

#17089

Powered by

Redstone

- 1.....**Meet the Team**
- 2.....**Our Vision**
- 3.....**Outreach**
 - 1.....Meetings with teams
 - 2.....School presentation
 - 3.....Czech FTC scrimmage
 - 4.....Maker Faire
- 4.....**Game Strategy**
- 5.....**Engineering**
 - 1.....Introduction
 - 2.....Drivetrain
 - 3..... Tilting wheel axis Intake
 - 4.....Launcher
 - 5.....Drum (Rotating Indexer)
 - 6.....Lift
 - 7.....Design philosophy
- 6.....**Robot control**
 - 1.....Introduction & Goals
 - 2.....Development process
 - 3.....Driver automatizations & Bindings
 - 4.....Driver feedback
 - 5.....Code philosophy & Structure
 - 6.....Autonomous
 - 7.....Localizer & Follower
 - 8.....April tag recognition
 - 9.....Indexer logic & Colour detection
 - 10.....Closed- loop motor control



FIRST AGE PRESENTED BY Qualcomm



FIRST
TECH
CHALLENGE

PRUSA
RESEARCH
by JOSEF PRUSA



1. Meet the team

Ms. Turtle

- Minecraft enthusiast
- Role : Being cute



Jenda

- *Team leader, Programmer*
- Rides a motorcycle
- Has a lovely dog Roxy
- Swimming and scuba diving

Dr. Lessner

- *Mentor*
- CODA spirit
- Big fan of ducks and Minecraft

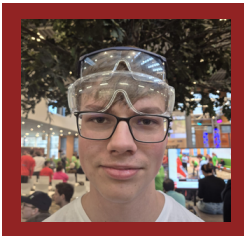


Dave

- *Head of Engineering*
- Enjoys watching The Office
- Fan of the Last of Us video game series
- Always has interesting ideas

Viki

- *Head of PR, Engineer*
- Loves to travel and wants to explore the world
- Archery
- Read lots of literature



Ernest

- *Programmer*
- Has two cats (Persík and Mia)
- Plays the drums
- Big fan of Star Wars

Petr

- *Programmer*
- History, politics, debate
- Loves Dune, Star Wars and The Social Network
- Big reader

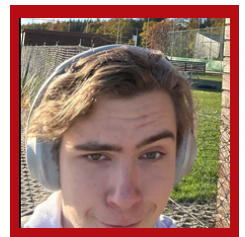


Honza

- *Engineer, 3D printing*
- Always ready to explore nature and to go on a hike
- Leader of the 3D printing club
- Worked at Prusa

Chris

- *Engineer*
- Warhammer collector
- Loves competitive games
- Always sleeping
- Proud Prusa fanboy

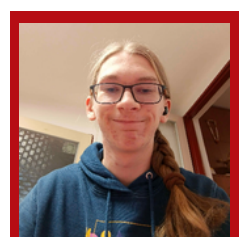
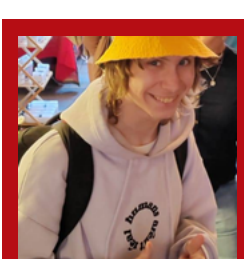


Jirka

- *Engineer*
- Loves 3D printing
- Indoor climbing
- Aviation enthusiast

Honza

- *Cables, "Rule master"*
- Loves video games, TOOL and cats
- Math, Logic and Physics
- Plays the bass



Jáchym

- *PR, Design*
- Creator of the game Five nights at Lessner's
- Cosplayer and snowboarder
- Plays League of Legends + Geometry dash

Dan

- *Alumni (Engineer)*
- Netherlands
- Bouldering
- Delft Aerospace Engineering

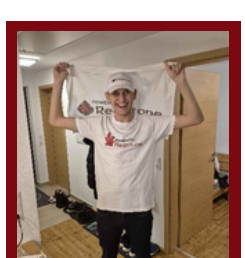


Rošťa

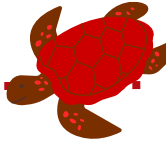
- *Alumni (Team leader)*
- Fan of Football and FIFA
- TU/e Mechanical Engineering
- Sport enthusiast

Jirmy

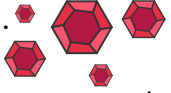
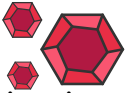
- *Alumni (Engineer)*
- Plays electric guitar
- Ghost fan
- TU/e Architecture



2. Our vision

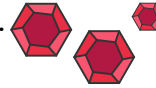


We are a school robotics team based in Open Gate, bringing together students who are interested in STEM fields as well as students interested in photography, design, and social media. Our club creates an environment where students with different talents can collaborate through teamwork and contribute to a shared goal. As a legacy team, we place strong emphasis on preserving knowledge and experience, which helps new members integrate quickly into the team.

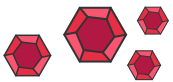


We focus on a hands-on learning approach for our new members. We believe active participation is the most effective way to build technical skills and confidence, encouraging members to experiment, discover new techniques, and step outside their comfort zones. During weekly team meetings, we assign tasks, review progress, and maintain an open environment where everyone is encouraged to ask questions and share ideas. To achieve this, we use Coda as our central organizational platform. It helps us structure our team, track tasks, monitor progress, and clearly divide responsibilities. Keeping all documentation in one place strengthens inclusion, as both new and experienced members can easily access information and actively contribute.

Our main motivation is knowing that the time we spend in the workshop is meaningful and productive. Through strong teamwork, mutual support, and treating challenges as opportunities for innovation and learning we aim to represent the Czech Republic in FTC. Beyond competition, we actively work to help build a Czech FTC community by sharing knowledge, materials, and access to manufacturing resources with other teams. We are also committed to preserving and passing on our team's legacy to future generations of members. Many of our alumni continue their studies at technical universities such as TU Delft and TU Eindhoven, which inspires younger members and demonstrates the long-term impact of our team.



For our budget, we have been preparing it every year based on our experience from previous seasons, taking in consideration our performance, expenses and challenges. This helps us to estimate the budget for the upcoming season. Once the new season is announced, we update the plan according to the materials and game elements needed. Our budget is divided into four main categories: programming, outreach, engineering, and emergency, which covers unexpected costs. We maintain a spreadsheet where we record all expenditures.



3. Outreach

3.1 Meetings with Teams

During our competition in Romania, we established several **long-term connections** that had direct impacts on our team. Through follow up communication with teams such as CyberMoon, Robo Dojo, Brute Force and more, we were able to **reflect on our season** and exchange concrete technical and organizational insights.

In our call with CyberMoon, we discussed not only robot design but also our **CAD workflow**. Where they have recommended and helped us to transition from Onshape to Fusion 360. Overall, collaborating with other FTC teams proved to be highly beneficial, as sharing concrete experiences and solutions directly improved our efficiency and development throughout the season.



Fun celebration of Friendship day with Brutforce





3.2 School Presentation

One of our main goals is to **inspire** more students to join robotics and to spark their interest in STEM. As a **legacy team** our main goal is to keep recruiting new members. Recently we got the chance to present our team and the FIRST AGE to the entire school during an assembly. We introduced it in a way in which could approach students with different interests, focusing on the creativity, teamwork, problem solving and curiosity. We have finished the presentation with a funny video showing the part of robotics, where you meet people with the same interests and hobbies, laugh at everything and you feel like you are part of a bigger family.

