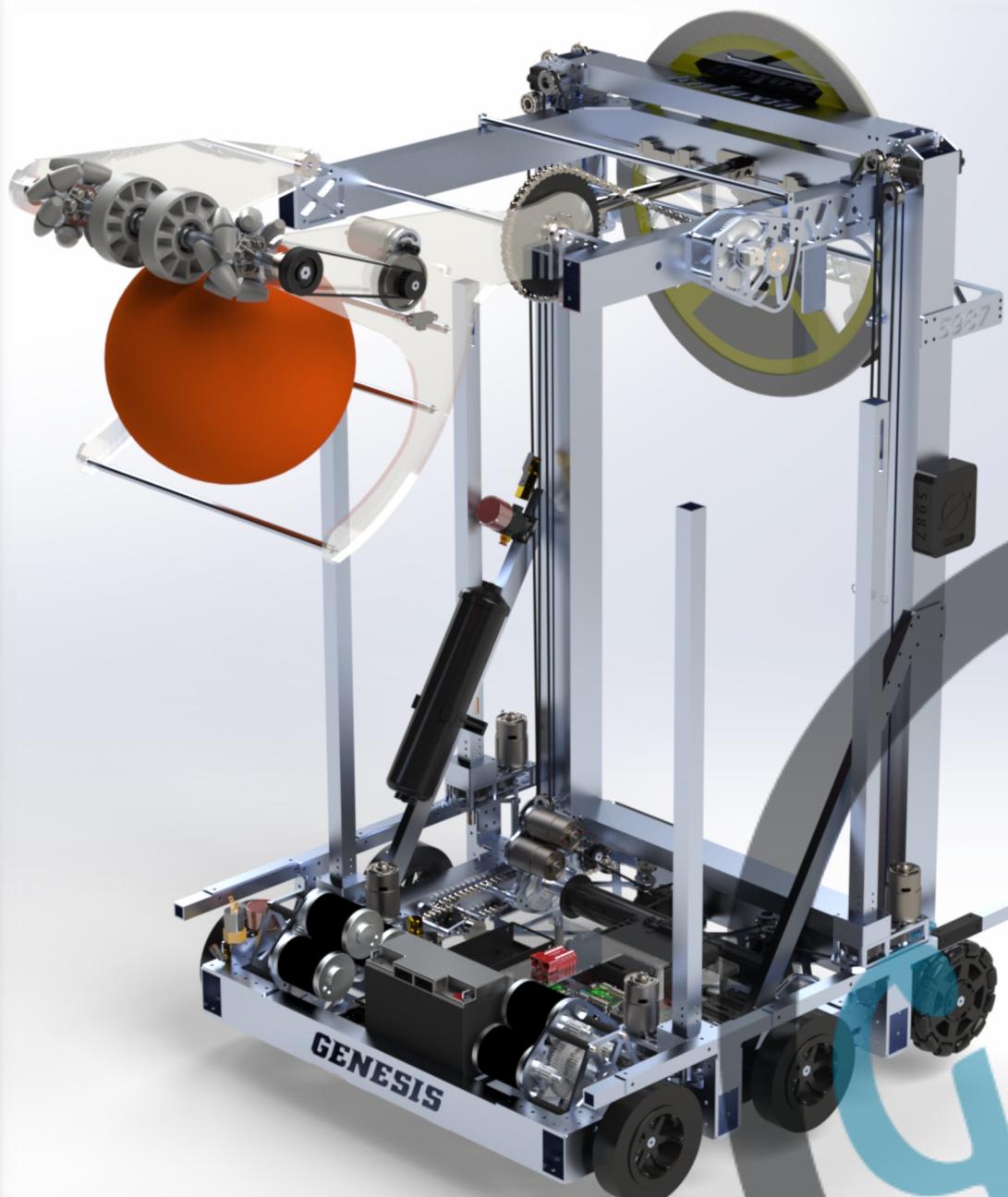


# Technical binder

**2018-2019**





# Table of contents

## Design

- Chassis—tank drive base 3-5
- Lift—continuous lift 6-7
- Cargo intake—horizontal roller 8-10
- Hatch grabber—latching fingers 11-12
- Climbing—mechanical pistons 13-17
- Gearboxes 18
- Manufacturing 19

