the robot was already using LSS-HS1 motors which had a maximum static holding torque of 78.48Ncm or 8kg-cm. However, as seen by the graph in figure [93], the RPM would experience a heavy drop before the motors would ever experience that amount of torque, but 5 is still within the

acceptable range of the motors.

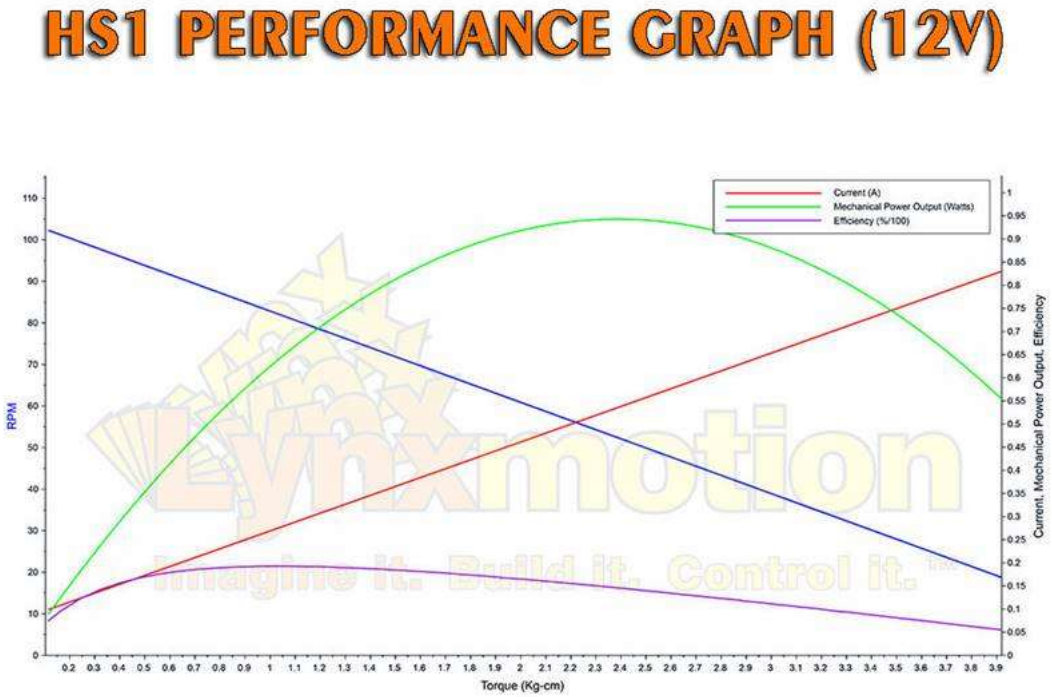


Figure 93 LSS HS1 MOTORS PERFORMANCE GRAPH. THE BLUE LINE SHOWS THE RPM AT DIFFERENT TORQUE VALUES. (Lynxmotion, 2024b)

The lead screw that originally came with the robot has a pitch of 4.2mm while the whole travel length of the Y axis is 110mm. That means that the lead screw must be turned 26 times for the full movement. That gives a total time for the full movement to be 16 seconds if the motor has a constant RPM of 100. One of the main goals of this project is to make a robot that is fun to play with at an exhibition and a relatively slow robot does not really fit into that description.

There are at least two things that can be done to increase the speed of the Y axis. Option number 1 would be to add a gear ratio. This increases the required torque, but in exchange for torque you increase the rotational speed. Option number 2 would be to increase the pitch of the lead screw which results in less total rotations needed for the full movement of the Y axis, but also requires more torque. At this stage, it was determined that testing would be important to know which change should be implemented while testing would also aid as supplement to the results of the previous torque calculations.

Motor and RPM testing

The goal of the testing is figuring out how much we can increase the speed of our system by applying more torque to the motor in return for more RPM. It is also desired that the motor runs as smoothly as possible, meaning little to no deviation from a set RPM while the system is moving. The program to turn on and measure the RPM of the motor is called “LSS Configuration Software” and is taken from the Lynxmotion wiki website.(Lynxmotion, 2024)

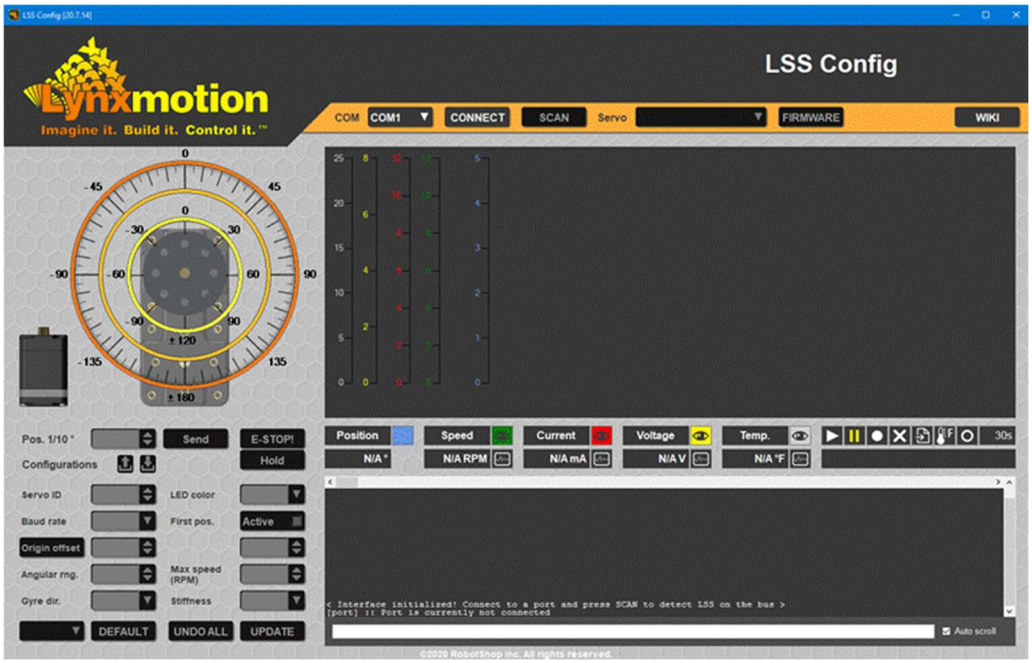


Figure 94 LSS CONFIGURATION SOFTWARE USED FOR THE LSS HS1 MOTORS. (Lynxmotion, 2024a)

The test began by first changing the gear ratio of the lead screw from 2:1 to 1:2. This was done since speed was our priority, and it is the easiest change to implement. Then a black dot was drawn on the gear which is placed on the motor. The motor is then connected to the LSS Configuration Software" where RPM is set to max. After, the motor is turned on by entering the wanted position in the program. The motor is then placed by hand on the gear connected to the lead screw, making the lead screw turn and the system moves in the Y axis. The motor RPM during the whole process can be read on the computer program. To confirm that the RPM readings on the program are correct, the testing process is filmed. Then the total amount of rotations during the full amount of time for the whole movement of the Y axis is to be counted manually. This process confirmed that the readings we were getting on the computer were correct.

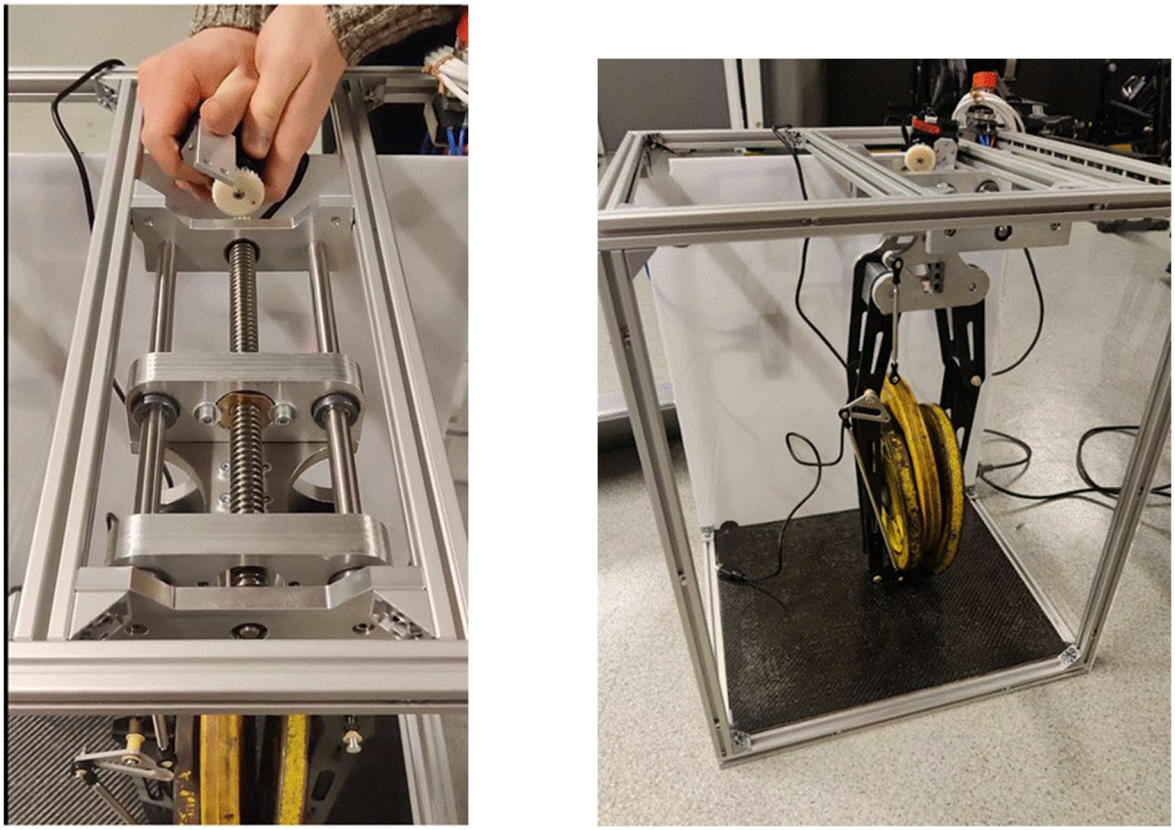


Figure 95 THE LEFTMOST PICTURE SHOWS THE LEAD SCREW TESTING SETUP WHILE THE RIGHTMOST PICTURE GIVES AN OVERVIEW OF THE TESTING SETUP WITH WEIGHTS

This process was done with different weights put onto the robot and the RPM was always set to 100. These weights would then simulate an increase in torque required from the motor to move the system. Table [3] shows the result from the testing.

Table 3 SHOWS TEST RESULTS OF THE LOWEST RPM RECEIVED DURING THE TESTING OF THE LEAD SCREW SYSTEM WHEN ADDING EXTRA WEIGHT

Added Weight(Kg):	Lowest RPM:
0	99
2	99
4	97
6	91
9	70
15	40

As seen by table [3], there was not a significant RPM drop before 6Kgs of weight were added. It was therefor decided that the amount of additional torque required to move the system with 6Kgs of weight should not exceed the additional amount of torque that a new pitch on the lead screw would create. With the added weight of 6kg, the amount of torque required to move the system can be found by again applying formula [6] which results in 0.077Nm.

To figure out the increase in pitch that matches a torque value below 0.077Nm, the system's desired speed first needs to be decided. The current total time to move the Y axis after changing the gear ratio was 6 seconds and the group decided that making this faster would make the robot more fun to use, which matches with one of our main goals of making this an exhibition-ready robot. Therefor the new pitch on the new lead screw was decided to be 6mm, which increases the pitch of the lead screw by 42% and should also make the robot 42% faster. But first, a torque check is needed to figure out if the motor has enough torque to fully support a trapezoidal lead screw with a pitch of 6mm. This can be done by again applying equation [6] to get a result of 0.043Nm which is well within the tested torque range of 0.077Nm.

There is however one more way to increase the efficiency of the motors, decreasing the torque required even more. The original lead screw was trapezoidal, meaning the threads of the screws are shaped in the form of a trapezoid. For our system, trapezoidal lead screw is less torque efficient than, for example, square threads. This is shown by applying equation [5] with the same overall values that you would apply in equation [6]. The results would then show that for this specific lead screw system, a square threaded lead screw would be around 10% more torque efficient. For this reason, the group had decided to go with a square threaded lead screw. However, upon inspecting the manufacturers website that the group was planning on getting the other lead screw from, there was no option for a square lead screw that fitted the correct dimensional requirements. Because of the benefits of getting as much as possible from one place, the decision was made to get the trapezoidal lead screw instead of the square lead screw.

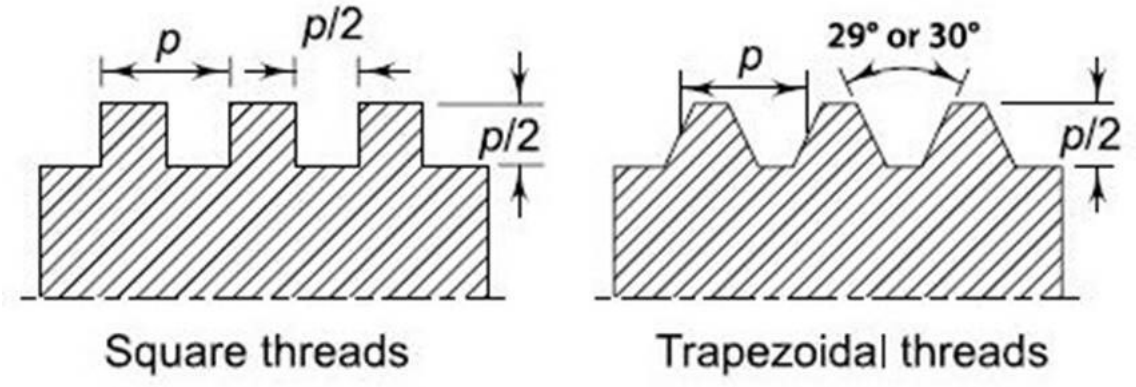


Figure 96 SHOWS THE DIFFERENCE BETWEEN SQUARE AND TRAPEZIODAL THREADS (Collins, n.d)

FIGURE 5.2.6

However, the whole design process of the robot ended up taking longer than expected, and that combined with the long shipping and manufacturing times of the lead screw made it very difficult to get the new lead screw in time for the final deadline for this project. This means that getting the lead screw with a 6mm pitch was unfortunately not possible, so the group decided to make use of the lead screw that was already there and only change the gear ratio from 2:1 to 1:2, which was the gear ratio used during testing.

Gripper motor

The gripper of the robot has a motor to control the opening and closing sequence of the gripper. This is done with the help of another lead screw. Because of the decision to change out the original two motors on the arms with two Dynamixel motors, the group had access to an excess amount of HSS-LS1 motors. For this reason, the decision was made to design the lead screw of the gripper opening and closing system around the HSS-LS1 motor.

