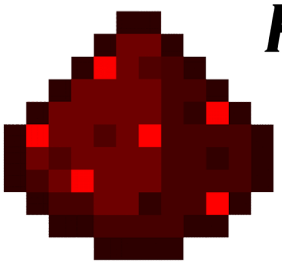
2020/2021

Engineering Notebook



POWERED BY Redstone

of the team

#17089

Open Gate School

SPECIAL THANKS TO THE PEOPLE THAT ARE HELPING US MAKE *POWERED BY REDSTONE* A REALITY!

ESET®

TABLE OF CONTENTS

1. INTRODUCTION

1.1. ABOUT US

1.1.1. TEAM MEMBERS

TEAM LEADERS

BERTRAM ŠKAŘUPA

I have been programming since the age of 14, but I found my true love for it when I started programming the robot and it was beautiful to see my code execute something physically!

PETR DOBIÁŠ

Even since I was a kid, I enjoyed being creative. I used to play with all kinds of construction kits and make contraptions. I think these experiences translate into what we do in robotics.

ENGINEERING

PETR KUŽELA

Since my childhood, I enjoyed building and tinkering with LEGO. The creative and constructive skills have nicely transferred into designing robots for FTC. I hope to continue my creative development into my future profession.

JOSEF SCHNEIDER

Since I was a kid I was interested in building, mostly Lego, but I always wanted to try something more challenging. For me personally helping with building a new robot is very relaxing and also helpful with my fine motor skills.

MILI MACKOVÁ

I really admire my team members who are very passionate about what they do, as they lead the team and create an amazing and enthusiastic atmosphere during each meeting.

ŠIMON MIREK JAKUB MRÁZ

I first thought that joining the team would be just something to fill my free time. Yet as soon as we got into working on the robot. I realized it fills me with joy to be near the robot and to see how we progress with the work.

OLIVER ŠKAŘUPA

This year, I joined the engineering team to learn how to create 3D models in the program *Fusion 360*. One of the biggest opportunities of the Robotics team is collaborating with many team members on a single large project.

PROGRAMMING

JÁCHYM DOLEŽAL

I have always loved creating and building stuff. I am a creative person who loves learning new things. I joined this team at its early beginning.

MARTIN LIBOR MATURA

Ever since I was a kid, I have always been fascinated with technology. That is why I immediately jumped at the opportunity to join the robotics team, so that I can pursue and express my passion.

SÁRA NOVÁKOVÁ

I have always admired the robotics team and the amazing things they work on together. Therefore, I decided to try something new this year and join the team myself, so that I can learn some new skills.

SOFIE POPOVA

I only started with programming recently and I was looking for an opportunity for developing the freshly gained skills, my love for problem-solving and sharing this passion with other people in the field of my interest. When I first heard of the school robotics club, it seemed like a brilliant chance for me.

POWERED BY REDSTONE

Team *Powered by Redstone* is a Czech FTC team that consists of members studying at Open Gate School in grade ten, eleven and twelve Open Gate has been mainly a language school in the past, but it has decided to promote STEM fields during the recent years. Our robotics team serves as one of the ways to achieve this goal. As a part of this project, our team is extensively supported and funded by our school and we were given our own academic club and a full workshop.

Open Gate is a fairly small school, as it has only about three hundred students. Even so, we have over 10 active members with occasional help from others. All of our members are passionate about robotics in general, and many of us would like to pursue the goal of working in the field of engineering. As a result, even though our school schedules are not ideal, we are still meeting twice a week in the late afternoon to actively work on the robot. Furthermore, our members are willing to sacrifice their free time in order to work on our robot during the week or even during weekends. We are taking part in FTC for the third year with previously participating in two tournaments hosted in Barcelona. This was our very first experience, from which we also much learned. We hope that all of our effort and work will pay off in this year's challenge.

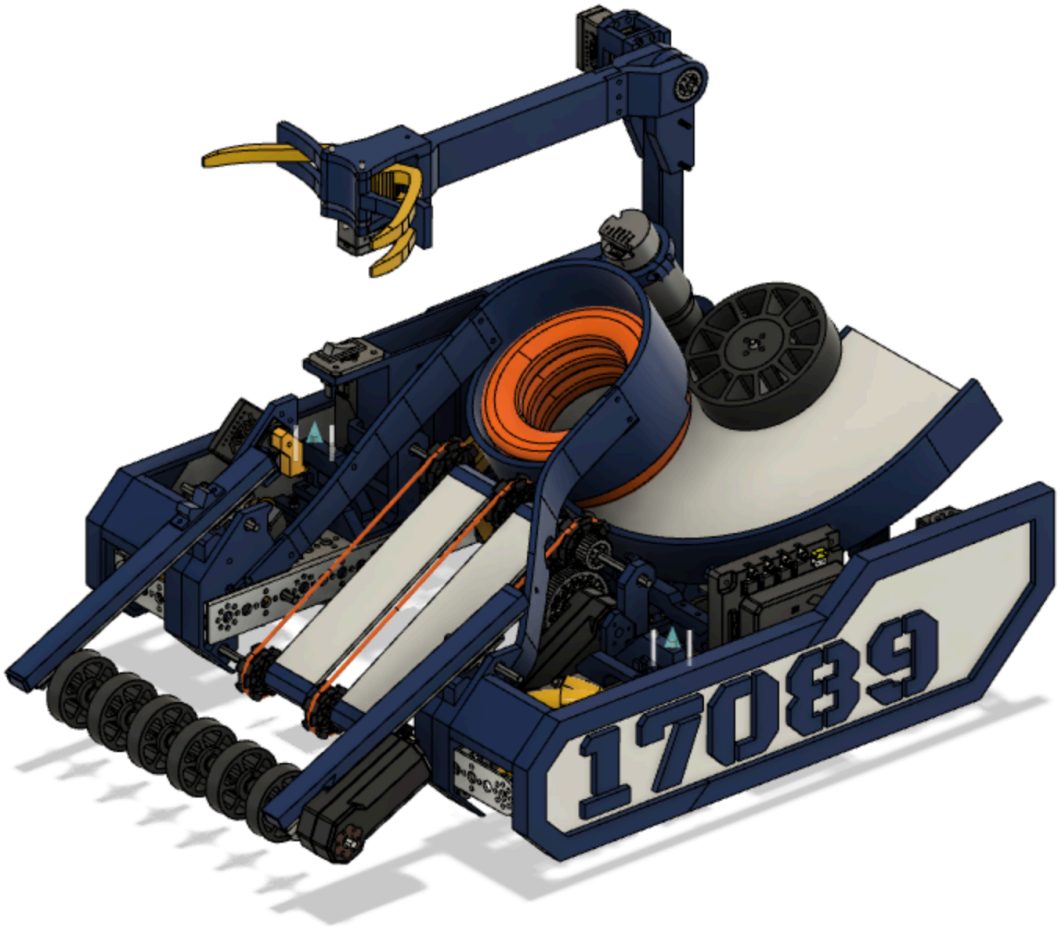
For season 2020/2021, we unfortunately spent the vast majority of the season in lockdown, meaning we couldn't work on in person and test our robot only until two weeks before the online competition organized in Spain. Even so, we managed to produce possibly the best robot to date.

2. ENGINEERING

2.1. INTRO

This season, for the first time, we have had access to a 3D printer from the beginning of the designing process. This allowed us to redirect our focus from figuring out the methods to utilise stock components and freed us to think outside of the box to develop custom components. This possibility proved to be essential, when the COVID-19 lockdown started. Being schooled from home, we did not have access to our workshop. However, we continued to design our robot in Autodesk's Fusion 360. Having the possibility to custom-design and 3D print most of our rigid components, we had a very comfortable transition from in-person sessions to online sessions.

The general vision was to make the custom components angular wherever possible.



2.2. CHASSIS

This year we chose to make use of the similar chassis frame as last year - a simple H shape made of standard tetrix shafts. The only difference from last year's structure is that we use one shaft across the frame instead of two (as seen in figure 2.1.).

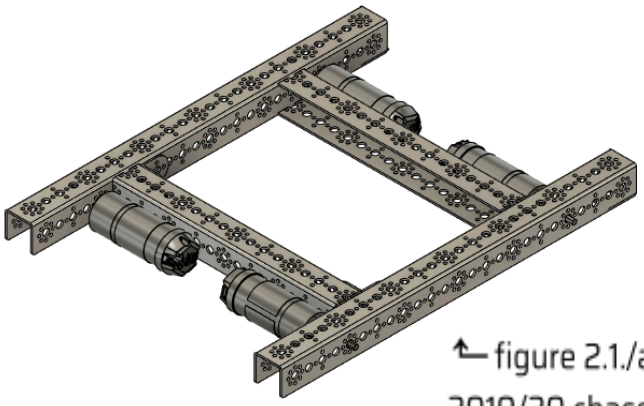


figure 2.1./a
2019/20 chassis design

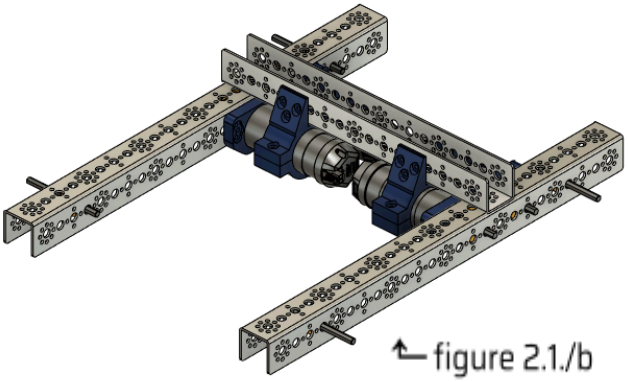


figure 2.1./b
2020/21 chassis design

The main purpose of this change is to allow enough space for a ramp that transports rings towards the launcher.

Overall, this chassis design was chosen because we wanted it to have the following characteristics:

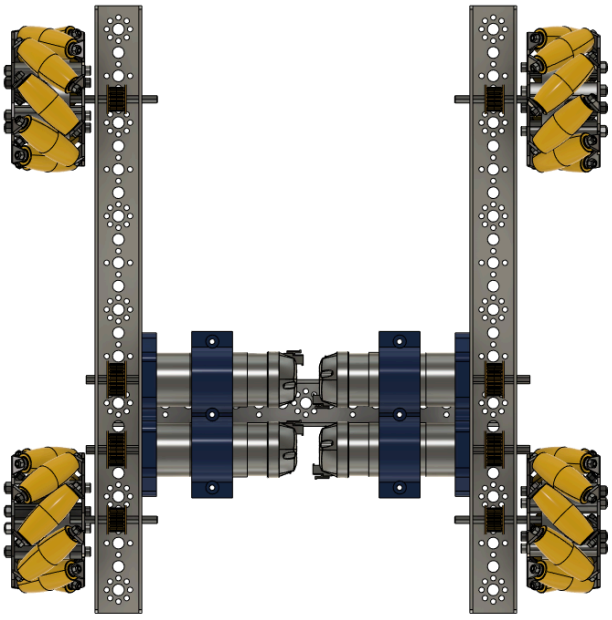
- Rigidity - it needs to be able to bear the stresses that the weight of the launcher presents
- Drive motor placement - we needed the motors to be out of the way so that we can fit a ramp that transports rings to the front part of the robot.
- Wiring friendly - the hollow structure of the tetrix shafts allows cables to fit inside it. This prevents the wiring from disturbing the visual appearance and functionality of the robot

2.3. POWERTRAIN

This year, we wanted two improvements for our robot form last year.

1. Faster robot - more speed
2. All motors to be in one location

The two requirements were solved this year by transferring the power from the motor to the wheels using belts. We wanted to increase the RPM of the wheels, so we modeled custom 3D printed pulleys. The ratio of the gears is 3:2. We calculated the length of the gears and ordered the belts online. Because we used U-shaped tetrix extrusions, we designed the belt to fit inside and not limit the functionality of any other components. Because we had a lot of space opened up in the middle of the robot, we had no problems fitting the transport mechanism (see more in section 2.5).