## Vision

- Cameras 20
- Framework 21
- Neural networks 22-24
- Jetson 25
- Ordinary vision 26

## Control

- Path pursuing 27
- Arm control 28
- Lift 29



# Chassis- tank drive base

## Goals

Stable base for a high center of mass robot

Sturdy for push fights

Agile for fast maneuvers

Reliable and simple

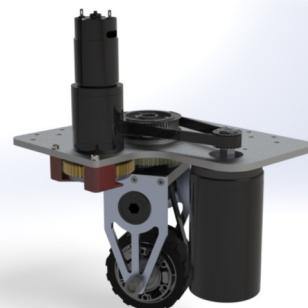
Easy for maintenance

Can go on the ramp reliably

## **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**

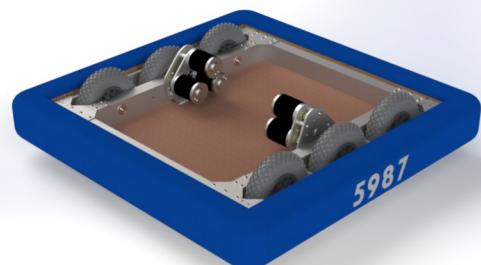
Tank is the “normal” FRC chassis drive. it has 2-4 wheels on each side and a belt or chain connecting the wheels.

#### Pros

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

#### Cons

- Can't strafe





## Our choice - 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. 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.

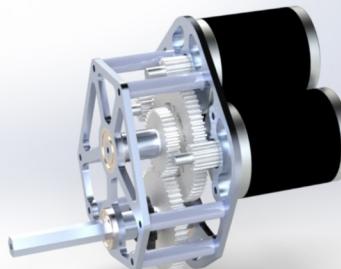
### Prototype conclusions

- WCD works better than the Kit of Parts chassis (AndyMark chassis) and is simpler to build and maintain
- E4T encoders have many problems with connection and are breaking
- 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
- 60x40mm tubing for



### 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.



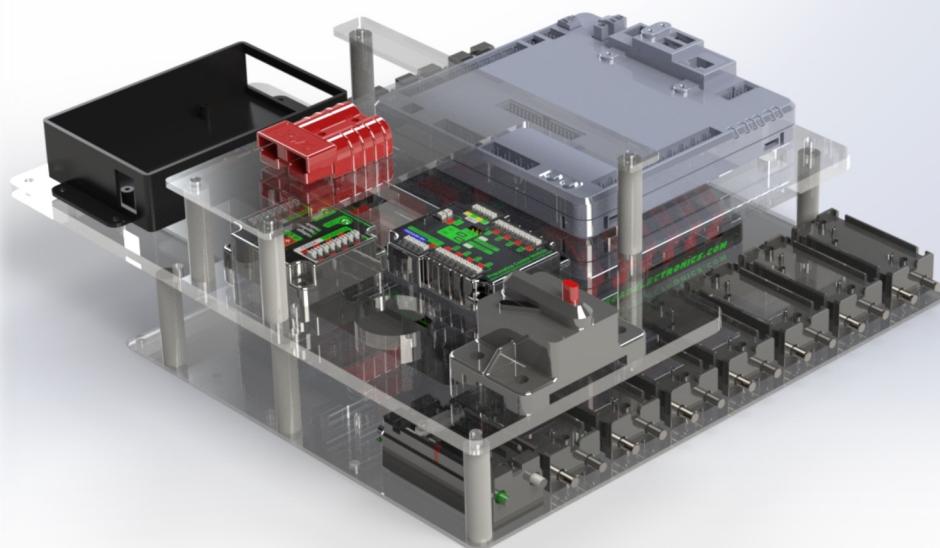
## Bellypan

The Bellypan has 2 goals, to give structural rigidity to the chassis and to help connect the mechanisms on the bellypan, because in the beginning we had wanted to connect the electronics onto the bellypan, we needed material that was rigid and light, first, 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 conductor and heat conductor, however, it was quite heavy and not rigid enough, as some of the pulleys that hold the lift are connected straight to the bellypan and 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 electrical.