This season we have welcomed **many new members**. Our team currently has around 20 members, with more joining regularly. We also work closely with the 3D printing and programming clubs, motivating their members to join the FTC.

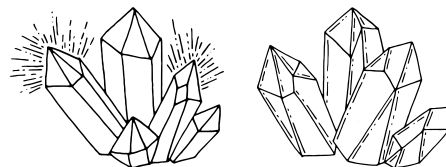
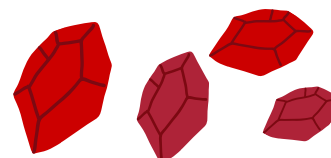
3.3 Czech scrimmage

Before the Benelux competition we were invited to a training scrimmage organized by team AutoFocus, which proved to be a highly **valuable engineering** and **driver-practice opportunity**. During practice matches, including intentional ramming scenarios, we were able to test mechanical components in conditions that are difficult to simulate alone. This revealed a critical weakness in our intake guides, and that repeated high-energy impacts caused significant stress, leading us to redesign and reprint them for **improved durability**. We also participated in **joint driver practice** on a half field. This was our first experience in such conditions and allowed us to explore match tactics, including defensive

possibilities such as partially blocking opposing launchers. These insights directly influenced our strategic planning for future matches. One of the most impactful outcomes of the scrimmage was a **major improvement to our intake system**. After observing and discussing solutions with other teams, we replaced our original carpet-based intake surface with zip ties inspired by Autofocus's design. This change significantly increased intake reliability and consistency, representing a clear performance improvement that carried over into competition. Overall, this experience proved that **collaboration within the FTC community is key** for a success and we are very grateful that we could get this possibility.

3.4 Maker Faire

In May 2025, our team participated in the Prague Maker Faire, where we created an alliance of Czech FTC, FRC and FLL robotics teams, showing to the public the variety which FIRST offers. The main attraction was our interactive booth, where we presented our values and our engineering process. We also had a presentation which we presented on the stage, introducing FTC to the people.

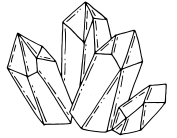




Maker Faire proved to be a chance to inspire people to join the world of robotics while sharing knowledge about the FTC program and its opportunities for learning and innovation. By presenting robotics as an **exciting, accessible, and collaborative** field, we hope to have ignited a spark in future makers, programmers and engineers. Looking ahead, our team is excited to continue building these relationships, growing our skills, and pushing the boundaries of what we can achieve in FTC.



4. Game Strategy



Our strategy focuses on creating a versatile robot that can adapt to alliance needs and changing match situations.

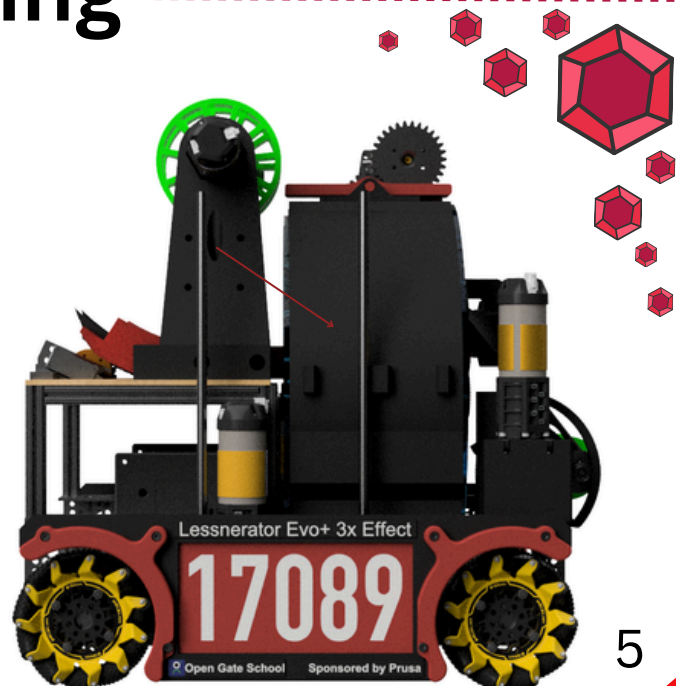
We can launch game elements from both the front and rear launching zones and are also capable of scoring while positioned inside a zone. Since most points are awarded for completing scoring patterns, accuracy and correct sequencing are critical. In real matches, game elements are shared between alliance robots and intake control can be slow and unpredictable, so we sort and manage game elements before launching. Our revolving drum allows us to hold multiple pieces and launch them individually, giving us precise, on-demand control over scoring. This enables us to correct or complete scoring patterns if they are disrupted during the match. At the same time, the system supports fully automatic pattern launching using color recognition and auto-sorting when conditions are stable.

We also implemented a lift to target an additional ranking point. Choosing between lifting and continued scoring is a decision made directly during the match and it is based on the time, score difference, and alliance priorities, allowing us to balance match points and ranking points as needed.

5. Engineering

5.1. Introduction

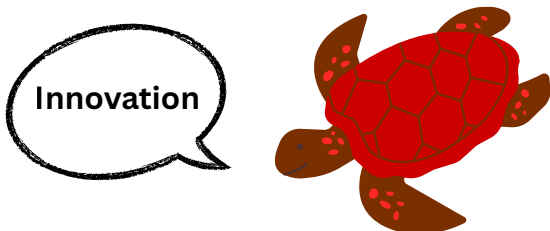
We are willing to explore unconventional solutions because they help us learn faster and often lead to improved performance, showcasing our team's innovation. We see the robot development process as a continuous journey where teamwork plays a key role, as ideas evolve through discussion, collaboration, and shared problem-solving. Since experimental ideas carry risks, we rely on peer design reviews, structured planning, and parallel development paths that allow us to test new concepts without delaying critical robot functions. A major advantage supporting this approach is our access to advanced manufacturing technologies. Thanks to 3D printing and CNC machinery, we can rapidly prototype, iterate custom-fit components, and refine our designs efficiently.



5.2. Drivetrain

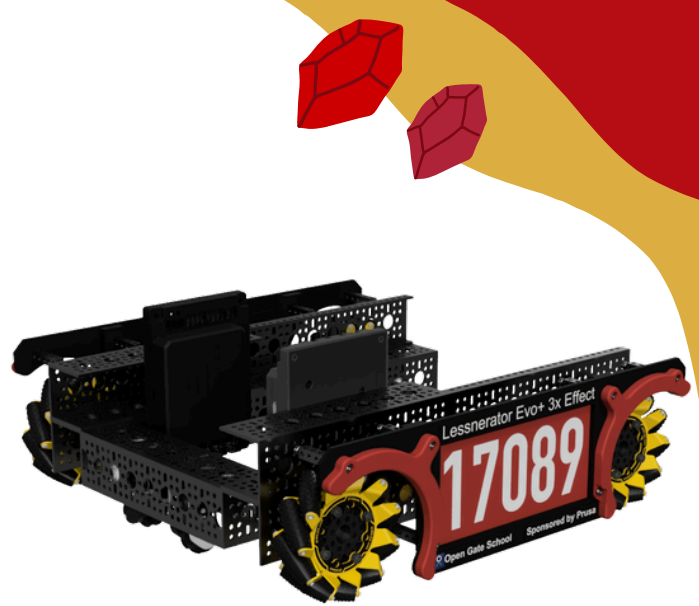
Unlike in previous years, where multiple tools were needed to remove or add a single component, this year's design solved this issue by using the **Gobilda U-channels** as the core of the drive train. It offers excellent. **stability and wide range of mounting points**. Building upon the core, we designed the drivetrain to be disassembled with only one m4 hex key. For example, each motor can be removed by loosening two bolts, allowing it to slide out from the bottom of the drivetrain, allowing us full access to the entire bottom u-channels. Furthermore a **base plate** was added to the drive train to protect the inside cables from being dragged on the ground.

