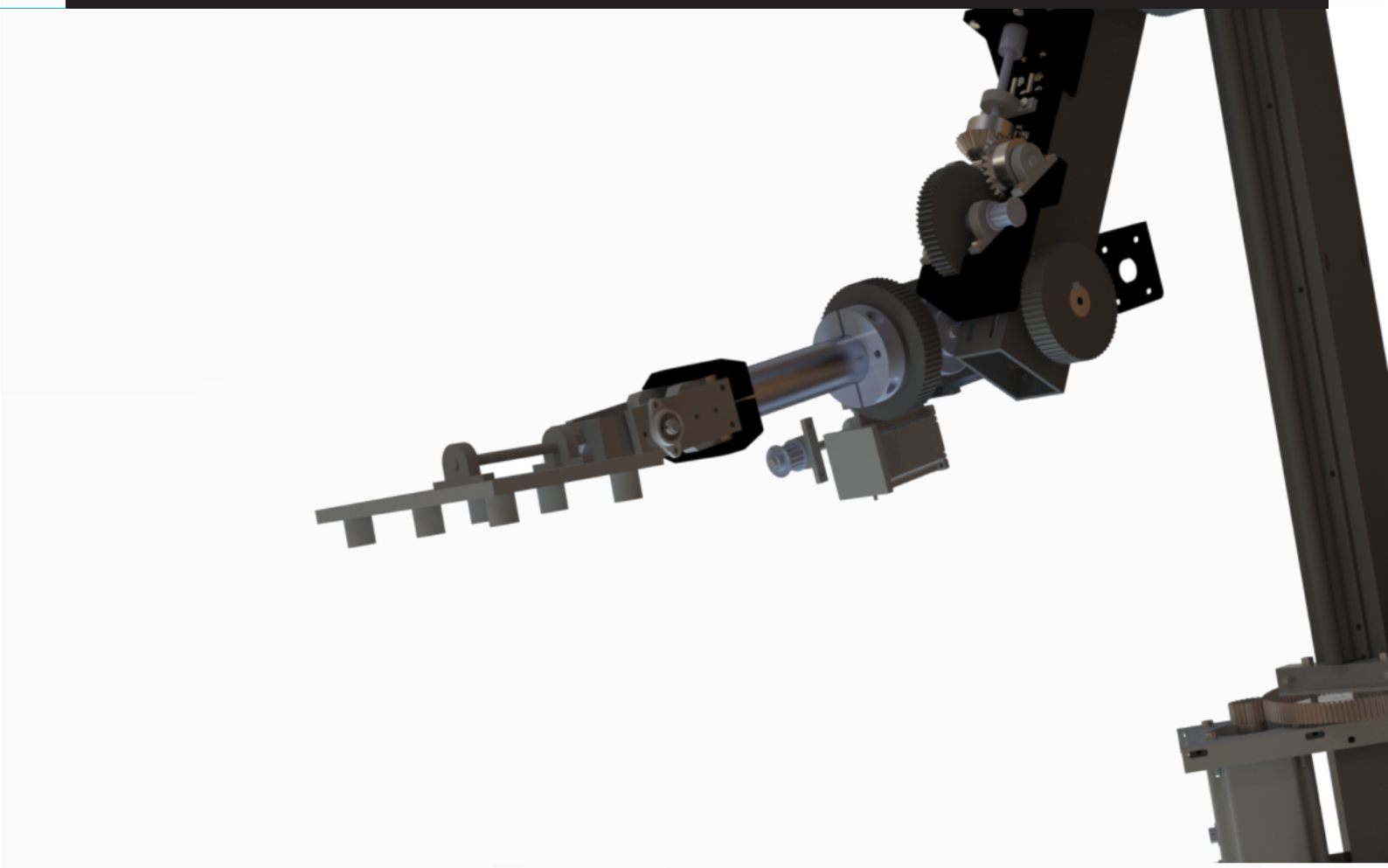
Handover Document

Team 7 –Vanderlande

Bălţăreţu, Ana
Boekestijn, Lotte
Nieuwenhuizen, Dominique
Noordermeer, Mark
Stuijt Giacaman, Willem
Zuidema, Bart



Tetris

PACKING FOR THE FUTURE

TU Delft

Handover Document

by

Team 7 –Vanderlande

To document the current state and future recommendations for the Parcel ULD Loading project

Team members:	
Ana Băltăreţu	Software
Lotte Boekestijn	Design
Dominique Nieuwenhuizen	Mechanics
Mark Noordermeer	Mechanics
Willem Stuijt Giacaman	Software
Bart Zuidema	Electronics

January 2022

Team:	7 (Vanderlande)
Project duration:	August 31, 2021 - January 28, 2022
Minor coordinator:	Martin Klomp
Teaching Assistant:	Jeffrey Pronk
Client:	Hans Steijaert (System Architect at Vanderlande)
	Stijn de Looijer (Technology Owner Robotics at Vanderlande)
Festo contact:	Menno Goedhart (Project Engineer at Festo)

TU Delft

Contents

1

Executive Summary

This handover document contains all relevant information about the robotic arm designed and implemented as an assignment for the company Vanderlande. In short, the assignment is to design and implement a system which is able to automatically stack parcels into an airplane container, a so called Unified Loading Device (ULD). Currently this is done manually by an operator because existing robots are still not comparable to an operator when looking at tight stacking, throughput of the parcels/hour and handling of the variety of different parcels. A wide range of different ULD's is used in the airplane industry. However, for this project only one ULD is considered. This particular type of ULD is shown in Figure 1.1.

Figure 1.1: *ULD considered for this project, a type A2N container [1].*

As can be seen in Figure 1.1, the ULD has a relatively small door which makes it difficult for a robotic arm to reach all corners. Furthermore, when stacking all the parcels in the container, a large robotic arm will easily interfere with itself, or bump into previous stacked parcels. The client asks for a robotic solution to this problem which has characteristics like high speed, high accuracy, high stiffness, low mass, advanced motion control and intelligent software. In addition, the robotic system should be able to stack 400 parcels per hour.

Tetris, a parcel loading robotic prototype is the start of the journey to solve this problem. The robotic solution designed and implemented during this project is a 7 degrees of freedom robot arm with a suction cup gripper. The entire robot is made from scratch and all parts are customized for this robot. The main idea for this solution is that the entire robotic arm is moved into the ULD with an extending rail instead of using an extremely large robotic arm that is located outside of the ULD. As a result the robotic arm does not have to reach far from itself such that the robot can be more compact and light weighted. However, for this project the extending rail system is not considered so the robot has a fixed position inside the ULD. The model of the 7 degrees of freedom arm built for this project is shown in Figure 1.2.

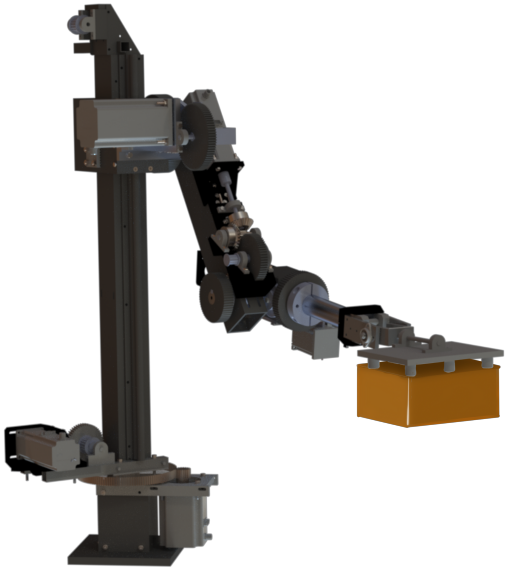


Figure 1.2: *Model for the robot built for this project.*

The results are very promising. Due to the 7 degrees of freedom and the unique ability to turn past itself at both the shoulder and elbow joint the robot has an extremely high manoeuvrability. Almost all joint have the ability to turn 360 degrees. That is, from the initial position 180 degrees clockwise and 180 degrees anti clockwise. This improves the manoeuvrability and possibilities to find complex paths from the initial position to the final position even more. Furthermore, the robot arm is capable of sliding along the pole as a whole. This way a shorter robot arm is sufficient to reach the upper corners of the ULD compared to a fixed 6 degrees of freedom arm on the ground. Additionally, the elbow joint can work both below and above relative to the shoulder joint. Figure 1.2 shows an example of the elbow joint working below the shoulder joint. With most existing industrial robots this is not possible. This can be an advantage when stacking the upper layers of the ULD.

Currently the robot is capable of gripping multiple packages from a fixed position and placing them at different spots in the ULD. It is also capable of stacking multiple boxes on each other. The spots for all parcels are completely determined by the software including the path from the initial to the final position. The software only needs to know the size of the packages and the order in which the packages are delivered to the robot. The ability to automatically detect the parcels and their orientation was being implemented. However, this has not been implemented in the final delivery due to corona issues.

This robot could definitely be the beginning of solving the ULD loading problem. It shows high manoeuvrability while being very compact and light weighted. The robot arm also has the ability to attach a wide range of different grippers. This way a more advanced gripper can still be used in combination with the current robot arm configuration.

2

Current state of the robot

In this chapter the current state of the robot is presented. This includes descriptions of the different mechanisms, how they work, and visual representations for clarification purposes.

2.1. Design

2.1.1. Appearance

Figures 2.1 and 2.2 show the 3D model and final robot on the demoday. The final appearance of the robot is industrial and functional, mostly because the beams/shafts are still visible. The robot uses shells to make the operation of the robot safer. More on the shells in chapter 3.1.3.

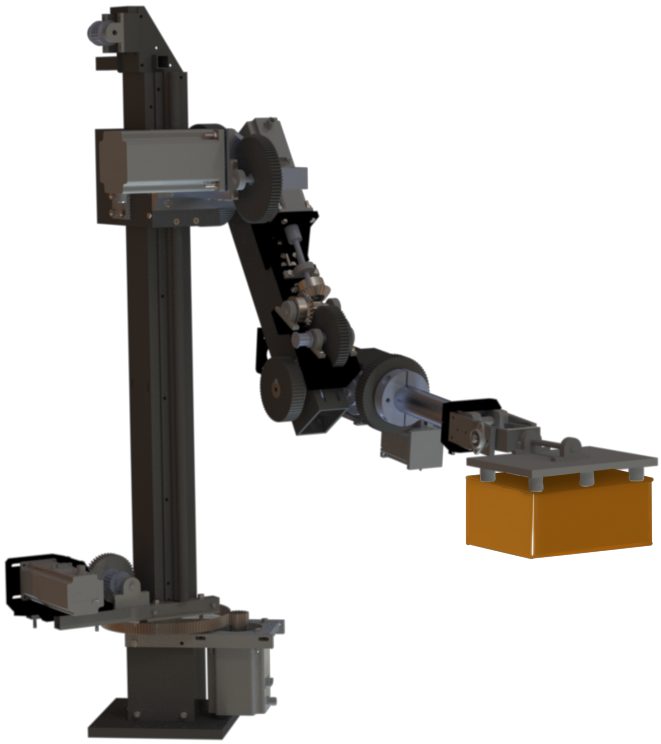


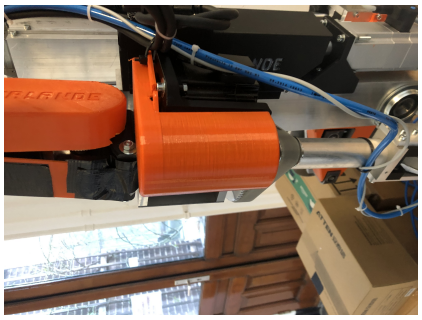
Figure 2.1: *Tetris 3D model without shells.*

Figure 2.2: *Tetris on Demoday.*

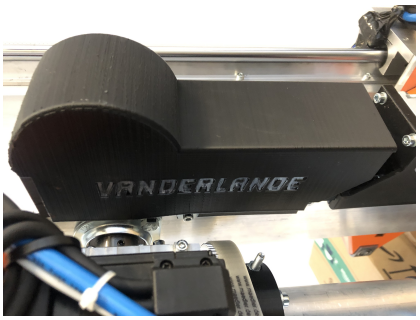
2.1.2. Shell

Due to the COVID-19 circumstances and broken 3D printers there are 4 shells on the robot instead of the pre-conceived 8 shells. Some of the shells did not fit due to the slide-ability of some components that needed to be tuned (for belt tensioning or properly connecting gears). In Figure 2.3 (2.3a to 2.3d) you can see the 4 printed shells. The choice is made to imprint the logo of Vanderlande on the shell and use the orange colour of the company. For safety all the gears need to be covered. Currently this is not the case. More on this is in chapter 3.1.3

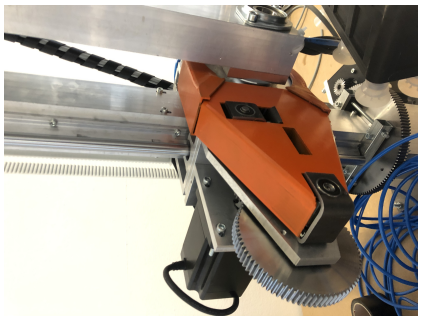
(a) *Shell 1.*



(b) *Shell 2.*



(c) *Shell 3.*



(d) *Shell 4.*

Figure 2.3: *Shells.*

2.2. Mechanics

2.2.1. Current state

The current state of the robot is shown in Figure 2.4 with its full test setup and on its own in Figure 2.5. The robot has been designed for the type A2N container which is approximately 3 x 2 x 2 m (width x depth x height) in size, but it has been scaled down by 50% for budget and time considerations. Figure 2.4 shows the robot bolted to the middle of a large wooden plate which functions as the ULD floor. This represents the connection between what is deemed to be the robot and what will from now on be called the test setup. Each of these will be discussed respectively.

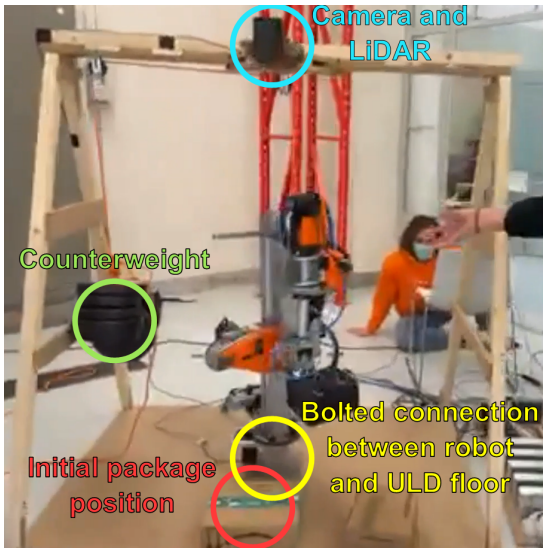


Figure 2.4: *Complete current test setup of the Tetris robot.*

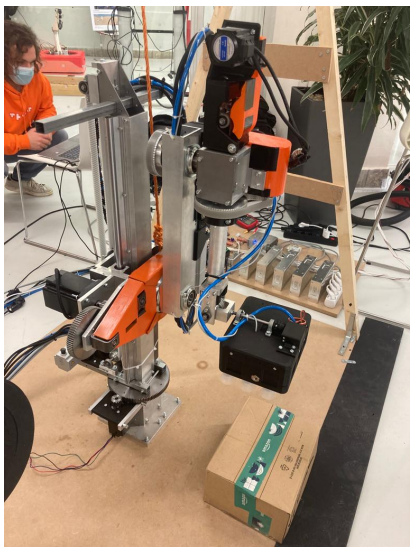


Figure 2.5: *Current state of the Tetris robot.*

Current test setup: The test setup consists of the aforementioned ULD floor and a surrounding

wooden structure. This structure represents the side walls and top of the ULD. A camera and LiDAR are used for package detection and mounted above a hypothetical ULD door as shown in Figure 2.4. The camera is mounted directly onto the wooden structure at the top and the LiDAR is mounted through a 3D-printed component as shown in Figures 2.6 and 2.7.

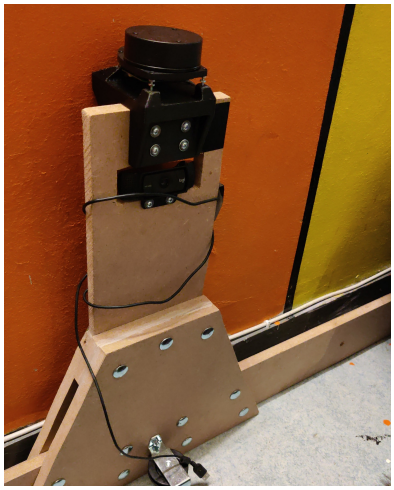
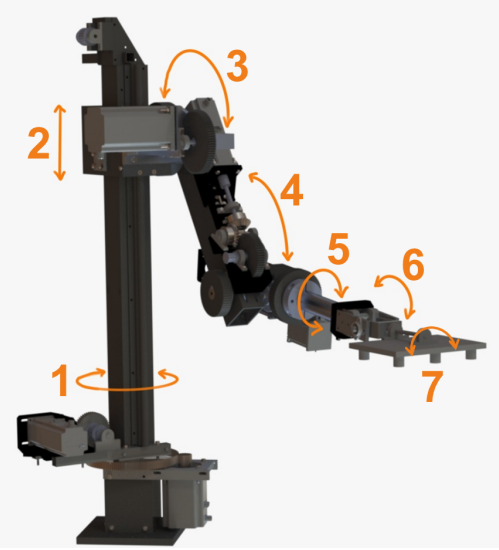


Figure 2.6: *Camera and LiDAR mount from a distance.*

Figure 2.7: *Camera and LiDAR mount from nearby.*

In addition to the camera and LiDAR, there are also two pulleys bolted to the top structure. These are used to guide a rope which is connected to the robot carriage on one side and a 22 kg counterweight on the other side. This system was introduced as a safety measure such that the robotic arm cannot fall down onto the gear boxes in the base unexpectedly during the testing phase.

