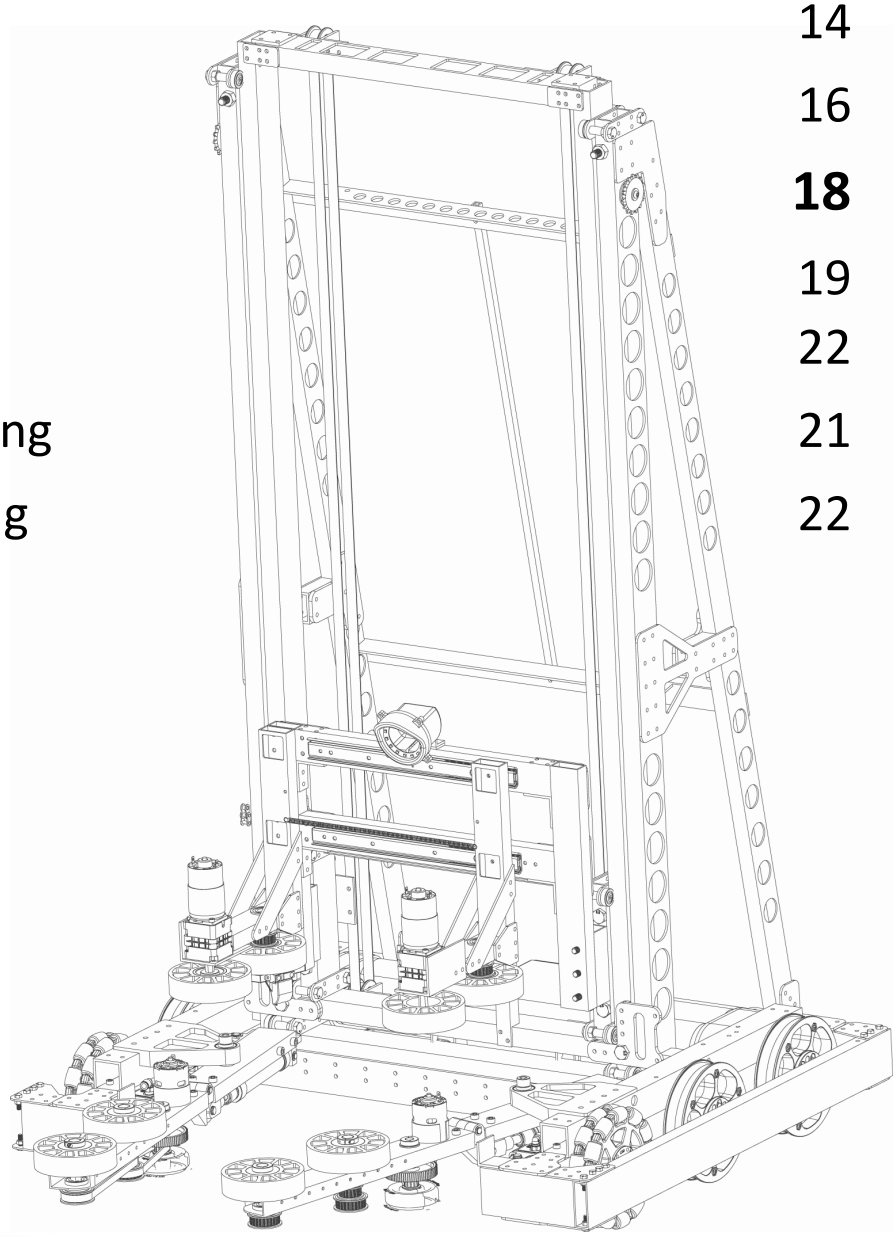
Longbow

2018 Season

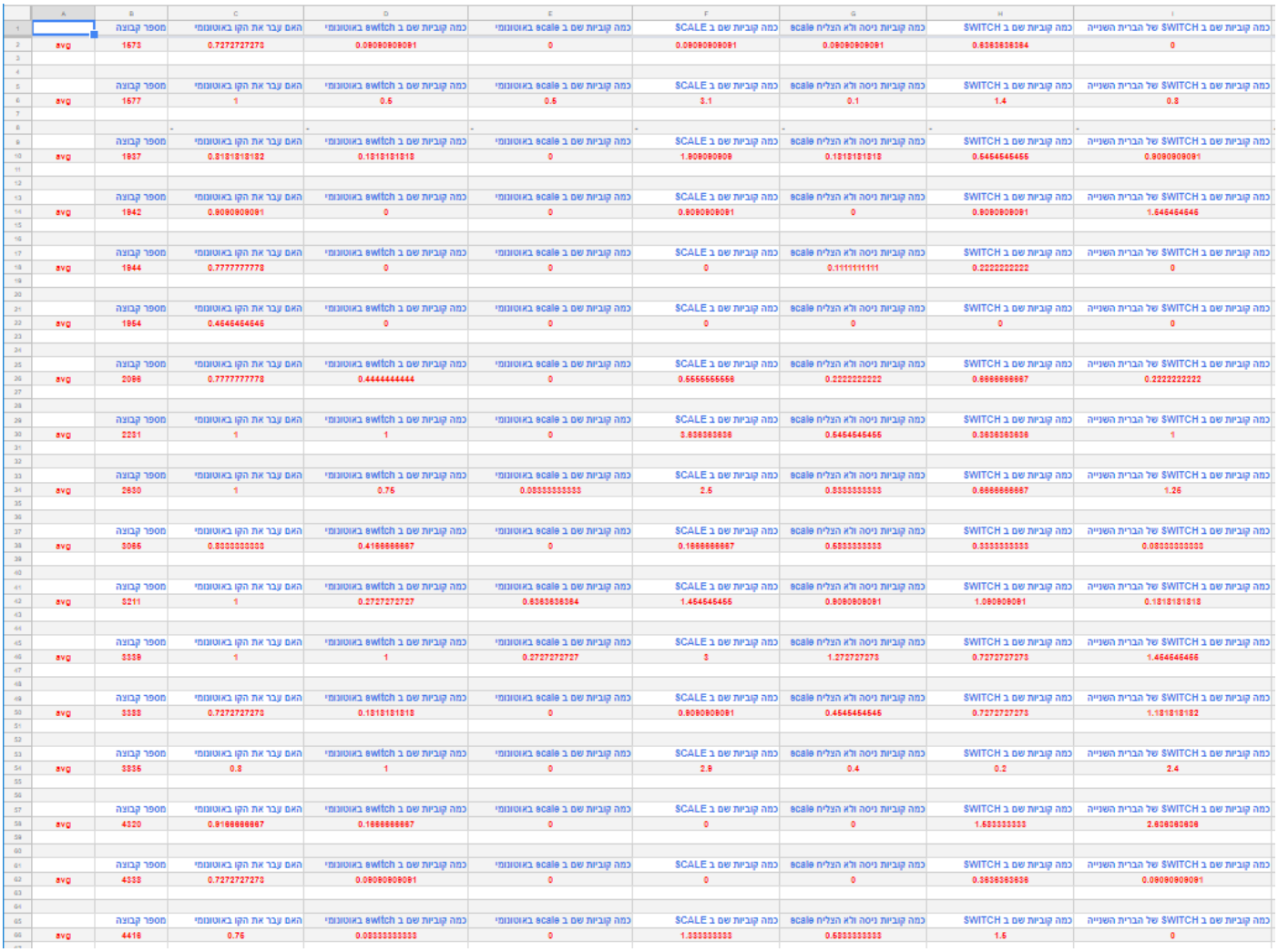
Galaxia #5987

In memory of David Zohar

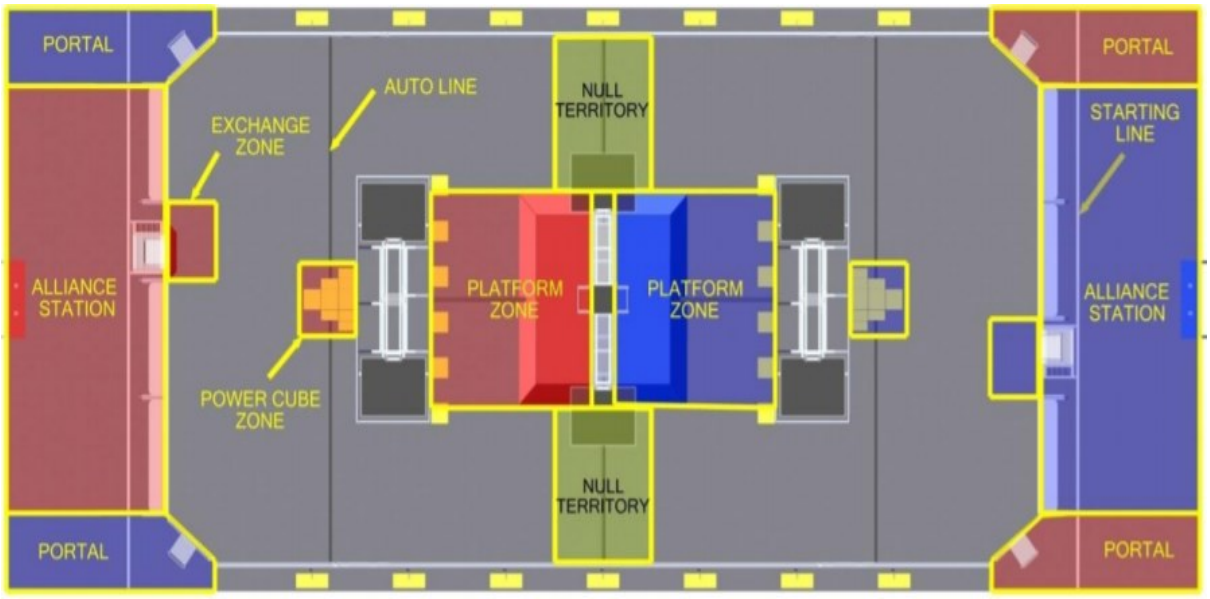
Table of Contents



Strategy



	A	B	C	D	E	F	G	H	I
1		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
2	avg	1573	0.7272727273	0.09090909091	0	0.09090909091	0.09090909091	0.8383838384	0
3									
4									
5		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
6	avg	1577	1	0.5	0.5	3.1	0.1	1.4	0.8
7									
8									
9		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
10	avg	1937	0.8181818182	0.1818181818	0	1.909090909	0.1818181818	0.6454545455	0.9090909091
11									
12									
13		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
14	avg	1942	0.9090909091	0	0	0.9090909091	0	0.9090909091	1.545454545
15									
16									
17		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
18	avg	1944	0.7777777778	0	0	0	0.1111111111	0.2222222222	0
19									
20									
21		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
22	avg	1954	0.4646464646	0	0	0	0	0	0
23									
24									
25		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
26	avg	2098	0.7777777778	0.4444444444	0	0.5555555556	0.2222222222	0.8888888887	0.2222222222
27									
28									
29		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
30	avg	2231	1	1	0	3.838383838	0.5454545455	0.3838383838	1
31									
32									
33		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
34	avg	2630	1	0.75	0.08333333333	2.5	0.8333333333	0.8888888887	1.25
35									
36									
37		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
38	avg	3065	0.8333333333	0.4166666667	0	0.1666666667	0.5833333333	0.3333333333	0.08333333333
39									
40									
41		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
42	avg	3211	1	0.2727272727	0.8383838384	1.454545455	0.9090909091	1.090909091	0.1818181818
43									
44									
45		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
46	avg	3338	1	1	0.2727272727	3	1.272727273	0.7272727273	1.454545455
47									
48									
49		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
50	avg	3388	0.7272727273	0.1818181818	0	0.9090909091	0.4545454545	0.7272727273	1.181818182
