Technical Binder

2018-2019

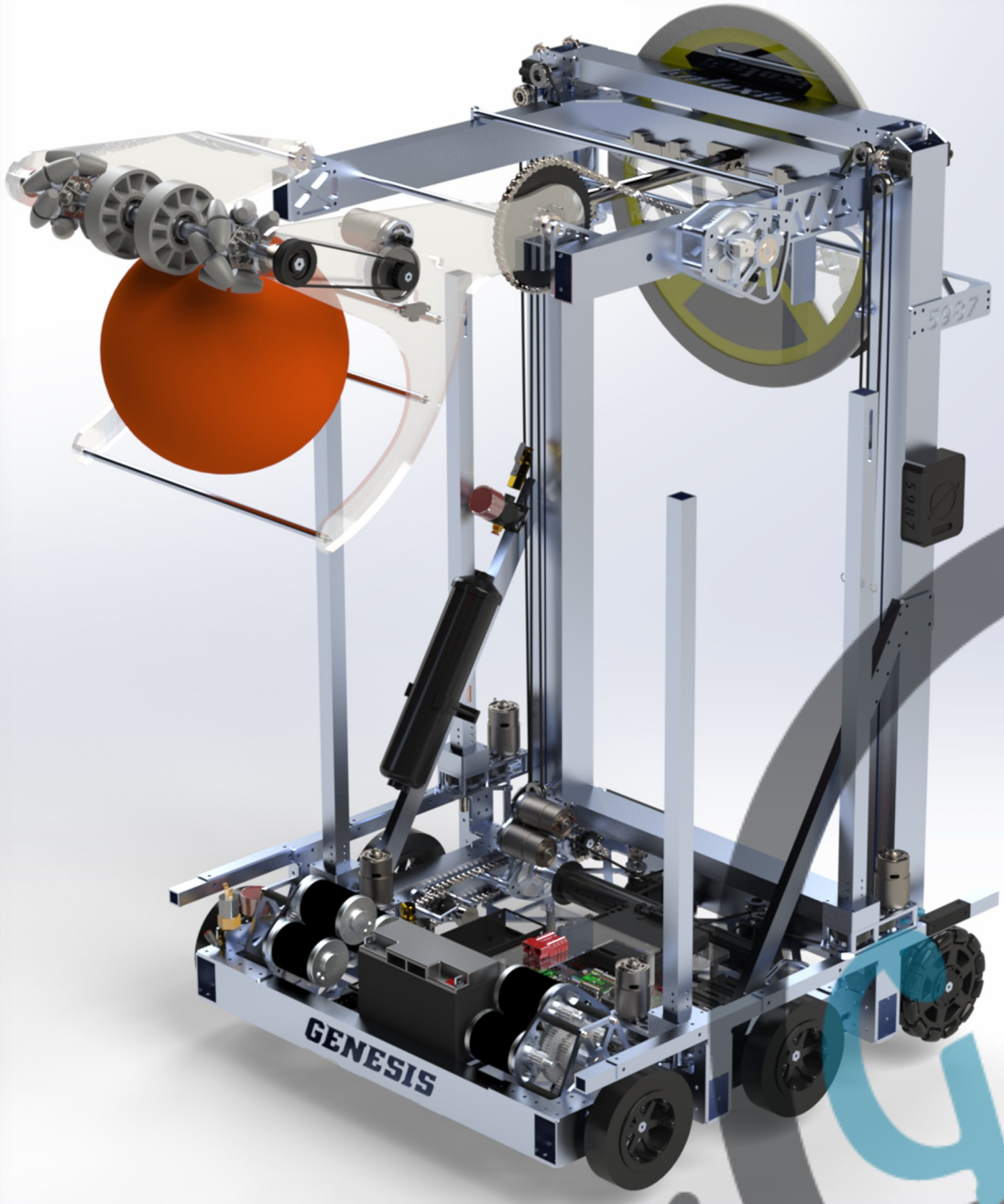


Table of Contents

Design

Vision

Control

Chassis—Tank Drive Base

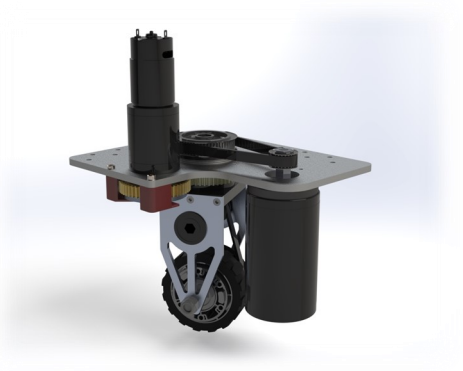
Goals

1. A stable base for a robot with a high center of mass.
2. Sturdiness for defense.
3. Agility for fast maneuvering.
4. Reliability and simplicity.
5. Easy maintenance.
6. Consistent ramp climbing.

Engineering design and prototypes

Swerve:

Swerve is a drive system that uses two motors per wheel: One motor controlling the orientation of the wheel and the second one rotating the wheel itself.



Pros:

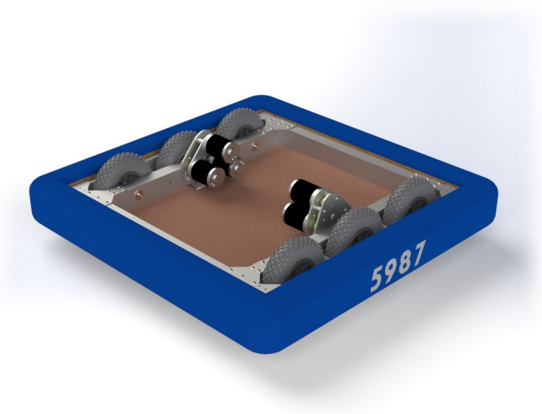
- High maneuverability
- Can strafe
- Can't be pushed easily

Cons:

- Complicated
- Heavy
- Makes the chassis higher

Tank:

Tank is the most common FRC chassis drive. it has 2-4 wheels on each side and a belt or chain connecting the wheels.



Pros:

- Can’t be pushed easily
- Robust, sturdy
- Simple to drive and design

Cons:

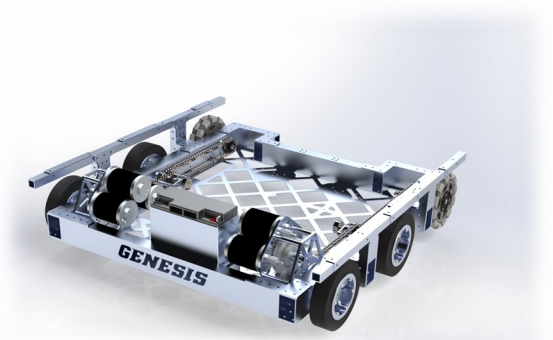
- Can’t strafe

Final Mechanism—Tank

This year we decided to make our own custom made chassis. We chose to make a West Coast Drive, because tubing is our most accessible form of aluminum production. The WCD (West Coast drive) is a standard design among FRC teams.
On the off-season we used 8” pneumatic wheels for off road driving.
We decided not to make the octocanum and the swerve prototypes because of time and cost problems.