However, one thing that our design didn't account for were the vibrations that were made by the launcher spinning at over 3200 rpm, which gradually loosened the bolts on the drive train. We solved this issue by introducing **3D printed TPU rings**, that act as a **locking mechanism** by adding friction between the bolt and the plate, preventing them from coming loose during the match.



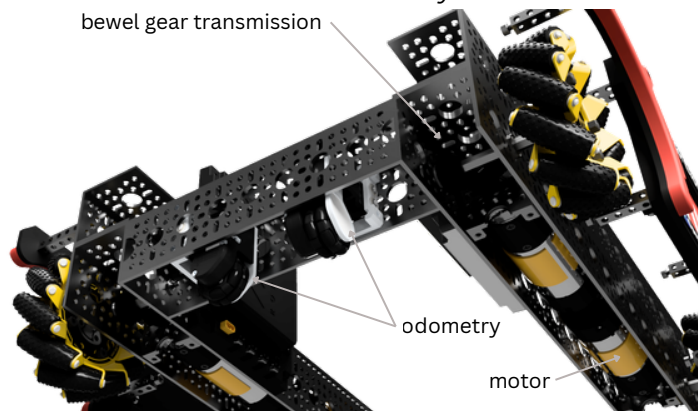
5.3. Tilting wheel axis Intake

The tilting-axis intake was developed through **multiple iterations**, where each is addressing failures identified in the previous design. The first prototype that was made is a basic proof of concept intended to **test the feasibility of a tilting wheel intake**. Upon the successful test of the prototype we decided to iterate the design, where the second iteration focused on **improving the form and stability**. The intake was made sleek and **fully 3D printed**. While this reduced size and improved integration, it introduced major issues. Placing the motor directly on the tilting axis made the system front-heavy, and the printed gear transmission proved unreliable, with gears failing quickly under load.



Drivetrain Assembly

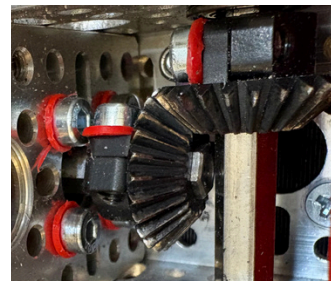
bewel gear transmission



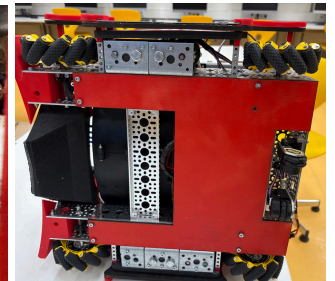
odometry

motor

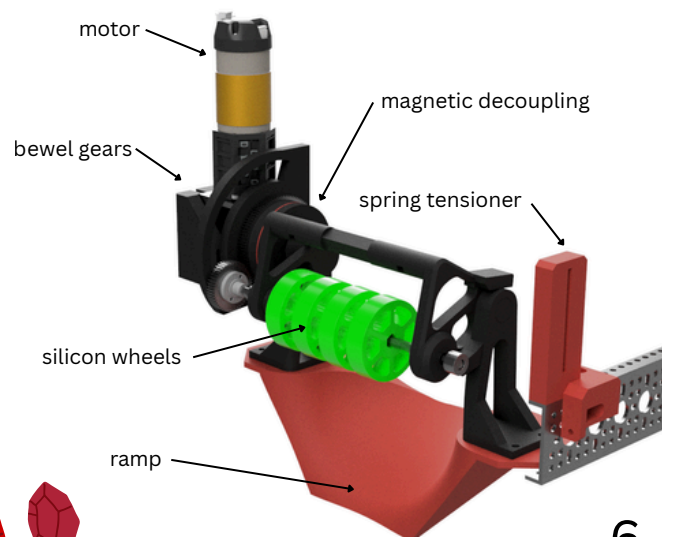
Bottom view of the drivetrain



TPU rings on motors



base plate



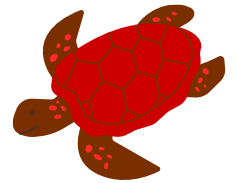
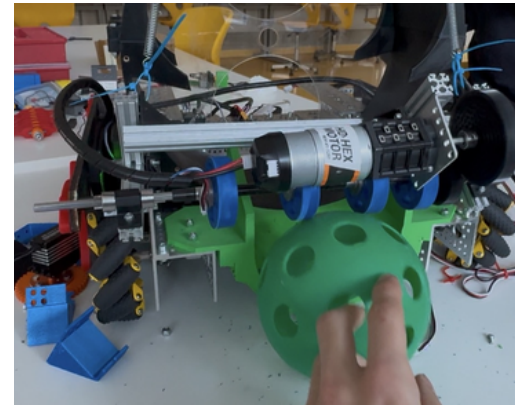
Intake Design v4

The third iteration built on this design by **relocating the motor** from the tilting axis onto the robot chassis. The transmission was upgraded to **metal gears**, significantly improving its reliability. This version also included the first attempt at a **decoupling mechanism**, that was intended to prevent the intake from suffering damage, when a ball gets stuck inside of it. However, with this iteration new problems emerged. The intake spun at such a high speeds that the **PLA bearings** overheated and fused together. The system was also too light, allowing the intake to remain stuck in the upright position when a ball was loaded. Additionally, the decoupling approach relied on gear skipping, which led to repeated gear damage.

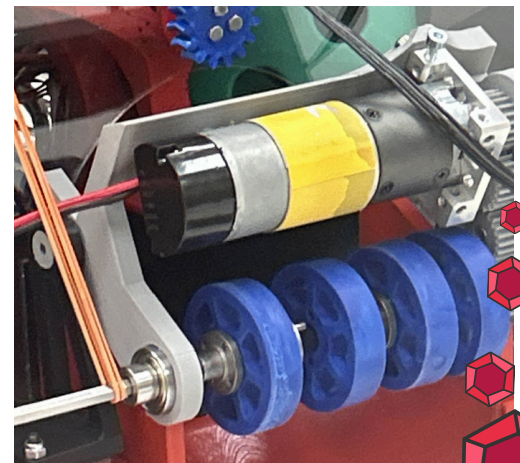
The fourth iteration addressed these issues directly. The 3D-printed bearings were replaced with metal bearings, **eliminating the melting problem**. The decoupling mechanism was redesigned to use **magnetic coupling** instead of intentional gear skipping, **preventing transmission damage** while still allowing disengagement under excessive load, caused by a stuck ball.

