Engineering Notebook

Team Slovenia *Vegamind*

22903

Outreach **2**

Build and design process **5**

Integration with REV parts kit **6**

Lifter design **6**
- Angled lifter 6
- PassThrough chassis 7
- Custom telescopic slide 7
- Gripper design 8

Drivetrain **9**

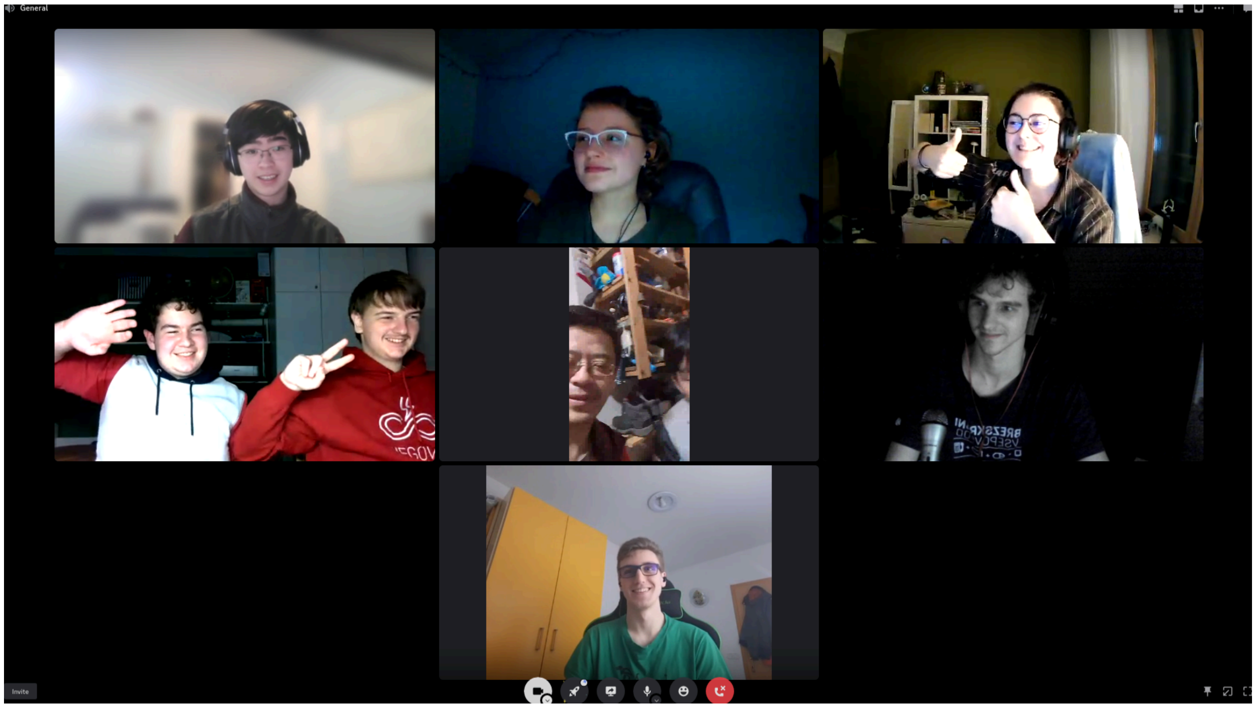
Software **10**
- General software 10
  - Code uploader 10
- TeleOp 10
  - Driving 10
  - Lifting 11
- Autonomous 11

Outreach

Let’s start at the beginning. As a junior team this is our first appearance in *First Tech Challenge* and as such we had to adjust our perception of all FIRST competitions. The previous team was freshly put together for FGC 2022 and had many graduating students for which this is the last year of FIRST competitions–as such we focus on laying a good foundation for our successors. We, the FTC team, have been assembled from a part of the FGC team and a few others that we reached through presentations at school. We figured out this type of recruiting was quite resourceful, so we will continue doing it in the years to come. Next time, we hope to branch out and expand out of our home school.

We also have an Instagram (@first.slovenia) and a website (firstglobal.si).
We also tried getting in touch with other FTC teams from German teams based in Stuttgart FTC Frogs 10183 and Frox 10089 and went to field practice over the weekend there.