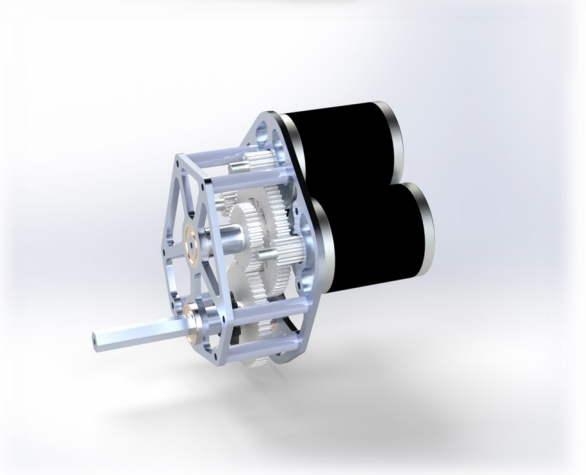
Prototype conclusions:

- WCD works better than the Kit of Parts chassis (AndyMark chassis) and is
- simpler to build and maintain.
- E4T encoders have various connectivity issues and break easily.
- 3 mini CIMs are better than 2 CIMs and are worth the space.



Mechanism breakdown:

- Four 6” pneumatic wheels for falling off ramp.
- Two 6” omni wheels for easier turning.
- #35 chain for driving all the wheels.
- Maximum speed of 5.1 m/s.
- 3 mini CIM gearbox for each side.
- 63R encoder on the middle wheel axis.



Gearbox:

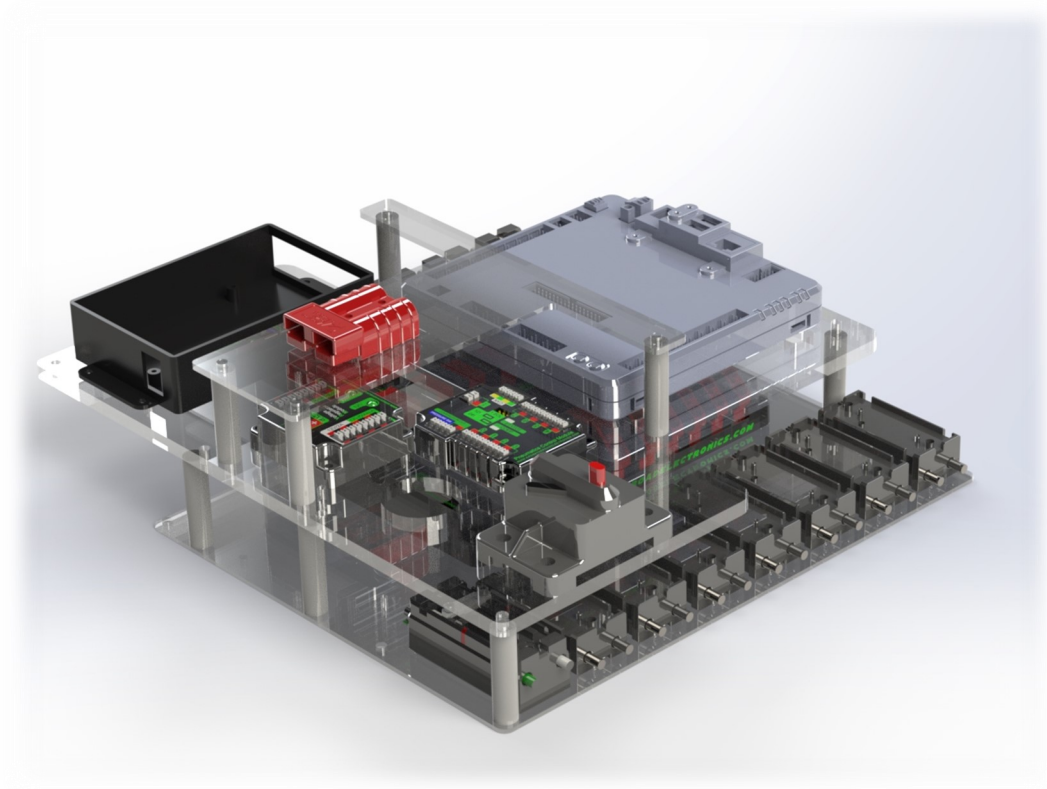
We have decided to use 3 mini-CIMs configuration instead of the more common 2 CIMs configuration, because mini-CIMs cool faster than CIMs, and when the heat expands the bushings, in the CIM it affects the CIM’s efficiency a lot more than the bearings in the mini-CIM; so, to prepare for long competition days, we chose the 3 mini-CIMs.

Bellypan:

The Bellypan has 2 goals: 1) Give structural rigidity to the chassis, and 2) Help connect the mechanisms on the bellypan. Since at first we wanted to connect the electronics onto the bellypan, we needed material that was rigid and light. We tried Pertinax, which is extremely thin plywood held together by epoxy, commonly used for industrial electronic boards as it is a very bad electricity and heat conductor. However, it was quite heavy and not rigid enough, as some of the pulleys that hold the lift that were connected straight to the bellypan broke it. After that incident, we swapped the bellypan to be out of 3mm aluminum 6061-t6, and designed a new electronics board that is not connected immediately to the bellypan so the electriconics board doesn't touch the electrically-conducting aluminum.

Electronics board:

In the off-season, we wanted to make the electronics board an integral part of the bellypan, by making it be the bellypan itself. However, after we realized that finishing the electronics board before the integration into the skeleton of the robot could save us a lot of time, we decided to redesign the entire thing into a different, 3-levels high electronics board out of 4mm polycarbonate.



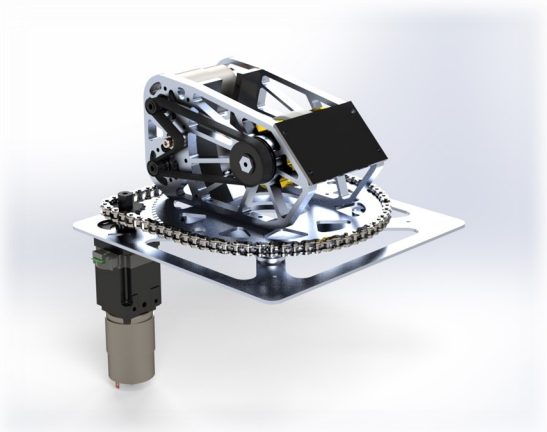
Lifting—Continuous Elevator

Goals

1. Placement of Cargo and Hatch panels in the Cargo ship and in all levels of the rocket.
2. Scoring from both sides (forwards and backwards).
3. A low center of mass.

Literature Review

Shooter:



Pros:

- Allows for a very low center of mass.
- Lightest lifting mechanism.
- Allows scoring from all points of the field.

Cons:

- Requires high degrees of precision- Cargo is 13” in diameter, Port is 16”.
- Requires a lot of prototyping and iteration.
- Requires a different mechanism for hatch panels.

Multi-jointed Arm:

Pros:

- Allows the easiest scoring from both sides.
- Simplest and most reliable.

Cons:

- Harder to control (both for drivers and software).
- Constantly shifts the center of gravity.

Cascade/Continuous Elevator:

Pros:

- Fastest scoring.
- Easiest to control and program.
- Most experience with design (2018).

