NTNU

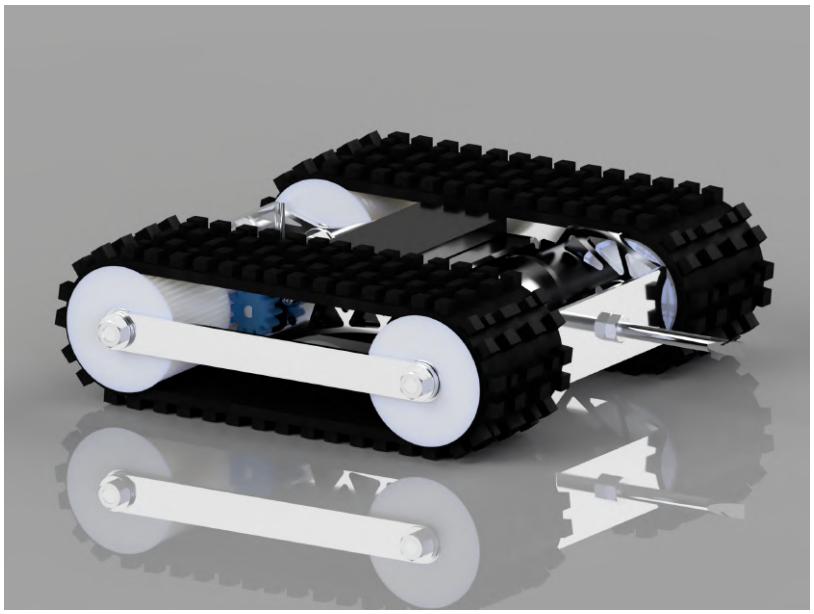
Kunnskap for en bedre verden

Department of Mechanical engineering

TMM4150 - Machine Design and Mechatronics

Battlebot Project

Authors:
Giuseppe D'Auria
Jonas Halbgewachs
Brage Sterkeby Hole
Marlon Guttormsen Mathisen
Kevin Velde Njå
Simon Desta Selassie
Sigurd Lennasønn Tjelle
Boyan Yu



Date: 30th January 2025

Table of Contents

List of Figures

List of Tables

1 Introduction

The objective of this project was to design and produce a battlebot that would be able to compete against other bots in a competition. The final concept and its specifications support the team's strategy of stability, torque, and mobility to beat the opponents. Due to limitations stated in the competition guidelines, trade-offs had to be made. In this case, the main trade-off was to neglect a major weapon in favor of a more stable battlebot with more powerful motors. The driving capabilities were to be achieved using drill motors, combined with continuous tracks, consequently allowing the battlebot to climb any obstacle in the arena, as well as force the opponents into the sandpits. The choice of weapon was therefore a spear, hooking onto the opponent's chassis. The rigidity and mobility of the battlebot, combined with the weapon solution was anticipated to be a competitive strategy able to perform well in the competition.

2 Process Description

The process of designing and building the battlebot followed an agile development process, characterized by iterative, collaborative methodologies, and weekly scrum meetings (documented in Appendix D). The design phase started with extensive discussions and brainstorming sessions to generate different concepts for the bot. Concurrently, tasks were strategically planned and assigned among pairs of group members, creating a collaborative and organized workflow. After the completion of some tasks, the systems would undergo testing either individually or in conjunction with others. These tests were then evaluated and each system would go through more iterations of refinement and improvement. This approach persisted until the competition date, enabling the battlebot to evolve and steadily improve.

2.1 Idea Generation

The initial weeks were dedicated to brainstorming concepts for the bot. The goal was to generate a wide range of ideas to evaluate, focusing on systems such as movement, chassis, weapons, and actuators (some ideas discussed are shown in Appendix E). There was a screening process conducted to select the concepts for further development based on factors such as weight and energy consumption. The selected concepts included the choice between tracks or wheels, the selection of electronic components, and considerations about incorporating magnets into the chassis, along with the possibility of adding a plow.

However, a consensus was not reached on the weapon system. Ideas discussed include a pneumatically powered spear, a plow, a motor-powered spinning weapon, or a passive net to disrupt the spinning weapons of the competitors. The prioritization of other systems was agreed upon by the team, and the decision to implement the most suitable weapon was agreed upon in a future scrum meeting.

By the third scrum meeting, a concrete concept for the bot had been established. The chosen design incorporated the use of tracks to maximize surface contact and powerful Rs550 DC motors to ensure higher torque. The inclusion of magnets in the chassis aimed at artificially increasing the down-force, while a plow was chosen as a defensive mechanism. The integration of these elements into the concept provided a strategic advantage. Specifically allowing the bot to effectively push opponents into one of the sand pits, forcing them to remain stuck.

2.2 Concept Development and Prototyping

The process of developing and prototyping the concept proved to be both time-consuming and more challenging than anticipated. Various ideas were explored for both mechanical and mechatronic systems as mentioned in Section 2.1. The upcoming section will delve into further exploration of the outcomes of the idea generation and screening phase, and the reasons behind the decisions made for the final concept.

2.2.1 Steering System

The steering system was a critical aspect of the design process, with two initial concepts considered: Ackermann Steering using traditional wheels and tank steering with continuous tracks. The team first delved into Ackermann Steering, given its potential for good maneuverability and predictable handling. However, during the early stages of CAD modeling, drawbacks emerged, revealing its large turning radius, potential reduction in steering advantages during abrupt maneuvers[1], and the need for additional components adding weight, complexity, and susceptibility to impact damage. Figure 1 (a) shows a 3D model of the Ackermann Steering concept.

The tank steering concept offered distinct advantages such as the ability to drive equally well in both directions, simplicity, and fewer moving parts. The main disadvantage of this concept, compared to Ackermann, is that well performing and easily controllable steering is harder to achieve [1].

After thorough evaluation considering competition limitations, the tank steering system, featuring continuous tracks and a drive-train with drill motors, was chosendue to its superior mobility. Figure 1 (b) shows the assembly of the steering system.

(a) Ackermann steering concept (b) Tank steering assembly

Figure 1: Steering concepts

The steering system consisted of drive wheels and passive wheels. To establish a robust connection between the wheels, a pair of 8mm diameter aluminum rods served as axles. These rods were supported by bearings on either side of the wheels, enhancing overall stability. The inner section of the drive wheel was designed as a spur gear, meshing with a separate gear attached to the motor shaft. The gearing improved motor efficiency and reduced the resistance generated by the tracks. Both types of wheels were equipped with outer flanges, preventing lateral sliding of the tracks. To enhance traction between the tracks and wheels, the outer surface was designed with a pattern. Different patterns were tested to maximize friction including axial lines along the surface and knurling patterns in different sizes. The wheelbase and outer diameter of the wheels went through many iterations to achieve the correct amount of slack in the tracks. Refer to Figure 2 (b) for detailed drive wheel assembly.

Figure 3 shows iterations of the steering system assembled to the chassis, excluding tracks.

2.2.2 Motor and Gears

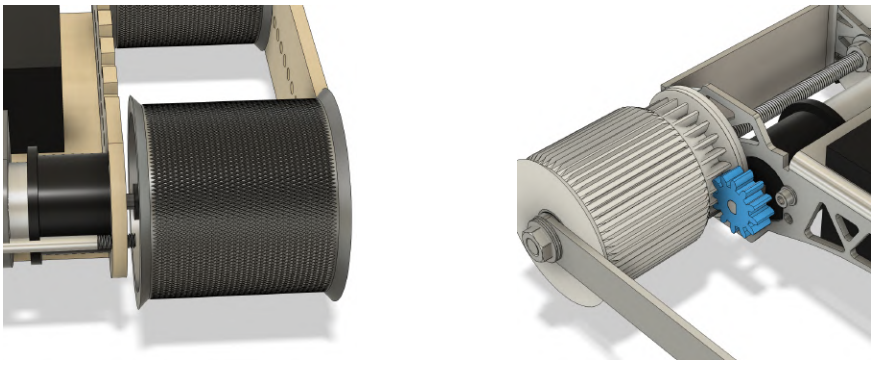
In the battlebot, two 12 V universal RS550 DC motors were utilized, with one for each track. The selection of these motors was influenced by some members' familiarity with them. Additionally, the motors' power and two-stage planetary gearbox were advantageous for achieving low RPM and high torque. The gearing specifics of the gearboxes were unknown, but a practical approach was used to evaluate if the gearing was adequate. Early testing without analytical gear ratio determination revealed satisfactory speed when connected directly to a 12 V battery.

However, when the motors were attached directly to the drive wheels, a concern arose regarding power consumption. This was due to the friction of the track systems having low slack, causing the motors to have a lot of resistance hence requiring more power to rotate. Another factor increasing friction in the drive train was the motors being directly connected to the drive wheels. This resulted in a bending moment on the gearboxes, causing excessive resistance in the gears. To address this, a solution was attempted by altering the wheel mounting from the motor shaft directly to a separate axle passing through the wheels, with a spur gear connecting the motor axle to the wheel.

Initially opting for a low gear ratio to increase motor speed, the bot's speed proved too slow during

testing, prompting a shift to a 1.5:1.0 ratio to regain most of the speed. The adjustment of placing the wheels on a separate axle through the chassis proved superior for both chassis strength and provided a more robust mount compared to the previous configuration where the motor axle only extended 15mm into one side of the wheel.

Spur gears were chosen over herringbone gears as it was easier to print and install. Helical gears were dismissed as they would produce an axial force, potentially loosening the gear. The shorter lifespan of spur gears was deemed negligible due to the maximum required lifetime of the battlebot being nine minutes (three matches, each three minutes long).



(a) Wheels mounted on motor axle

(b) Wheels mounted on separate axle

Figure 2: Wheel mounting

2.2.3 Chassis

The battlebot's chassis played a key role throughout the production and testing phase, undergoing continuous refinement. Initially, a 3D model of the chassis was created using Fusion 360. This software choice allowed efficient file management and let team members easily contribute to the assembly while working on different components.

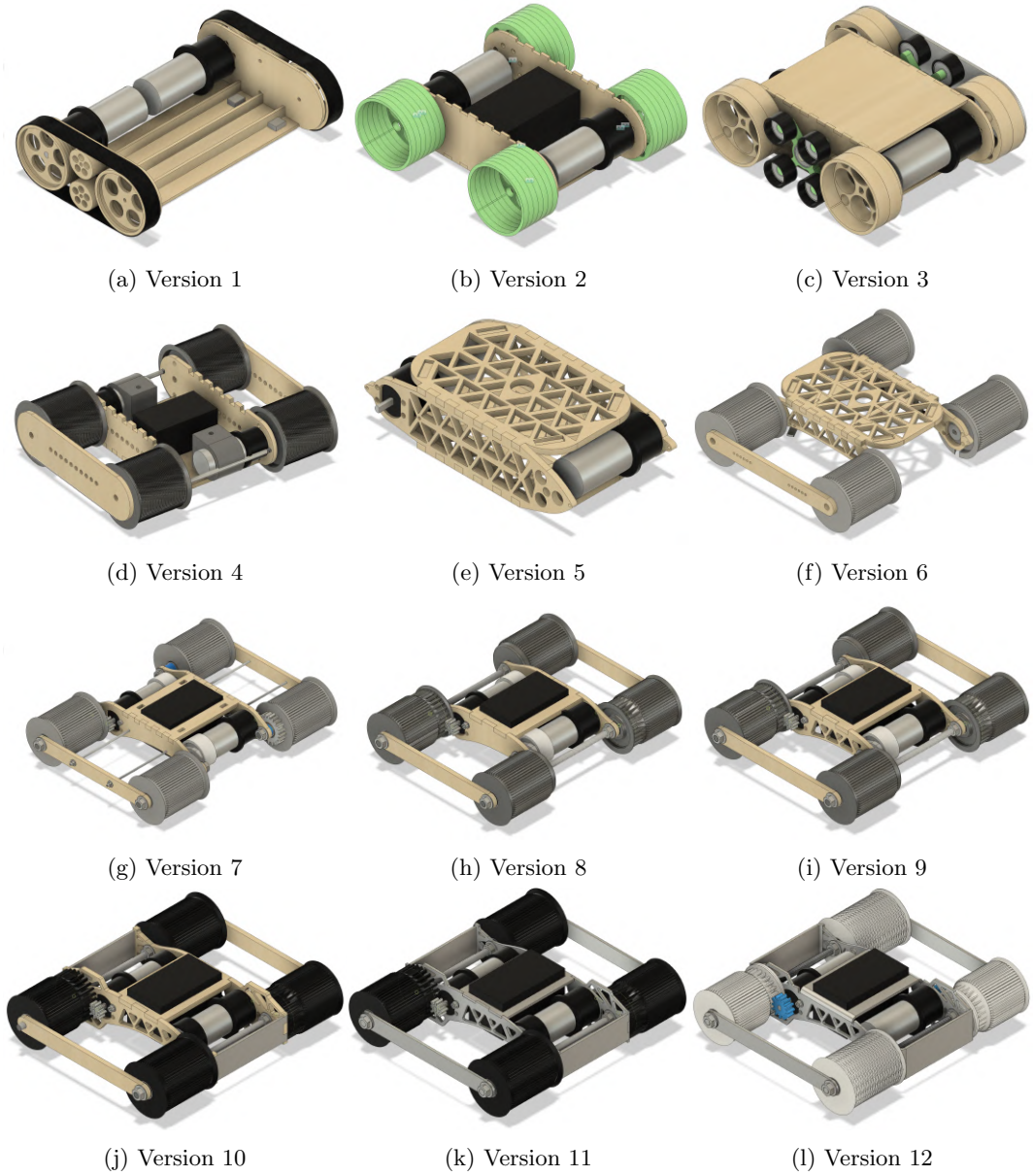
When finalizing the 3D model iterations, the different chassis elements were prepared for laser cutting. The produced components were assembled by hand to create the prototype structure for testing. This allowed for rapid prototype testing .Figure 3 shows the 3D models of each iteration.