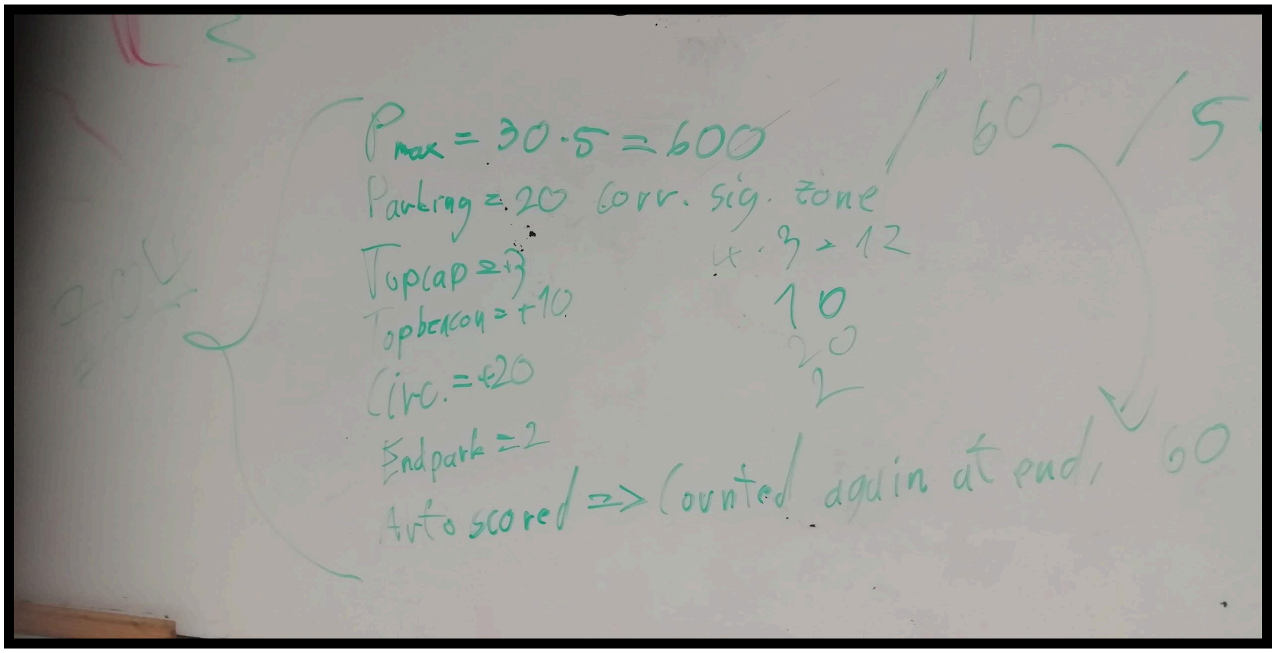
We also got in contact with the Canadian FTC team from **Vancouver, 16031 PARABELLUM** and discussed our designs on Discord.