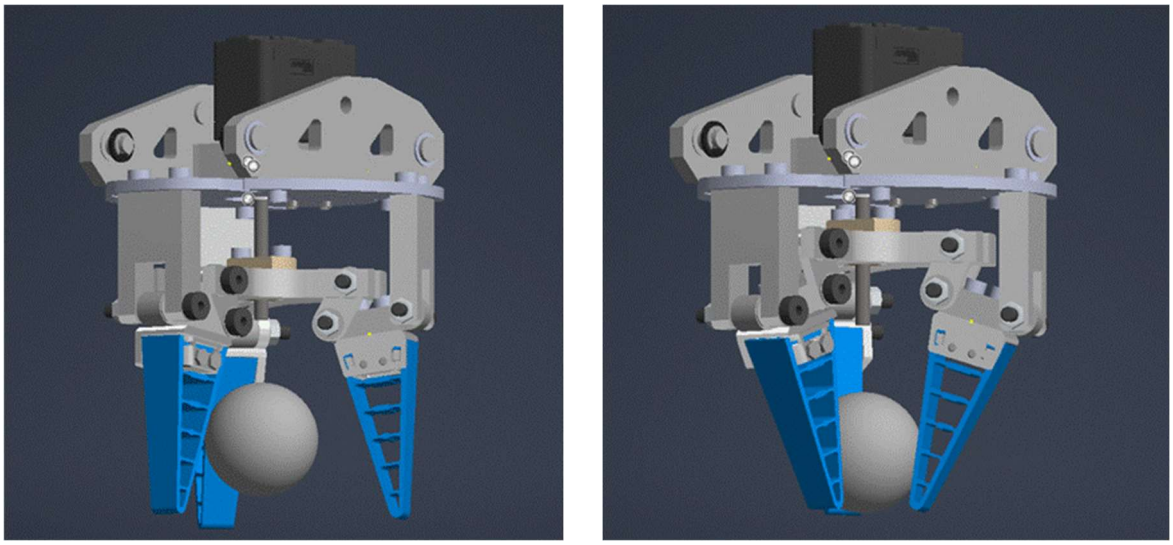


Figure 97 THE LEFTMOST PICTURE SHOWS THE GRIPPER IN A CLOSED POSITION, WHILE THE RIGHTMOST PICTURE SHOWS THE GRIPPER IN AN OPEN POSITION. WHEN THE SCREW TURNS, THE MIDDLE PLATE IS MOVED UP OR DOWN, CLOSING AND OPENING THE GRIPPER

The lead screws diameter had to be quite small to fit the motor adapter on the new gripper. Because of this the desired lead screw was one with a diameter of no more than 6mm. It was also known that to fully open and close the gripper design, the full motion on the lead screw was 10mm. With this information, a lead screw pitch can be chosen so that the gripper opens and closes at a good pace. If the gripper should close in less than 2 seconds, the lead screw should have a pitch of 3.2mm or higher. If the lead screw is assumed to be square threaded with a diameter of 6mm, we can reorient equation [5] to get a max pitch of over 50mm. This result means that the pitch used on the chosen lead screw is not going to cause a torque problem. However, when looking at manufacturer's websites, it was noticeable that high pitch M6 lead square screws are not really being made. There is however one alternative thread type on lead screws, the high helix thread. Because of its unusual thread type it is harder to calculate the torque values required to move a helix thread, however, the manufacturer Igus has its own calculator on its website to show if the lead screw is compatible with your system. Since a pitch of 3.2mm or higher is required, let's check if a high helix lead screw with a pitch of 5mm is compatible with our system. (Igus, n.d)

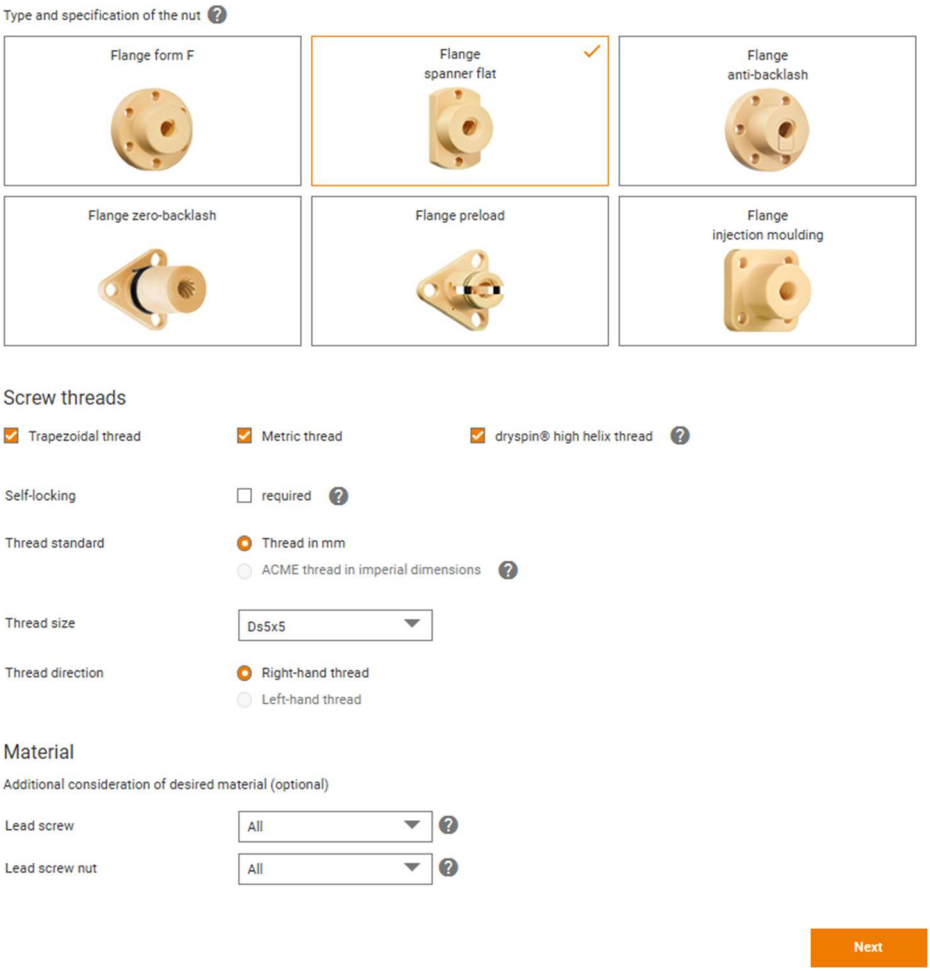


Figure 98 SHOWS THE FIRST STEP IN THE IGUS CALCULATOR, THIS SECTION LETS YOU SELECT WHICH TYPE OF THREADING NUT YOU WANT TO USE FOR YOUR LEAD SCREW SYSTEM. NOTE THE INPUT OF THREAD TYPE DS5X5, WHICH IS A HIGH HELIX LEAD SCREW WITH A PITCH OF 5MM (Igus, n.d)

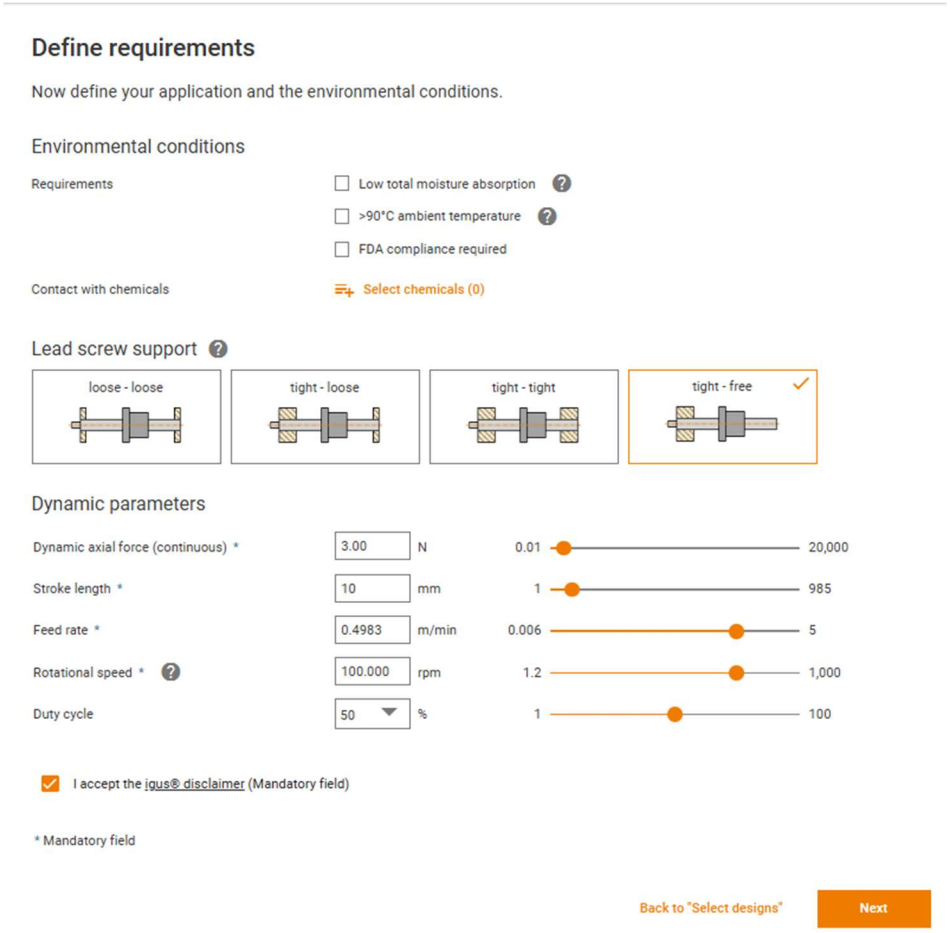


Figure 99 SHOWS THE SECOND STEP IN THE IGUS CALCULATOR. HERE, WE CAN PLACE IN THE SPECIFIC VALUES FOR OUR LEAD SCREW SYSTEM (Igus, n.d)

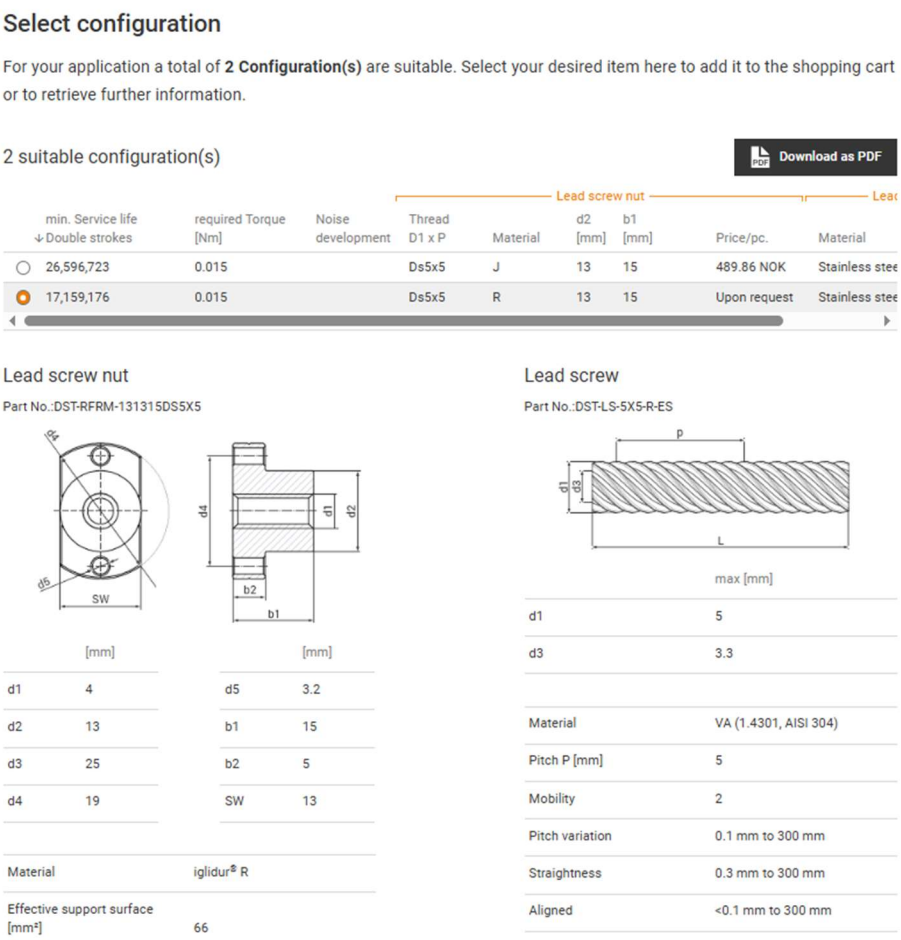


Figure 100 SHOWS THE THIRD STEP IN THE IGUS CALCULATOR, WHICH SHOWS US WHAT TYPES OF LEAD SCREWS AND NUTS FIT OUR REQUIREMENTS (Igus, n.d)

As shown in the above figure [100], the required torque is 0.015Nm on a Ds5x5 high helix thread screw with a pitch of 5mm. This amount of torque should not be a cause of concern for the HSS-LS1 motors. This pitch theoretically results in an opening and close time of 1.25 seconds for the gripper. This is the screw that the team ended up choosing for the final product. However, for the same reasons mentioned before with the design process taking longer than expected and the long shipping and manufacturing time of the Ds5x5 high helix lead screw, it will not be possible to use the Ds5x5 for the final product deadline for this project. Instead, a M6 screw will be used with a pitch of 1mm. What this means is that the lead screw system will probably be 5 times slower than what was originally planned. This is unfortunate, but this way the opening and closing mechanism can at the very least be tested to confirm that it works.

6. Control system design and development

6.1 Final hardware integration

The complete system integrates several components including the Raspberry Pi 4 Model B, LSS HS1 servo motors, Dynamixel servo motors, 7-inch display, PS4 controller, drivers, Beam break sensor, safety button and a power supply. These components were assembled and tested to meet the updated criteria for wireless control, gripper, interactive display and improved movement.

Power is delivered through the power supply connected to a wall socket providing AC current. Both DC outputs provide 12 and 5 volts for the Pi and the HS1 driver, and the sensor is powered directly from the GIOP's of the PI, with an additional breadboard. It is important to remember the Dynamixel driver (Power hub) is not connected to the power supply (RD-85) but connected to a wall socket with its own adapter providing 12-volt input as well as the power supply. The movement is controlled by a PS4 controller, directly connected to the Pi's Bluetooth capability and can control all servo motors in the system.

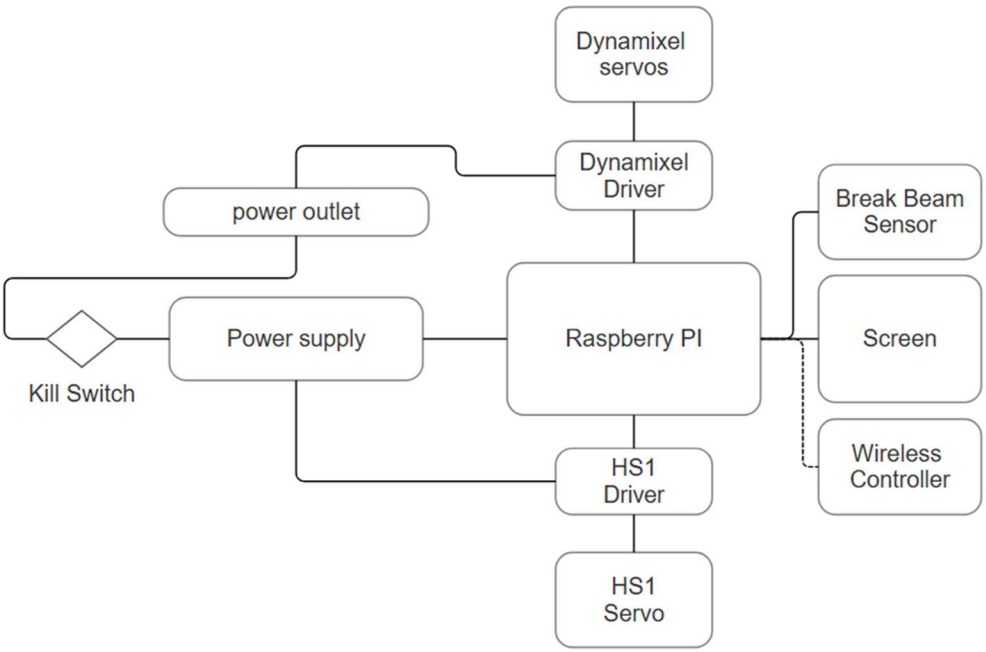


Figure 101 Final hardware electric diagram where the black lines represent electrical wires.

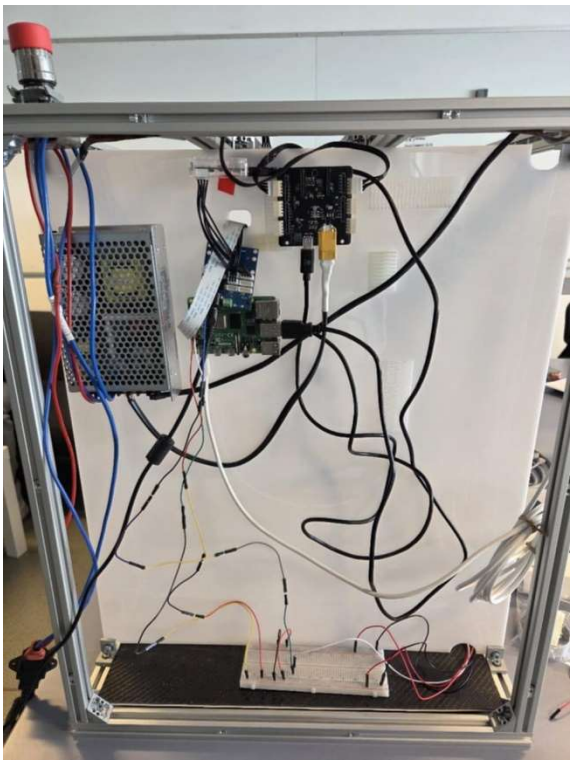


Figure 102 Final physical hardware design and implementation of the whole system. The drivers are located on the top side of the backplate, while the grippers furthest theoretical position are down at the bottom.