## Electronics board

In the off-season, we wanted to make the electronics board an integral part of the bellypan, with all the electronics board being, in fact, the bellypan, however, after we realized that it can save us a lot of time to finish the electronics board before the

integration into the skeleton of the robot, we decided to redesign the entire thing into a different, 3 levels high electronics board out of 4mm polycarbonate.





# Lifting – continuous elevator

## Goals

Can place Cargo and Hatch panels in the Cargo ship and all levels of the rocket

Can score from both sides (forwards and backwards)

Has a low center of mass

## Literature review

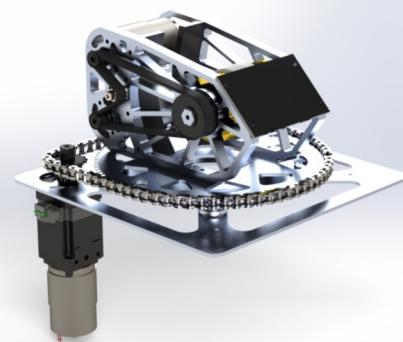
### Shooter

#### 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 easiest scoring from both sides
- Simplest and most reliable

#### Cons

- Harder to control (both 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

- Most complex mechanism
- Takes up a lot of space on the robot



## Final mechanism- Double stage wristed continuous elevator

### Elevator stages

Doesn't exceed 1.2m when stowed

2 stages to Reach 3rd level port and

5x16x5 mm bearings press against outer face and sides of 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 low weight as a conclusion from previous robots (2018)

3D printed pulleys fit on the side of the elevator to avoid cable collision with Cargo in backwards shooting

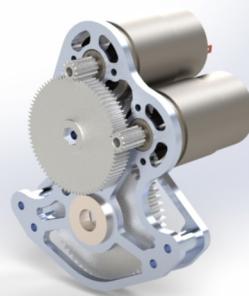
Horizontal and diagonal system of pulleys brings the cables to the side of the robot from the central winch

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
- Can reach max speed of up to 3 m/s
- 2 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 threaded winch was used
- originally 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

- Must grab cargo quickly
- Must be able to go out/in the robot's frame perimeter
- Must hold the balls tight, so they won't get off when other robots are playing defense
- Simple and easy to fix

## Engineering design and prototype:

### 3 fingers arm

A 3 fingers arm is a robotic arm with 3 fingers that grabs the balls just like a real human arm

Pros

- Centering

Cons

- No experience in FIRST
- Slow
- Complicated
- takes a lot of space

### Platypus (delphi)

A platypus mechanism is a T shaped rubber band intake. Looks like a platypus tail

Pros

- Centering
- Touch it own it
- A lot of points of contact

Cons

- Takes a lot of space
- Needs a separate mechanism for the elevator
- Heavy



## Rubber bands

Rubber bands moving into the robot for intake.

Pros

- Fast

Cons

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

## Horizontal roller

An horizontal roller is a roller that spins and takes balls. Can be with/without wheels

Pros

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

Cons

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



## Our choice - horizontal roller with wheels

### Prototype conclusions - Horizontal roller

This mechanism grabbed very quickly and efficiently and did answer all of our goals.

Found distance to catch all balls within the given tolerance

### Mechanism breakdown

The mechanism goes inside the frame perimeter for 2 main reasons:

start of match

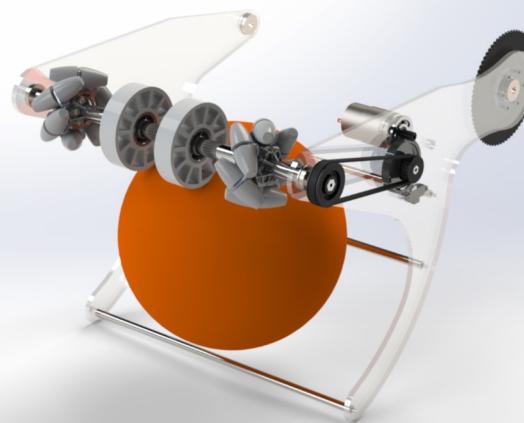
backwards shooting

Usage of 4 wheels in total: 2 4in. mechanums and 2 4in. compliant.