The chassis design aimed to fulfill two essential requirements: the ability to drive upside down and maneuver out of the sandpits. The initial concept, Version 1 (V1), was abandoned in CAD due to the risk of getting stuck in the sand or on the edges. V2, employing a flipped motor configuration, was the foundation for further testing and iteration. V3 was a refinement of V2 with the addition of pulley wheels. However, the pulley wheel assembly was deemed too heavy to be used. V4, a fully parametric model, facilitated quick alterations and also introduced side skirts to the chassis to maintain the stiffness of the wheels. V5 addressed weight concerns with cutouts while maintaining structural integrity.

The removal of the floor in V6 improved ground clearance and mobility in the sandpits. Testing was done with a larger battery than what was going to be used for the competition. However, when testing a smaller battery, it became clear that there was too much friction in the drive wheel assembly. The suspected reason for this was that the drive wheel created a bending moment on the planetary gear box as previously mentioned. V7 addressed this by connecting motors to a spur gear, reducing friction, and allowing for gearing experimentation. However, it widened the battlebot by approximately 4 cm. V8 aimed to maintain the compact dimensions of V6 by integrating the spur gear internally as shown in Figure 2, and V9 refined this design with weight reduction. The V9 chassis ended up weighing 85g using 6 mm MDF sheet.

As the competition date neared, it became apparent that competitors' weapons posed a greater threat than anticipated. In response, V10 introduced aluminum plates in the front and in the back of the

chassis for enhanced protection against spinning weapons, while also providing additional space for electronics. During crash testing, it became apparent that the MDF roof was the weakest point in the chassis. Subsequent versions (V11 and V12) used aluminum for the entire chassis. V11 used the same shape as V10 epoxied together with the final version, V12, being a one-piece bent and welded assembly. V12 was preferred for its stiffness and durability. Overall, the chassis was designed as compactly as possible both to keep the weight down, but also to minimize the footprint and maximize the mobility.



(a) Version 1 (b) Version 2 (c) Version 3

(d) Version 4 (e) Version 5 (f) Version 6

(g) Version 7 (h) Version 8 (i) Version 9

(j) Version 10 (k) Version 11 (l) Version 12

Figure 3: Evolution of Chassis

The group considered improving the bot's grip by utilizing magnets on the ground, given that the battle arena floor is magnetic. Two concepts were explored for this purpose. The first involved a magnet that could be toggled on and off. This magnet could be activated and lowered from the bots' chassis to hinder opponents from pushing the bot, functioning as an anchor when activated. However, solutions involving the toggling of magnets and securing them to the chassis were deemed overly complex compared to the potential benefits.

The second concept was a more passive approach, involving mounting magnets on the chassis as close to the floor as possible without making direct contact. However, this concept was abandoned to prevent excessive resistance on the motors due to increased friction from the floor, which could drain

the battery more quickly and potentially deplete it during the competition.

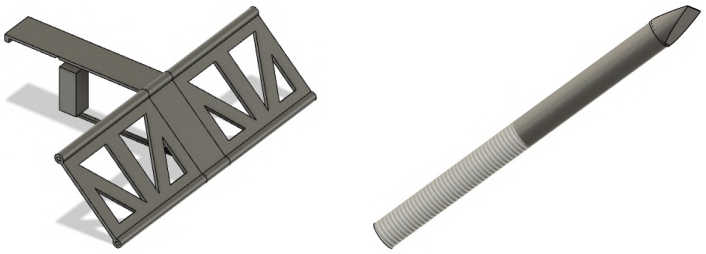
After careful consideration, magnets were excluded from the bots' features. The first idea with toggleable magnets seemed too complicated, and the second one lacked strong enough magnets. The bot performed well without them. In tests, it easily moved through sand, climbed a steep slope, and showed good speed. The bot's grip, torque, and speed were impressive, giving confidence in its abilities without relying on magnets.

2.2.4 Weapon

The exploration of weapon concepts, as mentioned in Section 2.1, primarily centered around the choice between an active and passive weapon. The emphasis during concept development was on assessing the advantages and drawbacks of each option. An active weapon, capable of delivering more significant damage, was considered. Particularly one with a vertically spinning mechanism utilizing high kinetic energy.

However, the group recognized certain drawbacks associated with an active weapon, specifically in terms of power consumption and weight. The potential drain on available power raised concerns about compromising driving capabilities and overall mobility by requiring the use of weaker motors. Additionally, the active weapon's assembly would contribute to increased weight and spatial requirements, impacting the bot's agility in the arena.

Practical constraints, including power consumption and size considerations, favored the decision for a passive weapon. The selected passive weapon concepts were a plow or a spear (shown in Figure 4). While designing the plow, concerns arose about its impact on the bot's ability to drive when flipped and climb out of sand pits. Additionally, the plow presented challenges in assembly, added significant weight, and hindered mobility in the arena, particularly on the ramps out of the sandpits.



(a) Plow concept (b) Spear concept

Figure 4: Weapon concepts

The chosen weapon was a spear fixed onto the chassis and axle. The spear comprised two parts: a threaded aluminum rod and a steel tip. The aluminum rod's threaded section is connected to a threaded mount on the front axle. Stabilization and reinforcement were further ensured by nuts on each side of the front wall. Opting for an aluminum rod aimed to cut weight, while the steel tip was chosen for its hardness and resistance to deformation [2]. The steel tip's threaded connection to the rod, reinforced with epoxy, accounted for structural compatibility and weight conservation. The spear allowed functionality when the bot was upside-down and had minimal impact on the ability to climb out of the sandpit due to its length. It could potentially inflict mor damage than the plow due to its sharp edge, backed by the bot's motor speed, capable of dislodging opponents' components.

A string reel was integrated into the chassis and spear. The reel connected to the front axle, and the string linked to an indentation in the spear. This setup served as a defense against spinning weapons, where the string would release from the reel and entangle the spinning weapon upon contact, diminishing its effectiveness.

2.2.5 Mechatronics

As mentioned above, the concept demanded high torque and grip. Therefore, the mechatronic prototyping consisted of trying to power the Rs550 DC motors. Initial tests with the battery proved that it was sufficient to power both motors. A powerful and efficient motor control system was necessary.

The initial phase of motor control involved the use of an DFRobot Dual H-Bridge based on the L298N IC to manage the speed and direction of two motors. Unfortunately, this controller proved inadequate as it only permitted speed adjustments and lacked the capability for direction changes. A defective controller was identified as the cause, validated by successful direction changes with a smaller controller. It was decided to continue testing with a similar controller based on the same L298N IC.

Tests with an Arduino Motor Shield based on the same L298N as the first controller, allowed for both direction and speed control, yet the overheating issues persisted. Despite adhering to the specified 2 A current limits for each motor, as indicated in the shield's data-sheet, the motors experienced significant power reduction. It suffered from a 4.9 V voltage drop at maximum power output, resulting in a substantial 40% heat loss. Dissatisfied with these constraints, the team decided to design a more efficient motor controller to enhance performance.

The primary goal of the custom motor controller was to create a straightforward circuit while maintaining optimal control over the motors. The initial MOSFET H-bridge, designed for concept testing lacked pull-down resistors, leading to control challenges. Pins left floating while the Arduino initialized allowed the current to flow, causing internal short circuits, excessive heat, and MOSFET damage. The second version of the H-bridge transitioned to a 2P-channel 2N-channel design, prioritizing simplicity and ease of production. This change facilitated a more straightforward design approach.

In response to issues with internal shorting, the final version incorporated pull-up and pull-down resistors to ensure safe powering before Arduino initialization. To address the voltage difference between the 5 V output pins and the 12 V on the source for the P-channel MOSFETs, NPN transistors were introduced to drive said MOSFETs. This involved grounding the gate through a transistor and allowing the pull-up pin to manage the MOSFETs' off state.

A buck converter was used to power the Arduino, even though the Arduino could technically be powered from the 12 V on the battery. This was done because the buck converter smooths any voltage spikes caused by the motor inductance. The buck converter then protects the Arduino Vin pin from excessive voltages.

2.2.6 Motor Control Arduino Code

Motor control with the Arduino utilized a remote control channel as an activation condition, allowing deactivation to prevent unintentional movement. Control stick values were mapped to a -255 to 255 scale. To simplify and due to limited testing, motor control operated in a binary mode, with motors either stopped or at maximum power.

When the remote control's threshold value of 150 was reached, relevant pins were set to HIGH. If the value fell below 100, the pins returned to LOW, stopping the motor. A brief delay was included in the code when changing directions to ensure MOSFET gates discharged and closed before reopening in the opposite direction. The delay length was determined by using an oscilloscope to measure the discharge time of the MOSFET gates.

2.2.7 Battery Management System (BMS) and Streaming

Using an Arduino Uno to measure and display the battery level was considered. The battery level was measured using a simple voltage divider. This proved to be too space-inefficient of a solution. Some changes, like reducing the amount of LEDs and switching to Arduino Nano, were implemented, but it was eventually decided to integrate BMS into the streaming system. This was due to the lack of available I/O pins on the Arduino Nano handling the remote control. The final BMS circuit schematics is found in Figure 8.

nRF24L01 modules were initially used to send motor data from the battlebot's Arduino to a receiver connected to another Arduino linked to a laptop. The plan involved printing data on the Serial Monitor in the Arduino IDE and eventually posting it to a web server. However, the nRF24L01 modules lacked support for streaming to a web server during the event. Additionally, data stability and quality issues were observed during testing, making it challenging to show useful information for the Serial Monitor. There were also difficulties in confirming the proper functionality of the nRF24L01 modules.

To address these challenges, a solution involving the DOIT ESP32 DevKit v1 module, which is based on the ESP32 micro controller, was proposed. This module offers WiFi, Bluetooth, Ethernet, and Low power capabilities on a single chip.