The summarize the final hardware design compared to the previous group with great focus on reusing components as much as possible is the implementation of the Dynamixel motors, sensors, wireless PS4 controller the removal of all Arduino components.

6.1.1 Daisy Chain

The essential upgrade enabling inverse kinematics to function properly during control is credited to the implementation of the two Dynamixel motors. The final setup, focusing on the Dynamixel components in this paragraph, included two motors of the same type (XC430-W150), along with the U2D2, both connected to the driver. When both motors were mounted in their designated locations by the mechanical department, as shown in Figure 103, a wire length issue occurred. Specifically, when the railing was positioned at its furthest point from the backside of the system, where the drivers are mounted and one of the Dynamixel motor cables was too short to reach the driver.

Based on information from the manufacturer, Dynamixel motors support a feature called Daisy Chaining, which in electronics is another term for serial connection (Robotis, 2025). This allows only one motor to be connected directly to the driver, while the second motor to be connected with the shorter cable to the first motor instead. This solution was implemented and is also illustrated in Figure 103.

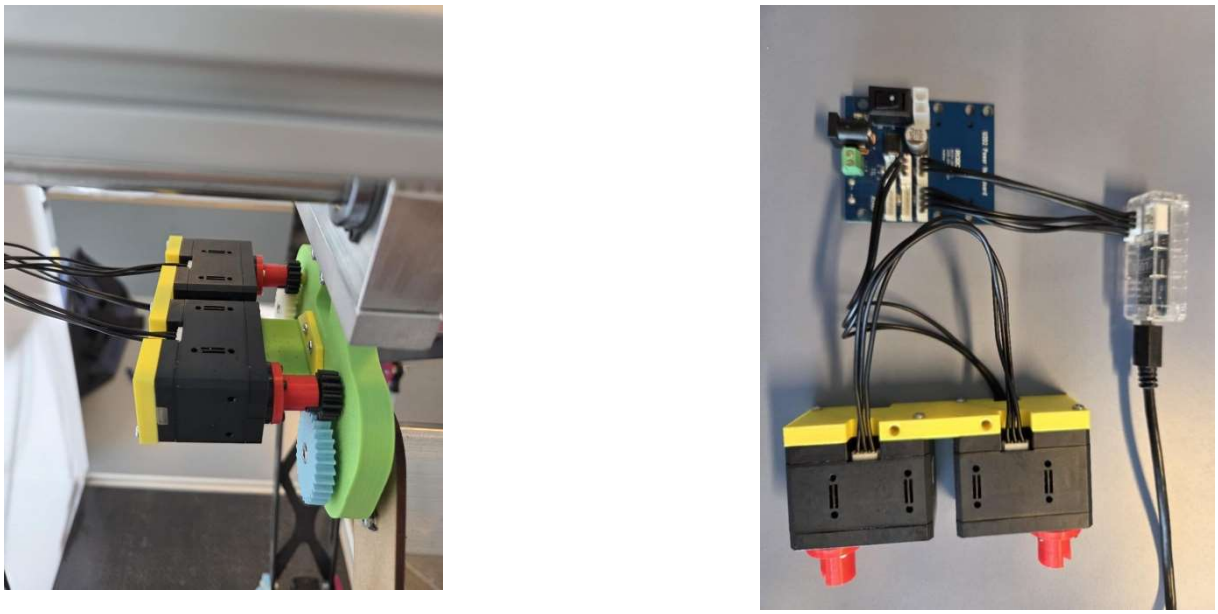


Figure 103 leftmost picture shows both motors mounted to the designed positions while the rightmost picture shows the whole implementations of dynamixel components with both motors Daisy Chained

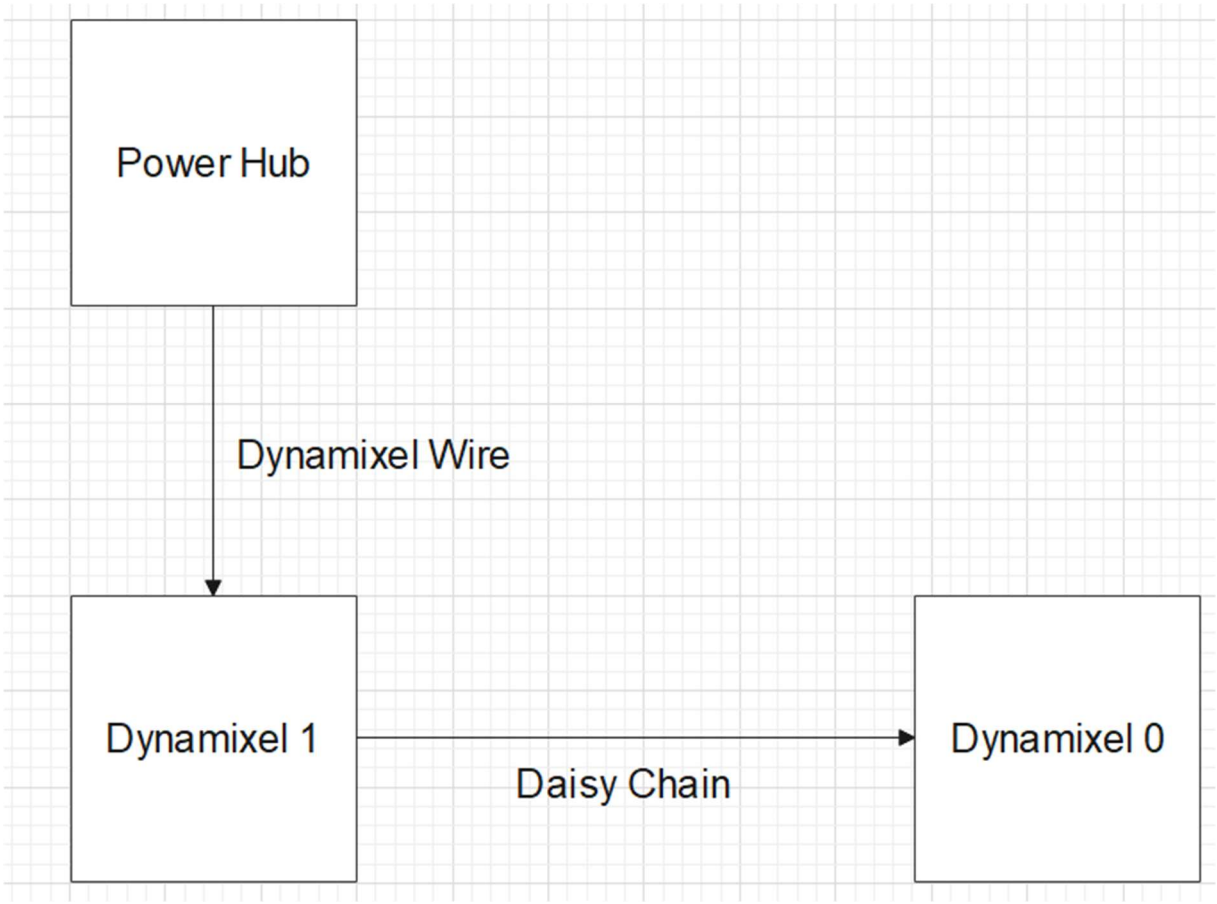


Figure 104 Block diagram on how the Daisy chain is implemented in more detail

6.1.2 Total Draw

The drivers are contributing power to the servo motors received by the power supply. Both HS1 motors together produce current at 1.3 A (Nantel, 2024) and both Dynamixel motors at 2.8 A (Robotis, n.d-b). Each connector of the HS1 driver can handle maximum 3A (Nantel, n.d) and the Dynamixel driver can handle maximum 10 A (Robotis, n.d-a). This whole implementation is well within the limitations of the drivers where power is provided by the power supply and not directly from the Raspberry.

The **Pi itself** takes a maximum of 5V with maximum 3A, this means it can operate at maximum 15W (Ltd, 2024).

$$P = VI,\ 5V*3A = 15W$$

The maximum draw in total for the 4 USB3 inputs from the Pi is at 1.1 A (Ltd, 2024). The display operates at 200 mA (Ltd, n.d-a) and the sensor at 25mA (RoboShop), both at 5 Volt input, including two USB3 are occupied for data transfer by both drivers. This is well inside the power limitations of the Pi and should ensure no overload on the board during usage.

6.1.3 PS4 Controller

Through our own research into Arduino boards and their wireless communication capabilities, we found that neither the Arduino Mega (Arduino, 2025a) nor the Nano (Arduino, 2025b) described in their thesis has built in Bluetooth capabilities. While Bluetooth communication could in theory be achieved with other types of Arduino Nano boards such as the Nano 33 BLE (Arduino, 2025c), the Arduino Mega which was eventually used for the controller can only possess Bluetooth with an additional module called HC-05. The module is not made by Arduino themselves, but is provided by third party suppliers (101, 2021). Since the previous group's thesis did not have a mention of such a module, it is assumed that their controller was intended to function as a wired controller.

Instead of designing a new controller by integrating a Bluetooth module for the Mega board to reuse components as much as possible, or a Nano board with Bluetooth capabilities such as the Nano BLE 33, there was a strong desire from Tronrud's supervisor to have a wireless controller from common console companies.

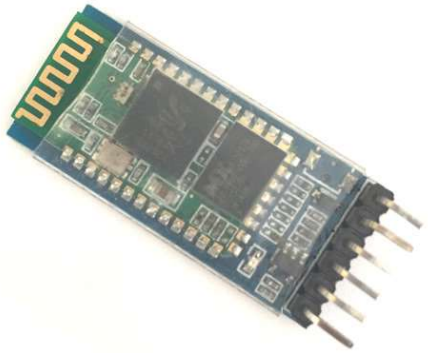


Figure 105 Bluetooth module for Arduino boards (101, 2021).

6.1.4 Wire issues

The DSI connector is a flexible cable enabling graphics to occur on the display by transferring data from the Pi. The connector worked as expected, but there was a slight issue with its length. The cable was too short to be displayed in the front of the robot, while the Pi at the same time is attached to the backside, providing data transfer. Depicted in figure 106

is how the original set up regarding the display connector where it is clear the wire was too short. The solution was eventually solved by ordering a longer cable at 100 cm length in figure 107.

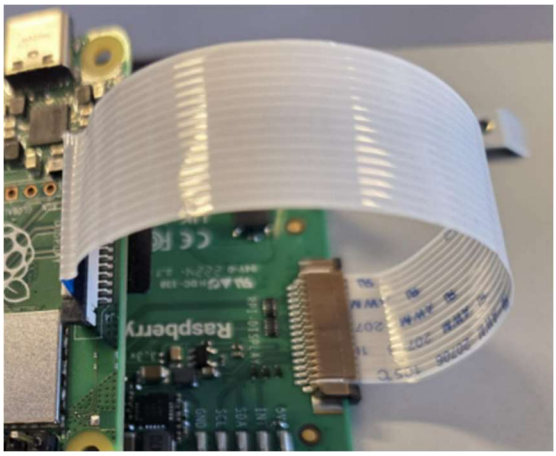


Figure 106 Raspberry Pi 4 connected to the backside of the display

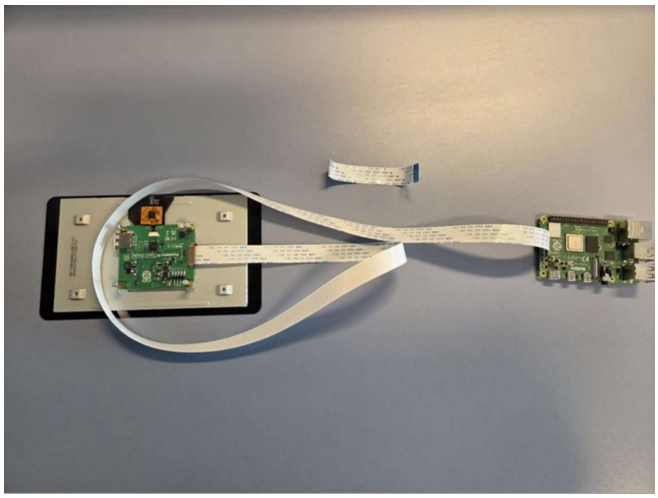


Figure 107 New DSI connector at 100 cm compared to the original DSI connector.

Beam break sensor

There was an issue with the compatibility of the wires attached to both parts of the sensor. Both GPIO pins on the Raspberry and the sensor wires are male type, which means the sensor cannot be directly connected to the Pi. The desired outcome was to connect the sensor direct to the GIOP of the Pi as female to male connection. Instead, a Breadboard had to be implemented.

On the breadboard, common voltage and ground are shared across the railings provided from the Pi. An additional male to male wire had to be implemented and connected for the GIOP 17 from the PI with a cross line on the breadboard, where the receiving sensor can transfer data.

Figure 108 Beam break sensors with male connectors.

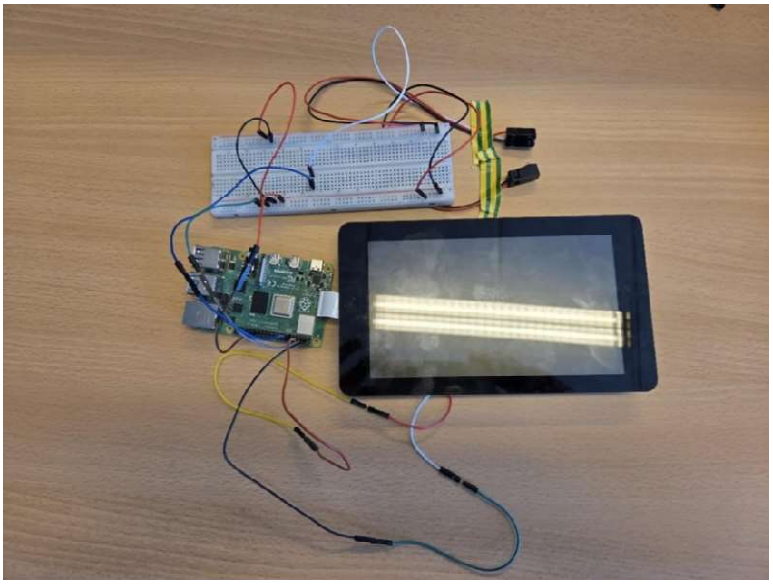


Figure 109 The sensor wired with a breadboard with the display and the Pi isolated for testing purposes.

Jump wires

To power both the sensor from the breadboard and the display with the Raspberry Pi, the issue was wires not long enough to reach from the Pi to the front of the robot, where the display would be located, and to the bottom, where the sensor would be placed.

Using jumper wires, typically seen in common hobby equipment such as Arduino, the problem for both parts was solved by assembling a set of males to female connections to create a series of extended jumper wires. This implementation can be seen in Figure 101.

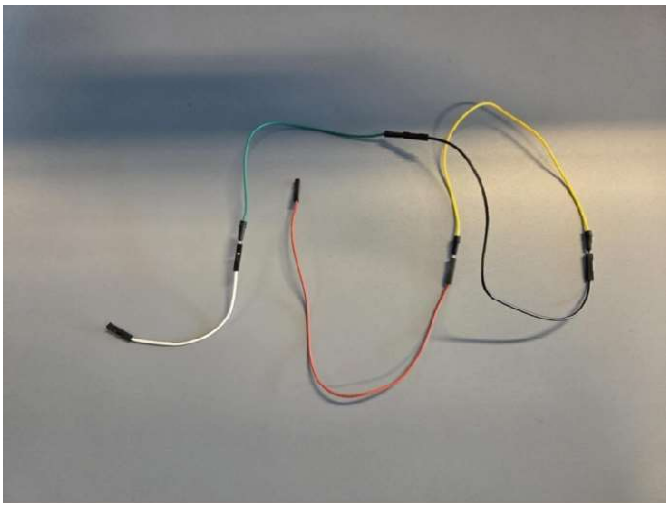


Figure 110 Male and female jump wires in a series of connection to increase length of wire.

6.1.5 Soldering

Gripper HS1 motor

As mentioned in Section 4.2.3, both HS1 motors were equipped for the railing and the gripper and replaced by Dynamixel motors for the arm due to torque differences. The driver for both HS1 motors is mounted on the backside of the system along with the other hardware components. The location of the railing motor allows the wire from the driver to reach it without any issues. However, unlike the railing motor, a wire length issue occurred regarding the HS1 motor installed at the gripper because its location is much further down compared to the railing.