Current robot: In order to understand the mechanical design of the current robot, it is useful to first define its seven joints (showcased on a render of the robot in Figure 2.8):



1. Base/pole rotation
2. Lift
3. Shoulder rotation
4. Elbow rotation
5. Forearm rotation
6. Wrist pitch
7. Wrist rotation

Figure 2.8: *Visualisation of the seven degrees of freedom of the robot.*

The robot has been designed with the aim of it being highly maneuverable so that it is able to easily reach anywhere within the ULD. By having it placed inside the ULD instead of at the door, the arm can be a lot shorter and requires less heavy motors and structural components. Using the lift and rotating pole allows the arm to be of even smaller size and the arm is also capable of rotating fully past itself, thus allowing it to more easily move past previously stacked packages. This move-

ment was tested using a simple CAD model which was iterated upon to find the best joint positions and component sizes. The simple model with finalized general dimensions was used to perform a speed/joint movement analysis. This analysis and further description of it can be found in the "Speed/joint movement analysis" file within the "Documents" folder.

To gain further understanding of the mechanical design, each of its sub-assemblies will be discussed. These are defined in Figure 2.9.

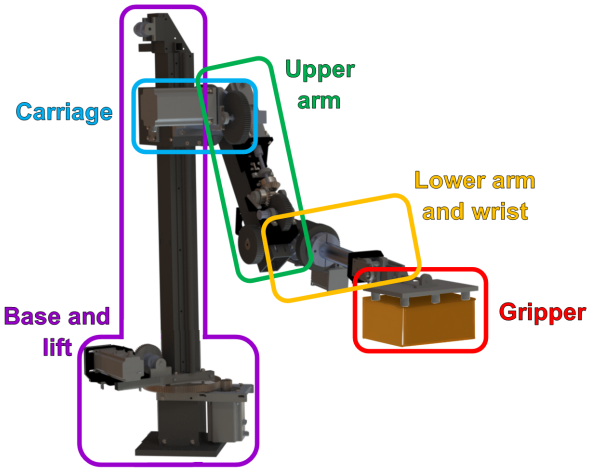


Figure 2.9: *Definitions of the different sub-assemblies.*

Each final sub-assembly of the actual robot will be shown side-to-side with its CAD model and discussed. First are the base and lift of the robot. The CAD model is presented in Figures 2.10, 2.11 and 2.13 and the actual sub-assembly is shown in Figures 2.12 and 2.14. The base rotation is driven by a NEMA 23 motor in combination with a single gear conversion of 20:127 (≈ 6.35). The 20T gear is connected to the motor using set screws. The motor can be moved to and from the large gear and fixed at the desired position at which the gears line up properly. The large 127T gear is connected directly to the outer structure of the pole. This outer structure is connected to an inner structure through a series of bearings allowing the outer structure to rotate with respect to the base.

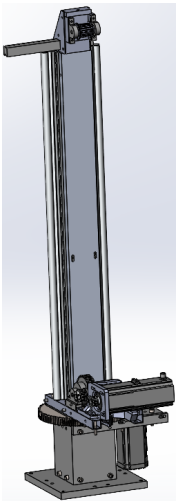


Figure 2.10: *Zoomed out CAD model of the base and lift sub-assembly.*

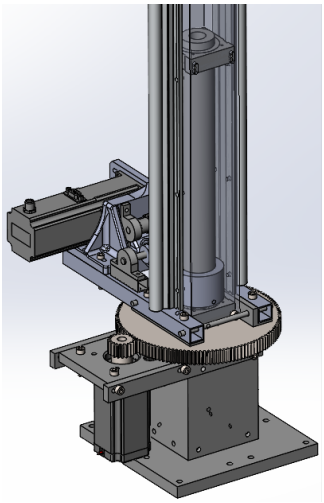


Figure 2.11: *CAD model of the base and lift sub-assembly with transparent outer pole structure.*

Figure 2.12: *Actual base and lift sub-assembly.*

The lift is driven by a NEMA 57 motor from Festo with an encoder and brake (EMMS-ST-57-M-SEB-G2). It is connected through a belt conversion of 32:40 and a gear conversion of 14:50 to a large AT5 belt of 25mm width (≈ 4.46). All belt pulleys and gears are fixed to their shafts using set screws. The large AT5 belt goes along the entire pole to a belt pulley at the top and is connected to the carriage through two clamping plates. The first belt connecting the motor to the second small belt pulley can be tensioned by moving the motor. It is fixed with four bolts through a 3D printed components with slots in it. The large belt is tensioned at the top of the pole. The component holding the belt pulley can be shifted upwards and fixed.

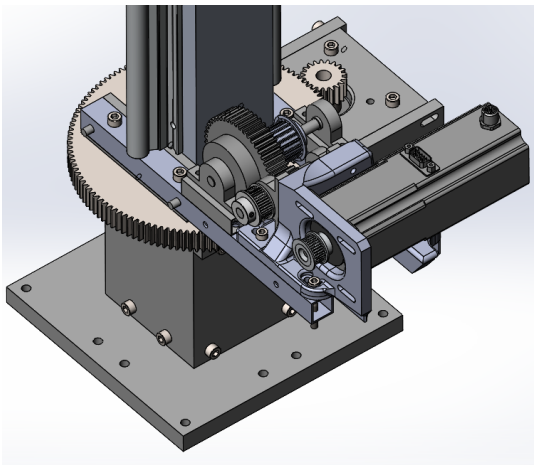


Figure 2.13: *Zoomed in CAD model of the base and lift sub-assembly.*

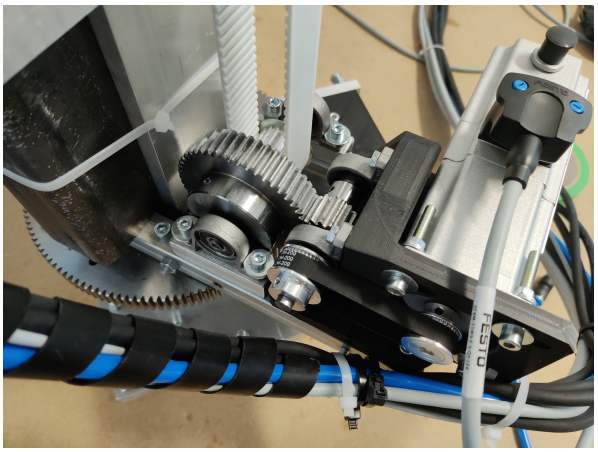


Figure 2.14: *Zoomed in actual base and lift sub-assembly.*

The CAD model of the carriage is shown in Figures 2.15 and 2.16 and the actual carriage sub-assembly is shown in Figures 2.17 and 2.18. As previously mentioned, the clamping plates at the backside clamp onto the lift belt. On the inside structure there are, in total, four rail sliding blocks which guide the carriage along the linear rails attached to the outer pole structure.

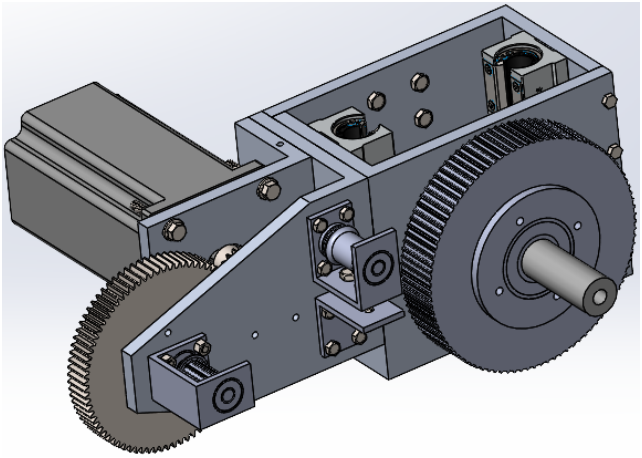


Figure 2.15: *CAD model of the carriage (front).*

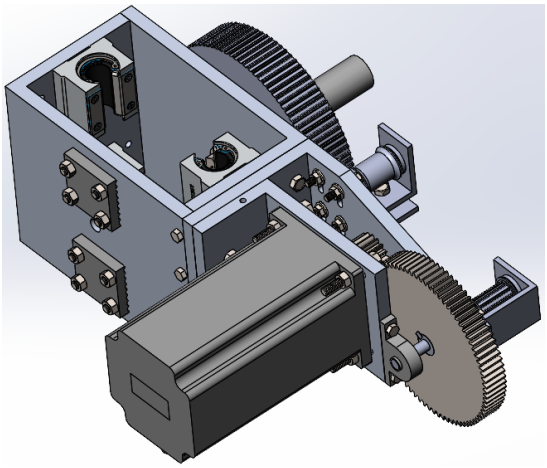


Figure 2.16: *CAD model of the carriage (back).*

The shoulder rotation is driven by a NEMA 34 motor which is bolted to the carriage through an L-profile. There is a power conversion (≈ 35.56) through two gears (18:80) and a belt (12:96). The motor is able to be positioned against the large gear by moving in slots within the L-profile and being fixed at the appropriate position. The 18T gear is connected to its shaft with set screws and the 80T gear has both set screws and a 5 mm gear key. The 80T gear is connected to a bearing fixed on the same L-profile as the motor and a larger one used to also fix the components on the front

side thus creating a stiff sandwich structure.

The large gear is connected to the small belt pulley by being on the same shaft. This belt pulley, being quite small, is both glued onto the shaft and has set screws going into its lower toothed surfaces. It is connected to the 96T belt pulley through a 25 mm width AT5 belt which also loops around a 3D-printed idle pulley. This pulley, in combination with some small steel components, functions as the tensioning mechanism. The U-profile in which it is fixed is fully pulled down by two M6 bolts and fixed at the desired position. Lastly, the large belt pulley is connected to the shoulder shaft through a single bearing and thus free to rotate about it. The four holes visible in Figure 2.15 are used to mount it to the upper arm as will be elaborated upon.

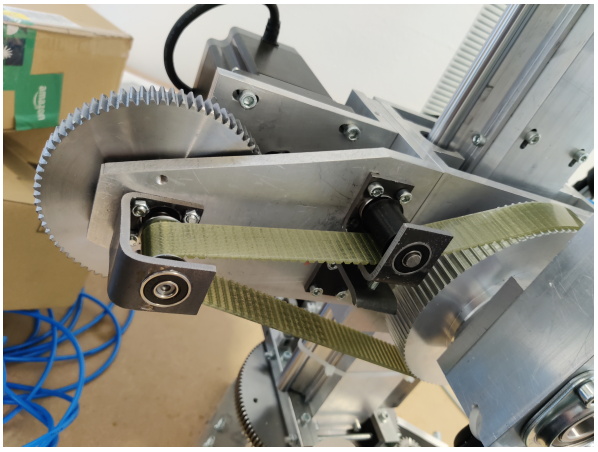


Figure 2.17: *Actual sub-assembly of the carriage (front).*

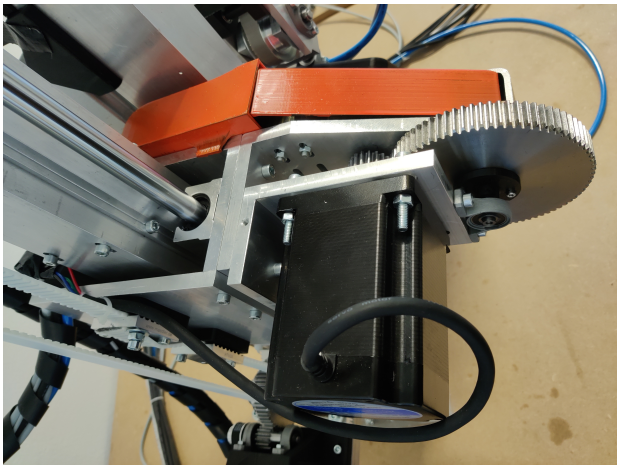


Figure 2.18: *Actual sub-assembly of the carriage (back).*

The CAD model of the upper arm sub-assembly is shown in Figures 2.19 and 2.20 and the actual upper arm sub-assembly is shown in Figures 2.21 and 2.22. The upper arm is connected to the large belt pulley with the four long bolts as can be seen in the bottom left corner of Figure 2.20. It is fixed on the shoulder shaft through set screws in its large flange bearings.

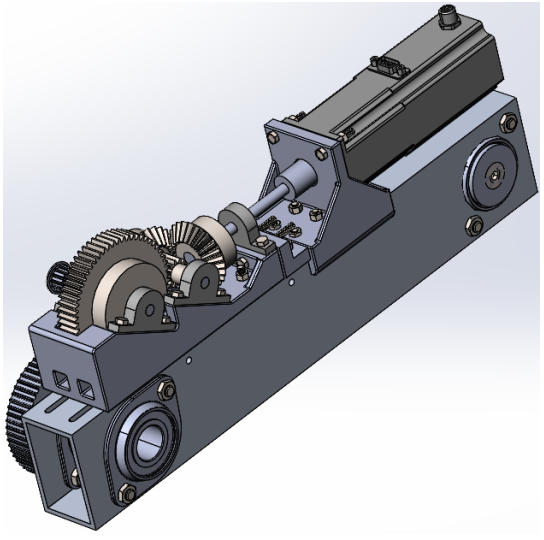


Figure 2.19: *CAD model of the upper arm (front).*

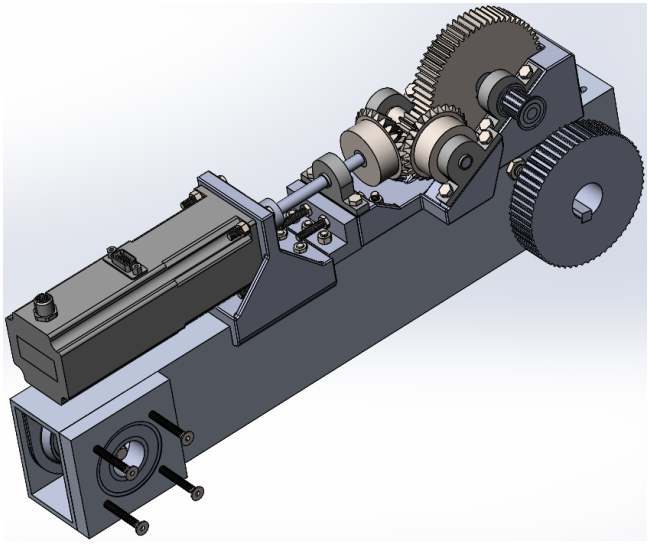


Figure 2.20: *CAD model of the upper arm (back).*

Atop the upper arm structure there are three 3D-printed components: The first holds a NEMA 57 motor with encoder and brake from Festo (EMMS-ST-57-M-SEB-G2). This motor drives the elbow rotation. It is mounted in line with the upper arm structure so that it does not intervene with the

movement of the upper and lower arm past each other. This does however require a right angle bevel gear transition. The motor is connected to the bevel gears through a long shaft on which it is fixed with a motor coupling. From the bevel gears there is also a gear transmission of 12:60 and a belt transmission of 12:60 (≈ 25.00).

All gears are held in a 3D-printed gearbox structure and fixed onto their respective shafts using set screws. The small belt pulley is on the same shaft as the 60T gear and fixed with both glue and set screws. A 16 mm width AT5 belt runs from this small pulley to a larger one to ultimately drive the elbow rotation. To tension this belt the entire gear box is moved over the upper arm structure and fixed through four slots within this structure. It is also connected to the last small 3D-printed component. This component is a small L-profile which sits stationary on the structure. Two long M5 bolts pull the entire gear box structure towards it for tensioning the belt.

Lastly, there are two flange bearings fixed to the structure at the elbow. There are set screws which clamp the shaft connected to the lower arm. The large belt pulley also connects to this shaft with set screws, but in combination with a 8 mm gear key.

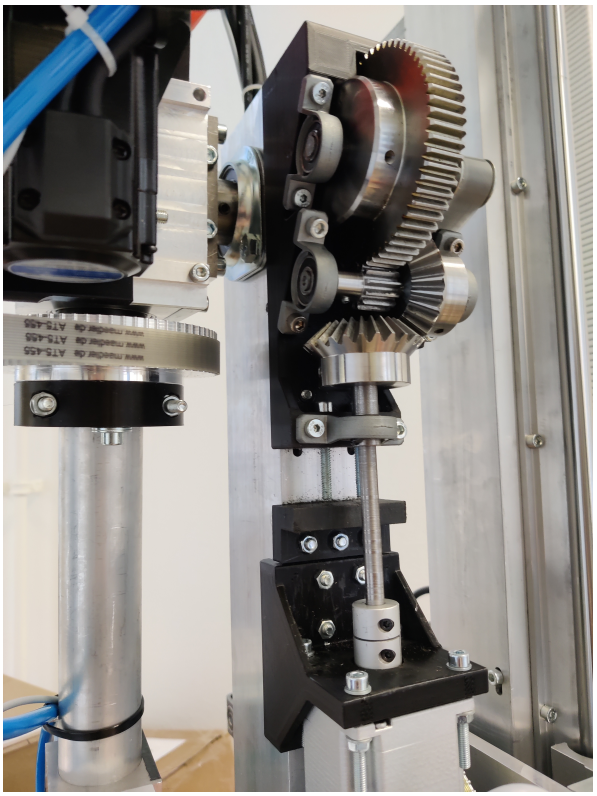


Figure 2.21: *Actual upper arm sub-assembly (front).*

Figure 2.22: *Actual upper arm sub-assembly (back).*

The CAD model of the forearm sub-assembly is shown in Figures 2.23, 2.25 and 2.27 and the actual forearm sub-assembly is shown in Figures 2.24, 2.26 and 2.28. The shaft from the elbow is clamped by two pieces onto the main elbow structure of the lower arm. The sub-assembly has two motors on it: one to drive the forearm rotation and one to drive the wrist pitch.