Despite these improvements, the intake still exhibited some functional flaws. The ball occasionally fell out of the intake, and the ball sometimes slipped on the ramp. Slipping was resolved by adding **rubber to increase its grip**. On the other hand preventing the ball from falling out proved to be more challenging. An initial attempt used a **miniature carpet** inside the barrel, which reduced fall-outs but introduced a new failure: the ball could jam midway, severely reducing its reliability and sometimes even causing us to be locked out of the match. We tried adjusting the carpet position, but it did not yield any significant results.

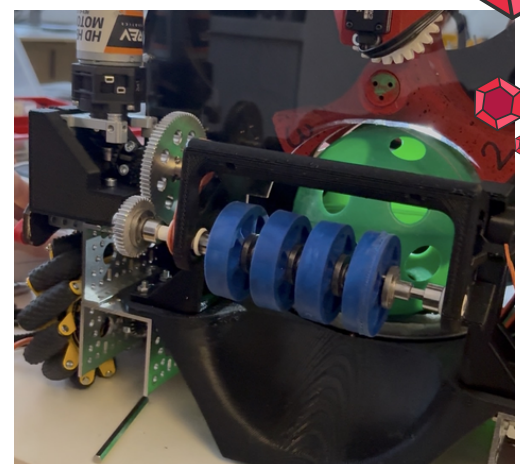
The **final solution** that resolved the issue was a **change in approach**. Zip ties were added to the forward-facing end of the intake to act as a flexible blocker. **Zip ties** were chosen because they provide directional flexibility while remaining stiff enough to retain the ball, which is a **mechanical behavior** that would be difficult to achieve reliably with a 3D-printed component from materials such as TPU. This solution significantly improved consistency without introducing new failure points.



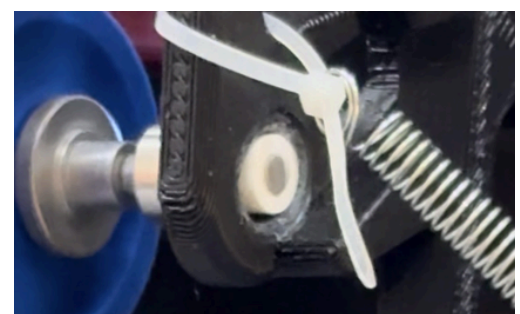
First Prototype



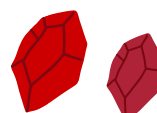
Second Iteration



Third Iteration



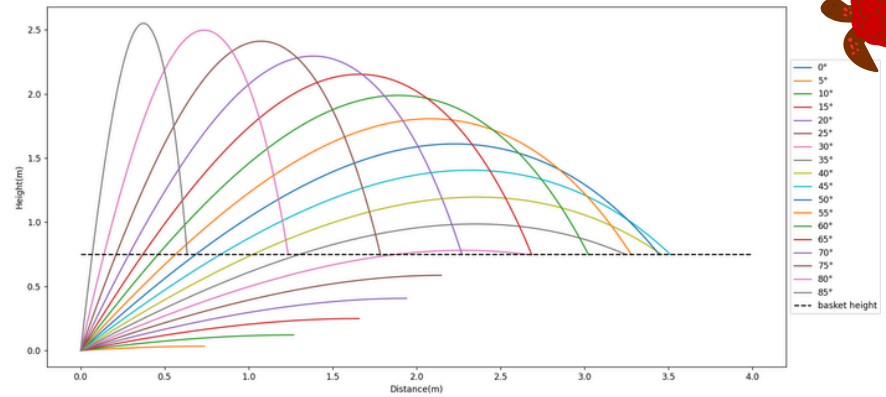
melting issue



5.4. Launcher

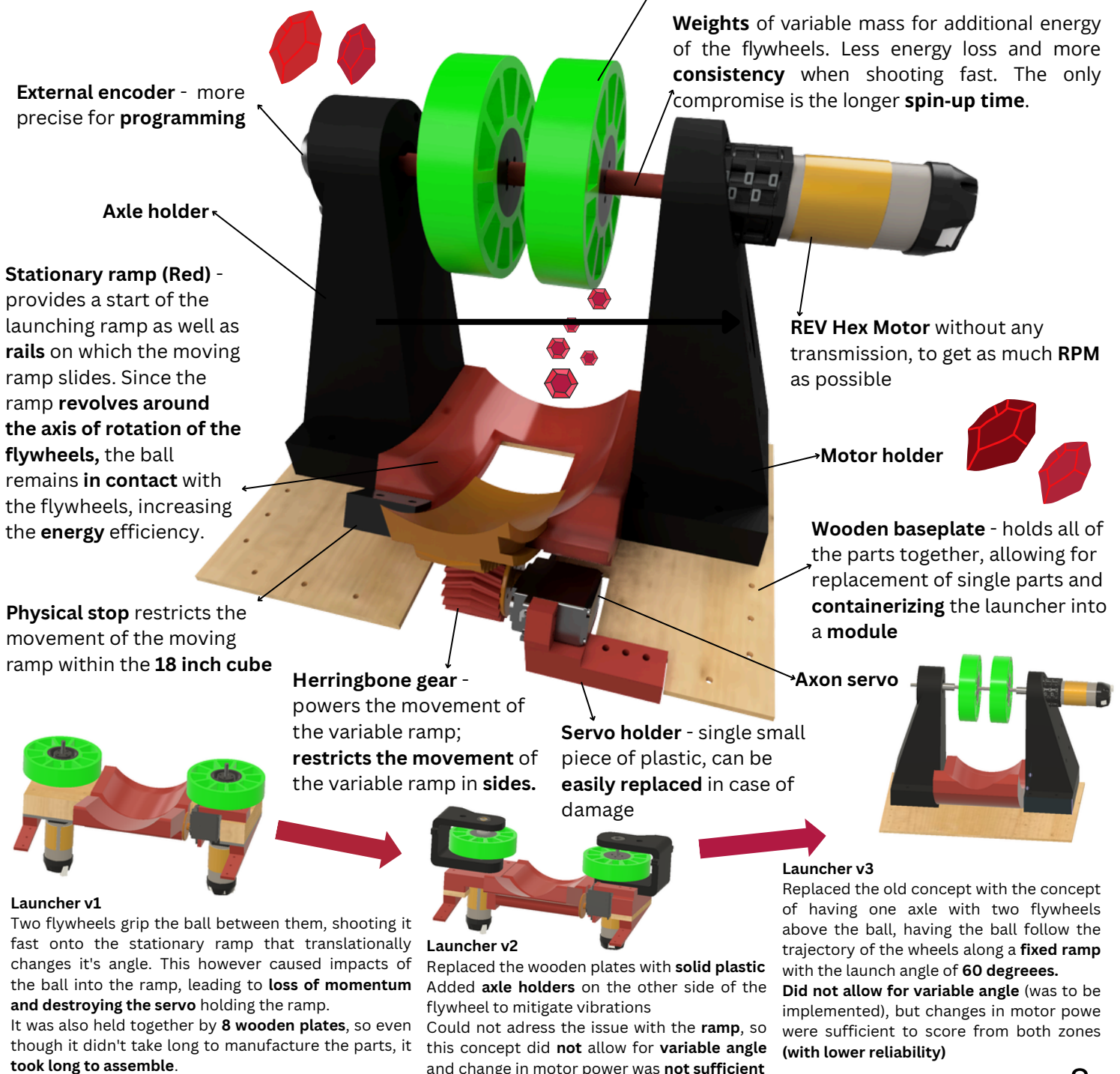
A graph showing the trajectory of balls launched with speed of 10 m/s and **different initial angles**.