The previous group had only wires with a length of 20 cm available, measured with a measuring tape in figure 111, which was too short for the gripper from the top of the backside. Researching the manufacturer's website, we found that wires were only available in lengths up to 30 cm (Nantel, 2024b). Therefore, we ordered several wires of that length and soldered them together using a soldering machine in figure 113.

Figure 111 The only available length of wire for the HS1 motor from previous group. Length measured with a measuring tape.

Once the wires arrived, each end was cut with a scissor, and the wires were soldered together. It was crucial to follow the mapped input configuration for the HS1 motor provided by the manufacturer, shown in figure 112, to correctly solder the I/O's together (Nantel, 2024c). In the final stages of soldering, the resulting wire was simply a longer version of the original 30 cm wires. Each soldered section was insulated using electrical tape. To verify the success of the soldering, the extended wire was connected between the driver and the gripper motor. The motor lit up in the color defined in the firmware, indicating that it was receiving both data and power from the driver.

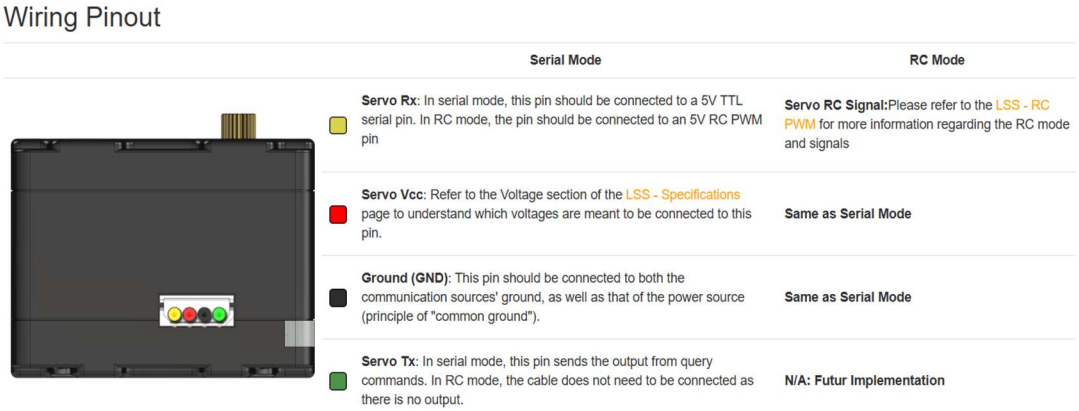


Figure 112 I/O's of a HS1 motor provided on the website of manufacturer (Nantel, 2024c).

Figure 113 Soldering machine tin

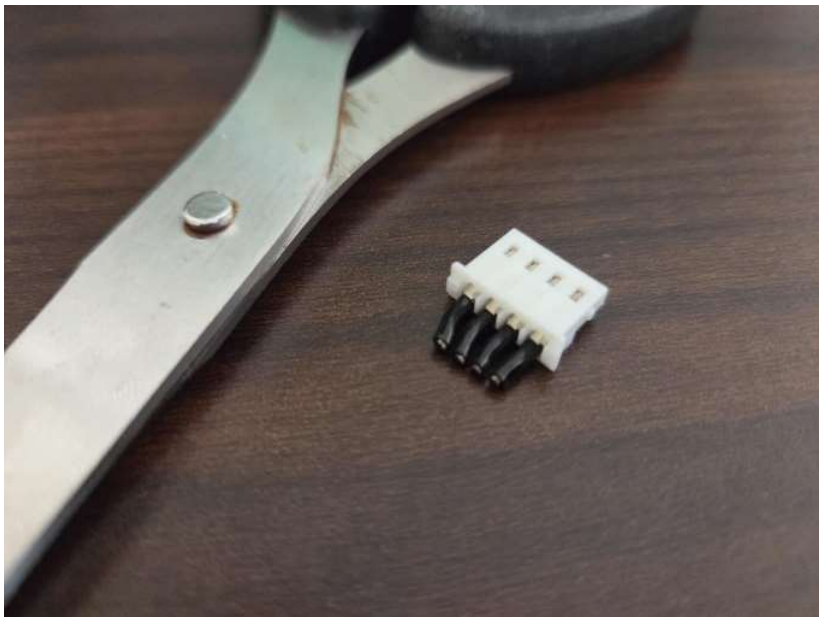


Figure 114 End wire cut

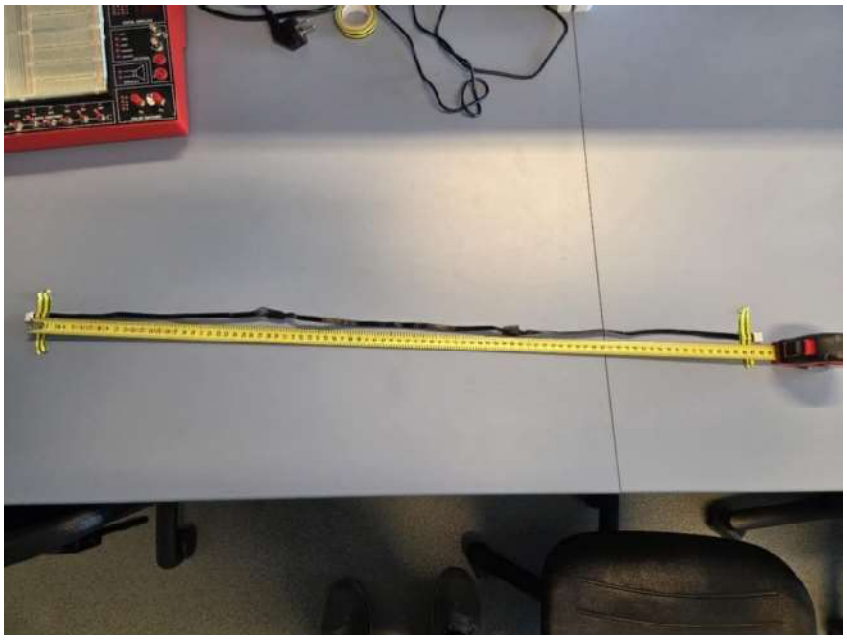


Figure 115 total length of soldered wire at approximately 87cm

Figure 116 Soldered wire with electrical tape for isolating each soldered point across the wire

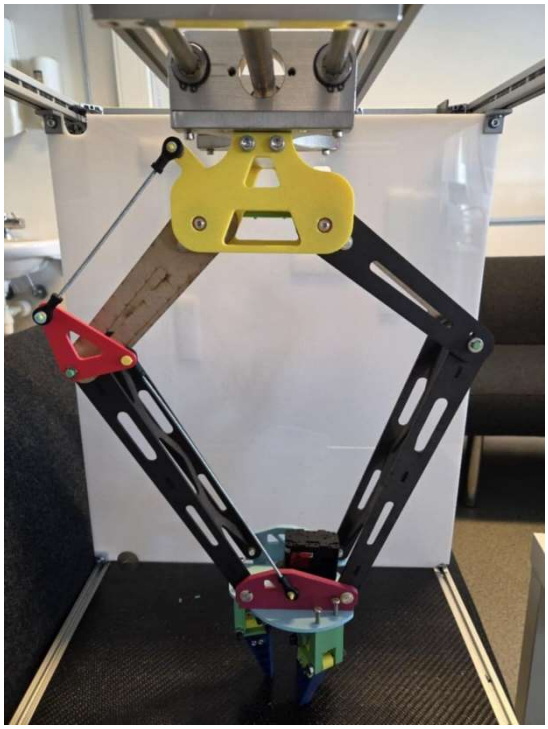


Figure 117 leftmost picture shows soldered wire connected to backside of HS1 gripper motor. A white light from the motor, indicating both the flow of currect and successfully soldered wire. Rightmost picture shows the gripper at the bottom compared to the railing

Further Dynamixel wire issues

Similar wire length issues as those encountered with the HS1 gripper motor also occurred with the Dynamixel motors. The initial solution for the Dynamixel motors, as described in Section 6.1.1, was to use a manufacturer technique called Daisy Chaining, which solved the previous length problem between the driver and the motors.

However, this time the wire connected to one of the Dynamixel motor and not the wire for daisy chaining, proved to be just slightly too short when the railing was positioned at the other side in the Y-direction of the system. Since the earlier soldering of the HS1 gripper motor wires had proven to be successful, the same approach was chosen here. Several wires of approximately 17 cm, which is the longest wires the manufacturer had, were ordered and soldered together using a soldering machine, as shown in Figure 113, to achieve the required length.

Figure 118 length of original Dynamixel wire at approxately 17cm, slightly too short.

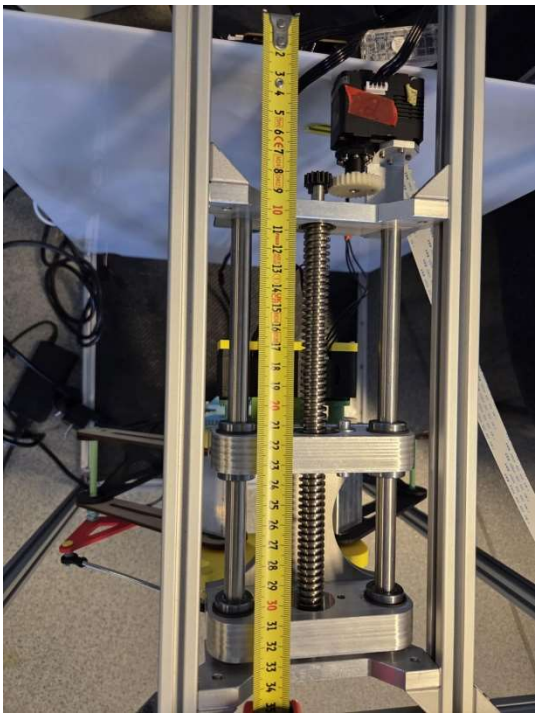


Figure 119 Total length from the power hub to the other side of the railing at approximately 30cm, while to the dynamixel is around 21cm

Figure 120 The total length after soldering at approximately 31cm

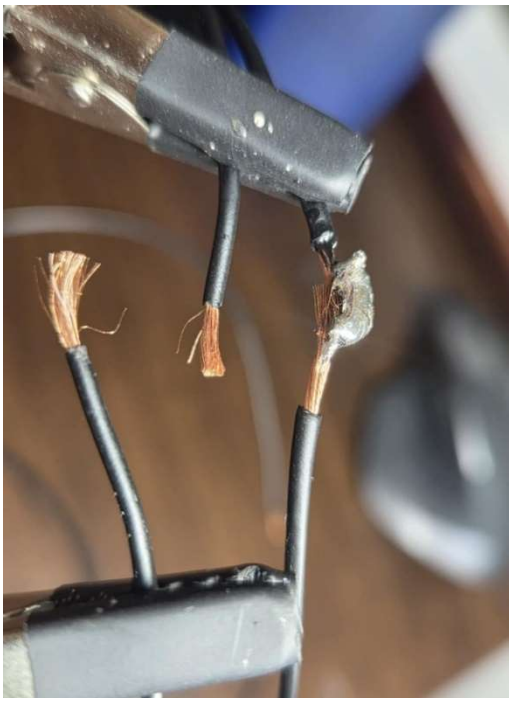


Figure 121 Dynamixel wires cut and soldered with Tin alloy

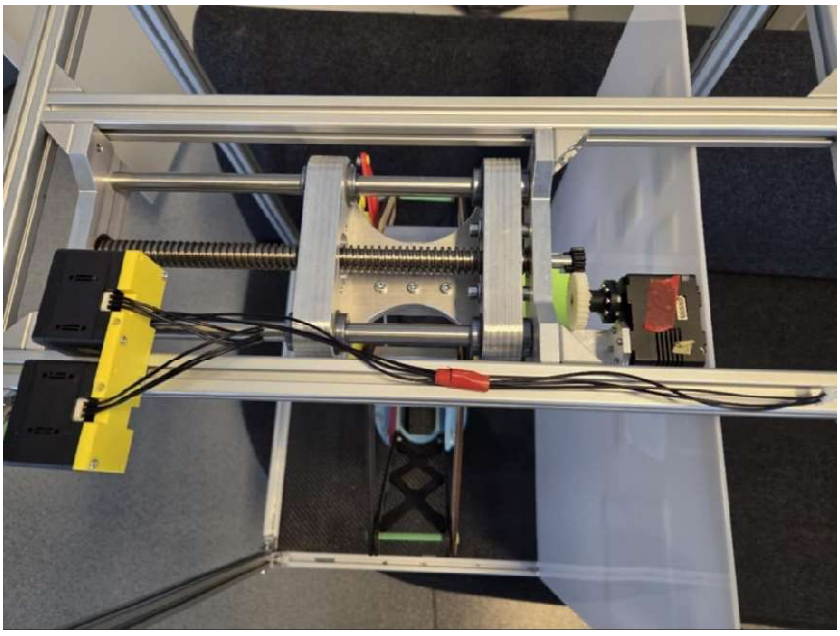


Figure 122 The Dynamixel wires on the other end of the railing, where the soldered wire is long enough to solve the length issue.

6.2 Programming and Software Development

All software runs on a Raspberry Pi with Python 3.13. Input from a Bluetooth paired DualShock 4 (PlayStation 4) controller is translated into Cartesian target positions. An inverse kinematics (IK) solver converts these coordinates into joint angles. A boundary safety layer checks that each angle stays within permitted limits before the values are packaged for the Dynamixel and HS-1 servomotors and sent over the USB interface.

The on-screen menu shows four items:

1. **Tutorial**: quick start guide for new users
2. **Start Game**: launches the basketball demonstration
3. **High Score Table**: displays the leaderboard
4. **Free Roam**: enables manual operation with a live simulation overlay

Figure 123 presents a high-level block diagram of the software modules, Figure 124 shows the runtime message flow, and Figure 125 maps the external library dependencies.