51									
52									
53		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
54	avg	3835	0.8	1	0	2.9	0.4	0.2	2.4
55									
56									
57		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
58	avg	4320	0.9166666667	0.1666666667	0	0	0	1.583333333	2.838383838
59									
60									
61		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
62	avg	4338	0.7272727273	0.09090909091	0	0	0	0.3838383838	0.09090909091
63									
64									
65		מספר קבוצה	האם עבר את הקו באוטונומי	כמה קוביות שם ב switch באוטונומי	כמה קוביות שם ב scale באוטונומי	כמה קוביות שם ב SCALE	כמה קוביות ניסה ולא הצליח scale	כמה קוביות שם ב SWITCH	כמה קוביות שם ב SWITCH של הברית השנייה
66	avg	4418	0.75	0.08333333333	0	1.333333333	0.5833333333	1.5	0



Action	Criteria	MATCH Points		Ranking Points
		AUTO	TELEOP	
AUTO-RUN	For each ROBOT that breaks the vertical plane of the AUTO LINE with its BUMPER at any point in the AUTO stage	5	-	-
OWNERSHIP	SCALE	2 + 2/sec	1 + 1/sec	-
	ALLIANCE'S SWITCH	2 + 2/sec	1 + 1/sec	-
VAULT	For each POWER CUBE placed in the VAULT	-	5	-
PARKING	For each ROBOT fully supported by the SCALE (either directly or transitively), not at all in the opponent's PLATFORM ZONE, and has not CLIMBED	-	5	-
CLIMBING	For each ROBOT fully supported by the SCALE (either directly or transitively) with BUMPERS fully above the BRICKS at T=0, not in direct contact with their PLATFORM, and not at all in the opponent's PLATFORM ZONE	-	30	-
FACE THE BOSS	All three (3) ALLIANCE ROBOTS have CLIMBED or two (2) ROBOTS have CLIMBED and the ALLIANCE has played the LEVITATE POWER UP	-	-	1
AUTO QUEST	ALLIANCE completes three (3) AUTO-RUNS and has OWNERSHIP of their SWITCH at T=0 of the AUTO stage	-	-	1
Win	ALLIANCE's final MATCH score exceeds their opponents'	-	-	2
Tie	ALLIANCE's final MATCH score equals their opponents'	-	-	1

Game analysis

From our analysis we realized that the scale seems the be the most valuable factor in this year's game. owning the scale earns our alliance a point for each second of ownership while preventing the opposing alliance earning those points, although it's the hardest to control and manage during the match.

