Miro Tuovinen, Jori Anola-Pukkila, Eetu Vehnämäki, Josia Orava

Innovation Project, Garage Club, Autumn 2024

Metropolia University of Applied Sciences

Bachelor of Engineering

Smart IoT

Innovation Project Report

9 December 2024

Abstract

Authors:	Miro, Jori, Eetu, Josia
Title:	Innovation Project
Number of Pages:	45 pages
Date:	9 December 2024
Degree:	Bachelor of Engineering
Degree Programme:	Information & Communication Technology
Professional Major:	Smart IoT
Supervisors:	Saana Vallenius, Lead Teacher Niko Hutri, Project Manager

---

In this project, the team designed and built a competitive robot for the Robosota event as part of the Innovation Project course. The robot utilized a 3D-printed custom-designed case and wheels. As its weapon it used a vertically spinning shuriken to flip opponents and make them unstable from the impact.

This report describes the design, construction, and testing of our robot in a step-by-step manner: the problems that were encountered, solutions found, and results obtained.

Contents

List of Abbreviations

CAD: Computer Aided Design. Use of computers in various design aspects by means of CAD software.

BIM: Building Information Modeling. Provides digital representation of physical buildings.

ESC: Electronic Speed Controller. Electronic circuits used to control the speed of electric motors.

PLA: Polylactic acid. Thermoplastic polyester obtained by condensation of lactic acid. Biodegradable. Also referred to as polylactide. Commonly used as 3D printing material.

PLA-CF: Polylactic acid - carbon fiber. Polylactic acid with chopped carbon fiber added. Used in 3D printing.

ABS: Acrylonitrile butadiene styrene. Thermoplastic polymer made by polymerizing styrene and acrylonitrile in presence of polybutadiene. Commonly used plastic. Used in Legos, electronics enclosures, 3D printing etc.

TPU: Thermoplastic polyurethane. Plastics with elastic properties. Used in various applications, including 3D printing.

TPC: Thermoplastic copolyester. Plastics with elastic properties. Used in various applications, including 3D printing.

PA: Polyamide. Polymers that occur naturally and artificially. Synthetic polymers include nylons that are used in various applications, including 3D printing.

LIDAR: Light detection and ranging. Used to determine distance with measuring time for the laser to be reflected back to the receiver. Can be used for surveying, seismology, laser guidance etc.

Li-ion: Lithium-Ion. Rechargeable battery technology using reversible intercalation of Li^+ ions into conductive solids for energy storage.

Li-Po: Lithium-Polymer. Rechargeable battery technology of lithium-ion technology using a polymer electrolyte instead of liquid electrolyte.

Li-Fe: Lithium iron phosphate. Same as $LiFePo_4$. Rechargeable battery technology of lithium-ion technology using lithium iron phosphate as cathode and graphitic carbon electrode with metallic backing as anode.

NiZn: Nickel-Zinc. Rechargeable battery technology.

NiMh: Nickel-Metal hydride. Rechargeable battery technology.

PS4: Playstation 4 game console.

MAC: A Medium Access Control (MAC) address is a unique identifier assigned to a network interface controller.

LED: Light-emitting diode.

G-Code: The most widely used 3D-printing programming language.

1 Introduction

During the start of the innovation project course we were shown some possible projects to join in through Metropolia, one of which was Garage Club which had a few options to choose from, out of the options we were most interested in drone course development which would have included building your own drone, completing a drone piloting license and then making the future drone courses better with our feedback. The other option we were interested in was joining the Robosota competition with a bot we would make ourselves, this is the option we ended up choosing and we decided early on we would make a 450g robot for it since it was our first time building one, other options would have been 150g or 150g plastic league.

1.1 Garage club

After finding out what Garage Club was, we contacted Garage Club host Fayez Bassalat about how to proceed on collaborating with Garage Club on the project, Niko Hutri replied to us and invited us to join their weekly meeting to find out more about Garage Club. We went there and set up a proper meeting to start our project for the next week. On the first actual project meeting we were told how Robo Garage works, we had a budget of around 500€, we could buy parts that were bought from outside the places Garage Club ordered their parts from and would get reimbursed for them afterwards, Niko gave us an assignment for the next week of having a project plan made as a document and showing it, along with preparing to use a modified SCRUM by selecting one person as a SCRUM master, all Garage Club projects use SCRUM as a requirement.

1.2 SCRUM

Josia Orava volunteered to be our SCRUM master, which made him responsible for managing the SCRUM tasks, we used the Miro web application as a platform for our SCRUM whiteboard, which consisted of a kanban board which had all our tasks and an active board for the current week long sprint

tasks. We would move tasks from “Not started” to “In progress” when we started working on them and moved them to “Finished” after we had completed them, the requirements for completion were the task being actually completed, pictures/videos and a short documentation in our teams repository. Every week we would have a meeting with Niko Hutri and Fayez Bassalat where we showed our progress through the SCRUM board and our teams. One person would take notes of the meeting, the person responsible for that was rotated on a weekly basis.

1.3 Robosota

The Robosota event was to held in Robo Garage on the 23rd of November, which was around 3 months from the start of our innovation project course, registration for the event would end on the 10th day of November, there were a few weight classes to choose from, 150g, 150g plastic league which required all parts except electronics to be made out of plastics and the 450g league which we opted to join because we felt it was more forgiving of a weight class for first timers, the bots would have a few rules applied to them which limited some build options for the robot which we will go over in the next part.

1.3.1 Robosota competition rules

Maximum weight of the robot is 150g or 450g depending on the league, maximum allowed length is 30cm, maximum circumference speed of the weapon is 100m/s, except in the 150g plastic league where it is limited to 20m/s.

The frame cannot be made from flammable materials (Wood is not considered flammable). Ranged weapons are prohibited which include radio interference, shooting/projectiles, liquids, powders, electricity, laser, light and radiation, heat and fire, intentional entanglement is also prohibited.

Permitted power sources include NiZn, NiMh, Li-ion, Li-Po and Li-Fe batteries, as well as alkaline batteries, internal combustion engines are prohibited, electric, pneumatic and hydraulic motors are allowed. Pneumatics, hydraulics, springs and mechanically stored energy devices have their own rules but are generally allowed.

All robots must have a power switch included and an external indicator light for the power being on. All robots must have a failsafe for shutting down the robot's drive and weapons if signal is lost and the weapon must be able to be mechanically locked when not in combat.

Robot maximum weight is determined by the category, there is no minimum weight. Robots are weighed in their battle-ready state and all alternative parts must fit the weight limit; spare identical parts are not counted as alternative parts.

You can get a weight advantage if your robot has any of the following: crankshaft mechanisms, vibration-powered, gyro walkers, and hovering robots receive a weight advantage of +50%. Two degrees of freedom walking robots receive a weight advantage of +100%.
Autonomous robots where the intelligence and sensors are internal get a weight advantage of +100%, if the intelligence and sensors are external, for example, a laptop, the advantage is +50%, autonomous robots must be stoppable via remote control and the weight advantages cannot be combined.

Robots can be multi-part and there is no limit on the number except that they must all fit the starting box.

For control methods, radio control, bluetooth and wifi are allowed, radio frequencies are allowed within the legal limits, analog frequencies are prohibited along with controlling via infrared, ultrasonic, wired and visible light.

2 Frame Design

The majority of the robot, outside of electronics and the weapon, was manufactured with 3D printing technology. The outer frame, wheels, bearing hubs and everything inside the robot was designed by our team, except the N20 driving motor upper mounts. The 3D printing materials (filaments) and the machines themselves were provided by the Robo Garage.

The aforementioned parts were designed using a free CAD program called FreeCAD 0.21.1, which is open-source general purpose parametric 3D CAD BIM software.

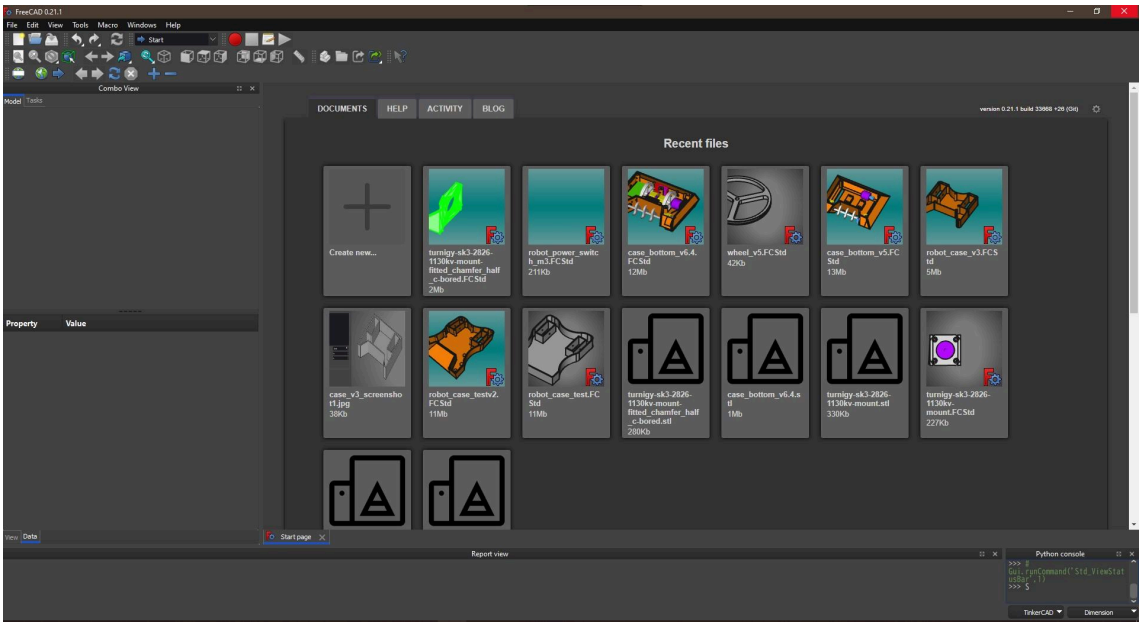

Figure 1. FreeCAD 0.21.1 Start Page.

2.1 Design Process

The design process of the main frame was multifaceted, as there were many competition rules to be followed and design features to be implemented, some of which provided unique challenges in the process.

The team started with the sketching phase, and multiple unique sketches were produced.

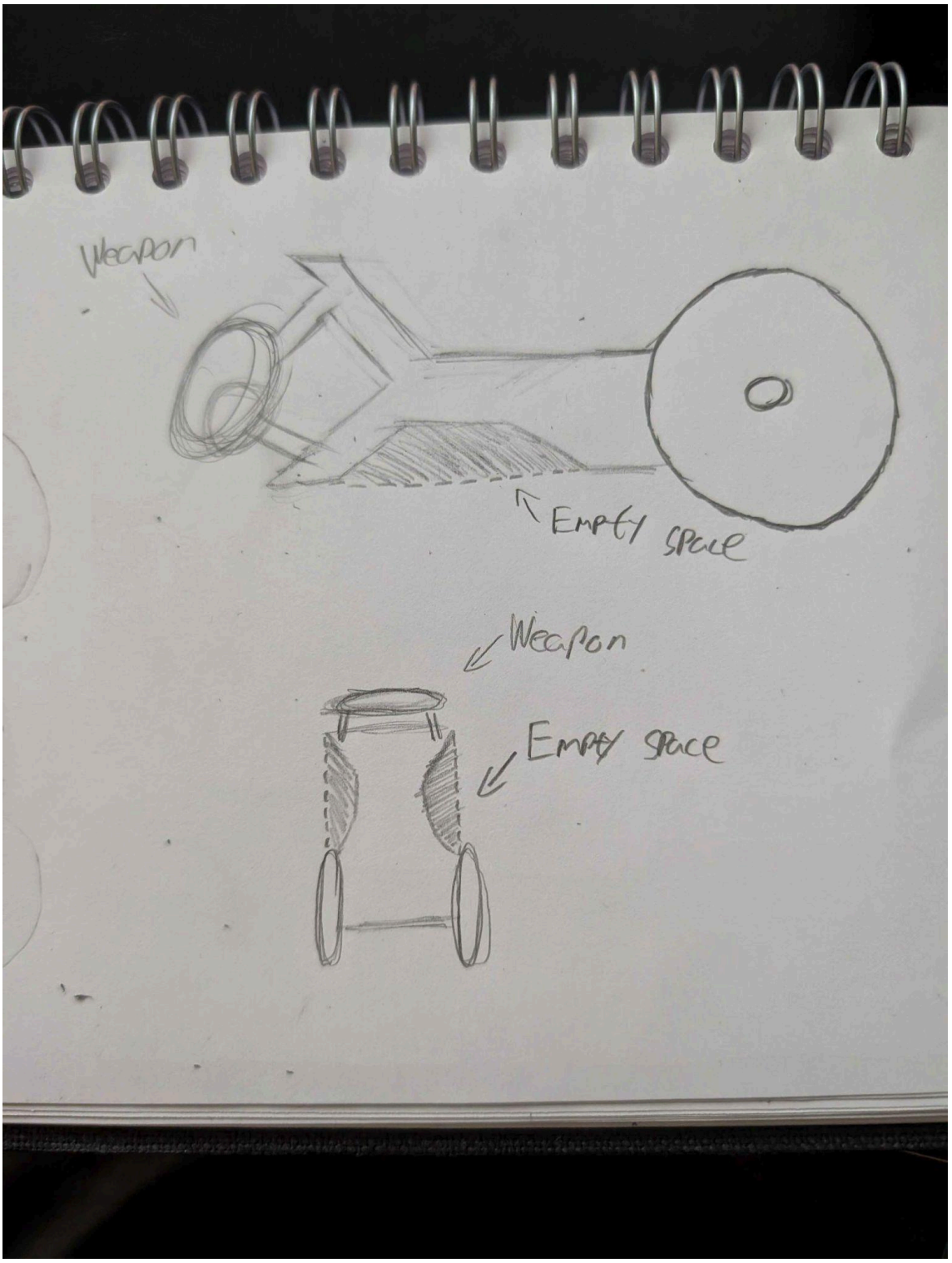

Figure 2. The first draft.