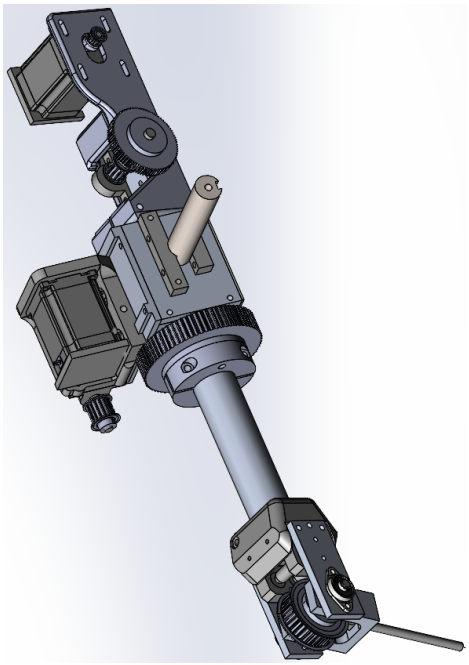


Figure 2.23: *CAD model of the complete lower arm and wrist sub-assembly.*

Figure 2.24: *Actual complete lower arm and wrist sub-assembly.*

The forearm rotation is done by a NEMA 23 motor after going through two belt-driven power conversions: one of 20:60 with a 6 mm width T2.5 belt and one of 15:72 with a 16 mm width AT5 belt (≈ 14.40). The motor is mounted in a 3D-printed component as shown in Figure 2.25. It is fixed at the desired position within four slots to tension the smaller belt. The 20T pulley is directly mounted on the motor shaft with set screws and the 60T pulley is connected to the 3D-printed component through a set of bearings. It is also fixed to the shaft using set screws. The 15T belt pulley lies along the same shaft as the 60T one and is also connected using set screws. Its belt is tensioned by fully shifting the 3D-printed component over the rectangular structure at the elbow and fixing it at the desired location. Lastly, the largest belt pulley is clamped and fixed onto the round forearm structure.

The second motor within the forearm sub-assembly is also a NEMA 23 and drives the wrist pitch through a 16:84 and 12:36 belt conversion (≈ 15.75). The motor itself is again mounted in a slotted 3D-printed component to allow for tensioning the first belt. The second belt runs through the inside of the forearm. It is tensioned by moving a small metal component on which the 84T and 12T belt pulleys are both mounted.

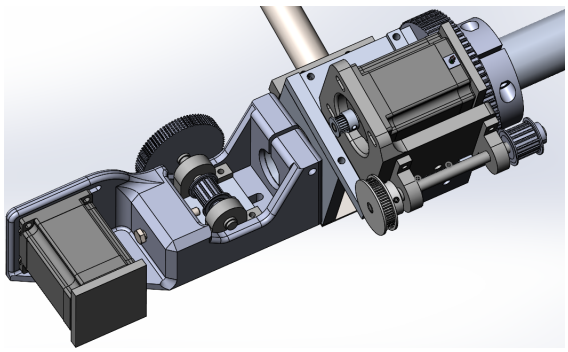


Figure 2.25: *CAD model of the back of the lower arm.*

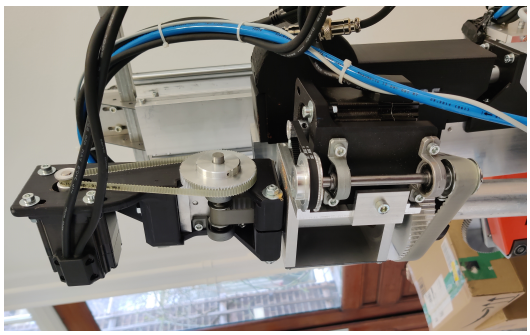


Figure 2.26: *Actual back of the lower arm.*

At the side of the wrist, the belt is guided along two 3D-printed idle pulley so as to not scrape past the sides of the metal structure. It is then connected to the 36T belt pulley which is connected to a small U-profile through its shaft which in turn moves the entire wrist up and down.

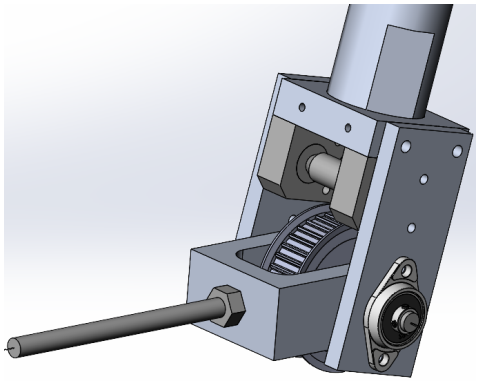


Figure 2.27: *CAD model of the wrist.*

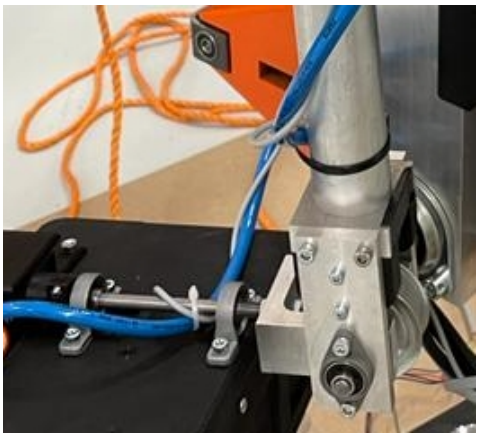


Figure 2.28: *Actual wrist.*

The last element of the mechanical design is the gripper and the last degree of freedom: the wrist rotation. The CAD model of the gripper is shown in Figures 2.29 and 2.41 and the actual gripper is shown in Figures 2.30 and 2.42. The main structure of the gripper is fully 3D-printed and consists of three main parts: The bottom plate is a box of sorts. The suction cups are all clamped with a nut on either side onto the bottom plate. The venturi and tubing are all also contained within this bottom plate.

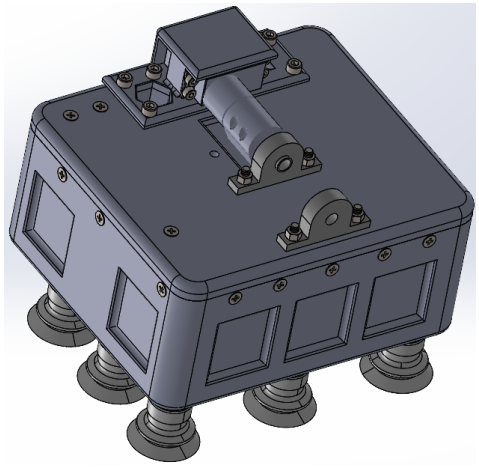


Figure 2.29: *CAD model of the gripper (top).*

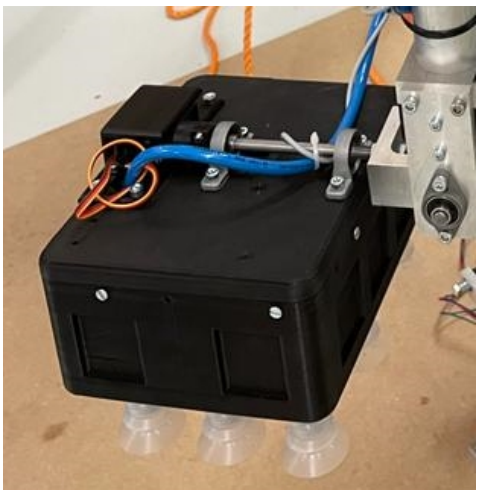


Figure 2.30: *Actual gripper (top).*

The second component is the top plate which functions as a lid and keeps all suction elements contained and safe within the inside of the gripper. It is screwed on from the sides so it stays attached to the bottom plate. Lastly there is a small component also bolted onto this lid which functions as the mounting element for a 360 degree servo motor to drive the wrist rotation. It is connected to the wrist shaft through a motor coupling and a pair of bearing screwed onto the top plate.

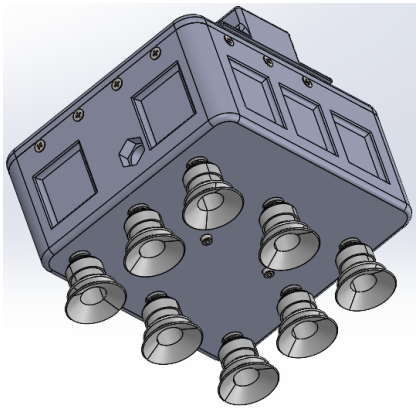


Figure 2.31: *CAD model of the gripper (bottom).*

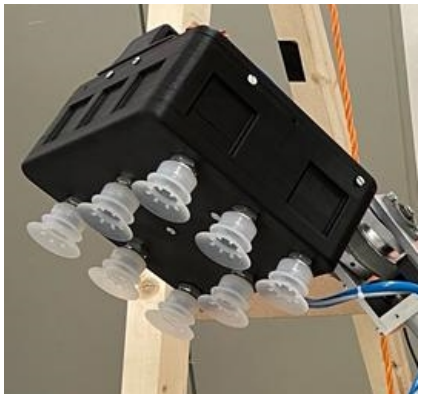


Figure 2.32: *Actual gripper (bottom).*

2.2.2. Part list

The robot has over 220 different parts. The file containing these parts and info is "Part list.xls" and can be found in the "Documents" folder.

2.2.3. Assemblage plan

In this report multiple pictures of the different subassemblies will be shown with additional information for clarification. However the real assembly drawings can be found in an additional File which hold all the parts/naming. If one were to reassemble the robot it is recommended that both the assembly drawings and this section of the report are used.
The assemblage plan in this report runs from the base of the robot to the gripper. This section is build up to be able to build the robot in the right way and order. So this means the individual subassemblies as well as the correct order and way of joining the subassemblies.

Assembly plan subassemblies

1. **Sub-assembly 1: base and lift** For the base and lift subassembly, it is recommended to begin with the base, a picture of this is shown below:

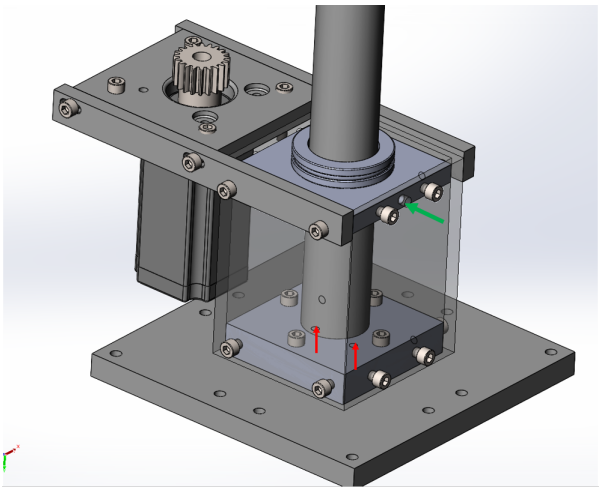


Figure 2.33: *Base assembly.*

A few things to note here: The aluminium tube in the centre is screwed on by two M5 bolts at the red arrows, and to cancel any play due too the fitting in the top plate not being exact, a

M6 setscrew needs to be placed at the green arrow. The axial bearing is placed last in the base assembly, the part of the bearing with the tight 40 mm fit needs to be resting on the top plate, instead of the other way around.

After the base is assembled, the 127 teeth M1,5 gear needs to be installed, before this, press the 40 mm bearing in the designated spot in the gear. When installing the gear, note that the bearing needs to be faced vertically up, instead of down. After this, the two steel beams need to be installed, For this, a few things to note:

(a) The two steel beams are not identical, there is a left and a right version. The left version of the steel beams needs to be placed on the two tapped holes furthest from the center of the 127 teeth gear,
(b) The side with the two flow drilled holes need to face the outside.
(c) The side with multiple holes with in the middle two flow drilled holes need to be faced vertical up.
(d) The two 3d printed parts between the steel beams and the gear.
(e) Do not tighten the bolts yet, the pole structure first needs to be installed and pressed between the two steel beams, after which the bolts can be tightened.

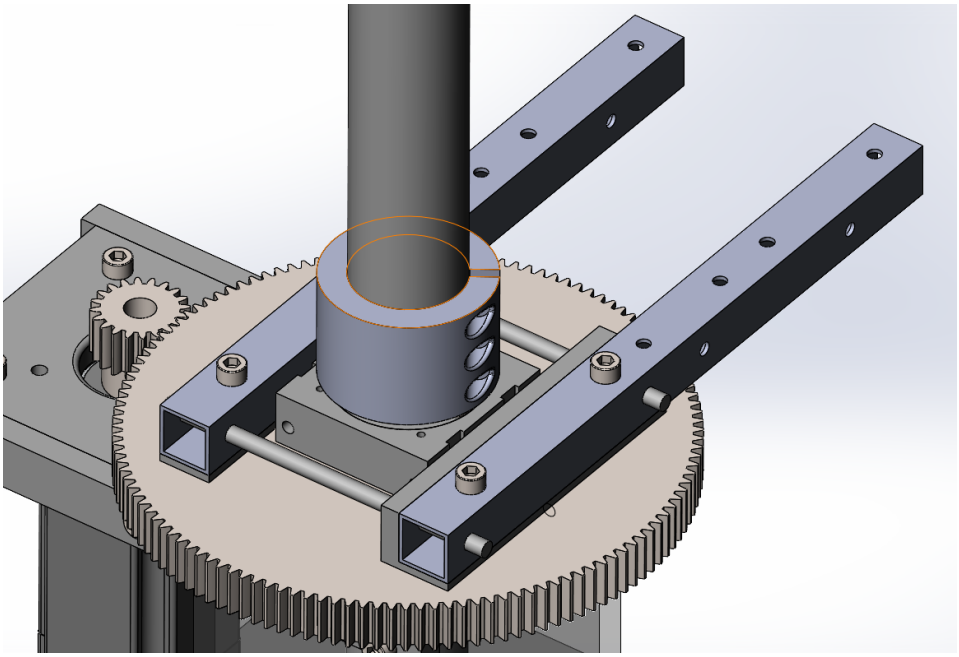


Figure 2.34: *Parallel steel beam assemblage.*

The two 3d printed parts seen in 2.34 around the vertical pole are there two lock the vertical pole in place and to keep the vertical pole from going upwards, this means the clamping 3d printed part can be tightened firmly to ensure the vertical position is kept. The square bottom 3d printed part needs a bearing to be installed previous to assembling it shown in 2.34, This 40 mm bearing needs to be placed and clamped using 4 M3 bolts with a washer. After this the part can be installed.

After the base is assembled according to 2.34, The pole needs to be assembled separately, this will be covered next. To assemble the pole, first the top piece needs to be installed, first the

two sheet metal parts, then the two machined aluminium parts. After this the two linear rails are screwed on and at last, the 3d printed part in the centre of the pole. This 3d printed part also holds a 40 mm bearing and this bearing needs to be installed previous to installing the part in the pole. The assembly is shown in 2.48

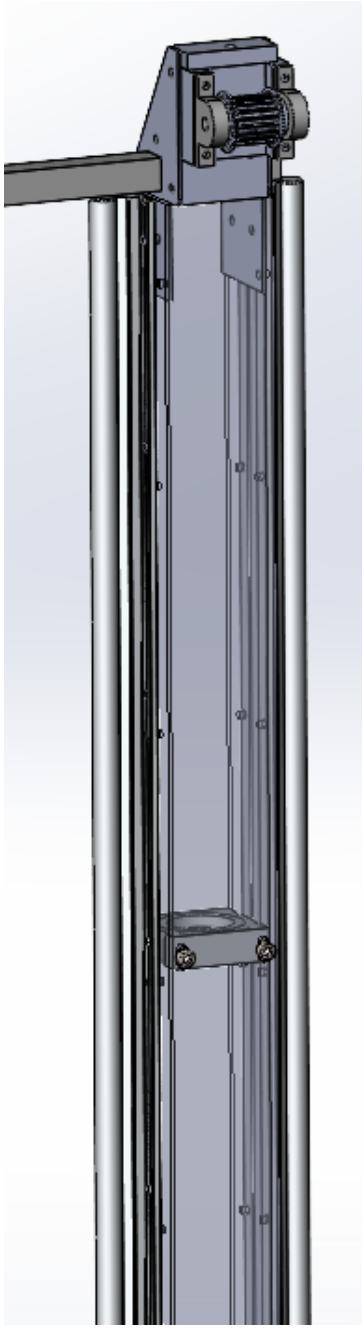


Figure 2.35: *Vertical pole assembly.*

After the pole is assembled, the pole can be placed on the base, the 3d printed square part can be bolted to the pole, and the two parallel steel beams can be clambed to the pole with the two M6 threaded shafts. After the two steel beams are clambed to the vertical pole, the four bolts which go into the 127 teeth gear can be tightened. The only thing left to do is to assemble the gearbox for the lift.

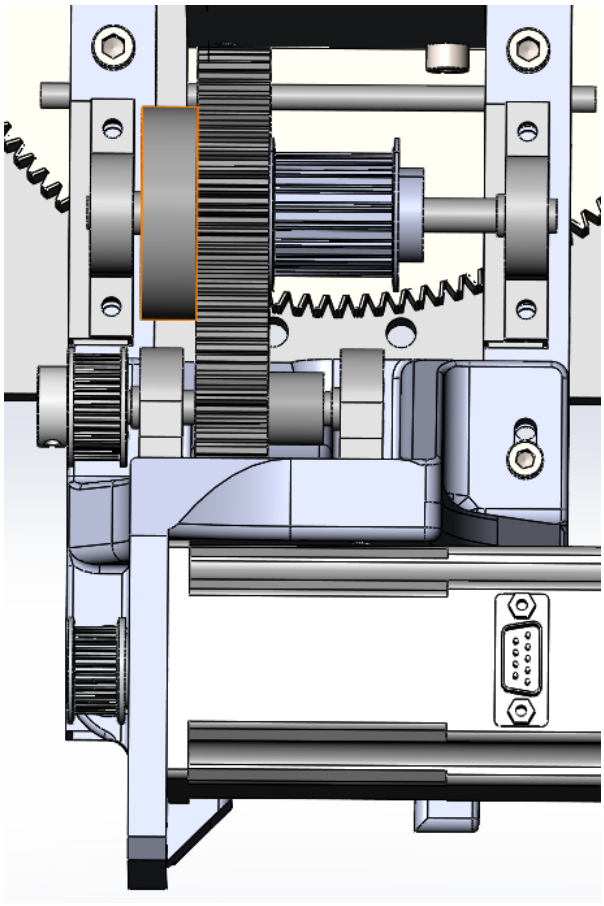


Figure 2.36: *Gearbox lift.*

For the gearbox of the lift, it is best to start with assembling the shaft with the belt pulley for the gt2 belt and the small 14 teeth gear on the 3d printed motor holder. Do not install the motor yet. After this, install one of the two 15x15 aluminium beam pieces with a bearing on top and put the shaft with the 50 teeth gear and 19 teeth belt pulley in place. After this the second 15x15 beam piece with the bearing on top can be installed. Tighten both bolts to the flow drilled holes in the two steel beams below.
When this is done, place the 3d printed motorholder and allign the two steal gears, when the two gears are alligned and on the right centerdistance, tighten the 3d printed motorholder to the two steel beams.
At last put the motor in place, tighten the belt and tighten the bolts which hold the motor.
The assembly of the base and pole is complete.