	RR	RL	LR	LL	AFTER AUTO	SW Sc	SW	SC	None	TELEOP	SW Sc	SW	SC	None	ENDGAME		
	RR	RL	LR	LL		SW Sc	SW	SC	None		SW Sc	SW	SC	None		SW Sc	SW
straight	line	line	line	line		NA	Vault	NA	Vault		bring cube to scale	Vault->defend	Vault->defend	Vault->defend		vault-> defend	vault-> defend
scale	switch	switch	switch	switch		NA	Scale	NA	NA		scale/osw	Scale	switch	Scale		osw	scale
straight	line	line	line	line		NA	bring cube to scale	NA	Vault		vault->defend	Bring cube to Scale	bring cube to switch	Bring cube to Scale		defend	bring cube to scale
straight	line	line	line	line		NA	Bring cube to Scale	Vault	Vault		defend	Vault->defend	Vault->defend	Vault->defend		vault-> defend	vault-> defend
straight	line	line	line	line		NA	Vault	Vault	Bring cube to Scale		vault->defend	Bring cube to Scale	bring cube to switch	Bring cube to Scale		defend	bring cube to scale
scale	turn right switch	turn right switch	switch	scale		NA	Scale	Switch	Scale		scale/osw	Scale	switch	Scale		osw	scale
scale	scale	switch	turn left switch	turn left switch		NA	Scale	Switch	Scale		scale/osw	Scale	switch	Scale		osw	scale
straight	line	line	line	line		NA	Vault	Vault	Bring cube to Scale		defend	Vault->defend	Vault->defend	Vault->defend		vault-> defend	vault-> defend
straight	line	line	line	line		NA	Bring cube to scale	Vault	Vault		vault->defend	Bring cube to Scale	bring cube to switch	Bring cube to Scale		scale	bring cube to scale
scale a	line	scale	turn left switch	scale		Scale	OSW	OSW	NA		vault->defend/bring cube to scale	Scale	switch	Scale		vault-> defend	scale
scale b	switch	switch	switch	switch		Vault	Vault	Switch	NA		Scale/osw	vault->scale	Vault->osw	Scale		osw	scale
scale c	scale	turn right switch	scale	line		OSW	Scale	Vault	NA		osw	Scale	osw	switch		Scale	vault-> defend
scale	scale	scale	scale	scale		Scale	Scale	OSW	NA		scale/osw	scale	osw	scale		scale	scale
switch	switch	switch	switch	switch		OSW	Switch	Switch	NA		osw	bring cube to scale->osw	switch	switch		osw	osw
straight	line	line	line	line		Vault	bring cube to Scale	Vault	NA		vault->defend	vault->bring cube to scale	Vault->bring cube to switch	bring cube to scale		vault-> defend	bring cube to scale
scale	scale	scale	turn left switch	switch		OSW	Scale	OSW	NA		osw	scale	osw	scale		scale	scale
straight	line	line	line	line		Vault	Bring cube to scale	Vault	NA		vault->defend	vault->bring cube to scale	Vault->bring cube to switch	bring cube to scale		vault-> defend	bring cube to scale
switch	turn right switch	turn right switch	line	line		Switch	Switch	Switch	NA		osw	bring cube to scale->osw	switch	vault		osw	osw
straight	line	line	line	line		Vault	vault	Vault	NA		vault->defend	vault->bring cube to scale	Vault->bring cube to switch	bring cube to scale		vault-> defend	bring cube to scale
switch	switch	switch	switch	switch		osw	Bring cube to Scale	Switch	NA		osw	bring cube to scale->osw	switch	switch		osw	osw
scale	scale	line	scale	line		Scale	Scale	OSW	NA		scale/osw	scale	osw	scale		scale	scale
switch	line	line	switch	switch		Switch	Bring cube to Scale	switch	NA		osw	bring cube to scale->osw	switch	switch		osw	osw
scale	switch	switch	line	line		Scale	Scale	OSW	NA		scale/osw	scale	osw	scale		scale	scale
straight	line	line	line	line		Vault	vault	Vault	NA		vault->defend	vault->bring cube to scale	Vault->bring cube to switch	bring cube to scale		vault-> defend	bring cube to scale
straight	line	line	line	line		defend	bring cube to scale	Vault	NA		vault->defend	vault->bring cube to scale	Vault->bring cube to switch	bring cube to scale		vault-> defend	bring cube to scale

Strategy development

Looking at our estimated scale cycle time for the game, we began calculating how much value each scoring cycle provides, given various alliance combinations.

We decided that our first goal is to put power cubes on the switch so we can secure it, and after that, our second goal is to put power cubes on the scale, no matter which alliance has ownership on it.

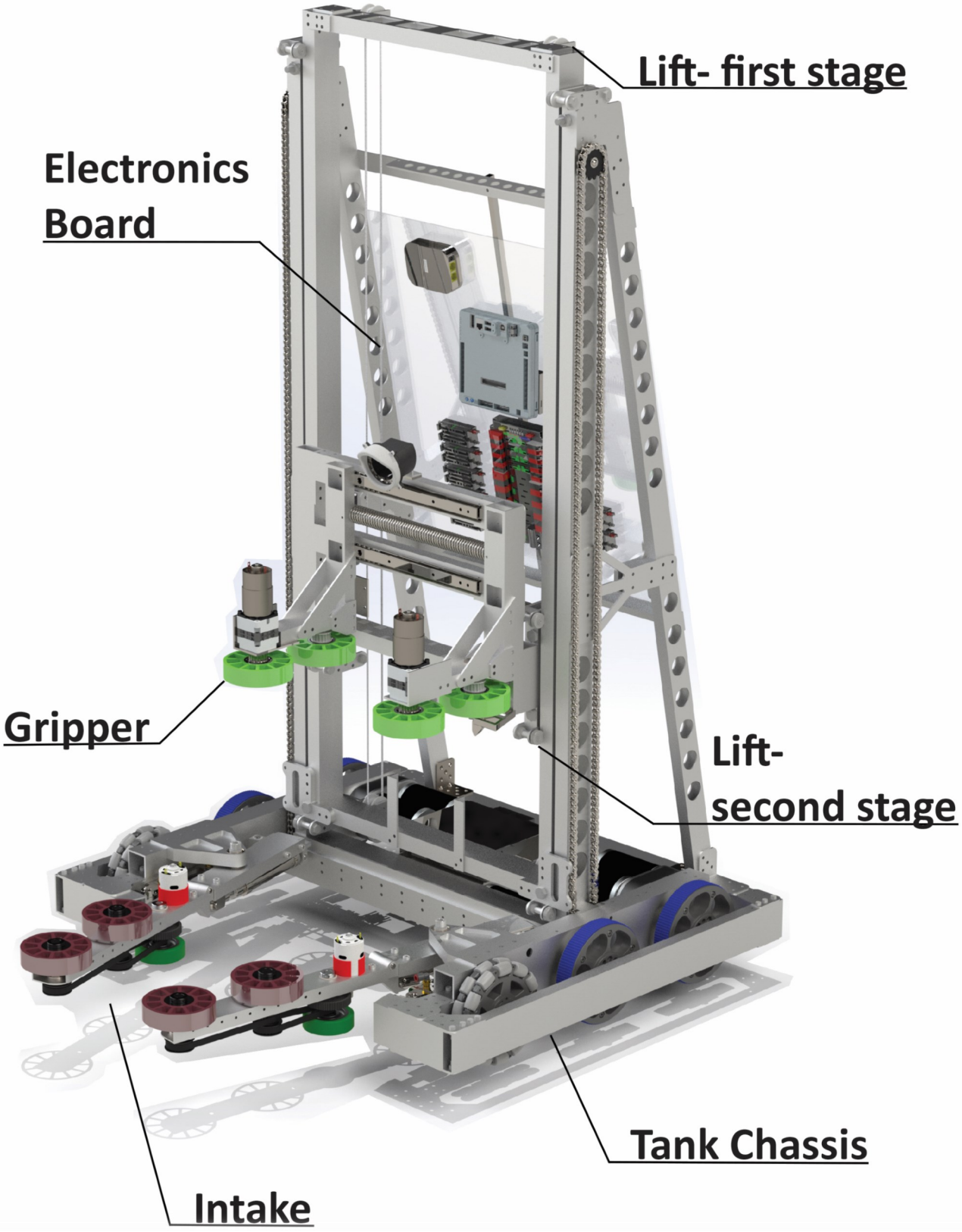
While planning our autonomous mode, we were primarily concerned with getting the ranking point for three robots passing the auto line and owning our switch. Because of that, only if we can be sure that our alliance partners can take ownership of the switch, we can use one of our other two autonomous strategies:

a. placing power cubes on the scale, whether our alliance corresponding plate is close to our starting position or it is the far one

b. driving forward to cross the auto line

During teleoperated period, in order to optimize scoring cycles, we decided to focus on scoring power cubes on the switch, and resume putting power cubes on the scale once we have our switch secured. Every match, we spend as much time as necessary to ensure that we have ownership of the switch. after that, we will focus on the scale, securing it too. When including our alliance in the strategy, we plan that our alliance partners will take ownership of our switch and will bring at least three power cubes to the exchange in order to secure the Levitate power up. Meanwhile, we will try and secure the scale and possibly also the opponent's switch.

Design


Lift- first stage
Electronics Board
Gripper
Lift- second stage
Tank Chassis
Intake

Chassis *tank drive base*

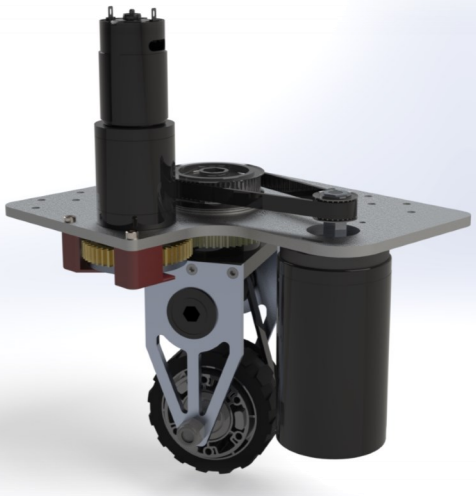
Goals

- 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
- Wide enough to allow power cubes inside

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

Mecanum

Mecanum is a wheel that can strafe and go forward like a normal wheel by rotating the wheels in the opposite direction.

Pros

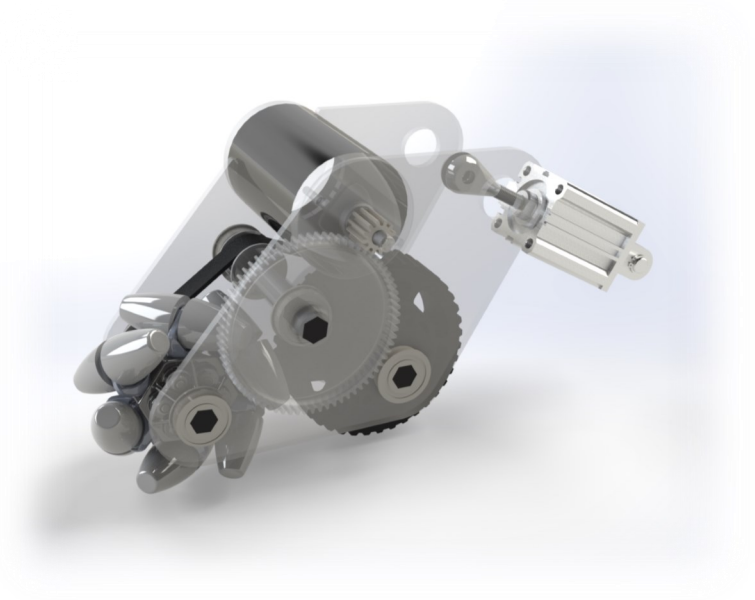
- Can strafe

Cons

- Can be easily pushed

Octocanum

Octocanum is a combination of normal wheel with Mecanum wheel with a piston changing between them.



Pros

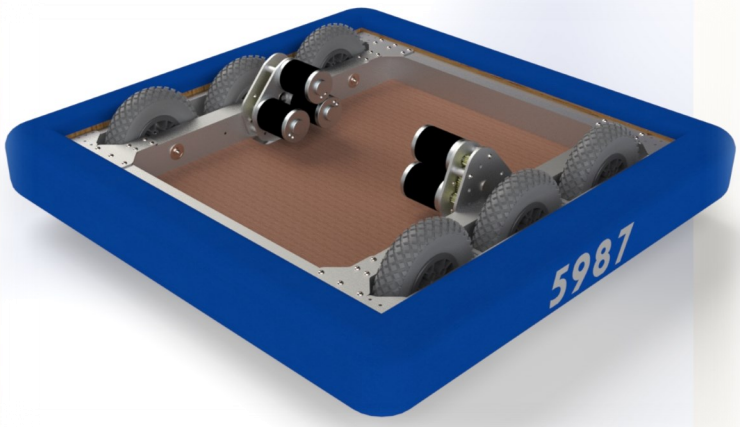
- Can strafe
- Cannot be pushed easily

cons

- Complicated
- Heavy

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

- Cannot strafe

Our choice – Tank

This year we decided to make our own custom made chassis. We designed our first chassis in the off-season. 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. One of our mentors suggested we try a chain in tube design for our off-season drivetrain. A chain tube is where you put all the chains and sprockets connecting the wheels together. The chain in tube saves space and prevents the chain from skipping.

On the off-season we used 8” pneumatic wheels for off road driving.

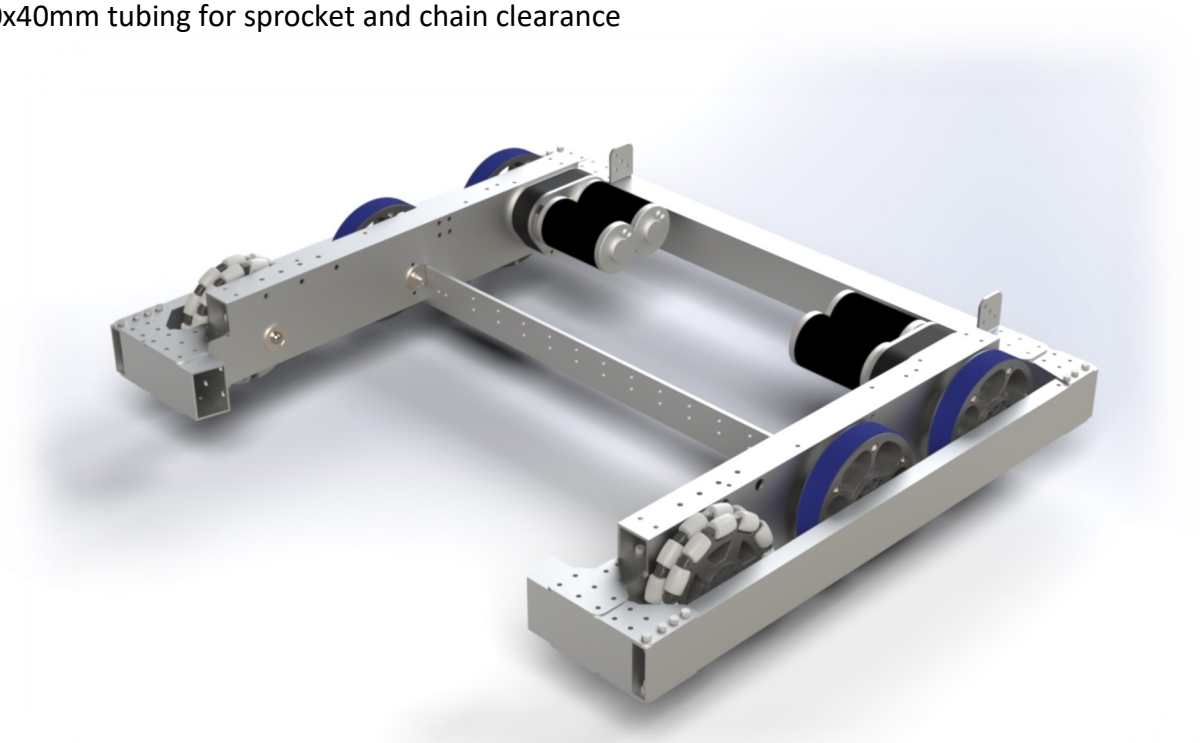
We decided not to make the octocanum and the swerve prototypes because of time and cost problems.