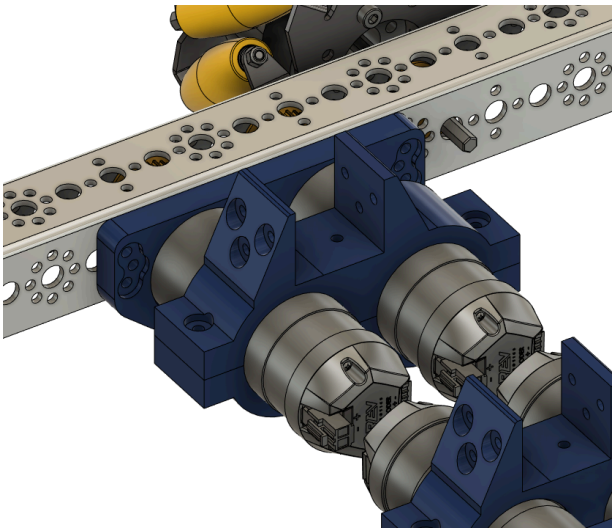


figure 2.3./a

figure 2.3./b

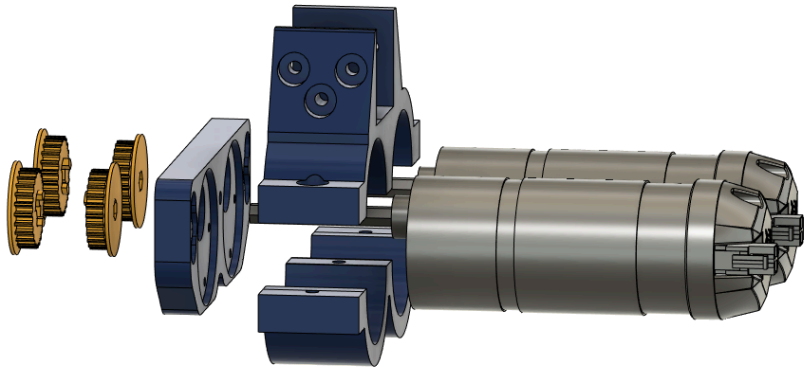


figure 2.3./c

In figure 2.3./c is visualized how the motor is mounted with custom 3D printed parts that are connected to the chassis. Figures 2.3./d and 2.3./e are photographs that show the mounting of our belt system.

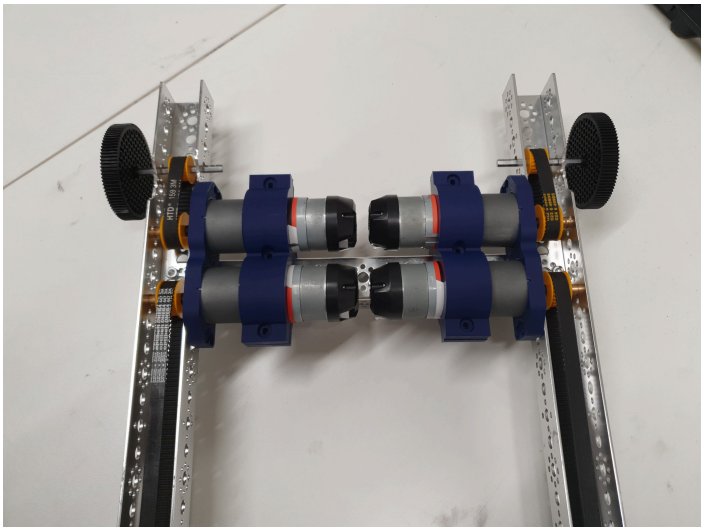


figure 2.3./d

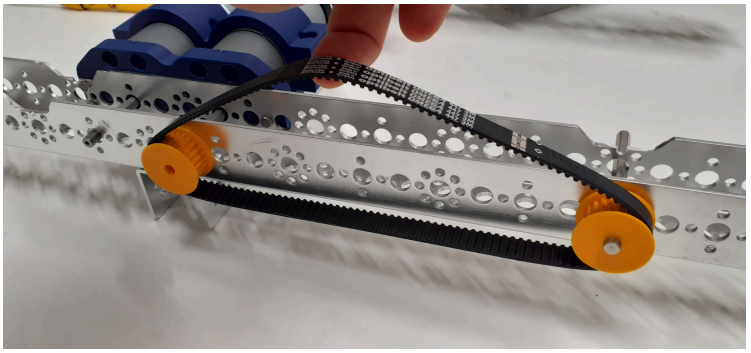


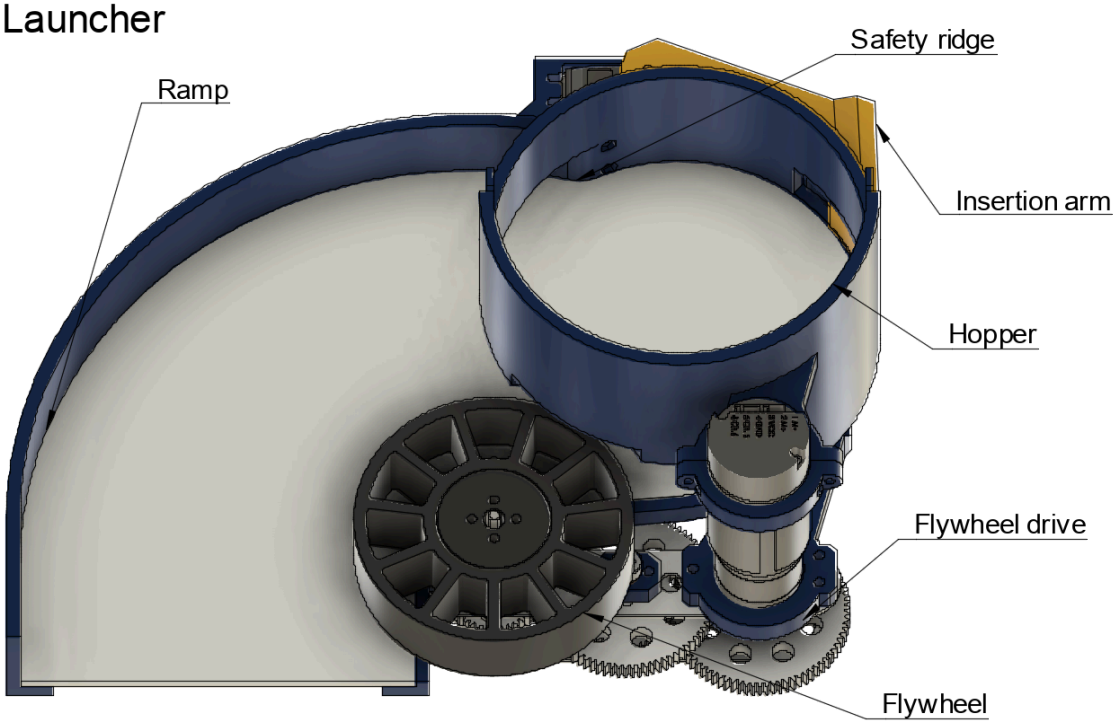
figure 2.3./e

2.4. LAUNCHER

Being the most essential part of this year's challenge, the launcher was naturally the primary focus of the robot's design. The first requirement our team unanimously agreed upon was a single flywheel that would propel the ring and give it rotational inertia which would increase its stability and accuracy. The next step was to figure out the loading of rings into the shooting position. In the end, we decided to load the rings into a gravity-based magazine before launching them (see figure 2.4./a).

This would give us the possibility to load all three rings into position at the start without the inconvenience of stacking them onto the transport mechanism.

Figure 2.4./a
Launcher



From the hopper, the rings would take a circular path in contact with the flywheel, while being in a slight compression against a guardrail for more contact area with the flywheel. At the bottom of the hopper, there is a small servo-operated arm that pushes the bottom-most ring into the flywheel, which causes it to be launched. The bottom of the launcher is

made of acrylic, while the body is made out of 3D printed PLA. When the rings are in the hopper, they are secured by a protruding ridge that is on the boundary of the hopper and the acceleration platform. This prevents the rings from accidentally slipping into the spinning flywheel (see figure 2.4./a).

The rings are meant to be launched strictly by the trigger servo. The entire launcher is angled at 30° relative to the ground. From 3D model inspections, with a straight line trajectory, the ring would reach the top of the goal when launched at this angle from the half-way field divider. When accounting for gravity, we estimate that the ring is going to go into the upper goal. This estimation confirmed itself in testing. The entire launcher is mounted at four points using custom 3D printed braces.

However, the motor itself does not have high enough RPM to eject the ring into the goal. Our solution was to create a gearbox to increase the RPM of the flywheel. We mounted the motor to the launcher body and used metal gears mounted in custom 3d printed parts to increase the rpm of the launcher wheel by four times.

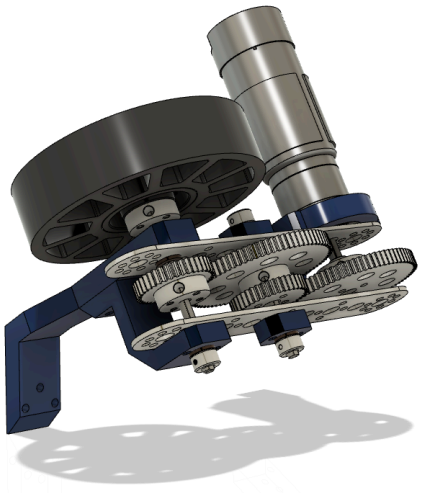


figure 2.4./b

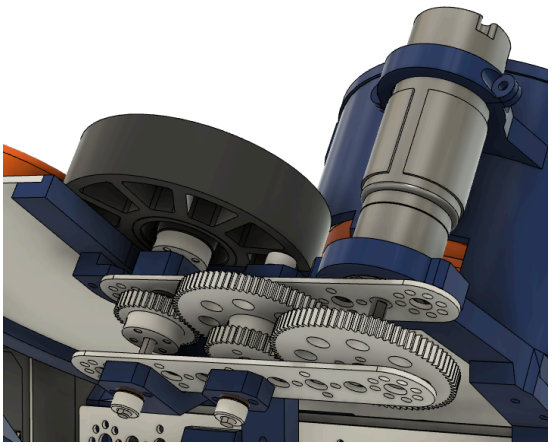


figure 2.4./c

This solution worked perfectly. The launcher wheel now spins about 5000 RPM and the trajectory the wheels are ejected matches with our calculations.

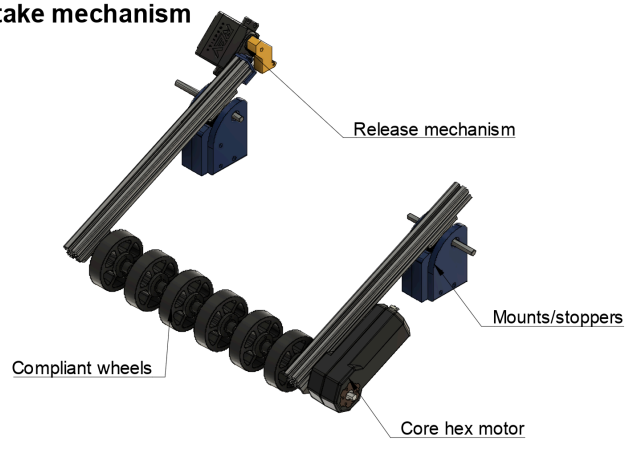
2.5. TRANSPORT MECHANISM

To shoot the ring, the robot first has to pick it up. Facing this challenge, we designed a transportation mechanism and an arm that loads the rings into the launcher. The first task was to pick the ring up off the ground. Our solution is two wheels that spin in opposite directions. These wheels will pull the ring onto a transport belt that will funnel the rings into the launcher.

The concept of the intake was clear from the beginning: we wanted to have the intake at the rear of the robot, with a transport belt that would deliver the rings into the hopper of the launcher. The belt had to be sloped at a reasonably shallow angle, to avoid problems with pulling the rings up. To get the rings onto the belt, we wanted to have a shaft with an opposite spin to the belt, which would rest outside of the robot. Our thought process was that the intake can be put into place and does not have to be removed for the rest of the match. Therefore, the entire arm that will be lowered does not have to be motorized. The idea was that the arm would crash down into place, hitting a solid piece of plastic that would transfer the shock into the chassis of the robot. The entire engagement of the arm was to be forced by gravity. The initial locking and release is ensured by a servo with a small wedge, that prevents the arm from falling preemptively (the initial prototype can be seen in figure 2.5./a).