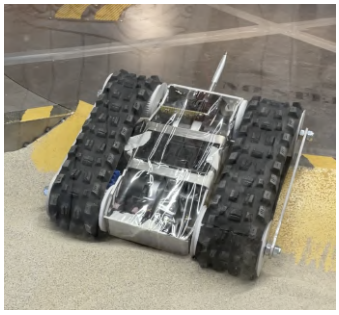
The experience with the new chip was not as smooth as expected. A significant amount of time was used on troubleshooting and debugging. Due to lack of prior experience, the most difficult part was often not solving the problem, but rather identifying the problem. Some of the problems that were encountered are the following:

- **Connecting to Eduroam.** This issue was solved by connecting the module to a wireless hotspot instead.
- **Incompatible Arduino IDE version.** SPIFFS (Serial Peripheral Interface Flash File System) were used to manage files in the web server folder. To be able to use SPIFFS the "ESP32 Sketch Data Upload" feature was needed, which is not yet compatible with the Arduino 2.0.
- **Insufficient amount of pin I/Os.** The original idea was to send motor data onto the web server. To do this, the ESP-Module needed to be connected to the motor driver module (Arduino Nano), which was difficult due to the limited amount of I/O pins available on the Arduino. To work around this, the battery level was posted onto the web server instead. This only required the ESP32-module to be connected to the battery instead of the motor driver.
- **Calibration of battery voltage readings.** Although the ESP-32 offers a 12-bit ADC (Analog input values go from 0 - 4095), The ADC did not behave linearly and led to inaccurate readings. To solve this, the analog-to-voltage calculations were calibrated using external mathematical formulas (shown in appendix A) found by interpolating different readings. With this, the module could read the critical voltage (11.9 V) with an error of approximately ± 0.05 V.

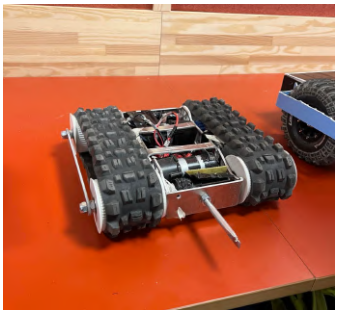
3 Product Documentation

3.1 Final Product - Beltebassen

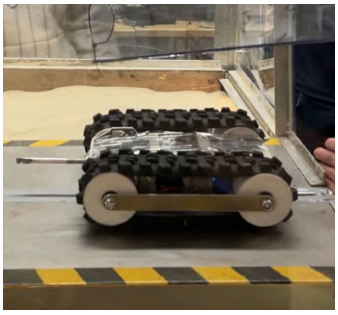
The bot is called Beltebassen. The final battlebot design as shown in Figure 5 incorporates continuous tracks made from a mountain bike tire, leveraging its great friction properties and maximized ground contact to optimize traction. Each track is powered by an individual motor, providing the bot with substantial torque. The exceptional traction from the tracks, coupled with the powerful motor torque, enable the bot to overpower opponents during collisions. The motors are controlled by a self-built H-Bridge.



(a) Beltebassen climbing out of sandpit



(b) Presentation prior to competition



(c) Beltebassen driving upside down

Figure 5: Final product: Beltebassen

For offensive capabilities, the battlebot features a front-mounted spear with a steel tip, increasing damage inflicted on opponents during collisions. This spear also serves as a protective measure, keeping opponents at a distance and safeguarding the chassis during frontal collisions. The string can disarm spinning weapons by entangling itself in them during contact.

The battlebot's design has a sleek, compact and lightweight aluminum chassis, offering resistance against damage. Additionally, side skirts on the outer edges of the tracks contribute to structural integrity and minimize potential damage to the tracks from opponents' weapons.

3.2 Part List

Figure 6 shows the exploded view of the chassis design in fusion 360.

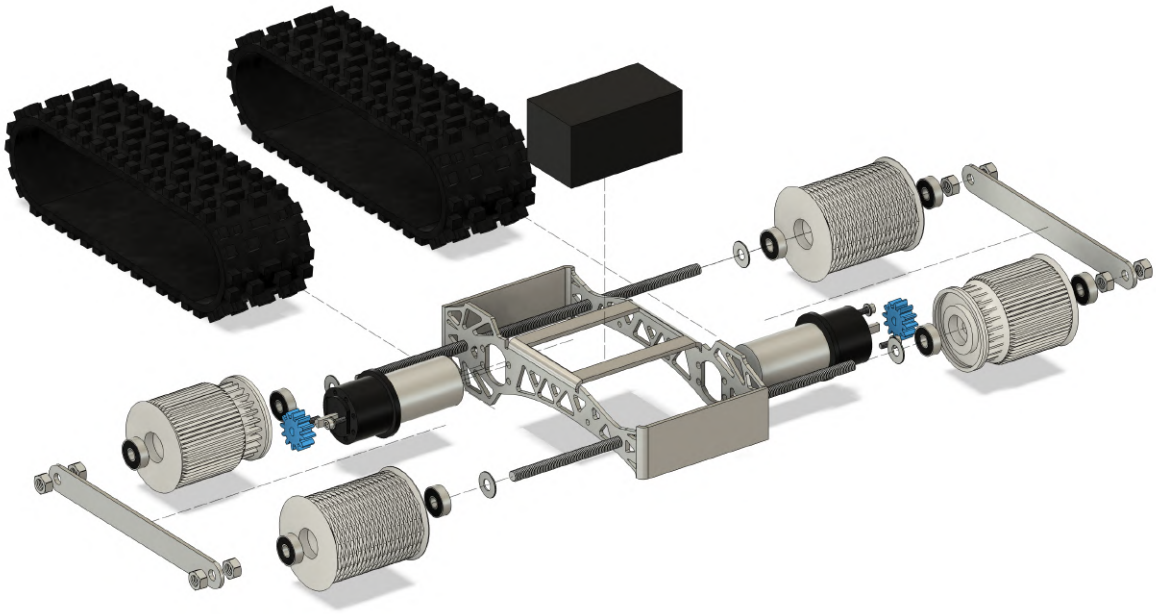


Figure 6: Exploded view of the final assembly

Table 1: Parts List

ITEM	**PART NAME**	**QTY**	**MATERIAL**	**WEIGHT**[g]
1	MOTOR W/ PLANETARY GEARBOX	2	-	340
2	SPUR GEAR	2	PLA	4
3	CHASSIS W/ SIDESKIRTS	1	ALUMINUM	210
4	SPEAR	1	ALUMINUM W/ STEEL	12
5	M4 SETSCREW	4	STEEL	0.25
6	M4 BOLT	4	STEEL	0.7
7	M8 NUT	10	STEEL	4
8	DRIVE WHEEL	2	PLA	55
9	PASSIVE WHEEL	2	PLA	50
10	BEARING	8	STEEL	10
10	CONTINUOUS TRACK	2	RUBBER/BUTYL	280
12	STRING	2M	Nylon	5
13	STRING REEL	1	PLA	3
14	ROD	2	STEEL	50
15	ARDUINO NANO	1	-	19
16	H-BRIDGE	1	-	22
17	DOIT ESP32 DEVKIT V1	1	-	10
18	BUCK CONVERTER	1	-	11
19	MISC. ELECTRICAL COMPONENTS	-	-	15
20	TAPE	-	-	10
TOTAL				1998.8

3.3 Streaming

The streaming was solved by building an ESP32 web server. The battery voltage was displayed in a radial gauge. The gauges were constructed with the help of the canvas-gauges JavaScript library. The readings are updated automatically on the web page using Server-Sent Events (SSE). The final web page is displayed in Figure 7.

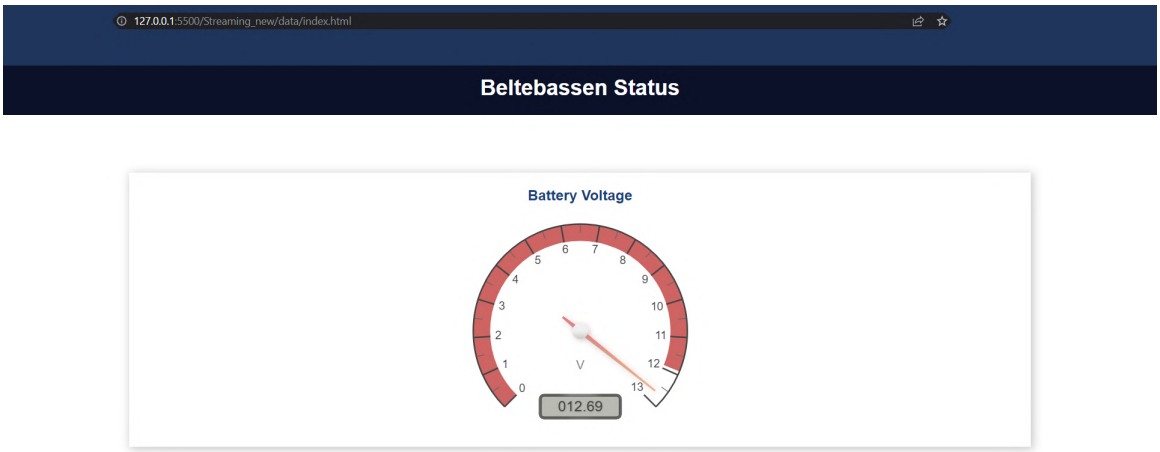


Figure 7: Screenshot of streaming web page

3.3.1 Arduino Code

The arduino code is run on Arduino IDE, and imports the following external libraries:

- AsyncTCP
- ESPAsyncWebServer
- Arduino_JSON

The function initWiFi will connect the ESP32 module to the WiFi "the fridge upstairs" using its network credentials.

The function getHighestValue(int[]) will find the highest values inside list parameter, and return it. This list is updated every second, appended with the newest analog input values. These values is stored there for 20 seconds. After this time period they will get overwritten by newer values. This function was implemented to counter voltage drops when the motors started running, causing the voltage readings to not correspond to the real-time battery voltage. With this function, the voltage pits were ignored, and the gauge would solely display the highest value read in the last 20 seconds.

The function initSPIFFS will check if the SPIFFS is mounted correctly.

In the setup function, the web server will be initialized. To make debugging more effective, it will send messages to the Serial Monitor if an error was to occur.

In the loop function, the ESP32 module will send updates to the web server continuously every second. The JSON text sent will also be printed out in the Serial Monitor to check whether the system is working. The final Arduino code for streaming is in appendix A

3.3.2 JavaScript Code

The code starts by initializing the event source protocol. Then it adds an event listener for the new_readings event. Consequently, the radial gauge is created and will display the latest sensor readings from the new_readings event. When accessing the web page for the first time, the code will make an HTTP GET request for the current sensor readings. The final JavaScript code can be found in appendix A

3.4 Mechatronics

3.4.1 Electronic Circuit

The final circuit was constructed as shown in figure Figure 8. The final design incorporated pull up and pull down resistors. The P channel MOSFETs were controlled through NPN-transistors. The H-bridge allowed each motor to be supplied with 5 A for a short while, and 3 A continuous. It uses a Buck Converter to power the Arduino. This is done to account for fly-back voltage spikes through the H-bridge which could destroy the Arduino.

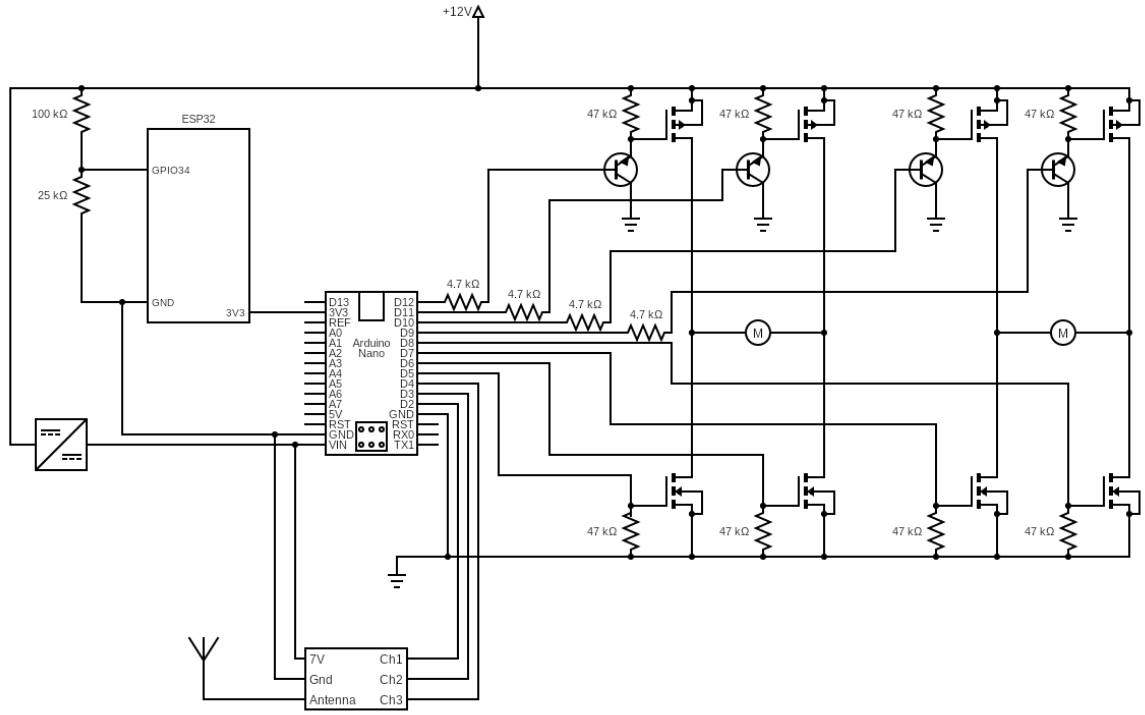


Figure 8: The final circuit design for the battlebot.