Cons:

Final Mechanism—Double Stage Wristed Continuous Elevator

Elevator stages:

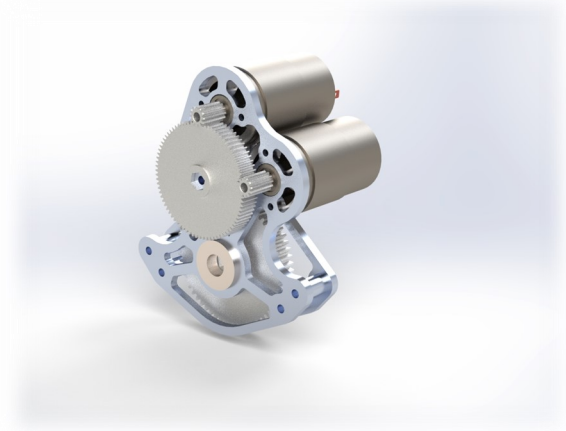
- Doesn't exceed 1.2m when stowed.
- 2 elevator stages to reach 3rd level Rocket and 5x16x5 mm bearings press against outer face of the tubings and slide off the elevator tubes.
- Bearing blocks were initially 3D printed but swapped out for aluminum.

Continuous cable rigging:

- Cascade was preferred initially, but the geometry didn't allow for backward cargo shooting.
- Cables were preferred for their very light weight, as demonstrated in previous robots (2018).
- 3D printed pulleys fit on the side of the elevator to avoid cable collision with Cargo in backwards shooting.
- Horizontal and diagonal system of pulleys brings the cables to the side of the robot from the central winch. Cable tensioners fit on both pull-ups to keep both sides equally tensioned.

2 775 RedLine Flipped Elevator Gearbox:

- 20:1 reduction Gear ratio reduction.
- 3.6V stall voltage allows motors to stall without burning.
- Max speed of up to 3 m/s.
- Two 3D printed 45 mm diameter PLA winches with Delrin core for extra strength.
- 4 tapped holes along the winch to allow iteration on fleet angle.
- A thread winch was originally used, but was proven useless as the strong tension forces in horizontal rigging caused the cable to skip threads.
- SRX Mag encoder on winch axis.



Cargo Intake—Horizontal Roller wrist

Goals

1. Quick Cargo grabbing.
2. Tight holding of the Cargo, so it won't fall when facing defense.
3. Moving out from/in to the robot's frame perimeter.
4. Simplicity and easy fixing.

Engineering Design and Prototype

3 fingers arm:

A 3 fingers arm is a robotic arm with 3 fingers that grabs the Cargo similarly to a human arm.

Pros:

- Helps center game pieces to ensure the Hatches' stability when attached.

Cons:

- Uncommon in FIRST.
- Slow.
- Complicated.
- Takes a lot of space.

Horizontal roller:

Spins and grabs Cargo. Can be with/without wheels.

Pros:

- Fast.
- Easy to design, build, and maintain.
- Reliable.

Cons:

- No centering ability.
- Hard to use by drivers.
- Weight.

Rubber bands:

Rubber bands move into the robot for intake.

Pros:

- Fast.

Cons:

- Harder to use for the drivers.
- A lot of points of failure.

Final Mechanism—Horizontal Roller with Wheels

Prototype conclusions—Horizontal roller:

This mechanism grabbed very quickly and efficiently and did answer all of our goals. Prototyped with the right compression to grab all Cargo within given tolerance.

Mechanism breakdown:

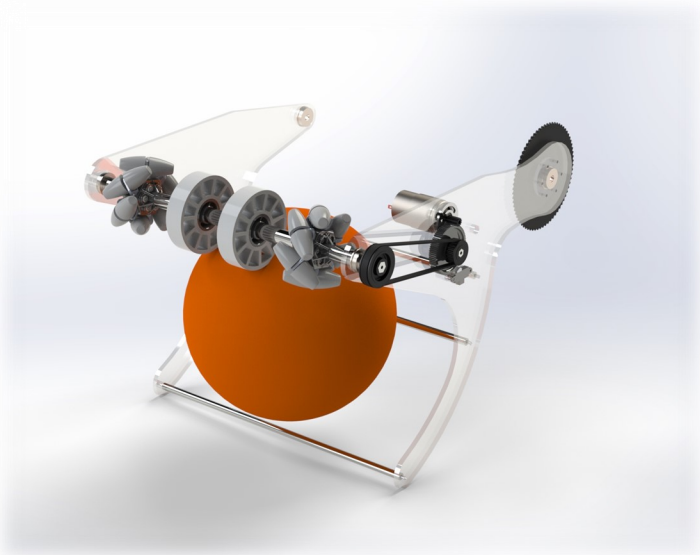
The mechanism goes inside the frame perimeter in 2 situations:

1. At the beginning of a match.
2. Backwards shooting.

Uses 4 wheels in total: Two 4” mechanum and two 4” compliant.

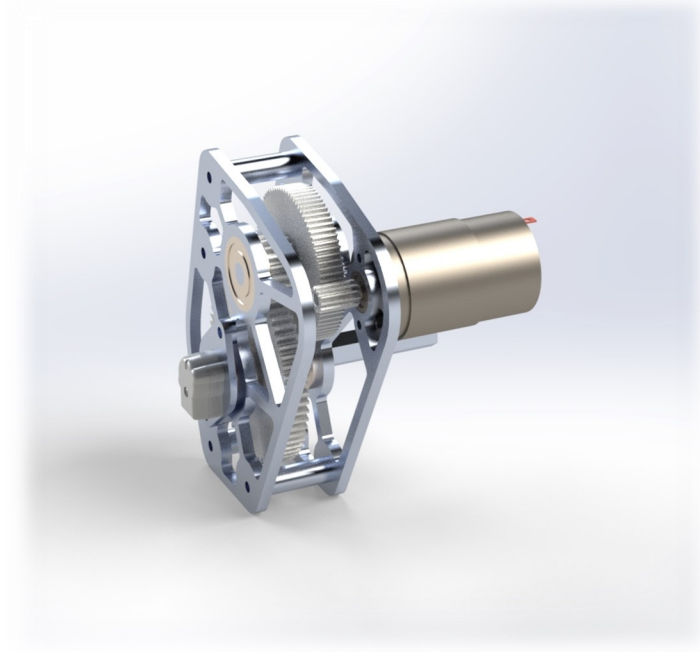
Very simple and easy to change, fix or replace.

Curved inwards on the sides to grab Cargo from wider angles.



Wrist gearbox:

Made to retract the whole intake inside the robot's frame perimeter at the beginning of the match, as well as help to shoot the cargo backwards. This gearbox had to change the angle of the intake with a cargo inside it. Because of that, we wanted the gearbox to be strong as well as fast. We used one 775pro motor with a 88:1 ratio to achieve the best speed:torque ratio.



Hatch Grabber—Latching Fingers

Goals

1. Grabbing of Hatches without dropping them on the floor.
2. Low precision required for alignment on the loading station.
3. Extention outwards for easier Hatch placing.
4. Light weight.

Engineering Design and Prototype

Pivoting arm:

Rotating arm that can put Hatches on the Bay.

Pros:

- Can grab from the floor.