Very simple and easy to change,

fix or replace.

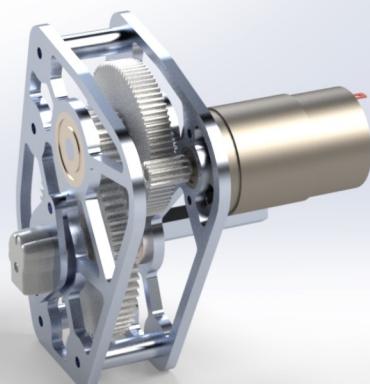
Curved inwards on the sides to  
get balls from wider angles.



### Wrist gearbox

This gearbox is made to get the whole intake inside the robot's frame perimeter at the beginning of the match. Also, it helps up 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 and fast as well. We used one 775 pro motor with a 88:1 ratio to get the best speed: torque ratio.





# Hatch grabber - latching fingers

## Goals:

- Grab hatches dropping them on the floor
- Require low precision alignment on the depot
- Extend outwards for easier hatch placing
- 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 the hook and loop tape on the hatch panels to grab and hold, weaker than those on the rocket and cargo ship

#### pros:

- Easy
- Light
- Few actuations required

#### cons:

- Requires alignment
- Regular wear - competition might not change out their hook and loop tape, 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
- Lots of actuations required



## Final mechanism: latching fingers

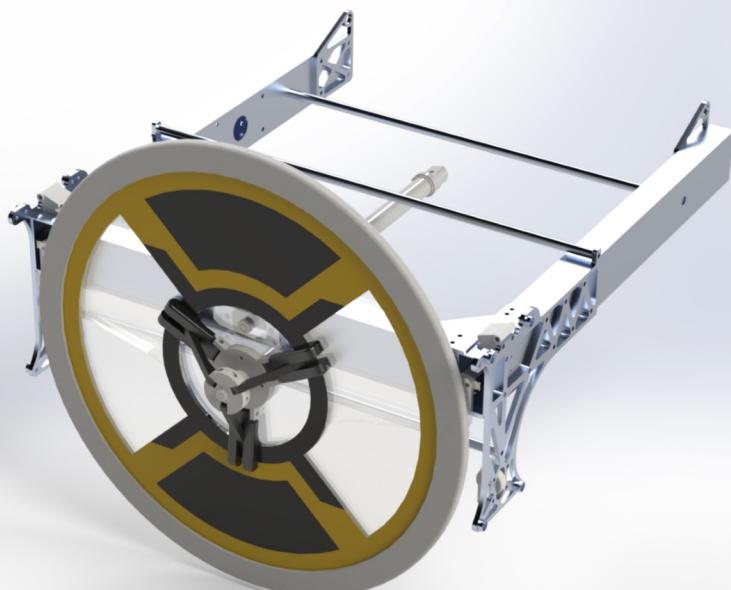
### Fingers:

Fingers powered by a pneumatic piston  
Head and back of the fingers is 3D printed out of PLA  
Fingers and pivot points printed out of rubber for extra gripping and to not to break with unexpected loads



### Back plate:

Back plate out of polycarbonate to give normal force to all three fingers  
Back plate can extend 250mm with a pneumatic piston  
Custom rails made out of a 3D printed block and a delrin rod connected to the back plate, so it won't be able to rotate on the piston.  
An attempt was made to use rails, however, our tolerances weren't good enough and broke the rails.





# Climbing four lead screw based me-

## Goals

- Climb in a short duration of time
- Is reliable, does not endanger the robot
- Doesn't require a very specific placing of the robot in front of the Hab
- Can operate 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 system

Unstable at some parts of the climb



## Two linear motion systems with angled arm in front

Two linear motion systems at the back of the robot, with an angled up at the front. The robot is pushed from the back, and slides onto the step.

