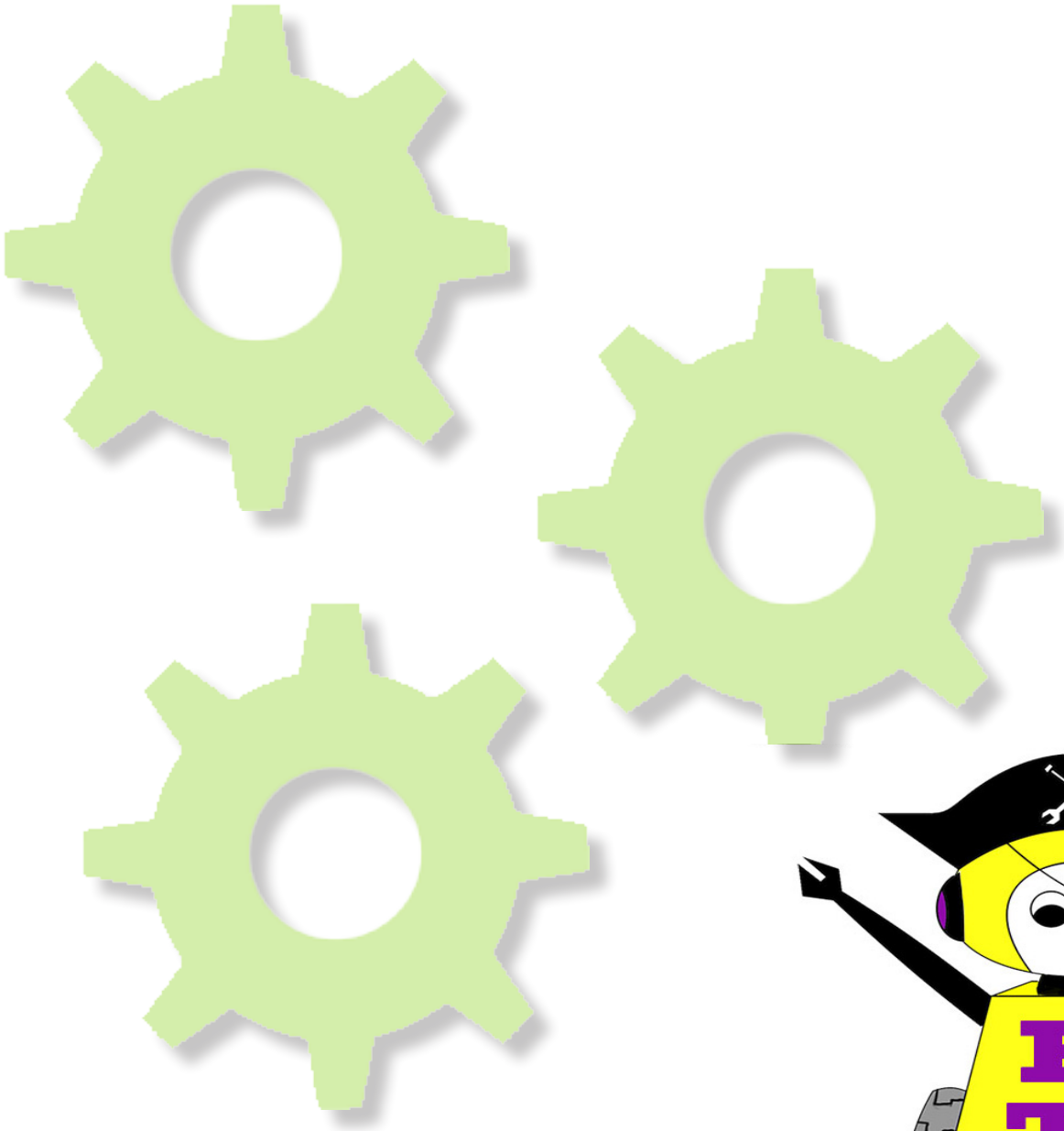
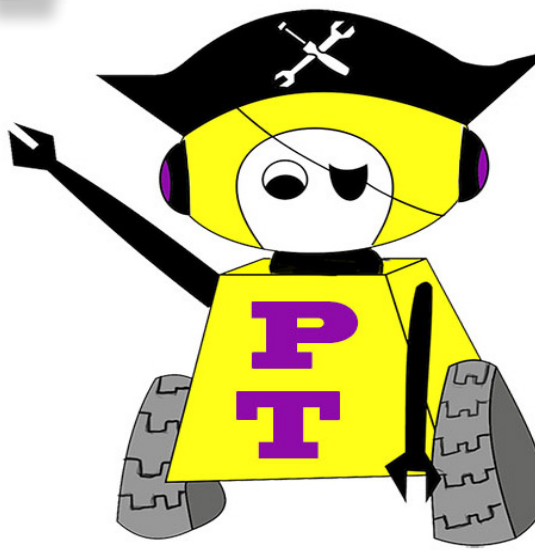
Pirate Tekerz 11109