Figure 2.5./a
Intake mechanism

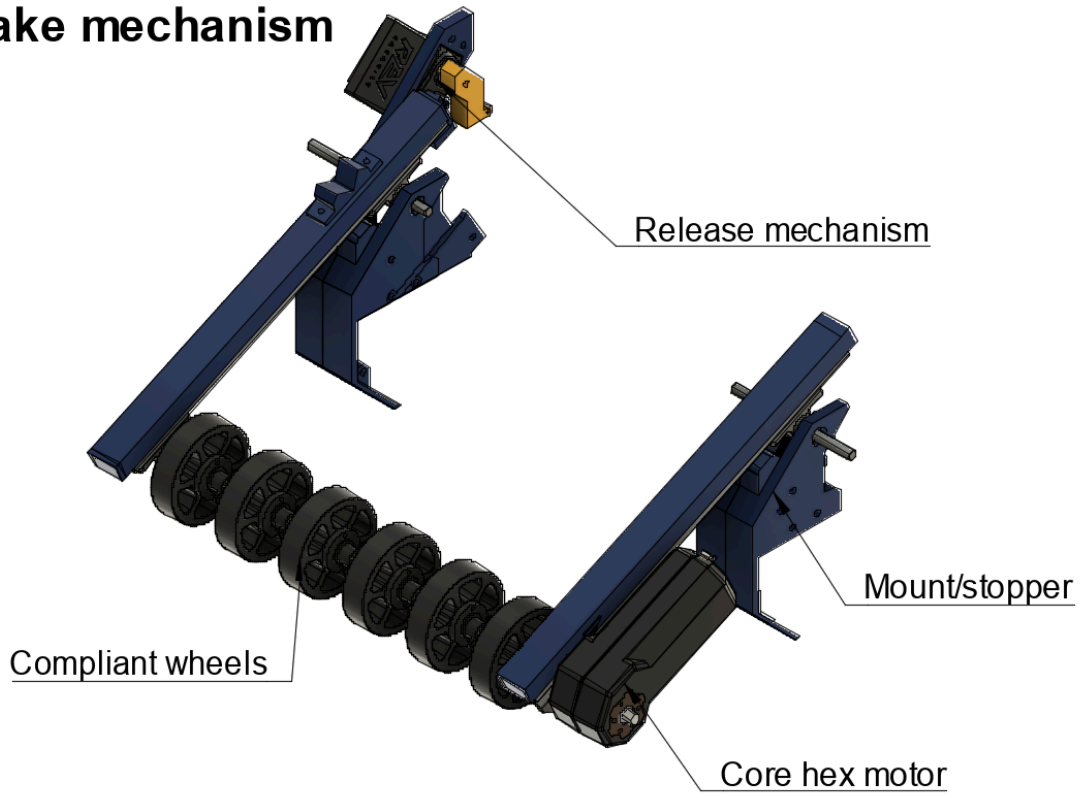
Initial prototype



The arm has a shaft with compliant wheels and a core hex motor attached at the end. These compliant wheels will be lifted approximately 1 cm off the ground, so that they have efficient contact with the ring on the floor. Aside from being compliant themselves, the arm is not resisted by anything other than gravity. Therefore, the ring intake will not force the robot off the ground.

In the final version, we made the pieces more aesthetically pleasing and we added other functionality to the mounts (see figure 2.5/b).

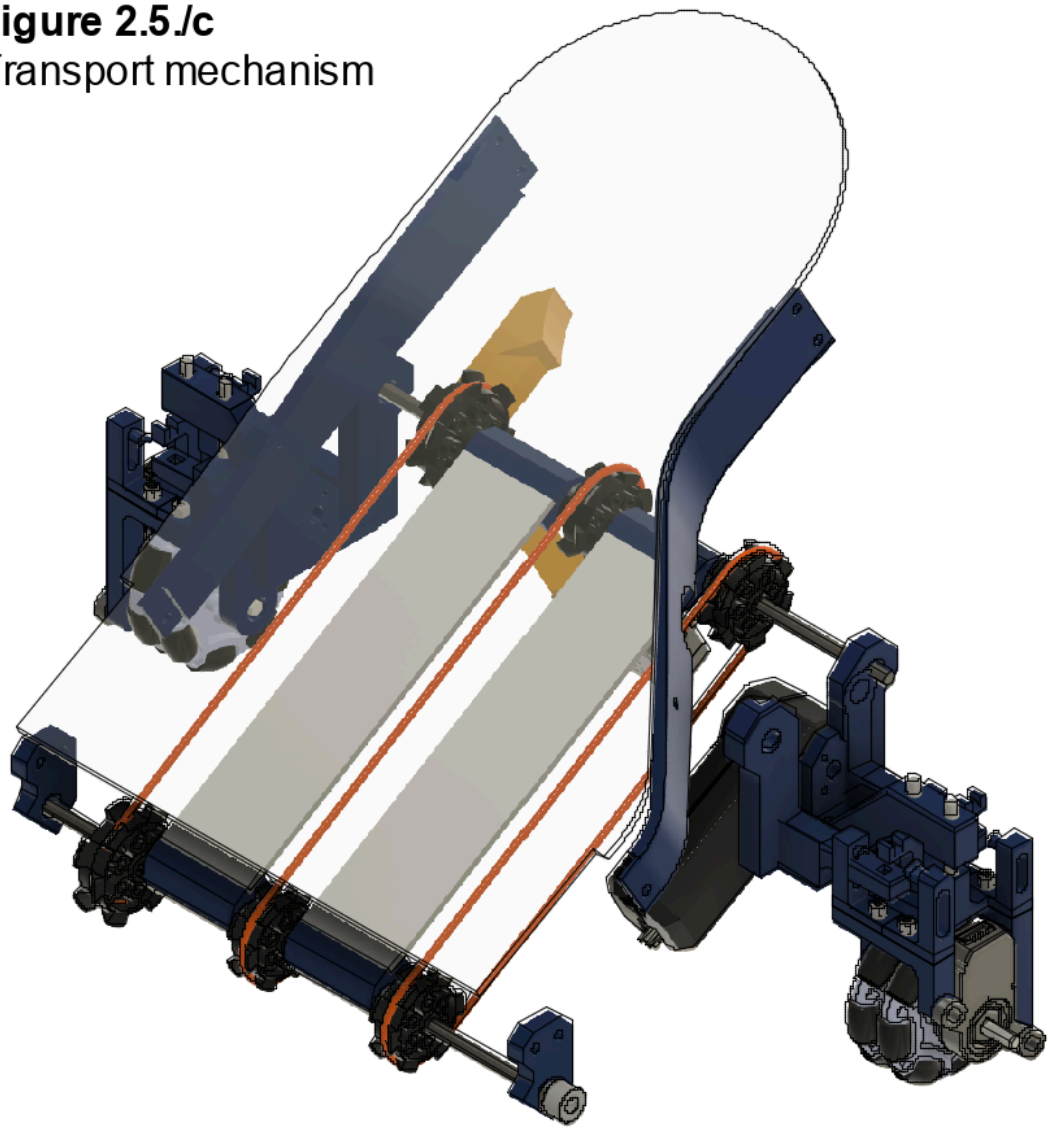
Figure 2.5./b Final version
Intake mechanism



We had slight issues with our transportation belts. The initial concept counted on polyurethane belts, which immediately proved to have low friction, so we had to change them out for rubber wires. The side plates of the transport mechanism are designed to funnel the rings into the hopper==. Since the odometry components were in a convenient place, we joined up the parts and made the odometry holders function as upper

transport belt holders. The transport belt is covered up by an acrylic plate to ensure adhesion of the ring to the belt.

Figure 2.5./c
Transport mechanism



2.6. ODOMETRY

In order to know the precise position of the robot on the playing field, we needed to implement odometry encoders. The solution was placing three encoders reading data from their independent wheels.

To increase data accuracy, we created a custom 3D printed odometry design with a seperate suspension system. Each encoder is set up to be pushed into the ground by rubber bands in order to maintain constant contact with the ground, even when driving over irregular terrain. Our first design is outlined in *figure 2.6/a*. Two axles are mounted to the robot's chassis using a simple part. The odometry encapsule is then slid on the two axles and attached using rubber bands. The rubber bands pressure the odometry wheels into the ground and therefore maintain contact with the ground even when driving over irregular terrain. With this solution, the code can fully rely on position data from our odometry encoders, and use that for advanced control of the robot's movement.

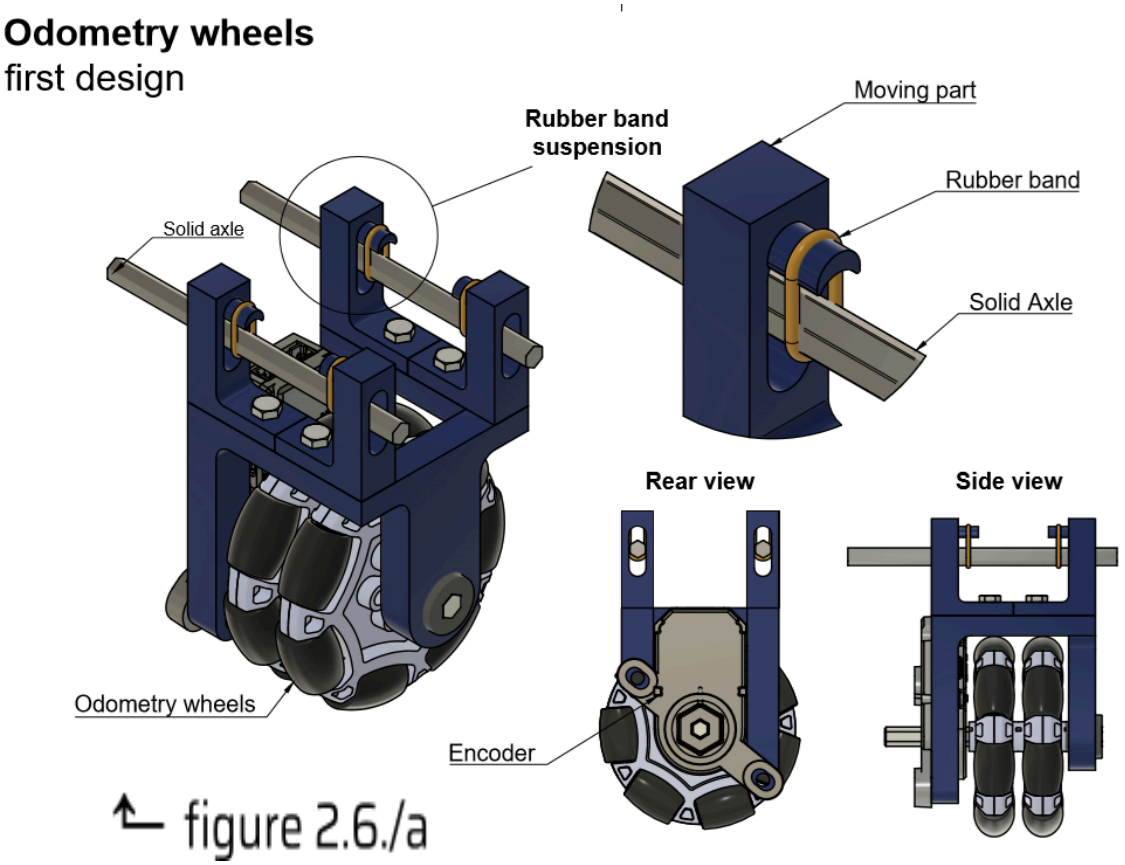
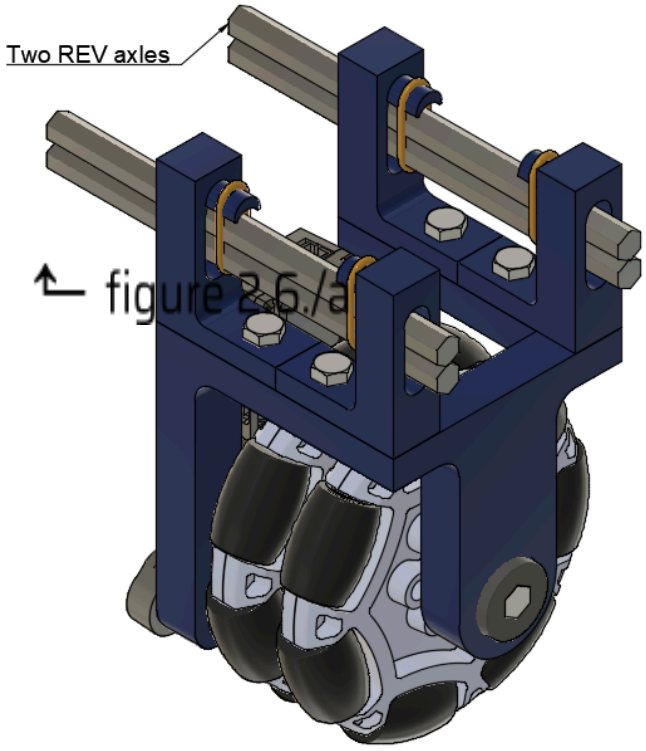


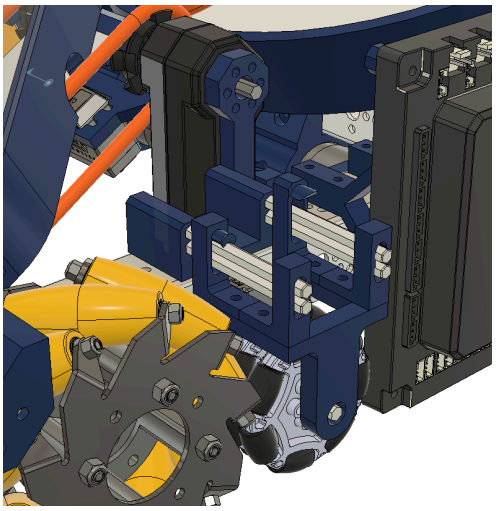
figure 2.6./a

However, this solution did not work as well as we had hoped. There were many problems. The first big issue was with the odometry wheels moving sideways because the hole for the axles was too big. If we made the hole smaller, the axles wouldn't be able to slide easily. This turned out to be a major problem for this design.