Cons:

- Heavy.

Hook and loop tape:

Using a weaker hook and loop tape than those on the Rocket and Cargo Ship to grab and hold Hatch panels.

Pros:

- Easy.
- Light.
- Requires few actuations.

Cons:

- Requires alignment.
- Regular wear—The hook and loop tape on the competition field might not be changed, causing us to be unable to score or unable to grab.
- Unreliable under certain loads and speeds of the lift.

Latching fingers:

Fingers that grab the hatch panel from the inside.

Pros:

- No chance of hatch panel falling.
- Requires low alignment.

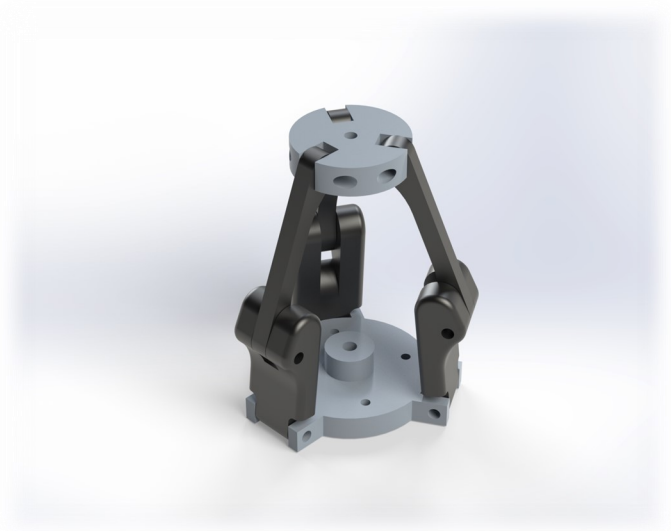
Cons:

- Heavy.
- Requires many actuations.

Final Mechanism—Latching Fingers

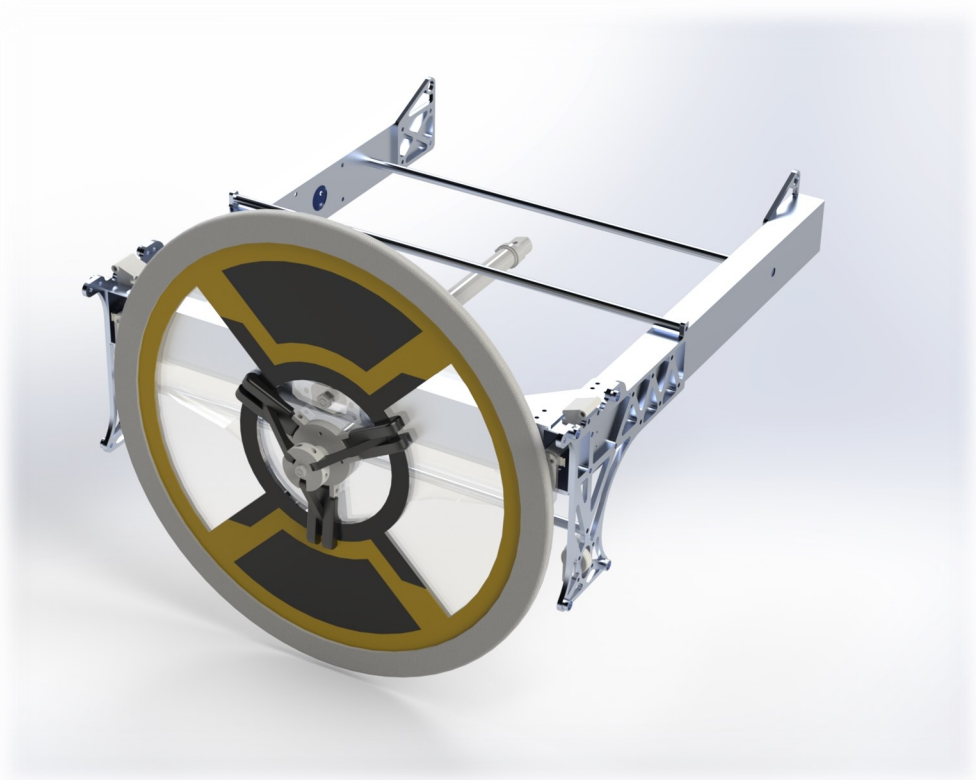
Fingers:

- Fingers powered by a pneumatic piston.
- Head and back of the fingers are 3D printed out of PLA.
- Fingers and pivot points printed out of rubber for extra gripping and for avoiding any breaking from unexpected loads.



Back plate:

- Back plate out of polycarbonate for equal force distribution between all 3 fingers.
- Back plate can extend to 250mm with a pneumatic piston.
- Custom rails made out of a 3D printed block and a Delrin rod connected to the back plate, to avoid rotation on the piston.
- There was an attempt to use rails, however, our tolerances weren't good enough and broke them.



Climbing Mechanism—Lead Screws

Goals

1. Climbing in a short duration of time.
2. Reliability, no danger to the robot.
3. No requirement of a very specific placing of the robot in front of the Hab.
4. Operation on low voltage, as the climb is at the end of the game.

Engineering designs

Prototypes

Four stretcher-like legs:

Four legs hidden in the chassis that would be pushed open by four linear motion systems.

Pros:

- Compact – fits inside the chassis.
- Fast.

Cons:

- Requires four, very strong linear motion systems.
- Unstable during some parts of the climb.

Two linear motion systems with motored wheels on the front:

Two linear motion systems at the back of the robot, with motored wheels mounted on an arm at the front. The robot gets pushed from the back, drives on the wall with the wheels and then slides onto the platform.

Pros:

- Low weight.
- Fast.

Cons:

- Unreliable, the arm can slide.

Parallelogram climb:

A bunch of profiles that form a parallelogram.

Pros:

- Compact – fits inside the chassis.
- Fast.

Cons:

- Requires precise placing of the robot.
- Very complicated.
- Needs to be modeled alongside the chassis and is hard to make changes to.

Four linear motion systems climb:

Four linear systems on the four corners of the robot that lift the robot up. The robot is then driven forward by a propulsion system that is mounted at the bottom of the linear motion system.

Pros:

- Fast.
- Stable.
- Does not require specific placement of the robot.
- Does not endanger the robot, as in the case of power loss it will just go back down.

Linear Motion System

Pneumatic pistons:

Extending pistons based on air pressure.

Pros:

- Easy to get.
- Comes in a variety of strength and strokes.
- Does not require electricity – no power loss danger.

Cons:

- Only has two states: fully extended or fully retracted.
- Uses air pressure, which is also used by other systems on the robot, meaning that the pressure in the tank may go too low for the robot to use the system.
- Weighs a lot compare to alternatives.
- Unstable due to only having two states.

Mechanical pistons (lead screws):

Extending pistons based on lead screws powered by a motor.

Pros:

- When using trapezoidal lead screws, the screws self-lock, meaning they don't move unless the motor moves, so in case of power loss the robot won't fall down.
- Doesn't need a big gearbox as the screw itself does most of the RPM to torque conversion .
- Takes minimal room on the robot.
- Low weight compare to alternatives.