The angles between 40° and 65° contain a large range of distances reached, from 2.75m to 3.5m, and smaller distances can be reached with **smaller motor power**.



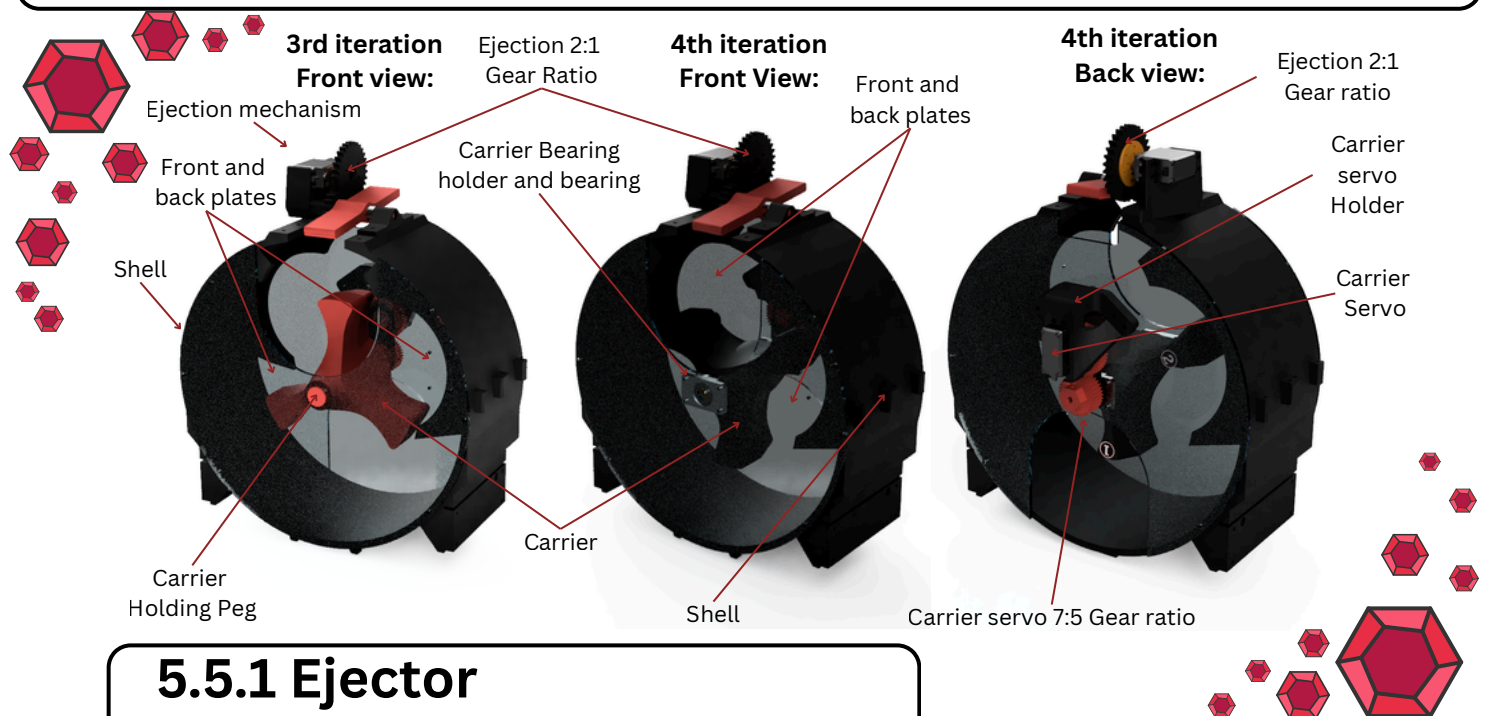
Moving ramp (Orange) - allows the initial launch angle of the ball to be varied from **43 to 65 degrees**. Unlike in the previous version of the launcher, it allows for the use of steeper angle when shooting from the near zone and shallower angle, when shooting from the far zone.

Two flywheels were chosen from the beginning, as to increase the contact area with the ball, so that **more energy is transferred**, and to increase the moment of inertia, so that the time taken to **stabilize the wheels speed** is lower



5.5 Drum (Rotating Indexer)

In the first iteration we simply had the drum 'shell' with two sheets of plexiglass on each circular base, and an internal carrier used to move the balls in a circular motion. This however had several problems regarding reliability. As such the second iteration worked on this, increasing the thickness of the carriers holding pegs and adding a gear ratio to speed up rotation. This introduced problems with the rotational precision with there being significant uncertainty in the barrels position at all times due to gear slack, this lead to the drum having trouble intaking in scenarios where there were other balls inside of the drum, as the balls' weight caused axial misalignment, blocking the intake and ejection of balls. The third iteration thus further focused on improving the systems reliability, mainly by chamfering the edges of the carrier increasing the tolerance for intaking balls. For the fourth iteration we switched to using bearings to hold the carrier, which was changed as well, shortening it from both ends to remove the pegs and adding a 5mm hex hole throughout the axis of rotation. These changes have made the drum even more reliable, reducing the gear slack to a negligible amount, allowing for consistent intaking in any situation.



5.5.1 Ejector

The first iteration was a simple pusher, because of the reliability. This idea was quickly scrapped, even before prototyping, due to concerns that the carrier could accidentally damage the ejection mechanism while sorting balls. This set a precedent that any future ejection should be resistant to breaking from the drums use. Thus the second iteration was based on a spinning wheel on the top of the drum, contacting the balls through the cutout. This was also scrapped in design because of size limits, additionally this iteration had problems with reliability due to the mechanically complex and unsustainable wheel mounting. Since we cannot risk a failing ejection in game we chose a simple flap connected to a servo for the next iteration due to the simplicity. To prevent mechanical failure due to the drum the flap is short enough to where it physically cannot interact with anything but the ball. This design worked really well, combining the mechanical reliability of the pusher and the sustainability of the wheel ejector, only consisting of 5 parts in total. The only problem with this was its speed of ejection, because the direct attachment to the servo meant that the flap speed was only around 60 deg/s. So for the current iteration we chose to take the flap design and add a gear 2:1 ratio between the servo and flap, allowing for fast ejection, while still being simple and reliable enough that we have not had to repair it once since installation

5.6 Lift

The lift ended up being the last item on our functions list, as we wanted it to be built **around the other components**, so that we do not limit their **functionality** by it. The lift consists of **two parts**, each located on one side of the robot, the first one having 2 threaded rods, the second only 1. This enables to form an almost-right triangle from the rods, which can be placed into the corner of the parking zone, so that the alliance robot can use the full area of the square. **Only the 2-rodged version is shown**, as the 1-rodged is the same, except the **exclusion of the second rod and its gear**.