Ultimate Goal





2020-2021

Pirate Tekerz 11109

Lucas (17yrs) - 4th yr on team:
- Lead programmer
- Human player
- Team captian

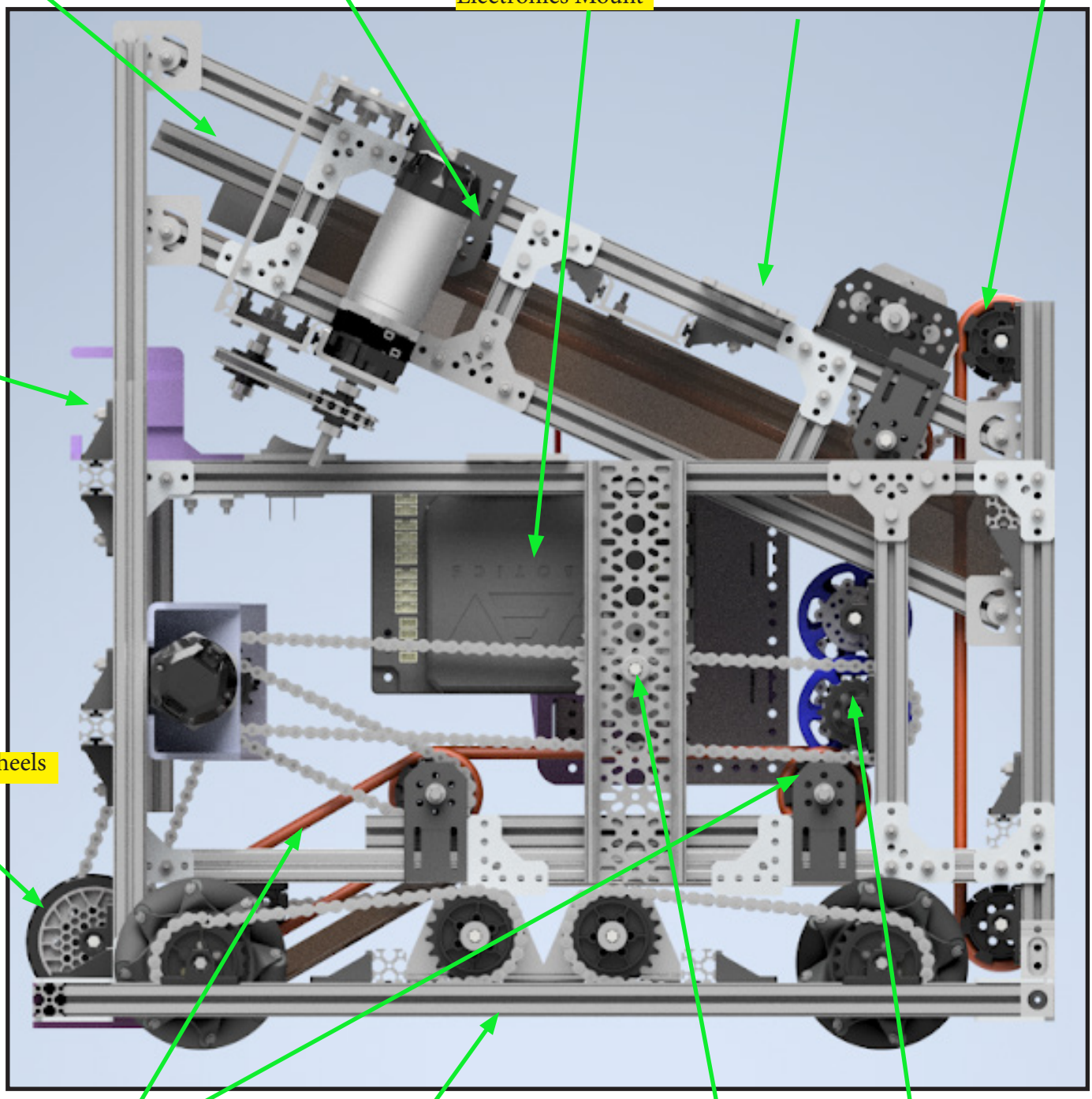
Caleb (15yrs) - 2nd yr on team:
- Driver
- CAD lead
- Portfolio outline and content

Braeden (14yrs) - 2nd yr on team:
- Portfolio design and graphics
- Assembled drivetrain and frame
- Built lower bent polycarbonate and polycord intake system

Finn (15yrs) - 2nd yr on team:
- 3D printer
- Assisted CAD
- Designed battery holder

Nathan (15yrs) - 4th yr on team:
- Assembled shooter, elevator, eletronics
- Tweaked full assembly
- Prototyped and Gathered Data
- Tested the final robot build
- Co-Pilot

Parts of our Robot



Game Strategy

Living in a world of unknown

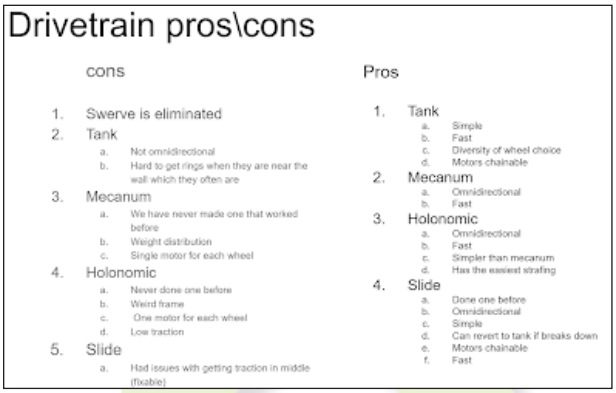
Due to COVID there was a possibility that we would not be able to drive in-person. Our strategy had to incorporate potentially driving alone or remotely. We made sure that our robot would be able to play with other robots if necessary. Some of our solutions were:

- Try to use a remote driving software (Tele-Drive)
- Design a robot that can be driven by one person
- Make a robot able to function in both a remote and traditional game

Our first step in developing the robot was to make a comprehensive strategy of the game. We achieved this by:

- Creating a table to visualize levels of game challenges
- Assigning resultant game scores to each level of difficulty
- Evaluating what robot components were needed to achieve those scores

Robot needs:	Time period	Easy	Medium	Hard
All 3 Shooter Intake Reliable robot	Auto	1. Mid goal 2. Park 18+5=23	1. High goals 2. *Wobble* 3. Park 36+15+5=56	1. *Wobble* 2. Power shot 3. Park 15+45+5=65
2 Wobble goal device	Tele-Op	1. Mid goal (6*3)*4=72	1. High Goals 2. Move wobble (6*3)*6=108	1. High Goals (6*3)*6=108
	End game	1. Mid goal (2*3)*4=24	1. Drop zone 2. High goals (6*3)+20	1. Power shots 2. Drop zone 45+20=65
	Estimated Score:	119	202	238



The next step was to prioritize our robot components.