Cons:

- Hard to get (but possible).
- Requires a motor.
- Comes in specific strengths and strokes.
- Not common in FIRST.

Pull down with ropes:

A rope connecting between the winch and the top of the profile, that when the winch spins gets pulled down and lifts up the robot.

Pros:

- Very common in first.
- Requires a motor.

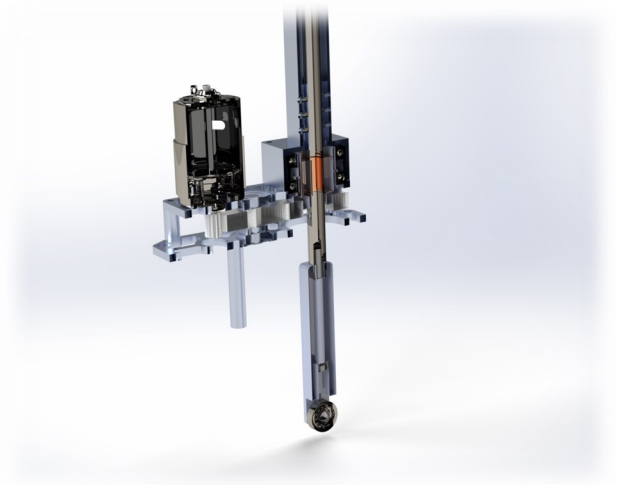
Cons:

- Weighs a lot compare to alternatives.
- Takes up a lot of room on the robot.

Final Mechanism: Climbing

Climbing:

- 775pro propels each leg with a ratio of 5:1 before leadscrew.
- 4 lead screws with 12mm diameter and 6mm pitch, each can hold up to 1000N of force.
- Custom made aluminum blocks to hold the nut.
- Can climb up the ramp in 1.3 seconds.
- Adapted gear to move the nut. Gearboxes connected to bumper holders and main profiles of drivetrain.

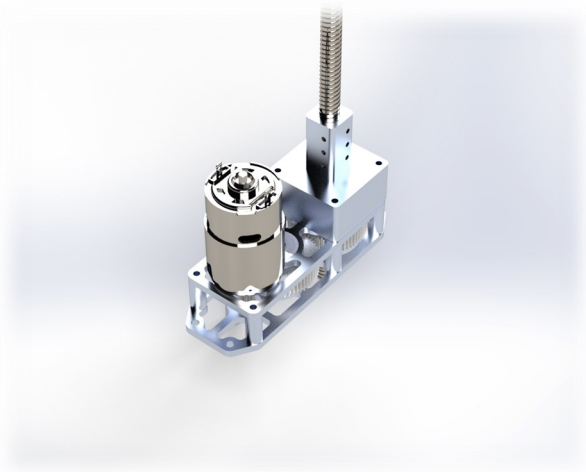


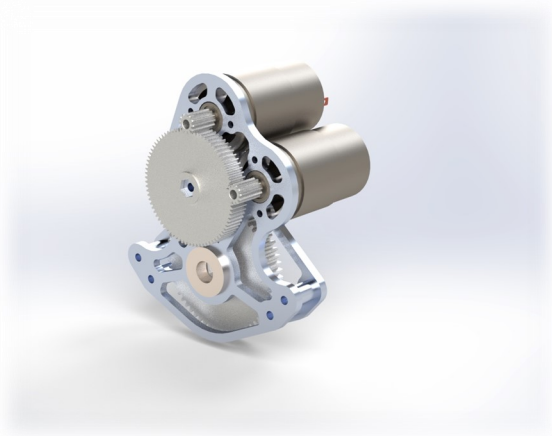
Motion Forward:

- 80:1 sports planetary gearbox driven by a 775 pro.
- Shaft in profile to drive both sides at the same time.
- Custom 3D printed gears to drive both sides.

Custom gearboxes:

We started building our own custom gearboxes in 2018 with the help of our mentors. designing your own gearboxes has a lot of benefits, with very few disadvantages. Custom gearboxes are cheaper, lighter, more robust and are more easily maintainable than bought gearboxes; also, with the AMB design calculator that our mentors made for us and for the rest of FRC, we can design our gearboxes to be faster and fit the mechanism's needs, stay high on the efficiency curve and not ruin our motors. On the robot, we have 8 custom gearboxes and 1 sports planetary gearbox.





Mechanism Gear Ratio Calculator

775pro/Redline		Adjusted
Free Speed (RPM)	18730	18730
Stall Torque (N*m)	0.71	0.57
Stall Current (A)	134	134.0
Free Current (A)	0.7	0.7
Power (W)	348.1	278.5
# Motors	1	
Voltage Applied (V)	12	
Gearbox Efficiency	80%	
Radius (in)	2	
Load (lbs)	3	

	By Ratio	Max. Power	Max. Efficienc	At Stall	Rot. Speed	Lin. Speed	Current	Stall Load
Gear Ratio	7.20:1	2.39:1	17.71:1	1.19:1	102.85:1	162.25:1	5.24:1	7.96:1
Rot. Free Speed (rps)	43.36	130.76	17.63	261.52	3.04	1.92	59.60	39.23
Rot. Loaded Speed (rps)	36.17	65.38	16.44	0.00	3.00	1.91	46.02	33.34
Lin. Free Speed (ft/s)	45.40	136.93	18.46	273.86	3.18	2.01	62.41	41.08
Lin. Loaded Speed (ft/s)	37.88	68.47	17.22	0.00	3.14	2.00	48.19	34.92
Current Per Motor (A)	18.4	54.0	7.9	107.3	1.9	1.5	25.0	16.7
Stall Load (lbs)	18.1	6.0	44.5	3.0	258.5	407.8	13.2	20.0
Stall Voltage (V)	2.0	6.0	0.8	12.0	0.1	0.1	2.7	1.8