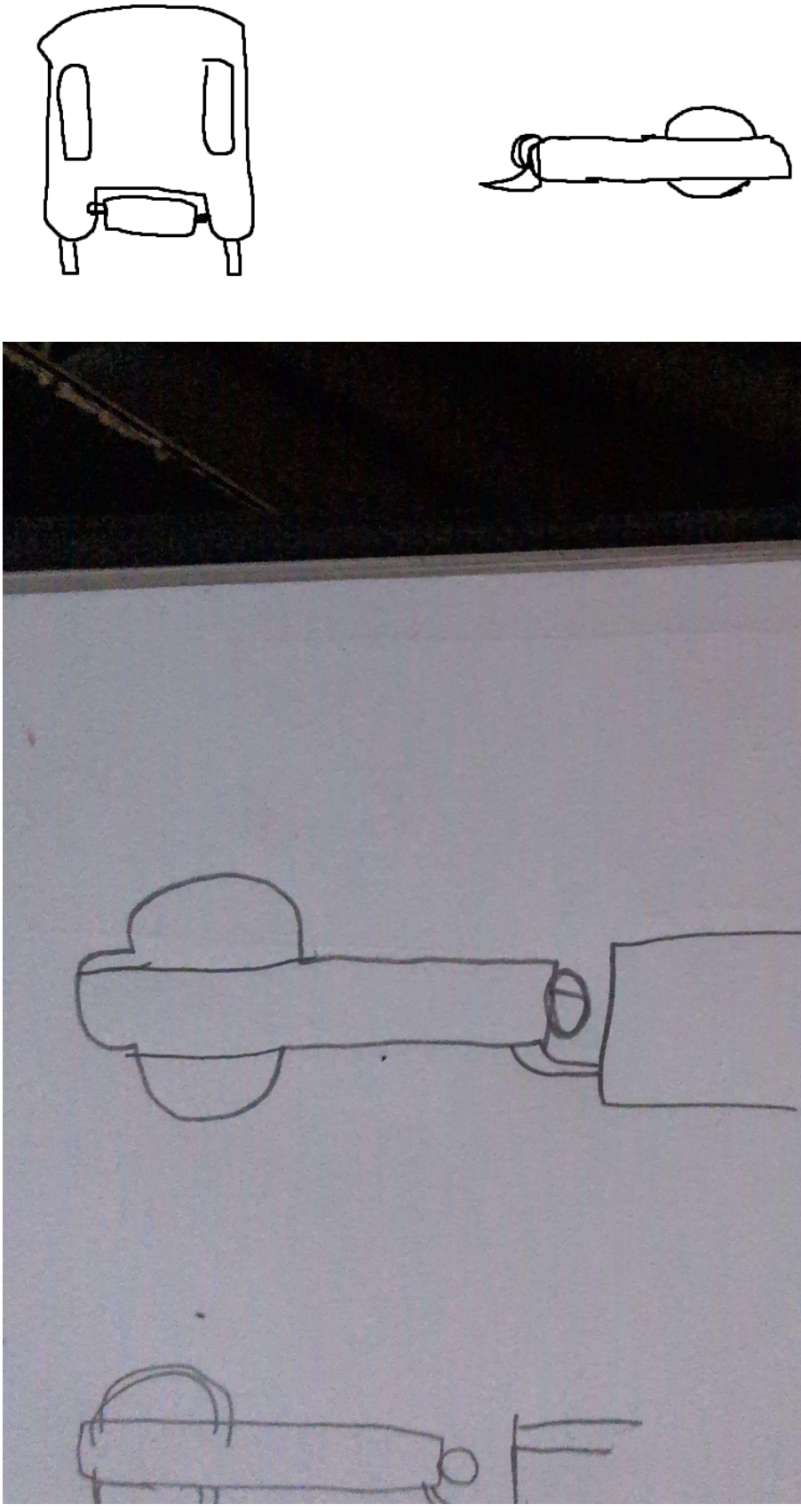

Figure 3. Second drafts.

The team had several main criteria. Firstly, the robot needed to have large wheels that protrude on both sides to facilitate the ability to drive upside down if needed. Secondly, the robot needed to fit a small spinning weapon with a live shaft on the front, which would be used to flip opposing robots. Thirdly, the robot had size requirements. The robot had to be under 30 cm, and weigh under 450 grams. The weapon had a limit, which mandated that the tip speed of the spinning blade would not exceed 100 m/s.

The first 3D design was under these constraints, but was too heavy to be considered, and the design had to be shrunken down.

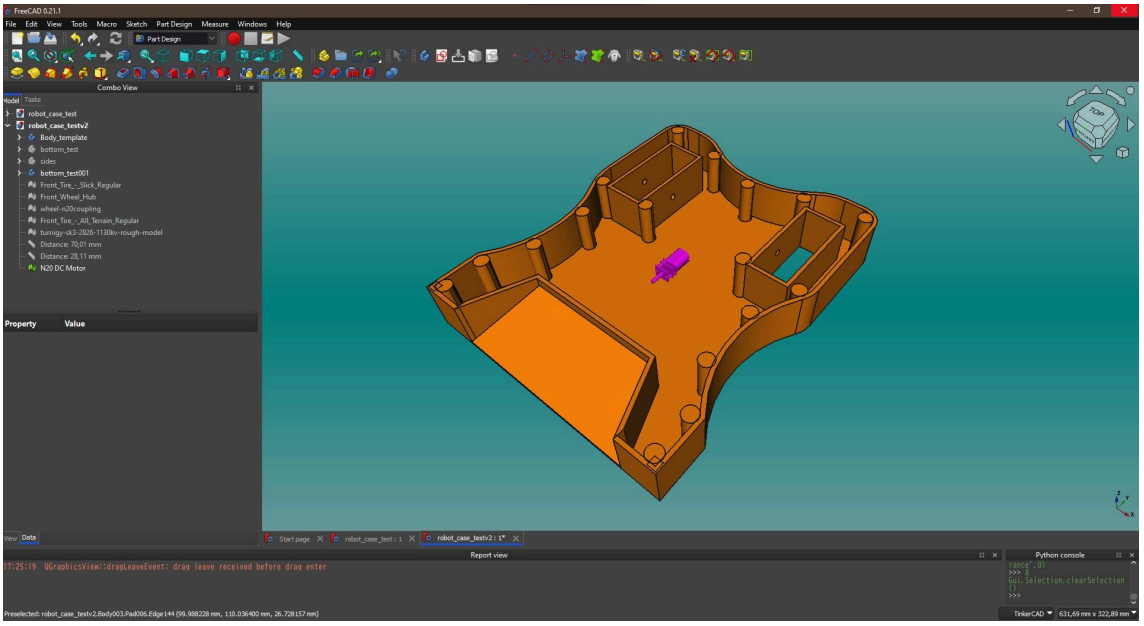

Figure 4. First case design, N20 drive motor for scale.

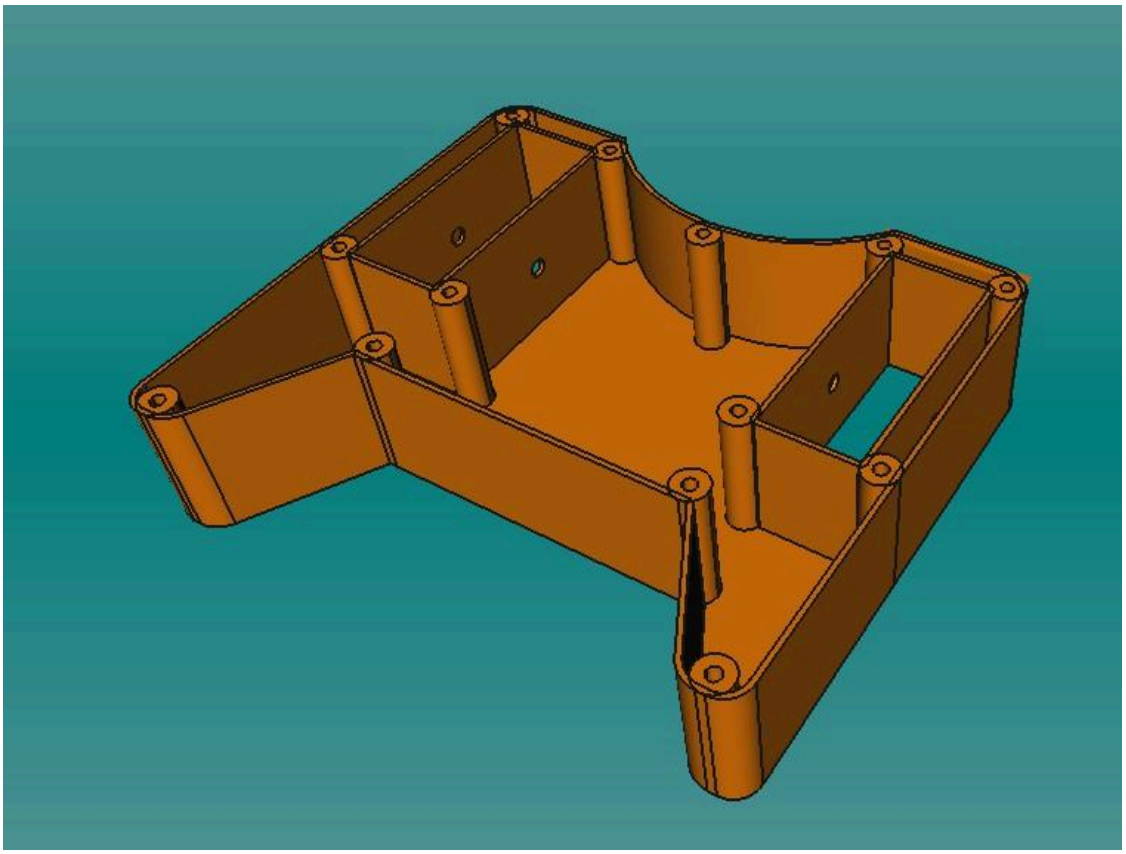

Figure 5. Second design, the robot is much smaller now.

During one of early SCRUM meetings, it was decided that the walls needed to be angled in a way that would prevent anything from grabbing onto them. The third design followed this guideline.

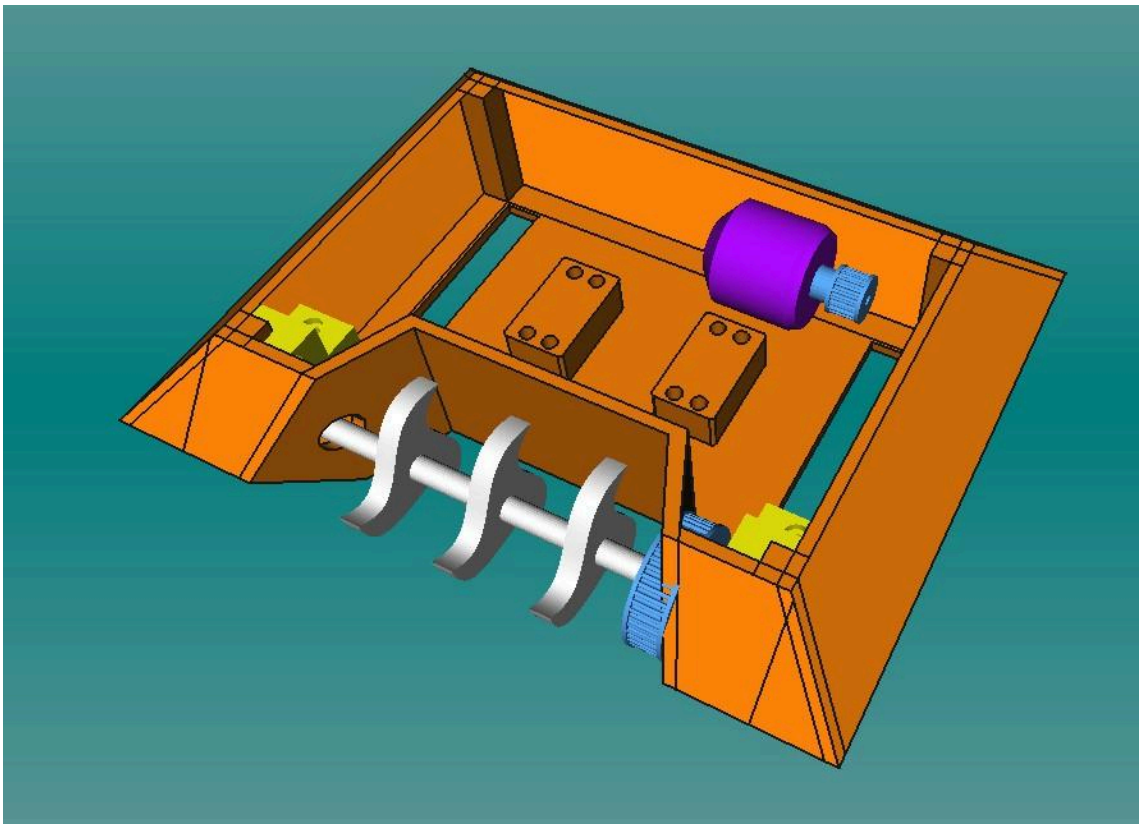

Figure 6. Third design, the walls are at an angle.

As can be noted from the picture, fitting all the parts to facilitate live shaft on the weapon and large wheels provided unique challenges, as the weapon motor was difficult to fit in a way that provided a clear path for a timing belt for appropriate length. The wheels were designed to be more thin to save weight and to make it easier to design around them. The first design was much larger than the final one.

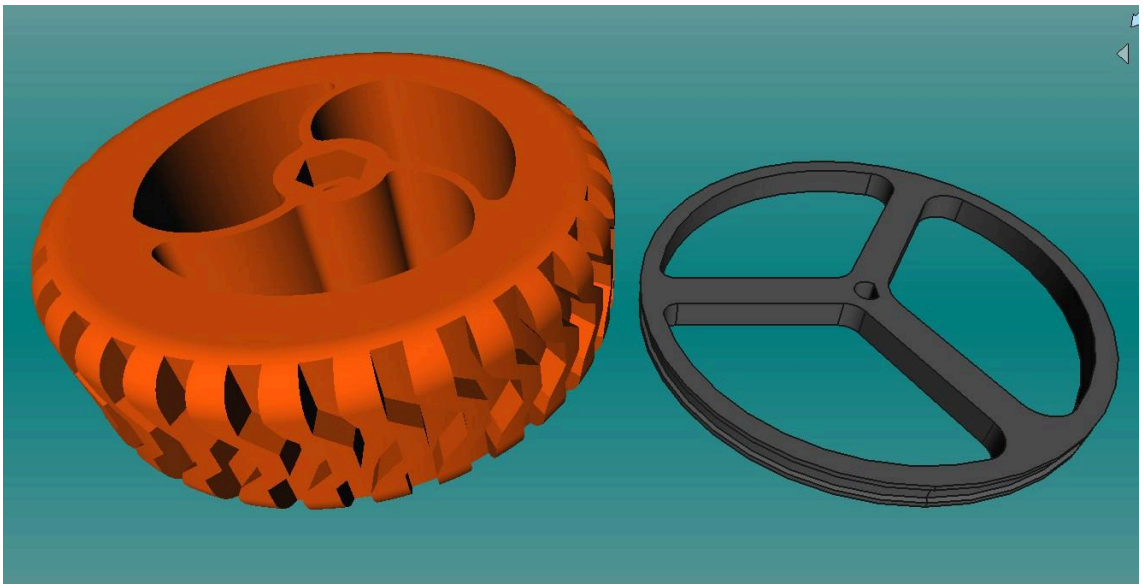

Figure 7. Wheel design process, from first to final design.

After researching timing belts and pulleys, a 200 mm GT2 timing belt with 2 mm pitch and 6 mm width was found in the shop, which satisfied the needs of our robot. After many steps and iterations, the final design was ready.

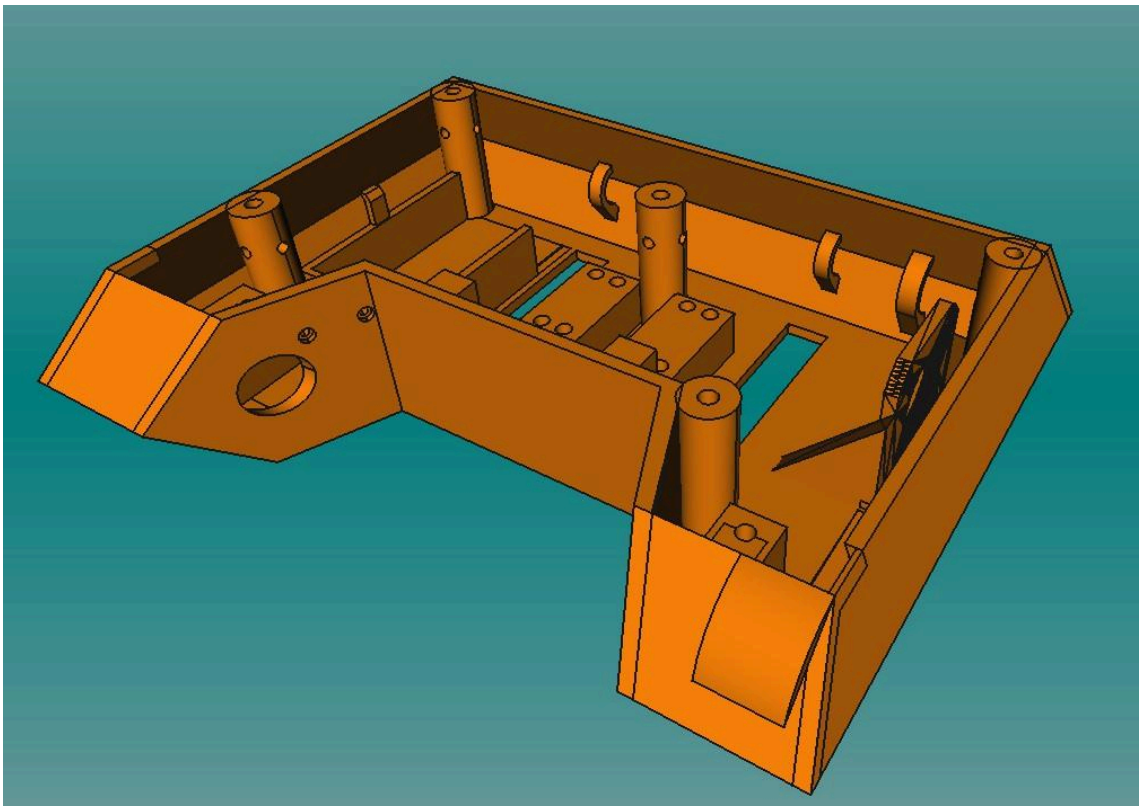

Figure 8. The final frame design.