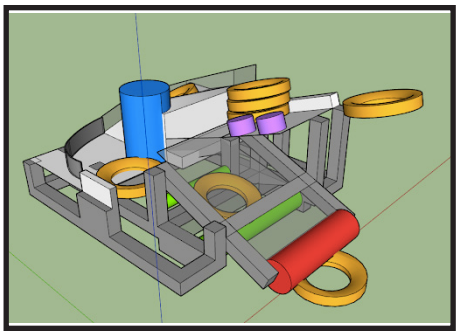
- Shooter -- Would be our highest source of points
- Intake -- The rules guaranteed that rings would be on the floor
- Drivetrain -- We decided omnidirectional drive would be easier to position to get rings
- Auto -- Would bring a high percent of our total points
- Wobble mechanism -- Lower priority due to lower points per difficulty level

Strategic Trade-offs

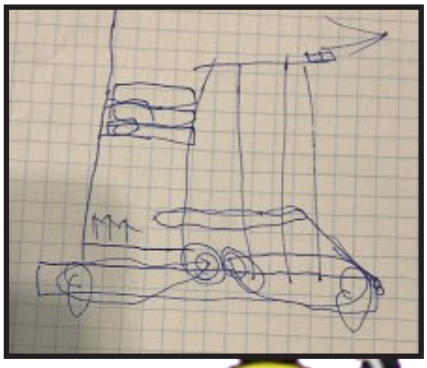
Wobble: We decided that shooting rings would score faster than trying to manipulate the wobble. Our final verdict was that we would ignore the wobble until we had shooting down and then if we had time we would consider it.

Auto Power Shots: We deferred working on power shots as the difficulty of making shots out-weighed the points gained.

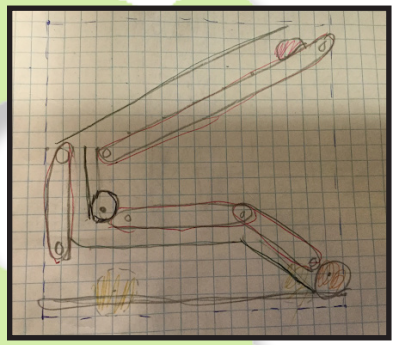
Summary: High goals would be the best and easiest form of making points throughout the entirety of the game. We focused here to ensure greatest potential with the unknown future.



Design

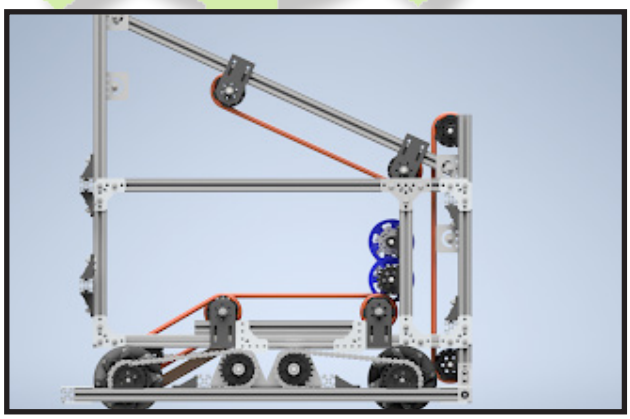


During initial design discussions we came up with three overall ideas: **Polycord**, Stacked ring lifter, continuous semi-circular ramp.



We chose the polycord system because:

- It is simple
- We believed it had the least failure points
- It is easier to manufacture

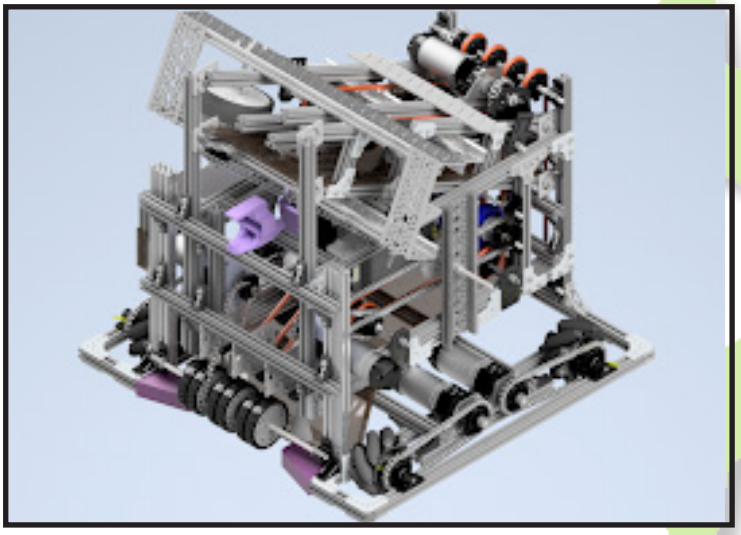


Our initial design was

- Intake and shoot from the same side
- Use polycord loops to move the ring through the entire system
- Use sensors to index rings
- Use mecanum drivetrain
- Use single wheel shooter
- Employ camera to autonomously move