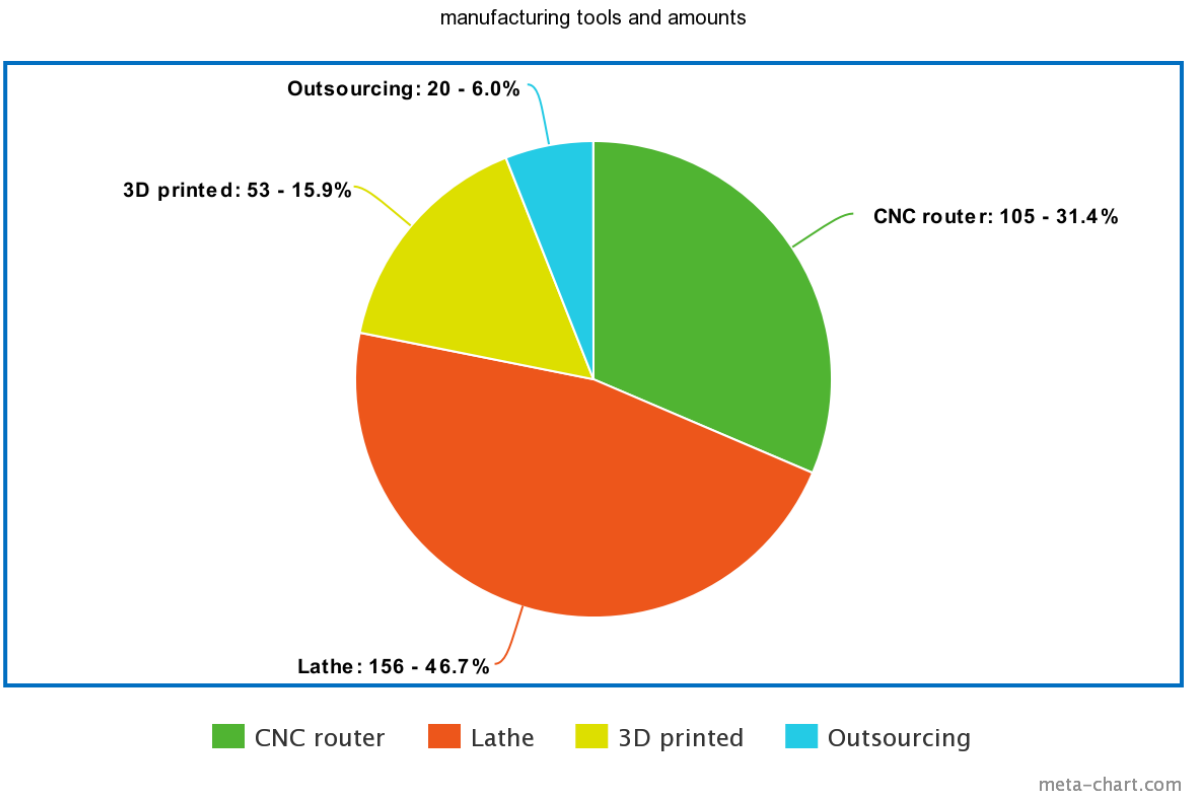
Selected values should be between the associated Max. Power and Max. Efficiency values to keep the motor in a safe place on its curve

EXAMPLE USES:

Winch: As an endgame, you need to winch up the robot by spinning a cable around a circular drum. For radius, enter the drum's radius. For load, enter the weight

Manufacturing tools:

This year, our team got a new 1.2x1.2 meters CNC router and a new 3D printer, so we expanded our manufacturing tools options, and increased our precision and speed, thanks to the router we were able to manufacture almost exclusively in house.



Vision

RealSense

The RealSense D435 camera is an Intel depth camera. Unlike most cameras used in FRC robots that provide a two-dimensional color image (i.e., without any distance information from the objects in the picture), the RealSense camera additionally provides a 3D black and white image that corresponds to the color image. The RealSense camera calculates the distance from each pixel in the image by using an IR beam projector and a pair of adjacent stereo cameras. Because there are two cameras, there's a slight difference between the two images, Using this difference the RealSense camera utilizes a trigonometric calculation to calculate the distance from each pixel.

Device	RealSense	Microsoft LifeCam HD-3000
Pros	Accurate distance measurement, small, High image quality	Cheap, small, easy to use, used in previous years
Cons	Very expensive, Requires relatively strong processing power, Harder to use	Mediocre image quality, no additional measurements

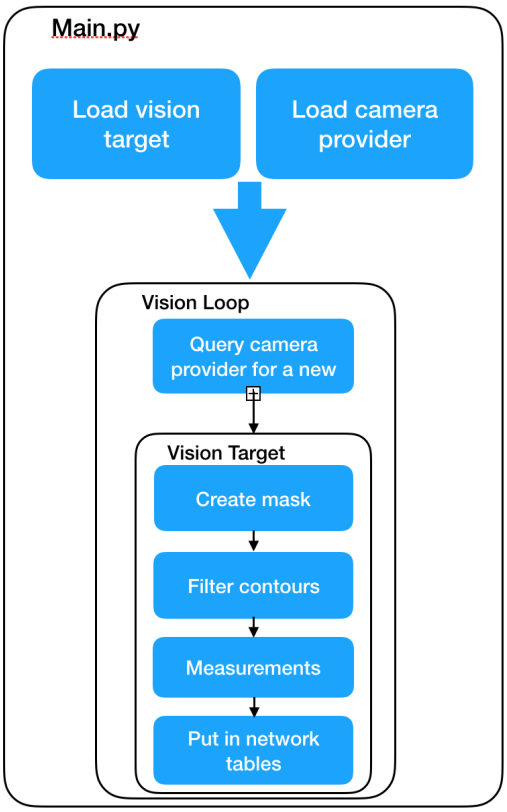
We chose to use RealSense because the advantage of accurate distance measurement is necessary in order to autonomously align the robot with targets and game pieces quickly and accurately. Additionally, specifically this year it's much harder to measure accurate distance from game pieces using traditional ways due to the nature of the objects, the RealSense easily solved this problem.
With the benefits of the RealSense camera it also comes with many problems we had to deal with, such as the need for strong processing power, the difficulty of operating the two cameras simultaneously, and working with the camera as opposed to a webcam. During the season we were able to overcome all the problems through many attempts, joint thinking and reading articles on the internet. After learning how to use the camera, we started working with two cameras simultaneously, improving camera accuracy and improving code performance when using the cameras.

On our robot there are two RealSense cameras facing two different directions. The first is directed towards the robot's Hatch Panel system and is responsible for detecting Hatch Panels on the floor and light reflectors of both game pieces. The second camera is directed in the other direction and is responsible for the identification of Cargo and the identification of Cargo light reflectors. In this way the robot can reach any target without driver assistance.

Framework:

We noticed a common trait about past years Vision code and that was that it wasn't very well organized making it hard to understand and there was also a lot of repetition. So during pre-season we decided we wanted to change that and after putting some thought into it we decided to build a modern modular framework for vision that will make it much easier for us to create vision targets.

Modern modular design usually includes a static main file that loads other files that can be modularly created, in our case each file was a vision target with pre-set functions that could be overridden for every target, this allows for customizability and simplicity at the same time while being simple to use. You can choose to override only specific functions and leave the rest default which works for most vision targets.



Our idea:

This design made it very easy to add new targets and made sure we didn't have to repeat any code because all we had to do is create a new vision target instead of wasting time on everything around it.

Neural Network:

Up until this year we used classic detection for every target, this year during pre-season we decided to test out a new and modern way to do object detection - neural networks.

Neural networks are inspired in their design by the neural networks that constitute human and animal brains.

Neural networks are computing systems, that try to simulate the brain with a mathematical model that takes a lot of inputs, put them through a very complicated mathematical expression, repeat a few times and lets out a series of outputs, the job of neural networks is to find extremely complicated patterns in seamlessly patternless things.

We started by researching everything revolving deep learning and neural networks for object detection and found that there are many different technologies and ways to go about it.

Neural network VS classic detection for game pieces comparison:

Criteria	Classic detection	Neural networks
Ability to detect complex objects	Could be virtually impossible to get consistent results	Not a problem due to its learning method
Speed	Usually fast, depends on complexity	Fairly fast
Accuracy	Usually not accurate	Very accurate
Calibration	Requires color calibration	None required
Complexity	Fairly simple to implement	Harder to implement
Reusability	Can't be reused, object specific	Can be reused for new objects
Lighting variability	Very susceptible	Usually not susceptible

Types of Object Detection Networks

Segmentation:

Provides you with specific contour and separates image into segments

Pros:

- Provides you with easy to work with and more specific information.