2.2 Wheel Design

The wheels of the robot were 3D printed out of PLA-CF filament using different 3D printers at Robo Garage. For tires our team used basic rubber bands bought from a normal supermarket, as they satisfied the requirements, providing enough friction as well as being easily held in place with glue. The wheels themselves had a single isosceles trapezoid shaped groove 2 mm (inner width) to 2.5 mm (outer width) wide and 0.5 mm deep to fit the rubber bands, as shown in figure 9.

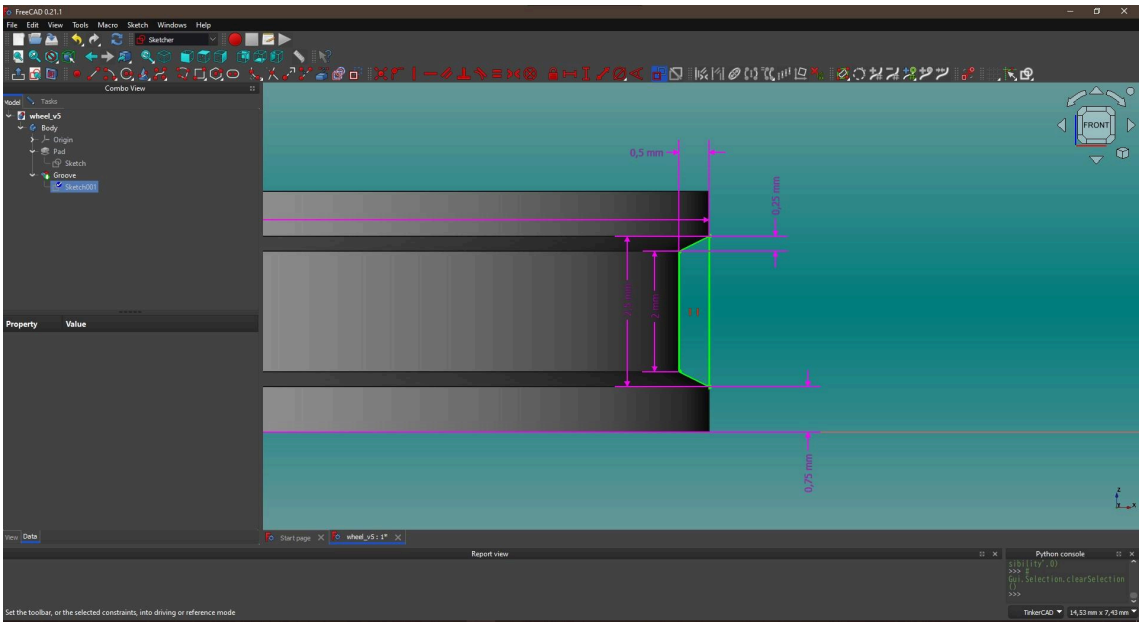

Figure 9. The wheel shown in FreeCAD with groove measurements.

The wheels were 4 mm thick and had a diameter of 60 mm.

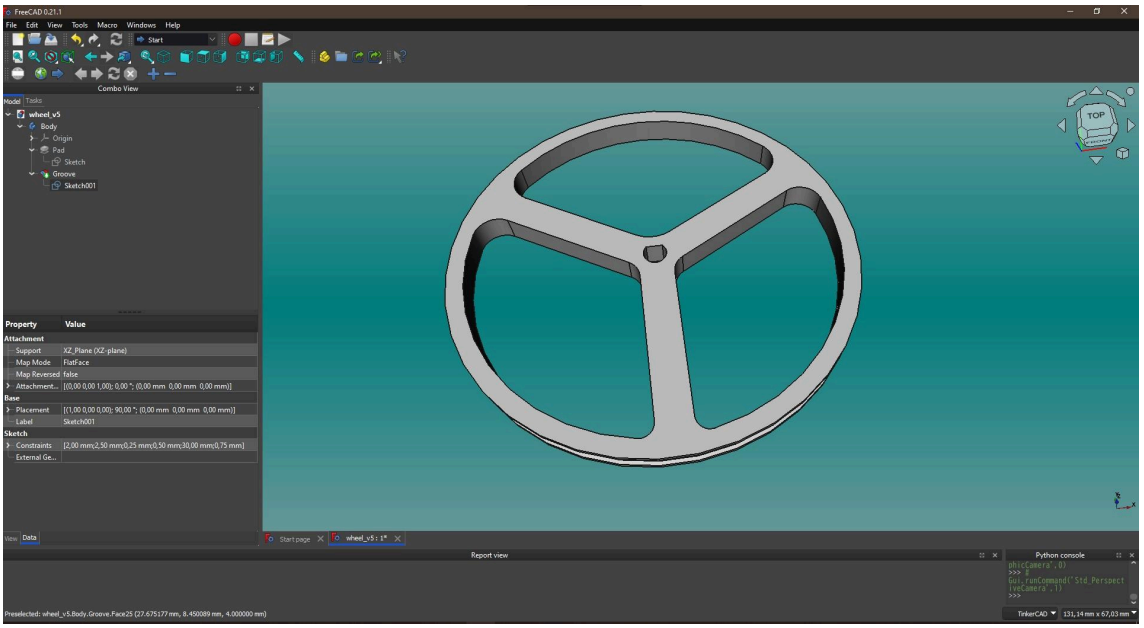

Figure 10. Wheel in FreeCAD perspective view.

Wheels made out of ABS were tested, but cracked too easily with even low force hits.

2.3 Outer frame

The frame of the robot was also manufactured with a 3D printer and printed out of TPU with shore hardness of 95A, which is an elastic plastic, providing increased defense from blunt hits as the bending provides shock absorption without cracking easily. The final version of the frame ended up being ≈ 137 mm (137,28 mm) long and ≈ 185 mm (184,56 mm) wide, both measured from the furthest point of the angled wall to the other.

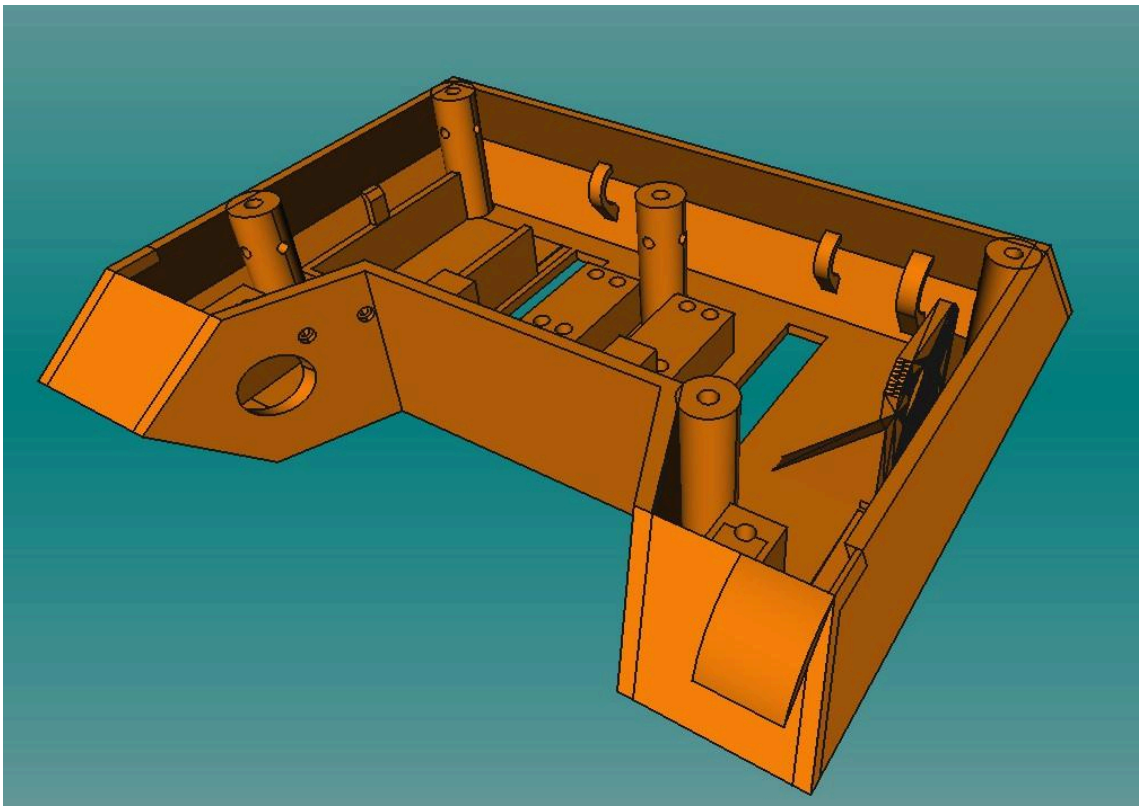

Figure 11. The outer frame of the robot.

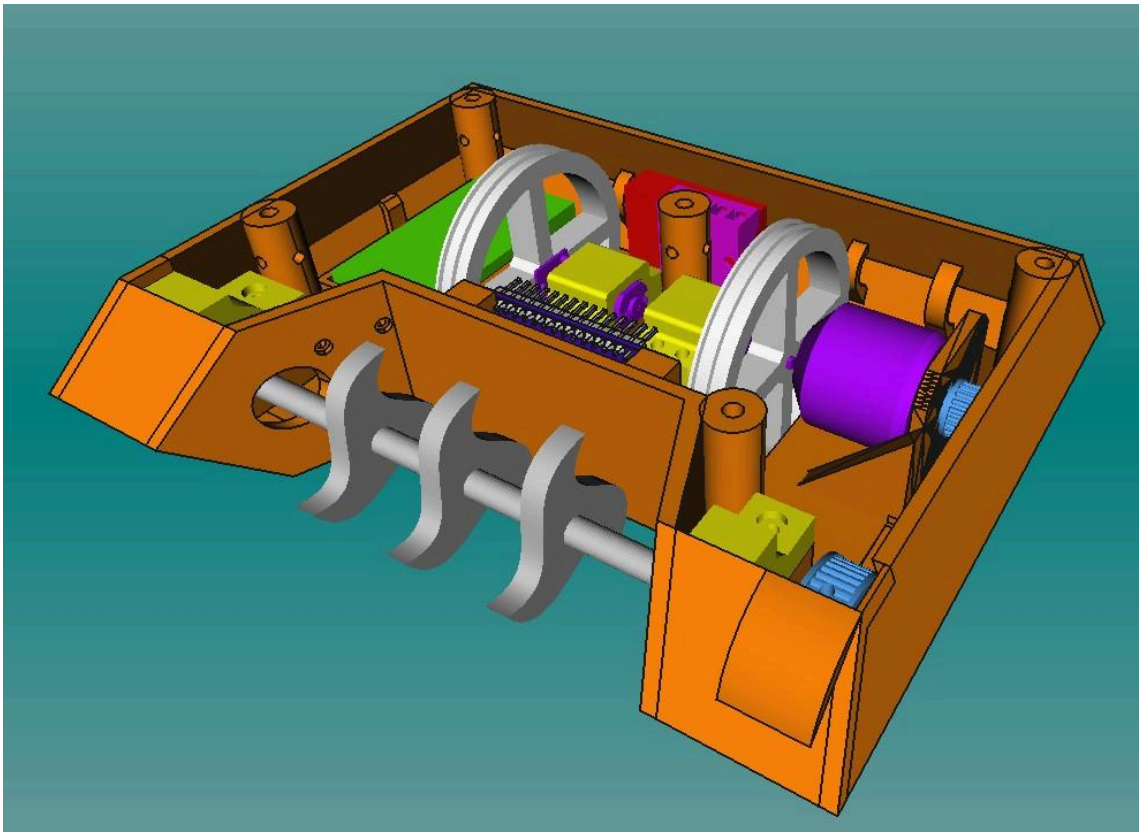

Figure 12. Assembling the robot in FreeCAD.

The walls were angled to prevent opposing robots from grabbing onto our robot, whilst also providing some protection from horizontal cutting attacks. The opposing blades would hopefully slide along the walls without cutting too deep, as our robot did not have metal plates for protection due to weight concerns. The wall angles were 110° degrees on both sides as shown in figure 13 and 135° degrees on the front and again 110° degrees on the back, as shown in figure 14.

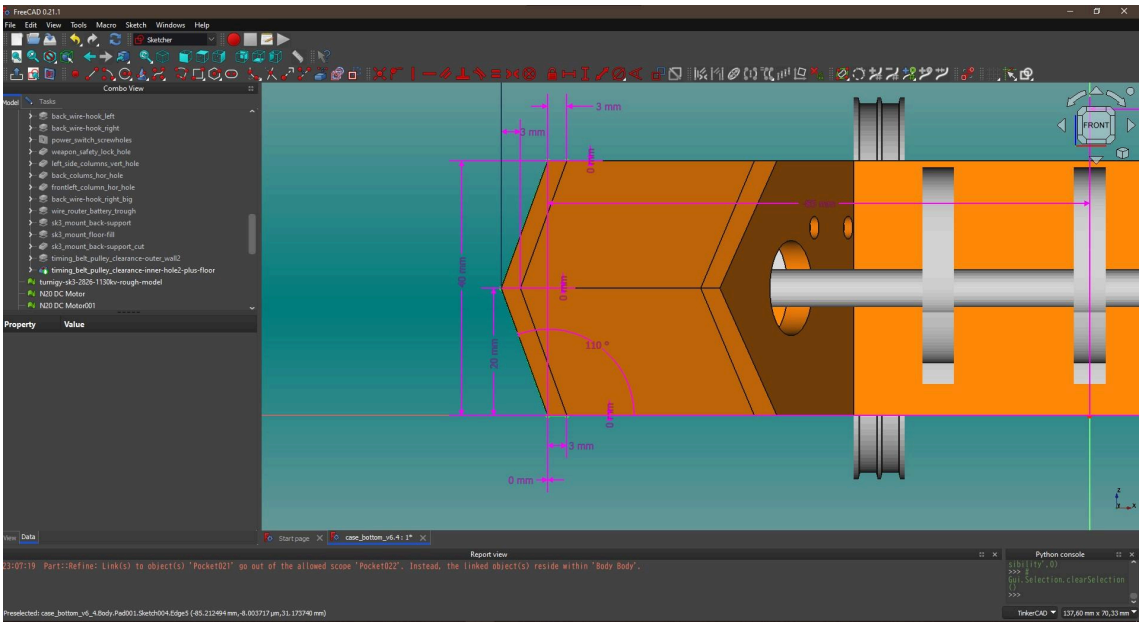

Figure 13. CAD front view showing angle of side wall.