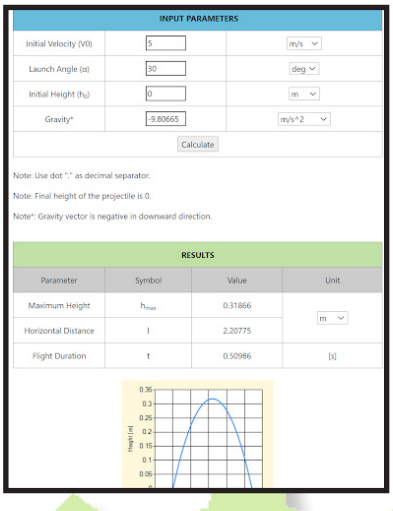
Programing challenges to attempt

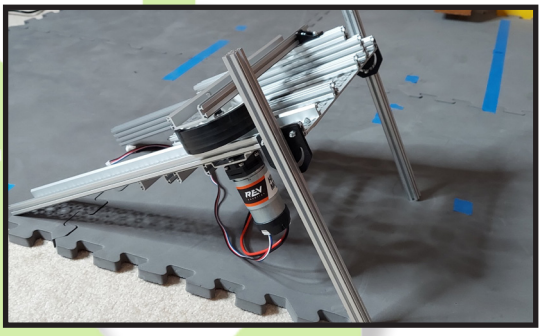
- Auto indexing
- Auto aim
- Auto shoot
- Omnidirectional drivetrain.
- Vision target navigation



Prototyping

We created a prototype shooter, to test flight characteristics of the ring. Using an online object trajectory calculator, we determined a base angle and speed projections to start our tests.



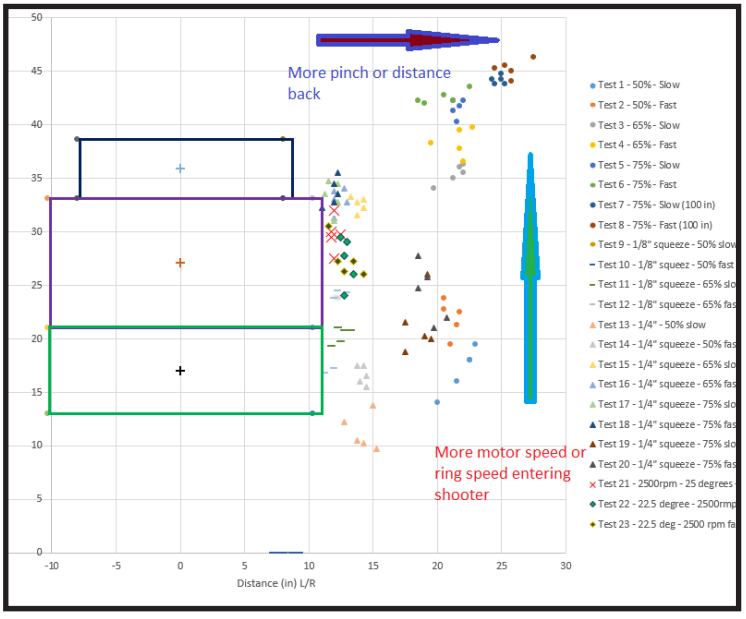


Data was gathered by shooting rings against a cardboard mock-up of the goals. We recorded over 120 shots in groups of 5 while changing the following four variables:

- Shooter angle
- Shooter distance from target
- Motor speed
- Ring squeeze

We compiled our data about shot groupings, relative shot position, and actual shot position into an Excel spreadsheet and created a data table and scatter plot. Because of our thorough testing and analysis, the first three shots from our completed robot scored in the high goal.

Test	Input Speed	Motor Speed	Distance from wal	Shot Iterration	landed distance from Wall	Inches L/R from center	distance from ground	L/R from center true	distance from ground
base	---	---	---	0	0	0	0	0	0
1	Slow	50%	64 in	1	0	20	8	20	14
1	Slow	50%	64 in	2	0	22.5	12	22.5	18
1	Slow	50%	64 in	3	0	21.5	10	21.5	16
1	Slow	50%	64 in	4	0	22.5	12	22.5	18
1	Slow	50%	64 in	5	0	23	13.5	23	19.5
2	Fast	50%	64 in	1	0	21	13.5	21	19.5
2	Fast	50%	64 in	2	0	21.5	15.25	21.5	21.25
2	Fast	50%	64 in	3	0	20.5	16.75	20.5	22.75
2	Fast	50%	64 in	4	0	21.75	16.5	21.75	22.5
2	Fast	50%	64 in	5	0	20.5	17.75	20.5	23.75
3	Slow	65%	64 in	1	0	19.75	28	19.75	34
3	Slow	65%	64 in	2	0	21.25	29	21.25	35
3	Slow	65%	64 in	3	0	21.75	30	21.75	36
3	Slow	65%	64 in	4	0	22	30.25	22	36.25
3	Slow	65%	64 in	5	0	22	29.5	22	35.5
4	Fast	65%	64 in	1	0	19.5	32.25	19.5	38.25
4	Fast	65%	64 in	2	0	22	30.5	22	36.5
4	Fast	65%	64 in	3	0	21.75	33.5	21.75	39.5
4	Fast	65%	64 in	4	0	21.75	31.75	21.75	37.75
4	Fast	65%	64 in	5	0	22.75	33.75	22.75	39.75
5	Slow	75%	64 in	1	0	21.5	34.25	21.5	40.25
5	Slow	75%	64 in	2	0	21.25	36.25	21.25	42.25
5	Slow	75%	64 in	3	0	22	36.25	22	42.25
5	Slow	75%	64 in	4	0	21.25	35.25	21.25	41.25
5	Slow	75%	64 in	5	0	21.75	35.75	21.75	41.75
6	Fast	75%	64 in	1	0	18.5	36.25	18.5	42.25
6	Fast	75%	64 in	2	0	19	36	19	42
6	Fast	75%	64 in	3	0	20.5	36.75	20.5	42.75
6	Fast	75%	64 in	4	0	22.5	37.5	22.5	43.5



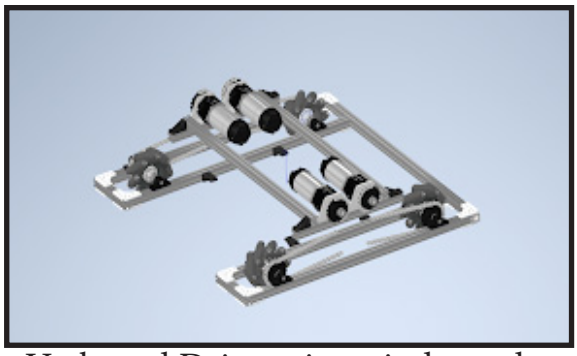
CAD process

For the first time as a team, we utilized Autodesk inventor to completely design and imagine our robot prior to construction.