Cons:

- Very heavy, requires a lot of processing power.
- Requires more data.
- Much harder to label.

Bounding box:

Outputs a bounding rectangle that includes parts that aren't the actual object

Pros:

- Relatively fast.
- Accurate.
- Easy to label.

Cons:

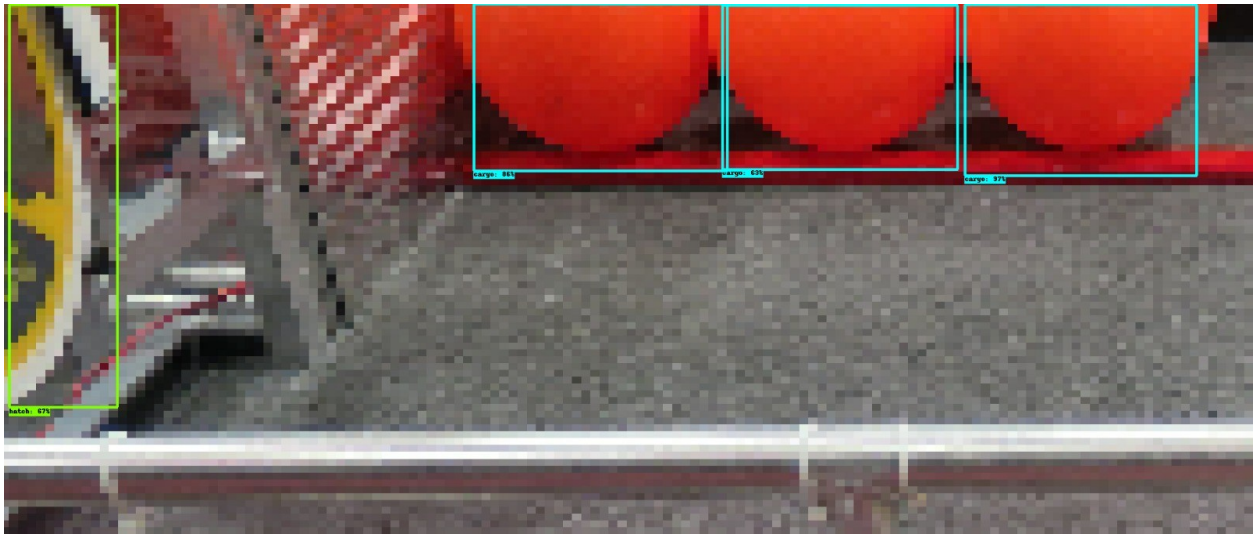
- Hard to work with and not a specific bounding box.

We decided to use a bounding box network due to speed and the fact that we don't necessarily need the specific contour and rather a bounding box suffices.

Dataset:

The dataset is the input of the neural network, we took a few thousand photos of the object we wanted to detect and labeled them (selected the object in every photo using a bounding rectangle).

After you've created your dataset you need to train the neural network, we used a pre-configured model that was already trained on the COCO dataset which is a huge dataset containing around 300 thousand labeled images. By using this we improved training time and accuracy because the network already knows what objects are and now only has to learn one more.
In order to train effectively you need a strong computer and usually one with a strong graphics processor, we used a server that had a Quadro P100.
During training, we used tensorboard which visualized all the data so we could use it effectively and know what to change between training sessions, we displayed images that it was inferring every so often during training and data about accuracy.



Neural Networks for Cargo detection—a complex solution for a simpler problem.

While attempting to create a neural network for cargo detection we realized that were much better off making simpler detection to it, this is because the cargo is very easy to distinguish from other targets due to its color and shape. Sometimes looking back at what you're doing goes a long way.

Results:

The results were very surprising, the neural network was able to infer in real time at around 10fps on the Jetson co-processor. The network was also able to detect a partially obscured hatch and a hatch that was very harshly exposed, including when the which we noticed happened quite often.

Cargo:

As cargo is a very simple and distinct target, recognizing it with traditional image recognition is easier and more efficient.
Using OpenCV, we acquired the contours and first filtered them out based on the cargo's unique orange color and the contour's solidity, then eliminated any inner contours created by reflections, and finally applied edge detection to separate one cargo from another.

Reflection Tape:

The reflecting tape targets this year were unique in the fact that each correct pair consisted of two (2) tape pieces angled inwards (i. e., towards each other), which was useful for recognizing and telling apart the targets. We first detected them individually and then used their rotation in relation to x in order to pair them.
The measurements required for control were taken from the middle point between the two, meaning the distance was the average of the distances from their centers.

Bay Chooser:

We decided to create an algorithm that chooses a bay (i. e., tape target) to go to based on the game piece attached to it and the game piece the robot is carrying. This will aid drivers and at the sandstorm period since we are able to know which cargo bay to drive to automatically.
We recognize the tape targets and the pieces present using neural networks and classic detection, after that we check if they're in the correct threshold of coordinates and choose depending on which game piece is currently on the robot.
After that we were able to send the measurements of the chosen target back to the NetworkTables for control.

Camera Offset:

This year since the cameras weren't centered, we had to compensate the center angle and distance so that the robot can center the target correctly instead of centering in relation to the camera. We did this using some basic trigonometry.

Jetson:

The NVIDIA Jetson is an embedded computing device designed to provide performance while being small and power efficient. It is currently considered the best and most powerful co-processor for FRC teams among many other options that we considered:

Device	Raspberry Pi	Intel UP board	NVIDIA Jetson
Pros	Cheap, easy to use	Fairly high performance, easy to use	Very high performance, GPU
Cons	Very low performance	Less information, medium price	High cost, hard to set up, requires

We first received an NVIDIA Jetson TX1 development kit through FIRST Choice so we started exploring Its capabilities and working with it. later on we purchased an additional Jetson TX2 which has about twice the performance of the TX1, It was nice to know that the performance definitely won't be a limiting factor for us this year.

Powering the Jetson:

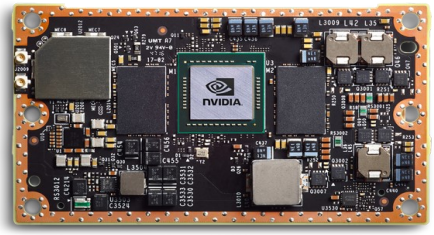
The Jetson cannot be connected to the normal VRM modules and instead has to be powered using an external buck booster.

Results:

The Jetson performed very well, on the TX2 we were able to run classic detection and a neural network while pulling frames from both RealSense cameras.

Issues:

- The Jetson TX1/TX2 comes with a development kit that includes a lot of IO and options but is very large physically. There are smaller carrier boards that you can purchase to fit the Jetson module onto a smaller footprint, after some research we found that the Auvedia J120 is the most viable option.
- Operating the Jetson module requires moderate knowledge of embedded systems and linux.