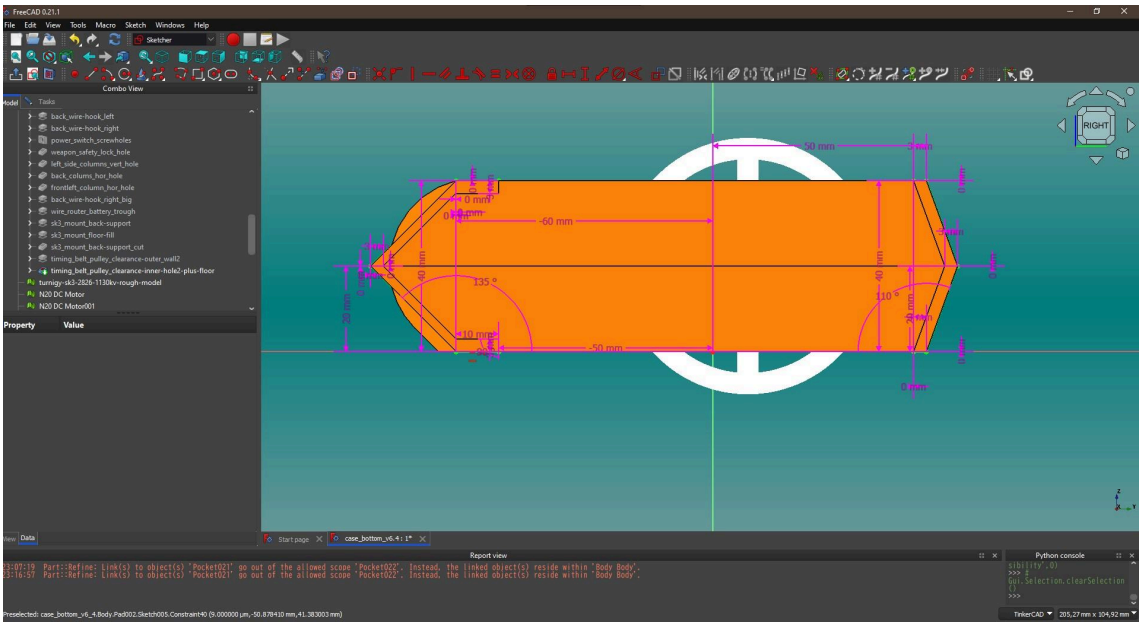

Figure 14. CAD side view showing angle of front and back walls.

The left side of the front wall has a circular protrusion to help fit the GT2 timing pulley inside, fitted on the weapon axle.

The frame has a slot in the front to fit the weapon and its blades, measuring at around 122 mm on the widest and 80 mm on the narrowest part. The depth of the pocket is 50 mm. The front wall is 3 mm thick and takes space in this slot, making the space for the weapon slightly smaller.

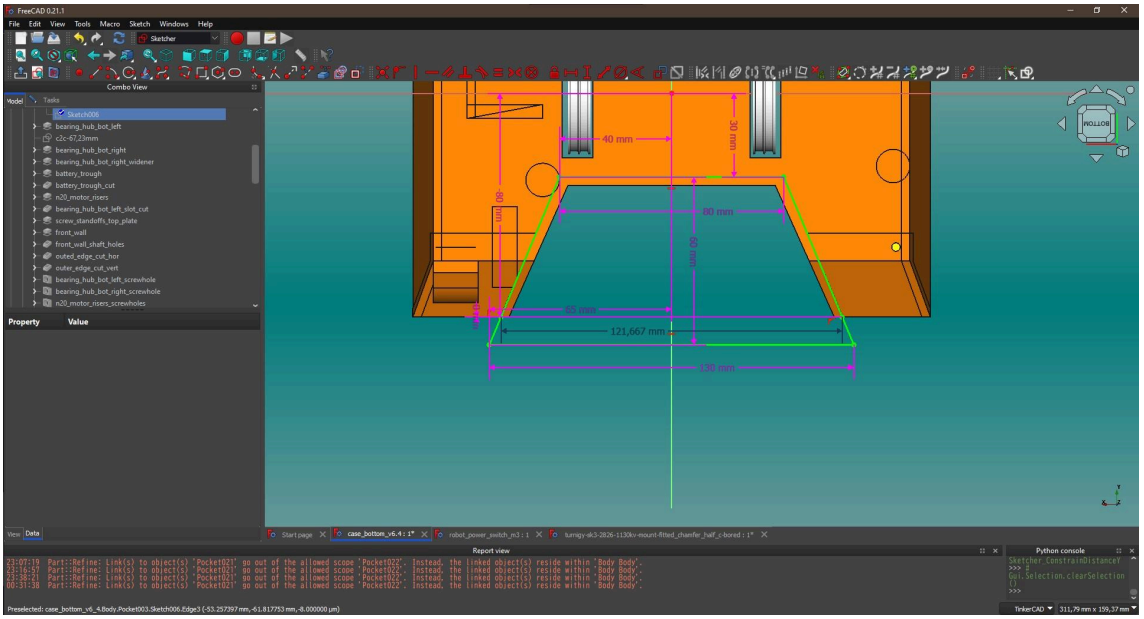

Figure 15. The slot for the weapon, seen from the bottom.

The frame is 40 mm tall not including the 1 mm thick top panel to allow the wheels to come out of both sides, providing ability to drive upside down if the robot is flipped. The top plate is attached to the bottom frame using M3 screws, which screw on heat-set inserts that are inserted in the standoffs. The top plate was made out of PLA-CF, but TPC with shore hardness of 45D was also tested, but was too flexible as 1 mm thick.

The power switch was a simple design, using two M3 nuts and a single M3 screw. The upper nut was de-threaded beforehand and as such, the electrical connection was only made when the base of the screw touched the upper nut.

The bottom nut had threads on it, holding the screw in place.

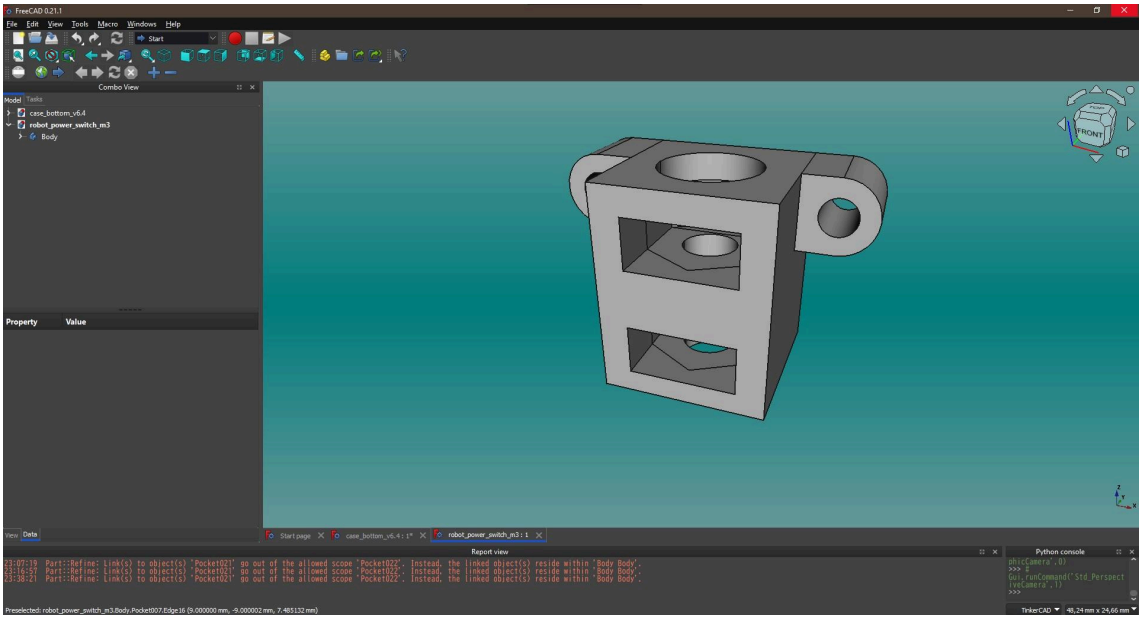

Figure 16. The power switch 3D model in FreeCAD.

The switch was attached to the frame with M2 screws. The reason for this design was that as per rules of the competition, the robot needed to be able to be switched on and off quickly, and our simplistic design allowed the aforementioned action with the top plate being closed, whilst also being a very lightweight solution. The power switch was printed out of PA in the beginning, but cracked easily between the layers when the bolt was fastened, and as such, was printed out of TPU in the end.

The battery had a trough designed to fit it in the case, which prevented it from moving and getting between wheels and outside of the robot. The standoff pillars had holes through them that were designed to attach wires if needed. There were also small hooks that were part of the main frame print, mainly behind the wheels but also a single one next to the battery. These were used to help route the wiring and also to hold electronics such as the ESC in place.

The weapon motor was held in place with a mount, also a part of the main 3D printed frame.

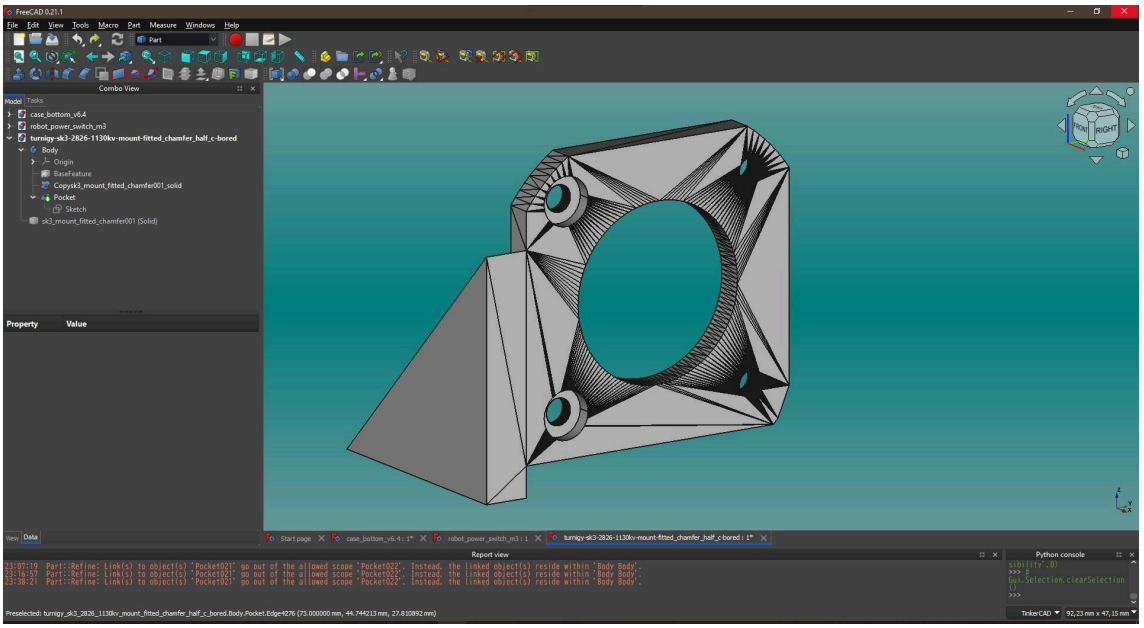

Figure 17. Turnigy SK3 2826 mount.

As shown in the previous figure, the motor and its metal cross-shaped attachment were attached to the mount with four M3 screws and nuts, one on each corner. The figure does not show the additional support that was added to the case design itself behind the mount.

The drive motors were held in place with specific N20 motor mounts. The 3D model of the mount was the only thing we did not design ourselves.

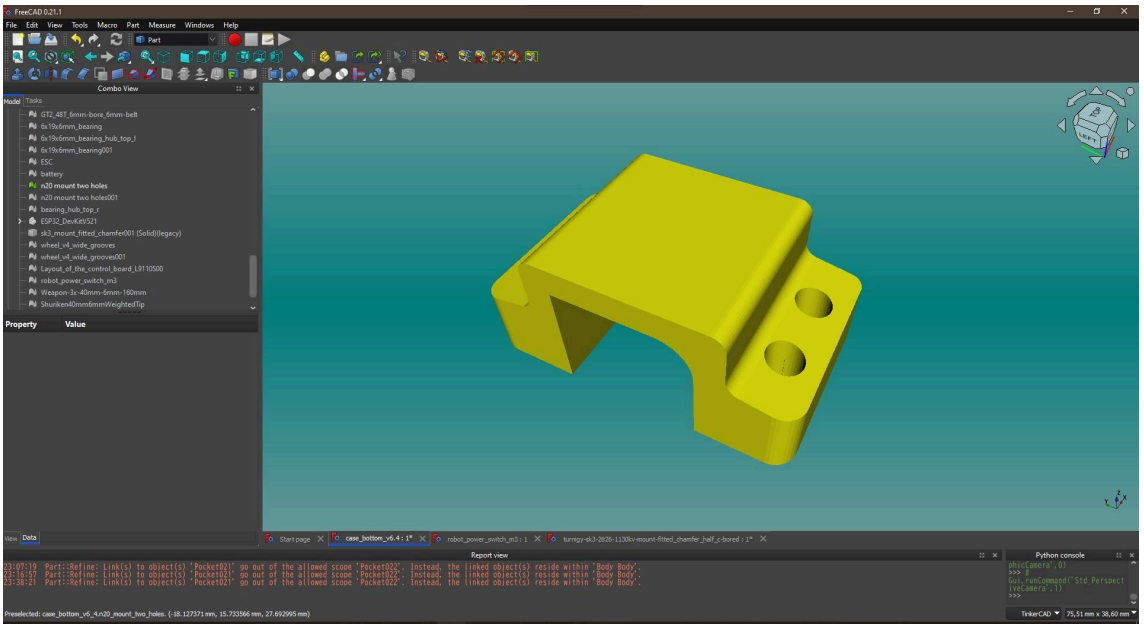

Figure 18. N20 drive motor mount.

The mounts had four M3 screw size holes to attach them to the frame, which had M3 heat-set inserts inserted into it. In the end we only ended up using two screws and inserts for each wheel to save weight, as the extra screws were not necessary.

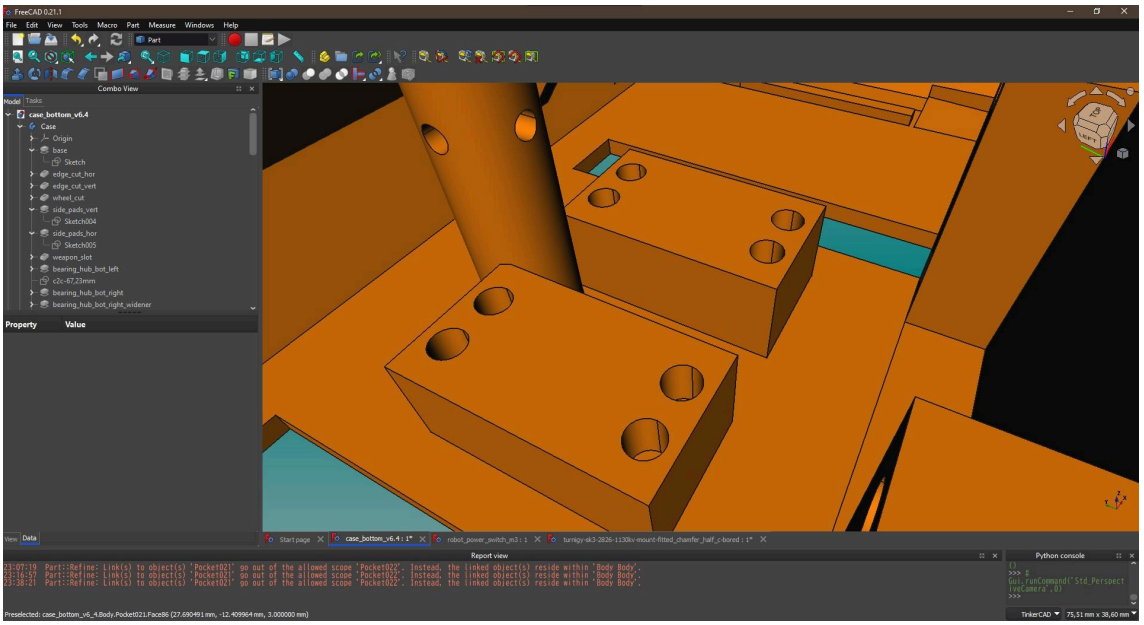

Figure 19. Motor mount points.