2. **Sub-assembly 2: Carriage**

The assembly for the carriage is straightforward except for the axis with the 80 teeth gear and the pulley, as well as the order to assemble the big L profiles. The carriage needs to be assembled on the lift, this because otherwise the carriage can not be put on the linear rail as the tensioning mechanism for the vertical lift is in the way. Removing the tensioning mechanism is an alternative approach, which also works. In this report the focus will be on the assembly on the pole.

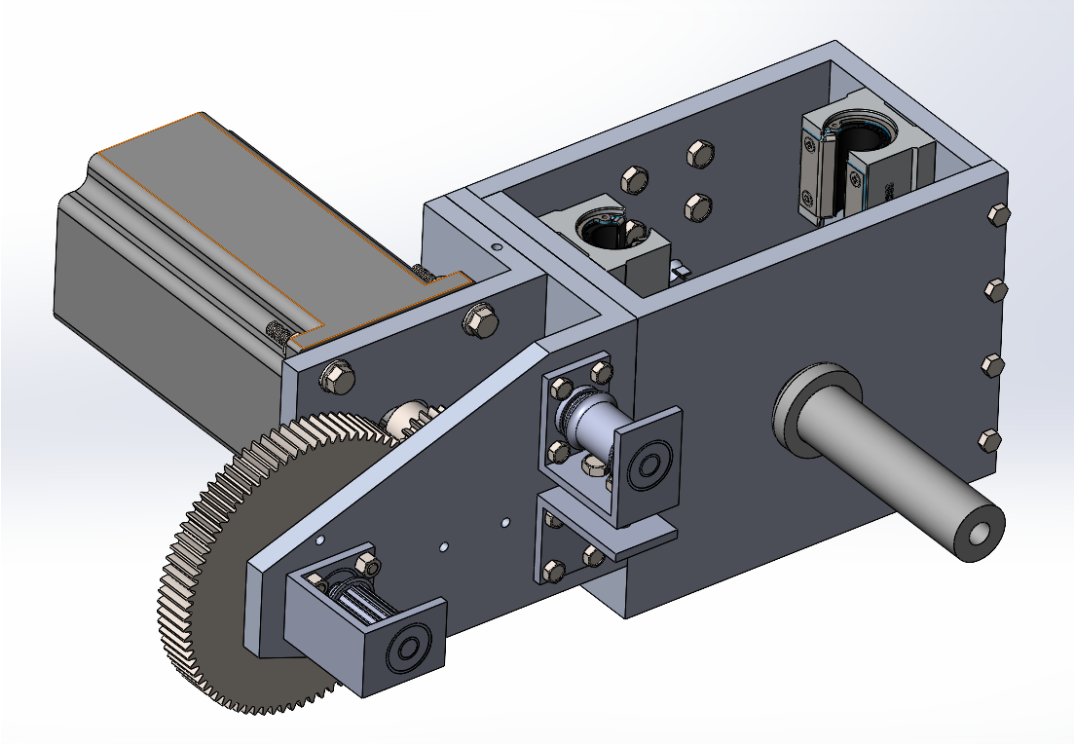


Figure 2.37: *Carriage global.*

To begin, insert the 4 linear bearings on the rail, two on each rail. After this, the 25 mm steel shaft is to be placed and tightened with high force, if the shaft is in place, put the 3d printer spacer over the steel shaft. After this, the two L profiles can be put and screwed on the linear bearings and screwed to each other. Now the system should look like a vertical pole with a box around it. After this, bigger, triangle shaped L profile should be installed. After this, the tensioning steel U profiles should be installed as well as the bearings which go inside the U profiles.

When this is done, Put the 80 teeth gear on the shaft and install the key in the keyway as well as the axial locking part. Before installing the shaft in the bearings, put the belt around the pulley(which is not yet installed). When this is done, place the pulley in the designated spot, and simaltaniously put the shaft with the 80 teeth gear in from the other side. Tighten the setscrews and apply loctite to make sure the setscrews as well as the pulley will not slip. After this, Place the last L profile, first install the bearing which holds the other side of the shaft where the 80 teeth gear is attached to. Make sure to tighten the setscrews on the bearings to hold the shaft axially. After this install the smaller gear on the motor, and install the motor with the right centerdistance so that the gears run smoothly.

For the lift belt installation, put the open belt through both belt pulleys (on top of the pole and at the bottom) and make sure the ends of the belt arrive at the back side of the carriage. When this is the case, clamp the ends of the belt with the designated clamping blocks, and cut away the part of the belt that is too long. After this, tension the lift belt by turning the screw on top of the tensioning mechanism clockwise till the belt is under tension.

The assembly of the carriage is complete.

3. **Sub-assembly 3: Upper arm**

   The Upper arm consist of 3 parts, the bearings, the gearbox for the elbow and the mounting of the big pulley which is driven by a belt to move the shoulder joint. It is best to begin with the mounting of the big pulley and the gearbox, this to be able to reach all places without the bearings interfering. The mounting of the big pulley is fairly easy, the pulley is bolted on and clambed by 4 bolts, between the upper arm structure and the pulley is a square block which hold a bearing.
   Note, before mounting the big pulley, a bearing needs to be pressed in the designated bearing hole.

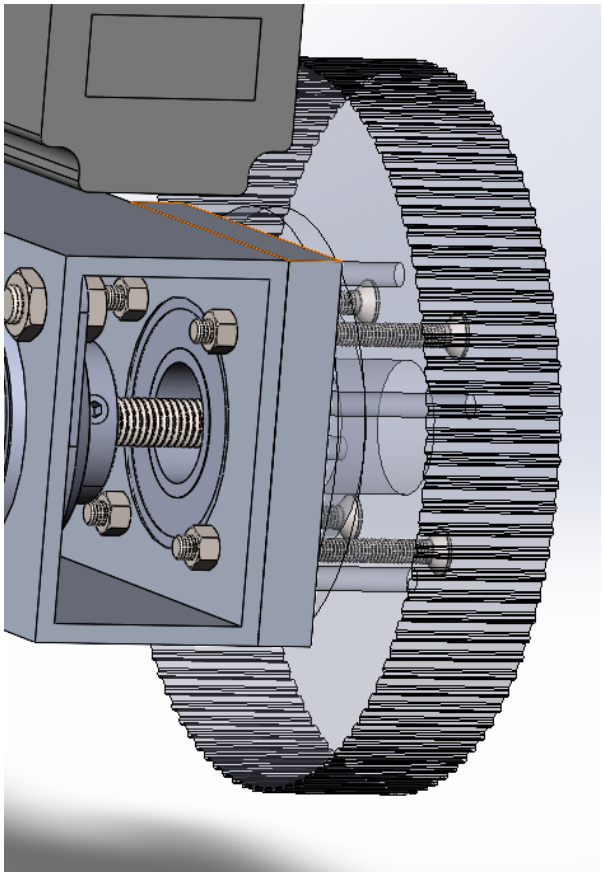


Figure 2.38: *Mounting big pulley arm.*

After the pulley is mounted, the gearbox needs to be assembled, this is done in the following steps:

(a) Mounting of motor on holder, mounting shaft coupler on motor shaft.

(b) Mounting of motor holder on upper arm structure

(c) Mounting of 3d printed L profile on upper arm structure

(d) Putting all the gears on the indiviual shaft and locking them with loctite and setscrews

(e) applying bearings around the shafts, after this, mount the bearings, with the gears between them, on the 3d printed gearboxholder

(f) Mount the complete gearbox on the upper arm structure, however do not yet lock the screws down.

(g) Put the shaft with the one bevel gear attached in the motor coupler, however do not yet lock the screws down.

The reason why the screws do not need to be tightened yet is because the belt which connects to the forearm shaft first needs to be applied and tightened. For this, First the forearm needs to be assembled.

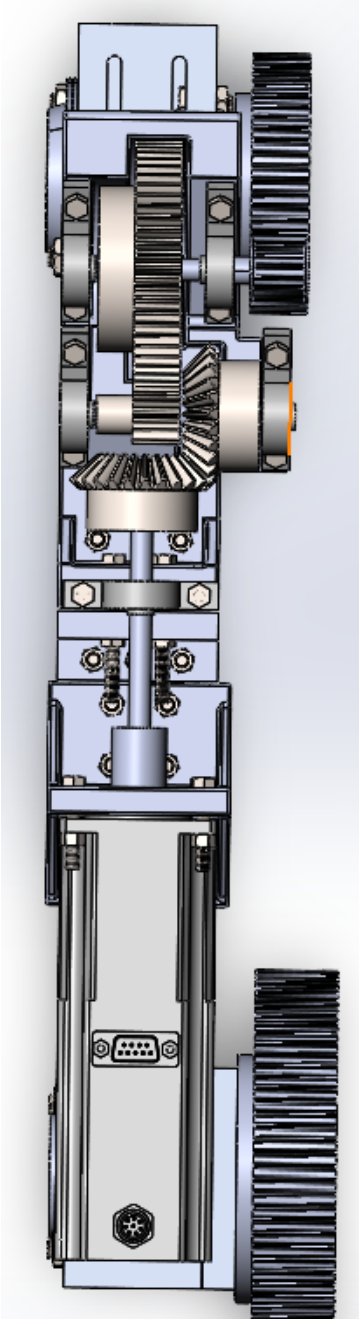


Figure 2.39: *Top view upper arm.*

the bearings are last. As can be seen in 2.43 the bearings are made of of two flanges which hold a bearing. The flanges need to be clamped on to each other to operate without play. The bearings need to be assembled according to 2.43

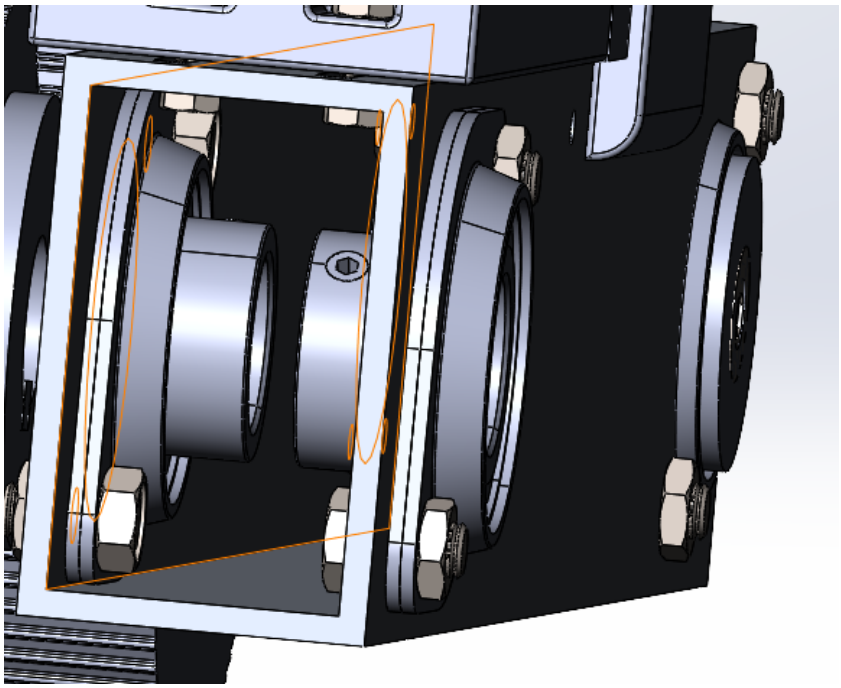


Figure 2.40: *Bearings upper arm.*

This completes the assembly of the upper arm.

Not that the forearm is completely assembled, it can be mounted on the carriage, this is done in the following steps:

- Make sure carriage is elevated to the highest level to allow the forearm to 'hang' from the carriage.
- When sliding the forarm onto the 25 mm steel shaft, put the belt from the carriage on the big pulley. Slide the forarm on the shaft till the pulley hits the 3d printed spacer. After this, tighten the setscrews to keep the forearm from falling off the carriage.
- The belt is installed but still too loose, it needs to be tightend, to do this, first put the 8 mm shaft from the tensioning pulley in one bearing of the tensioning U profile. While simultaniously putting the tensioning pulley in line with the bearing, make sure the belt is between the U profile and the tensioning pulley. When the shaft and pulley are aligned, put the shaft through the pulley.
- The only thing left to do is to tighten the bolts which connect the two U profiles. This as a result tensions the belt.
- tighten the screws which mount the U profiles to the rectangular L profile

After this, the upper arm is assembled correctly on the carriage.

4. **Sub-assembly 4: Lower arm and Wrist**

   The Lower arm consist of two main parts, the 'base' of the lower arm, and the lower arm itself. The base of the lower arm needs to be assembled first.

   The base of the lowr arm can be seen in the figures below. A few things to note here:

   - the 40 mm bearings need to be pressed in the two aluminium profiles.
   - The steel 25 mm shaft which will connect to the upper arm needs to be bolded on very tight to ensure no play is present in the system.

- The two aluminium bars above and below the 25 mm steel shaft are there to transfer the moment from the shaft to the forearm. The plates clamp to the shaft, this clamping also needs to be very tight to ensure a good power transmission.
- It is right that the motor is not shown in place here yet. First the belt which goes around the forearm needs to be tensioned, after which the 3d printed motor-holder can be fastened. This can only be done if the motor is not present.

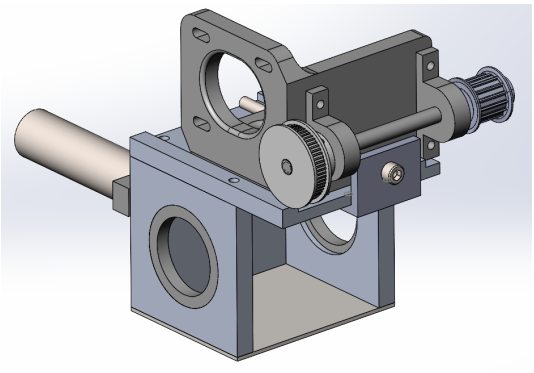


Figure 2.41: *Lower arm base view 1.*

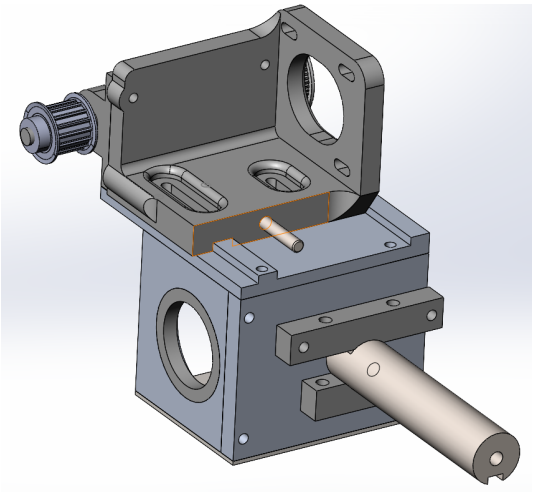


Figure 2.42: *Lower arm base view 2.*

After the base of the lower arm is assembled, The lower arm itself needs to be assembled. The main thing to not here is the belt which goes through the lower arm to the wrist. For this, the back of the lower arm needs to be assembled first. See the picture below:

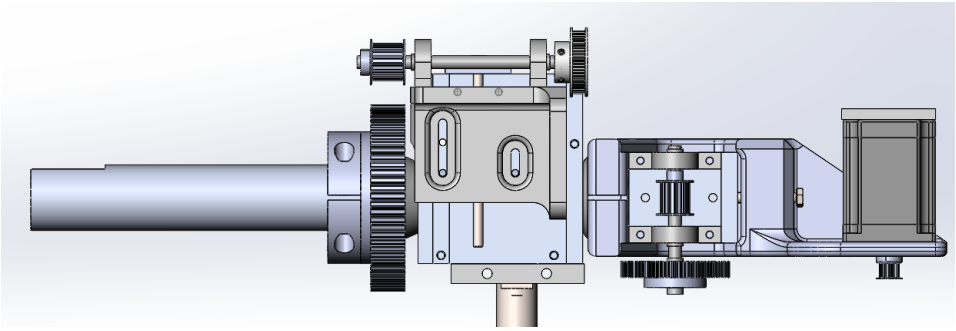


Figure 2.43: *Lower arm without wrist.*

The lower arm 40 mm tube needs to be installed, after which the big pulley + clamping plates can be put in place and tightened, as well as the 3d printed motor/tensioning mechanism holder, which can also be tightened. Now that the big pulley is in place, the belt for the lower arm rotation can be installed. When the belt is installed, the belt can be tensioned by tighting the M6 bolt going through the base of the 3d printed motorholder on top of the base of the lower arm.

After the lower arm rotation belt is tensioned, the 3d printed motorholder on top of the base of the lower arm can be tightened by the screws in the bottom of the part. After which the

motor can be mounted, and the smaller gt2 belt can be tensioned. The fifth joints is now finished.