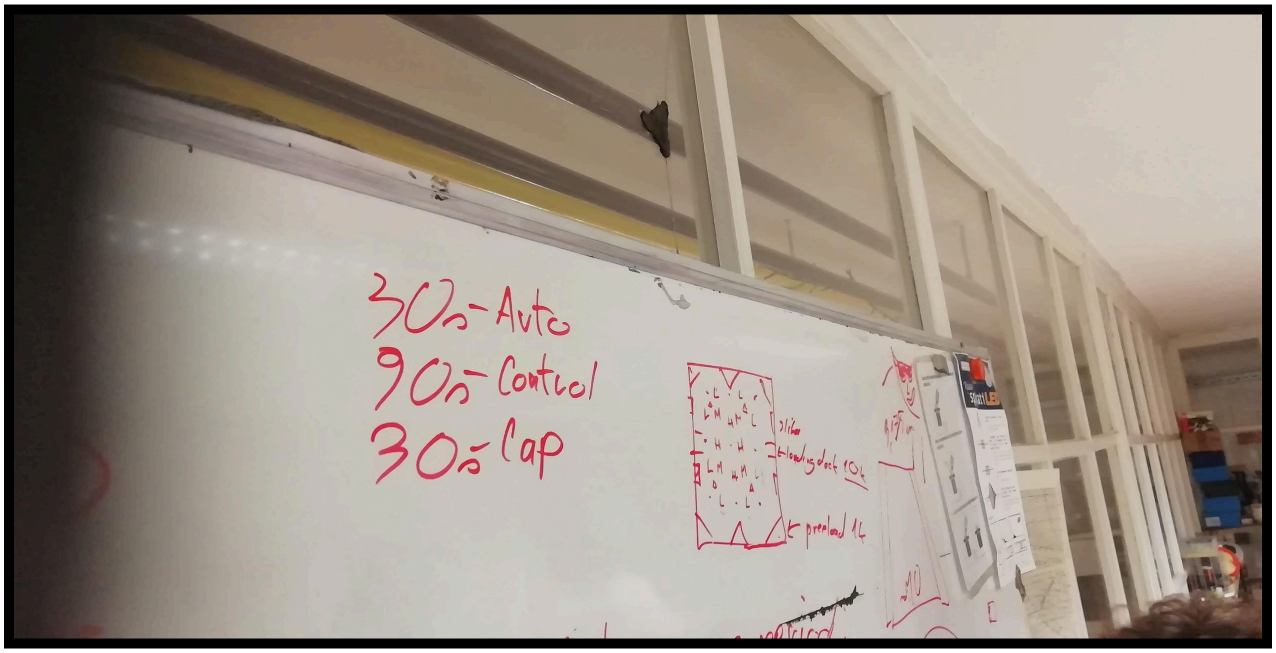
Getting to work

Getting a team together is the easy part. Getting work done, however is much *much* harder, so accordingly we made a plan of action to:

1. analyze the challenge and begin brainstorming ways we would approach it

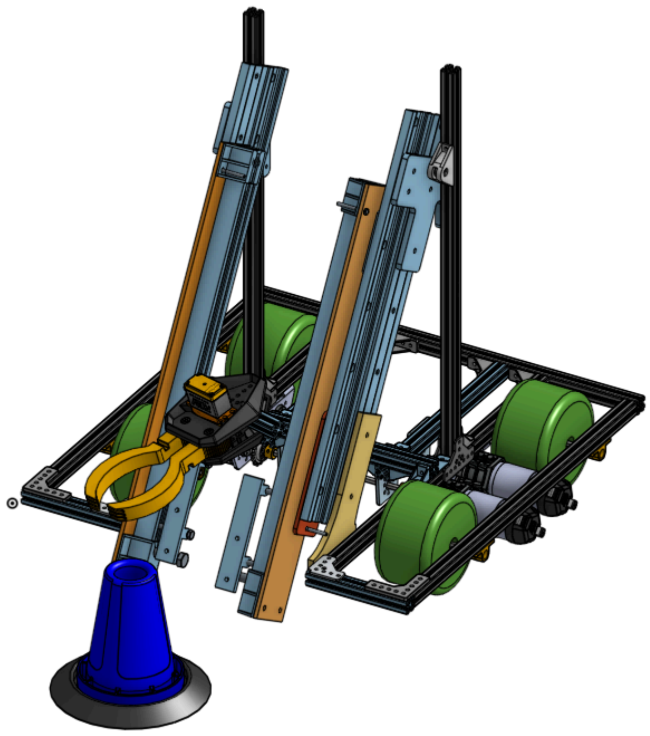

$P_{max} = 30 \cdot 5 = 600$
Parking = 20 Corr. sig. zone
Topcap = +3
Topbeacon = +10
Circ. = +20
Endpark = 2
Auto scored => Counted again at end,
$4 \cdot 3 = 12$
10
20
2
/ 60
/ 5
60