3.4.2 Arduino Code

The final Arduino code for the motor control and remote integration is in appendix A. The code incorporated a drive mode and an off mode, so the Arduino could be properly initialized without driving away. It had binary control for the motors, i.e no PWM control.

4 Discussion

4.1 Competition

During the pre-fight weighing, the bot weighed 2000 g excluding the battery. Subsequently, the battery was connected to the system, and a layer of tape was added to the chassis. This secured the wires in place and provided protection to the electronic components from conductive particles in the sand.

4.1.1 Battle 1: FrankTheTank

In this battle, Beltebassen experienced the most significant damage. Upon reflection of this battle, the sting proved to be ineffective against the opponent's spinning weapon. The opponent's spinning weapon caused deformation to the spear and bent the chassis. Additionally, it damaged one of the drive wheels and chipped the passive wheel on the same side. This resulted in the track almost slipping off towards the end of the round. To fix these damages before the next round, the chassis was bent back in place and the damaged drive wheel was replaced with a spare wheel.

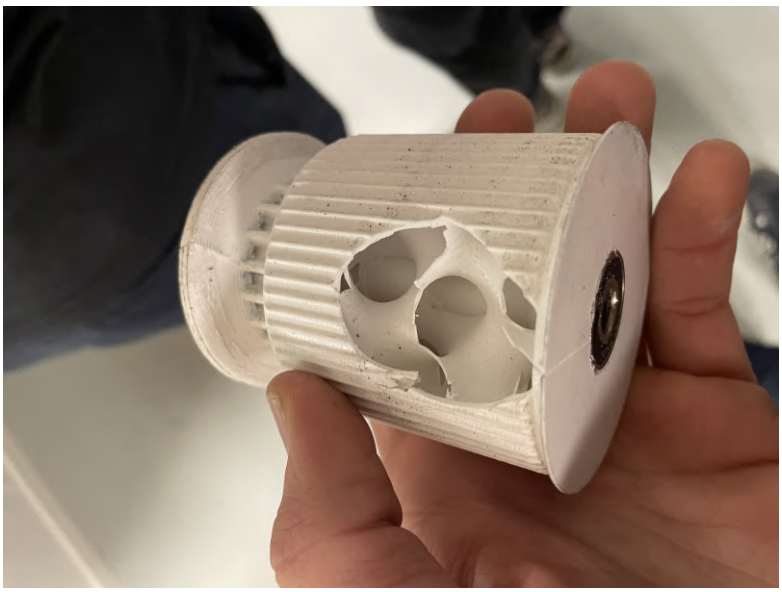


Figure 9: post-battle drive wheel

4.1.2 Battle 2: Tarantula Torque

This opponent was considerably larger than Beltebassen in size but utilized weaker motors and wheels. This resulted in a mechanical grip advantage for Beltebassen. Tarantula Torque had its chassis encased in foam, this worked in Beltebassen's advantage as it provided a great surface for the spear to stick into. The primary weapon of Tarantula Torque was a ramp that would flip its opponents. This did not happen during the match, and as Beltebassen was designed to be able to drive upside-down, it did not pose any threats. This match concluded with Beltebassen knocking out Tarantula Torque with a single strike, pushing it into the sand pits, making it unable to move.

4.1.3 Battle 3: Squirtle

This opponent sprayed water on its rivals, this strategy got them into the final round. Fortunately, this was not a concern for Beltebassen, thanks to prior protective measures. Since there was tape applied around the chassis, safeguarding internal components from sand, Beltebassen was already water resistant. In the initial round, Beltebassen successfully pushed the opponent into a sandpit which they

were unable to get out of. However, due to Beltebassen's high speed, it accidentally reversed into another sandpit. Beltebassen became immobilized due to its previously bent spear hindering the bot from driving up the ramp and the speed of the motors digging Beltebassen deeper into the pit. With both bots stuck, a second round was necessary to conclude the final.

Beltebassen had a lot better steering in the second round. this resulted in Blastoise being pushed into the sand pit and since Beltebassen was placed in front of it, Blastoise ended up digging itself into the sand.

Figure 10 shows how Beltebassen looked after winning the final with the tape and battery removed.

Figure 10: post-competition Beltebassen

4.2 Team work

The division into smaller subgroups helped in the time between the fights since each area of the bot had a responsible group. The group for the mechatronics disassembled the cover of the bot and disconnected the battery to save energy while the team that worked on the chassis and wheels did repairs after the first fight. Because of the subgroups everybody knew who was responsible for which parts which helped with having a high productivity in a very limited time frame.

4.3 Battery Management

The battery management was a smaller issue than expected due to the short duration of the fights. During the preparation, the assumption was that the fights would last up to three minutes which was a concern in terms of energy consumption. Since none of the battles lasted the entire 3 minutes, the battery voltage after the second fight was still at roughly 12.6 V. Both finalists were offered a new battery for the last battle and the voltage on that one was 12.8 V. To be safe, the old battery was replaced.

4.4 Chassis and wheels

The aluminium chassis played a crucial role in protecting the bot, especially during the initial battle, shielding it from potential damage by competitors. If a previous MDF version had been used, it might not have endured the impact of other bots.

In terms of design improvements, the chassis could benefit from slight weight optimization. The asymmetrical geometry was optimized for drive wheels with only an outer flange to contain the tracks. However, the latest version of the drive wheels allowed for flanges on both sides due to a different gearing ratio, which made the large sides where the motor was mounted less useful. As the inner

flange of the drive wheel was smaller than the outer flange, the large sides still served a purpose as a backup protection to hinder the tracks sliding over the chassis.

A weight reduction would allow for more weight to be used on the wheels. A larger wall thickness or a larger infill percentage would have made the wheels more robust. As seen in Figure 9, the wall thickness and infill were kept at a minimum to save as much weight as possible. This made the wheels prone to damage from high-rpm spinning weapons.

As for aesthetic improvements, painting the entire chassis black would have been a nice touch, but time constraints prevented this from happening.

4.5 Mechatronics

One significant advantage of the MOSFET H-bridge was the power consumption. In contrast to the Arduino motor controller it didn't have any significant heat loss. This was a clear advantage over the other groups who stuck with the Arduino motor controller. In addition, the H-bridge did not have the same 2 A current limit for the Arduino motor controller, allowing more current to be drawn from the battery. This showed during the competition when Beltebassen easily pushed away the other competitors with the increased power.

One disadvantage was that the H-bridge required 8 out of 14 digital pins available. Given more time, investigating the implementation of logic ICs to control the H-bridge would have been advantageous, as it would free up the pins on the Arduino Nano. The final H-bridge design was also complicated, and ended up being split into two separate parts. One part that did the signal processing, and another part that consisted of the H bridge circuit itself. In a future design, combining these two on a single prototype board would be advantageous.

A significant disadvantage was the lack of fine motor control. This made the battlebot difficult to control, which showed during the competition. PWM-control could have been implemented if there was more time to test the limits of the H-bridge.

5 Conclusion

Several strengths and weaknesses of Beltebassen were revealed during the competition.Firstly, Beltebassen proved mobile in the arena. It was able to climb out of the sandpit using the ramp, but also the vertical edge. In addition, the compact design proved beneficial as it was able to turn and maneuver rapidly in tight spaces, whereas larger battlebots got stuck in the sandpit due to their size. Furthermore, the fights highlighted the advantages of the powerful motors. The speed and force produced by the motors overpowered all of the opponents.

A second strength can be explained in terms of battery management. The choice of using a passive weapon proved beneficial in terms of battery consumption. Even though the motors' energy consumption was significant, measuring the battery after the fights showed it was well within the limit.

A severe disadvantage of the bot was the limitations of the motor controller. Only being able to turn the motors on or off in each direction made Beltebassen hard to control. The maneuverability could be improved by including a PWM control to manage its velocity. Another consequence of this was that it had a somewhat reduced capability to get out of the sand. Sometimes it spun the sand out of the way effectively beaching itself due to the inability to accelerate slowly.

The incorporation of the string turned out to be ineffective at hindering spinning weapons. This was due to a lack of testing and worked against Beltebassen as it only increased the weight of the bot.

As an overall summary of the project, the battle bot outperformed its competitors due to its simplicity and straightforward approach. Beltebassen optimized its torque and speed to defeat its opponents.

Bibliography

[1] H. Vestad, *Lecture 8 - components*, University Lecture, 2023.

[2] Erectastep. 'Aluminum or steel: Which material is better?' (), [Online]. Available: https://www.erectastep.com/aluminum-steel-metal-better/ (visited on 13th Nov. 2023).

Appendix

A Code

Folder Structure

```
Main Battlebot Streaming folder
├── Streaming.ino
└── data
    ├── index.html
    ├── style.css
    └── script.js
```

- **Streaming.ino** is the arduino sketch that handles the web server
- **index.html** is the html-file that defines the webpage
- **style.css** provides web page styling
- **script.js** is the script that programs the behavior of the web page and handles web server responses, events, create the gauges, etc.

Arduino Streaming code

```
#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include "SPIFFS.h"
#include <Arduino_JSON.h>

// Network credentials
const char* ssid = "The fridge upstairs";
const char* password = "boiboiboi";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

// Create an Event Source on /events
AsyncEventSource events("/events");

// Json Variable to Hold Volt Readings
JSONVar readings;

// Timer variables
unsigned long lastTime = 0;
unsigned long timerDelay = 1000;

// Define pins for the analogInput
const int analogInput = 34;

// Variables for the voltage divider
float v_out = 0.0;
float v_in = 0.0;
float R1 = 102000; // 47k + 56k
float R2 = 26700; // 27k
int value = 0;
float calibrated_v_in = 0.0;

// Counter & list
int counter = 0;
int list[20];
int max_analog_value = 0;

// Get Volt Readings and return JSON object
```

```
String getVoltReadings(int value) {
  // value = analogRead(analogInput);
  v_out = (value * 3.3) / 4095;
  v_in = v_out / (R2 / (R1 + R2));
  calibrated_v_in = ((v_in * 1.03697) + 0.43971);
  readings["input_voltage"] = String(calibrated_v_in);

  String jsonString = JSON.stringify(readings);
  return jsonString;
}

int getHighestValue(int list[]) {
  int max_v = INT_MIN;
  for ( int i = 0; i < 20; i++ ) {
    Serial.println(list[i]);
    if ( list[i] > max_v ) {
      max_v = list[i];
    }
  }
  return max_v;
}

// Initialize SPIFFS
void initSPIFFS() {
  if (!SPIFFS.begin()) {
    Serial.println("An error has occurred while mounting SPIFFS");
  }
  Serial.println("SPIFFS mounted successfully");
}

// Initialize WiFi
void initWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  Serial.println(WiFi.localIP());
}

void setup() {
  // Serial port for debugging purposes
  Serial.begin(115200);
  initWiFi();
  initSPIFFS();

  // Web Server Root URL
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {
    request->send(SPIFFS, "/index.html", "text/html");
    Serial.println("wawawiwow");
  });

  server.serveStatic("/", SPIFFS, "/");

  // Request for the latest volt readings
  server.on("/readings", HTTP_GET, [](AsyncWebServerRequest *request) {
    String json = getVoltReadings(analogRead(analogInput));
    request->send(200, "application/json", json);
    json = String();
  });

  events.onConnect([](AsyncEventSourceClient *client) {
    if (client->lastId()) {
      Serial.printf("Client reconnected! Last message ID that it got is: %u\n", client
    ->lastId());
    }
    // send event with message "wawawiwow!", id current millis
    // and set reconnect delay to 1 second
    client->send("wawawiwow", NULL, millis(), 10000);
  });
  server.addHandler(&events);
```

```
114
115   // Start server
116   server.begin();
117 }
118
119 void loop() {
120   if (counter < 20) {
121     if ((millis() - lastTime) > timerDelay) {
122       list[counter] = analogRead(analogInput);
123       max_analog_value = getHighestValue(list);
124       // Send Events to the client with the Volt Readings Every second
125       events.send("ping", NULL, millis());
126       events.send(getVoltReadings(max_analog_value).c_str(), "new_readings", millis());
127       Serial.println(getVoltReadings(max_analog_value).c_str());
128       lastTime = millis();
129
130       counter += 1;
131     }
132   } else { counter = 0; }
133 }
```