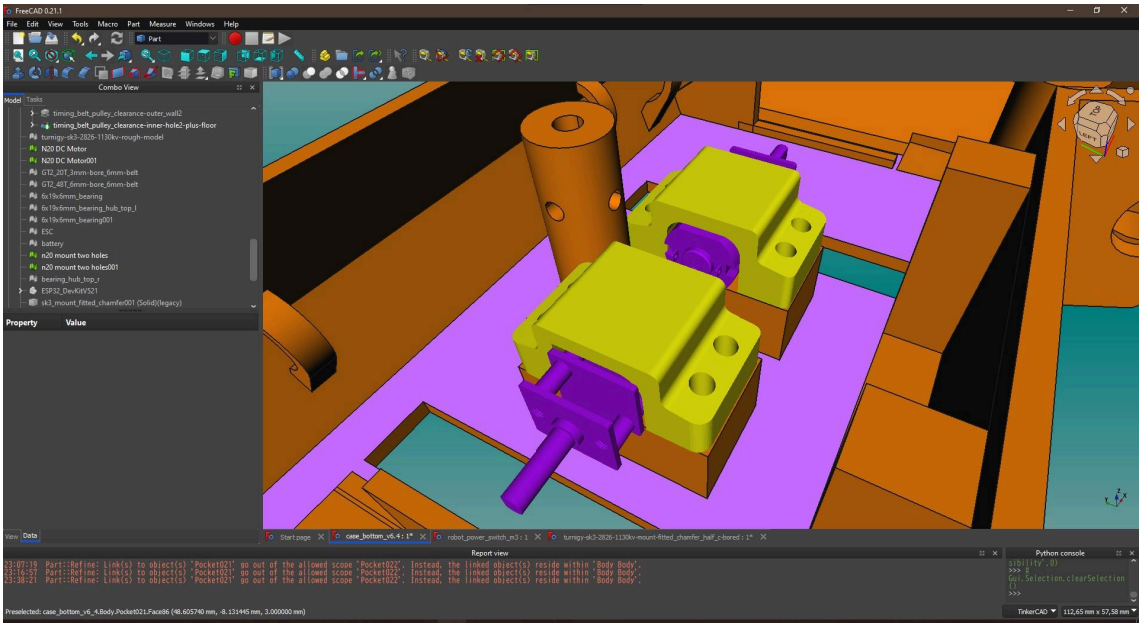

Figure 20. Motor mount points with motors and mounts in view.

The weapon needed to be held in place with ball bearings, so the case design was designed to fit 6x19x6 mm bearings on both sides (6 mm inside diameter/bore, 19 mm outer diameter and 6 mm width).

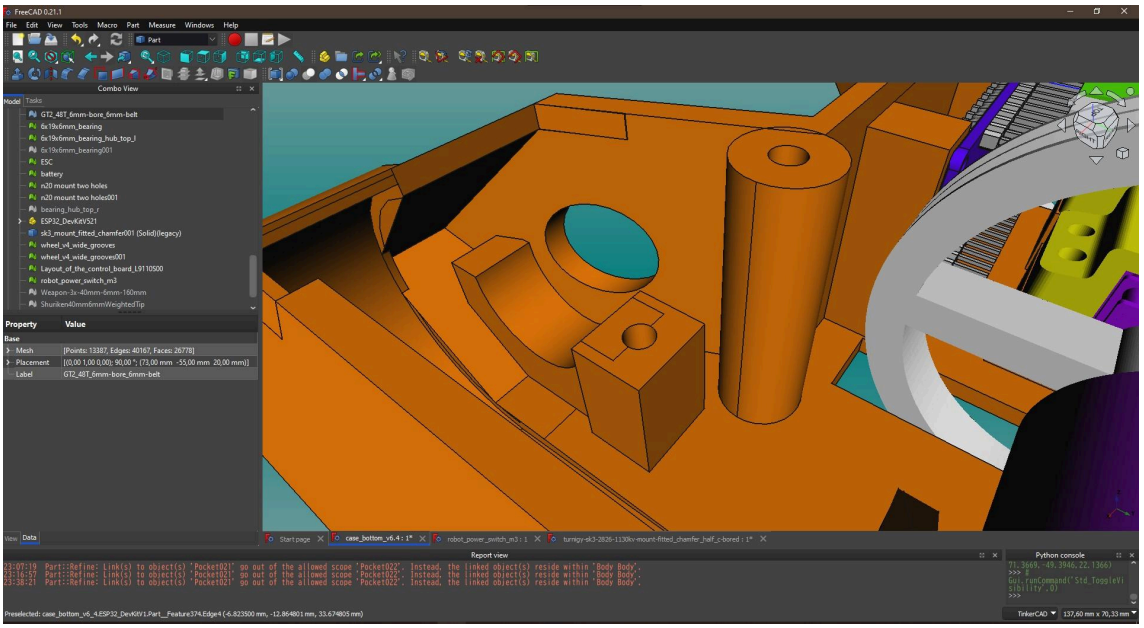

Figure 21. Left side bearing hub bottom.

The sides are different from each other, as shown in these figures.

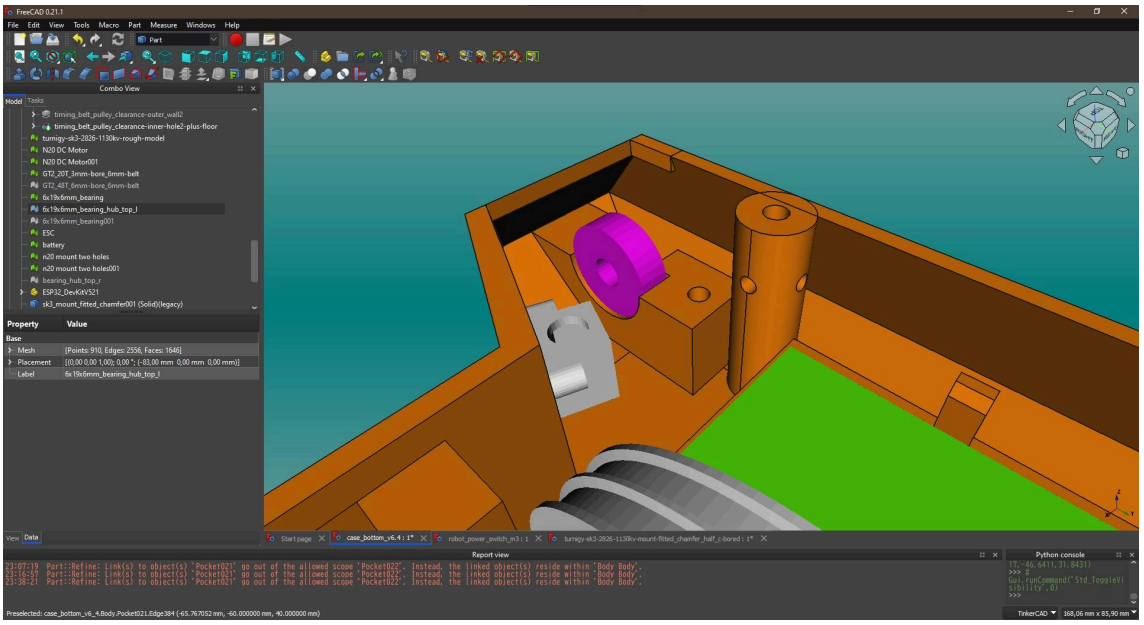

Figure 22. Right side bearing hub bottom with bearing shown.

The bearing hubs each have a top part that holds the bearings in place. They were attached to the frame with M3 screws screwed into heat-set inserts in the frame.

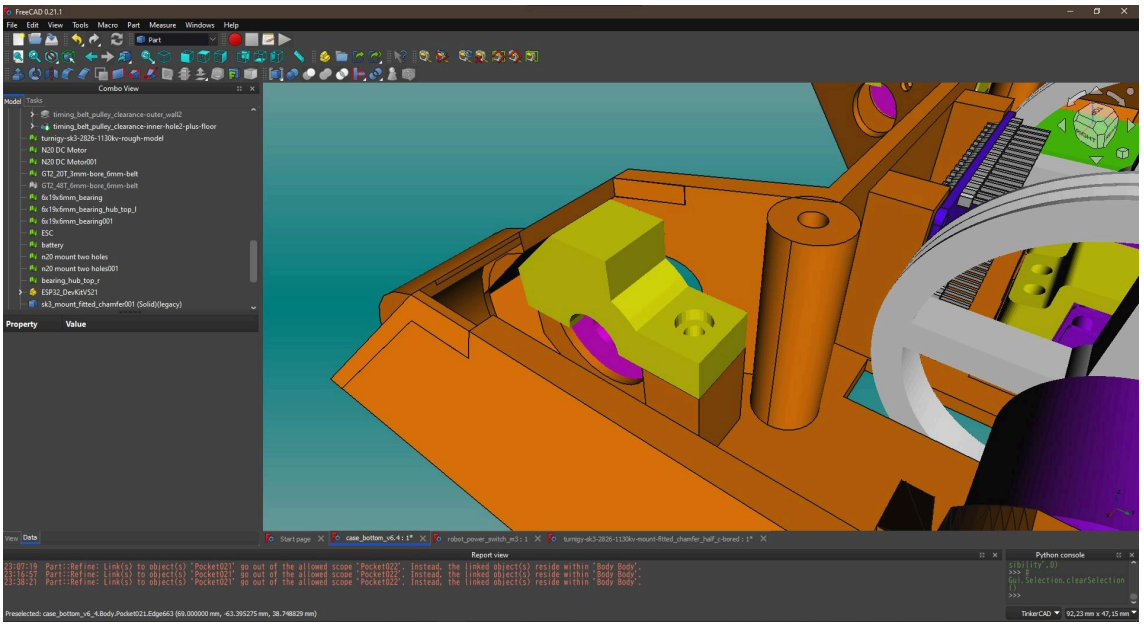

Figure 23. Left side bearing hub with top part shown.

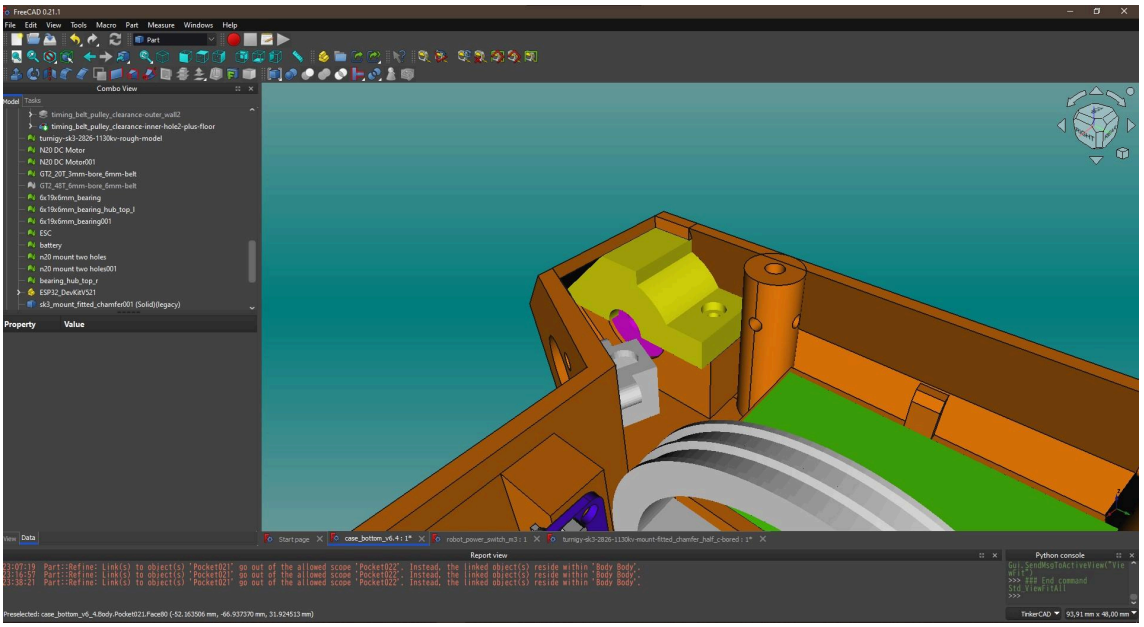

Figure 24. Right side bearing hub with top part shown.

To allow the weapon axle to fit between the bearings, the front panel of the robot had two circular holes with a diameter of 15 mm.

The robot had two wedges that were 3D printed out of PLA. The purpose of these wedges was to help the robot move without as much friction, as the robot leaned on these wedges, and to lift the opposing robot into the spinning weapon

blades, increasing bite and the effectiveness of the weapon. The final version had two of these wedges, one on each side of the blades.

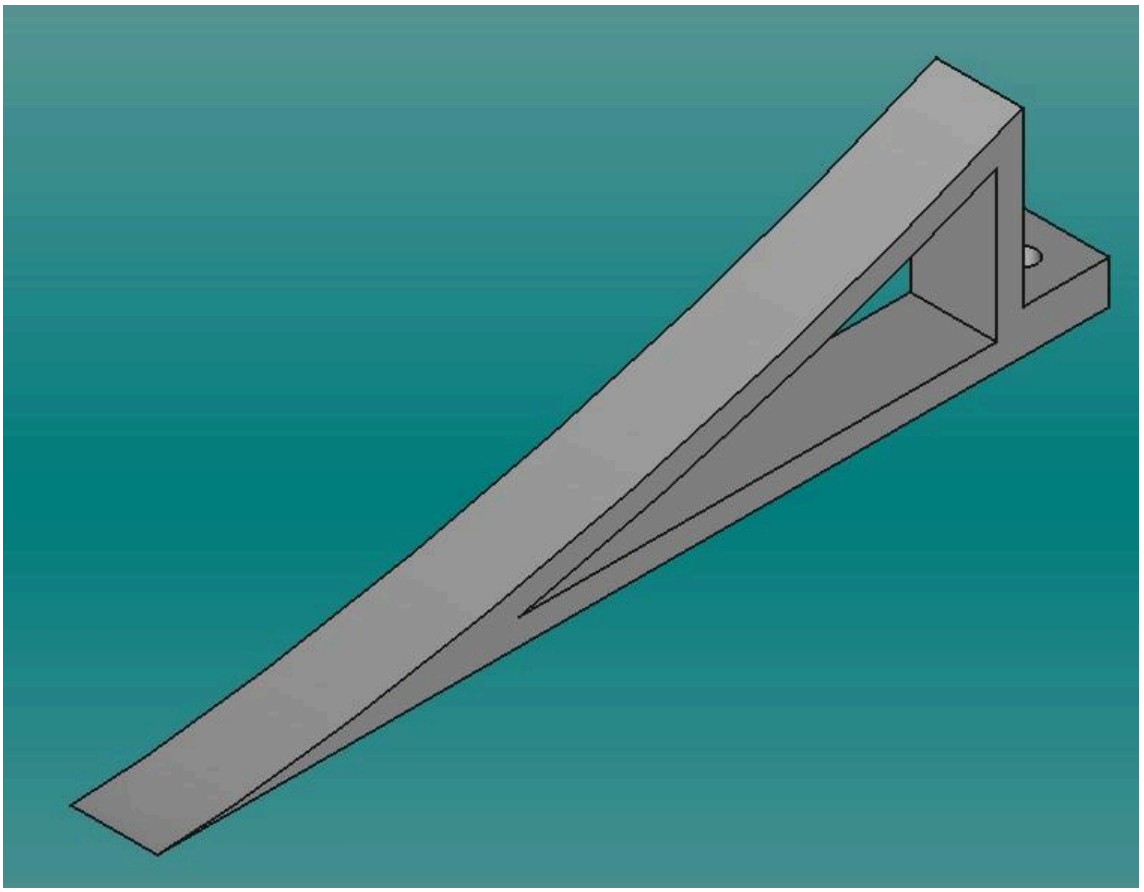

Figure 25. Wedge shown in isometric view.