Picture: Calculations of points

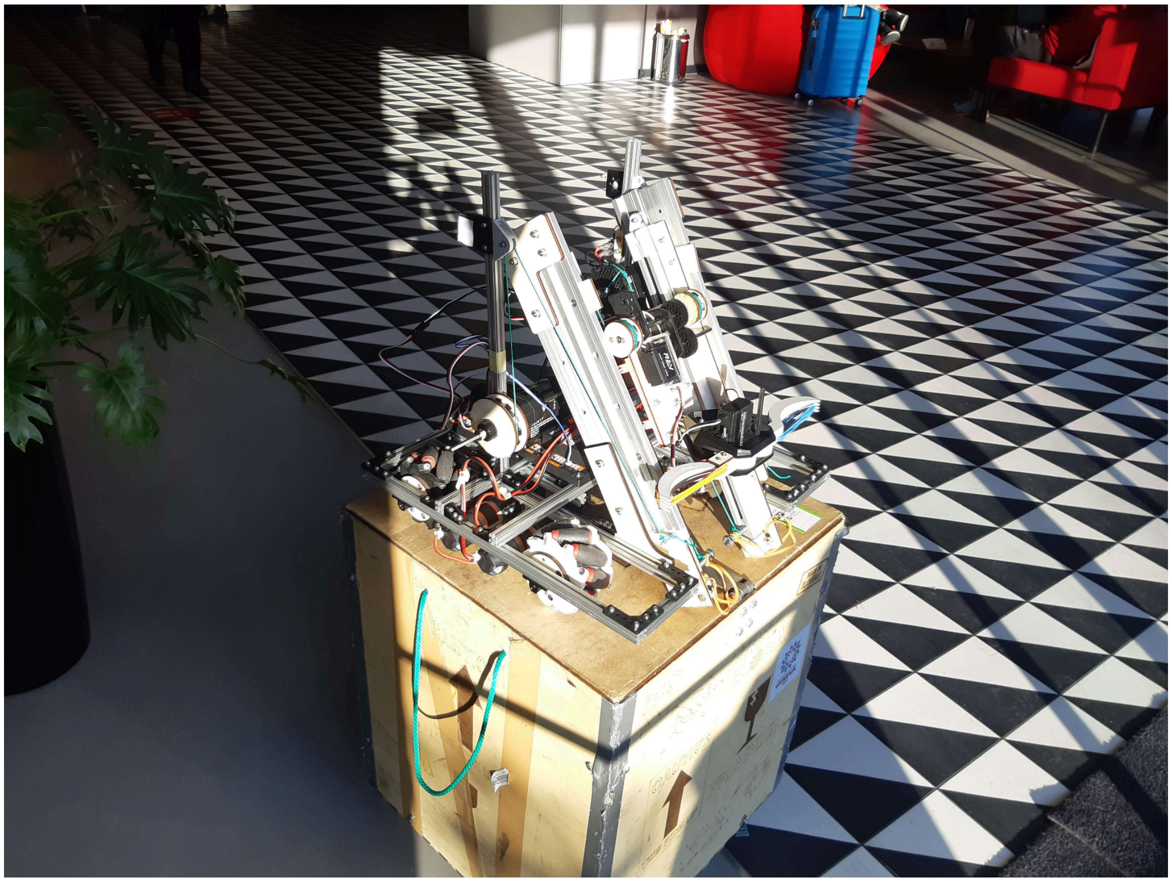


Picture: Time distribution

2. Assign roles like programmers, builders, designers…
3. Get designing



4. Make it happen–build the robot



Build and design process

As a team we came up with many ways to tackle the challenge with the help of mathematics and CAD program **Onshape**. There we tried planning out the entire robot and then building it as a whole. Yet soon we found out that we are too unfamiliar with many of the production processes and switched to physical design solutions where the team members could get a hands-on experience. Next, we tried to parallelly develop the CAD model along with the hardware of the robot. During this stage a lot of disagreements appeared between different parts of our team; mostly designers and builders. With that in mind, in the end we chose to scrap as much as we could from the CAD file and built the robot by vision. We also had to prepare the CAD environment (importing files (REV), fixing models and building examples) and training the team members, which took a lot. As many of senior team members were very familiar with REV parts we designed the robot around those parts.

At our workshop, we had access to a lot of different machinery and tools so it was easy to find what we needed. All designs are based around our low budget as we have little to no income and even less time since many of our team engineers are close to graduation and will soon have to sit their end of the year tests and exams.

Integration with REV parts kit

We designed slides in CAD and cut them on the miller, then assembled them with the adapters and mecanum wheels we 3D printed. We later upgraded the wheels with electrical wire shrinks for better grip. We also redesigned the REV claw assembly to 3D printed claws with rubber bands for better grip and force distribution.
Having developed these separate mechanism REV parts kit, that we salvaged from the 2021 FGC competition, served us well to integrate these mechanisms in a robotic system. Each stage of the lifting mechanism is powered by a motor, the first two stages of a three stage system are powered by one core-hex motor each and the last stage (the lightest and smallest) is powered just by a single servo motor. To bind all the parts and systems together we used laser cut adapters out of wood.