### Pros:

- Low weight

- Fast

### Cons:

- Requires a full side of the bumpers

- The angled arm is complicated (needs to stick out of the bumper perimeter)

- The prototype we made didn't work due to too much friction on the front angled arm

- Unreliable, the arm can slide

## 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 step.

### Pros:

- Low weight

- Fast

### Cons:

- Requires a full side of the bumpers

- The angled arm is complicated (needs to stick out of the bumper perimeter)

- 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 planned / modeled alongside the chassis and will be 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 placing 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.

Cannot be made in house

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
- Can be made in house
- Doesn't need a big gearbox as the screw itself does most of the RPM to torque conversion
- Takes very little 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 a winch and the top of a profile, that when the winch spins gets pulled down and lifts up the robot.

### Pros:

- Very common in first
- Can be made in house
- Requires a motor

### Cons:

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



## Final mechanism: climbing

### Climbing

775pro driving each leg with a ratio of 5:1 before leadscrew

4 lead screws 12mm diameter 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

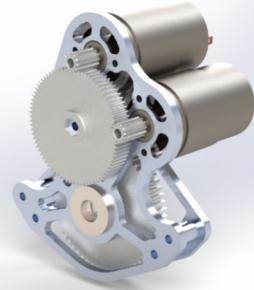
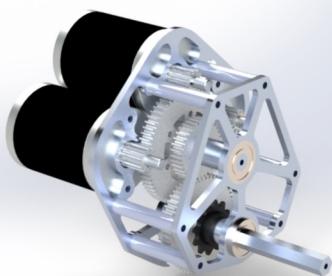
Custom 3d printed gears to drive both sides





## Custom gearboxes

We started building our own custom gearboxes in 2018 with the help of our mentors designing your own gearboxes have 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

By Ratio	Max. Power	Max. Efficiency 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	162.26:1
Rot. Free Speed (rps)	43.36	130.76	17.63	261.52	3.04	1.92
Rot. Loaded Speed (rps)	36.17	65.38	16.44	0.00	3.00	1.91
Lin. Free Speed (ft/s)	45.40	136.93	18.46	273.86	3.18	2.01
Lin. Loaded Speed (ft/s)	37.88	68.47	17.22	0.00	3.14	2.00
Current Per Motor (A)	18.4	54.0	7.9	107.3	1.9	1.5
Stall Load (lbs)	18.1	6.0	44.5	3.0	258.5	407.8
Stall Voltage (V)	2.0	6.0	0.8	12.0	0.1	0.1

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:

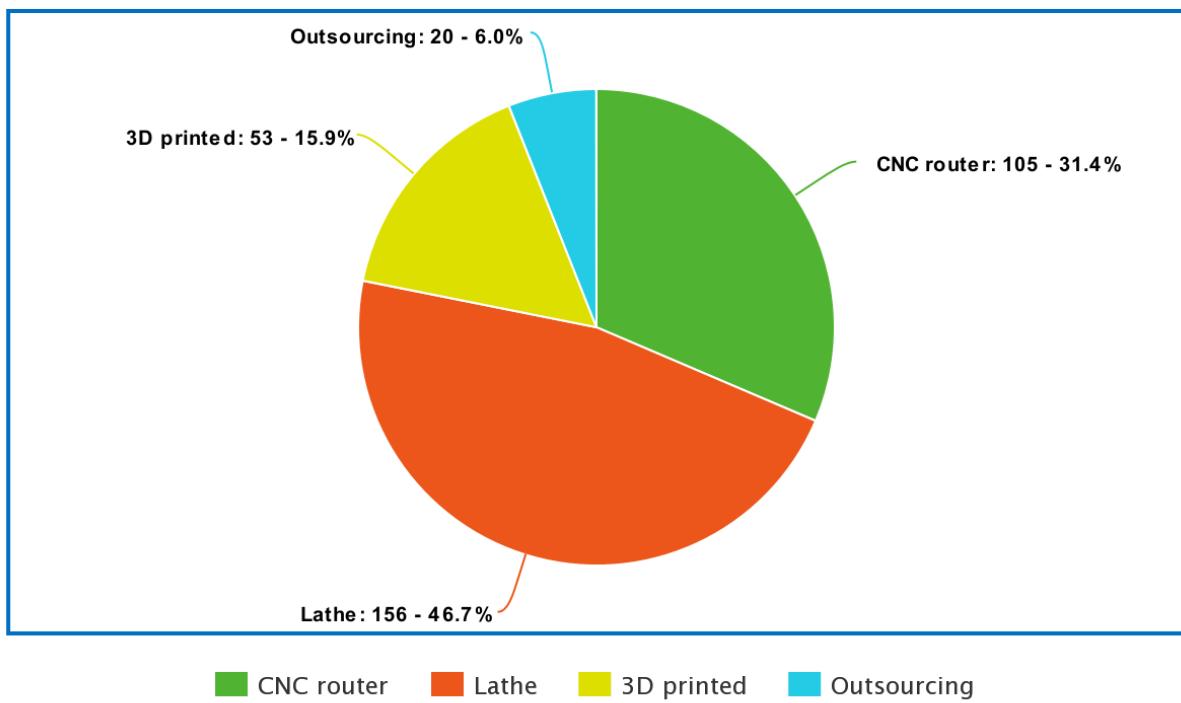
Winch: As an endgame, you need to winch up the robot by spinning a cable around a circular drum. For radius, enter the drum's radius. For load, enter the weight