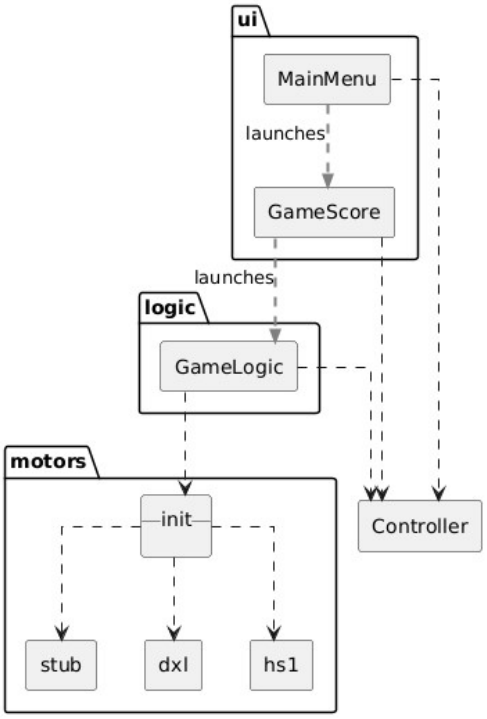


Figure 123 Software architecture block diagram

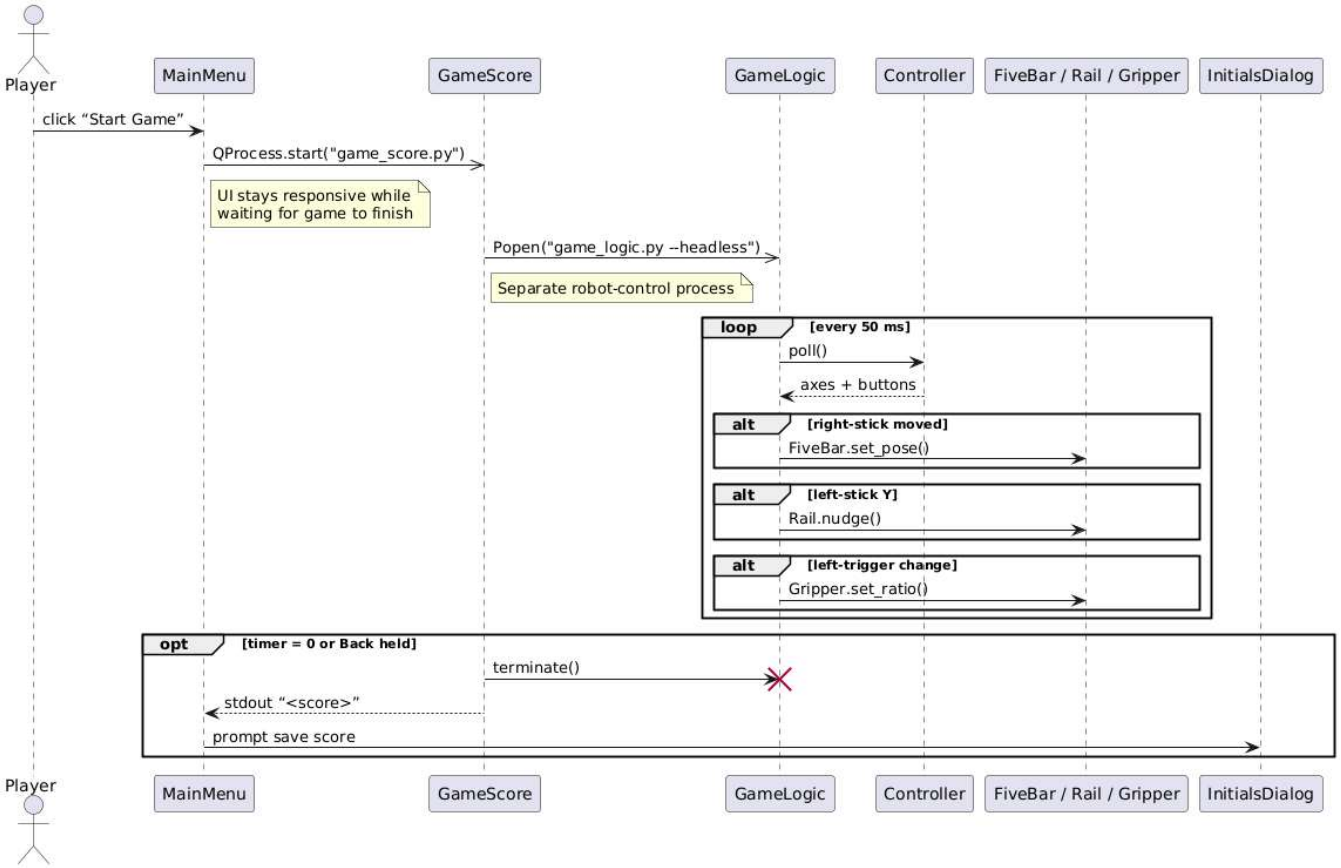


Figure 124 Runtime sequence and data flow between modules

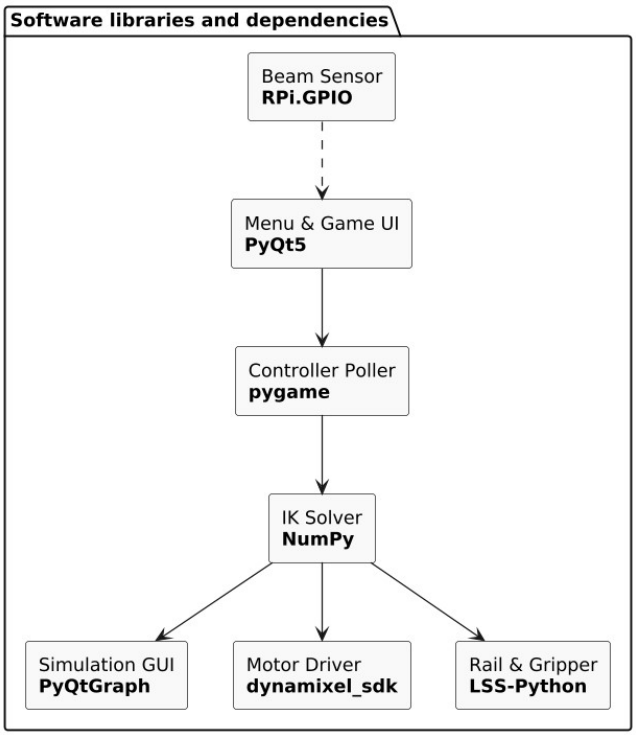


Figure 125 External library dependencies

At a high level, controller.py handles all input events and is used by both the user interface (UI) and game logic layers. *main_menu.py* is the program entry point: it loads the remaining modules, shows the score board, and provides access to the tutorial sheet. The overall architecture appears in Figure 123.

After launching, the main loop continuously polls controller.py for inputs, to enable menu navigation. When the user selects Start Game, *main_menu.py* starts *game_score.py*, which instructs *game_logic.py* to run in headless mode, which means, without the simulation window. The display therefore shows the scoring interface while the robot operates in the background. *game_score.py* monitors the break beam sensor; a low to high transition increments the score and triggers a celebratory GIF. When the countdown timer expires, players who have a non-zero score can enter three initials, which are stored in a JSON file. Overall data flow between these programs is illustrated in Figure 124.

6.2.1 Simulation

Because operating the physical prototype involves safety, cost, and time constraints, a desktop-level simulation was created to act as a digital twin of the arm. This virtual model serves four complementary purposes:

- **Early code validation** executing control algorithms on a digital twin exposes software defects before power is applied to hardware.
- **Inverse-kinematics verification** confirms that the solver produces correct joint angles across the reachable workspace while physical parts are still unavailable.
- **Mechanical-design support** provides the CAD team with instant visual feedback, helping them tune link lengths and joint placements.
- **Rapid iteration** allows parameter sweeps and experiments to run in seconds, eliminating downtime caused by physical re-rigging.

To implement inverse kinematics, several software tools were considered based on criteria such as learning curve, simulation capabilities, licensing costs, and compatibility with our hardware platform.

- **MATLAB**: Despite powerful analytical tools, MATLAB was ruled out due to a steep learning curve and limited availability of ready-to-use inverse kinematic modules specifically suited to our needs.
- **ROS**: The previous group successfully used ROS due to its comprehensive robotic simulation capabilities. However, running ROS optimally would require a virtual machine, since Linux is its primary operating system. Given this additional complexity, ROS was initially postponed in favor of exploring simpler alternatives.
- **Python (PyBullet)**: Initially chosen due to its 3D physics-based environment, PyBullet was found unsuitable for our robot arm since it supported limited joint types, primarily ball joints, resulting in unrealistic simulations and inaccurate results. An early development framework with ball joints is depicted Figure 126.

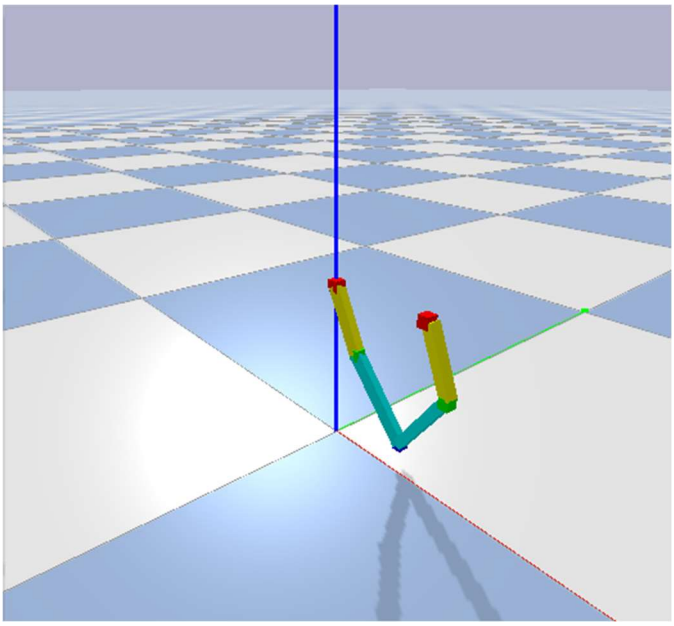


Figure 126 Simple PyBullet simulation with ball joints

After reevaluating project requirements, we determined that a simpler, two-dimensional simulation would suffice because the inverse kinematics calculation was only required along two axes (X and Y). This led to selecting Python (NumPy + PyQtGraph): NumPy, a standard math library, combined with PyQtGraph for fast graphical rendering, allowed rapid iteration and accurate visualization. A functional 2D simulation was quickly achieved, demonstrating the correctness of our inverse kinematics algorithms.

The simulation focuses exclusively on planar motion in the XY plane. The vertical Z axis is actuated by a separate linear servo, so it falls outside the inverse-kinematics (IK) problem. The planar portion behaves as a five-bar parallel linkage. For each target Cartesian coordinate the solver evaluates a closed-form matrix equation to obtain the two active joint angles; link-length constants are read from a configuration file that can be reloaded at runtime. During prototyping the target coordinate is nudged with the keyboard arrow keys: each key-press translates the desired point by 5 mm in the corresponding direction. This minimal input method sufficed to execute the IK-solver and confirm smooth rendering. The render loop executes three stages on every frame:

1. **IK stage:** Computes the joint angles (θ_1, θ_2) that place the end-effector at the updated target.

2. **Forward-kinematics stage:** Derives the Cartesian positions of both elbow joints from these angles.

3. **Draw stage** plots the base, elbow joints, and end-effector, then connect the points with line segments to display the current five-bar configuration(Figure 127).

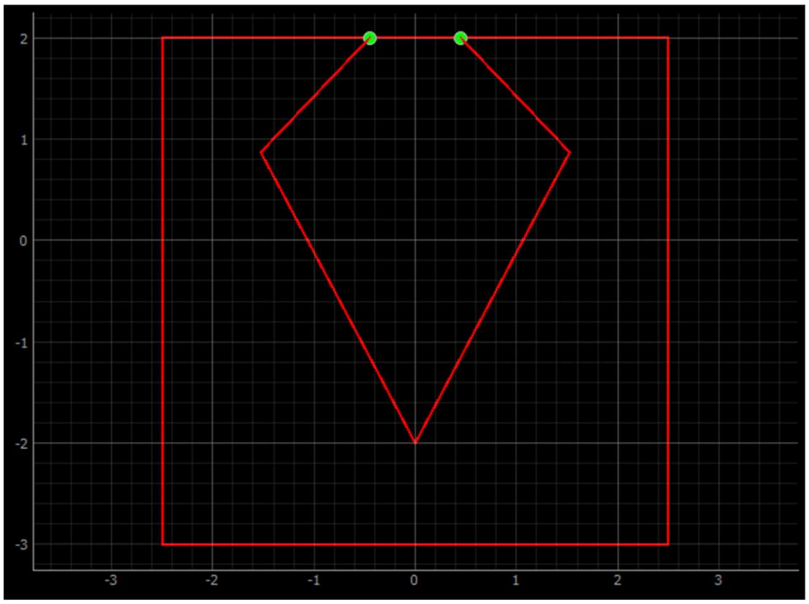


Figure 127 First generation real-time simulation for the Five Link robot arm with PyQtGraph and NumPy

Moving from PyBullet to a lightweight 2-D NumPy + PyQtGraph model inevitably sacrifices realism in two areas. First, the Z-axis rail is not animated; it is treated as a fixed offset because its motion is governed by an independent linear actuator that requires no inverse-kinematics. Second, the simulator omits physics**,** mass, inertia, friction, and gravity—so it cannot predict loads or collision forces. These omissions are acceptable for the present thesis since coordinated planar motion is the critical challenge, and the chosen servomotors include encoders and closed-loop position control that can compensate for minor un-modelled effects. Should future work require dynamic analysis or 3-D path-planning, the model could be migrated back to a physics engine such as PyBullet.

6.2.2 Boundary Checking and Safety Measures

Table 4 lists every parameter that the program validates before accepting a new target coordinate. When the IK solver is called it first checks whether the arm is physically long enough to reach the requested point; if not, the solver returns *None*. It then confirms that the Cartesian coordinate lies inside the permitted play area so the end-effector cannot collide with the side walls or the basket hoop. Finally, after θ_1 (the base-link angle) has been computed, an additional check ensures that the angle remains within a safe range that prevents contact with the rail beam and avoids crossing the base link value 0, at which point the servo will try and go all the way around, possibly breaking the arm.

Table 4 Boundary spread sheet.

What			Unit	Description
Link length L1 (base → elbow)	156.0	156.0	mm	Fixed first segment

Link length L2 (elbow → wrist)	325.0	325.0	mm	Fixed second segment
Reach per base (L1 + L2)	0.0	481.0	mm	d_1, $d_2 \leq 481$ mm
Workspace x-coordinate	−250.0	250.0	mm	Allowed X range
Workspace y-coordinate	−200.0		mm	Allowed Y lower bound
Basket hoop zone	—	—	—	x > 100 mm ∧ y < −150 mm
θ_1 (left link)	−4.20	−1.60	rad	(−240.6 °, −91.7 °) before servo mapping
θ_2 (right link)	−1.55	1.19	rad	(−88.8 °, 68.2 °) before servo mapping

6.2.3 Inverse Kinematics Algorithm

With the workspace boundaries now defined, the inverse-kinematics routine advances through the stages that verify a requested end-effector pose and return the joint angles if the pose is admissible. Each stage ties directly to the expressions derived in Theory 3.5. The algorithm steps are as follows:

- **Workspace limits:** Return *None* if x is not [−250, 250] mm or y < −200 mm.
- **Reach test:** The distances $d_1 = |E - A_1|$ and $d_2 = |E - A_2|$ must satisfy $d_i \leq L_1 + L_2 = 481$ mm. Any violation returns *None* and no change will occur.
- **IK calculations:** Angles α_1, α_2, β_1, β_2 are obtained directly from Eqs. 9 - 12. Actuator angles follow Eqs. 7 - 8.
- **Joint-limit filter:** Allowed ranges: $-4.20 \text{ rad} \leq \theta_1 \leq -1.60 \text{ rad}$, $-1.55 \text{ rad} \leq \theta_2 \leq 1.19$ rad. Solutions outside this range result in returning None and no change will occur.

6.2.4 Controller Integration

The next development milestone was replacing the crude arrow-key interface with a handheld game controller. Although the previous group started designing a custom controller, we determined that building our own offered little benefit relative to the effort involved. With limited manpower and time, purchasing a commercial Sony DualShock 4 for about 400 NOK gave us a proven device that outperforms any homemade solution in ergonomics, sensor resolution, and firmware reliability. The controller's standard USB-Bluetooth HID profile is fully supported on PC, and the open-source PyGame library reads every button, D-pad state, analogue stick, and pressure-sensitive trigger through a single event loop. Using this library, the hardware team contributed a basic input module that plugged straight into the existing simulation, which until this point had accepted only keyboard commands.

Table 4 Input mapping for PC and Raspberry Pi.

Control	**PC Mapping**	**Raspberry Pi Mapping**	**Signal Type / Range**
D-pad ↑	Button 11	HAT 0 (0,+1)	Digital

D-pad ↓	Button 12	HAT 0 (0,−1)	Digital
D-pad ←	Button 13	HAT 0 (−1,0)	Digital
D-pad →	Button 14	HAT 0 (+1,0)	Digital
Select (×)	Button 0	Button 0	Digital
Back (◯)	Button 1	Button 1	Digital
Left-stick X (**lx**)	Axis 0	Axis 0	−1 … +1 (float)
Left-stick Y (**ly**)	Axis 1	Axis 1	−1 … +1 (float)
Right-stick X (**rx**)	Axis 2	Axis 3	−1 … +1 (float)
Right-stick Y (**ry**)	Axis 3	Axis 4	−1 … +1 (float)
L-Trigger (analog **lt**)	Axis 4	Axis 2	−1 … +1 (float)

Controller input mappings differ between a desktop PC and the Raspberry Pi (see Table 5). During development the code is occasionally run on a Windows computer for rapid debugging, but the production system will execute exclusively on Raspberry Pi. To accommodate the two environments, we keep two programs, *controller_pc.py* and *controller_pi.py,* and simply manually rename the appropriate file to *controller.py* whenever we switch platforms. Because this change is infrequent, adding automatic detection or run-time remapping was determined to be redundant.

Both analogue sticks report *0.0* at rest and vary smoothly to *+1.0 / −1.0* at full tilt; the pressure-sensitive triggers operate over the same numeric range but start at *−1.0* when fully released. A small dead-zone (±0.1) is applied in software to suppress jitter. The controller is polled every 50 ms, a value that gives snappy response without increasing CPU load or raising the Pi's temperature.

6.2.*5* User Interface and Interactive Game

The file *main_menu.py* implements a lightweight, full-screen front end written in PyQt 5 (QtWidgets). Figure 128 shows the layout; the key design choices are summarized below.