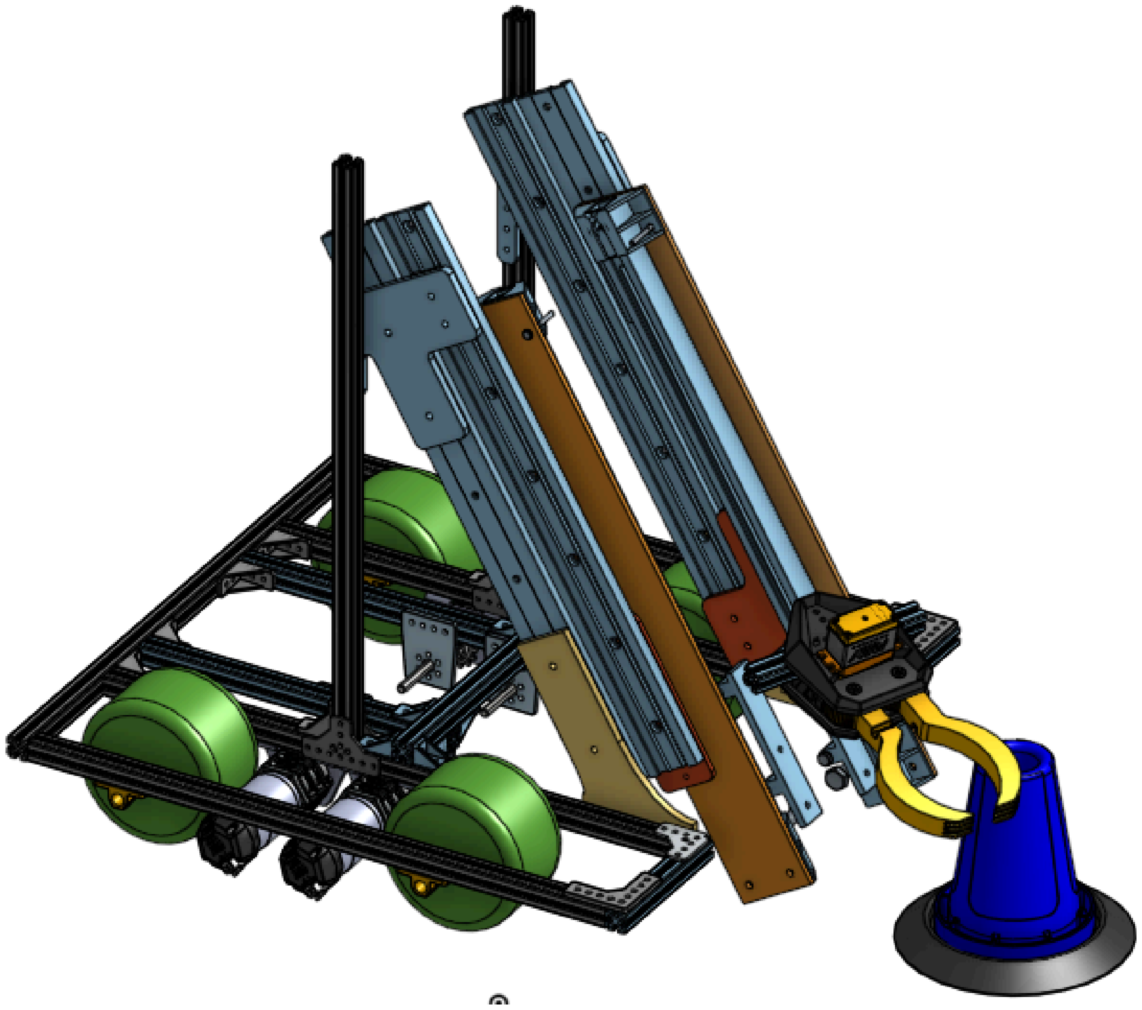
Lifter design

Angled lifter

We iterated through many concepts and built a proof of concept mechanism and decided that we have to decrease the overall complexity of the system, while still keeping all the target parameters in check–like the PassThrough chassis. Achieve the highest height we can, the highest speed and last but not least a simple design consisting of as few movable parts as possible.

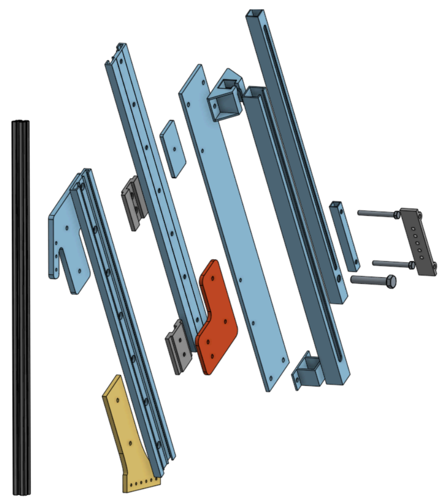
PassThrough chassis

The goal was to pick up the cones on one side and drop them off on the other side.