Threaded gears - 3d printed gear, which have a **long M8 nut** inserted into them **mid-print**. As the rotate, the nut screws on the threaded rod, but since it is restricted in movement by the **plastic spacers and metal Ls**, the robot **moves up** instead. As to reduce **friction**, the critical contacts with the spacers and Ls use **bearings** and the whole gearbox is **lubricated**.

Ls - customized metal Ls form an **U-channel**, on which the lift stands, and which is **attached** to the rest of the robot

Stands - the rods are screwed into them and they stand on the ground, while the robot is lifting, **providing ground and stability**. To increase **stability**, they are **interconnected by a square rod**.

REV Hex Motor - with **4:1** transmission, as to give the gears enough torque, while maintaining a decent lift time (**11.5s**). Our short experiemnce has shown, that 3:1 lacks the **required torque**, while 4:1 should suffice

$$t = (\text{Height/pitch diameter}) / (\text{rpm}/60)$$

$$t = (430/1.5) / (1500/60)$$

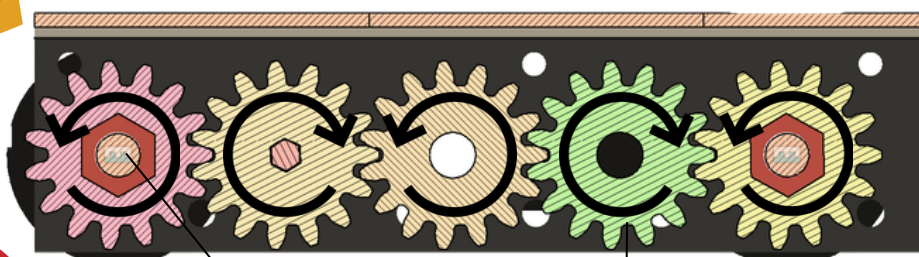
$$= 11.5s$$

Time (3:1) = 8.6 s
Time (5:1) = 14.3 s

Threaded rods (M8) provide more **stability** and less **slipping**, compared to linear sliders, and their movement mechanism takes **less space**

Motor gear - powers the whole **gearbox**

Transfer gears - While the motor power could be transferred with a **belt or a chain**, which would have lower loss of energy due to friction, we still ended up choosing gears to transfer the energy. The belt would be prone to **slipping**, as it would hold **half of the robots mass** and the chain would be challanging to properly **tighten**, given the **space limitations**, and would still require 2 **gear-to-gear** interactions, as the non-3d-printed parts can't be on the same axle as the threaded rods, due to **strength and mounting issues**.



Space for the **M8 long nut**

Direction of rotation when going up

We are going up !!

5.7 Design Philosophy

Our engineering process is **solution based** rather than linear. Design, manufacturing, and testing are closely linked, and it is expected that we loop back to earlier stages after testing. In this workflow, **failure is treated as an input, not an exception**, each failed test provides data that informs the next design iteration. A key enabler of this process is 3D printing, which allows rapid prototyping and **fast validation of ideas**. We regularly operate in a **same-day CAD-manufacturing-testing** loop, enabling us to evaluate concepts almost immediately after they are designed. Early prototypes are often intentionally rough, as we prioritize **speed and functionality over visual refinement** during initial development.

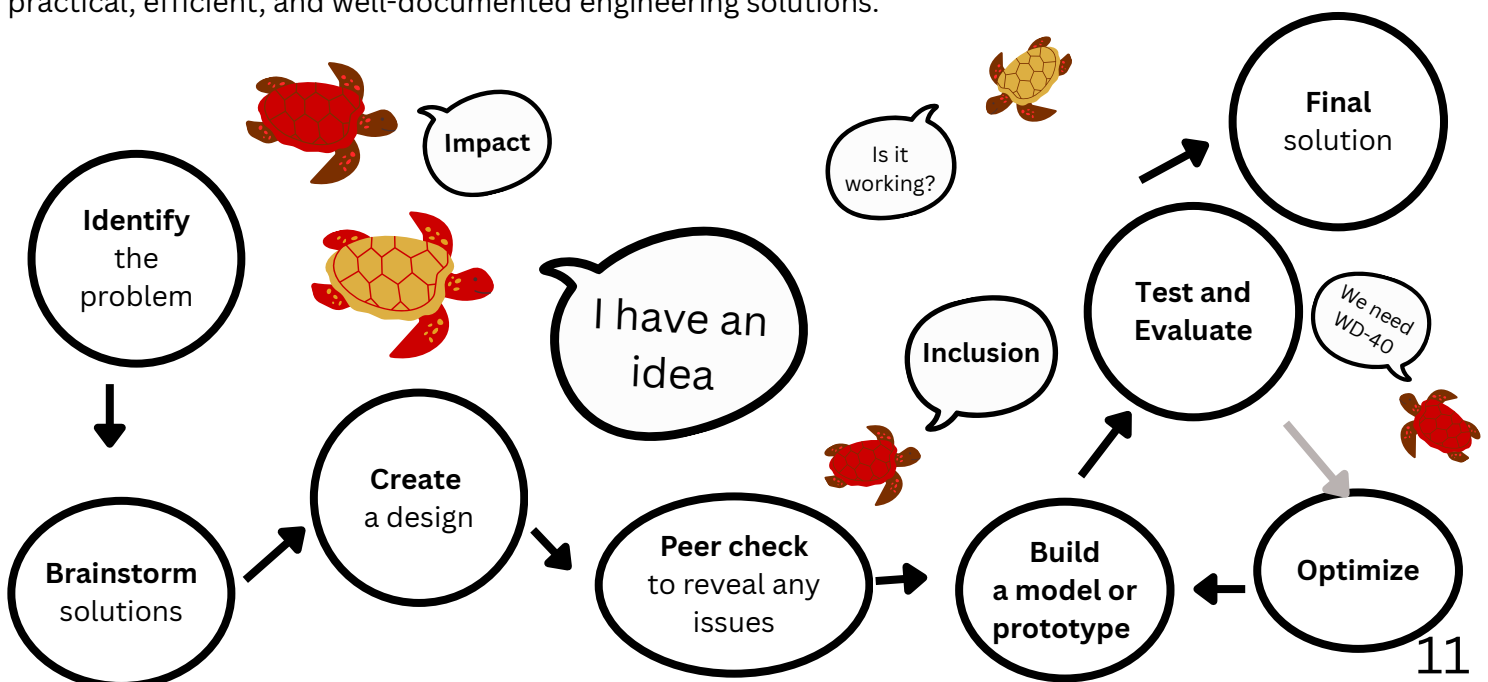
All new ideas are first **discussed with the entire engineering team** to identify potential issues that might be overlooked by a single designer. As part of this brainstorming phase, we also create benchmarking parts to guide design decisions. For example, we printed TPU test cubes with varying infill percentages to **evaluate** rigidity and determine the **optimal infill** for our bumpers.

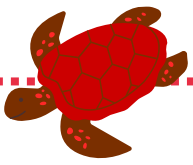
Before parts are integrated into the robot, they undergo **peer review**, where assemblies are evaluated for **accessibility, serviceability, tolerances, and potential failure points**. This step helps us reduce maintenance time and improve long-term reliability.