- **Navigation logic:** A single 10 ms timer polls the game-pad state. ↑/↓ move select focus and × activate current focus, while ◯ (Back) always returns to the previous screen or aborts the running subprocess. The menu works with mouse by default and did not need any adjustment.
- **Window & assets:** The menu window is opened in fullscreen to warp fir any screen, including the touchscreen on the Raspberry Pi enclosure and uses a QSS theme (*theme.qss*) to keep the visual style separate from logic. The project logo and tutorial graphic are ordinary PNG Photos, making it easy to update the artwork without touching code.
- **Buttons:** The button layout is depicted in Figure 128. in the depiction, scoreboard is empty, but the button will update itself to include the top three players with their initials.
- **Busy-state guard:** While a process or modal dialog is open, *_set_busy(True)* disables the other buttons. This prevents double-launches and keeps controller inputs predictable.
- **Score handling:** Game scores printed, so *mein_menu.py* can pick it up. The menu parses the first integer it finds, pops up an *InitialsDialog*, then writes the entry to a JSON file (*scores.json*). Because both the dialog and the scoreboard are rendered with the same font and timer-based poll loop, the entire user-interface remains consistent.

The primary goal is to showcase the robot rather than provide a fully featured video game experience. We did this with an interactive basketball game, asking players to score as many baskets as possible in a 60-second round. All game logic runs in *game_score.py*. An infrared break-beam sensor mounted inside the hoop generates a low-to-high signal each time the ball passes through; the script captures this transition and counts the goal. A celebratory GIF is randomly chosen from the folder (*goal gifs*) and displayed for instant feedback. When the remaining time falls to 10s or less the digits turn red, alerting the player. At time-out, the program verifies that the score is not zero; if so, the player can enter three initials using the D-pad, shown in Figure 129. The score and initials are stored in *scores.json*. The top three are continuously rendered on the main menu button to spark friendly competition.

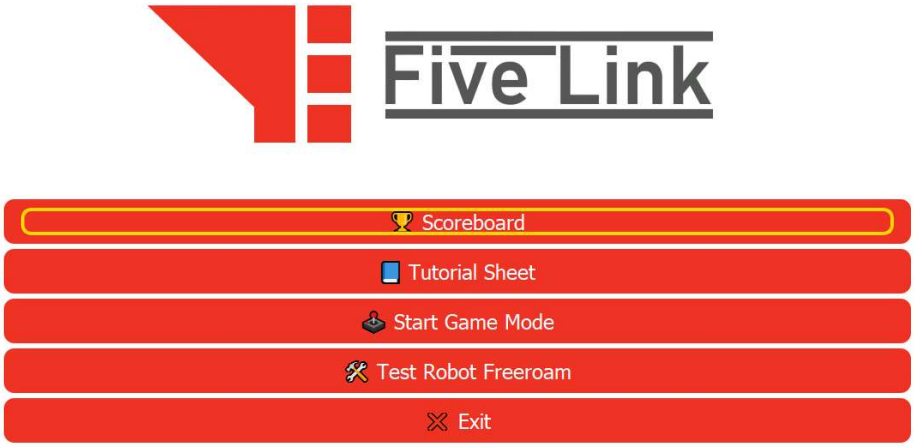


Figure 128 Menu button layout, with yellow outline being current focus selection

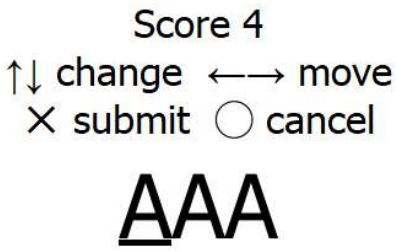


Figure 129 Initials score menu, where players can save their score like an arcade game

7. Performance evaluation

7.1 Motor evaluation

The testing setup consists of the whole robot put together as shown in Figure [130]. The testing will consist of measuring the time it took for both lead screw systems to make its full range of movement, while the X and Z axis will only require a visual check to see if it does what it is required to do: To pick up small objects, move the gripper into the air and release. Figure [130] shows the test setup.

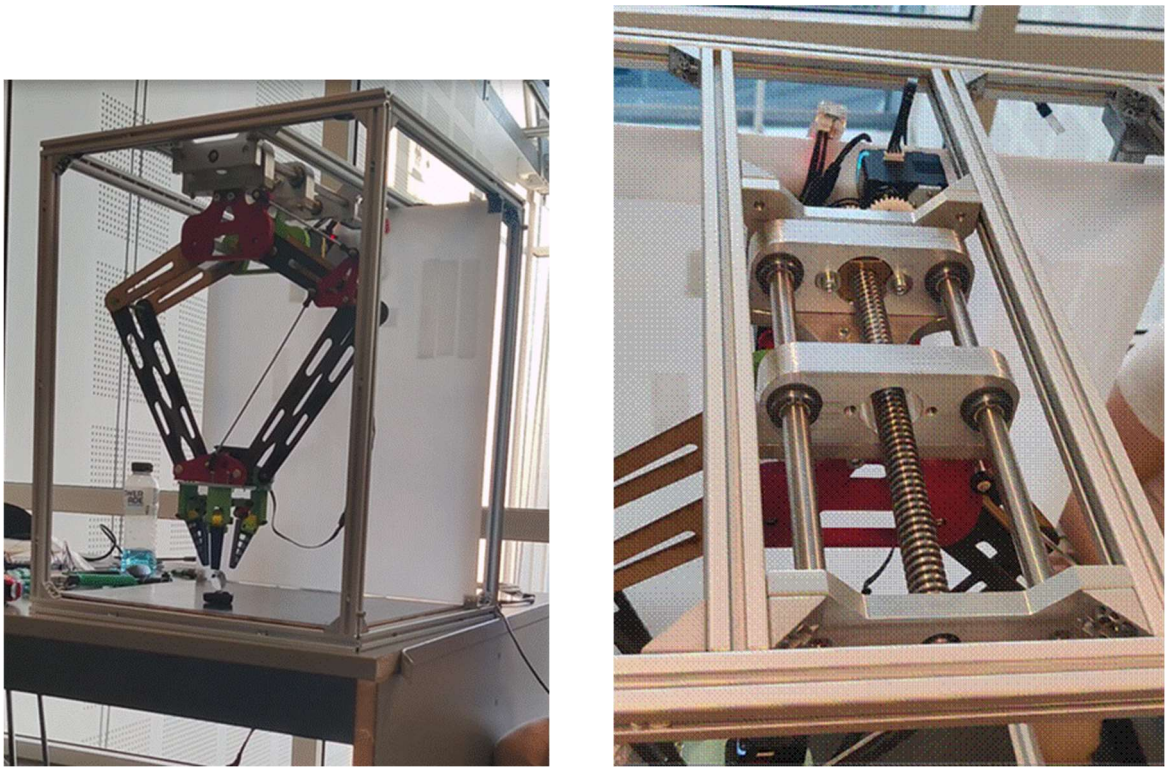


Figure 130 SHOWS THE TESTING SETUP FOR THE MOTORS. THE LEFTMOST PICTURE SHOWS AN OVERVIEW OF THE ROBOT, WHILE THE RIGHTMOST PICTURE SHOWS THE RAILING.

The first test was performed to measure the time to travel for the whole distance of the railing. Firstly, the robot was turned on, then a camera was pointed at the railing as shown in figure [130]. The camera was then turned on and the footage was later used to measure the time for the whole travel length of the railing. Then, the controller is used to move the railing to the start position as shown in figure [130]. The last step is to make the railing move from one side to the other, and then back again to its original position. After repeating this process 3 times, the total travel time was recorded at 10 seconds to move the railing forward and another 10 seconds to move it back to its original position.

The second test was performed on the gripper, this time to measure the time to fully close and open the gripper. The steps in this test are done quite similarly to the first test. Firstly, the robot was turned on and a camera placed and pointed in a position to record the opening and closing sequence from a close-up perspective as shown in figure [131]. Then, the camera is turned on and lastly the controller is used to make the gripper go from a completely open position to closed position as it grabs a object as shown in figure [131]. After repeating this 3 times, the result was that it takes 7 seconds to close the gripper around the object, and another 7 seconds to open the gripper again.

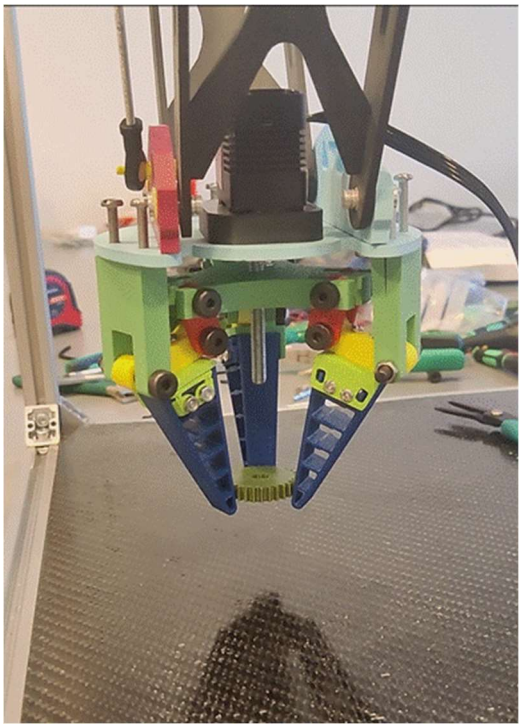


Figure 131 SHOWS THE GRIPPER TESTING SETUP

The last test is done by simply turning the robot on and then, with the help of the controller, move the arms up and to both sides while it is holding an object. This is done to see if the Dynamixel motors have enough torque to perform these actions. The robot was able to perform this test without any visible issues.

7.2 Hardware testing

To confirm whether the final hardware design was a successful implementation within the desired operating limits provided by the manufacturers, we conducted a series of tests. Several temperature tests were performed on each servo motor and the Raspberry Pi over a specific time interval, focusing on temperature and the risk of overheating. These tests were carried out individually for each motor under a controlled load. The applied load was substantially greater than what the motors would experience in the final system configuration. Therefore, the results from these tests are assumed to reflect safe operating behavior and confirm that the components will withstand the demands of the final product.

7.2.1 Raspberry Pi

The Raspberry Pi was the only component tested for heat while the full system was intact and operating. The system was kept on for 10 minutes with all components powered and functioning at a normal user rate, meaning the robot was being used as it typically would be, by performing actions such as picking up an apple or a table tennis ball. During this 10-minute period, we collected the temperature reading from the Raspberry terminal.’

Table 5 Temperatures measured every 2 minutes for a total of 10 minutes withing the terminal by a fixed command.

Total time: 10 minutes	Temperatures in Celsius
2 min	44.3
4 min	45.7
6 min	46.2
8 min	46.2
10 min	47.2

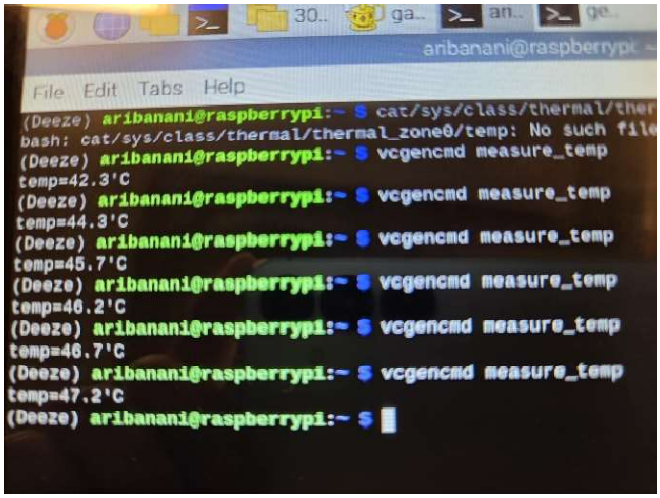


Figure 132 The terminal in the Raspberry Pi recording started from 44.3 Celsius.

7.2.2 Dynamixel Motor Testing

Both Dynamixel motors were tested individually using their own firmware. Since the motors can be connected either to the Raspberry Pi or to the firmware on a laptop, it was not possible to collect temperature data from the firmware while the motors were running within the fully assembled system. To solve this issue, we decided to test each Dynamixel motor individually by connecting them to a laptop using a USB cable and their driver. This allowed us to gather the temperatures during operation.

The motor was tested for 1 minute. It was kept in the positions where it would be placed as in the final setup and was used to lift the gripper with a 0.5-liter bottle filled with water, repeatedly up and down in a vertical motion at velocity mode within the firmware. The results were not the desired expectations.

When lifting the arm with the bottle the motor would go in to stall mode and blink a red light. Lifting the bottle with one of the Dynamixel motor was tested 3 times, and the same result would appear. Therefore, we concluded a 0.5-liter bottle of water was too much for one dynamixel and instead tested the motor by lifting the arm without any added weight, only the arm and the gripper intact.

Figure 133 leftmost picture shows motor in stall mode during the first test while the rightmost picture shows a 0.5liter water bottle during the first test before stall mode

After 1 minute without any added weight, the recorded motor temperatures for both motors were stable at 53 degrees Celsius at velocity mode where both motors were controlled manually. According to the manufacturer, the ideal operating temperature is from -5 to 80 degrees Celsius, therefore the dynamixel will unlikely cause any overheating when the robot is operated at a high demand (Robotis, n.d).

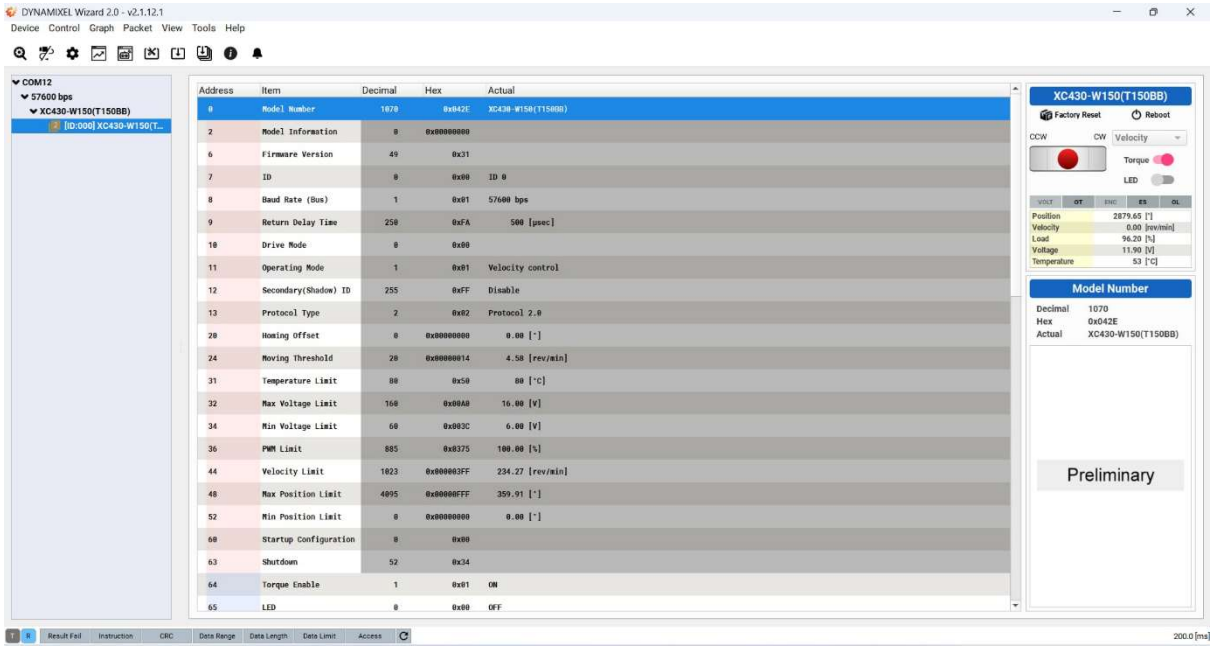


Figure 134 Dynamixel Wizard 2.0, the firmware for the first dynamixel motor at 53 degrees Celsius seen at the top right corner of the figure.