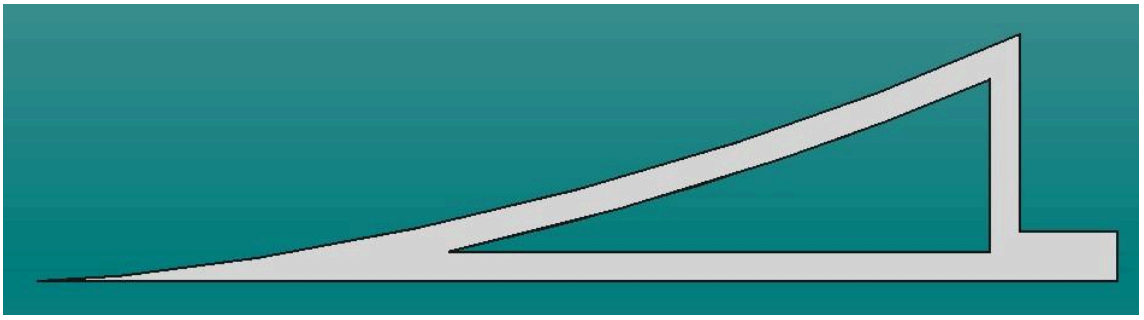

Figure 26. Wedge from the side.

The final version was made thinner using a slicer program. To attach the wedges, M2 screws and nuts were used.

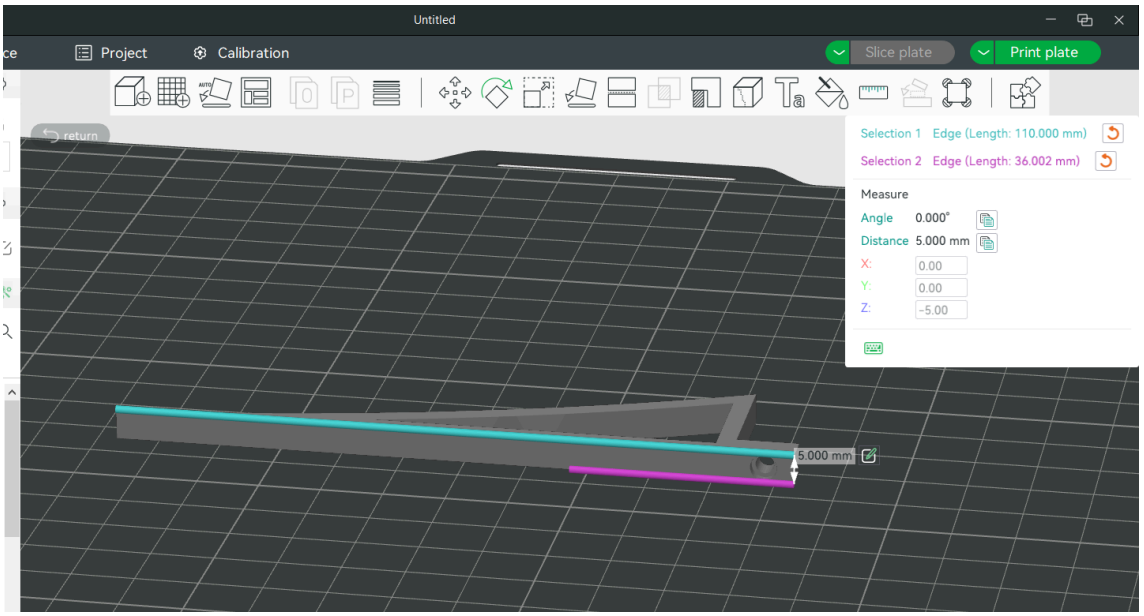

Figure 27. Thin wedge in Bambu Studio.

As shown in figure 27, the wedges were 5 mm thick.

2.4 Timing Belt and Pulleys

The timing belt chosen was a GT2 timing belt with a length of 200 mm, width of 6 mm and pitch of 2 mm. The belt satisfied the requirements for the robot and as it was easily available at Robo Garage, was the easiest choice.
Experiments with different gear ratios and teeth amounts were done, and in the end the weapon was using a 1:1 ratio with both timing pulleys having 33 teeth, as this was closest to the optimal distance for the belt, which was 67,23 cm with the first set of pulleys being 16 teeth and 48 teeth, meaning the ratio was 1:3.
For 33 teeth pulleys and 200 mm length GT2 belt with 2 mm pitch the optimal center to center distance was 67,00 mm.

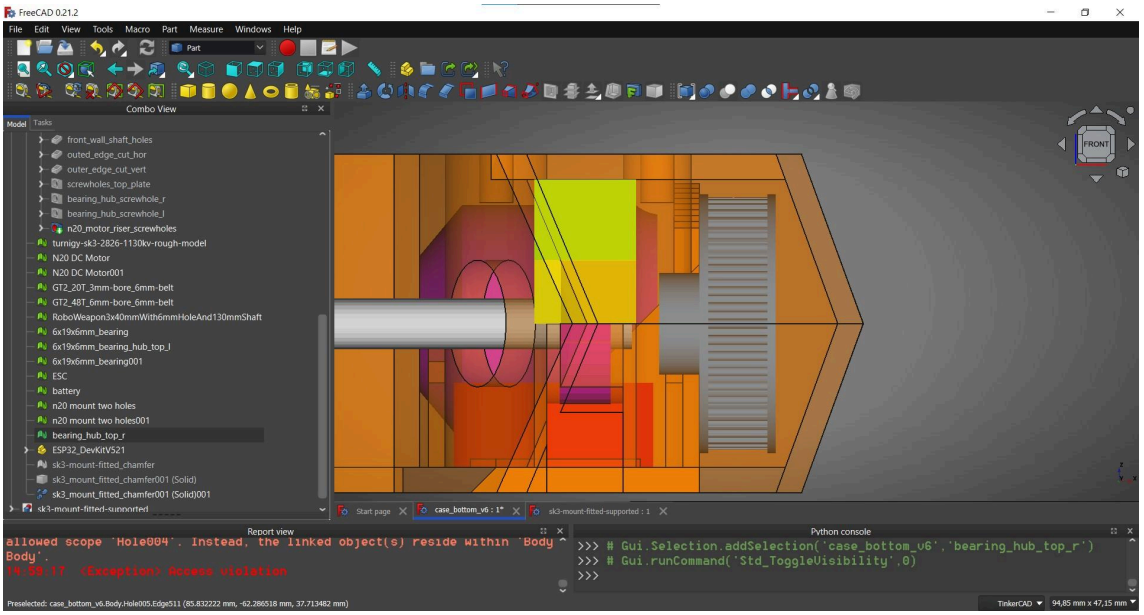

Figure 28. 48 teeth timing belt pulley shown in front, transparent case.

The pulleys were printed out of PLA-CF, and modifications were made to the model using FreeCAD's spreadsheet functionality that allows the changing of various parameters of the model with ease.

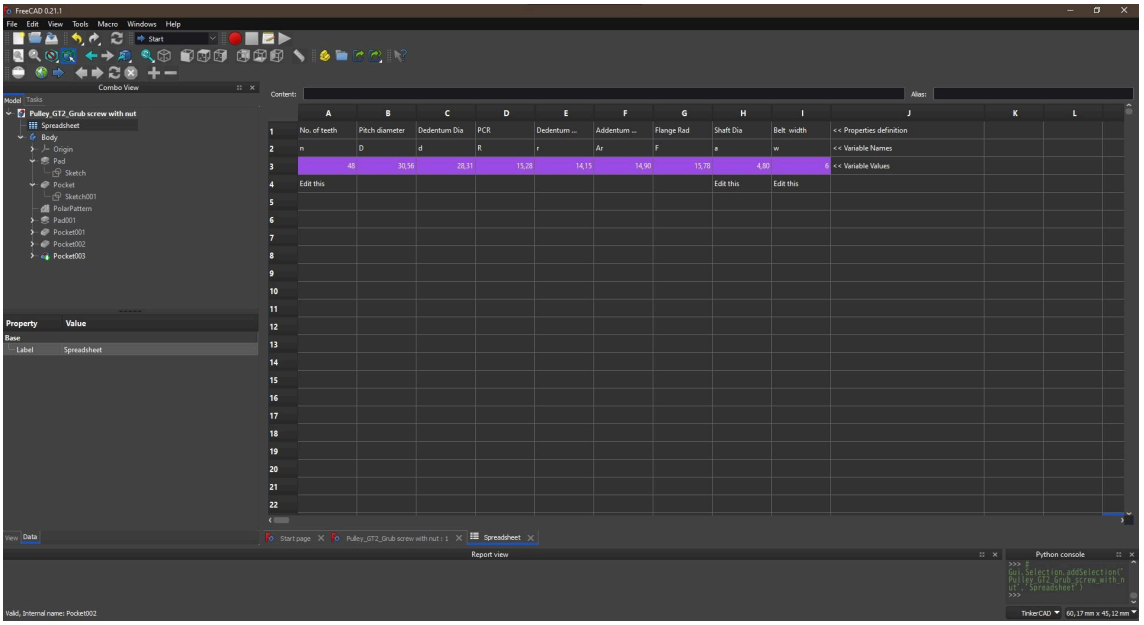

Figure 29. Timing belt pulley parameters in FreeCAD spreadsheet.

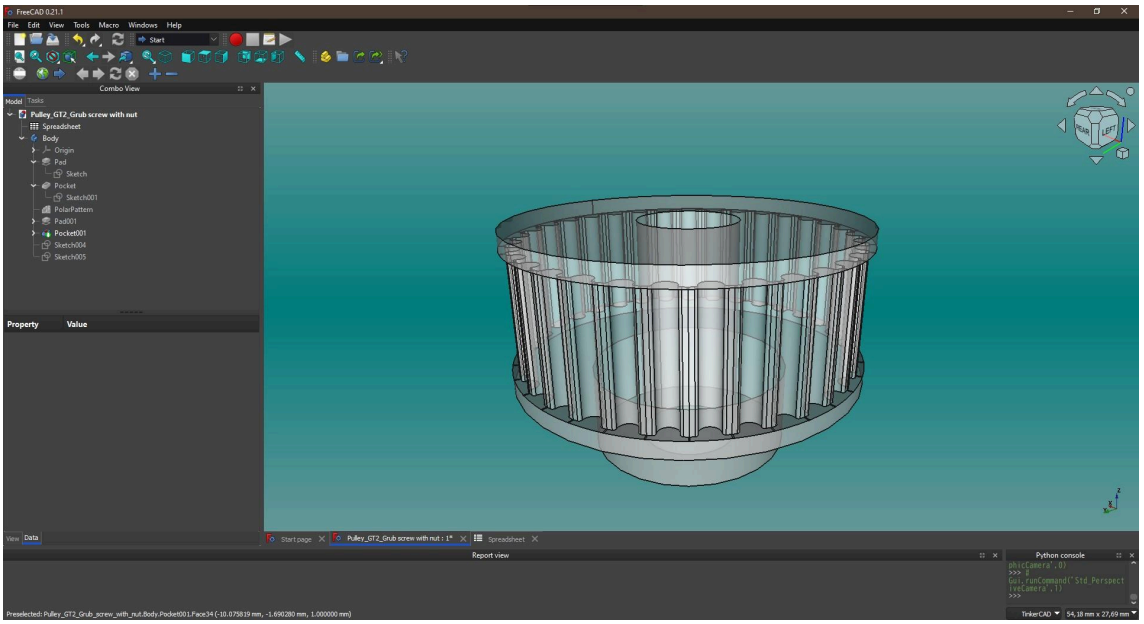

Figure 30. Timing belt pulley 3D model, 33 teeth.

3 Code

The code for the robot was written in the C++ programming language. It used the Arduino framework and the aed3 PS4-ESP32 library from github. The code handles movement of the robot and weapon controlling. Movement happens with tank controls meaning you can either turn or go straight. The weapon is toggled meaning you don't need to keep the button pushed to have the weapon on. The code also has two failsafes integrated.

3.1 Movement

Controlling the robot was done using tank controlling. Meaning that each wheel is controlled separately. It could be done by using joysticks or a D-pad. In the code controlling wheels was done by the following code:

```
void set_motor_a(int stateIA, int stateIB){
 digitalWrite(MOTOR_A_IA, stateIA);
 digitalWrite(MOTOR_A_IB, stateIB);
}
```

```
void set_motor_b(int stateIA, int stateIB){
 digitalWrite(MOTOR_B_IA, stateIA);
 digitalWrite(MOTOR_B_IB, stateIB);
}
```

Figure 31. The integer parameters are read from the controller and then the value is written to the pins that connect to the driver.

3.2 Weapon

The weapon used a drone motor meaning it would commonly be used with radio signal controllers. Since we used a bluetooth device to control our robot we needed to “simulate” that radio signal to control the weapon.

```
void set_throttle(int pulseWidth){
 digitalWrite(WEAPONMOTOR, HIGH);
 delayMicroseconds(pulseWidth);
 digitalWrite(WEAPONMOTOR, LOW);
 delayMicroseconds(REFRESH_INTERVAL - pulseWidth);
}
```

Figure 32. In the code we pass the “pulseWidth” parameter for the function. We had two values for it, either minimum or maximum. Meaning our weapon was either off or at full throttle. The weapon works in 20kHz and throttle goes from 1000 to 2000. So when we want to throttle the weapon to full speed, we pass pulseWidth as 2000, then we write the weapon pin high and wait for 2000 microseconds, after that we write the pin low and wait 20 000 - 2000 microseconds. Because the weapon works in 20kHz we need that 20 000 - 2000 calculation so everything happens in one 20kHz cycle.

3.3 Failsafe

The robowars competition required every robot to have a failsafe in the code where if the controller gets disconnected from the robot, the robot will shut down its movement and weapon. In our code we had two failsafes, one for controller disconnection and another for manual activation.

```
void failsafe(int level){
   set_motor_a(LOW, LOW);
   set_motor_b(LOW, LOW);
   weaponState = false;
   control_weapon();
   if (level == 1) {
       while(1){
           digitalWrite(LED_RED, LOW);
           delay(500);
           digitalWrite(LED_RED, HIGH);
           delay(500);
       }
   }
}
```

Figure 33. If the failsafe activates it will write every movement motor pin and weapon motor pin to low. If the failsafe was activated by detecting controller disconnection then it would go back to the main loop and wait for the controller to reconnect. If however the failsafe was activated by the user manually, it will have level 1 meaning it goes to infinite loop and starts to flash LED on and off. The infinite loop was used so that if the user activates the failsafe then even when the controller is still connected to the robot, nothing can turn the weapon or movement back on even if someone tried to fiddle with the controller.