Listing 1: Arduino Streaming Code

Battlebot Arduino Control Code

```
const int H1Q1pin= 12;
const int H1Q3pin = 11;
const int H1Q2pin = 10;
const int H1Q4pin = 9;

const int H2Q1pin= 8;
const int H2Q3pin = 7;
const int H2Q2pin = 6;
const int H2Q4pin= 5;

int rc_ch1 = 4;
int rc_ch2 = 3;
int rc_ch3 = 2;

int ch1Min = 1300;
int ch1Max = 1300;
int ch2Min = 1300;
int ch2Max = 1300;

int ch1V;
int ch2V;
int ch3V;

int ch1Counter = 0;
int ch2Counter = 0;

void channelWrite(int chV, int Q1pin, int Q2pin, int Q3pin, int Q4pin, int deadband){
  if(chV > deadband){
    digitalWrite(Q1pin, LOW);
    digitalWrite(Q4pin, LOW);
    delay(20);
    digitalWrite(Q2pin, HIGH);
    digitalWrite(Q3pin, HIGH);
  }
  else if(chV < -(deadband)){
    digitalWrite(Q2pin, LOW);
    digitalWrite(Q3pin, LOW);
    delay(20);
    digitalWrite(Q1pin, HIGH);
    digitalWrite(Q4pin, HIGH);
  }
  else if(abs(chV) <= deadband){
    digitalWrite(Q1pin, LOW);
    digitalWrite(Q2pin, LOW);
    digitalWrite(Q3pin, LOW);
    digitalWrite(Q4pin, LOW);
  }
}

// SETUP ----------------------------------------------------------
void setup() {
  //For debugging purposes
  //Serial.begin(9600);

  // Defines the output and input pins
  pinMode(rc_ch1, INPUT);
  pinMode(rc_ch2, INPUT);
  pinMode(rc_ch3, INPUT);

}

// LOOP -----------------------------------------------------------
void loop() {
  ch1V = pulseIn(rc_ch2, HIGH);
  ch2V = pulseIn(rc_ch1, HIGH);
  ch3V = pulseIn(rc_ch3, HIGH);

  if(ch3V >= 1200){
      if (ch1V > ch1Max) {
        ch1Max = ch1V;
      }
```

```
      if (ch1V < ch1Min) {
        ch1Min = ch1V;
      }
      if (ch2V > ch2Max) {
        ch2Max = ch2V;
      }
      if (ch2V < ch2Min) {
        ch2Min = ch2V;
      }
  }

  if(ch3V >= 1200 || ch1V < 100){
    ch1V = 0;
    ch2V = 0;
    digitalWrite(H1Q1pin, LOW);
    digitalWrite(H1Q2pin, LOW);
    digitalWrite(H1Q3pin, LOW);
    digitalWrite(H1Q4pin, LOW);

    digitalWrite(H2Q1pin, LOW);
    digitalWrite(H2Q2pin, LOW);
    digitalWrite(H2Q3pin, LOW);
    digitalWrite(H2Q4pin, LOW);
  }
  else{
  ch1V = map(ch1V, ch1Min, ch1Max, -255, 255);
  ch2V = map(ch2V, ch2Min, ch2Max, -255, 255);
  }



// THIS IS THE CHECK FOR WIERD SPAZZY INPUTS

  if(abs(ch1V )>= 150){
  channelWrite(ch1V, H1Q1pin, H1Q2pin, H1Q3pin, H1Q4pin, 150);
  }

  if(abs(ch2V) >=150){
  channelWrite(ch2V, H2Q1pin, H2Q2pin, H2Q3pin, H2Q4pin, 150);
  }

  if(abs(ch1V) < 100){
    channelWrite(0, H1Q1pin, H1Q2pin, H1Q3pin, H1Q4pin, 150);
  }

  if(abs(ch2V) < 100){
    channelWrite(0, H2Q1pin, H2Q2pin, H2Q3pin, H2Q4pin, 150);
  }

}
```

Listing 2: Bot Control Arduino

HTML code

```
<!DOCTYPE html>
<html>
    <head>
        <title>Beltebassen Status</title>
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" type="text/css" href="style.css" />
        <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css" integrity="sha384-fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr" crossorigin="anonymous" />
        <script src="http://cdn.rawgit.com/Mikhus/canvas-gauges/gh-pages/download/2.1.7/all/gauge.min.js"></script>
    </head>

    <body>
        <div class="topnav">
            <h1>Beltebassen Status</h1>
        </div>
        <div class="content">
            <div class="card-grid">
                <div class="card">
                    <p class="card-title">Battery Voltage</p>
                    <canvas id="gauge-volt"></canvas>
                </div>
            </div>
        </div>
        <script src="script.js"></script>
    </body>
</html>
```

Listing 3: HTML Code

JavaScript Code

```
window.addEventListener('load', getReadings);

var gaugeVolt = new RadialGauge({
    renderTo: 'gauge-volt',
    width: 300,
    height: 300,
    units: 'V',
    minValue: 0,
    maxValue: 13,
    majorTicks: [
        '0',
        '1',
        '2',
        '3',
        '4',
        '5',
        '6',
        '7',
        '8',
        '9',
        '10',
        '11',
        '12',
        '13'
    ],
    minorTicks: 2,
    strokeTicks: true,
    highlights: [
        {
            "from": 0,
            "to": 11.9,
            "color": "rgba(200, 50, 50, .75)"
        }
    ],
    colorPlate: '#fff',
    borderShadowWidth: 0,
    borders: false,
    needleType: 'arrow',
    needleWidth: 2,
    needleCircleSize: 7,
    needleCircleOuter: true,
    needleCircleInner: false,
    animationDuration: 1500,
    animationRule: 'linear'
}).draw();

// Function to get current readings on the webpage when it loads for the first time
function getReadings() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            var myObj = JSON.parse(this.responseText);
            console.log(myObj);    // For debugging
            var volt = myObj.input_voltage;
            console.log(volt);    // For debugging
            gaugeVolt.value = volt;
        }
    };
    xhttp.open("GET", "/readings", true);
    xhttp.send();
}

if (!!window.EventSource) {
    var source = new EventSource('/events');

    source.addEventListener('open', function(e) {
        console.log("Events Connected");
    }, false);

    source.addEventListener('error', function(e) {
        if (e.target.readyState != EventSource.OPEN) {
```

```
            console.log("Events Disconnected");
        }
    }, false);

    source.addEventListener('message', function(e) {
        console.log("message", e.data);
    }, false);

    source.addEventListener('new_readings', function(e) {
        var myObj = JSON.parse(e.data);
        console.log(myObj);    // For debugging
        gaugeVolt.value = myObj.input_voltage;
    }, false);
}
```

Listing 4: JavaScript Code

CSS code

```
html {
    font-family: Arial, Helvetica, sans-serif;
    display: inline-block;
    text-align: center;
  }
  h1 {
    font-size: 1.8rem;
    color: white;
  }
  p {
    font-size: 1.4rem;
  }
  .topnav {
    overflow: hidden;
    background-color: #0A1128;
  }
  body {
    margin: 0;
  }
  .content {
    padding: 5%;
  }
  .card-grid {
    max-width: 1200px;
    margin: 0 auto;
    display: grid;
    grid-gap: 2rem;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  }
  .card {
    background-color: white;
    box-shadow: 2px 2px 12px 1px rgba(140,140,140,.5);
  }
  .card-title {
    font-size: 1.2rem;
    font-weight: bold;
    color: #034078
  }
```

Listing 5: CSS Code

B Self Evaluation

Self-evaluation

Pre-competition

Prediction of Competition Results

How do you feel like the group has worked together to reach the final result:

The initial weeks were slow, and extensive time was spent deliberating different concepts and solutions for the battle bot without any considerable progression. This stagnation was a consequence of poor communication and not being able to agree on a concept for further development fast enough. Eventually, a breakthrough was achieved when the group settled on a concept and was divided into subgroups and subdivided tasks.

The cooperation within the group and subgroups improved greatly and progress was made fast. Based on the progression of the different subgroups, the members have helped each other across them when necessary. This enhanced overall efficiency and improved the workflow. Overall, despite the slow start and occasional lack of communication, the group worked together well to reach the final result.

What will the concept/strategy be during the competition:

The strategy during the competition is to have a stable and rigid battle bot with good maneuverability that can push or throw the opponent into an inoperable state. The strategy of beating opponents with dangerous weapons is to get them into the sand pit. Maneuvering in the sand pit is where the design of our robot hopefully will exceed the opponent's. The battle bot is expected to withstand impact from the opponents, and with the powerful motors and good grip it will hopefully be able to overcome the capabilities of the opponents' motors and knock them over or put them in an immobile state.

In summary, the strategy during the competition is not to physically destroy the opponents solely using a typical weapon, but rather through a combination of being more rigid and stable, having powerful motors, and utilizing the arena landscape to defeat the opponent.

How do you expect to lose:

The most probable way of losing is disconnection of any electrical components or if one of the outer components of the chassis breaks, resulting in the steering system being weakened, and in a worst-case scenario the battle bot will not be able to move. This is very problematic as the main strategy of beating opponents is through the movement capabilities of the battle bot.

Another way of losing would be to run out of energy, since the chosen concept relies on two big motors which consume more energy compared to the smaller motors chosen by other groups. If the belts get damaged and have a higher resistance than expected, the available energy from the battery might not be enough.

One issue that has been noticed during testing is that the iron in the sand is attracted to the magnets in the motors of the bot. This can become a disadvantage as the sand particles can enter the motor and interfere with the movement of the brush inside the motor which worst case can result in the bot not moving. This has been counteracted by taping the open holes of the motor. However, if the tape somehow wears off, or the particles enter through a tiny, overlooked entrance, this will still be a concern.

Due to the tracks being placed higher than the chassis, the bot can also drive upside down. Therefore, being flipped will not be a big problem for our strategy.

How do you expect to rank:

The group expects the battle bot to be competitive and to stand a chance of winning over opponents. The goal is certainly to win the entire tournament, but good efforts from the other groups will make this challenging. The group believes the battle bot stand a chance, but much can happen, and unforeseen scenarios can occur during the fights. Overall, the group expects to rank well, but precise ranking is difficult to anticipate due to not knowing what opponents the group will meet, and also the respective opponents' battle bot design and functions. The ranking will also depend on the order in which the opponents are faced. From observations in the workshop, some opponents might damage our bot, thus reducing the chances of beating weaker opponents in later rounds.

Since the different opponents have different strengths and weaknesses, it is impossible to counteract everything and also remain within the guidelines for a fighting bot. Against groups that chose a flipping device as a main weapon we see high chances of winning since our bot can drive in a flipped state and has enough mobility to exit the sandpits. If a battle bot has lower mobility than ours and we manage to push them into the sandpits some groups might not be able to easily come back out.

Against groups that are using a vertical/horizontal spinning device we might sustain damage which can reduce our mobility. Therefore, we must be careful and avoid their weapons as much as possible. We estimate that the teams using horizontal spinning devices with screws attached are less of a threat than groups using a pointy spinning weapon.

Since our tires and tracks are relatively exposed and the tires are printed with low infill to reduce weight, the biggest danger of those weapons is to damage the tires.