Prototype conclusions

- Chain in tubing works seamlessly even without tensioners
- 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 not better than 2 CIMs and take more space

Mechanism breakdown

- four 6” plaction wheels for high grip
- two 6” omni wheels for easier turning
- #35 chain for driving all the wheels
- Chain in tube for less space consumption
- Maximum speed of 5.2 m/s
- 2 CIM gearbox for each side
- One CIMcoder on each side for autonomous driving distance measurement
- 60x40mm tubing for sprocket and chain clearance



Lift *two stage cascade elevator*

Goals

- Fast
- Can put power cubes on every orientation of the scale and switch
- Stable
- Accurate and reliable, reaches the same height every time
- Light, for low center of mass

Engineering design and prototypes

Elevator with arm

A one staged elevator that lifts an arm that can rotate 180 degrees and therefore can shoot power cubes forward and backwards.

Pros

- Can place power cubes on the switch without activating the elevator
- Can shoot backwards

Cons

- Size on the robot
- Complicated
- Hard to make accurate

Continuous elevator

A two stages elevator. The first stage goes up and when it reaches the top, the second stage goes up as well.

Pros

- Reliable
- Does not need to activate the second stage to place power cubes on the switch, thus the center of mass remains low

Cons

- Hard to design and manufacture

Cascade elevator

A two stage elevator. The first stage and the second stage go up simultaneously.

Pros

- Reliable
- Fast

Cons

- High center of mass when placing power cubes on the switch

4-bar

A parallel arm that always stays perpendicular to the floor.

Pros

- Dynamic

Cons

- Slow
- Size
- High center of mass when is open
- Hard to design and manufacture
- Hard to control

Our choice – Cascade elevator

Prototype conclusions

- We decided not to use pre made liner guideways because they require high accuracy while assembling
- Do not take extra tolerance between the profile and the bearing

Mechanism breakdown

- 2 stage cascading lift
- Maximum height of 2.4m
- One CIM with 1:9 planetary gearbox connected to another reduction for a total of 1:10.8
- Maximum speed of 1.9 m/s
- 14” gap for shooting back
- #35 chain for lifting the first stage

Intake *twin angle-changing arms*

Goals

- Must work well while driving through power cubes
- Can catch power cubes both on the 13" side and on the 11" side
- Has a high reach for catching power cubes from both the row near the switch and the power cube pile
- Needs to fit into the starting configuration of the robot

Engineering design and prototype

Horizontal roller

Horizontal shaft wrapped with surgical tubing. By spinning around its center axis, it pulls in power cubes via friction.

Pros

- Has a wide opening which makes it easier for the driver to catch power cubes

Cons

- Makes it harder to catch power cubes from different orientations
- Cannot lift power cubes independently of a different mechanical system

Claw arm

A double clawed arm that grabs power cubes from the sides.

Pros

- Can be used to both grab and lift power cubes
- Has a large contact area, allowing for more friction between the arm and power cubes

Cons

- Requires accurate alignment of the claws to catch power cubes
- Cannot catch power cubes from both possible orientations

Twin angle-changing intake arms

Two Intake arms that are pressed around a pivot point towards each other. Each arm has wheels mounted on it in order to pull power cubes into the robot.

Pros

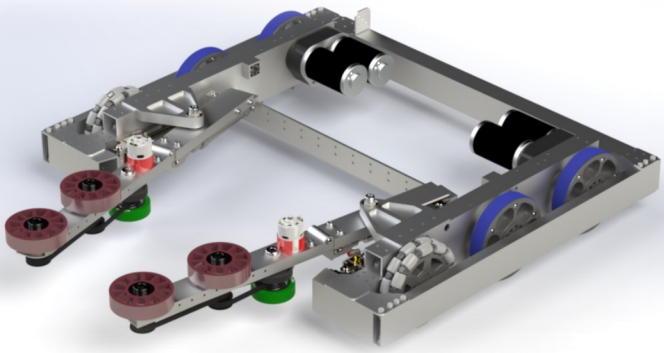
- Works well while driving through power cubes
- Can catch power cubes from both the 13” and 11” sides
- Has a long reach thanks to 16" allowed extension

Cons

- Sticks out of the bumpers, making it vulnerable to collisions with other robots

Our choice – Twin angle-changing intake arms

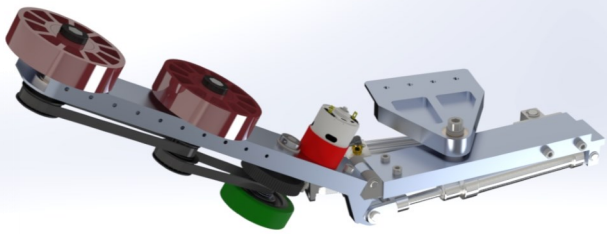
Prototype conclusions

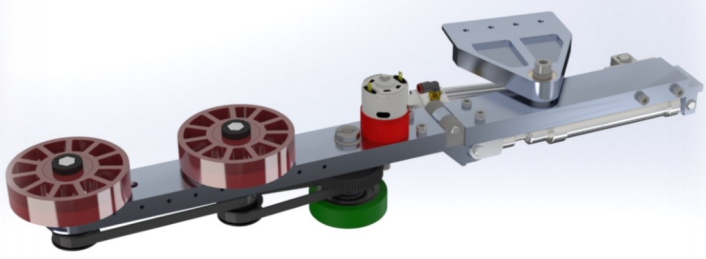


- We could not predict the spring constant we would want based on the prototype
- We would want the arms to be about 8.8” apart at their far end

Mechanism Breakdown

- Arms that fold with two pneumatic actuators and a custom made hinge
- The arms are pressed towards each other using two pneumatics actuators with constant pressure which act as an adjustable spring
- In order to pull power cubes in, we used two red compliant wheels and a BaneBots wheel
- Two Redline motors, one on each arm spin the wheels with four pulleys and two gears at full speed
- Custom made pin and hinge to make the structure mechanically stronger
- Custom made solid aluminum connection to the chassis to prevent the structure from disconnecting from the drive train





Gripper *twin parallel arms*

Goals

- Can hold power cubes at the start of each match
- Intake folding
- Allows two-sided power cubes shooting

Engineering design and prototypes

Linear moving single arm

Two arms pressed together with a spring, moving linearly on rails.

Pros

- Holds power cubes well in air
- Shoots power cubes more accurately
- Allows two-sided shooting relatively easily
- Simpler

Cons

- Heavy weight because of rails

Angular moving two arms

Two Gripper arms that are pressed around a pivot point towards each other. Each arm has wheels mounted on it in order to hold the power cube while the Lift is going up.