3.4 Bluetooth

As mentioned before, we used the aed3 PS4-ESP32 library from github. This library gave a nice api to work with a PS4 controller that was super easy and reliable to use. To connect the PS4 controller you had to first check your ESP32 MAC address. After that you wrote the MAC address inside the PS4 controller. To write a new MAC address into the PS4 controller we used SixaxisPairTool. After that it will look for that ESP32 when turned on thinking it's a “PS4”. It then connects and we can read values from the controller.

Here is a code snippet for waiting for connection to the controller.

```
void wait_for_connection(){
 while (!PS4.isConnected()) {
   PS4.begin(MAC_ADDRESS);
   Serial.println("Trying to connect");
}
```

Figure 34. Reading the values from the controller was also really simple. Here is a code snippet showcasing reading movement controls and also if the weapon was activated or if the failsafe was activated:

```
void handle_controls(){
 if (PS4.Up()) {
   move_forward();
 } else if (PS4.Down()) {
   move_backward();
 } else if (PS4.Left()) {
   turn_left();
 } else if (PS4.Right()) {
   turn_right();
 }
 if(PS4.R1() && PS4.L1()){
   Serial.println("FAILSAFE ACTIVATED!");
   failsafe(1);
```

```
 } else if(PS4.R1()){
   weaponState = !weaponState;
   Serial.println(weaponState ? "ON" : "OFF");
   delay(500);
}
```

Figure 35.

4 Schematics

The schematics for the robot were done using KiCad, an open-source electronic design automation (EDA) software. In the schematic the ESP32 can be seen as the brains, there's a voltage regulator that will drop 12V to 5V for the ESP32 and the L9110S motor driver. To the ESP32 there are also two LEDs connected to give a small user interface. The red LED was used to indicate power on and the blue LED was used to indicate bluetooth connection status.

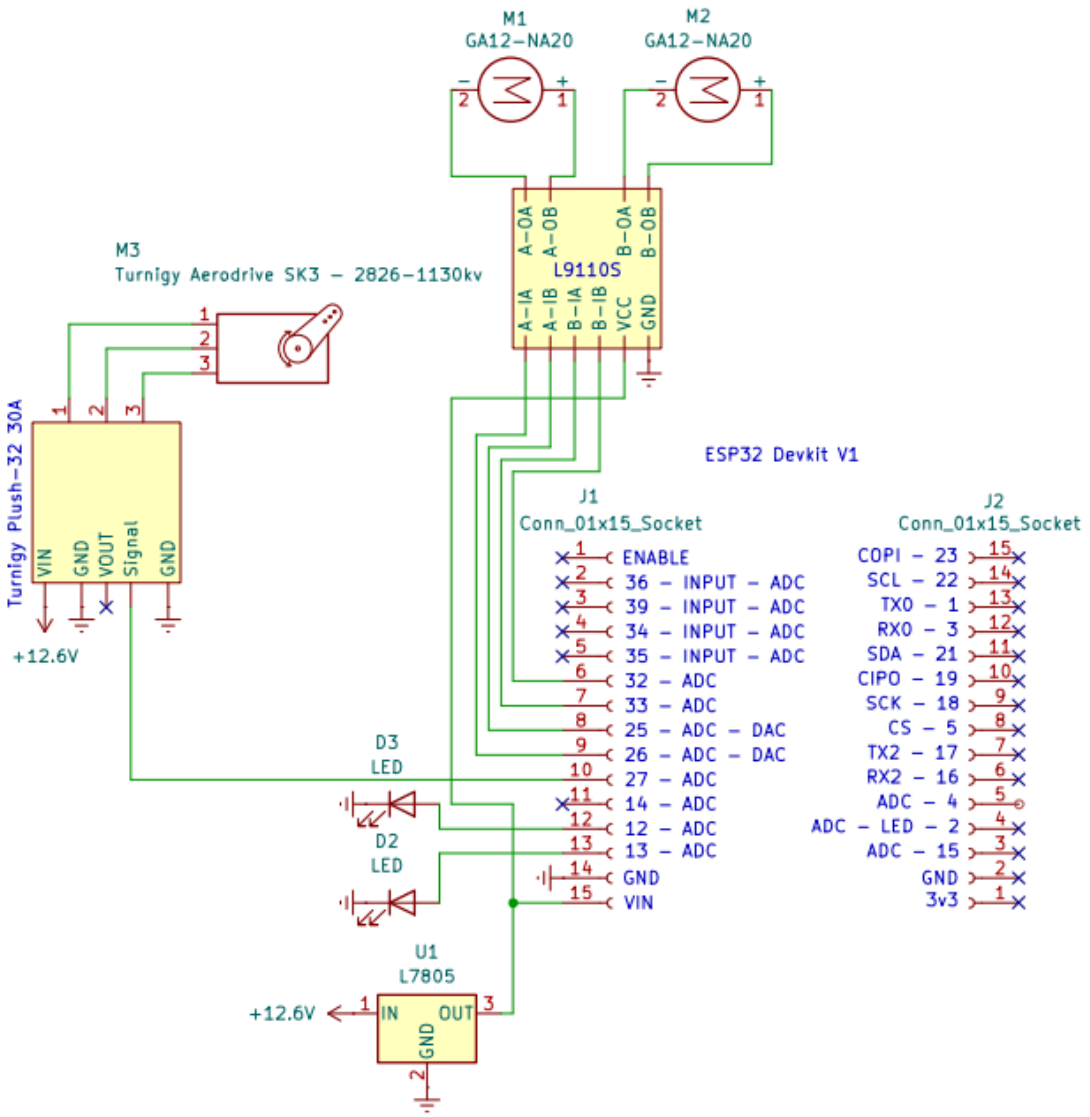

Figure 36. Picture showing schematics for the robot. It includes a motor driver for the weapon, motor driver for the wheels, two LEDs and one regulator.

4.1 Weapon ESC

Turnigy Plush-32 was the electronic speed controller (ESC) for our weapon and it was connected by one pin only to the ESP32 (Signal pin). It drew power straight from the battery because it could handle the 12V that the battery could output. From the weapon ESC there went three wires to the weapon motor. The

connection order of these didn’t matter much because the weapon is a brushless DC motor, meaning it works using electromagnets and so, the order of connection only affects which direction the weapon spins. This means that if the weapon spun in the wrong direction, we’d simply need to swap any two wires.

4.2 L9110S

To control our movement motors we used an L9110S driver that can drive two motors at the same time. As seen in the picture it is connected to the ESP32 with 4 pins, 2 for each motor and then 5V power and GND. It needed two pins for each motor because that way we can control what direction the motor is turning. The truth table in figure 37 showcases that:

Motor A Pin 1	Motor A Pin 2	Motor A State	Motor B Pin 1	Motor B Pin 2	Motor B State
0	0	Off	0	0	Off
0	1	Rotate CCW (Reverse)	0	1	Rotate CCW (Reverse)
1	0	Rotate CW (Forward)	1	0	Rotate CW (Forward)
1	1	Brake (Stopped)	1	1	Brake (Stopped)

Figure 37.

Explanation:

- **Off:** Both pins are LOW (0), and the motor does not receive power.
- **Rotate CW (Forward):** Pin 1 is HIGH (1) and Pin 2 is LOW (0), causing the motor to turn in one direction.
- **Rotate CCW (Reverse):** Pin 1 is LOW (0) and Pin 2 is HIGH (1), reversing the motor's direction.
- **Brake (Stopped):** Both pins are HIGH (1), shorting the motor terminals and creating an electrical brake.

This pattern applies to both Motor A and Motor B independently, using their respective pins.

5 Testing

The robot was tested in different stages of development. In this section we talk more about different tests used, what problems arose and how we fixed them.

5.1 Bluetooth connection

After we got our PS4 controller from the store, we started to test connecting it to the ESP32. As mentioned in the code section of this report, we used an external library for connecting the controller with the ESP32. Mostly it went smoothly but some problems arose because documentation didn’t mention everything and because of that it took some testing to get the controller working. One thing that wasn't mentioned in the documentation was what MAC address was used in the function connecting the controller. In the end it was found out that it was the ESP32's own MAC address and not the controller’s.

When we got the controller to connect to the ESP32 after that we started to test movement. We attached ESP32 to the breadboard and added LEDs to resemble wheel motors. There came the second problem. The LEDs were not lighting up when trying to use the controller. We printed the values to the serial console where it was seeing that the controller did work properly. After wondering and some tinkering we noticed that the breadboard seemed to have some loose connections. After we switched our breadboard to a new one everything worked fine so the problem was solved. After these everything worked correctly and we could proceed to test movement.

5.2 Initial movement testing

Initial movement testing was done as soon as we got our wheel design done and bluetooth connection to work. When running the test we had the first

problem straight when we tried to stress test wheels. We manually stopped them from rotating and that caused the wheel controller to blow smoke out. After that we went and read more documentation about wheel motors and then about the driver and we figured out that when the wheels are stopped from rotating it draws more current than the controller can handle. After careful consideration we came to the conclusion that it won't be a problem needing to be fixed because we shouldn't encounter something jamming our wheels in the competition. But during testing and additional assembling, we managed to blow up 7 drivers.

The second problem when running movement tests was that the robot would shut down after 10 seconds. It took some thinking to figure out that the problem was our voltage regulator. When the regulator dropped 12V to 5V constantly, it would heat up so much that it would shut down our robot after only 10 seconds. To fix this problem we found a heatsink that was attached to the regulator. After that we ran more tests and could drive the motor for up to 6 minutes, which was more than enough because one round had a max length of 3 minutes in the competition.

Figure 38. Heatsink was attached to the voltage regulator to maintain its temperature when running.

5.3 Weapon testing

When we got the weapon assembled, we started to test it. The weapon was first tested running with no resistance. After seeing that it works, we started to test it against our old case. To that we noticed that the weapon would only scratch the sides and not really do much damage or flip the case. We tried to add weight to the demo case but got the same results. Our weapon was running with a 3:1 gear ratio. We then changed it to run with a 1:1 ratio and tested again. We noticed improvements as it would now have more power to flip enemies. In the end that was the ratio we decided to stick with.

6 Manufacturing

The project needed most parts to be manufactured using 3D printers after being designed, this included a lot of testing how they work as physical parts instead of just 3D files in CAD, parts were modified after testing if need be or modified in some ways after noticing issues with the physical parts interacting with each other, we had two 3D printers available for us to use and a 3rd new one arrived during the last week of the project.

In addition to 3D printing we had the weapon parts laser cut from steel and then welded onto an axle, in the next section we will go over the machines we used.

6.1 3D printers

The 3D printers available for us were Ultimaker S5, Ultimaker 2+ Connect and the latest one which arrived during the last week of the project was a Bambulabs X1 Carbon, in this section we will go over the printers, issues we had and the splicers needed to use them

Figure 39. All the 3D printers at Robo Garage from left to right, Bambu Lab X1 carbon, UltiMaker 2+ Connect and UltiMaker S5

6.1.1 Ultimaker S5 and 2+ Connect

The Ultimaker printers we used the most because they were available from the start of the project at robo garage and had a wide variety of materials to use, provided that they were not in use by other groups, printing times were quite long for some larger parts and was heavily affected by the type of material being printed.

Both of the machines used Ultimaker Cura as the splicer, which you need to use to turn a 3D file into a printable object, telling the printer where to move the extruder to print the object, in Cura you could set where on the print bed the print would take place, orientation and other print settings like wall thickness, infill pattern, amount of infill, supports, what material is going to be used and which extruder if the printer had multiple ones.

Ultimaker 2+ Connect was the smaller one, print bed was of a normal size and it could have 1 spool of material loaded into it at a time and had a single extruder for printing, we used this machine to print smaller parts even though you could

fit a lot into the print bed, it also had a function to upload G-code print files into it over the network but we just used an USB stick with the files on it and chose it on the device after plugging it in, we didn't have any issues with materials using this printer of which we used PLA, PA and ABS. The only issue we had with the machine was one part that needed the print to be paused at a certain layer for us to add a nut into the print, which you could do as a function in Cura, adding a pause line of code at a certain layer that would tell the printer to pause the print, after a few tests and research we found out that the printer does not support that feature.

Ultimaker S5 we used for bigger prints like the chassis of the bot, it had slots for 2 spools of material and had 2 extruders to the materials which enabled material to be switched mid print, we did not use that feature. Materials used to print were mostly TPU 95A and TPC 45D, both of which are fairly similar flexible yet rigid materials but come with a longer print time, this machine supported G-code functions such as pause at layer, which we tested out but in the end did not need. With this machine we had a few problems such as material not printing properly which was an issue in one TPU 95A spool in which the material was too damp or the extruder getting blocked which leads to print being paused.

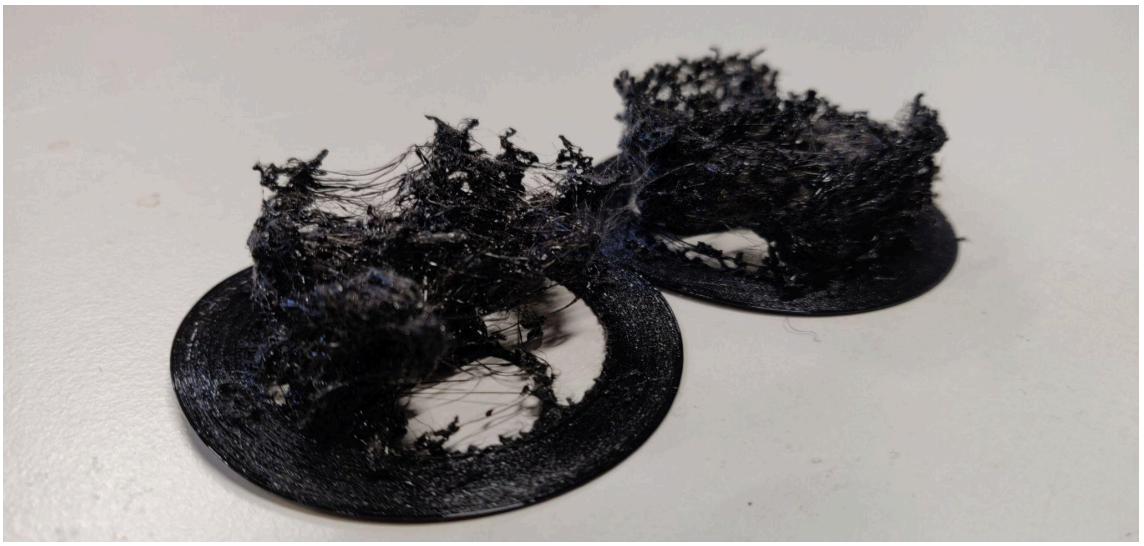

Figure 40. Damp material resulted in a failed print

Figure 41. Print paused because extruder/material spool was stuck, the error message is generic and just tells that it is a material feed issue

6.1.2 Bambu Lab X1 Carbon

This printer was the newest one and only arrived during the last week of our project, compared to the Ultimaker printers it was at least twice as fast, had a better print quality and could be monitored remotely using the Bambu Handy phone app, the materials available were PLA and PLA-CF but more variety was coming later because the printer used different size of material than what Robo Garage had for the Ultimaker printers, the printer also had an Automatic Material System addon attached which could be loaded with up to 4 spools of material and the machine would automatically load them into the extruder when printing began, this enabled the printer to change materials mid print or to use another material for supports. The printer had a camera and a LIDAR which it used to check if the material was being printed out correctly during the first layer and would send a message to the Bambu Handy app if there were problems. The last week of the project when this was available for us to use we did 14 prints with it, because of the speed of the printer sometimes the preparation for the print would take longer than the actual printing time.