Odometry wheels
double axle modification



↑ figure 2.6./a



↑ figure 2.6./b

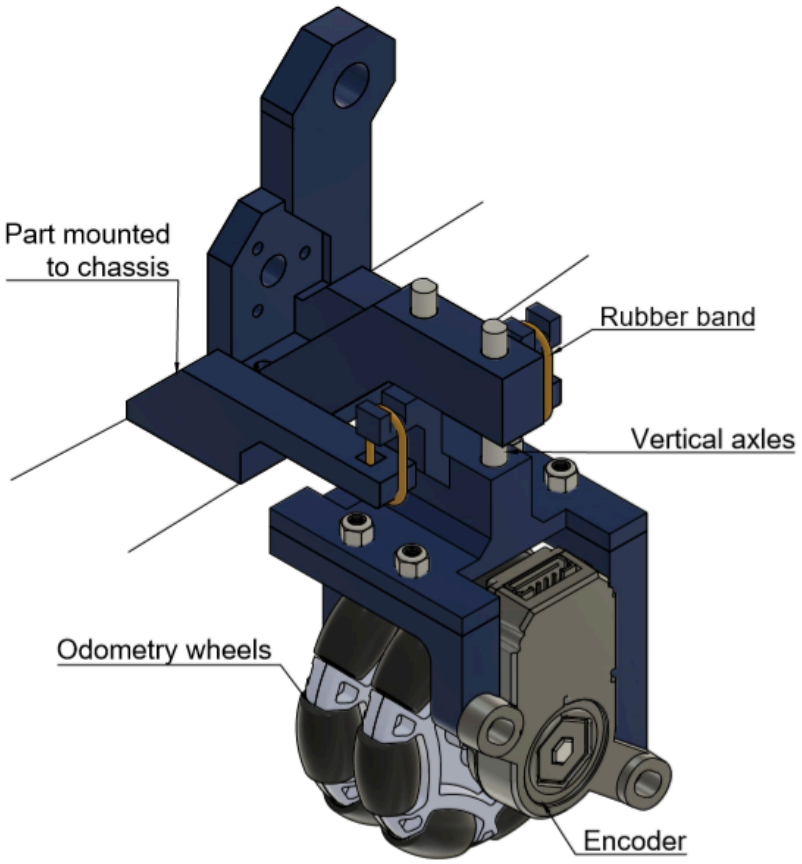
↑ figure 2.6./c

Our first attempt at fixing this problem was to add two more axles to prevent them from rotating as can be seen in *figure 2.6./b*. However, this was not enough to fix our problem and it also did not cover the problem of the odometry rotating in the other way when the robot is moving sideways. In *figure 2.6./c* the tight position of the odometry wheels is shown.

After coming up with some new ideas, we decided that the best solution is a redesign of the odometry suspension. The new design uses only one of the original parts and the rest were scrapped. This design has axles attached to the wheels positioned vertically. The two axles slide in a rig attached to the main chassis of the robot. This solves our problems because it allows the odometry setup to only move vertically with almost no side movement.

Odometry wheels
final design

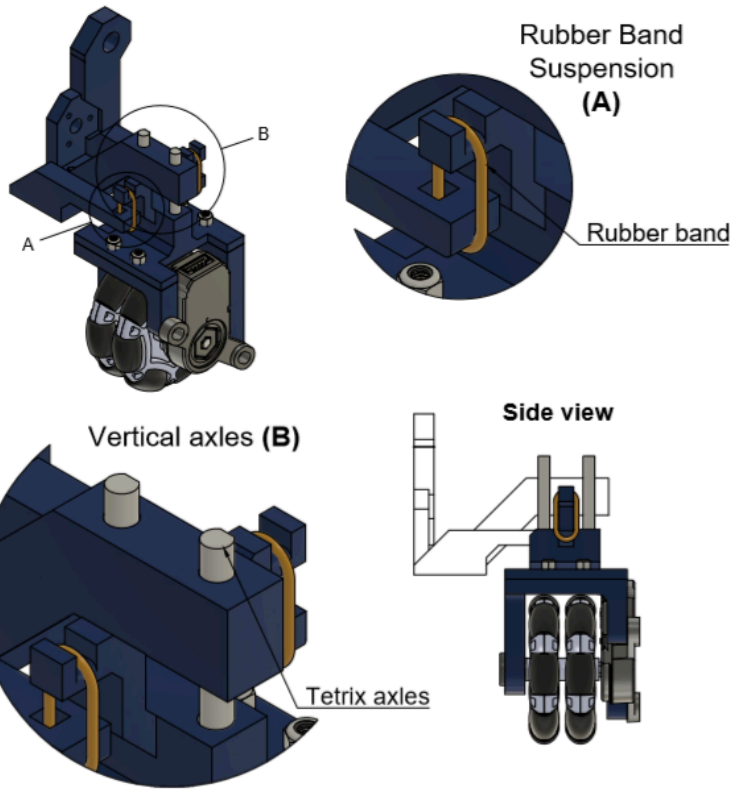
figure 2.6./d



Comparison of designs	First design with horizontal axles	Final design with vertical axles
Positives	• Low profile- allows for parts above it • Rubber bands can be attached easily	• No travel in any directions except for vertical • Adjustable pressure on wheels • Easier to assemble
Negatives	• Some side movement • Can move freely on the Y axis of the robot.	• Rubber bands must be tied around hole

---

Odometry wheels figure 2.6./e
detailed view



Once the two odometry encoders on the side were assembled, we were ready to tackle the final challenge of our odometry - position the third odometry encoder. The third odometry encoder is oriented sideways to the robot's direction of travel and measures the movement of the robot when moving sideways. We wanted to position the encoder along the robot's axis of rotation so it doesn't spin when the robot is rotating. This would ease

calculations for our algorithms. The final position was chosen to be very near the mounts for our motors (*figure 2.6./f*), so we had to change the part slightly to fit it in between the motor mounts. After testing, we found out that the distance between odometry wheels and the chassis was very small and the wheels were rubbing against the frame.

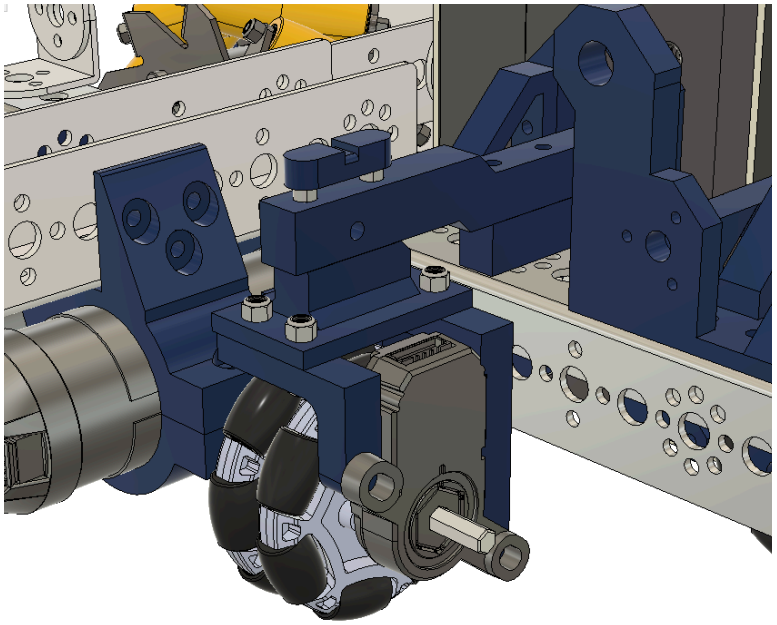


figure 2.6./f

To solve this problem, the only solution was to use the dremel and cut off a piece of the motor mount to make space for the third odometry encoder. After that the odometry no longer rubbed against the chassis and could move freely.

2.7. HANDLING ARM

The second game element that is used for scoring points is the wobble goal. To manipulate it, we decided to design a custom handling arm with 1 axis of rotation. The first rough concept was a “V” shaped piece, where the rod of the wobble goal would be kept in place by a single “finger” operated by a servo motor(seen in 2.7./a).

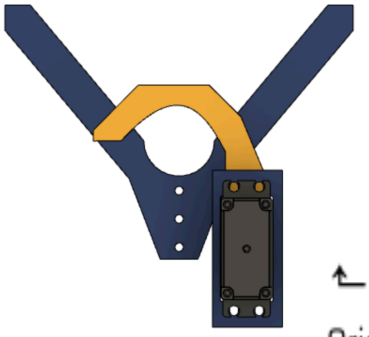


figure 2.7./a
Original gripper concept

Upon further thought, we decided that a redesign was in order, mostly for aesthetic purposes. The result was a gripper with two opposable synchronised finger pieces. The fingers are slightly curved to help guide the rod into a firm grip. The fingers are designed to interlock (see figure 2.7./c), so that they do not cause the wobble goal to twist under force. When gripped, the wobble goal rests in a dent in the backplate of the gripper, which prevents the rod from sliding into the sprockets. Only one of the fingers is powered by a servo motor. The other one is dependent