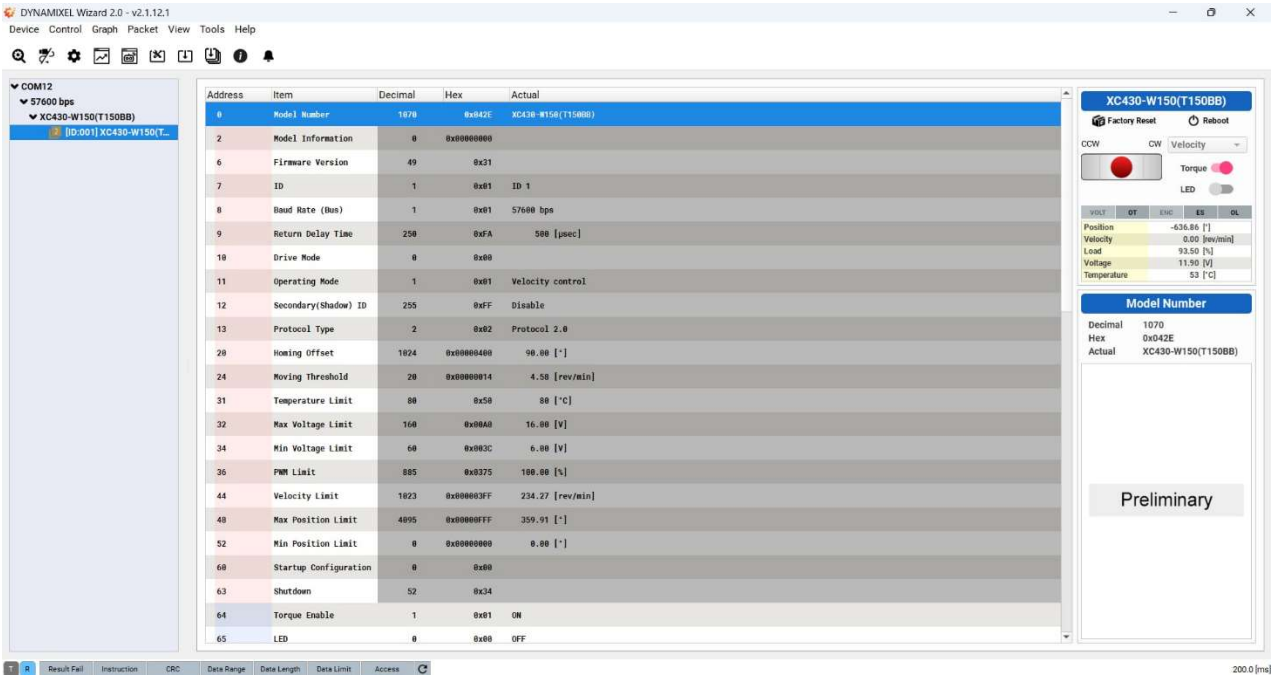
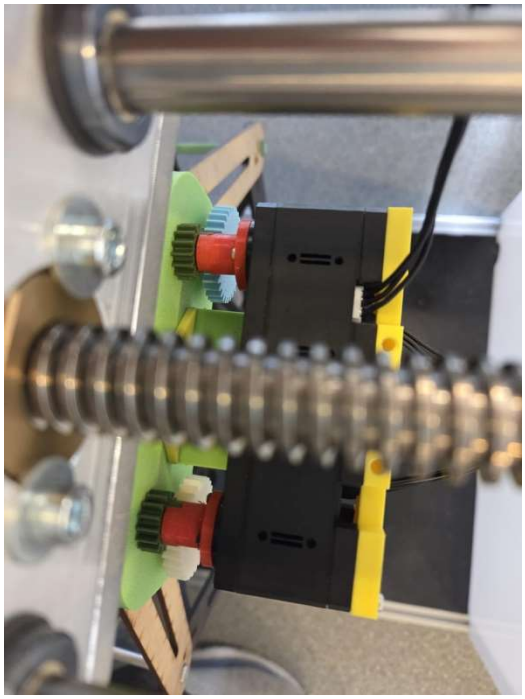


Figure 135 Dynamixel Wizard 2.0, the firmware for the second dynamixel motor at 53 degrees Celsius seen at the top right corner of the figure.



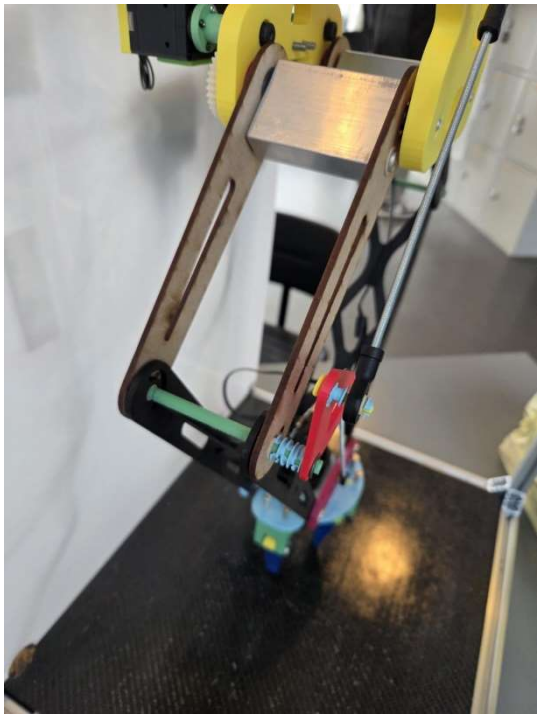


Figure 136 leftmost picture shows the placement of each motor during individual testing while the rightmost picture shows the motor tested without added weight, only the intact system

Figure 137 leftmost picture shows the gear after lifting with 0.5 liter water bottle, rightmost picture shows the gear after lifting with no load, only the system intact

7.2.3 LSS HS1 Motor Railing Testing

The same issue encountered with the Dynamixel motors also occurred with the HS1 motors where data could not be measured because the motors cannot be connected to both the Raspberry Pi and the laptop running the firmware simultaneously. As a result, the motor responsible for moving the railing was tested for overheating by connecting it individually to the laptop through the firmware.

Durin the test, the motor was rotated using the firmware at 1-minute intervals from one end of the railing to the other, with a 0.5-liter bottle attached to the gripper. The firmware consistently displayed a temperature of approximately 37 degrees Celsius, and a slight voltage drop. According to the manufacturer, the recommended operating temperature range is between 45°C and 75°C (Nantel, 2024). Therefore, under normal usage, overheating of the HS1 railing motor is considered unlikely.

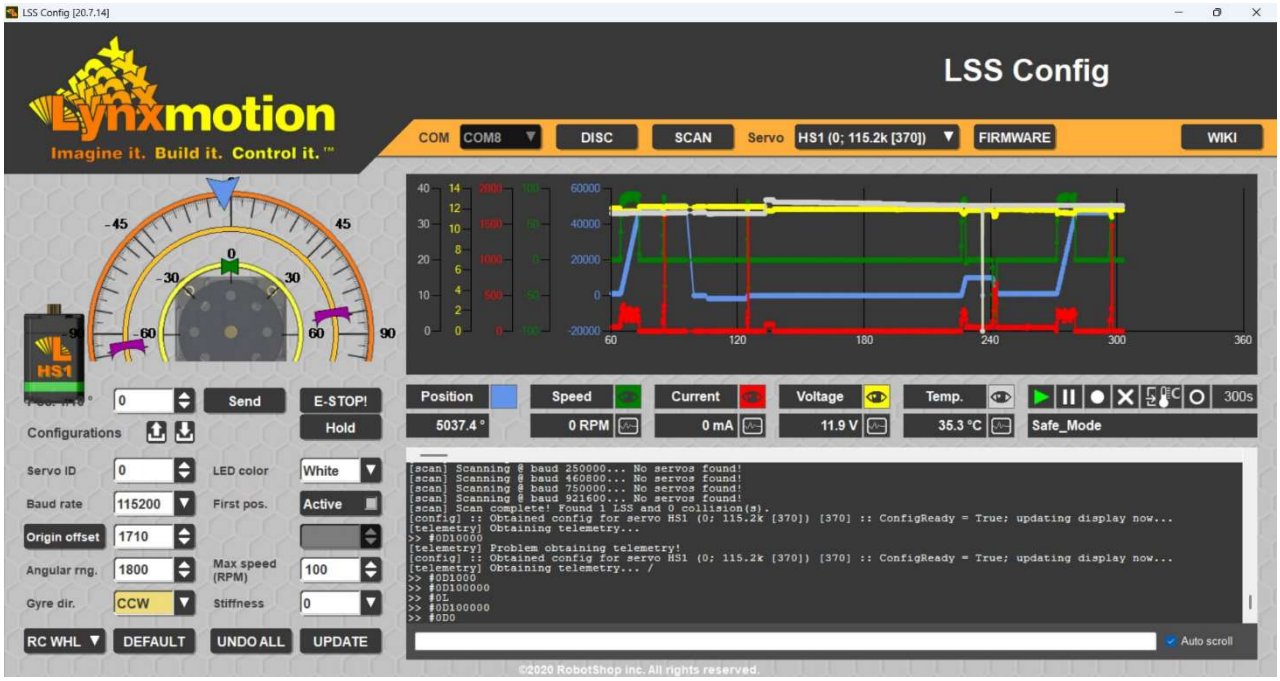


Figure 138 LSS Config, the firmware for gathering temperature from the HS1 Railing motor.

Figure 139 the leftmost picture shows placement of a HS1 motor over the controlling of the railing with a trapezoidal lead screw, while the rightmost picture shows a 0.5liter bottle with water on top of the gripper when testing the railing.

7.3 Software Performance Overview

Because of a compressed schedule and several late-stage gear failures, the team could not run enough tests in time to thoroughly test out the capabilities of the software interactions with hardware. We were fortunate to have a few number of willing participants that gave feedback and an insight as to how new users tackled the controls. Hard numbers are still missing; we state that fact plainly and flag it for future work.

7.3.1 Responsiveness

- **IK loop:** Runs every 50 ms without visible jitter. Even on a Raspberry Pi 4, the CPU temperature stays in the low 50 °C range, suggesting ample headroom.

- **End-to-end latency:** Thumb-stick motions translate to gripper movement instantly to the human eye; no operator has noticed lag during play-testing. (Oscilloscope verification still pending.)
- **GUI frame rate:** The PyQtGraph plot navigates smoothly to the screen; no stutter or tearing is observed.
- **Controller ergonomics:** All buttons feel natural except the gripper, which behaves awkwardly and unresponsive. An alternative has been drafted but not implemented. (The idea is for the left trigger to close and the left trigger button to open)

7.3.2 Robustness & Safety

- **Boundary clamp**: Early desynchronization bugs were eliminated by adding an isnan() guard and clamping targets to the workspace.
- **Dynamixel wrap-around**: Angle jumps from 4095 → 0 produced a full 360° sweeps. Limiting rotation to < 180° removed the risk without hurting reach.
- **Full-retraction glitch**: The gripper occasionally contracts completely for unknown reasons; suspected poll rate or timeout side-effect. Needs log-driven diagnosis.
- **Memory hygiene**: Animated GIF are now freed immediately after use, stopping the memory consumption that plagued an early build.
- **Break-beam sensor**: The refresh rate has been increased for more reliable detection; the hoop fixture still needs to be mounted and tested in an authentic environment.

7.3.3 Resource Use & Thermals

- Ten-minute mixed idle/gameplay run puts the Pi CPU at ~53 °C, well below the 80 °C throttling point.
- Formal CPU-% and RAM logs have not been captured, but nothing hints at resource exhaustion during use.

7.3.4 Accuracy & Simulation Fidelity

- Visually, the physical end-effector tracks the simulated end-effector.
- The CAD model omits one passive link, so workspace limits carry a small safety margin.
- No hard physical vs simulation measurements have been taken yet; true accuracy remains *unknown*.

7.3.5 General Code Assessment

The project reached its Minimum Viable Product stage without undergoing a thorough cleanup. Redundant functions and sparse comments remain.

7.3.6 Next Steps for software

1. **Instrument what we can't see**: IK loop; logs for CPU/RAM/temp; oscilloscope for latency.
2. **Stress the limits**: tighten poll period toward 10 ms, watch for renewed input loss and thermal rise.
3. **Measure accuracy**: collect encoder data for ≥ 10 poses; reconcile with the simplified CAD model.
4. **Polish the code**: remove redundancy, add doc-strings, run pylint and unit tests under CI.
5. **Finish the hoop rig**: validate break-beam timing with real eviorment.
6. **Refine user experience**: possibly remap trigger button or increas poll rate, expose parameters (rail step, servo speed, XY resolution) for on-site tuning.

Bottom line: Despite thin evidence, the software feels crisp, Raspberry Pi keeps cool and shows no obvious failure during demos. The remaining work is measurement, not firefighting.

8. Discussion Of Results

8.1 Motor evaluation discussion

Railing

The result of the motor testing differed for the different parts of the system. Firstly, the railing moved slower than expected. The expected time to travel went up from 6 to 10 seconds. 6 seconds was the number acquired from testing the railing system at the start of the project period, with an additional mass of 6kg to simulate a system that requires more torque. So, when we tested the railing on the final product this increase in time was quite unexpected. This increase in time could have been because of wear and tear of the motor or other mechanical parts of the system, or maybe the software was simply not making the motor go at full power, since the original test was done with a different software then what was used on the final version of the robot. The two tests could also have received different results because of the mounting methods during the two tests. As seen in figure [95] the mounting method for the motor was to simply hold the motor so that the two gears would press against each other, while in the final product as seen in figure [130] the mounting was done without the need of a human being near the motor.

Gripper and Arms

The results from the grippers perspective were well within expectations when it comes down to the total time to open and close the gripper. However, it is important to keep in mind that the test was performed with a 1mm pitch M6 screw instead of the Ds5x5 lead screw with a pitch of 5mm, which in theory should make the lead screw system go 5 times faster without a problem. However, because of time limitations it was not possible to acquire that lead screw in time for this project, meaning that if there are mistakes within the theory part of this report, then that could have significant impact on future results of the gripper. Also, because parts of the gripper are not made out of metal, like the gripper is designed to be, then that could also have significant impacts on future results.

The motors for the X and Z axis of the arms also worked well, as they could be used when picking up small objects while moving them around. However, like the gripper, because the planned materials have not yet been implemented, future results may vary if there are wrongdoings in the theory of this report or if some mechanical issues appear if the parts are implemented with the originally planned materials.

8.2 Hardware performance discussion

The main focus of the tests was to monitor temperature and check for any signs of overheating in the motors and the Raspberry Pi. Based on the guidelines from each manufacturer, none of the tested components showed signs of overheating during the test periods, and all performed reliably within their specified limits. However, there were some issues during the testing of the hardware components which might have been critical factors affecting the test results.

Dynamixel Motors

When testing the Dynamixel motors each individually with a 0.5-liter bottle filled with water loaded on the gripper, the motors stalled. This is highly likely because of the issues experienced with the gears. The gears at that moment were made of PLA material from a 3D printer. During the test, teeth of the gears would eventually break. By observation, one could

see that when rotational force was applied through the firmware, the motor was not able to continue its rotation because at some point on the gear, there would not be a step for it to move on. This would lead to the motor being forced to rotate when there was no longer a path for it to move on.

After realizing that a 0.5-liter bottle filled with water was too heavy for the motors, another similar test was conducted using newly installed gears and this time without any load, but similar issues still occurred. The teeth of the new gears eventually broke, preventing continuous rotation. However, this time the Dynamixel motors were able to jump over the missing teeth, likely due to the reduced weight compared to the previous test with a bottle. Therefore, the test under this circumstance able to measure a temperature after an interval of 1 minute was recorded in the firmware.

The second point to mention is that the Dynamixel motors each lifted the robot vertically, individually. The motors were never intended by the mechanical department to operate under such conditions. The purpose of this test was only to verify whether the motors could operate at reasonable temperatures when exposed to highly unlikely loading scenarios. In an ideal setting, the load would be distributed between both Dynamixel motors, meaning that the temperature would likely not increase significantly, even when the system is used over longer intervals. In addition, it is important to note that the entire system was not tested as a whole, but rather the performance of each individual servo motor was tested separately. This was due to the limitation of not being able to connect the servo motors' USB data cables to both the Raspberry Pi and the firmware software simultaneously during temperature analysis.

Raspberry Pi

The Raspberry Pi, which contains the system instructions, was also tested individually. However, unlike the motors, the Pi was tested while the entire system was intact. This was possible because the temperature could be monitored directly through the terminal, which could run simultaneously while the system was in operation.

The observed temperatures after the testing interval were well within safe limits, staying far below the Pi's overheating limit before the CPU throttles. While the Pi test can be regarded as successful, the results were more or less expected. This is mainly because the Pi does not power the motors, it only supplies the display and the sensor, both of which draw relatively low current and the connection to the servo motors is strictly for data transfer.