For the belt which goes through the lower arm, the following steps need to be taken:

- The shaft/bearing/pulley assembly needs to be made according to 2.44, before installing it on the aluminium machined part, put the belt around the smaller pulley and pull it through the lower arm.
- Bolt the bearings in place on the aluminium machined part and put the assembly in the 3d printed holder, make sure to push it as far forward as possible.
- put the tensioning screw in the 3d printed part and make sure it is in the aluminium part, however do not yet tighten it.

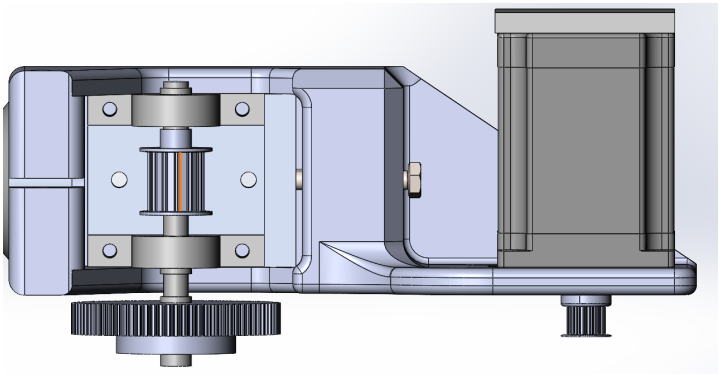


Figure 2.44: *Tension mechanism and motor holder.*

- Do not install the motor yet.
- Looking at the wrist, seen in 2.45. The first thing to do is to mount the base plate of the wrist, this is the vertical plate on the most right with two holes at the top. This base plate can be screwed in place with a setscrew.

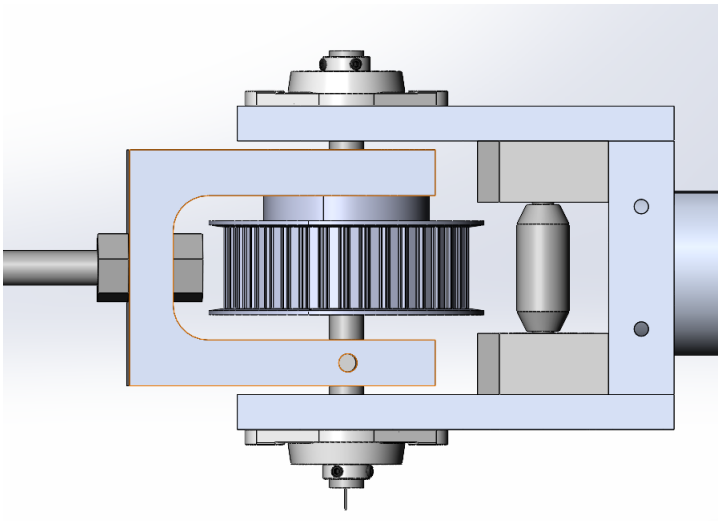


Figure 2.45: *Wrist.*

- Mount the side plates and bearings

- Mount the 2 3d printed belt spacer plates + shaft + 3d printed flat pulleys. The belt should go between those two flat pulleys. The pulleys are placed to make sure the belt does not rub against the aluminum walls which will damage the belt.
- lock the gripper shaft with the two bolts to the wrist machined part.
- Put the belt over the bigger 36 teeth pulley
- Mount the 36 teeth pulley to the wrist machined part, locking it with two screws to the side.
- place the shaft through the bearings and the wrist machined part as well as the 36 teeth pulley. Lock the wrist axially with a setscrew on the top.
- Going back to the other side of the lowr arm, tension the belt, lock the aluminium machined part from below.
- Mount the motor, attach the belt and tension it

The assembly of the lower arm and wrist is complete.

To mount the lower arm and wrist to the upper arm, the 25 mm shaft needs to be slided in the bearings of the upper arm till it hits the clamping plates. After this the big pulley can be installed, for this put the key in the keyway and the tighten the setscrews. When this is done, slide the 25 mm shaft back till the flange of the big pulley is 6 mm away from the upper arm structure. After this, lock the shaft axially by tightening the setscrews in the bearings from the upper arm. The last thing to do is to apply the belt, tension it by tightening the screws to pull the entire gearbox back. And locking all the screws, for both the gearbox and the motor coupling.

5. **Sub-assembly 5: Gripper** The gripper consist of two part, a bottom part and a top part, which are easily bolted onto each other. It does not matter which part is assembled first, This report will start with the bottom part.

   The bottompart is all 3d printed in one piece, the suction cups need to be placed from the bottom and screwed on from the inside. After this is done, The ventury which creates a vacuum from pressurised air needs to be installed. After this the only thing left to do is connect al the air tubes. Make sure all suction cups converge to 1 signle tube (with the help of T connections) which goes into the ventury. Make sure this air tube goes into the vacuum port instead of the pressure port of the ventury. The top and bottom image can be seen below:

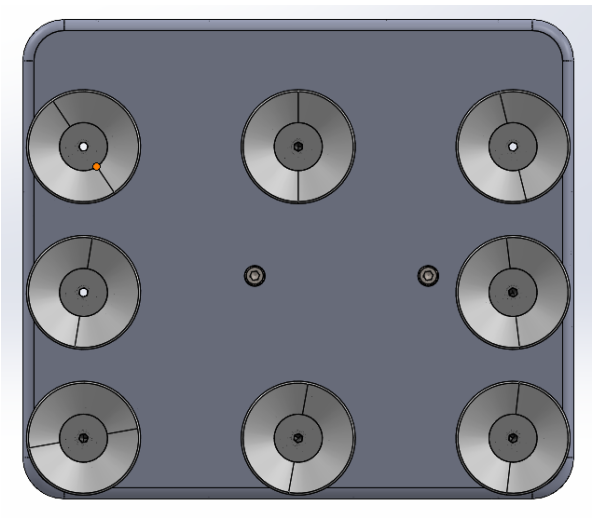


Figure 2.46: *Bottom view gripper.*

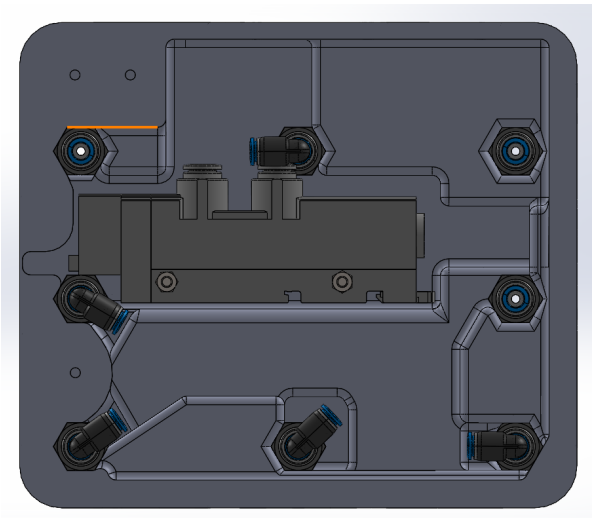


Figure 2.47: *Inner view gripper.*

The top part of the gripper holds the motor for the seventh axis and the bearings to support the shaft coupled to the motor and the wrist. To assemble the top plate, first bolt the bearings in place. After this bolt the Servo/shaft coupling to the servo, and assemble the servo in the servoholder. The servoholder bolts on the top plate. When this is done, assemble the pressure air tube coupling to the top plate. See the picture below for a visual of the top plate.

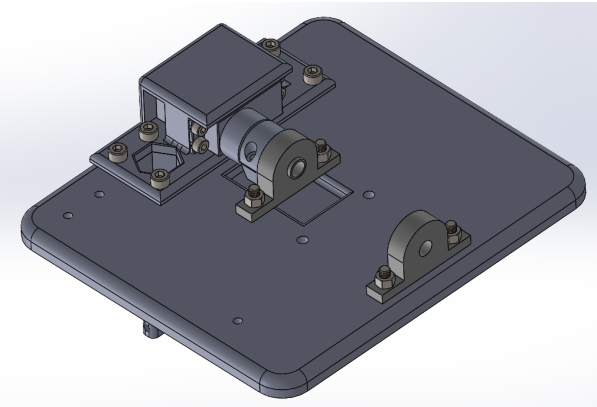


Figure 2.48: *Top plate gripper.*

After the top and bottom parts of the gripper are assembled individually, do not bolt them togheter yet. First attach the pressure air tube from the ventury to the top plate, and run the ventury signal cable to one of the holes in the middle of the top plate. After this, the two gripper parts can be bolded togheter and the gripper assembly is completed.

To mount the gripper to the wrist, put the shaft which comes out of the wrist in the bearings of the gripper and tighten the shaft to the servo shaft coupler. After this tightend the setscrews on the bearings to hold the shaft axially.

2.2.4. Maintenance

The maintenance section of this report is split into two types of maintenance that needs to be done to the robot:

1. Regular maintenance done once after every X amount of operation hours, this is applicable for the gears and the belts within the system.

2. Unplanned maintenance that needs to be done if errors or failures to the robot occur.

Regular maintenance The regular maintenance consist of the lubrication of the gears and the tensioning of the belts. Joints 1 to 4 all have steel gears as a transmission stage, which needs to stay lubricated and cleaned to prevent Dust and Dirt to come between the gears. This re-lubrication and cleaning needs to be done every 200-250 hours of operation.
As the gears are in the open air and operate at low speeds, so a grease is recommended for the lubrication. For the robot to function correctly, the belts need to be tensioned. However, after a while the tension in the belt decreases. This reduces the accuracy of the robot. It is adviced to check the tension in the belts every 3 months. If the robot is not in operation for longer than 3 months, it is adviced to check the tension before the first run. The belts are tensioned correctly if it vibrates and makes a low guitar string like sound if touched.

Unplanned maintenance The unplanned maintenance is build up in two parts, first the components/systems which have a higher risk of needing it + what to do to fix it. Second the identification of the signs the robot will give off when some components/system need maintenance as well as a link to which component/system it is.

The components of the system which have a higher risk unplanned maintenance are:

1. Axle hub connections done with setscrews and clue. The Axle hub connection of this robot consist of form-fitting connections and setscrews with glue. The form fitting connections are done with a keyway or bolts which clamp the flange of the gear to a part. These form-fitting connections are very reliable and have a low to zero change of needing unplanned maintenance as it is the strongest link of the chain, another weaker chain will fail fist.
   Which is the axle hub connections made with set screws and glue. If one of the set screws and glue connections fails, the following steps need to be taken:
   (a) Setscrews need to be unscrewed and gear removed from shaft.
   (b) Glue rests needs to be removed on both shaft as gear.
   (c) The shaft and connecting part of the gear need to be cleaned thoroughly, note! not with alcohol, as this will corrupt the glue.
   (d) Apply red Loctite glue on shaft.
   (e) re-apply the gear on the shaft.
   (f) Apply red Loctite on setscrews and tighten the setscrews firm.
2. Tension in belts. Like said earlier, for the robot to function correctly, the belts need to be tensioned. However, after a while the tension in the belt decreases. This reduces the accuracy of the robot and if the tension is way too low, the belt can even skip gears, which corrupts the robot. To fix this issue the belt needs to be retensioned.
3. Belt breaking, if a belt breaks it needs to be replaced, trying to repair a broken belt is strongly discouraged.

When the robot is in operation and something fails/needs unplanned maintenance, the robot will give off signals, the next section will enumerate the signals and how to identify them and get to the problem quickly. Once the signals are identified and the problem is known, one can go back to the above section and check how to resolve the problem.

1. Ticking noise + part of robot not moving according to the software: The first option for this is that a belt is too loose and slips over the gears, to resolve this the belt needs to be tensioned. The second option is a stepper motor slipping, which can be caused by the arm having reached it's mechanical stop or the load on the robot is too high(due too high acceleration or too much payload). To resolve this, first check which stepper motors slips, second check if the mechanical limit is reached, if yes, re-calibrate the robot. If no, the robots acceleration needs to be decreased or the payload.
2. Hearing a hard snapping sound, a belt is broken, first check which belt is broken, this can be done by visual inspection, or moving the joints and checking which is not moving accordingly. Replace the belt.
3. Robot does not move according to the software. However, no sound present. This means a axle hub connection has failed and is slipping. The axle hub connection needs to be remade.

2.3. Electronics

2.3.1. Electrical Control Panel

Currently all electrical components such as the power supplies, motor drivers, the micro controller and other smaller components are mounted on a wooden plate, the electrical control panel. All components can be turned on and off using the yellow physical control panel. This control panel consists of a switch and a physical emergency button. Turning on the switch will simply turn on all power supplies. When the emergency button is pressed all power supplies will turn off except for the 5 V power supply. The reason for this is that the main computer, micro controller and all sensors will stay active when the emergency button is pressed, and that only the motors are shut down. Make sure that the emergency button is only pressed at real emergencies since the robot can fall down on the ground when the power of the motors is cut off immediately. This is due to the lack of brakes at some motors. The motors and their brakes will be discussed more extensively in the next section.

2.3.2. Motors

At the current state of the robot six of the seven joints are driven by stepper motors. The seventh (wrist rotation) joint is not connected to a motor yet due to issues with the motor, but this will be solved in the near future. For this a simple 270 degree servo motor will be used. Due to driver issues the encoders of some motors are not connected. The cause for this problem is that the encoder resolution of the motor does not match the encoder resolution the motor driver is expecting. Furthermore, only two motors currently have brakes, all other motors do not have brakes. A summary of all motors with additional information about the brakes and encoders is shown in table 2.1.

Table 2.1: *Summary of motors used for every joint.*

Joint	Motor	Encoder	Brake
Pole rotation	Nema 23 stepper motor	no	no
Vertical lift	Festo 57M-SEB-G2	yes, but not connected	yes
Shoulder	Nema 34 stepper motor	no	no
Elbow	Festo 57M-SEB-G2	yes, but not connected	yes
Forearm rotation	Nema 23 stepper motor	yes	no
Wrist pitch	Nema 23 stepper motor	yes	no

As can be seen in the table, only two joints have encoders which results in a system which is mostly open loop. It is only a matter of time for the inaccuracies to occur. Furthermore, at only the vertical lift joint and elbow joint brakes are present. When the power is cut off immediately these joints remain in their current position as soon as the brakes are activated. However, the brakes are connected immediately to a 24 V power supply which results in a delay between cutting off the power of the motors and activating the brakes. In practice the robot will fall down a few centimeters when the power is cut off. How this problem can be solved will be discussed in the recommendations section.

2.3.3. Sensors

As mentioned earlier the system is mostly open loop. Due to a lack of time sensors for real time feedback were not implemented yet. How this could be done will be discussed extensively in the recommendations section.

2.4. Software

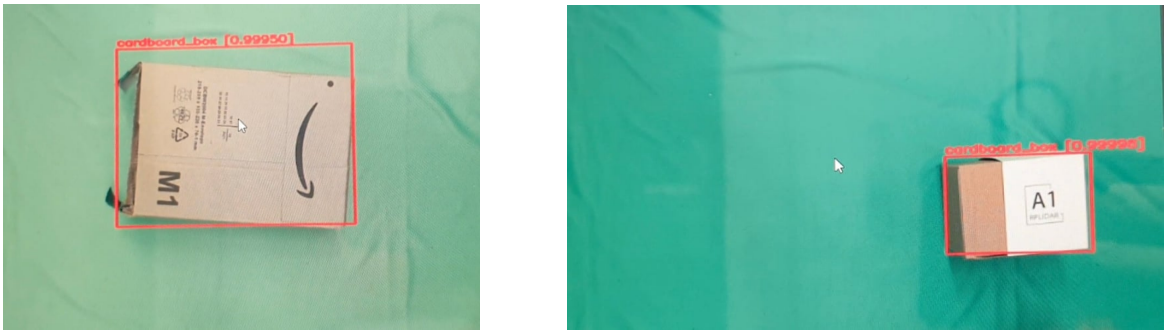
In the first part of this section the package detection part of the software will be explained, while in the second part there will be details on the motion and control side of the software.

The code for package detection is split in 2 parts:

1. package detection from a video, which uses the trained Tensorflow model, presented in 2.4.1
2. package localization from the data from the Lidar, explained in 2.4.2

2.4.1. Package detection from video

The general idea for this part is to train an model on a large dataset images of packages such that it can infer from new images where a package is located. The model creates a bounding box around a package with a relatively high precision and it does this in real time.



(a) *Detected package 1.* **(b)** *Detected package 2.*

Figure 2.49: *Detections of the Tensorflow model.*

Dataset

- about 700 different (image) entries, split into training, test and validation datasets.
- low number of different packages
- same package on multiple positions on the screen
- green screen as the constant background

Technologies

- Labelbox for labelling images in a collaborative environment.
- Roboflow for data processing and augmentation.
- Google Colaboratory for running the training and testing code externally
- SSD MobileNet V2 FPNLite 320x320 pretrained model from Tensorflow model zoo.

How to run the code

Currently the code for package detection from a video (1) is stored in a notebook from the provided handover zip file. The exact location for this file is `Handover_folder_for_Vanderlande/Documents/Package_detection_from_video/Testing.ipynb`. It can be opened using Google Colaboratory with the recommended setup of using a GPU as the hardware accelerator (can be configured in "Resources -> Change runtime type"). The only setup required for running this notebook is to make a folder in the Google Drive account that is being used, called "RoboticsMinor". Next, you can copy `Handover_folder_for_Vanderlande/Documents/Package_detection_from_video/cardboard_320x320_axis_aligned.zip` trained model into the newly created folder, go back to the notebook and at the top left click "Runtime -> Run all".

After some time, at the bottom of the notebook, a live camera feed will appear which shows what the model recognizes as a package. Note that by default the computer might be using the webcam, it can be configured to use an external USB camera from the web browser settings.