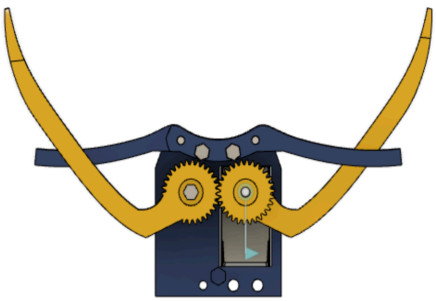


figure 2.7./b
Second gripper design top-down

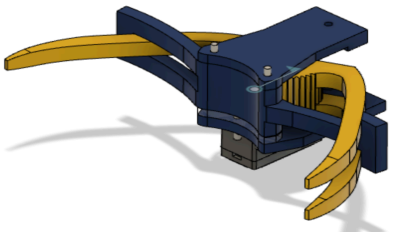


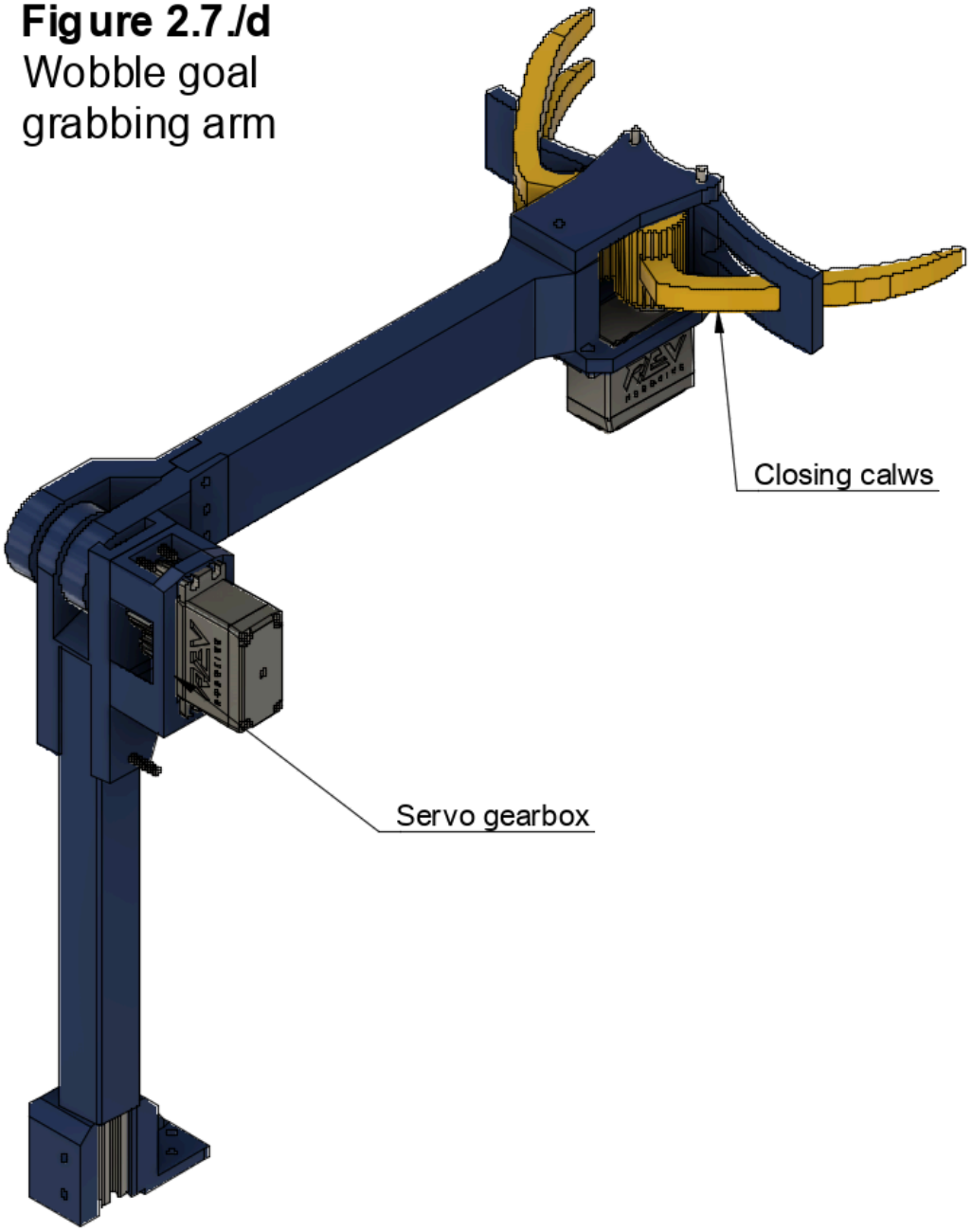
figure 2.7./c
Second gripper design overview

and inverted by a 1:1 gear ratio (see figure 2.7./b). When we printed out these parts, we encountered issues with construction. That was the basis for the third version of the gripper, which addressed the issues (see figure 2.7./d).

To ensure that the wobble goal is within the bounds of the robot at the start of the game, we decided to rotate the gripper and the shaft 180° with a servo. Initially, we opted to power the arm directly from the servo, but soon after, we discussed potential limitations of the servo´s torque output. We came to a conclusion that the servo might not be able to withstand the leverage exerted by the wobble goal at the end of the arm. Therefore, we added a 1:2 gear ratio to the servo, so that it can rotate the loaded arm more easily (see figure 2.7./d). The mechanism is placed on top of an aluminium rod that is covered by a plastic sleeve to prevent the rod from being a distracting element in an otherwise custom design (see figure 2.7./e). The rod is mounted at the bottom with a custom piece. In the new version, this piece is thicker, to support the entire body properly.

At the start of the game, the wobble goal with the arm will be within the dimensions of the robot. Upon start, the gripper and the upper shaft will be turned 180°. This will put it in position to manipulate the wobble goal in front of the robot. To achieve this motion, we modelled a component around REV Robotics gears to ensure the best power transfer from the servo to the beam.

Figure 2.7./d
Wobble goal grabbing arm



2.8. ELECTRONICS

Having figured out the functional parts of the robot, we went on to create a holder for the battery and control hubs. (*figure 2.8./a*)

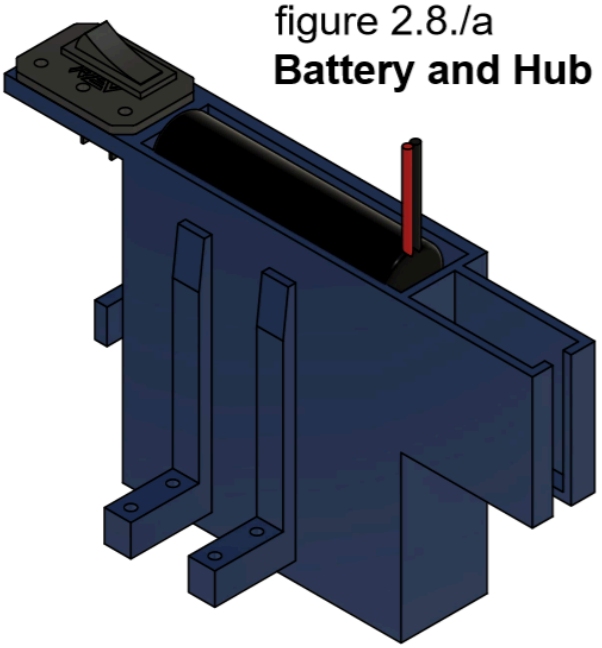


figure 2.8./a
Battery and Hub holder

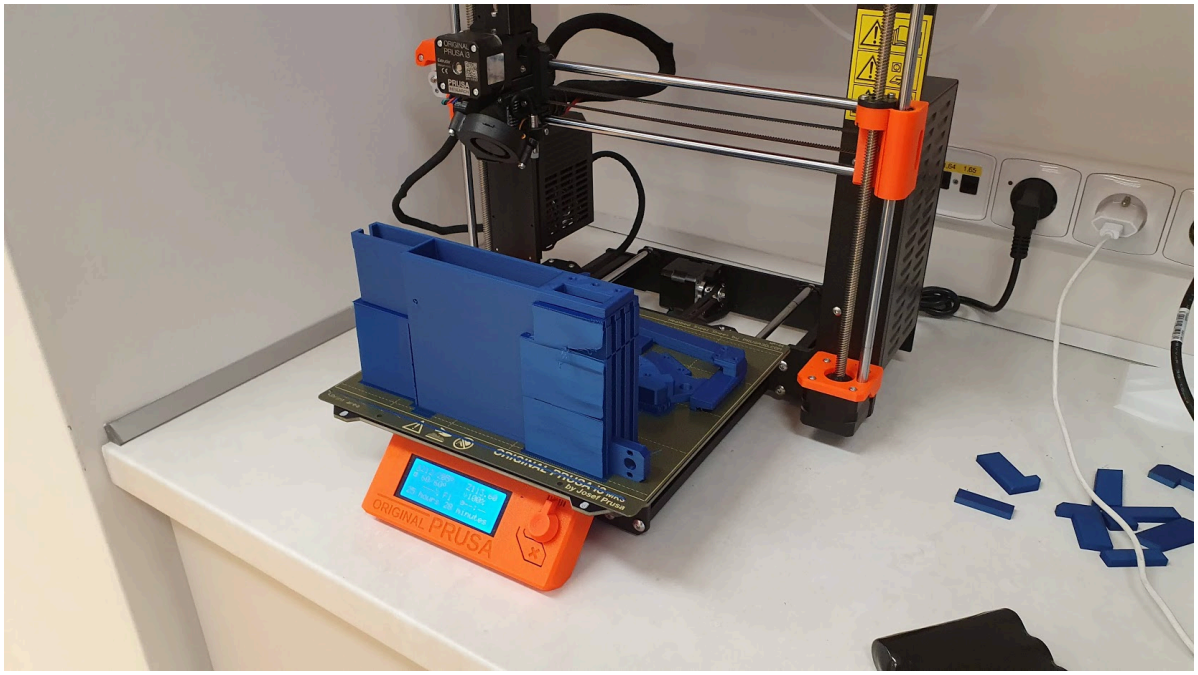


figure 2.8./b

This year we wanted the robot to have perfect cable management. To achieve this we did these things:

- Chassis design - We designed the chassis with cable management in mind. The hollow structure of the tetrix shafts allows cables to fit inside it. Not only does it protect the cables from but it also hides them from sight - improving the overall appearance of the robot.
- Shortening - Each cable was shortened to the exact length to prevent them from being loose, but also to save space. Some cables were tied together to shorten them, but most had to be cut and soldered.
- Hub placement - We designed the holders for our two expansion hubs in such a way that all important ports are easily accessible. The position of the hubs also allows for cables to be mostly led through the central tetrix bar.

Wire shortening process

The wire shortening process is something we had to repeat many times this year. The cable of each motor and servo and encoder had to be shortened or sometimes extended to match our robot's design.

1. First, we must measure how long the wire is supposed to be and mark how much we want to shorten it.
2. We choose the position of the cut where we want to shorten the wire. Ideally this will be in a position where the cable will not undergo any strain and also be easily accessible.
3. We cut the wire and strip the ends of the insulation to reveal the wire inside.
4. The ends of each wire that we want to solder are twisted together so they hold together.
5. Next we solder the two wires together
6. Finally, the wires are wrapped with insulating tape to prevent any short circuits.
7. As a final touch, we wrapped cables in black insulating tape to make them less noticeable.

3. PROGRAMMING

3.1. INTRO

If you take engineering as the body of our robot, then by that analogy, programming would be the brain.

The code makes the robot act. Starting from simple actions, like driving around the field and shooting rings, to more complicated ones like calculating trajectories and analyzing camera images. This year we decided to tackle a big task of creating our own odometry algorithm. It is something we have wanted to do for years, but this year, given how much spare time we had due to COVID restrictions, it was the perfect time to finally take on this challenge. With this algorithm, our autonomous mode will become way more precise and it will also simplify many actions for our drivers in TeleOp.

Our coach Daniel Lessner also plays a crucial part in our team. With his PhD in computer science, he provides valuable insight and makes our lessons very educational.

3.2. SET-UP

3.2.1. Environment:

From the beginning of our team we always used ***Android Studio*** for developing our robot.

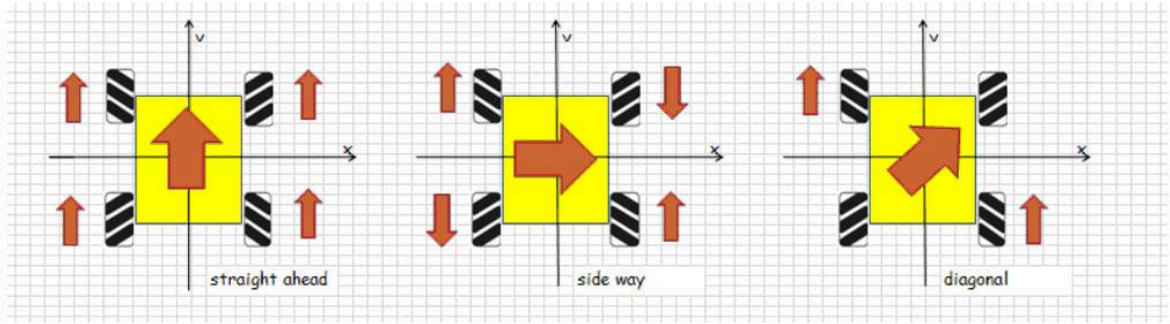
We also decided to use Android Studio because we wanted to have more flexibility with the repository as this year we created our own image recognition algorithm for detecting the skystone. More on that in the autonomous mode section.

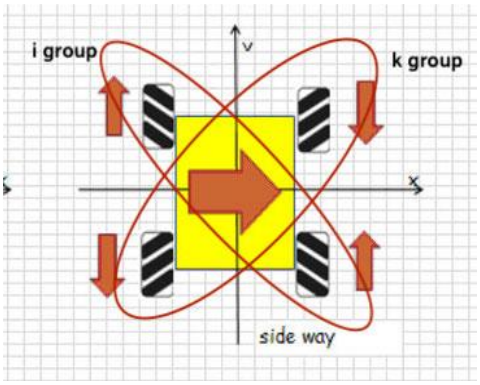
Furthermore, we use the linear op mode classes to develop our op modes. The process began with forking the new repository provided by First on Github and setting it as our own repository we can all collaborate on through **Git** and **Github**. Furthermore, we use **2 Motorola G5** phones that communicate through Wi-Fi direct to control our robot.