To ensure documentation and traceability, we use Coda. Every team member logs their **workload, tracks progress, and records design decisions**. Coda also organizes goals, stores key engineering and programming information, and provides resources for new members to quickly get up to speed. This system allows us to maintain a clear record of the development process throughout the season.

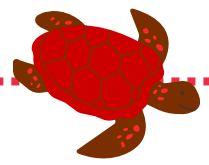
In terms of **sustainability**, while we are actively working toward broader **recycling solutions**, we already demonstrate long-term commitment by sorting and recycling plastic waste consistently for the past three years, reinforcing that **sustainability is an established practice** rather than a future intention.

For our process, we follow a structured process that guides us from **Identify the problem** to delivering the **Final solution**. We begin by clearly defining challenges and constraints during **Brainstorm solutions**, where every member contributes ideas. From there, we **Create a design** based on the most promising concepts and have a **Peer check** to reveal any issues, ensuring reliability, and avoidance of any risk. We then **Build a model or prototype**, allowing us to physically validate ideas and quickly detect weaknesses. Through **Test and Evaluate**, we measure performance against requirements and identify areas for improvement. When necessary, we **Optimize** the design by refining materials, structure, or functionality, often repeating earlier steps to improve outcomes. This iterative workflow ensures that each stage focuses on developing practical, efficient, and well-documented engineering solutions.





6. Robot control



6.1 Introduction & goals

Programming is important not only for the **autonomous period**, where we try to be compatible with other teams, while scoring as many points as possible, but also for the **TeleOp**. There we aim to offload as much as possible from the driver, so they can focus on higher level strategy. We try to build our code in a way so we can be **reuse logic** for both tasks.

Two years ago our team faced a complete change of the programming team, meaning a **completely new generation** of absolute rookies had to take over and since we didn't know better we wrote all code from scratch. This proved to be the wrong direction so this season we used tools like **NextFTC** and **PedroPathing** that have greatly aided us in making our code more **readable, reliable** and, most importantly, **easier to change** on the fly.

Lastly, a **clear goal** remains not to leave the **next generation of programmers** in the same position that we had found ourselves in. Thus we are constantly updating a shared document with tutorials and tips and tricks, which is now proving useful as we had **2 rookie programmers** join the team. Due to this we are also using **Git features** to a much larger extent. Git allows us to create issues to be assigned to specific programmers, create a new branch where they can isolate their changes. When the feature is tested we merge them and all features get **implemented without interfering with each other**.



6.2 Development process

A key focus of our entire software stack is the ability to make **quick changes**, both major and minor, and have **easier access to telemetry**/other data from the robot. The regular FTC development process is very inefficient and usually makes as many changes as possible before building the code (which takes up to 2 minutes) and is therefore extremely unfriendly for debugging, adjustment of parameters and similar minor changes. We opted for using a fastload plugin, which **speeds the build time up** to around 1-2 seconds, making writing **fast changes** frictionless and easy. Another tool we actively use is Panels, a dashboard that shows us live graphs, telemetry, allows selected **variables to be changed live** and even **displays the real-time robot position** on the field. This **streamlines the process of tuning PIDF controllers**, making **on-the-fly changes** to paths during practice etc.

OpMode selection

Virtual controller

Telemetry

Robot position and path

Change variables without building

Graph plot of any variable

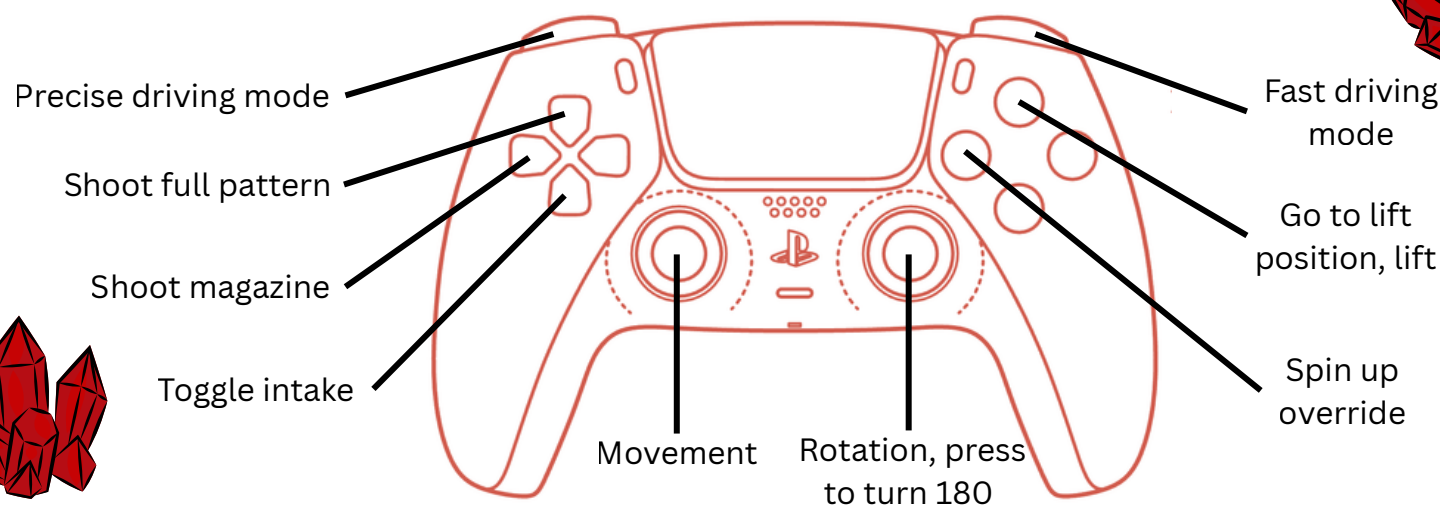
6.3 Driver Automations & Bindings

This season we decided to use only a **single controller** to control the robot to avoid any complicated coordination with the second driver and tried implementing a **large number of automatizations**. Intaking, for example, is almost completely autonomous – a single button toggles the intake, and the **barrel automatically rotates** to the appropriate positions to intake artifacts and detects their color. After loading exactly three, the motor reverses and spits out any surplus. The **launcher automatically starts spinning** up to the appropriate RPM based on the **distance to the goal** and the ramp angle also continuously adjusts.

For the launching itself we have two buttons. One shoots the **optimal permutation** of the loaded artifacts to maximize those in the correct slot and secondarily minimize distance to turn. We also take into account the **number of artifacts currently on the ramp** which is **tracked by the second driver**. The other button shoots no matter the order, taking care only which rotations of the barrel will result in the fastest unloading. Both launch all loaded balls in **rapid succession**, although we do wait until the launcher spins back up after each launch.

Since it is much more intuitive and the driver doesn't have to keep track of the robot's front, we are using **field-relative position**. We have three **driving modes** - "default" (60% motor power), "precise" (20%) for being able to better intake balls and perform alignments in endgame. The "fast" mode gives full power. Additionally, the robot can **autonomously align** itself inside the zone during the endgame.