Railing motor

There were no issues observed during the testing of the railing motor. The gears remained in place and intact throughout the entire test, even when lifting a 0.5-liter bottle, and the results were acceptable. As explained in Section 3.4, the railing operates at lower power compared to the Dynamixel motors, due to the specific gear ratios implemented for each motor. This may have resulted in less mechanical stress on the railing system during testing.

Gear

The gear failures during testing became a significant factor when evaluating how much stress a Dynamixel servo motor could handle during repeated vertical movements over a 1-minute interval. Although the motors operated within the manufacturer's maximum temperature limits, the lack of proper grip due to broken gear teeth prevented smooth motion. This inconsistency may have caused either reduced or elevated temperatures, depending on the level of strain at each rotation.

Both the railing motor and the Raspberry Pi delivered reasonable results, with no complications and values well within expected ranges throughout the testing process. While

the Dynamixel motors did provide useful temperature data, the testing conditions were not ideal due to the gear limitations.

However, what was crucial from the tests was how vulnerable the plastic gears attached to the Dynamixel motors were. Because of time constraints, the ideal gear material could not be delivered during the project period. With stronger gear materials, smoother motor rotations would likely have been achieved and stalling by missing steps would have been less likely even over extended use.

8.3 Software reflections

Gripper control is the primary limitation of the current software. Its behavior is unintuitive for first-time users and is not reliable. Experienced operators can compensate by moving gently, but the gripper remains the system's most pronounced usability weakness. Two deliverables slipped past the deadline and remain in progress: printing and installation of the basketball hoop needed for the interactive game, and comprehensive code annotation to assist future project teams. Quantitative performance data is still scarce owing to the compressed schedule and will be discussed further in the coming section 8.4.

8.4 Further Work

Further improvements can definitely be made to this project with more time, as there are plenty of things that the team could have done differently, especially with the new knowledge received from the testing of the final product. For example, the gears could be improved in many ways. There could be less amount of teeth in the gears, making the teeth thicker which in return makes each individual tooth stronger. Testing different gear ratios could also change the size of the individual teeth, which could help in making the teeth stronger, while also perhaps increasing motor efficiency. The gears could be printed with different settings or filaments and multiple other types of gears have yet to be considered.

Because of time limitations, the team could not get all the parts in time with the correct materials and lead screw specifications. To improve the reliability of the results, the product can be made with the correct materials. This could improve friction in rotational areas, as the robot commonly uses bushing to lessen the friction, but that bushing may work best against metal rather than plastic. We should also verify that the motors are being run at full capacity, while investigating wear and tear of the motors, to reduce future performance issues.

Another important aspect of the project was the Dynamixel servo motors. Due to gear limitations, the Dynamixel motors were never tested under ideal conditions. Proper testing could have revealed more about the temperatures of each Dynamixel motor, potentially providing useful data such as the approximate temperature of each motor. This would then help to evaluate the status of the system to create a solution if needed.

While the final integrated system was overall a success and fulfilled the scope outlined in the project assignment, there is room for improvement in how the system was physically assembled, particularly in terms of wiring and cable management. The soldered wires, while functional, introduce a higher risk of system failure. Replacing these with standardized wires of correct lengths and appropriate connectors could make the system more robust, reduce potential points of failure, and simplify future maintenance and upgrades.

One important area for further development is simplifying the assembly process of the mechanical components. During testing, this proved to be a recurring problem, as several parts, particularly in the Festehub and gripper assemblies, were difficult to assemble and required significant time and effort. At times, the complexity of the assembly consumed a

large portion of the available working time. On some occasions, parts even had to be reprinted with adjusted dimensions to make assembly easier.

Future software development could begin with rigorous quantitative instrumentation that transforms subjective impressions into measurable benchmarks. Execution-time profiling, resource logging and latency traces could expose hidden bottlenecks and establish a performance budget. Building on that data, the control-loop period could be adjusted stepwise below the current 50ms while monitoring processor load, and temperature, thereby stress-testing the real-time limits of the existing hardware and its capacity for future features.

Accuracy, maintainability, and user experience could also undergo further testing. Encoder traces collected across the workspace could be compared with simulated poses; errors beyond ±5 mm would prompt kinematic refinement or revised safety margins. Additionally, redundant code could be addressed, and code documentation could be expanded to assist future project teams. Completing the basketball-hoop rig would enable validation of the break-beam sensor under realistic conditions. At the same time, refined controller mapping and tunable parameters such as rail step and servo speed would enhance usability.

9. Conclusion

This bachelor project has attempted to further develop an exhibition friendly robotic arm that fulfilled requirements given by Tronrud Engineering. When we take a broader look at the results of the project, a lot of improvements have been made, but there are still some ways to go. The robotic arm can now move in all three axes, while it also has a functioning gripper. The robot can be controlled wirelessly and mostly intuitively with the help of inverse kinematics. However, the reliability of the robot has not yet been proven, as the gears sometimes break which is not ideal when the robot is being played with at exhibitions.

The railing and gripper of the current robot are also slower than we wished for, which could impact the enjoyment of the robotic arm when used by the audience at exhibitions. While we know that the gripper is relatively slow due to a lower pitch screw being used than what was originally wished for, we are not sure quite sure why the railing is slower than expected. But, even if we could not receive the correct materials in time for the final deadline of this report, we have still made a functioning robot with mostly intuitive controls and a functioning gripper to pick up objects.

REFERENCES

101, C. (2021, July 16, 2021). *HC-05 - Bluetooth Module*. Retrieved May 20, 2025 from https://components101.com/wireless/hc-05-bluetooth-module

Arduino. (2025a). *Arduino® Mega 2560 Rev3*. https://docs.arduino.cc/resources/datasheets/A000067-datasheet.pdf

Arduino. (2025b). *Arduino® Nano*. https://docs.arduino.cc/resources/datasheets/A000005-datasheet.pdf

Arduino. (2025c). *Arduino® Nano 33 BLE*. Arduino. https://docs.arduino.cc/resources/datasheets/ABX00030-datasheet.pdf

Autodesk. (2025). *Autodesk Inventor: 3D-modelleringsprogramvare for designere og ingeniører*. Autodesk. Retrieved 18 may from https://www.autodesk.com/no/products/inventor/overview

Bemis, C. (2022, 1. august). Keyseats and Keyways. https://www.gdandtbasics.com/keyseats-and-keyways/

Budynas, R. G., & Nisbett, J. K. (2020). *SHIGLEY'S MECHANICAL ENGINEERING DESIGN* (E. Edition, Ed.). McGraw-Hill Education.

Crooks, W., Vukasin, G., Vukasin, G., O'Sullivan, M., Messner, W., & Rogers, C. (2016). Fin Ray® Effect Inspired Soft Robotic Gripper: From the RoboSoft Grand Challenge toward Optimization. *Frontiers in Robotics and AI*, *Volume 3*. https://doi.org/10.3389/frobt.2016.00070

Electric, S. (n.d). *Kontaktelement med 1xNC kontakt og skruterminaler*. Retrieved May 17, 2025 from https://www.se.com/no/no/product/ZBE102/kontaktelement-med-1xnc-kontakt-og-skruterminaler/?parentSubCategoryId=89188

Elgeneidy, K., Fansa, A., Hussain, I., & Goher, K. (2020). Structural Optimization of Adaptive Soft Fin Ray Fingers with Variable Stiffening Capability. *IEEE Access*, 779-784. https://doi.org/10.1109/RoboSoft48309.2020.9115969

Festo. (n.d). *Applications*. Festo. Retrieved may 22 from https://www.festo.com/no/en/ap/overview/?q=~%3Acode-desc~%3AapplicationProducts~%3ADHAS&page=0&density=NORMAL

Finder, A. (n.d). *COMPARISON OF KINEMATIC MODELS*. Retrieved May 21, 2025 from https://autonoxfinder.com/en/Kinematics-comparison/

GeeksforGeeks. (2020, September 24, 2020). *PyQtGraph - Extensive Examples*. Retrieved May 22, 2025 from https://www.geeksforgeeks.org/pyqtgraph-extensive-examples/

GeeksforGeeks. (2022, December 19, 2022). *Python | Introduction to PyQt5*. Retrieved May 22, 2025 from https://www.geeksforgeeks.org/python-introduction-to-pyqt5/

GeeksforGeeks. (2025a, April 4, 2025). *Introduction to Python Raspberry Pi (RPiGPIO) Library*. https://www.geeksforgeeks.org/introduction-to-python-raspberry-pi-rpigpio-library/

GeeksforGeeks. (2025b, April 8, 2025). *PyGame Tutorial*. Retrieved May 22, 2025 from https://www.geeksforgeeks.org/pygame-tutorial/

GeeksforGeeks. (2025c, March 26, 2025). *Python NumPy*. Retrieved May 22, 2025 from https://www.geeksforgeeks.org/python-numpy/

HANDLING TOOL. *International Journal of Science & Technology*, *Volume 7*. https://www.researchgate.net/publication/317823110_Vacuum_Cup_Grippers_for_Material_Handling_In_Industry

Igus. (n.d). *Configure lead screw drives*. Retrieved May 18 from https://drylin-leadscrew-drives-expert.igus.tools/requirements

Kumar, B. (2017). VACUUM GRIPPER- AN IMPORTANT MATERIAL

KOLLMORGEN. (2020, September 29, 2020). *How Does a Servo Motor Work?* Retrieved May 18, 2025 from https://www.kollmorgen.com/en-us/blogs/how-servo-motors-work
KUKA. (n.d). *KR DELTA robot in Hygienic Design*. Retrieved May 21, 2025 from https://www.kuka.com/en-se/products/robotics-systems/industrial-robots/kr-delta-robot-hm
Ltd, R. P. (n.d-a). *Raspberry Pi Touch Display*. Retrieved May 4, 2025 from https://www.raspberrypi.com/documentation/accessories/display.html#raspberry-pi-touch-display
Ltd, R. P. (n.d). *Use Python on a Raspberry Pi*. Raspberry Pi Ltd. Retrieved April 14, 2025 from https://www.raspberrypi.com/documentation/computers/os.html#use-python-on-a-raspberry-pi
Ltd, R. P. T. (2024). *DATASHEET Raspberry Pi 4 Model B*. R. P. T. Ltd. https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf
Lab, B. (2025). *Bambu Studio*. Bambu Lab. Retrieved 18 may from https://bambulab.com/en-eu/download/studio
Le, T. D., Kang, H.-J., & Quang, V. D. (2017). A Method for Optimal Kinematic Design of Five-bar Planar Parallel Manipulators *Automation and Information Sciences* https://doi.org/10.1109/ICCAIS.2013.6720521
Lynxmotion. (2024). *08 - Software*. Lynxmotion. Retrieved 18 may from https://wiki.lynxmotion.com/info/wiki/lynxmotion/view/ses-v2/lynxmotion-smart-servo/lss-configuration-software/
Magnor, T., Lindbråten, A., Rasmussen, L. T., Overgaard, M., Kristiansen, A., & Gjerstad, S. (2024). *DuoArm* [Bachelor's thesis, University of south eastern norway, University of south eastern norway].
Matlab. (n.d). *Math. Graphics. Programming.* Retrieved May 22, 2025 from https://www.mathworks.com/products/matlab.html
Nantel, E. (2024a). *LSS Specifications*. Lynxmotion. https://wiki.lynxmotion.com/info/wiki/lynxmotion/view/ses-v2/lynxmotion-smart-servo/lss-specifications/
Nantel, E. (2024b, October 28, 2024). *Wiring*. Lynxmotion. https://wiki.lynxmotion.com/info/wiki/lynxmotion/view/servo-erector-set-system/ses-electronics/ses-wiring/
Nantel, E. (2024a, July 12, 2024). *LSS Electrical*. Lynxmotion. https://wiki.lynxmotion.com/info/wiki/lynxmotion/view/ses-v2/lynxmotion-smart-servo/lss-electrical/
Nantel, E. (n.d). *LSS-ADA Board (USB Mini)*. Lynmotion. Retrieved April 15, 2025 from https://wiki.lynxmotion.com/info/wiki/lynxmotion/view/servo-erector-set-system/ses-electronics/ses-modules/lss-adapter-board/
Nantel, E. (n.d-b). *LSS Libraries & Examples*. Retrieved May 22, 2025 from https://wiki.lynxmotion.com/info/wiki/lynxmotion/view/ses-v2/lynxmotion-smart-servo/lss-libraries/
Navas, E., Rodríguez-Nieto, D., Rodríguez-González, A. A., & Fernández, R. (2025). Parallel Fin Ray Soft Gripper with Embedded Mechano-Optical Force Sensor. *Applied sciences*, *Volume 15*. https://doi.org/10.3390/app15052576
PANDILOV, Z., & DUKOVSKI, V. (2012). *PARALLEL KINEMATICS MACHINE TOOLS: OVERVIEW- FROM HISTORY TO THE FUTURE*. https://annals.fih.upt.ro/pdf-full/2012/ANNALS-2012-2-16.pdf
Pi, R. (n.d). *Raspberry Pi Touch Display*. Raspberry Pi. Retrieved May 8, 2025 from https://www.raspberrypi.com/documentation/accessories/display.html
Python. (n.d). *Documentation*. Retrieved May 22, 2025 from https://www.python.org/doc/
RoboShop. *IR Break Beam Sensor (50cm)*. RoboShop. Retrieved April 15, 2025 from https://eu.robotshop.com/products/ir-break-beam-sensor-50cm?qd=78e9350b47962edb74116b611cc1e622

Robotis. (2025). *The Ultimate DYNAMIXEL Daisy Chaining Guide*. https://www.robotis.us/robotis-ir-pr-blog/the-ultimate-dynamixel-daisy-chaining-guide/
Robotis. (n.d-a). *U2D2*. Robotis. Retrieved April 18, 2025 from https://emanual.robotis.com/docs/en/parts/interface/u2d2/#introduction
Robotis. (n.d-b). *U2D2 Power Hub*. Robotis. Retrieved April 15, 2025 from https://emanual.robotis.com/docs/en/parts/interface/u2d2_power_hub/
Robotis. (n.d-c). *XC430-W150*. Robotis. Retrieved April 14, 2025 from https://emanual.robotis.com/docs/en/dxl/x/xc430-w150/?_gl=1*3srss4*_gcl_au*MTE2NzU0MDQwOC4xNzM4NzY3NjQ3
ROS. (n.d). *ROS - Robot Operating System*. Retrieved May 22, 2025 from https://www.ros.org/
Rosmo, R. (2023, June 2, 2023). *Sensorer og aktuatorer*. NDLA. Retrieved May 18, 2025 from https://ndla.no/r/konstruksjons--og-styringsteknikk-tp-tip-vg1/sensorer-og-aktuatorer/d91bfebade
SONY. (n.d). *DUALSHOCK 4 Wireless Controller*. Retrieved May 20, 2025 from https://www.playstation.com/en-us/accessories/dualshock-4-wireless-controller/?smcid=pdc%3Aen-us%3Aaccessories%3Aprimary%20nav%3Amsg-ps4%3Acontrollers
Shintake, J., Cacucciolo, V., Floreano, D., & Shea, H. (2018). Soft Robotic Grippers. *ADVANCED MATERIALS*, *Volume 30*. https://doi.org/10.1002/adma.201707035
Sun, Y., Liu, Y., Pancheri, F., & Lueth, T. C. (2022). LARG: A Lightweight Robotic Gripper With 3-D Topology Optimized Adaptive Fingers. *IEEE/ASME Transactions on Mechatronics*, *27*, 2026-2034. https://doi.org/10.1109/TMECH.2022.3170800
Well, M. (n.d). *85W Dual Output Switching Power Supply*. Retrieved May 17, 2025 from https://no.mouser.com/datasheet/2/260/RD_85_SPEC-1291492.pdf
Young, H. D., & Freedman, R. A. (2020). *University Physics with Modern Physics* (FIFTEENTH EDITION IN SI UNITS ed.). Pearson Education Limited.

Footnote - AI

The authors acknowledge the use of OpenAI ChatGPT (May 2025 model) to refine phrasing, clarify sections of the report, and suggest code-level improvements during development; all resulting text and software were reviewed, tested, and approved by the authors.

OpenAI. (2025, May 22). ChatGPT (May 2025 version) [Large language model]. Retrieved May 23, 2025, from https://chat.openai.com/

Appendix A – List of Figures

Figure number	Caption	Source/Origin	Page
Figure 139	the leftmost picture shows placement of a HS1 motor over the controlling of the railing with a trapezoidal lead screw, while the	Authors	105