Pros

- Catches power cubes from floor better
- Simpler

Cons

- Not holding power cubes well while in air
- Harder if possible at all for two-sided shooting

Ramp

A ramp that holds the cube and can push the cube forward and sideways using belts.

Pros

- Holds power cubes better while in air
- Shoots power cubes more accurately and further
- Gets higher

Cons

- Does not catch power cubes well from the Intake
- Very complicated
- Harder for two-sided shooting
- Heavy weight

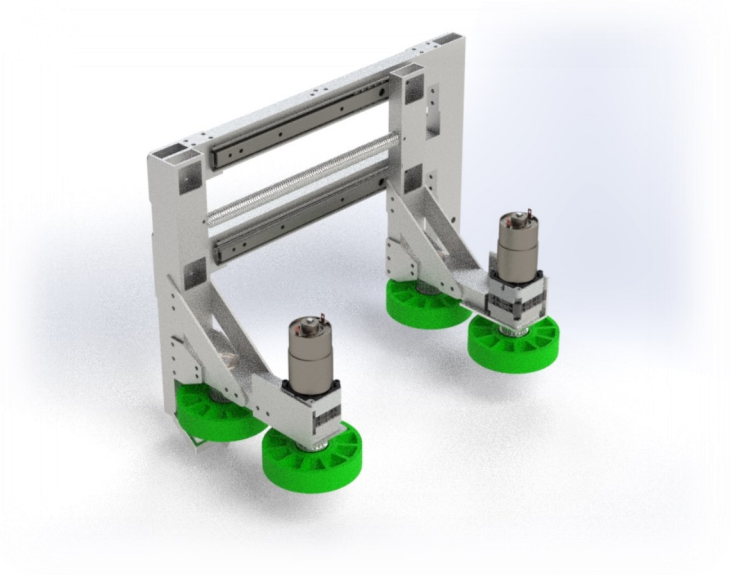
Our choice – Linear moving single arm

Prototype conclusions

- Linear movement is better because than we have more holding points on power cubes and allow two-sided shooting
- One rail breaks under the amount of torque
- We calculated an estimation of the spring constant we want for our spring. We used a rubber band and changed its tension to find the best spring constant. After that, using a force meter we calculated its spring constant. We found that we want the spring constant to be about 40 N/m.

Mechanism breakdown

- 775pro motor on each arm working full speed with four pulleys working full speed
- Two telescopic rails to allow linear motion of the arms
- A spring pulling the two arms to each other. The twin arms press on the power cube and hold it with a spring constant of about 40 N/m.
- 4" compliant wheels better hold the power cube
- Both rails and the spring are above the wheels' height to allow two-sided shooting



Programming

Code Preview (copy & paste this code into a CommandGroup)

```
addSequential(new PathPointsCommand(new Point[]{
    new Point(4.101644458283526, -0.01858027673711253 * Y_DIRECTION),
    new Point(4.196392153559519, -0.0133818285532612 * Y_DIRECTION),
    new Point(4.289857830047166, 0.002998554297554995 * Y_DIRECTION),
    new Point(4.380726520659326, 0.030330416474584142 * Y_DIRECTION),
    new Point(4.467719795261039, 0.06822922645199139 * Y_DIRECTION),
    new Point(4.500795522350906, 0.0860279623582285 * Y_DIRECTION),
    new Point(5.245173478337055, 0.5094226184980875 * Y_DIRECTION),
    new Point(5.3247273180621555, 0.5611462708113735 * Y_DIRECTION),
```

Control

Drivetrain

For the drivetrain we use four Victors, two on each side, controlled by PWM from the roboRIO and two CIMcoders, one on each side. In order to achieve our path following algorithm we apply PIDF velocity control tuned with empirical constants.

Lift

In order to position the Gripper and maintain a certain height we use a closed loop PIDF controller that runs periodically on the Talon SRX motor controller, with the help of a magnetic SRX encoder. PIDF is a control system that modifies the variable controlled, in this case the height of the lift, and multiplies it by a constant, retrieving the speed needed to reach the target value. Two sets of PIDF constants combined with nominal and peak outputs were used to ensure that the lift moves smoothly and quickly both upwards and downwards.

Intake and Gripper

We use a vast group of sensors and safety measures to make sure nothing happens that might damage the robot. We use a proximity sensor attached to the Gripper to make sure the Intake wheels stop spinning once a power cube has been collected. Additionally, we check the height of the Lift to make sure the Intake will not pull a power cube inside when we would not be able to pick it up with the Gripper.

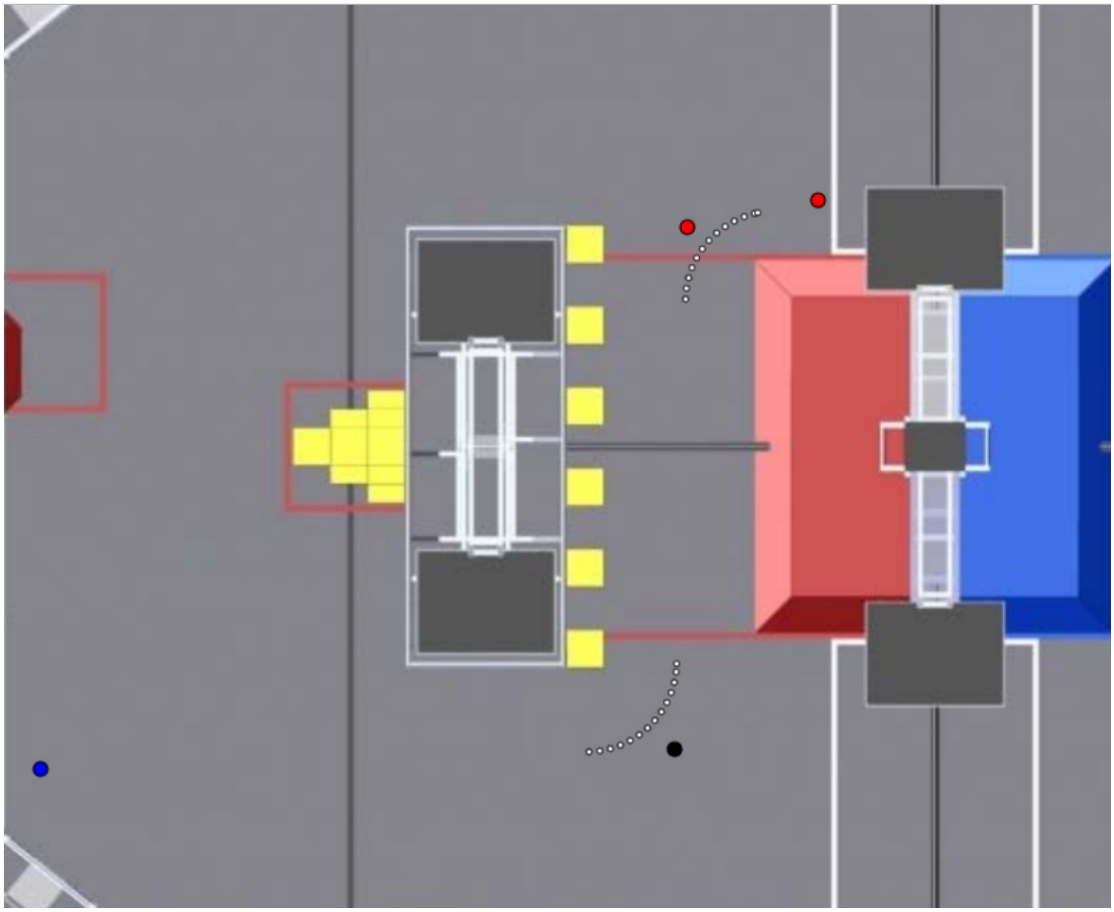
We use the same principles as mentioned above to try and prevent damages to the robot, but on top of those we also made sure the driver cannot shoot the cube backwards at certain heights that would harm the electronics board. We also built an override button for the driver to use and disable the safety measures if needed.