3.3. TELE OP

Teleop is the program, which is utilized during the Driver-Controlled Period, which is defined by the second game manual, as "The two minute time period in which the Drivers operate the Robots."

- The most important part of the tele op mode is driving
- The first thing we focused on as programmers was the mecanum wheels gamepad control algorithm.
- To begin with, based on this diagram from roboteq.com, we decided to split our wheels into pairs.





As the non rotational movement of the wheels is only determined by the power of the wheels that are diagonal to each other, we decided to split them into two groups (the naming was a little random).

We then decided on the control system that the left joystick will control the non-rotational movement (gamepad 1 in 3.1.3 section).

After this important step, creating the system was straightforward.

```
87          //Mecanum wheels drive algorithm
88          double y = gamepad1.left_stick_y;
89          double x = gamepad1.left_stick_x;
90          double iPower = Range.clip( number: (y - x) * speedCoe,  min: -1.0,  max: 1.0);
91          double kPower = Range.clip( number: (y + x) * speedCoe,  min: -1.0,  max: 1.0);
92          // Send calculated power to wheels
93          //i
94          robot.leftFront.setPower(iPower);
95          robot.rightBack.setPower(iPower);
96          //k
97          robot.rightFront.setPower(kPower);
98          robot.leftBack.setPower(kPower);
99
```

Now when we gather information about the direction from the controller we can combine them and set the motors' powers accordingly. We also added a speed coefficient which will make the controlling of the robot more precise for the drivers. Furthermore, we also clipped the power values to prevent any errors, which could result from imputing an invalid value into the motor power.

However, we still needed to implement turning for the robot. For that we used the x-axis on the right joystick.

```
87          //Mecanum wheels drive algorithm
88          double y = gamepad1.left_stick_y;
89          double x = gamepad1.left_stick_x;
90          double turn = gamepad1.right_stick_x * turnCoe;
91          double iPower = Range.clip( number: (y - x) * speedCoe,  min: -1.0,  max: 1.0);
92          double kPower = Range.clip( number: (y + x) * speedCoe,  min: -1.0,  max: 1.0);
93          // Send calculated power to wheels
94          //i
95          robot.leftFront.setPower(Range.clip( number: iPower - turn,  min: -1.0,  max: 1.0));
96          robot.rightBack.setPower(Range.clip( number: iPower + turn,  min: -1.0,  max: 1.0));
97          //k
98          robot.rightFront.setPower(Range.clip( number: kPower + turn,  min: -1.0,  max: 1.0));
99          robot.leftBack.setPower(Range.clip( number: kPower - turn,  min: -1.0,  max: 1.0));
100
```

We created a turn variable that we then adjusted to the according motor powers. For turning to happen the wheels on one side have to have a different power than the wheels on the other side. Therefore, the turn value was added to the motors on the right side and subtracted to the side on the left. We then decided to clip the power again to prevent any errors.

As soon as the chassis was finished we tested it out and it worked perfectly.

Moreover, we set up speed coefficients for the driver to move more precisely.

For this we decided to use the bumper buttons. The driver will press them when he wants to slow the driving down or to accelerate it.

```
74          double turnCoe = 0.65;
75          double speedCoe = 1;
76          if(gamepad1.left_bumper) {
77              turnCoe = 0.45;
78              speedCoe = 0.65;
79          }
80          if(gamepad1.right_bumper) {
81              turnCoe = 1;
82          }
```

To make it work we used two if statements and the values were determined on the *feedback* from our drivers.

3.4. AUTONOMOUS STRATEGY

This year, the Ultimate Goal game offers many possible ways to gain points in the autonomous period. Upon inspecting the scoring rules, we decided that the most efficient way to gain points was to focus on shooting down the power shots. In addition to that, we also wanted to focus on delivering both of the wobble goals to the delivery area.

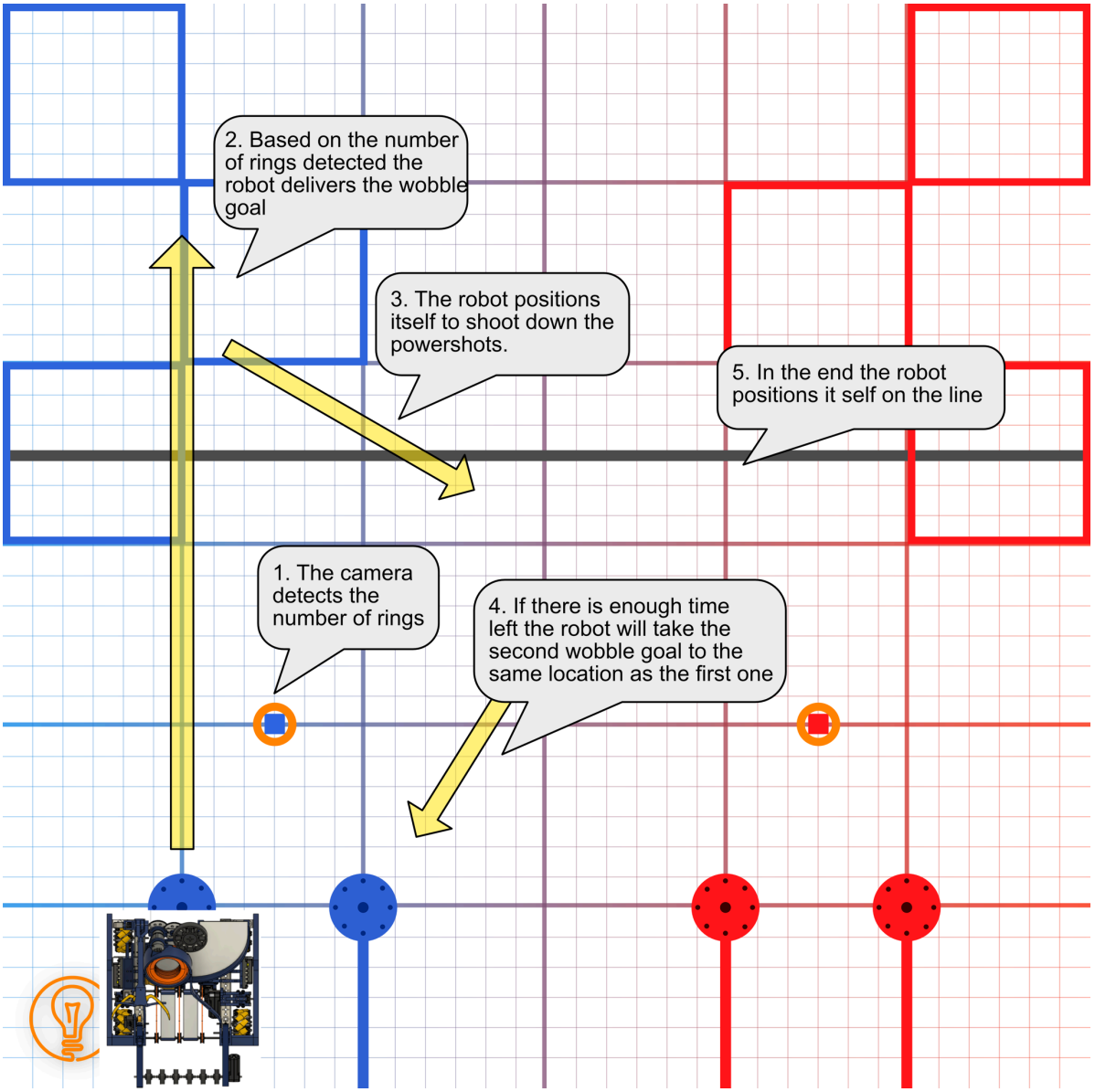


Diagram of our autonomous plan.

Firstly, while impacted by the pandemic and therefore had to work on our algorithms at home. The first algorithm we decided to work on was the

ring detection algorithm, which we managed to perfect at home (see next section 3.5). Once we were done with that, we moved on to creating an odometry algorithm that would utilize our new addition of encoders into the robot design (see section 3.6).

By the time we completed the odometry algorithm, we were let back into the school, meaning we could start creating and testing paths for the robot. Upon deep consideration, we decided it would be most efficient to first analyze the number of rings stacked on the playing field, and then deliver the already preloaded wobble goal to the corresponding delivery area. After that, the robot would get into position to shoot all three powershots.

At the start of the year, we had a long discussion on whether a machine learning algorithm or a hard coded one would be more efficient for detecting the number of stacked rings on the field. In the end we wrote down all the pros and cons for both the options and made our decision based on that.

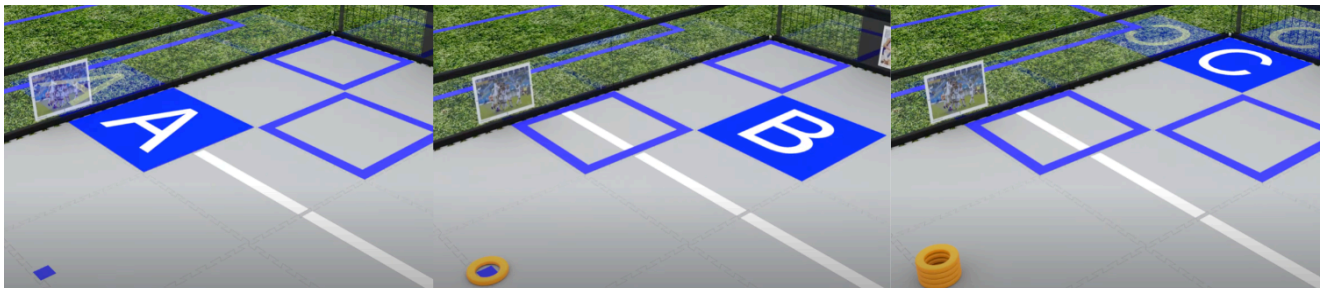
Comparison of algorithms	Hardcoded algorithm	Machine learning
Positives	Easier to setup - finding data to use as boundaries in the code is way faster than machine learning	More reliable in different scenarios - with enough data, the algorithm would be very reliable under any conditions
Negatives	Various variables can skew the results - for example different lighting conditions or backgrounds	Takes long - we would need to feed it a lot of data and pictures before it reaches reliable state

After evaluating all of the pros and cons, we concluded that the best decision was to create a hard coded algorithm. Although reliability under

any light conditions would be nice, given that the contest takes place online, we will be competing from a place where we can control the lighting, thus deeming this benefit not so valuable. Furthermore, given that the competition will happen from home, we wouldn't have many opportunities to gather data from different settings for the machine learning algorithm. It would also most likely take way longer to code, which we didn't want because we wanted to mainly focus on perfecting our odometry algorithm.

3.5. RING DETECTION

The first way to earn 15 points in the autonomous period is to deliver the wobble goals onto the correct target zone, determined by the amount of rings stacked. (See picture)



To solve the problem of where our target zone is we decided to use a computer vision algorithm.
A camera will be placed to face the location, where the rings are stacked. The phone will then take multiple pictures and analyze them. If a majority of the photo analyses match the same result the target zone will be chosen accordingly.