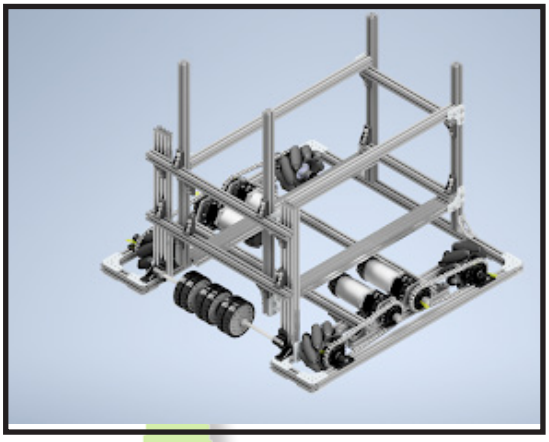
- Inventor was chosen over OnShape and Fusion 360 because we deemed them insufficient
- We had to expand our knowledge of CAD beyond elementary capabilities
- COVID created the need to be 100% remote in design and building
- CAD enabled us to locate problems without a tangible robot

Our Robot has a modular design letting us build parts of the robot while the rest was being finalized. This allowed for separate modules to be built independently at individual students' homes.

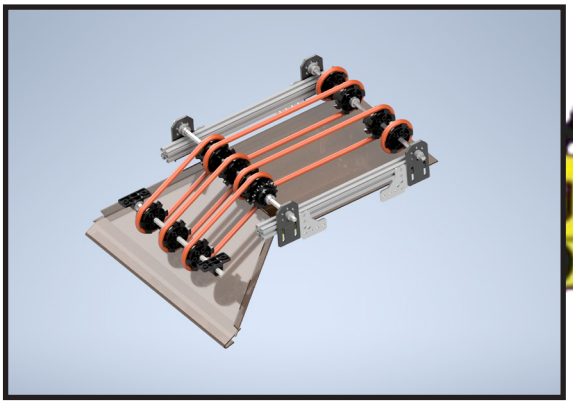
The Robot is built from more than 7 independant modules:



U-shaped Drivetrain w. independent power system.

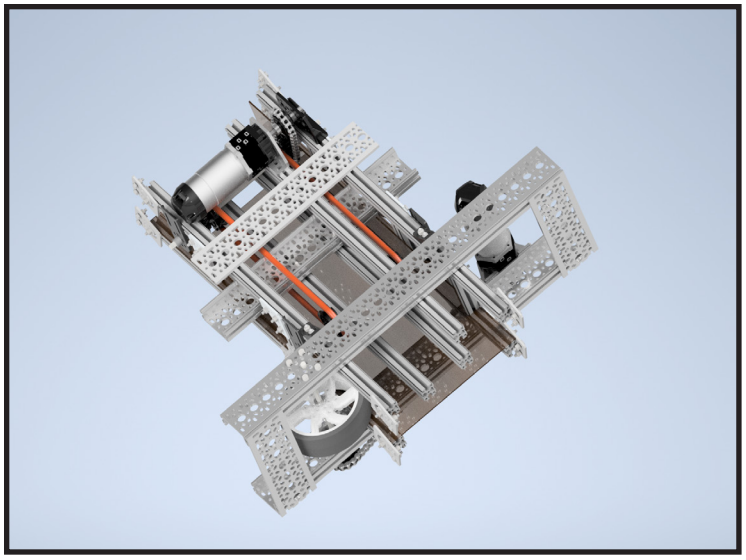


The main frame that holds all the other modules



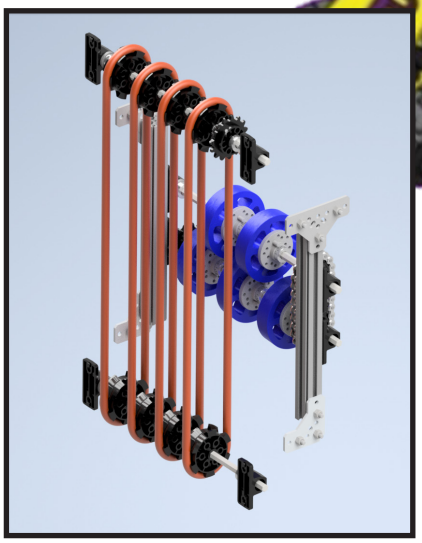
Lower Poly Cord system w. adjustable axles and independent adjustable polycarb floor.