SWOT analysis:

	Positive	Negative

Internal	**Strengths**
	• **Mobility**
	  o **Steering + ability to drive in sand**
	  o **Can drive upside down**
	• **Low center of gravity**
	• **Adapts well to the arena**
		Weakness
		• **Energy consumption**
		• **MDF instead of aluminum**

External	**Opportunity** • **Opponents have weaker motors** • **Inspiration for future battle bot projects**	**Threat** • **Tracks getting damaged** • **Weapons on other bots are more destructive than expected** • **Regulations of competition limit capabilities**

C Chassis and drive wheel

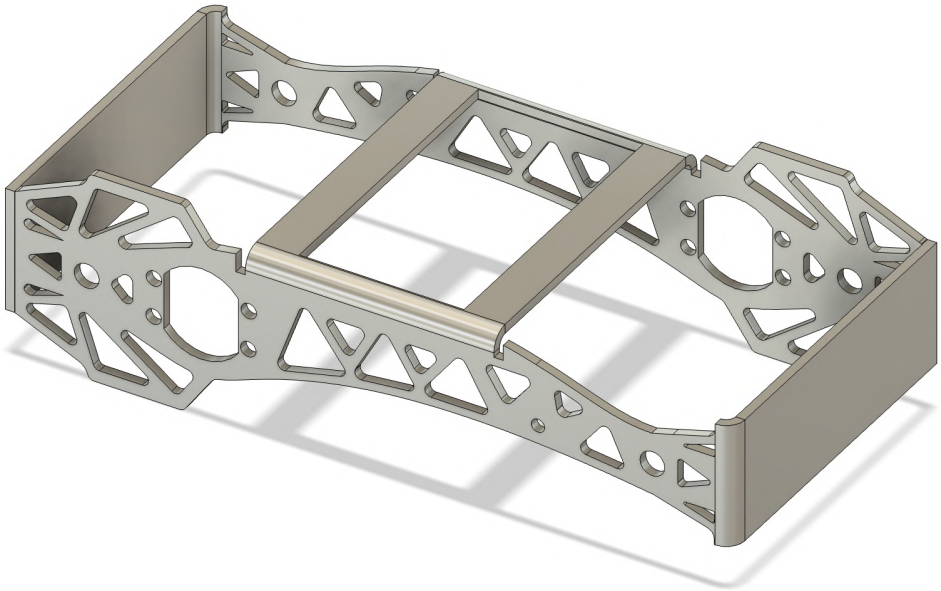


Figure 11: Final version of the chassis (V12).

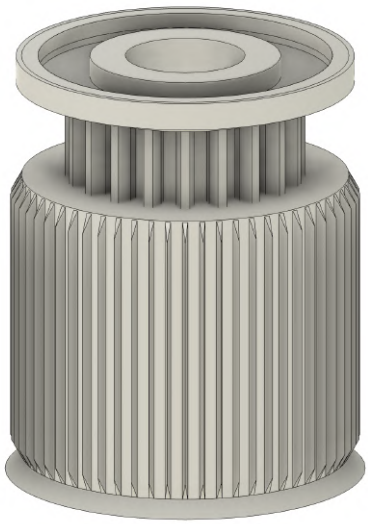
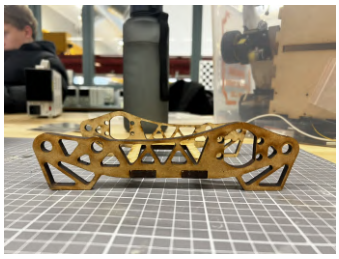
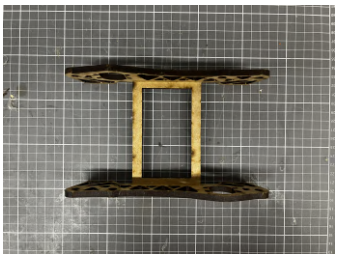


Figure 12: Drive wheel with integrated gear.



(a) MDF chassis picuture 1

(b) MDF chassis picuture 2



(c) MDF chassis picuture 2

Figure 13: Final product: Beltebassen

Figure 14: V12 produced

D Scrum Meetings

SCRUM MEETING WEEK 40

Group number:7

Referent:

Last week: Ideation

Name	Done since last meeting	Challenges	Discussion
Everyone	Discussed ideas and sketches	Finding clever solutions	Tank treads vs wheels for drive system
	Discussed drive system	Finding a robust solution	
	Weapon system (s)		
	Defense system (s)		

Next week: Prototyping a chassis and drive system

Name	What to do until next meeting	Challenges	Discussion
	Divide into smaller focus groups	What research to prioritize	Discussion energy storage and use of magnets
	Finding useful parts and components	Determining usefulness	What parts to use, which parts we need to find

Meeting review:

Name	What was good	What could be better
Sigurd, Marlon, Guiseppe, Brage	We found a plausible solution to chassis and defense system	Attendance
=	Brainstorming	Thinking more outside the box for other solution

SCRUM MEETING WEEK 41

Group number: 7

Referent: Boyan Yu

Last week:

Name	Done since last meeting	Challenges	Discussion
Boyan Brage Giuseppe Jonas Kevin Marlon Sigurd Simon	Attended laser cutting course. We split into groups, and decided what the different groups' tasks were. General research Concept brainstorm Made weight budget	Communication Meeting times Coordination	To solve the challenges, we decided to split into different groups. Meetings 10am on Fridays. Switched to Microsoft Teams for better file management.

Next week:

Name	What to do until next meeting	Challenges	Discussion
Boyan Giuseppe	Mechatronics, Arduino codes	BMS (Battery Management System)	Radio control, Wi-Fi streaming (Bonus point).
Brage Simon	Chassis / Magnet	Modular chassis	Passive or active down force system
Jonas Marlon	Drive Train Assembly	Gearing	One or two motors, differential system?
Kevin Sigurd	Steering system	Turning radius	Ackermann

Meeting review:

Name	What was good	What could be better
Boyan	Everyone was present.	
Brage	Everyone was participating the discussion	We could be more efficient
Giuseppe	There were a lot of ideas created	
Jonas	We created a team structure	There could be more communication attendance

Kevin	**More structured plan forward**	
Marlon	**A lot of good ideas**	
Sigurd	**We finally have a realistic idea that might work**	**Time spending**
Simon	**A lot of ideas were shared, and everyone was passionate about the concept.**	**Make sure we don't have to repeat our discussions during the meeting. Meeting structure**

SCRUM MEETING WEEK 42

Group number: 7
Referent: Kevin Velde
Last week:

Name	Done since last meeting	Challenges	Discussion
Kevin	Wheels and laser cutting, researched pneumatics, steering	Weapon options given space/weight limitations	Decided not to implement pneumatics
Brage	Made prototype chassis – tank threads, laser cutting	Quickly make chassis that is sufficient for testing	Implementation of magnets
Marlon	Made tank threads, prototype wheel, tested motor controller		
Boyan	Remote control code. Researched BMS implementation	Lack of remote controller	Sent mail to Håvard about controllers
Jonas	Motor controls, servo controls	Multiple chassis concepts	
Sigurd	Ackerman steering 3D model		
Giuseppe			
Simon	Made/laser cut prototype chassis – wheels. Researched pneumatic weapon option		Decided not to implement pneumatics

Next week:

Name	What to do until next meeting	Challenges	Discussion
Kevin	Assembly and testing, revision, weapon		
Brage	Testing and iteration of chassis		
Marlon	Make tank threads		
Boyan	Test controller		
Jonas	Drive train		
Sigurd	Test steering		
Giuseppe			
Simon	Improve CAD		

Meeting review:

Name	What was good	What could be better
Kevin		
Brage	We agree on plan forward	Communication outside of meetings
Marlon		
Boyan	People expressed opinions	
Jonas		
Sigurd		
Giuseppe		
Simon		

SCRUM MEETING WEEK 43

Group number: 7

Referent: Giuseppe D'Auria

Last week:

Name	Done since last meeting	Challenges	Discussion
Kevin	Assembly and testing, revision, weapon		
Brage	Testing and iteration of chassis and side skirts, improved CAD		
Marlon	Make tank threads, worked on mechatronics	Arduino's connections	
Boyan	Test controller		We got the controller. Changed from Adafruit to Arduino nano.
Jonas	Drive train		
Sigurd	Test steering, improved the belt and completed the weels. CAD		
Giuseppe	Research on amplifiers		It would be too complicate to use amplifiers
Simon	CAD (motor mounts), contributed to other members tasks	3d printers	

Next week:

Name	What to do until next meeting	Challenges	Discussion
Kevin			Contribute during tests
Brage	Test tank	Improve chassis and side skirts	
Marlon	Testing motor control and run time		
Boyan	Arduino coding and help with motor control	troubleshoot Arduino IDE	
Jonas			Gone for personal reasons
Sigurd	Make and test belts		

Giuseppe	CAD plow		
Simon	CAD internal mounts and test motor mounts		Contribute to other tests aswell

Meeting review:

Name	What was good	What could be better
Kevin		
Brage	Efficient	
Marlon	Efficient	Better communication outside the meetings
Boyan	We impressed the other groups (in a good way)	
Jonas		
Sigurd	Efficient	
Giuseppe	Good team work	
Simon	Team had a good communication during the week	

SCRUM MEETING WEEK 44

Group number:7

Referent: Simon

Last week:

Name	Done since last meeting	Challenges	Discussion
Kevin	Lazer cut Assembled the battle bot		Made drive wheels but the design got scraped
Brage	Tested the bot Designed chassis iterations Lazer cut multiple prototypes	Redesigned chassie to improve agility of the bot	
Marlon **Jonas**	Testing motor control	Did not get to test the run time due to bad motor control	Custom made an high power h bridge
Boyan	Arduino coding and help with motor control Also worked on the bsm	bms	
Sigurd	Tested the battle bot Redesigned iterations of the drive wheels Made belt	Errors during testing that have caused the team to redesign	
Giuseppe	CAD plow 3d Printed test chassie		Plow may get cancled due to weight budgeting
Simon	Printed wheels Assembled and tested the battlebot Lazer cut a prototype		Designed a konsept for the chassis that got cancled

Next week:

Name	What to do until next meeting	Challenges	Discussion
Kevin	Test battlebot		
Brage	Test chassie,		improve weight
Marlon **Jonas**	Complete testing of motor control and run time		

Boyan	Work on BMS and streeming		
Sigurd	Test the battlebot Design setup for weapon solution		
Giuseppe	Work on the plow if it gets implemented	Need to know the weight of the prototype	
Simon	Test battlebot		

Meeting review:

Name	**What was good**	**What could be better**
Kevin	**Good plan forward, good communication**	
Brage	**Efficient and good colaboration**	
Marlon	**Accomplished a lot**	
Boyan	**Good workflow**	
Sigurd	**Great work**	
Giuseppe		
Simon	**Got to do everything we planned to do**	
Jonas	**Good communication**	

SCRUM MEETING WEEK 45

Group number: 7

Referent: Boyan Yu

Last week:

Name	Done since last meeting	Challenges	Discussion
Boyan	Fixed BMS, Researched and attempted on streaming.	Connecting ESP32 to eduroam, connect ESP32 to Arduino. Finding working MicroUSB cable.	Streaming is difficult.
Giuseppe	Started writing report		
Brage	Made three new chassis iterations. A lot of testing.	To make good gearing system in compact chassis.	Discussed about different motor solutions, and how the gearing should work.
Jonas Marlon	Constructed H-bridge with exchangeable components for easier testing. Write code to control with remote.	Power surges. Gate input timing.	Found out that stalling current was the culprit.
Kevin	Cutting and 3d-printing. Modification of components.		
Sigurd	Fixed hardware (belts etc.) A lot of testing.	Lot of rolling resistance in belts	We switched from directly powering the wheels to powering through gears.
Simon	Troubleshooting		

Next week:

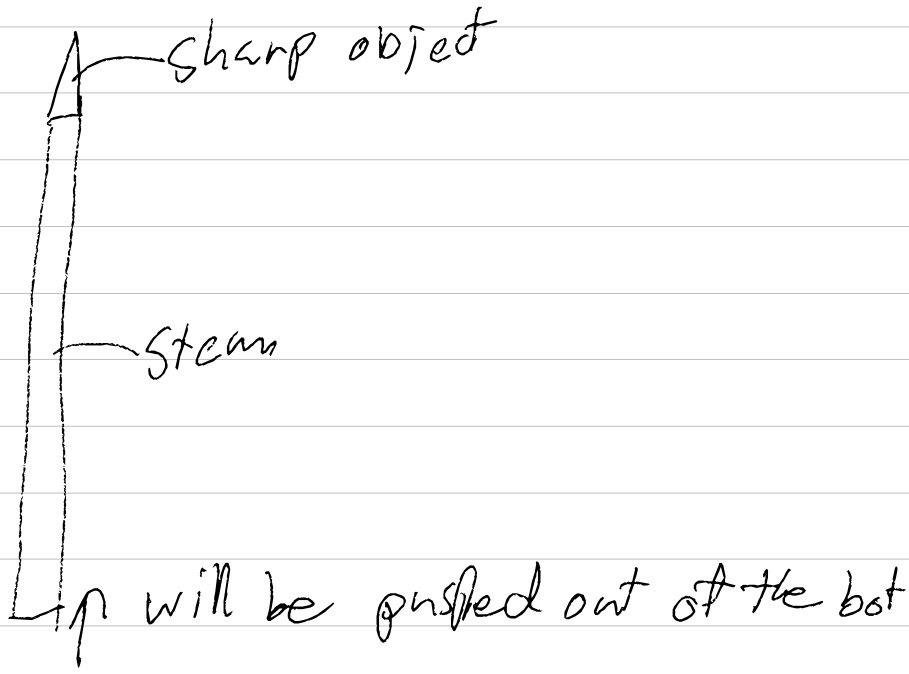
Name	What to do until next meeting	Challenges	Discussion
Boyan	Streaming	Streaming	Steaming is difficult
Giuseppe	Report		
Brage	Testing		
Jonas	Testing		

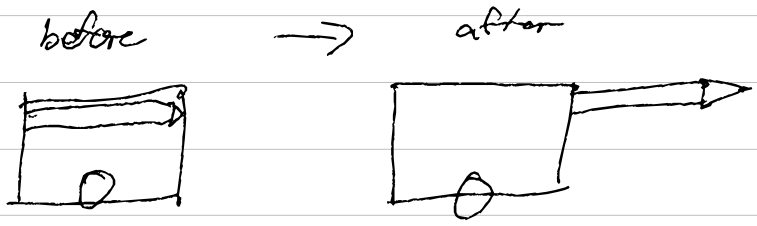
Marlon	**Testing**		
Kevin	**Testing**		
Sigurd	**Testing**		
Simon	**Testing**		

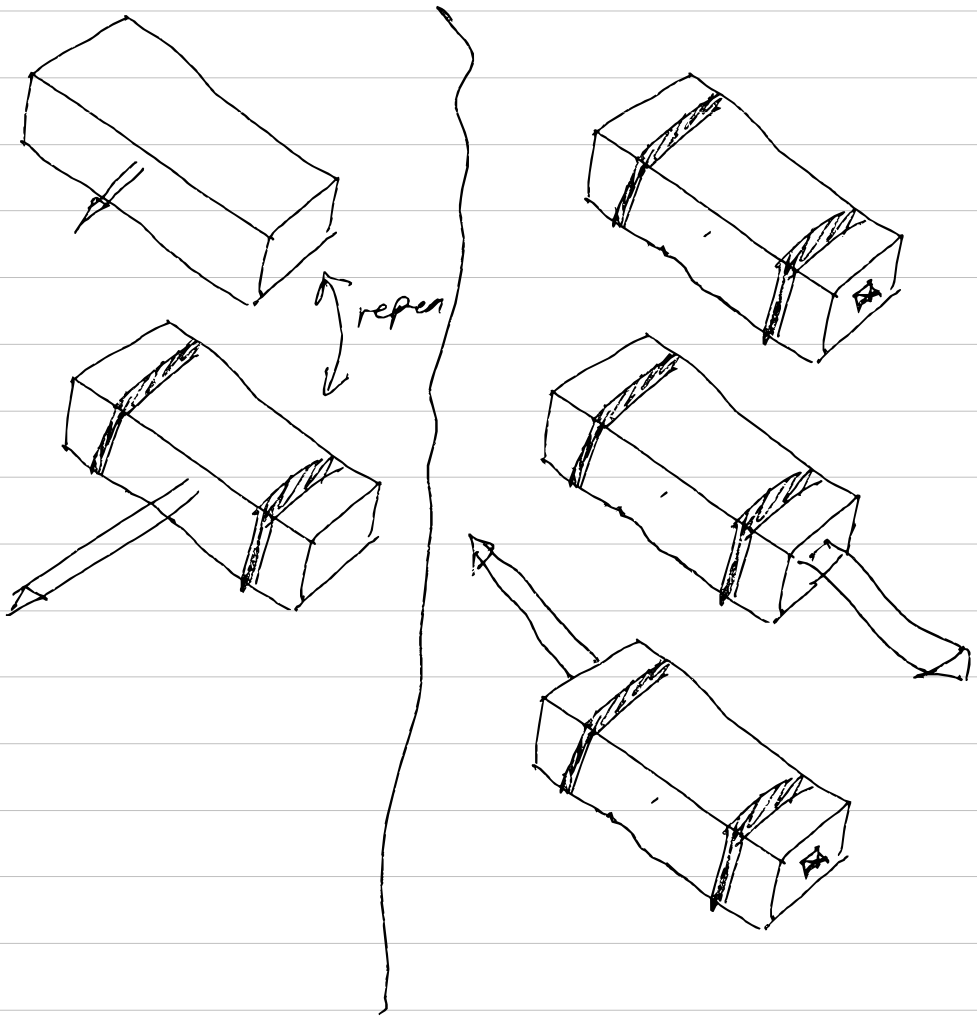
Meeting review:

Name	What was good	What could be better
Boyan **Giuseppe** **Brage** **Jonas** **Marlon** **Kevin** **Sigurd** **Jonas**	**Good workflow. Didn't interrupt working. Everyone knew what they were/should be doing**	**Not a lot of engagement in the meeting as we were focused on the work on our hands.**

E Brainstorm Consepts



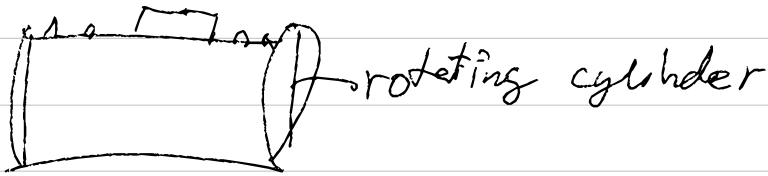


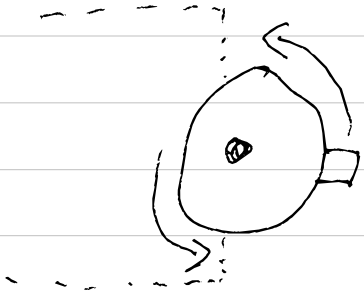


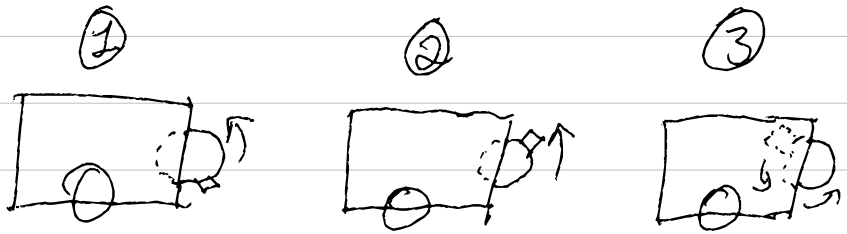
charge a motor during battle. and use it as a last chance

Pneumatics. (predecided comp.)