Firstly, we decided to detect the distribution of red, blue and green in each picture and compare it to predefined values set for situations with 0, 1 or 3 rings. The plan was to initially “feed” the program some already taken pictures of the rings on a playing field with different backgrounds, so that it could calculate average amounts of red, green and blue pixels in images with 0, 1 and 3 rings. Once we’ve reached satisfying values of the averages, we would then . The algorithm was very simple; however, it did not work well. The background, as well as the lighting situation in the room, skewed the red, green and blue pixel values too much for the results to be reliable. The scenery behind the playing field would often cover as much as half of the image, which meant that values taken from half of the

image were completely unpredictable. That meant that we had to set very broad boundaries for the red, green and blue pixel values, making the algorithm too imprecise and unreliable to use. The lighting situation in the room played a surprisingly large role in the precision of the algorithm. The difference

Therefore, we decided to approach the problem from another perspective. We knew the rings were always going to be orange and the background gray. So, we decided to score pictures based on the amount of orange. We wrote a function which adds up all the orange pixels in the image from the camera and came to conclude that the differences between a picture with 0, 1 or 3 rings are very easily distinguishable, meaning that we can easily and very precisely determine how many rings are in front of the robot the playing field just from the one number.

```
for (int x = 0; x < width; x++) {
    for (int y = 0; y < height; y++) {
        int pixel = rgbImage.getPixel(x, y);
        int orange = red(pixel) + green(pixel);
        if(orange > 350 && orange < 420 && blue(pixel) < 50){
            score += 1;
        }
    }
}
```

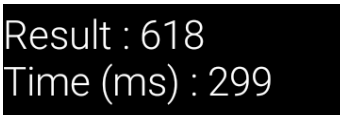
To ensure maximum precision, we plan on running the function multiple times, to minimize the possibility of a single skewed result ruining our entire autonomous period. That is why we want it to execute as quickly as possible, so we also added a few lines of code which measure the time it takes for the measure() function to execute. After some tweaking, we reached a steady speed of around 300 ms, with which we are satisfied.

```
while (opModeIsActive()) {

    long start = System.nanoTime();
    int score = measure();
    long end = System.nanoTime();
    long time = (end - start) / 1000000;

    telemetry.addData( caption: "Result", score);
    telemetry.addData( caption: "Time (ms)", time);
    telemetry.update();
```

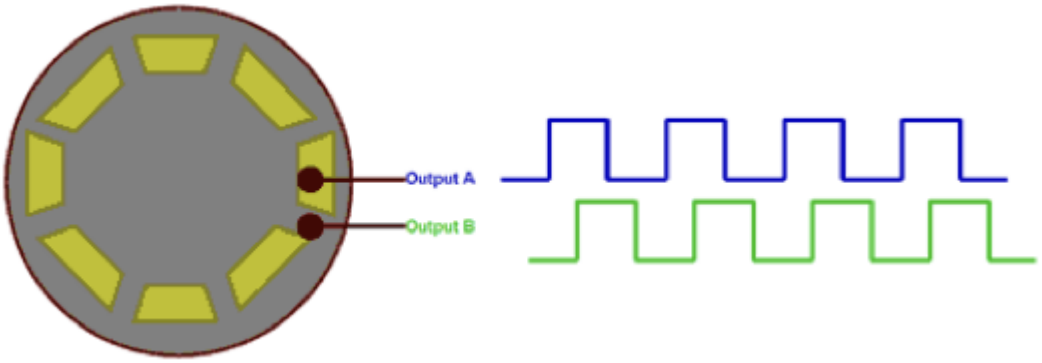
The following image shows the output from this code, when the camera is pointed at a field with 1 ring. The result shows the value of int score, which is the amount of orange pixels detected on the image from the camera. We have found that when pointed at a playing field with a certain amount of rings, the result number will always stay within a certain margin, according to the number of the rings, which is why we can use this number to precisely evaluate how many rings are present on the field.



3.6. ENCODERS & ODOMETRY

This year our team upgraded the robot with encoders, which are sensing devices that convert rotary or linear motion (mechanical movement) into a digital signal that can be analysed. As mentioned in the engineering section we used wheel encoders that interpret the rotary motion from the wheels into digital information.

A rotational encoder generates the data from a disk that is attached to the same axis as the wheel that is tracked. The disk has evenly spaced contact zones and two other contacts A and B. Every-time two contacts touch, a signal is generated. Two signals from the two different sources are created so that the direction of the rotation can be determined. Through a circuit, the signal is transformed into a digital numerical value. Based on how many contacts were made a number either increases or decreases, depending on the direction of rotation. For example, if the disk had 360 contacts for one full rotation the value would read 360. If the disk was then rotated by half a rotation in the opposite direction then the value would decrease to 180. By using this value, it is possible to calculate the physical displacement of the robot.

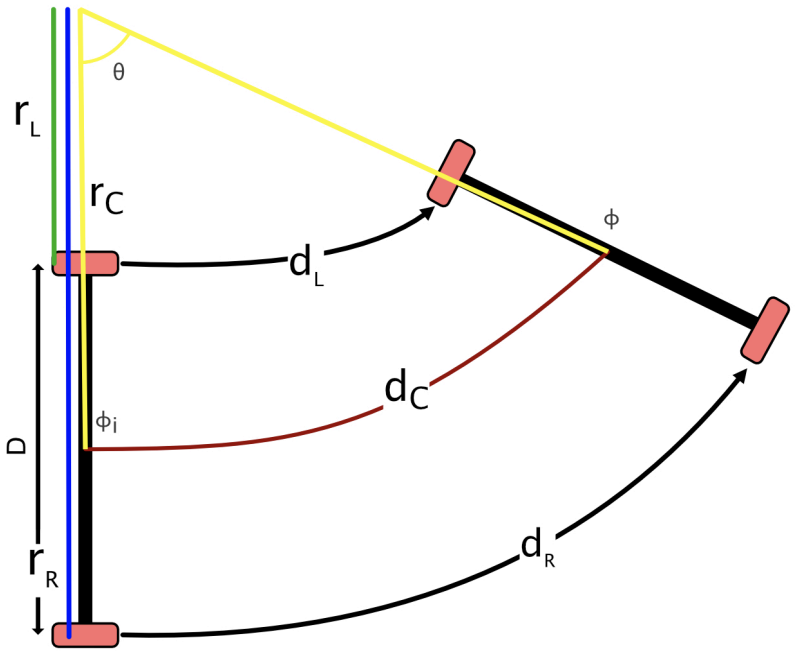


To calculate the displacement of the wheel that the encoder is attached to, it is important to know the diameter (d) of the wheel, for the circumference (C) to be calculated. $C = d\pi$

Furthermore, as in the previous example, I will assume that the encoder

disk has 360 contacts so that one rotation translates to a number of 360. The current number of contacts made will be represented by the variable a. Therefore, the displacement travelled (dt) by the wheel can be calculated by using the equation of: $\frac{a}{360} * C = dt$ The variable is divided by 360 to get the fraction of how much the disk has been rotationally displaced. Later, when multiplied by the circumference, the variable (dt) for displacement is obtained.

For the parallel wheel tracking, arcs can be used to calculate the change in displacement. It is important to mention that even driving in a straight line can be considered a movement in an arc with a radius of infinity.

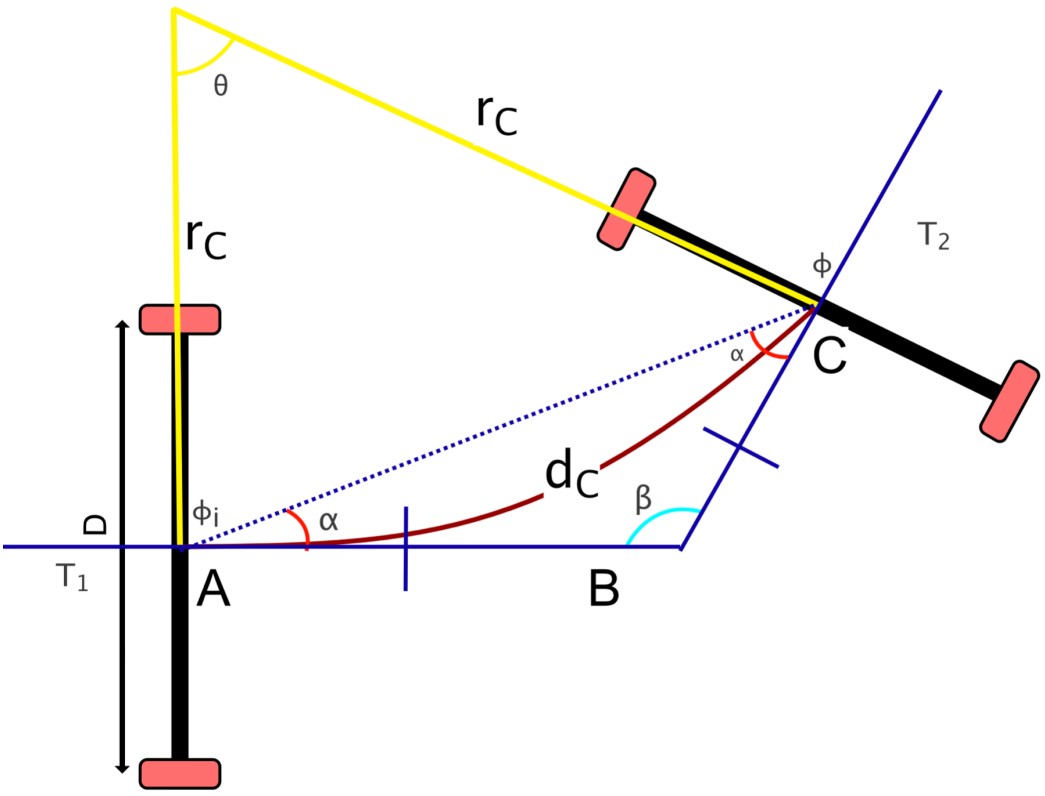


The figure demonstrates an example of how wheel displacement can be expressed through an arc of length. The computer that will be doing the calculations, will use the values from the encoders every

amount of milliseconds (depends on the speed of the computer). Therefore, the algorithm will be calculating the change in displacement between two instances in time or also called “frames”.

$d_C = \frac{d_r + d_L}{2}$The central displacement dC is equal to the average of dR and dL.

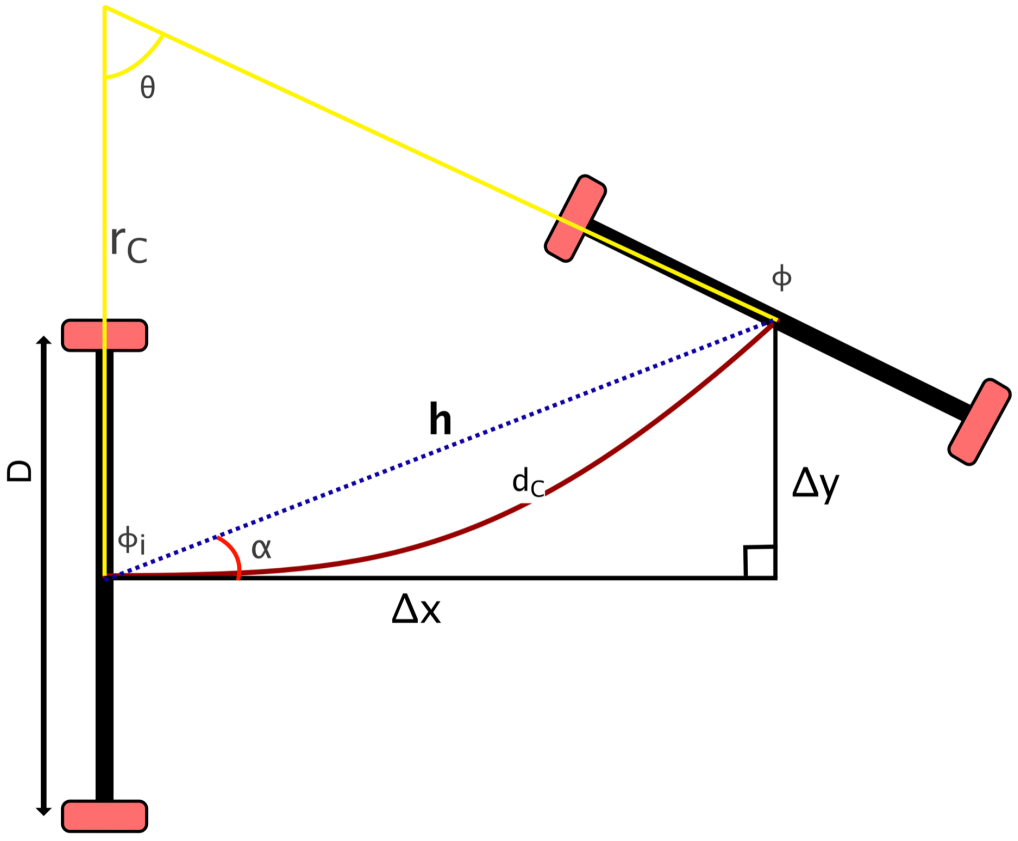
The equation for theta works out to be: $\theta = \frac{d_R - d_L}{D}$Theta is the angle the robot turners during the frame.



There are some relations that should be pointed out. Firstly, a quadrilateral is made of the two radii and the two tangent lines.