Modular Design

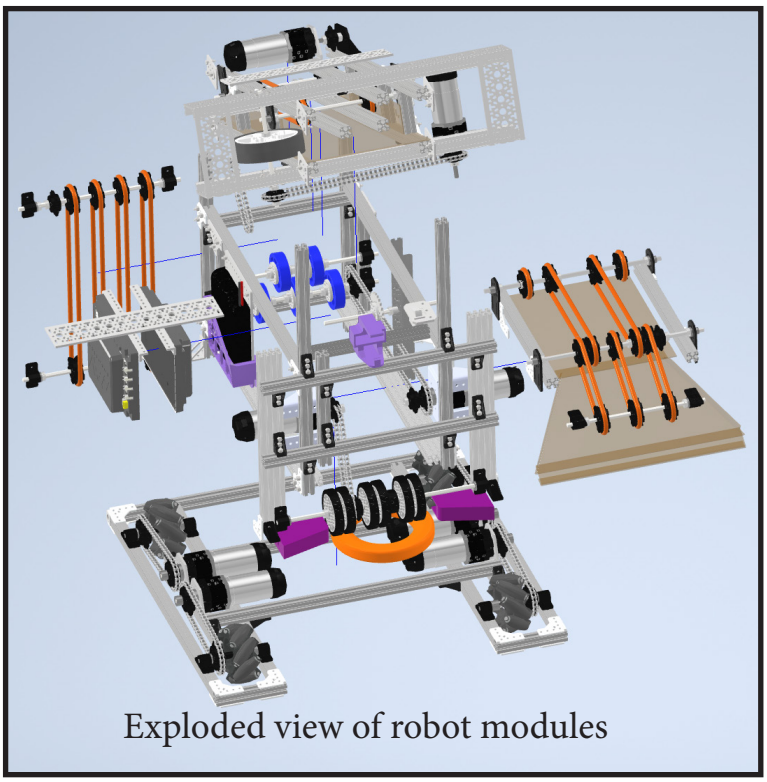


Shooter module designed for variance of shooter angle and ring squeeze.

Shooter module contains shooter wheel, ramp, polycord feeder, and frame subsystems. All which adjust independently of each other.



Elevator and Helper wheels transitions ring from lower to upper systems.



Exploded view of robot modules

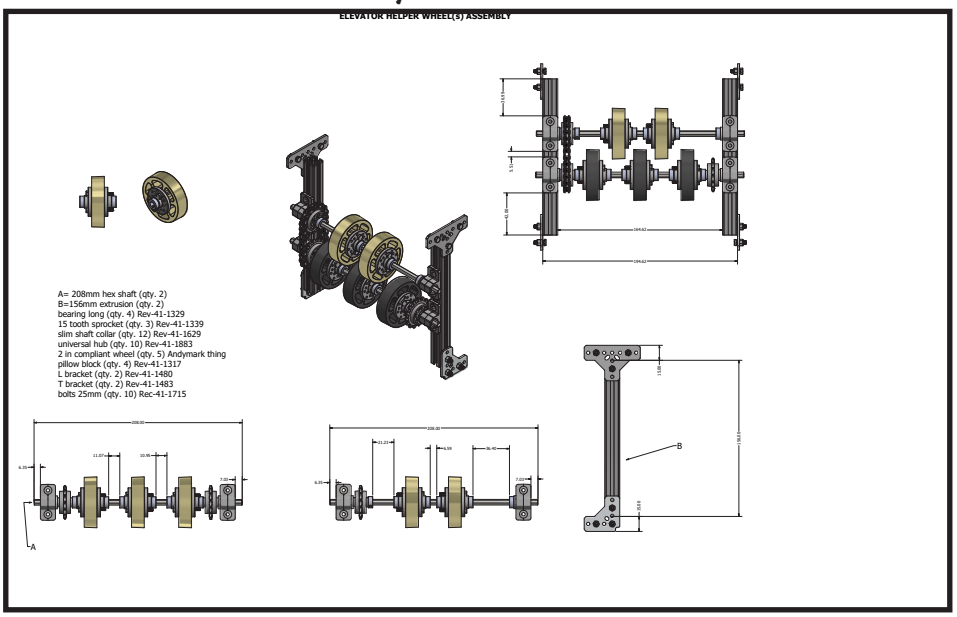
Other subsystems are

- Electronics Sensor and Camera systems
- Intake wheel system

Summary: We had to build while still CADing so we created tolerant, malleable, designs to accommodate post assembly real-world problems. This was handy as issues were found throughout competition weeks. Modules allowed us to re-move, fix, adjust problems areas smoothly.

Building instructions specifics

Due to COVID and 4-H rules we were completely unable to build in person. To solve this we produced individual module assembly instructions.



Assembly instructions include:

- part identification list
- detailed step-by-step instructions
- CAD precision dimensions

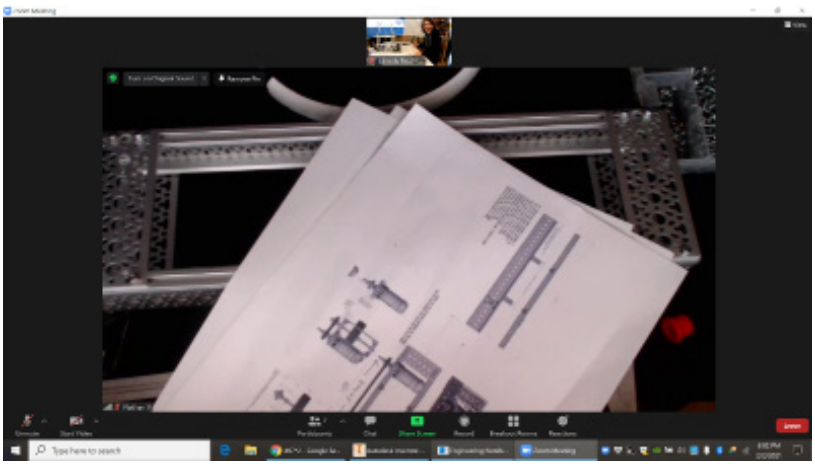
Priority		Part Type	Quantity	Sizes	Cut Type
1	Intake	5mm Hex Shaft	1 ea	269.2 mm	Outside Cut
1	Intake	Double Extrusion	2 ea	265 mm	Inside Cut
1	Base Frame	Double Extrusion	2 ea	384.5 mm Could cut 3 from 1m and 1 from	Inside Cut
1	Base Frame	Single Extrusion	2 ea	384.5 mm	Inside Cut
1	Base Frame	Single Extrusion	2 ea	370mm	Outside Cut
1	Base Frame	Single Extrusion	4 ea	300mm	Outside Cut
1	Base Frame	Single Extrusion	2 ea	194.62 mm	Inside Cut
2	Shooter Frame	Single Extrusion	4 ea	420 mm	NO CUT
2	Shooter Frame	Single Extrusion	3 ea	394 mm	Outside Cut
2	Shooter Frame	Single Extrusion	4 ea	54.4 mm	Inside Cut
2	Shooter Frame	C-channel 15mm	1 ea	248 mm	NO CUT

It was more efficient to have one person manufacture parts. To maintain accuracy we developed a master cut list.