The splicing software for the Bambu Lab printer was Bambu Studio, in which you could do the same tasks as Ultimaker Cura but in addition you could edit the 3D files before printing such as adding more shapes, cutting it or adding text to it, we used the cut feature on the front wedges and some other parts to get weight down in the last week. In order to get the G-code file into the printer you would scan a QR code on the printer with your phone using the Bambu Handy app and link it to your account, which would reserve the printer for your account to use, with the same account logged into the Bambu Studio software the printer would appear into it as well, which enabled you to send the G-code over the internet to the machine, along with selecting which material to use and monitoring the print over the studio software or Bambu Handy app, if you had an SD card in the machine you could also record a timelapse of the print using the internal camera.

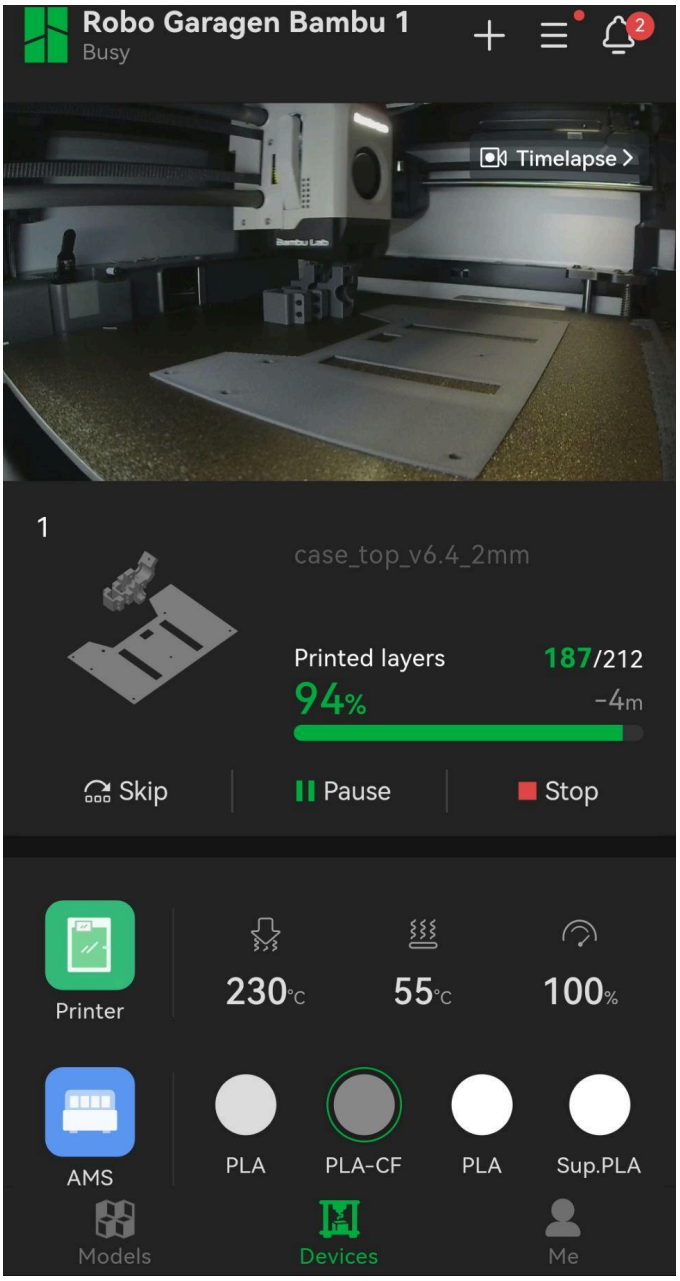

Figure 42. Bambu Handy app showing the progress of a print, camera enabled

6.2 Laser cutter

For the weapon we needed to have parts cut out from steel as we designed a shuriken, Niko Hutri, our project manager had got us access to the machining side of Metropolia Myyrmäki campus for us to get parts laser cut, the machine was a Prima Power Platino Fiber EVO 1530. We met up with project engineer Aapo Tapaninen who told us what we would need for the laser cut, which was a .dxf file of the shuriken, a 2D drawing of the 3D file, we exported a .dxf file from AutoDesk Fusion but it had probably corrupted during the export, Aapo fixed the file for us and then began the laser cut from a 2mm thick construction grade steel which was already loaded into the machine for other parts to be cut, in total we got 10 of the shuriken type parts cut of which we used 8.

Figure 43. Prima Power Platino Fiber EVO 1530 laser cutter from the side

Figure 44. parts being cut inside the machine

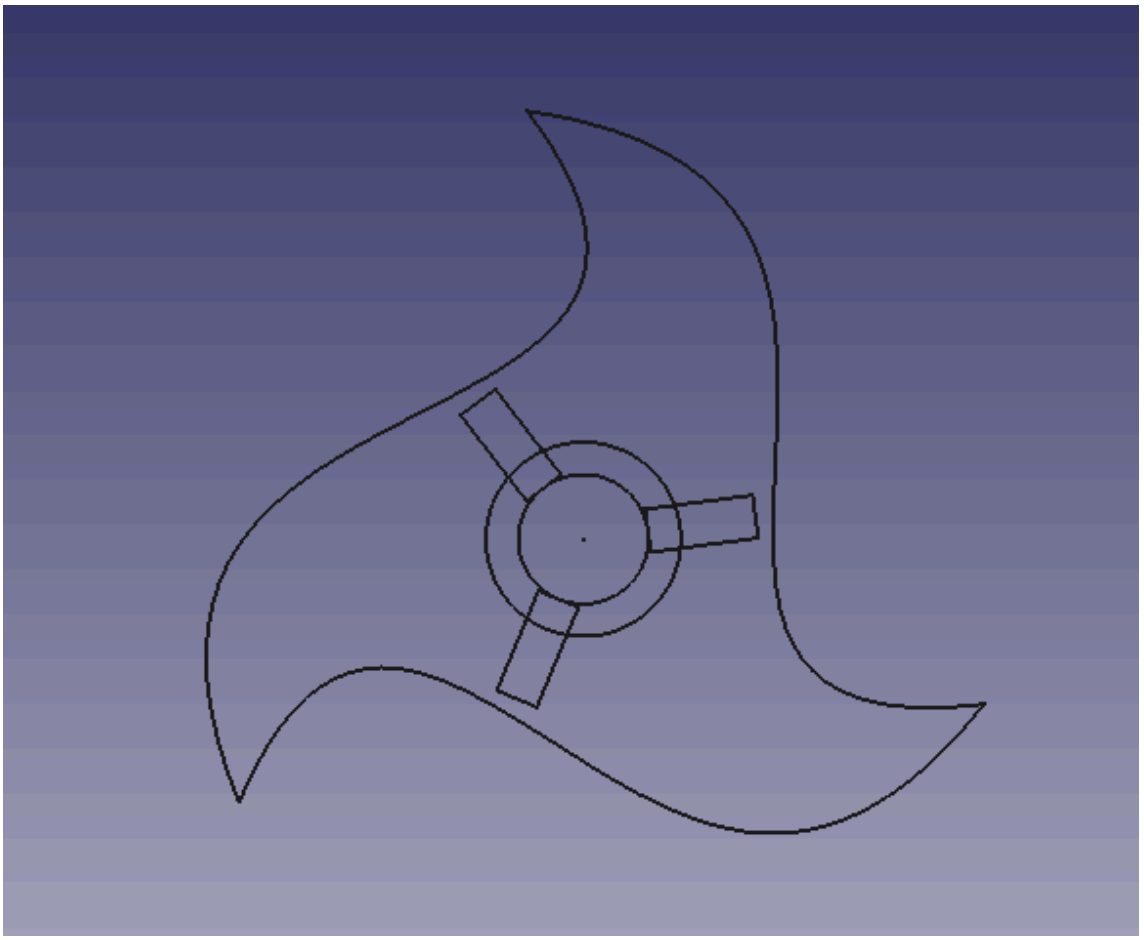

Figure 45. corrupted .dxf file, in the center you can see the proper model as a dot but its dimensions are smaller than it should be.

Figure 46. Finished shuriken after being laser cut

6.3 Welding

After we had our weapon shurikens laser cut we went to the machining side again to look for someone to aid us in the welding which was required for the weapon, instead of getting supervised for the welding process we thankfully got the help of Metropolia staff member Juho Jalava-Kanervio (Project Manager in machining side) who said he could do the weld, provided that we made a technical document describing the welding process, where to weld and what, which we delivered and he approved it, we don’t know what technique or style of welding was used for it.

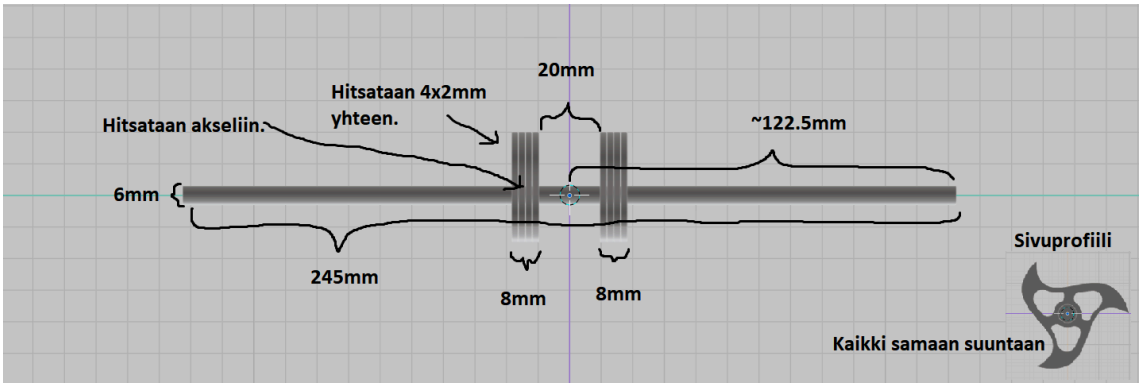

Figure 47. Technical document for the weld

Figure 48. Finished weld

7 Results

Overall the result of the project was a success, our goal was to have a finished bot and join the robosota competition without being disqualified, we had finished our bot a day before the competition and had printed spare parts for it just in case, which we ended up using, in the competition we exceeded our goal of just joining and competing in it by getting into the finals and placing 2nd in the 450g weight class, we also ended up promoting robo garage a bit by showcasing our project to a class of 3rd year embedded systems students a day after our project presentation.

Technical details of the bot
Weight: 447g
Dimensions: 18 x 14 x 4 cm
Weapon Tip Speed: 27 m/s
Weapon RPM: 12 800
Weapon Force: 34 N or 3.5 kg-force
Driving Top Speed: 1.2 m/s
Build Time: ~3 months
Body Materials: TPU, PLA, PLA-CF
Weapon Material: Laser cut steel welded to a steel rod.
Battery: 12.6V Li-Po 500mAh
Controller: PS4 Dualshock via Bluetooth

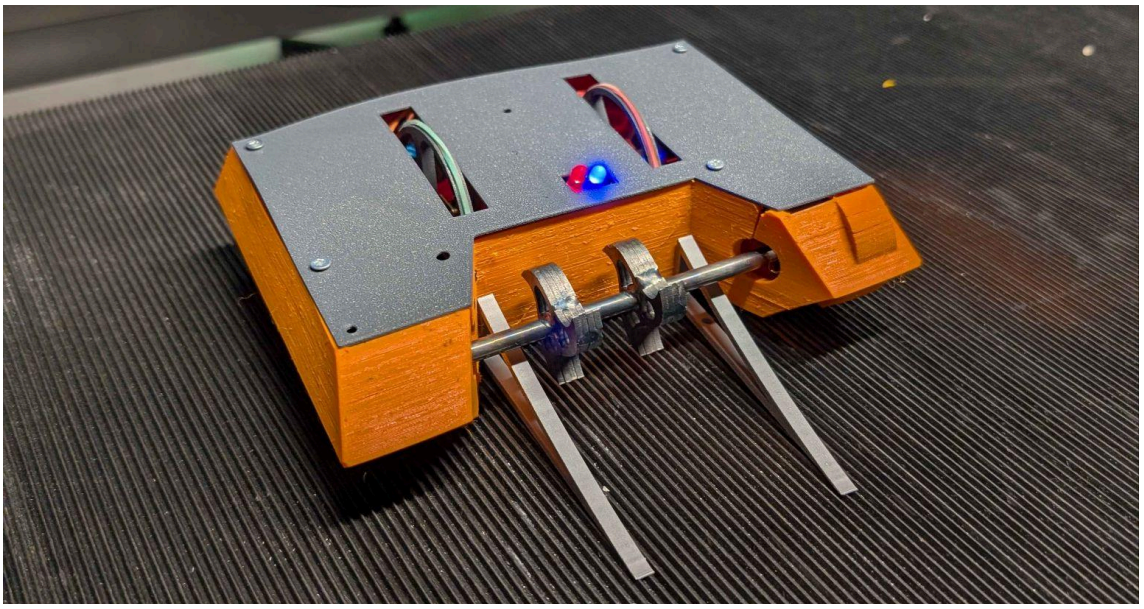

Figure 49. Finished bot with cover and power on

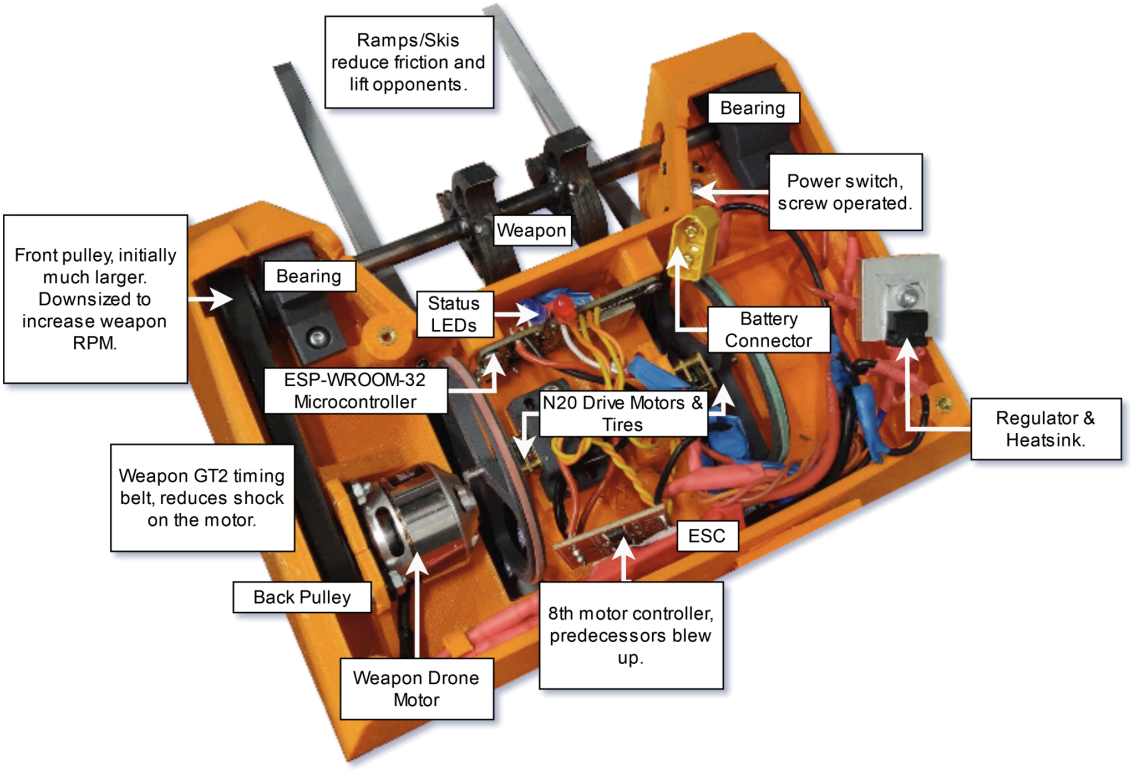

Figure 50. Bot without cover and showcasing parts