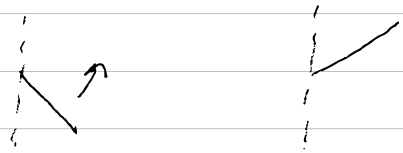
70W mini compressor

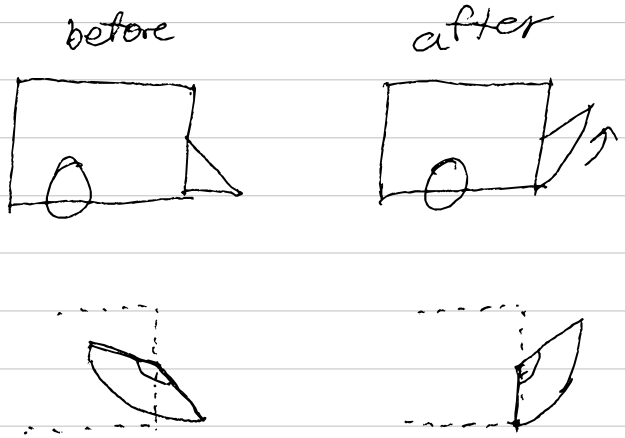

rotating cylinder

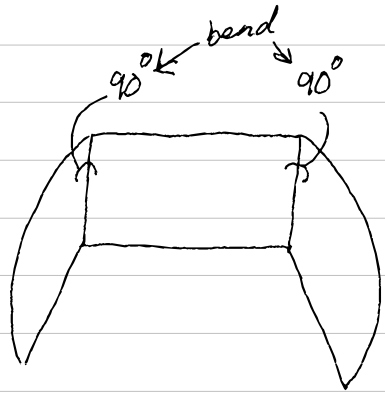



①
②
③

flicker

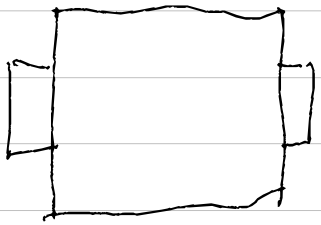


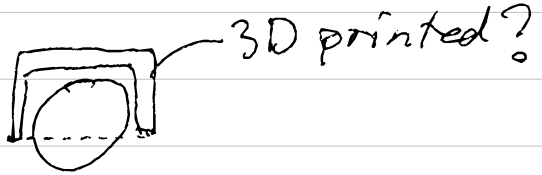


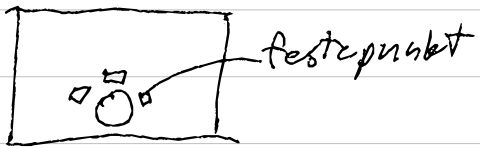


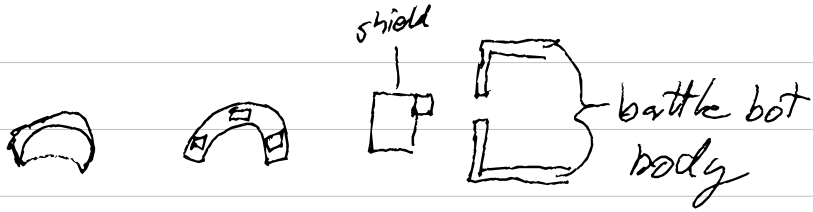
Pneumatics

Shield over wheels

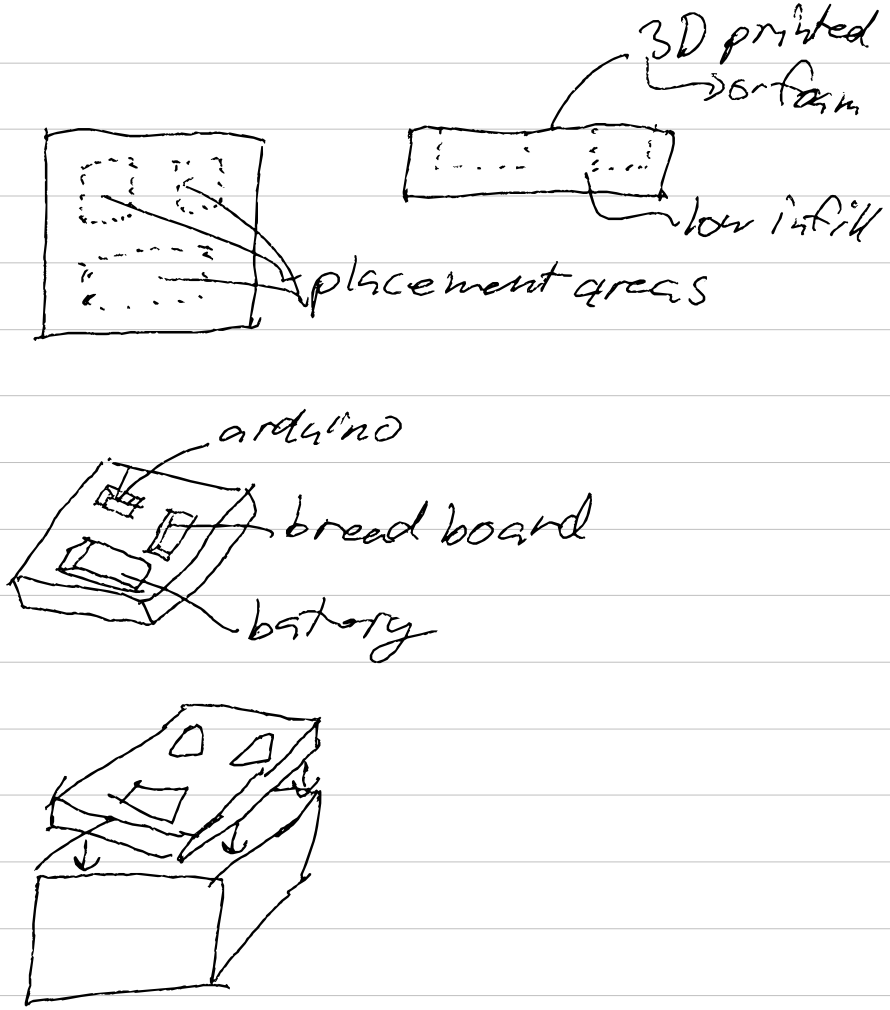




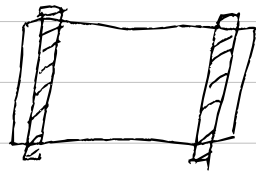


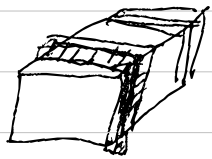


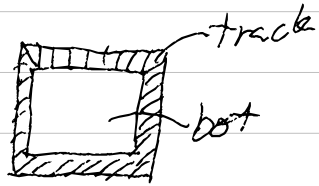
inside the bot



tracks instead of wheels





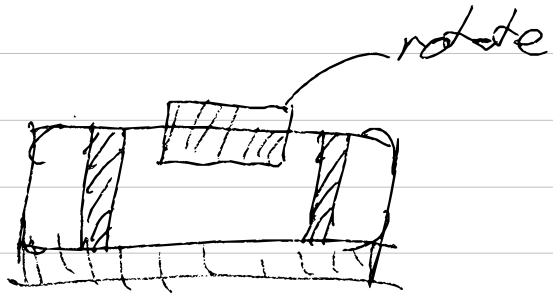


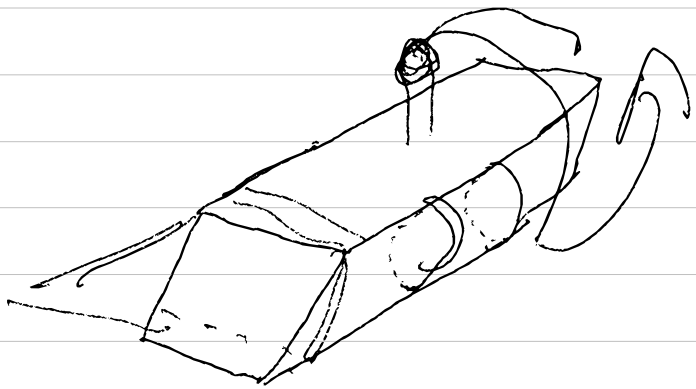
if track is light & cheap

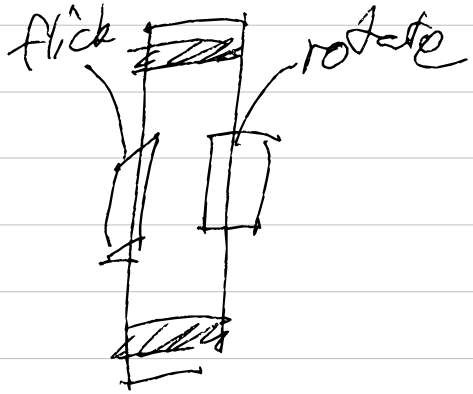
- advantage on sand
- create our own gears

Wheels

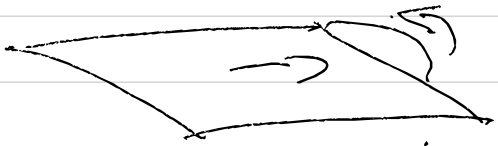
ackerman steering.


rotate

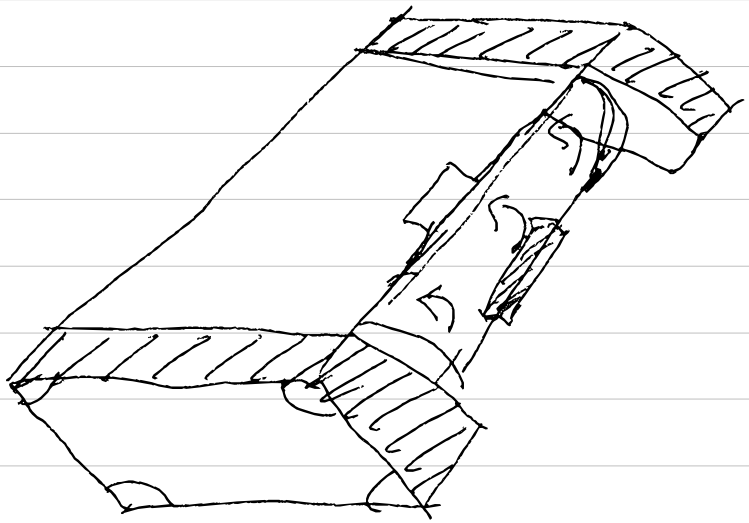



flick
rotate

Bot shape

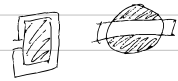


side view



Bot Ideas

movement



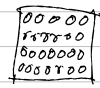
- belts
  - ↳ wheels

Weapons

- Cylinder w/rubber
- spinning saw blade
- plow (flick)
- pool que (push)
- spinning chains
- fork w/ net

Defense

- wheel on top & bottom
- hole on the metal walls



- low torque motors
- warm gear
- mini vacume mot

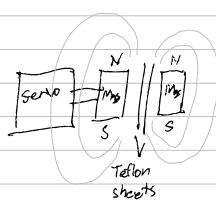
Battery

Buck converter

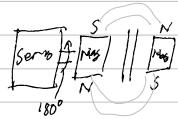
Joule thief

Magnet

- Diametrically magnetized Magnets

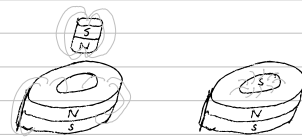


this has to be shielded by a 3D printed design

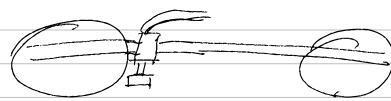


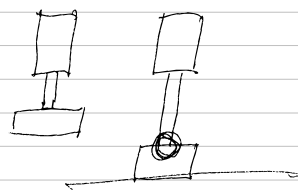
- How do we stick the magnet to the servo?

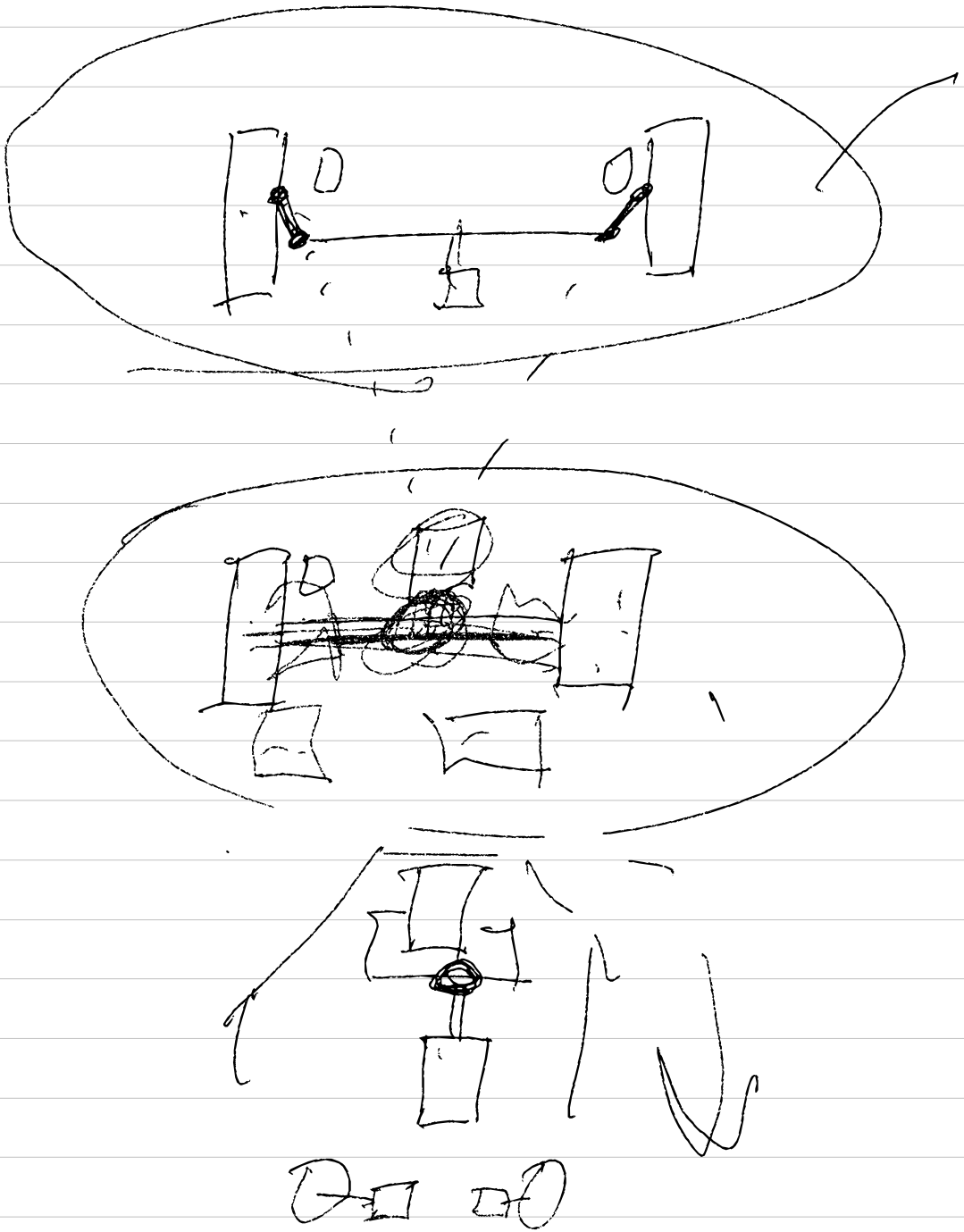
- Maybe we can use an axially magnetized ring magnet and insert a disc magnet to turn it off

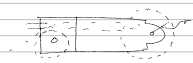


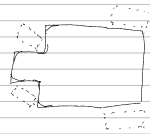
- Put the magnet on the same level as the wheels with a Solenoid

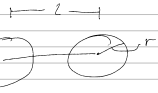




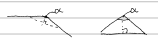




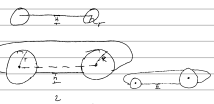


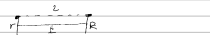


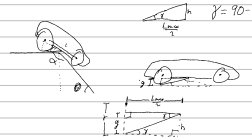
r to L ratio



$$h = \cos\left(\frac{180-\theta}{2}\right)$$







L is max when g = h

$$\tan\gamma = \frac{g}{\frac{L_{max}}{2}} \Rightarrow \tan\gamma = \frac{2g}{L_{max}}$$

$$L_{max} = \frac{2g}{\tan\gamma}$$

g = gap between bottom plate and floor.

γ = angle from center of bottom plate to wheel-floor contact point

L_{max} = max distance between the front and rear wheels

$$h = \cos\left(\frac{180-\theta}{2}\right)$$

θ = angle of elevation from sand pit to tiger area

h = minimum gap between bottom plate and floor