Master cut list included:

- priority & module
- part type & quantity
- size
- cut type (inside/outside)

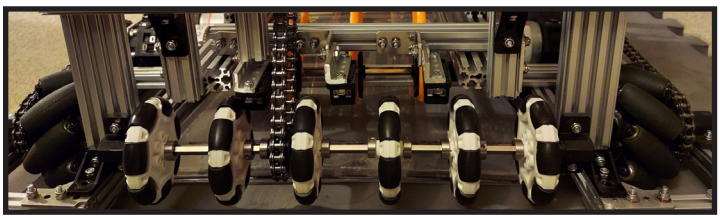
Students received assembly packages. They were comprised of pre-cut cut and manufactured parts, fasteners, and assembly instruction sheets. This allowed a student to completely build a module and pass it off for later use.



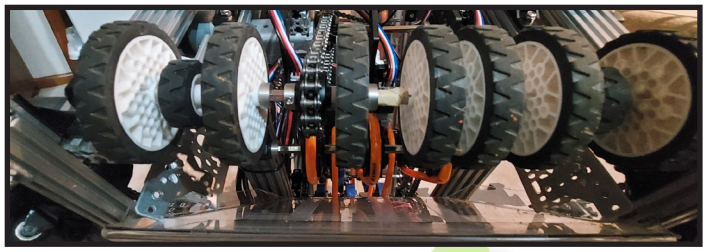
Testing

After our initial build and through our first competition we encountered several issues that needed improvement.

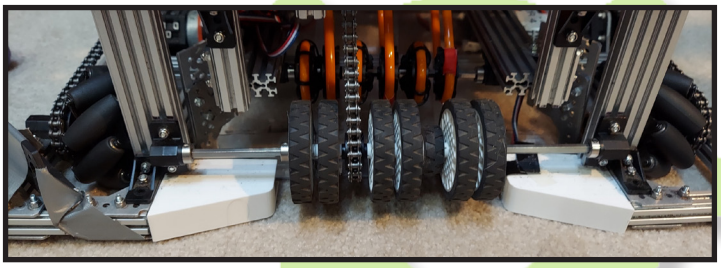
Intake Changes



Intake #1: We initially tried 60 mm omni wheels, but discovered that the rings would occasionally get stuck and damaged by segments on the wheels.



Intake #2: We learned that using 60 mm traction wheels increased grip with no ring damage. The rings continued to get stuck occasionally when they entered under areas between wheels and sometimes they jammed in the side flanks. We added metal flanges to the sides which solved some jamming.

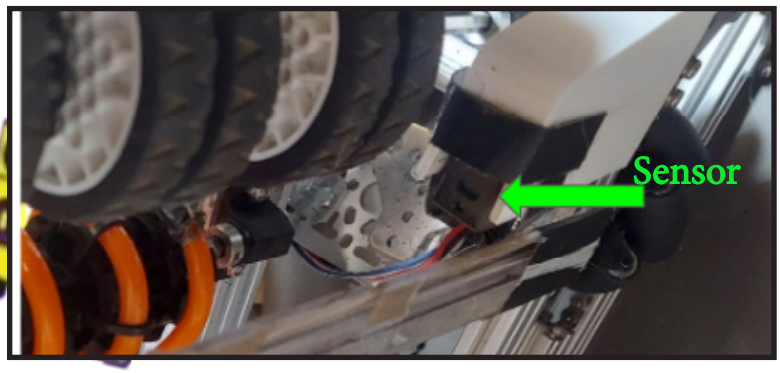


Intake #3: Finally, We adjusted wheel spacing to close gaps and added 3D printed side wedges to guide rings into the intake. This solved our stuck ring problem. The intake went from 11 inch wide to 5.5 inches.

Sensor Control Changes

Intake Sensor Setup #1: Because our intake was 11in wide we needed 3 sensors. While it did work during our first competition, we discovered that the sensors would sometimes become misaligned and not detect rings consistently.

Intake Sensor Setup #2: After switching to a 5.5 inch intake, we found that adding a sensor to the side and mounted inside the wedge gave more consistent ring detection. The top three sensors were removed.

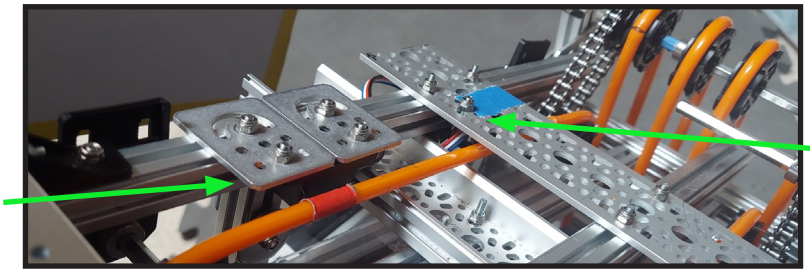


More Testing

Shooter Ramp Sensors

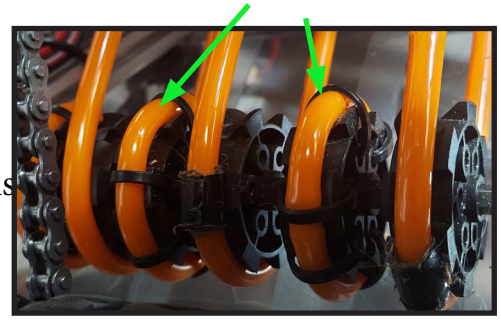
Sensor Setup #1: Initially a sensor was used to detect and stop a ring to prepare it for shooting autonomously or manually. Sensor readings were not always accurate (varied up to 12 mm). After research and testing, we learned that adding reflective bicycle tape reduced the variance to 1 or 2mm.