2.4.2. Package detection from lidar data

The general idea for this part was to calculate the distances to the points where a package "ends" and to do this, the algorithm looks for a large drop in the distance between the lidar and where the package might be located. By having these 2 points it can also calculate the width of the package, given that the package is axis aligned.

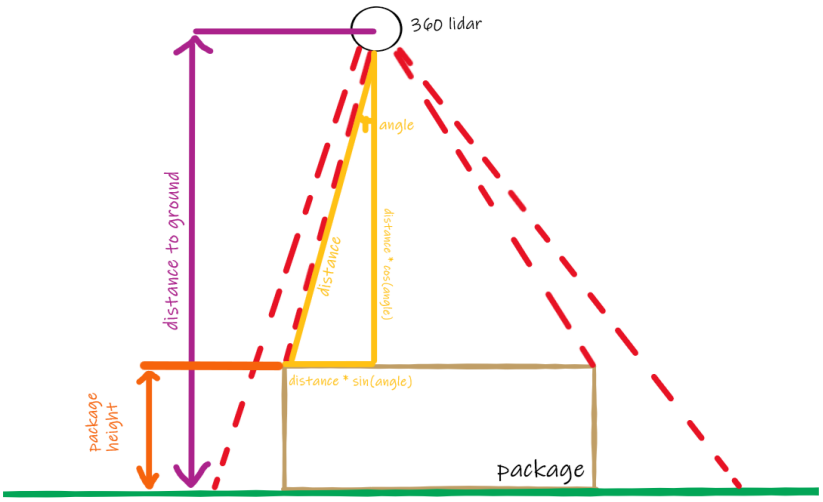


Figure 2.50: *Representation of Lidar and camera data.*

Assumptions

- 1 package at a time
- package is axis aligned
- position of the lidar is fixed

Details on the code

The documentation on how to run this part of the code is in the `README.md` file of the repository at the following location: `vanderlande/ros_ws/src/vision`. Do note that in order to work properly, the lidar needs to be calibrated to the position at which it is located. This only needs to be done if the robot is moved to a new location. To calibrate the lidar, the values at the top of the `vanderlande/ros_ws/src/vision/src/noise_remover.py` file need to be adjusted and there needs to be enough space around the robot such that other objects do not interfere in the lidar reading, more specifically there needs to be no objects on the ground in the angles of interest. These angles can be adjusted using the `MAX_NEGATIVE_ANGLE_OF_INTEREST` and `MIN_POSITIVE_ANGLE_OF_INTEREST`. Some other values of interest are:

1. `DISTANCE_FROM_LIDAR_TO_GROUND` which specifies how far away packages can be located from the lidar, larger values are ignored.
2. `DISTANCE_FROM_LIDAR_TO_GROUND_ON_Y_AXIS` which is used to ignore values that are further away from the ground, but only for the Y axis

3. `DISTANCE_FROM_LIDAR_TO_GROUND_ON_Y_AXIS_TO_SUBTRACT_FROM` which is used to calculate the height of the package, can be adjusted to result in the correct height in case the readings are slightly off.

2.4.3. Robot motion and control

The robot can be controlled by issuing commands through a graphical user interface.

Technologies

- **ROS noetic** as middleware to connect every component with each other
- **ROS control** for robot joint control
- **MoveIt** for motion planning
- **RRTConnect** OMPL path planner
- **TRAC_IK** for inverse kinematics solving
- **Pygame** to program the graphical user interface
- **Gazebo** for simulations

Robot tasks

The robot orders it's tasks as a state machine. With all of the tasks shown in Figure 2.51 with a detailed description in the user manual. The ROS node in charge of managing the state machine and orchestrating the use of other services is the commander node. This node is found at `ros_ws/src/commander/src/commander_node.py`. How to interact with this node through the user interface to control the robot is described in detail on Chapter 5.

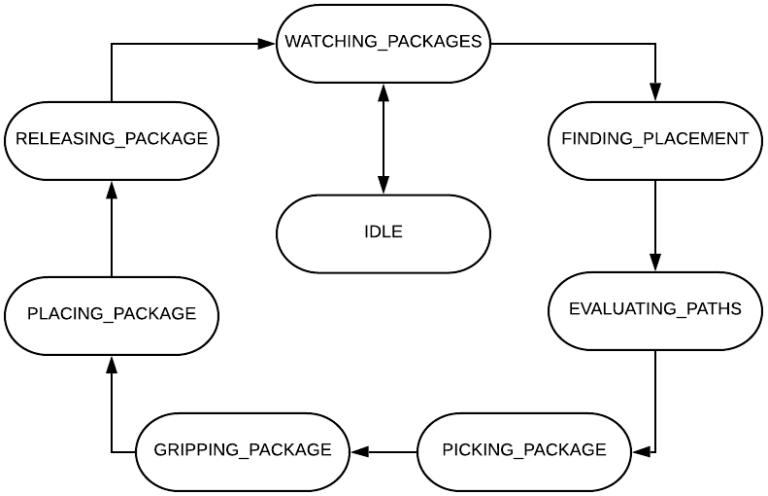


Figure 2.51: *All of the robot states in a pick-place cycle.*

Motion Planning

When the robot commander node wants to make the robot move towards a package or a placement position, it issues a service call to the "dancer" node. The code of this node can be found at `ros_ws/src/dancer/src/dancer_node.py`. This node provides a concise interface for all motion related

tasks. For example, when the commander node has received information that a package is ready to be picked, it issues a call to the pick service offered by the dancer node with the package coordinates. The dancer node then coordinates everything with MoveIt to make the robot arm move towards the package and communicates its results back to the commander node. This allows the commander node to know when it is done and can move to the next state, or if an error is found it can notify the user interface for operator intervention (this mechanism is further described in the User Manual on Subsection 5.3.6.

Stacking Algorithm
Everything related to package stacking is found inside the "stacker" node. The code for this package can be found at `ros_ws/src/stacker/src/stacker_node.py`. The main responsibilities consist of keeping track of the current state of the stack and finding a place for incoming packages. The algorithm used for stacking is a simple greedy algorithm that sweeps through the area of the ULD and places the package in the first place that fits. Collisions are checked between already placed packages and the robot itself. The algorithm also tries to place packages on top of each other if there are already sufficient packages in the current layer. The algorithm can be configured with `WALL_PADDING` and `PACKAGE_PADDING` constants set in the beginning of the file. These two values denote the the minimum distance in meters between the ULD walls and other packages respectively. The current defaults are 9cm and 3cm respectively. This worked well for the demo.

Vacuum Gripper
The vacuum gripper is managed by a simple node that takes in commands through a ROS topic. Just a simple on/off switch. Although it contains additional logic to make sure the gripper also works in the Gazebo simulated environment.

Hardware Interface
To communicate with the actuators all of the necessary data is sent to the Teensy through serial USB. This is done by the hardware node. All related code can be found in the `ros_ws/src/pole dancer_hw` directory. This node executes a hardware read and write loop at a rate of 10Hz. On each loop iteration it obtains all the desired velocities for each joint from the standard hardware interface of ros_control, as well as the current vacuum gripper state and sends it to the Teensy in a serial package. The package structure can be visualized on Figure 2.52. Note how the joint velocities are sent as integers instead of as floating point numbers, this was due to difficulties in serializing and deserializing floating point numbers on the Teensy. This node can also receive a recalibration order, which commands the robot to move itself to the initial position but makes sure to disable all communication with the Teensy so that no velocities are sent while readjusting the internal joint state of the robot.

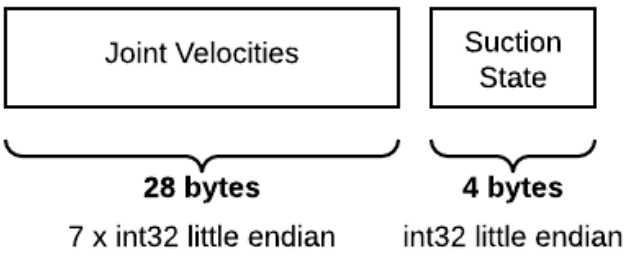


Figure 2.52: *Byte format of the package sent to the Teensy on each hardware loop update.*

3

Recommendations

3.1. Design

3.1.1. Appearence

The robot has a industrial appearance, during the building of the robot surrounding people where intimidated by the looks and feel. The orange colour of Vanderlande that is used in the shells already help to minimize this while keeping the robot professional. But if the goal of our client is to provoke a more safe and human look and feel of the robot, it is best to star looking different at the robot. For example looking at other ways to cover up metal and protruding parts and use more organic shapes to get rid of the harshness of the robot. If the goal of the client is to keep the robot industrial and a little rough, the focus should lay on keeping the metal shaft/beams visible while using the shells to cover up any unsafe parts , like gears and belts.

3.1.2. Safety

The safety is one of the most important things when building a robot for industrial purposes. Due to last minute changes and corona circumstances not all the shells were manufactured. This meant that when the robot is operating many parts, including gears, were reachable. This is dangerous so the recommendation will be to put a shell over all components that can harm the packages and operator. More on this on chapter 3.1.3. The robot uses a virtual and physical emergency stop. Since the robot has a lot of power, to make operating near the robot safer, it is suggested to invest in a light curtain that shuts down the robot immediately when a person comes though the infrared light. This eliminates the chances of the robot coming in contact with a person during operation time. The final safety recommendation is about the control panel. In figure 3.1 you can see the control panel is open and does not have a cover. For safety it is highly advised to build a cover so that the operator etc. do not hit it by mistake.

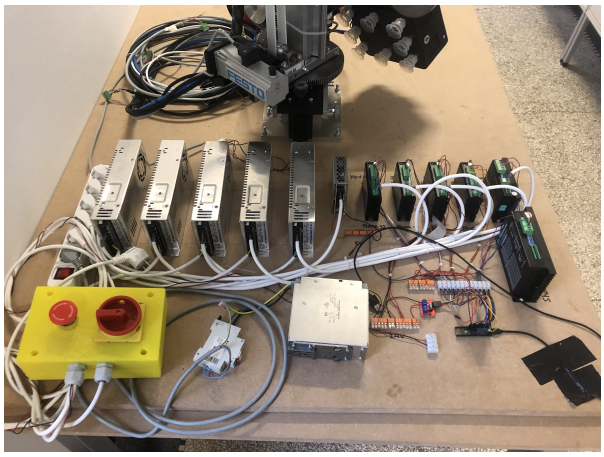


Figure 3.1: *Photo of the control panel.*

3.1.3. Shell

In addition to that, research is done to find the most sustainable material for the shells, PLA was the most feasible material for this project due to time and finances. If there is more budget you can invest in a mould for injection moulding or use the manufacturing method vacuum forming instead of 3d printing. Both of these manufacturing methods can result in a precise and lightweight shell. Even if the material of the shell is not going to change, it is recommended to reprint the shells since some of the shells did not fit due to the slidability of some components that needed to be tuned. It is suggested to measure the physical robot to get a more accurate measurement that the virtual 3d model. These new measurement can then be implemented into the 3d models of the shells and be reprinted.

3.2. Mechanics

3.2.1. Robot placement inside the ULD

As described in Subsection 2.2.1, the robot is, in its current state, bolted to the ULD floor and placed in the middle. This is useful for showing the robot during a demonstration but cannot be easily put into practice. It would require an operator mounting and dismounting it per ULD. This would not only leave a large amount of space in the ULD empty (which would then still have to be loaded manually), but also severely slow down cycle times.

In the initial phase of generating concepts for the "Parcel ULD Loading" project, the robotic pole-dancer was actually generated as a concept with one more degree of freedom: a rail extender. This concept is shown in Figure 3.2. This extending system would have a base mounted to the ground outside the line of ULDs. When a ULD arrives at the robot, the pole, mounted at the end of the extending system atop a wheel, would move into the ULD. It would stop once it reaches the middle of the ULD. It would then allow for a conveyor to be placed over the extending system which would then transport the parcels to the robot. Once the ULD is mostly filled, the rail would retract slightly so that the robot can fill the part of the ULD where it was standing before. This step can be repeated for an optimal number of times until the robot is once again outside the ULD and it is completely filled.

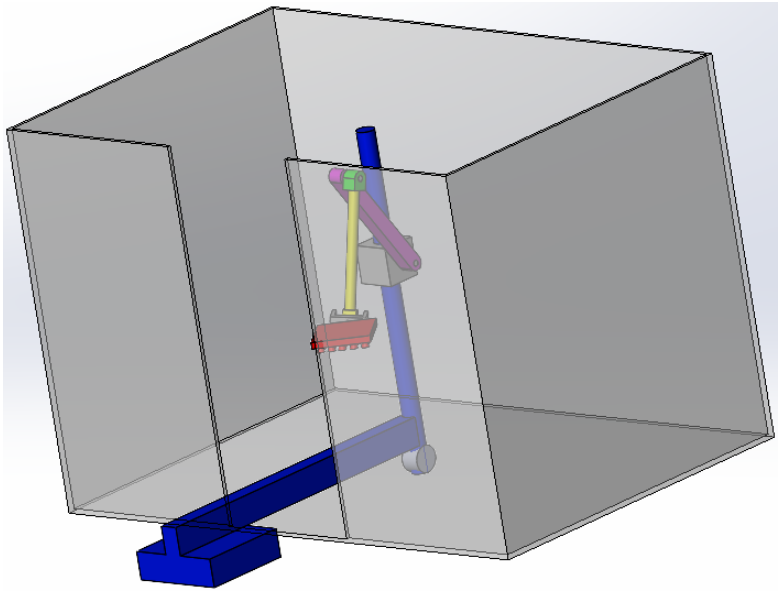


Figure 3.2: *Poledancer robot concept with extending rail system.*

For the Tetris robot to actually be feasible as an autonomous parcel ULD loading robot, this ex-

tending system would have to be designed and build. Most likely this would also require some redesigning of the rest of the system.

3.2.2. Scale

As previously mentioned, the current Tetris robot is a 50% scale model and it can only carry packages of a maximum weight of 3 kg. In reality, it needs to be designed such that it fills a ULD of 3 x 2 x 2 m (width x depth x height) with packages that weigh at most 25 kg. To make these goals feasible for the Tetris robot it needs to be scaled up and most likely require some parts to be redesigned.

One of the most notable components which requires redesigning for this purpose is the lift system. It does not function as intended without the added counterweight. This might be fixed quite simply by redesigning it for a stronger motor or increasing the gear ratio, but that would have to be investigated further in the future.

3.2.3. Gripper

The current suction cup gripper used on the Tetris robot is not ideal. It was designed to be easy to manufacture, robust and serve the purpose for a demonstration of the robotic poledancer concept. It can be majorly improved in one of two ways:

- Redesign and further develop suction cup gripper: The suction cup gripper concept has a lot of potential. Suction cups can grab a large variety of packages and only use little surface contact. This allows them to place packages in smaller spaces compared to, for example, a claw gripper. In combination with an advanced vision system, suction cups also make it possible to re-position packages in an accurate manner when necessary.

  However, the current suction cup gripper would have to be developed quite a bit further. The amount and type of suction cups were chosen quite arbitrarily and could both be optimized by an in-depth research of the packages that the robot needs to be able to grab. It would also be highly advised to implement a system where every individual suction cup can be turned on or off instead of all needing to be on at the same time. This would make it possible to grab more different package sizes in a more efficient manner. Lastly the outer structure of the gripper can be optimized. It is currently around 1 kg and unnecessarily bulky. This was done for simplicity, but if the Tetris concept is developed further definitely an area for improvement.
- Use a completely different gripper: A large upside to the current Tetris robot design is that the gripper is almost completely interchangeable. It would only be necessary to manufacture a new 8 mm diameter shaft of the appropriate length which connects the gripper to the wrist. If future testing shows that a suction cup gripper is not ideal (or if it has to be a completely different suction cup gripper, for example in combination with an adaptive gripper) this could be mounted almost immediately on the Tetris robot.

3.2.4. LiDAR and camera placement

As shown in Figure 2.4, the LiDAR and camera are currently bolted to the wooden structure of the test setup. This structure is not present when actually stacking parcels in a ULD and thus these should be mounted in a different manner. Best would be to include them in the gripper design. This was omitted from the Tetris project due to time constraints, but has a lot of potential for placing parcels with high accuracy.

If it were preferred to not have the camera and LiDAR on the robot, they could be implemented by designing a completely new component. For example, something similar to a large C-clamp could be made. It would need some way to mount the camera and LiDAR to it, but this could be done in a similar manner to the wooden plate. With every ULD, an operator would need to remove this clamp from the previous ULD and mount it on the board above the door of the next before starting

operation. It would not be 100% efficient, but still quick and simple.

3.2.5. Optimize maneuverability

As previously mentioned in Subsection 2.2.1 a "Speed/joint movement analysis" was preformed to determine the joint positions. This method helped to quickly converge to a global design. However, to further optimize the robot this analysis should be done in further detail and with a more detailed model.

For example, it was found that the current design of the lower arm would most likely bump into the top of the ULD in most positions. Part of how this could be solved and the complete robot could be made more compact is by adding precise compact pre-manufactured gearboxes. These were not used in this project due to budget considerations.

In further optimizing the full robotic structure, it is most import to try and make it lighter and more compact. By analysing the different components individually, it will most likely be found that some can be made smaller. This means going through the whole design purely in an attempt to optimize component weight.

3.2.6. Reliability

Due to the large possibility of manufacturing errors and the glued axle-hub connections, the robot is not fully reliable. The same counts for some of the cable management and shell attachments. Before the Tetris robot could be fully put into practice some components may need to be re-manufactured with stricter tolerances, especially at the glued connections.

Secondly, the assembly should be partially redone. The Tetris robot currently is not attached with as many bolts as there are holes. The robot should be fully checked top to bottom, re-tightening all existing bolts and adding those at positions where they are missing.