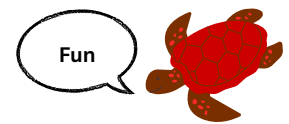
The second controller also has another **layer of bindings** between which you can toggle, which gives complete **manual control** over essentially all parts of the robot.



6.4 Driver feedback

Sometimes the driver loses view of the robot so we need to find other ways to indicate what is happening. We could also want to **inform the driver** of a failure to execute or **warn** that some time period is approaching. For this we opted against using an LED directly on the robot, which could sometimes be obstructed and instead we are using **lights and rumble directly on the controller**, which in the end also proved more intuitive. Specifically, we signal which color artifact we need to intake to complete the pattern, lighting the LEDs that green or purple and any time a command cannot be executed (e.g. trying to intake with a full barrel, shooting with an empty one) we flash red so the driver knows it is not a malfunction. Lastly, from training we know when to empty the ramp to end with a full pattern and when to go to the parking zone to lift which we signal using a rumble and LED effect.

6.5 Code Philosophy and Structure



To keep our **code organised** we are using a structure of **subsystems and commands**, which enables us to create all logic for a certain hardware/software feature once and **reuse it anywhere**, be it an autonomous routine or TeleOp code.

COMMAND: code that can execute logic when it's first run, while it's updating and when it reaches a stop condition. This allows us to mimic the **behavior of multi-threaded hardware access** by simply alternate through their individual update logic.

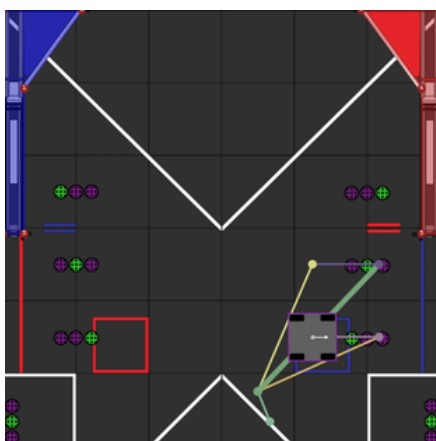
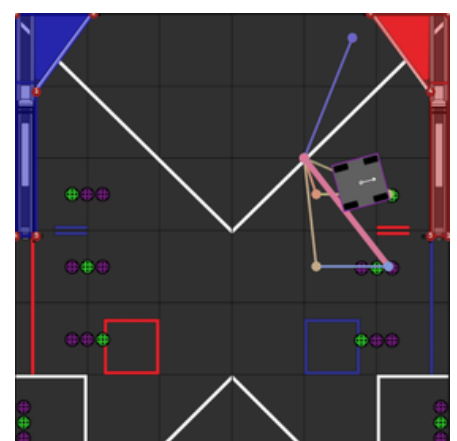
SUBSYSTEM: class that **packages robot hardware** together with simple functions and commands, **abstracting it** from the rest of the robot code and **restricting access** to it. This allows changes to the specific details of the implementation to be isolated and centralized.

Each command "requires" some subsystem and thus the command scheduler can ensure **no two commands accessing the same hardware are run** by interrupting or waiting for completion.

6.6 Autonomous

One of the key objectives of the robot design is **fast indexing** so that our autonomous can focus on filling the ramp with the correct pattern. We are still **compatible with other teams'** strategies, meaning we have the ability to delay starts or emptying the ramp. Furthermore, we try to be **as adaptable as possible** by accounting for when things don't go according to plan. Our autonomous can shoot the pre-loaded magazine and load our magazine an additional two times meaning it can consistently achieve **9 classified artefacts**. We recognize our autonomous **performance could be improved** by intaking from the emptying ramp, sadly we did not have the time to iterate on this as our field is too different from the official.

The biggest source of issues in autonomous robot movement is the lack of human input and error correction. Although Pedro Pathing can adjust for error while moving, it is also necessary to create **safety nets, edge cases and predict failures** that could occur and lead to the robot getting stuck, damaging itself or violating a game rule. We have therefore implemented a wide range of **checks**, mostly deadlines (timeouts that **shut down an action if it is taking too long**) and mechanisms like ejecting balls if one gets stuck in the intake and we detect that it is stalling. A number of **automatic optimizations** are also put in place, generally taking form of cancelling a path the robot is currently following if the action associated with it (intaking, reading an AprilTag) has successfully finished, which **prevents the robot from spending unnecessary time** on movement when that action is already done.



6.7 Localizer and Follower

Localization refers to our ability to constantly know the robot's position on the field, and the follower can use this information to smoothly follow paths. The primary logic for localization has been delegated to the **GoBilda Pinpoint** odometry computer and two **deadwheel encoders**. Path following logic is handled via the **Pedro Pathing library**, which enables us to use the same for both **Bezier Spline following** and **TeleOp** gamepad movement as well as **smooth transitions** between the two.

6.8 April Tag Recognition

Before autonomous starts we have to detect the **april tag on the obelisk**. On initialization the camera **streaming is stopped**, which was a **tough lesson learned** since without doing this it took up all CPU and bandwidth resources and everything ran much slower. Only after long debugging did we consider this could be causing our problem. A command then turns the camera back on and **asynchronously detects** until it identifies IDs of either 21, 22 or 23 after which it sets the appropriate pattern and **closes the camera completely**.



Colour sensor in the barrel

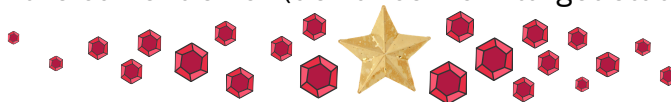
6.9 Indexer Logic and Colour Detection

The main advantage of our system is the indexer, which can **chose to eject any ball** we have loaded. We **store information** about which ball, if any, is in each slot and can quickly **find the closest slot** to minimize rotating. The servo can turn to intake and outtake the three balls as well as turn to a locked position where none can get out.

When intaking we have to recognize that we have successfully gotten the ball into the prepared slot of the indexer and identify the colour. Both of these tasks are done using the **REV colour sensor**. The command waits until the distance measured by the sensor is less than 200mm. That means the ball successfully entered the chamber and we can start with colour detection. To **increase reliability** we don't want to base it off a single measurement since that **could have random error**, so we start an **0.1 second period** during which the hue of the ball is calculated from RGB values and added to the **running average**. At the end of this period we decide whether the **hue angle** is greater or less than 170 and assign the appropriate choice of Green or Purple.

6.10 Closed-loop motor control

Oftentimes it is necessary to precisely control the **position or velocity of motors**. This can be achieved using what is called a closed loop system, meaning the motor is getting **feedback on its current state** and can adjust to errors. A specific controller is called the **PID**, where three coefficients determine how much the current error (deviance from target state), its integral, and its derivative affect the output.



This control is for instance necessary for the **flywheels**, which are able to stay in a **10RPM range** from the target and before each shooting we wait for them to achieve this target velocity. The same goes for the intake, since we want it to be **independent of battery level**.

We use the described way of control in a more inovative way in the lift. One side of it extends faster than the other, dangerously tilting the robot. To **effectively mitigate** this, we are using the **inertial measurement unit** of the control hub. We set both motors to full speed and, based on which way the robot tilts, use the controller not to directly set the power but instead a **decrease from the full power** of the robot causing the disturbance.