## 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.

manufacturing tools and amounts



meta-chart.com



# 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.



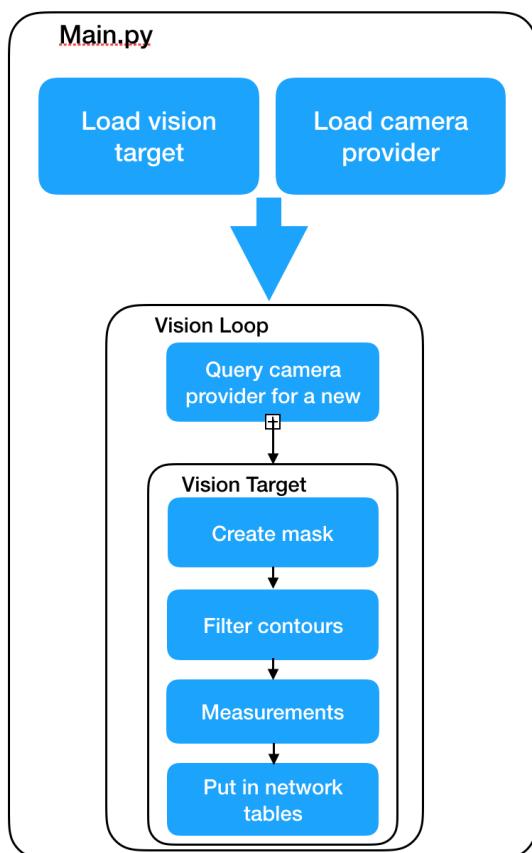


## Framework - 1 page

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 preset 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 - 4 pages

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).