3.3. Electronics

3.3.1. Electrical Control Box

As can be seen in figure 3.1 all electrical components are mounted on a wooden plate. For simplicity reason (to be able to assemble and disassemble all components easily) the components were spread over a large surface. For a prototype this is of course sufficient. However, several things can be done to improve the electrical control panel:

- The electrical control panel can be made more compact by assembling all electrical components into a professional control box. Note that there should still be a distance between the power supplies and motor drivers for cooling. This distance could be decreased by adding vans to the control box.
- Multiple power supplies were used for supplying 48V to all motors since buying multiple power supplies turned out to be cheaper than buying one power supply with high power output. The power supplies used for this project have an efficiency of 86% which is not superior. This can of course be improved by buying one more expensive power supply with high power output.
- In the current state of the robot all low current components are connected using wires. This could be improved by designing a Printed Circuit Board (PCB) for connecting all low current components and for simple transistor circuits used for turning on and off the vacuum in the suction cup gripper, for example. By designing a PCB the size of the electrical control box could be decreased even more.

3.3.2. Motors

Due to multiple issues involving motor drivers and Festo motors the motor configuration is far from optimal in its current state. The following things can be done to improve the motor configuration:

- At all joints an encoder should be present to enable closed loop feedback for all joints. As mentioned earlier the resolution of the Festo encoders is not matching the resolution the closed loop stepper drivers are expecting. In fact, the resolution of the Festo encoders are 500 pulses/rev and the driver is expecting 1000 pulses/rev. This can by solved by either doubling the frequency of the encoder signal, or by buying a more expensive Festo driver and a Festo Programmable Logic Controller (PLC) for easy control.
- At all motors in the robotic arm brakes should be present to prevent it from falling down when the power is cut off suddenly. Furthermore, the brakes are currently connected directly to a 24 V power supply which deactivates the brakes when the system is turned on. However, in practice there is a small delay between releasing the brakes and powering the motors. Because of this delay the robot arm falls down a few centimeters when it is turned on or off. This can be solved easily by making the brakes software controlled by adding a transistor circuit. This way the brakes can be programmed such that the brake is only released when the rotational speed of the motor is higher than zero.

3.3.3. Emergency stop

Currently the emergency stop only cuts of the power of all power supplies except for the 5 V power supply. Multiple actions can be taken to make the physical emergency stop even more save:

- At first a relay can be added such that a signal will be send to the computer when the power of the motors is cut off. This way the software knows that the physical emergency stop is pressed.
- When the emergency button is pressed the motors will not stop immediately due to discharging capacitors in the power supplies. Using a relay enables the ability to stop the motors earlier by the software. When the software receives the signal that the physical emergency button is pressed all motors should slow down to zero velocity with maximum deceleration. This is considerably faster than the method which is currently used.

3.3.4. Sensors

Currently no sensors except for some encoders are used in the robotic arm. To provide feedback on all joints multiple kinds of sensors can be used. Box recognition using a camera and Lidar sensor was being implemented in the system, but unfortunately this was not finished in time. This concept will be discussed in the next section. The following methods could be used to provide closed loop feedback on the robotic arm:

- Because of the lack of feedback on the positions of the joints calibrating the robot needs to be done manually. This is of course not desirable. This can be improved by adding sensors to the robot such as hall effect sensors and/or Inertial Measurement Unit (IMU sensors). This way absolute positions or angles of all joints can be determined. As a result the robot can be calibrated automatically. Additionally the IMU sensor provides other relevant data such as the acceleration in all directions which also can be used in the software.
- In stead of using simple sensors to calibrate and track the robotic arm cameras can also be used to do this. However, this is a difficult process when looking from the software perspective. But this is an efficient way to do it since cameras are already used to detect the boxes. Additionally, the cameras can be used to provide feedback on how the robot placed the package or when a package falls down. This enables the ability for the robot to correct itself. However, this requires extremely advanced algorithms which will not be implemented in the near future.

3.4. Software

The “vision” side of the robot was not finished during the project due to timing constraints, so this section will start with an explanation on how the package detection software can be integrated with the current robot, and continue with some suggestions of future improvements. Furthermore, this section contains a description of possible improvements for the path planner and the package stacking algorithms.

3.4.1. Integrating package detection with the current robot

The first step in integrating the vision with the rest of the robot is moving the code from the “Testing.ipynb” notebook into local Python (.py) files. This step needs to be taken with caution because the notebook code is a combination of python code and terminal commands, but in theory just running the commands in a terminal locally in the same order as they are in the notebook should work.

After the entire code is moved and is properly working from local python files, a ROS node should be created. A basic setup has been configured for this in the current source code at `vanderlande/ros_ws/src/vision/src/box_detector.py`. This ROS node should call a method and pass a compressed image to the Tensorflow model, which should then determine the location of the package using the `get_coords(image_np,detections)` method. This method returns the top, left, bottom, right pixel coordinates, which can be used to determine the bounding box of the detected object.

Assuming everything works, the next step is to merge the data from the `box_detector` node with the data from the `noise_remover` node, which is the node responsible with making sense of the lidar data.

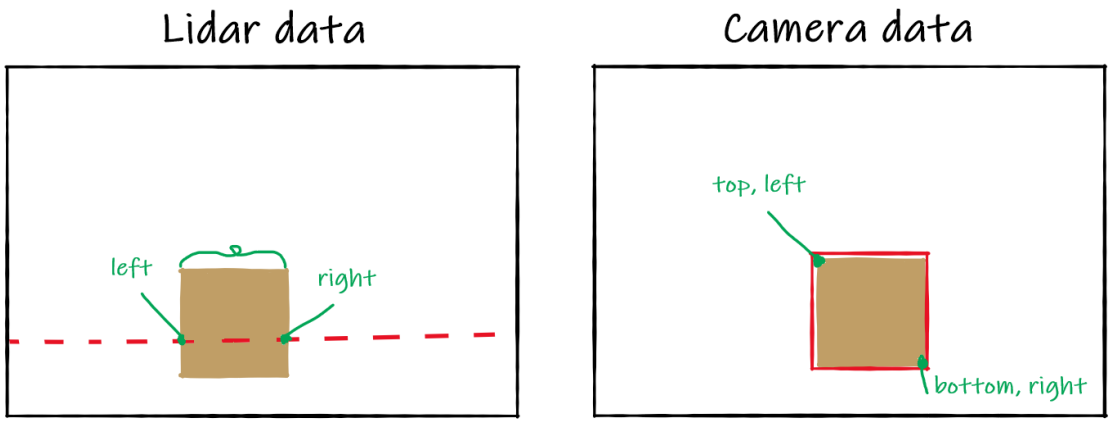


Figure 3.3: *Representation of Lidar and camera data.*

Besides what is represented in the image, the lidar data also returns the height of the package (from the ground). To calculate the 3rd dimension (the depth, i.e. from top of the screen to bottom of the screen) the camera data can be used to approximate the width in pixels, right - left for axis aligned packages, as well as the “depth” (which from the point of view of an 2d image would be the height of the box, but let’s stick to “depth” for consistency), top - bottom. These values can then be used to compare the width in pixels with the actual width returned by the lidar, and calculate the depth using the simple rule of 3. At this point the robot would know all 3 dimensions of the package as well as its position, this can be further passed on to the `vision_node` which communicates with the path planning algorithm.

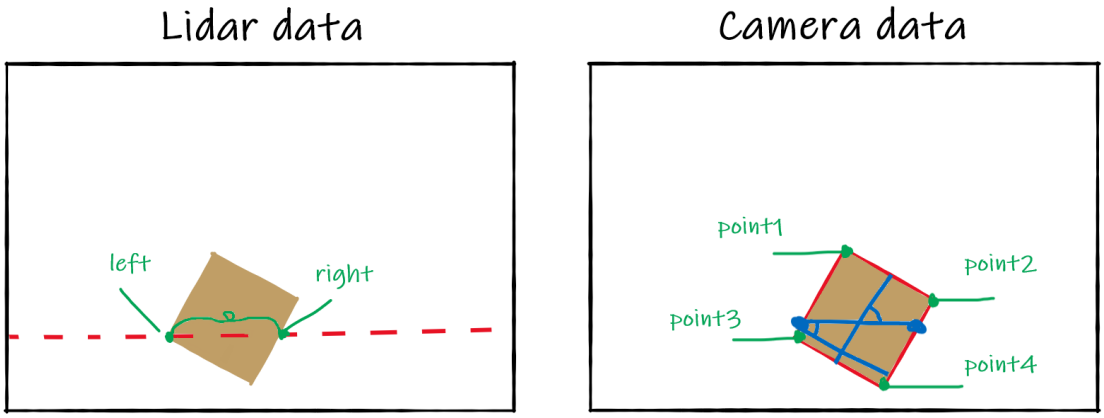


Figure 3.4: *Rotated package representation of Lidar and camera data.*

This can be further extended after the package detection model is improved to detect masks instead of axis aligned bounding boxes. The math is not that straight forward though and has been outside of the scope of this project, but the general idea can be visualized in the previous image.

3.4.2. Enhancing the localization of packages

The package localization system could be significantly improved in terms of accuracy and flexibility by providing more resources to the following aspects:

- **Data.** During the project the training and testing data was limited due to resource and timing constraints. Generally, models use comparatively more data, and very complex models can reach up to 2 million data points (in this case images). In a large company such as Vanderlande this can be significantly improved because Vanderlande has more resources and a larger workforce than a group of students. The training data should be as varied as possible, so firstly the model requires more images that contain packages of different shapes, sizes and colors. Secondly, some of the images should contain other objects that might appear in an image, not only packages, such as for example hands, the robot, etc. This is meant to improve the accuracy of what the model recognizes as a package and what it shouldn't consider a package, and is especially helpful when the robot arm goes in front of a package. Another important point to consider is training time, the current model was trained on 10 thousand steps, which itself is a large number, and it took only 2-3 hours to train. The time will obviously increase with more training images, but the training steps should also be increased, even up to 100 thousand steps.

- **Mask R-CNN.** Currently the result of the detection of packages is a group of axis aligned bounding boxes. The Mask R-CNN technique would have a large impact on rotated packages and objects that are not of rectangular shape. This method finds the contour around an object instead of finding the bounding box. For the rotated objects it would allow the robot to make more accurate movements in the wrist when picking up packages, and for the irregularly shaped packages it would allow the gripper to know which suction cups it needs to turn on in order to not lose pressure. This method does require significant work to setup though, firstly because instead of just making 2 clicks when labelling, the worker would need to click at least 4 times to mask around the rectangular objects, and even more times for uneven shaped packages. This significantly increases the labeling time. On a similar note, the Mask R-CNN technique also requires more time when training, for example, the current setup took on average 22 ms to train on 1 image, while with the new setup it would take 301 ms to train on

an image, which is a considerable difference and it will largely affect the training time. Even though this method has some disadvantages, the final result could be considered as a worthy trade-off due to the accuracy and flexibility improvements.

3.4.3. Path planner

The robot currently uses MoveIt for motion planning. This is working quite well but there are a few problems:

- **Speed:** For some poses the path planner takes up to 10 seconds to find a path, this is definitely not ideal. To fix this a few things could be done. First would be to simplify the STL meshes of the robot model, this will reduce the number of collision calculations that MoveIt has to perform to evaluate the paths. Also other path planners could be tried. Right now the robot is using RRTConnect, an OMPL planner, which works well and is highly recommended by most experts but there could be other, more optimal planners such as CHOMP. This has to be benchmarked.

- **Accuracy:** The path planner currently has a goal tolerance that deviates +-2cm from the desired position. This parameter can be configured at `ros\_ws/src/dancer/src/dancer\_node.py` by changing GOAL_TOLERANCE. But setting this value any lower increases the time it takes for the path planner to find a solution, so this is a trade-off between accuracy and speed.

- **Real Time:** Right now the robot detects a package and assumes that the package will not move while it is being picked up. This is ok, but if in the future the package must be picked up from a moving conveyor belt then the current implementation won't detect that motion. The robot also assumes that the stack of packages cannot suddenly collapse, this is a relatively safe assumption to do but in case any package falls while the robot is moving there will be a mismatch between the world and the robot's perception of it. This will result in faulty package positioning and collisions.

3.4.4. Package Stacking

Since the mechanics of the robot movement were a more important priority for the project than finding optimal package stack, not as much energy was placed into this area. The current stacking algorithm is really simple, a brute force search that sweeps through the ULD for a place to put the next package where there are no collisions between other packages and the robot area itself. The search starts on the inside right corner of the ULD and moves to the left. It can also be configured to not look for package placements on left side of the ULD to avoid collisions with the counterweight used for the real demo (this is the default setting) Some padding is also added to compensate for slight inaccuracies in the robot's motion. It also attempts to place a package on top of another when it finds no collisions and when there are enough packages placed on a layer.

The problem of finding a stack of packages, formally known as 3D bin packing, is not an easy problem. Classical algorithms are probably not a great solution for a problem with such a large search space. Deep reinforcement learning seems like a good choice here. Robots can learn through experience and build an intuition of how to stack packages properly. A promising paper showcasing a deep reinforcement learning algorithm outperforming human operators on package stacking can be found here [2].

4

Prototype - Pros and cons

In this chapter the some of the pros and cons of the current system are presented based on the different categories they are part of.

4.1. Design

Pros

- Industrial look and feel, which was the mail goal
- Lightweight shells because of the use of 3d printing with PLA
- The robot is completely unique, and therefore there is no competition on the market

Cons

- Can seem intimidating for users
- Is not yet safe enough to operate, due to the incomplete set of shells
- It is time intensive to make because of all the unique manufactured parts

4.2. Mechanics

Pros

- Fully developed and integrated prototype for demonstration
- Capable of moving with 7 degrees of freedom, including the possibility to move past itself
- Can reach all positions within the ULD
- Can move even faster than the desired 9 seconds/parcel (exact possible speed unknown due to lack of testing)
- Fully functioning gripper that can grab a large variety of parcels

Cons

- Counterweight and ULD structure currently needed for successful operation
- Some parts were manufactured with too lenient tolerances
- Undesirable LiDAR and camera placement; currently requires the wooden ULD structure for use
- Missing nuts and bolts in some parts of the assembly
- Some gearboxes slip slightly due to imperfections in the design, part manufacturing and assembly

- Has to be mounted inside the ULD by a team of operators (too heavy and dangerous to be done by one person)
- Gripper is heavy and currently only capable of turning on all suction cups at once

4.3. Vision software

Pros:

- Whenever a package is in the video the accuracy is over 90%.
- Works in real-time, i.e it can be used on a video without having much delay.
- Training was relatively fast, a couple of hours for around 10,000 training steps and 500 training images.

Cons:

- Detects other objects as a package, e.g. hands or intense light on the screen.
- For some packages the bounding box isn't perfect and can be slightly larger or smaller than the package.
- Works for packages slightly tilted, but has issues when it approaches the 45 degree rotation (approximately 30-35 degrees max rotation works).

4.4. User Interface

Pros:

- Easy to use.
- Simplified interface for everything needed.
- Fast emergency stop button.

Cons:

- Lacks detailed graphs of joint states and other helpful debugging information.
- Not integrated with the vision yet.

4.5. Path planning

Pros:

- Finds a compact path to any position and orientation in the ULD.
- Avoids collisions with other packages.
- Stays within the physical velocity limits of each joint.

Cons:

- Slow computation of paths due to complex STL meshes of the model, could be easily simplified.
- Sometimes finds slightly unnatural paths that look a bit awkward.

5

User Manual

5.1. Physical Control Panel

With the physical control panel the robot can be turned on and off easily. By putting the plug into the power socket and switching the on/off switch, power is supplied to all electrical components and the robot is ready for use. Note that the emergency button on the physical control panel is only used in real emergencies since it cuts of the power of the whole robot immediately which can cause the robot arm to fall on the ground due to the lack of brakes at some joints. Using the emergency button in the software is always preferred. After turning on the robot the next step is to connect the computer to the micro controller using a micro USB cable. This will be explained extensively in the next section.

5.2. Software setup

To run the software of the robot a computer with the following specs is required:

- Ubuntu 20.04 (or any other linux distro that supports ROS noetic)
- A screen (required for the graphical user interface)
- A mouse
- A keyboard to easily go through the installation process
- A fast internet connection to install everything
- A USB Type-A to micro USB cable

The first step requires installing ROS noetic. This can be easily done by following the tutorials over at http://wiki.ros.org/noetic/Installation. It is recommended to run the Desktop-Full Install to be able to run the simulations as well.

5.2.1. Enabling the user to use serial ports

To allow the Ubuntu user to read and write to serial ports (for communication with the microcontroller) it has to be added to the dialout group. To do this execute the following command in a terminal window:

```
1 sudo adduser $USER dialout
```

After doing this, sign out and sign back in.

5.2.2. Installing ROS packages

The next step is to install the necessary ROS packages used by the robot. These can be easily installed with the following command:

```
sudo apt install \
    ros-noetic-joint-state-controller \
    ros-noetic-position-controllers \
    ros-noetic-joint-trajectory-controller \
    ros-noetic-gazebo-plugins \
    ros-noetic-moveit \
    ros-noetic-trac-ik-kinematics-plugin
```

Also the catkin tools package is required which can be installed through pip3:

```
sudo pip3 install -U catkin_tools
```

5.2.3. Building

The “vanderlande” directory in the handover bundle contains all the code necessary to run the robot. This directory must be copied to somewhere in the computer that will be running the code, for example in the Documents directory.

With an open terminal window, cd into vanderlande/ros_ws and run the following commands to build the project:

```
catkin build
```

5.2.4. Simulation

Everything is ready now to run the simulation. This simulation does not interact with the actuators. It is just to be able to visualize the motions of the robot in a safe, virtual environment.

Before running any ROS commands, open a terminal window and navigate to vanderlande/ros_ws and run the following command:

```
source devel/setup.bash
```

Now to run the simulation it is as simple as running:

```
roslaunch poledancer gazebo.launch
```

3 different windows should open up. The GUI, Gazebo with the simulated robot and RVIZ.