Autonomous paths

The driving path to the switch is calculated using the distance and angle from the vision target retrieved from the Raspberry Pi. Using vision processing makes our robot adaptable to slight changes in the arcade or in robot positioning.

GUI Path Generator

We developed a web app to easily define waypoints for the autonomous driving to the scale. We do this by plotting waypoints on a field diagram to generate the path code. The program simulates the robot position along the path on the field for faster tuning. The program also does the fillet on the path for faster turns.



Watchdoge

In the middle of our build season we started having a problem: Motors would overheat because they were pulling too many amps from the Power Distribution Panel (PDP) even after their breakers hit theirs limit.

We solved that problem by making the subsystems inherit an add-on class. In this new inherited class, we have four functions the subsystem implements that tell us when the motor should be or is enabled or disabled.

We created a function that goes through all of the mechanisms and checks the amount of amps they pull for a set amount of time. To make sure the motors won't be called by accident when they are disabled we used a `safeSetPower()` function that would first check if the mechanism is disabled before trying to set the values.

Image processing

We split the code to three different threads running in parallel on our Raspberry Pi, one that grabs frames, one that processes the most recent frame and one that shows the frame. We have done that in order to have a somewhat decent frame rate for the driver, since we use the camera not only for vision processing but also a way for the driver to see where the robot is. Since we have several targets with different features we needed to create different modes for each of them so that all you need to switch mode is to change a value on the vision network table.

Algorithm

- Get frame from the camera
- Analyze the frame:
    - Filter pixels by HSV values
    - Check if the target is one of the contours by checking its properties
    - If found:
        - Get the angle by getting the distance from the middle of the screen in pixels and using the focal length of the camera to calculate the angle
        - Get the distance by using the focal length, the real height of the target and the pixel height of the target
        - Upload the distance and angle to the vision network table
    - Upload to the network table whether the target is found
- Upload frame to a site on the raspberry pi for mjpg stream

For calculating the distance, we took a formula for getting the focal length of the camera using the physical width, distance and pixel width count of an object. We can then change the formula to get the distance using the pixel width:

$$focal\ length = \frac{pixel \cdot distance}{width} \qquad distance = \frac{focal\ length \cdot width}{pixel}$$

Unfortunately, there is a problem in using the width of an object because the visible width changes by the location we view the target from. This problem is easily solved by using the height of an object instead of the width in the calculation.

The Raspberry Pi retrieves a frame, filters all the pixels whose color values fit our predetermined range to create a mask on which the analysis will be performed. From there, it draws contours around the objects it sees in the mask, similarly to wrapping a string around their perimeter. The contours are then filtered by various properties, such as the contour area, to leave only our target. From there, we find the distance and angle from it.

Motion profiling

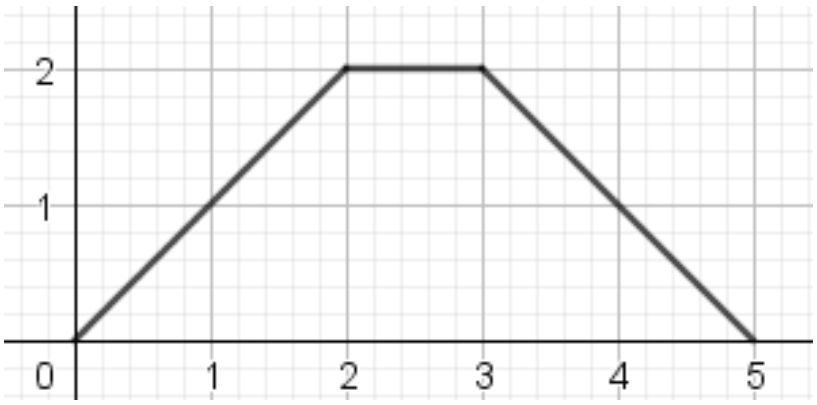
In our team we wanted to find a reliable way to travel long and short distances without the robot slipping, getting off-track or getting stuck. In order to do that we decided to use a method called Motion Profile.

Motion Profile is the idea of planning and controlling the velocity of the robot at a certain time or a certain position. To accomplish that we tried a couple of options for determining the robot's desired velocities.

Velocity by time

The idea was to use the formula $v(t) = v_0 + a \cdot t$ to create a trapezoid shaped function. The function trapezoid is divided into three sections: Acceleration, constant speed and deceleration, and will describe the velocity that the robot should be in at any given time.

The problem with that idea is that it lacks in precision, having a tolerance of decimeters. The way to make up for it is to have a closed loop for testing the distance the robot had moved by recalculating the velocities periodically using the current position of the robot as a reference point. But, that solution takes too much computing power that we lack, because of the Raspberry Pi taking it up for image recognition.



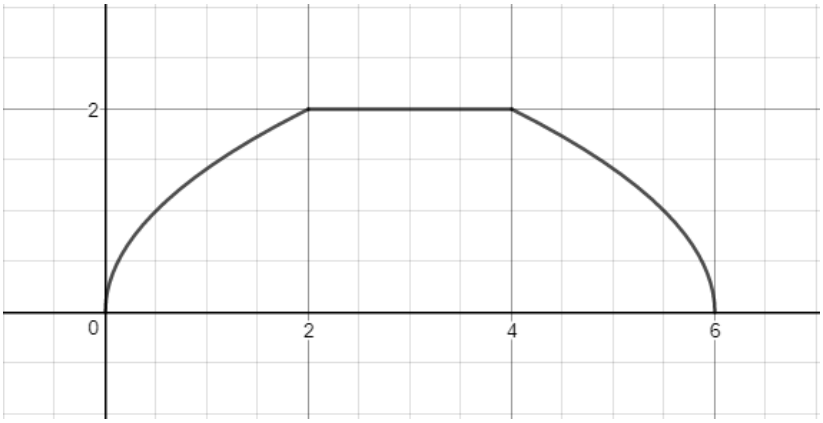
The graph on the left shows us the velocity by time equation, using a constant acceleration of $1 m/s^2$, and a max velocity of $2 m/s$ to travel a total of $6m$.

Velocity by distance

The idea was to use the same formula as above but to use the distance traveled as the parameter for this function. This time we had a tolerance of millimeters, much better than before. By using the distance as our parameter we also solved the problem of having to recalculating the velocities each iteration we did. But, with this new method we faced a new problem – the program was lacking in speed. As the robot got closer to its target, its velocity got smaller and smaller, never reaching its goal.

This time we decided to try and use both methods as described above together, which means using the velocities calculated by the distance, but instead of using $v(x) = v_0 + a \cdot x$ we decided to find a mathematical solution for the function $v(x)$ using constant acceleration.

By doing so we got both benefits from both methods without their accuracy and speed and without their drawbacks.



The graph on the left shows us the velocity by distance equation, using a constant acceleration of $1 m/s^2$, and a max velocity of 2m/s to travel a total distance of 6m.