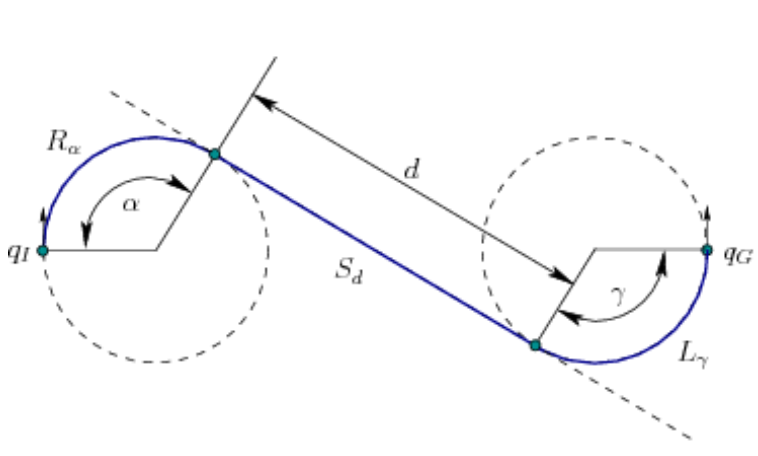
Autonomous Pathing and Path Pursuit

Starting from the pre-season we decided to try and use a pure pursuit controller for this year's autonomous drive. We based our algorithm on Team 1712's paper, which takes on the controller as the main source for our program. We spent the next few weeks learning, writing and improving the controller to meet our needs and expectations and by the end of the pre-season we had it almost completely sorted out with a few things we left for the time when we get our new robot for the 2019 season.

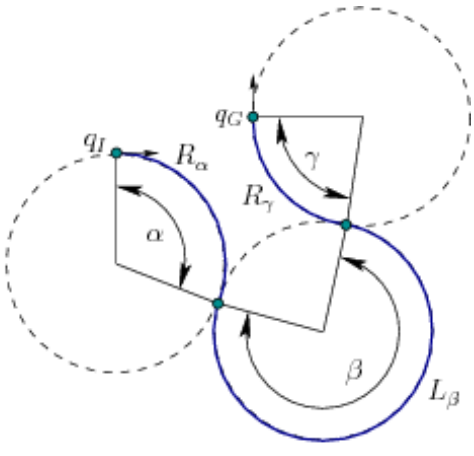
Once the season kicked off and we got our mission, we realized that the current path generation wouldn't help us in this year's game. Destination: deep Space requires a precise angle to be able to place the game pieces. Our current program was only able to reach a certain point without any consideration for the ending heading. We realized we have to find a different way to generate a path that is both effective and finishes at the right angle that's when we came across Dubins path. Dubins path is a method to find the shortest curve between two vectors in a two dimensional plane in consideration that the vehicle can only move forward. Dubins Path also takes into account the maximal curvature , The initial and terminal headings.

After a week or two of studying and writing the code for the method we had been able to use this algorithm and our pure pursuit controller to reach a certain point and angle on the field.

We thought we had everything ready for the new robot and so when we got it from Engineering sub-team we began running tests on it but things weren't running as smoothly as we were hoping it would. Small mistakes, things we forgot to do and some overdue things we should have done before the robot arrived set us back with testing fully autonomous code on the robot.



$R_\alpha S_d L_\gamma$



$R_\alpha L_\beta R_\gamma$

The Systems of Our Robot

The Drivetrain:

For the drivetrain's motor controllers we use one Talon SRX on each side and two Victor SPXs which follow the Talon SRX. On top of the motor controllers we use the navX-MXP as our IMU the navX is essential for our autonomous drive since this algorithm relies on the navX to know the current heading of the robot.

Hatch Intake:

The hatch intake system itself consists of two double solenoids that need open and fold. An issue that came with it was the fact that if we had either the gripper or the extension plate open while lifting the elevator past a certain height, the system would struck and hurt itself and the cameras. To solve this issue, we added a safety feature that closes the Hatch system before the elevator is raised or lowered past that breaking point. This is an improvement from last year's robot. In that robot, we just prevented the elevator from moving if the Power Cube intake was folded. This year we made the program a bit more complex and made the robot to reach the desired state (raising the elevator). As simple as it was to program the system delayed us quite a bit, a few mechanical problems that we didn't spot made us think we did something wrong and were hard at trying to find a solution for it while not knowing it wasn't even our issue.

Cargo Intake:

The Cargo Intake system uses one Talon SRX for controlling the angle of the wrist with an absolute CTRE SRX Mag Encoder as its feedback device. This system also uses a proximity sensor on the side of the wrist so it knows automatically when to stop intake process of the Cargo. The wrist uses the built in Talon SRX motion profiler "Motion Magic" to control the angle of the wrist. "Arbitrary feed-forward" is used to keep the wrist in place at any position.
The function we used was a simple cosine function with some adjustments to fit the needed current to support the wrist and ended with this function:

$$1.1 \cdot \left(0.2 \cdot \cos(x) + 0.025 \cdot \operatorname{sign}\left(\cos(x)\right)\right)$$

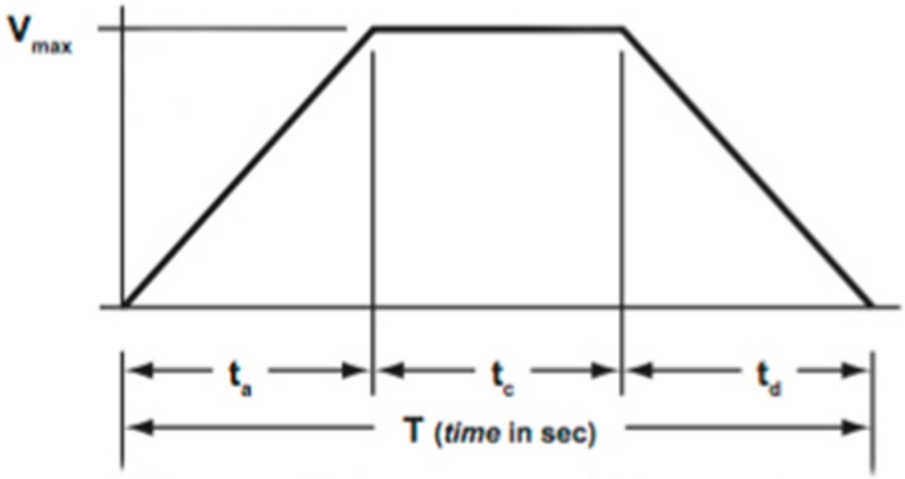
Elevator:

The elevator uses a Talon SRX motor controller to control the elevator movement, with a second motor controller allowing the elevator to rise.
To negate the effects of gravity, we use a constant arbitrary feedforward, which keeps the elevator in place.
Taking in consideration the way our elevator is built, we had to split this constant into two separate heights.
When programming the elevator, we put a lot of emphasis on preventing possible threats and clashes between the systems. we wanted the drivers not to have any worries about destroying the robot, so we added the following 'safety features':

- The elevator closes all mechanisms that might collide with the robot when moving.
- The elevator can't go above an allowed height that we pre-defined.
- The elevator automatically resets its position when reaching the bottom, to negate any error which had accumulated.

Motion Magic:

The main motion profile we used for our wrist and elevator subsystems was the talon SRX's motion magic. The unique thing about the motion magic motion profile is that in contrast to other motion profiles, it does not rely on generating trajectory points. Instead the motion profile uses a Trapezoidal Motion Profile, which can be explained using the following graph:



Because one of us is a lone star

But together we are a galaxy!