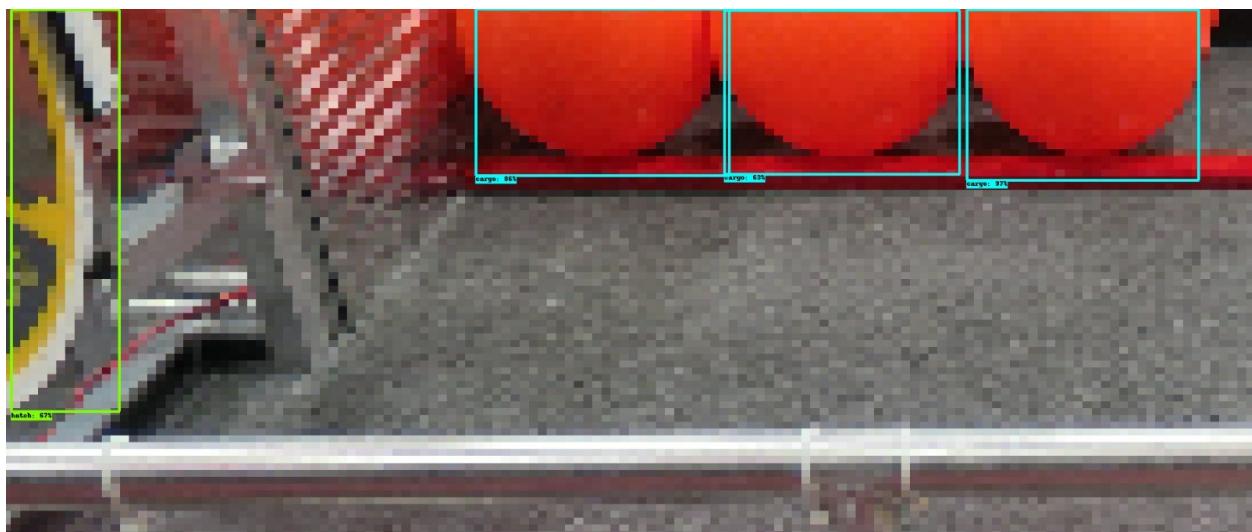




After you've created your dataset you need to train the neural network, we used a pre-configured model that was already trained on the COCO dataset which is a huge dataset containing around 300 thousand labeled images. By using this we improved training time and accuracy because the network already knows 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 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 separate board

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 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.





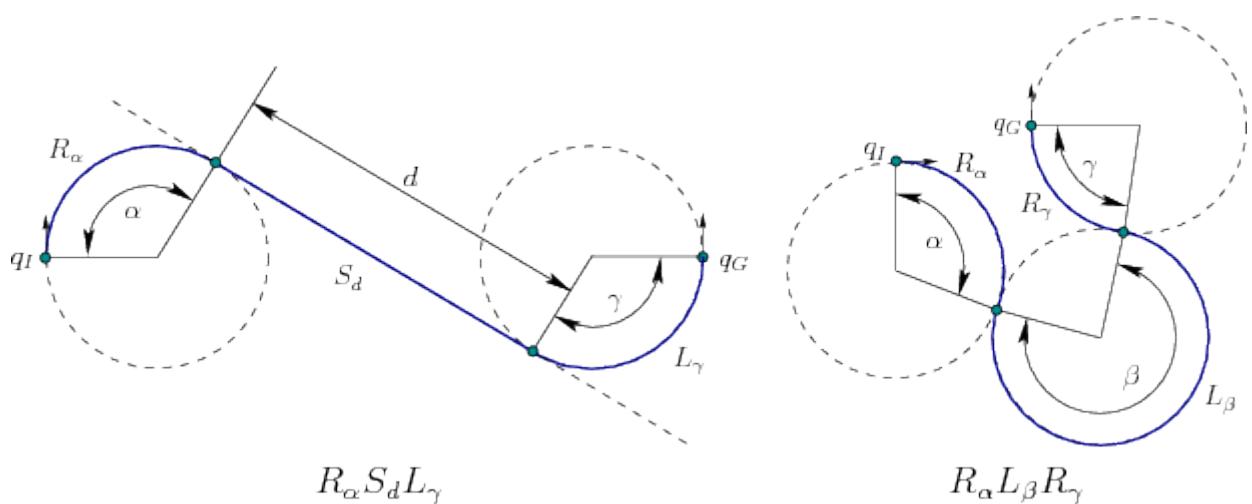
## Autonomous pathing and path pursuit needs a makeover

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.