5.2.5. Running the software for the real robot (not simulated)

To actually run the robot connected to the actuators another launch file has to be used. But before that the computer running the software has to be connected to the Teensy via micro USB. To verify that the connection to the Teensy is good run the following command:

```
roslaunch poledancer robot.launch
```

The output of that command should be a line containing somewhere “ttyACM0”

With the serial connection to the Teensy ready (and the robot powered on and fully connected), the software can finally be launched. To do this run the following command:

This command should only launch a GUI window as well as RVIZ.

5.3. Graphical User Interface

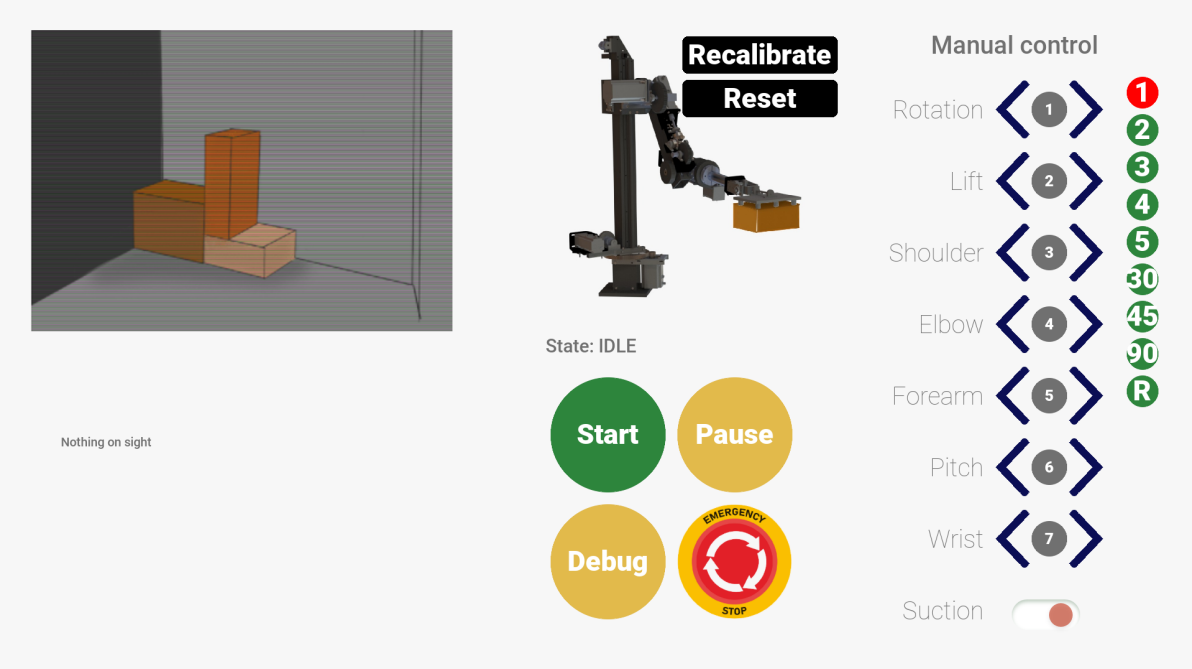


Figure 5.1: *Graphical User Interface.*

5.3.1. Manual Control

The manual control panel on the right side of the user interface allows the operator to move each joint individually to make sure that each joint is working properly and also to recalibrate whenever necessary (Calibration is further explained in the subsection below).

To move a joint manually one must first set the step size by clicking on one of the green buttons numbered from 1 to 90. This number represents the size of the step in degrees. The currently selected step is denoted by the button currently in red.

The last green button labeled “R” will move the robot to a random joint state and should NOT be pressed before calibrating the robot. This button is nice to use for showcasing the robot’s mobility.

With the step size selected, any of the left and right blue arrows can now be pressed and the corresponding joint should move counterclockwise or clockwise by the given step. The robot should NOT receive manual commands while it is moving, this will interrupt it’s current motion plan and might cause states to be skipped.

This panel also allows for the manual control of the suction gripper, with a switch on the bottom that denotes off when red and on when green. Clicking it will toggle the state of the suction.

5.3.2. Velocity Limits Configurations

All the maximum velocity limits of each joint can be found at `ros_ws/src/poledancer_moveit/config/joint_limits.yaml`. The `max_velocity` parameter of each joint should not be changed as it has been calculated by the mechanical engineers. But one can adjust the overall scale of the velocities by changing the `default_velocity_scaling_factor` parameter. For the demo a maximum scale of 0.50 was tested and worked properly. Anything higher than 0.50 caused missing steps on the motors which resulted in de-calibration.

5.3.3. Calibration

Before starting the robot it is important to go through the calibration process. This step is not necessary when running the robot in the simulation.

Due to the lack of encoders and other sensors, the robot is not really aware of its joint positions. So the operator must make sure that each joint in the real world has more or less the same value in the robot's software. One can easily see the robot's current perceived position through RVIZ as shown in Figure 5.2.

To calibrate the robot, one must use the Manual Control panel to manually move each joint of the real robot towards the initial position shown in Figure 5.2. When this joint state is reached, the "Recalibrate" button in the user interface has to be pressed. This will set the internal joint representation to the initial state without sending commands to the actuators. You should be able to see the robot joints readjusting in RVIZ.

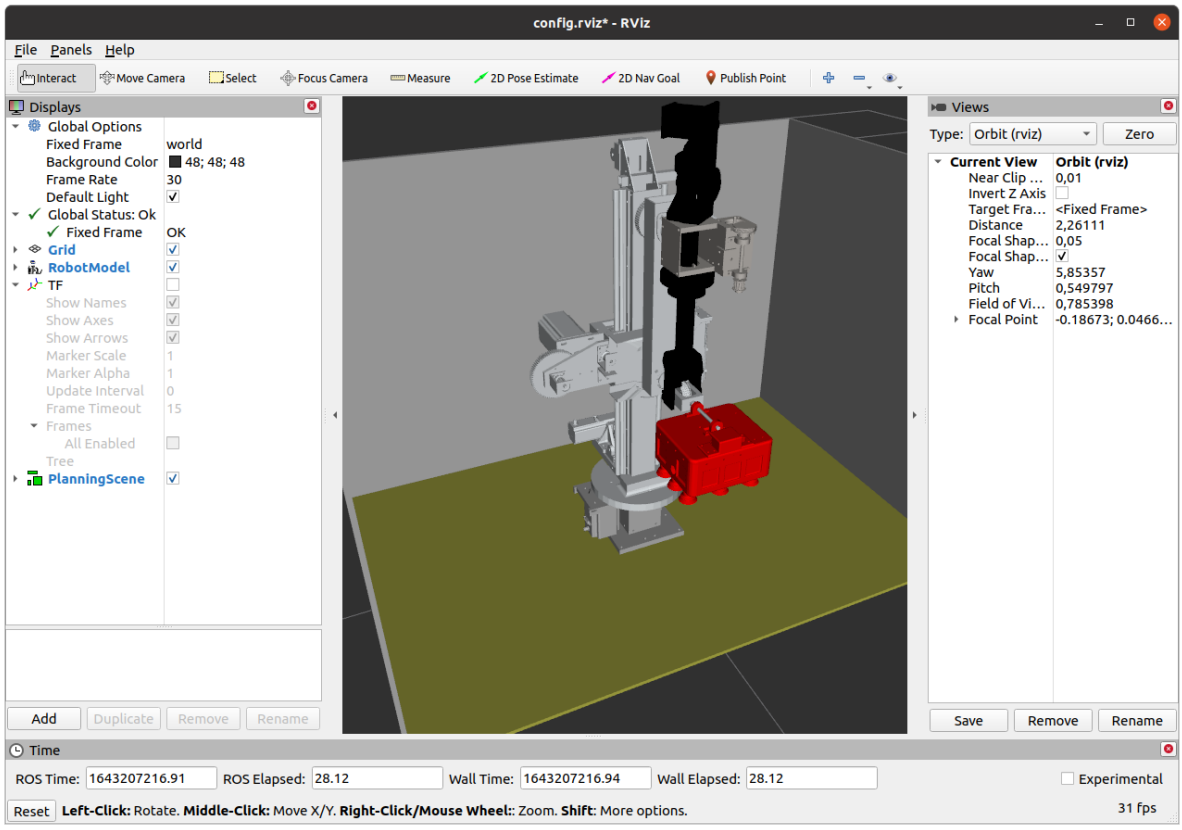


Figure 5.2: *RVIZ with robot at initial position.*

5.3.4. RVIZ

The RVIZ window contains valuable information about the current internal state of the robot. Everything seen in RVIZ is what the robot thinks the world looks like right now. Green boxes denote obstacles such as the packages to pick and place. A purple box means a package that the robot currently is gripping. This is illustrated in Figure 5.3

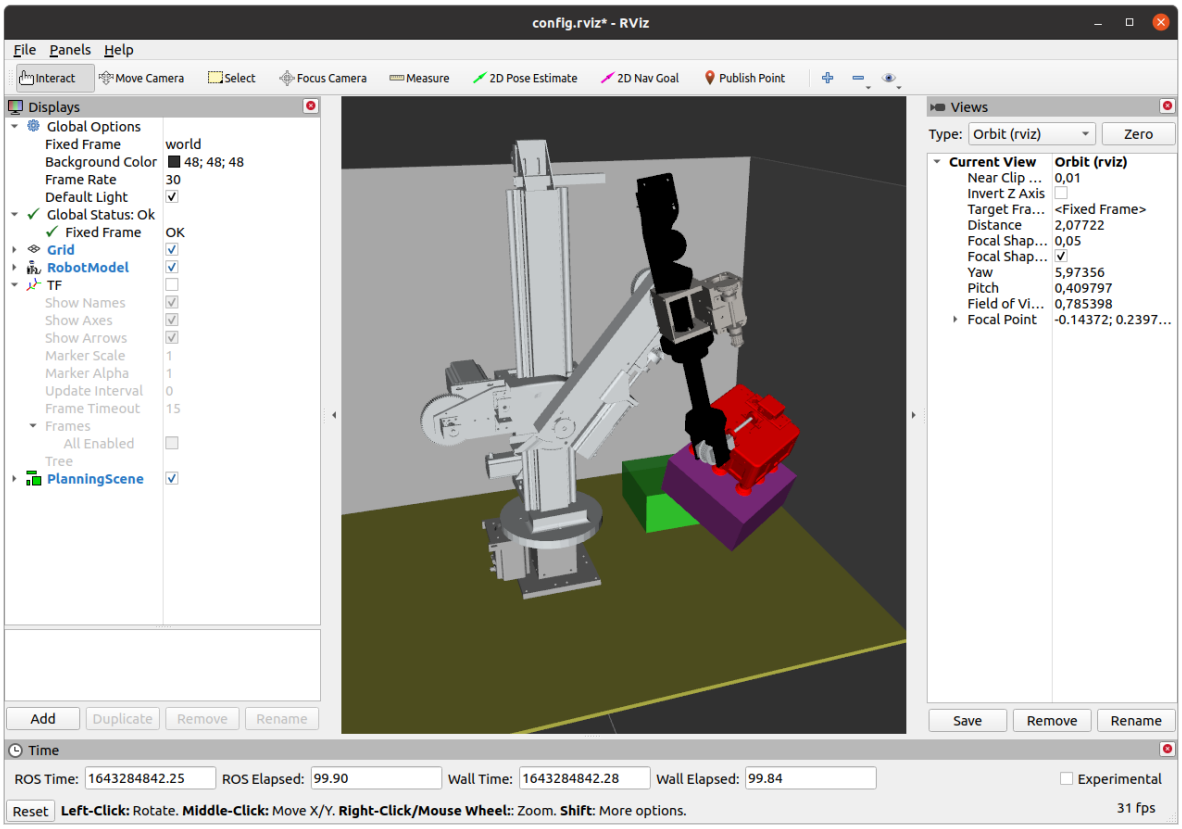


Figure 5.3: *RVIZ window of robot placing package. Green = obstacle, Purple = gripped box.*

5.3.5. Robot Commands

The section of the user interface next to the manual control panel is the robot command panel. Here we see a nice image of the robot and 6 very important buttons. “Recalibrate”, “Reset”, “Start”, “Pause”, “Debug” and “Emergency Stop”. The Recalibrate button has already been explained in the previous section so it won’t be explained again here. One can also see a “State” text in the middle showing the current state of the robot. This State is helpful to find out what exactly the robot is busy with right now.

States

Before issuing commands to the robot, it is important to understand each of the robot states.

- **IDLE:** The robot is not currently started and is not doing anything, in this state the robot is safe to receive manual control commands.
- **WATCHING_PACKAGES:** The robot is looking for packages to pick. (Since the vision system was not complete and integrated with the robot, in this state the robot will simply spawn a random package after a random delay in its representation of the world)
- **EVALUATING_PATHS:** The robot is looking for a feasible motion path to pick up the package.
- **PICKING_PACKAGE:** The robot is executing a motion path to move the gripper close to the package to pick it up.
- **GRIPPING_PACKAGE:** The robot is turning on the suction to grip the package.
- **FINDING_PLACEMENT:** The robot is searching for a valid position in the ULD to place the package.

- **PLACING_PACKAGE:** The robot is executing a motion path towards the placement position of the package.

- **RELEASING_PACKAGE:** The robot is turning off the suction to release the package.

Start

The Start button commands the robot to begin stacking packages. This moves the robot from an IDLE state into a WATCHING_PACKAGES state and the pick-place cycle is executed until it is stopped again. After clicking this button, it will toggle itself into a red “Stop” button. This “Stop” button can be clicked to stop the robot from performing any more pick-place cycles. So the robot will not immediately stop what it is doing, but after finishing the RELEASING_PACKAGE stage it will go back to IDLE and wait for new commands.

Pause

The Pause button is a helpful command to make the robot pause before starting the next stage. This will also toggle the Pause button into a Resume button which can be clicked to allow the robot to continue to the next state. For example, if the robot is currently EVALUATING_PATHS, clicking on the pause button will make the robot wait for a Resume command before going to the PICKING_PAGE state.

Debug

The Debug button works together with the Pause button. When Debug is on, the robot will by default pause on each state. This is useful to run the robot in a careful and controlled manner.

Emergency Stop

This button will cancel all current motions being executed by the robot and automatically pause the cycle. This button usually responds in less than 500ms, but it relies on the software working properly and a good connection to the Teensy. In case this button is not working properly then the physical Kill Switch should be used.

Reset

The reset button will abort the current cycle and move the robot back to the home position. The suction will remain On if it was already On to avoid any damage from dropping the package.

5.3.6. Error Handling

Whenever the robot encounters a known error it will display a popup on the user interface for the operator to handle. The popup is very simple, it consists of an error message and two buttons, “Retry” and “Cancel”. The Cancel button is equivalent to the Reset button, it will cancel the current cycle and bring the robot back to IDLE. The Retry button attempts to run the current stage of the robot again. For example on Figure 5.4 an error popped up saying that the robot was unable to find a path to the target position, this sometimes happens for difficult to pick packages. Clicking on Retry will make the robot reattempt to find a valid motion path.

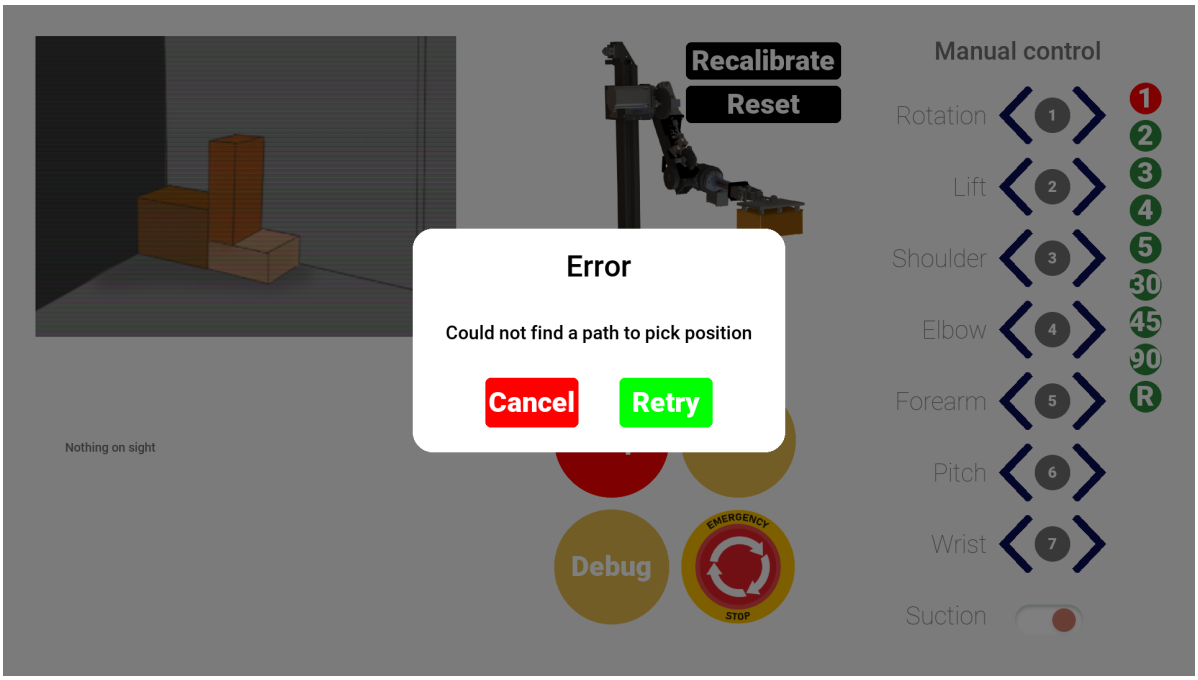


Figure 5.4: *User Interface showing an error.*

Bibliography

[1] Vanderlande, “Project 10: Parcel uld loading.” Minor project description, 2021.

[2] H. Zhao, Q. She, C. Zhu, Y. Yang, and K. Xu, “Online 3d bin packing with constrained deep reinforcement learning,” in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 741–749, AAAI Press, 2021.