Custom telescopic slide

We decided to manufacture our custom slides using square aluminum extrusion profiles which slide in one another in three sizes. The extrusion is cut on one side to enable the second and third extrusion to travel outside the first extrusions size limits creating a telescopic linear guide, the third extrusion is essentially a sled that is fully constrained in the second extrusion and has a 3D printed sleeve over it to reduce drag inside the second extrusion. We also 3D printed end caps for the first extrusion and pulley guides that are placed on top of the second extrusion. We laser cut some mounting plates out of wood to secure the slides to the robot. The extrusions' sizes are 20x20x400 (first extrusion), 15x15x400 (second extrusion), 10x10x400 (the sled).



Picture: Slider design (square extrusions, linear guides, and adapters)

This project was more of a proof of concept that it could be done and it had been successfully used as a means of education in design and manufacture of parts. The manufacture of this system cost us less than $10, so it's very cheap, since we mostly modified raw resources. In future we are looking to replace these parts with more shelf solutions.

Gripper design

We 3D printed two cones to test the functionality of our gripper. But first we had to design the gripper. Initially we tried using claw assembly from FGC and added just a view parts like 90 degree bracket and rubber wheels. This wasn't effective enough as the wheels had too small surface area that was actually touching the robot. To combat that we designed our own gripper. Well actually we just had a meeting with one of the Canadian FTC teams from **Vancouver 16031 PARABELLUM** (some of whose members we met on FGC). So we asked them to show us their gripper and it had a claw with rubber bands to get a better grip of the cone so we replicated their design. For the claw to be deemed good it needed to get a good grip of the cone but we also wanted to use least amount of servos as we could, one would be optimal; lightweigthess and simple design were also considered.

Drivetrain

In November our not yet team was invited to observe a friendly competition in Stuttgart, Germany because we made some good friends at FGC. We noticed that all of the robots there used macanum style drivetrain and it made sense because it allows for good maneuverability so that's what we went with. This was probably one of the first decisions we made. And problems just started. Soon we found out that REV mecanum wheels cost 150 USD + about as much for shipping. We decided to try with 3D printing similar wheels we found on Thingiverse as we have fiew Prusas in our school. After noticing that the wheels slip some modification was obviously necessary. The menthor bought us a bag of party balloons to dress the rolles in rubbery material. Sadly it didn't work either because of powder coating on the balloons to prevent sticking or because they are too thin. We finally improved the design by ading heat shrink tubing around rollers.
We have implemented two drive modes. Tank drive and "strafe drive". In tank drive mode both left wheels are combined on a group, same on the right side. Left joystick of the drivers' controller sets the power for left group and right for the right one. In strafe driving mode we only use one joystick that sets the angle at wich our robot moves. To correct for inaccuracies in that we use PID.

Software

General software

Software for our robot is coded in Java as well as our movement libraries. For our development environment, we have used Android Studio along with a custom upload tool.

Code uploader

This tool was made by reverse engineering OnBlockJava. We upload files to Control Hub using an API endpoint and start compilation. We track compilation using websocket endpoint. It saves a lot of time as the recompilation of the Control App is not required. This tool is currently still under development but it will be available as an Android Studio plugin in the future.

TeleOp

Two integral parts of our mission are driving and lifting. Both are represented by individual objects that are instantiated on OpMode initialization.

Driving

Driving was tough to get right since we are using Mechanum wheels. We are using two driving modes. Tank drive and our custom »strafe drive«.

No part could be made perfect though, so this is why we implemented the PID (proportional integral derivative) controller.

Our PID can be broken down into three separate components that affect the correction in different ways.
Proportional term (P): This term is proportional to the error and corrects proportional to the error magnitude.
Integral term (I): This term integrates the error over time and provides a correction for any accumulated error.
Derivative term (D): This term is the derivative of the error and provides a correction based on the rate of change of the error.
We total everything up in the end and adjust the robot's angular position.

Lifting

Our lifting mechanism uses two hex-core motors with encoders and a servo motor. Lifting is done in three stages. For the first two stages, the gamepad is used to set the extension percentage. For the third stage, we use separate buttons to lift the top of the arm precisely.

Autonomous

Since the strafe drive was already implemented we could focus on the computer vision side of things.

The game starts with robots scanning a cone in front of them. Robots must recognize the image and store the result for later. We have decided to use Vuforia. Since we made a custom sleeve for our team we had to train a custom Vuforia model. For Vuforia to recognize custom images we were required to compile a newly created model into the controller app.