## The systems of our robot

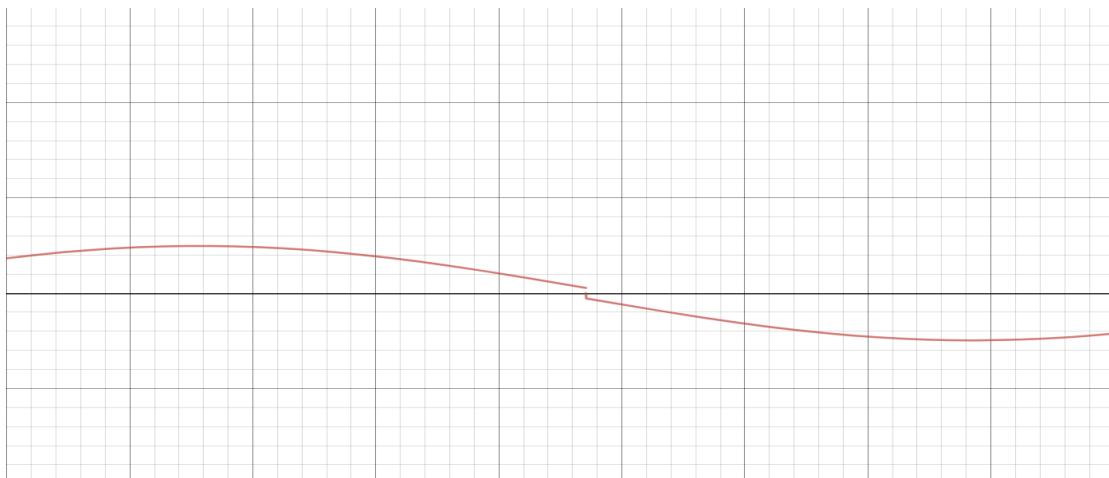
The Drivetrain - For the drivetrain's motor controllers we use one Talon SRX on each side and two Victor SPXs which follow the Talon SRX. On top of the motor controllers we use the navX-MXP as our IMU the navX is essential for our autonomous drive since this algorithm relies on the navX to know the current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

$$1.1 \cdot (0.2 \cdot \cos(x) + 0.025 \cdot \text{sign}(\cos(x)))$$





## 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.

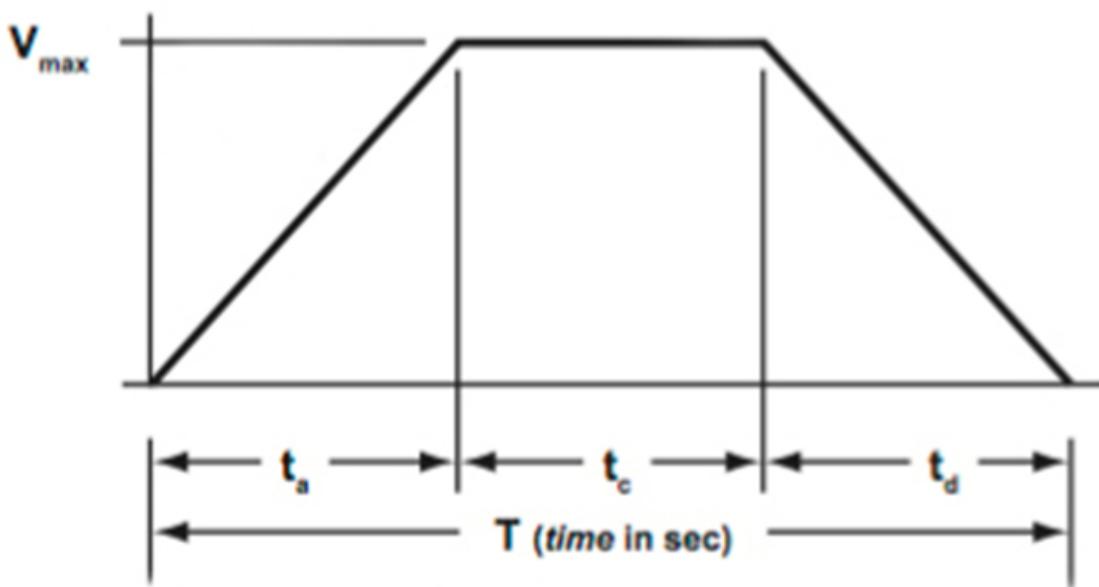
Taking in consideration the way our elevator is built, we had to split this constant into two separate heights.

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:





משרד החינוך



Because one of  
us is a lone star

But together we  
are a galaxy!