Therefore, two right angles are present in the quadrilateral. Making the angle relation: $\alpha = \frac{\theta}{2}$

Now, using right triangle trigonometry, it is possible to find delta X and delta Y by assuming that is the hypotenuse. This estimate is usually enough to achieve optimal precision as the computer can do these calculations hundreds of times per second. Therefore, in a time frame of a few milliseconds, the arc travelled is so small that it can be considered a straight line. However, over a longer period of time the error from this assumption could begin to accumulate and become significant. The tracking should be maximally precise and this is especially relevant if a slower computer is used. Fortunately, using mathematics, the exact value of the hypotenuse can be found.



By knowing θ and r_c it is possible to find by using the cosine rule.

$$c^2 = a^2 + b^2 - 2ab\,cosC$$

In this case of an isosceles triangle, a and b are equal to r_C.

$$h = \sqrt{2r^2_C - 2r^2_C cos\theta}$$

However, it is important to check whether θis equal to 0, which means the robot is driving straight, as otherwise h would then equal 0 as well. The algorithm checks if θ equals to 0 and if it does, then h set to d_C. When the robot is driving straight, d_C is a straight line and therefore, is the same as h.

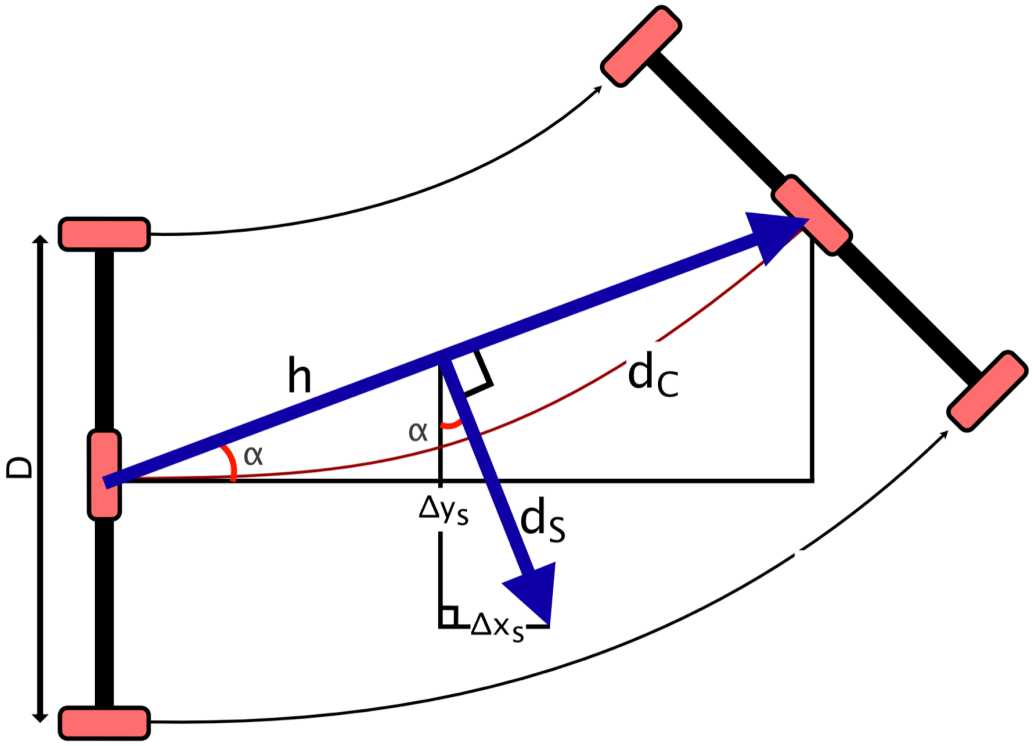
It is important to add the initial angle Φ_i, which is the angle the robot was oriented in, at the beginning of the time frame, in order to preserve the previous changes in orientation of the robot.

$$\Delta x = h \times cos(\Phi_i + \alpha)$$

$$\Delta y = h \times sin(\Phi_i + \alpha)$$

Perpendicular third wheel correction

As described before the robot has a third perpendicular tracking wheel in the middle. Because this wheel is exactly in the middle and is perpendicular, it does not rotate when the robot turns or drives straight. However, if the robot is pushed to the side or the robot is using mecanum wheels that allow it to drive sideways, then the two parallel wheels are unable to detect that. But, the third perpendicular wheel detects this shift.



As seen in figure 8, *dS* and *h* can be considered vectors. Because the wheel is placed perpendicularly to the two parallel wheels vectors *dS* and *h* are also perpendicular.

When a right angle triangle with *dS* as a hypotenuse is drawn the angle at the beginning of the vector *dS* is the same as the angle alpha, shown previously. Therefore, to account for the additional displacement, the shift of the robot adds to the overall displacement, basic right angle triangle trigonometry can be used almost in the same way as with the parallel wheels. The only difference is the switch of cos and sin because as seen in figure 8 the angle faces the *x* displacement axis.

$$\Delta x_s = d_S \times sin(\Phi_i + \alpha)$$

$$\Delta y_s = d_S \times cos(\Phi_i + \alpha)$$

So, when updating the global x, y and Φ coordinates of the robot in a cartesian system, both the parallel wheels and the perpendicular

wheel's tracking information has to be added, together with the displacement from the last calculation:

$\frac{a}{360} \times C = d_{R \vee L \vee S}$

$d_C = \frac{d_R + d_L}{2}$

$\theta = \frac{d_R - d_L}{D}$

$\alpha = \frac{\theta}{2}$

$h = \sqrt{2r_C^2 - 2r_C^2 cos\theta}$

$\Delta x = h \times cos(\phi_i + \alpha)$

$\Delta y = h \times sin(\phi_i + \alpha)$

$\Delta x_s = d_S \times sin(\phi_i + \alpha)$

$\Delta y_s = d_S \times cos(\phi_i + \alpha)$

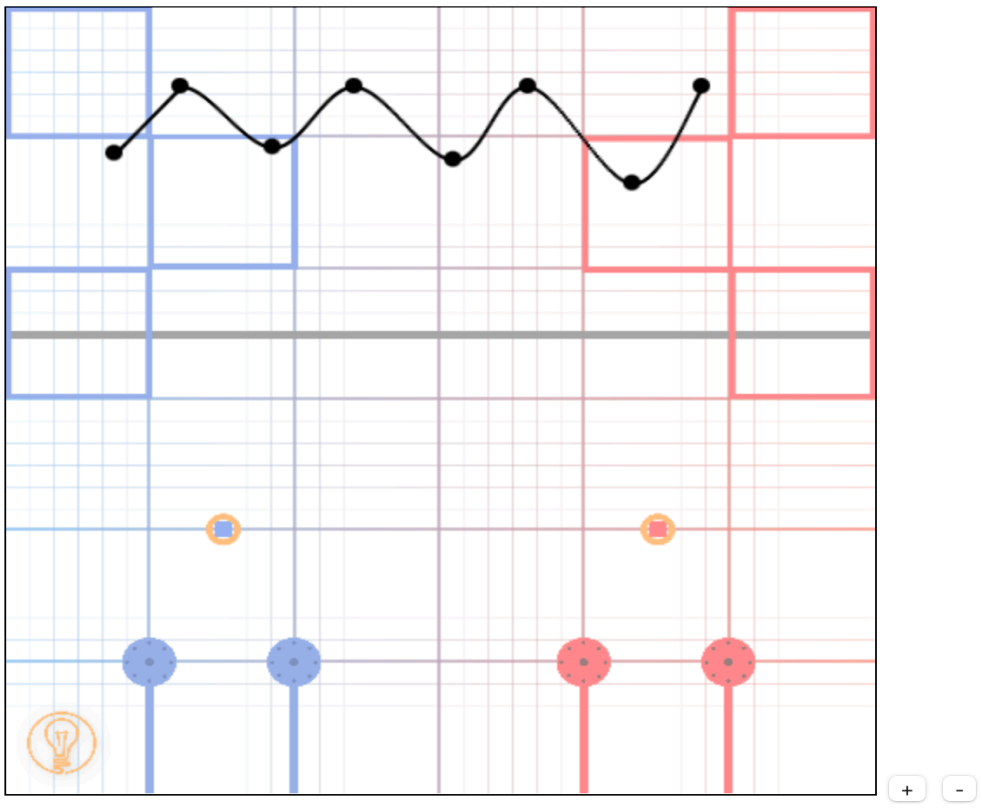
$x = x + \Delta x + \Delta x_s$

$y = y + \Delta y + \Delta y_s$

3.7. PATH DESIGN

With our implemented odometry algorithm we wanted to be able to quickly design a path that the robot can follow both in autonomous and tele-op modes. To improve our web development skills we decided to create a web application that would use the same spline algorithm to graphically represent a path in real-time we could adjust and then export into our robot.

To create our path we decided to use multiple Catmull-Rom splines connected together. We wanted the user to design a path just by moving key points around.



The frame updates 60 times per second and all the points are draggable. The points can be added or removed using the plus and minus buttons on the bottom. The algorithm then interpolates between these points in a seamless manner and generates the curves between. When the path is

finished the coordinates of the points are exported and when imputed into the robot, it will follow the same path.

4. LOGISTICS

4.1. DISCORD SERVER

COVID restriction made in-person meetings impossible for most of the year. Due to that, we had to improvise and move our meetings online. We decided to choose Discord for our online meetings for many reasons. For one, it was already used by many of our members, meaning that they were already familiar with its use and controls. Discord works through a server, in which the admin can create different voice and text channels. In our robotics server, we had general voice channels for everybody as well as separate channels for designers and programmers. The same was the case for our text channels, which we used to arrange times for calls and share files.

Therefore, even despite the pandemic we were able to have weekly meetings on Monday and Wednesday. Discord provided us with a great environment to collaborate!

4.2. BUDGET SHEET

4.3. One of our initial goals was to be more organized. Therefore, the first thing we did when we discovered that we will be receiving more money from school for our club is a Google budget sheet. We wrote down a list of all potential items that we would like to buy and added links to places where to buy. During this process we discovered how budgets work and that we have to deal with expenses carefully. It was a new and enlightening experience for us because we never had to deal with such budgets before. Moreover, we also divided these items into categories.

OBJEDNÁVKY

	Zkušební pole	Link	Množství	Cena za jeden	Celková cena	Stav
Sekce I	~~field perimeter~~	zde	0	18377	0	✗
	~~game set~~	zde	0	10350	0	✗
	PVC potrubí	-	1	2000	2000	
	pěnové podložky	zde	9	350	3150	
	Celková cena (Kč)	**5150**				

	Součástky z USA	Link	Množství	Cena	Celková cena	Stav
Sekce II	REV starter kit	zde	3	13800	41400	✓
	Tetrix kit	zde	1	20470	20470	✓
	REV gear addon pack	zde	1	4600	4600	✓
	REV sprocket and chain addon pack	zde	0	4600	0	✗
	REV 15mm hinge kit	zde	3	230	690	✓
	REV chain tool	zde	2	690	1380	✓
	REV linear motion kit	zde	0	276	0	✗
	Set of 4 Mechanum wheels	zde	1	3500	3500	
	GoBilda Planetary Gear Motor 1620	zde	1	906	906	
	GoBilda Planetary Gear Motor 43	zde	1	906	906	
	Adaptor wire JST VH to 3.5 mm Bullet	zde	3	80	240	
	Celková cena (Kč)	**74092**				

One of these categories was also the budget for team budgeting for travelling to international competitions.

This is a screenshot from our budget sheet that portrays every item we ever considered buying. The first column determines the section the product is from. The second column is the name of the product. The third column is the desired amount, then there is the price per article and overall price. The last column shows the status, if the item was ordered or not.

Overall, we have over 50 articles in the list. From which 45 have been bought and we now have them in our workshop.

For the following season 2020/2021, we had an additional budget of 1000 euro which we used mainly for 3D filaments and minor purchases .

4.3. FINANCES

2019/2020

To begin with, most of our finances come from our school that created different budgets for specific needs of the club. We also contacted some sponsors and attempted to work on our outreach while also increasing our budget.

Budget	Amount
Equipment	$2200 (50000Kč)
Tournaments	$5000 (120000Kč)
Sponsors	Specific items or future collaboration

2020/2021

Budget	Amount
Equipment	$1000 (25000Kč)
Tournaments	$3000 (62000Kč)