Calculating the distance using constant acceleration

By using the basic equations of motion with acceleration speed we found an equation that describes the time given a certain position. From there we put our new equation inside the velocity by time equation. As a result, we got the velocity by the distance traveled.

V – velocity **V_0 – initial velocity**

X_0 – initial position

a – acceleration **t – time**

$$v(x) = ?$$

$$v(t) = v_0 + a \cdot t$$

$$\int v(t)dt = x(t) = x_0 + v_0 \cdot t + \frac{1}{2} \cdot a \cdot t^2$$

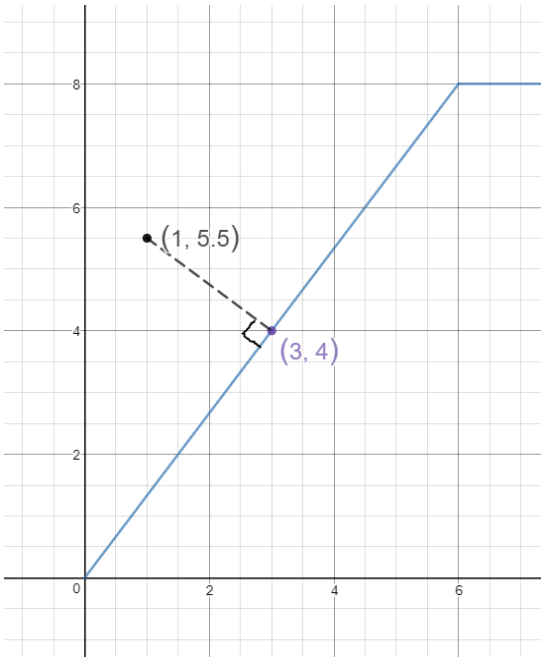
$$t(x) = \frac{-v_0 \pm \sqrt{v_0^2 + 2 \cdot a \cdot \Delta x}}{a}$$

$$v(x) = v(t(x)) = \pm\sqrt{v_0^2 + 2 \cdot a \cdot \Delta x}$$

Stream Path Controller

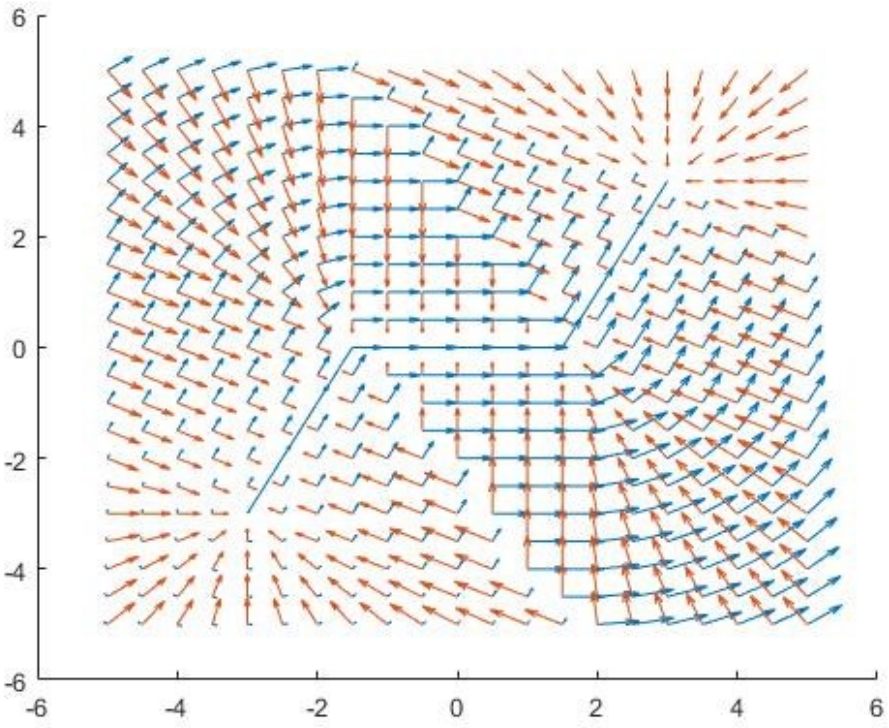
To transfer our calculation into the two dimensional world we had to describe the robot position as two distances. For convenience, instead of using a normal two dimensional grid, we used the distance the robot has traveled along its path and its deviation from the path.

For example, the point (1, 5.5) is exactly 2.5 units away from the path. The point (3, 4) is five units away from the starting point of the path, so in our grid this point will be described as (5, 2.5).



This method of showing our position is useful because as shown above we can easily use both the x distance and the y distance to determine a velocity parallel to the path, and a velocity perpendicular to the path. Using the equation we created, we are getting two new vectors of velocity. By combining the vectors, we get for every point in the field both the desired angle and the desired velocity the robot should drive according to.

In the graph below you can see the path we defined with the waypoints (-3, -3), (-1.5 ,0), (1.5, 0), (3, 3). The blue arrows indicate the wanted velocity of the robot parallel to the path, and the orange arrows indicate the wanted velocity of the robot perpendicular to the path.

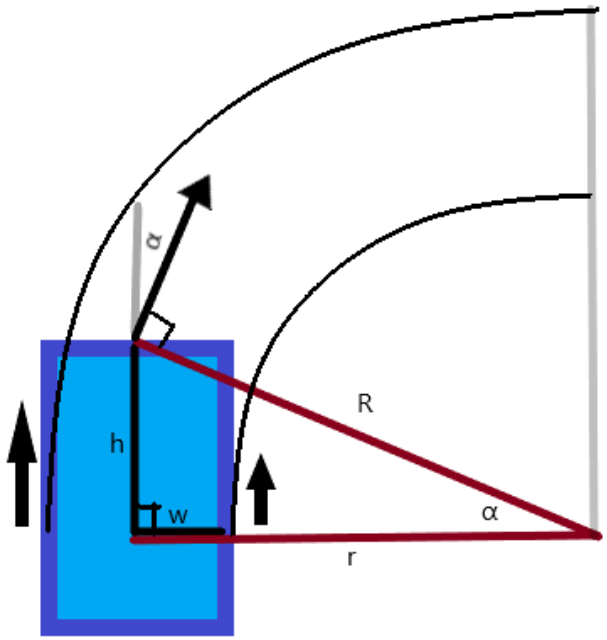


Conversion from the robot velocity into the left and right motors

By using the equation $\omega = v / r$ which should be equal for both sides of the robot, we get that $\omega = v / r = v_r / (r - w) = v_l / (r + w)$.

By using some trigonometry we can easily find R, r and ω. From there it's easy to find what is the velocity necessary for the left side of the robot and for the right side of the robot.

$$R = \frac{h}{sin(\alpha)}$$



$$\omega = \frac{v}{R} = \frac{v \cdot sin(\alpha)}{h}$$

$$r = \frac{h}{tan(\alpha)}$$

$$v_r = \omega \cdot (r - w)$$

$$v_l = \omega \cdot (r + w)$$

$$v_r = v \cdot cos(\alpha) - \frac{w}{h} \cdot v \cdot sin(\alpha)$$

$$v_l = v \cdot cos(\alpha) + \frac{w}{h} \cdot v \cdot sin(\alpha)$$

Elbit Systems

ALUBIN

HIGH TECHNOLOGY WINDOWS

yes PLANET

elg

בית הספר הריאלי העברי בחיפה

משרד החינוך

SOLIDWORKS