Sensor Setup #2: We had an issue where one ring was too close to another ring. When the first ring was shot, the second ring had already passed the sensor. We put in a second sensor to detect this issue and move the ring back to the first sensor.

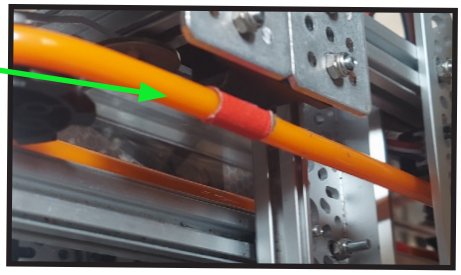


Changes to Improve Ring Flow

Initially, our rings would move slowly or get stuck in gaps in our polycord system. After testing, we found several changes improved the flow.

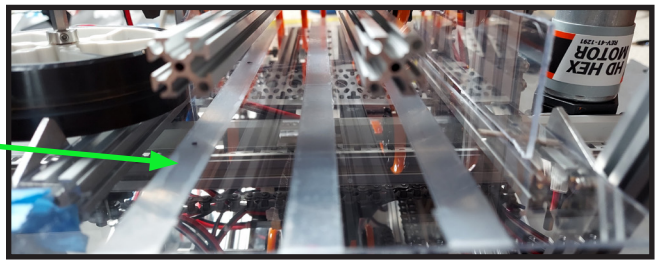


Polycord Wheels: To fill gaps we designed our own wheels by wrapping polycord around pulleys and securing with zip ties. It also provided more surface area for pulling.



Taped Polycord: To enhance polycord grab on rings we wrapped gaffers tape around the polycord every 4-5in.

Motor Torque: Our goal was to move the ring through the system as fast as we could but still have enough torque to get through smoothly. We tested a 5:1 planetary motor gear ratio, 9:1, 15:1 and ended up using a 20:1.



UMHW Tape: UHMW tape was added to our polycarb ramps to reduce friction by 2/3rds, allowing the rings to slip easier over the polycarb.

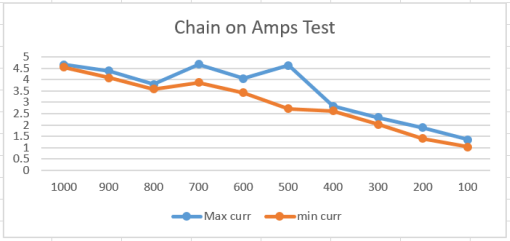
Tested to Exhaustion

Shooter Motor Gearing:

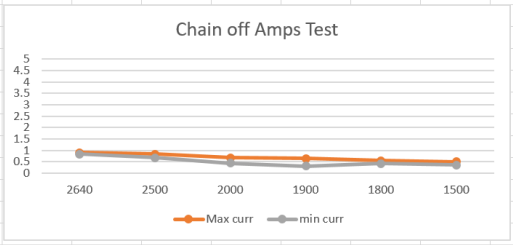
Shooter #1: Initially our shooter motor had a 1:1 gear ratio. During prototype testing we discovered, running at 75% motor speed and ¼ ring squeeze, the ring did not rise up to the high goal most of the time.

Shooter #2: Our goal was to try 1:2 or 1:3 ratio to increase wheel speed. Plastic gears in these ranges were out of stock, all we had were metal gears. Turns out the metal gear wouldn't work with a horizontal chain due to friction and gravity. We used the only plastic gears we had which were (15:40) ratio.
Testing didn't show the ring going higher, so we recorded our telemetry and entered it into a spreadsheet.

Set speed	Max speed	Min speed	Max curr	min curr
1100	1000	960	4.654	4.548
900	900	880	4.384	4.089
800	800	780	3.798	3.586
700	700	660	4.678	3.867
600	600	560	4.045	3.422
500	500	480	4.629	2.711
400	400	380	2.835	2.617
300	300	280	2.336	2.03
200	200	180	1.885	1.403
100	100	80	1.36	1.024



Set speed	Max speed	Min speed	Max curr	min curr
3000	2640	2580	0.893	0.834
2500	2500	2480	0.841	0.677
2000	2000	1980	0.674	0.43
1900	1900	1880	0.637	0.3
1800	1800	1780	0.546	0.411
1500	1500	1480	0.495	0.36



We discovered:

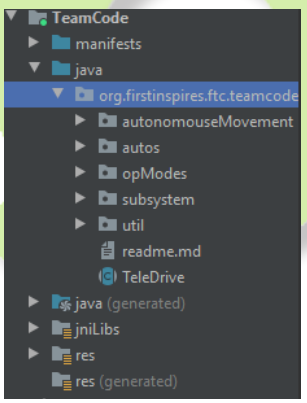
1. With the chain on spinning the wheel we couldn't get higher than 1000 RPM This meant the motor was maxing the electrical current it could use.
2. Since 900 rpm was our target goal, we would have to run the motor at 90% to achieve the goal. This causes the motor to overheat and could damage it.
3. This didn't make sense until we discovered we had a lot of friction in our shooter system. Such as: rubbing metal and misaligned shafts

Shooter #3: After fixing our friction issues and getting 26 tooth gears in we switched to a 15:26 gear ratio. To help the wheel spin better we used lithium grease on the bearings holding the wheel.

This gear ratio was very successful in scoring high goal shots.

Programming

- Our program is structured in a multi-class system in which each separate class deals with only one thing and its methods are then useable in others
- Our main separation is between subsystems and opmodes
- We use about 8 different subsystems for our teleop program

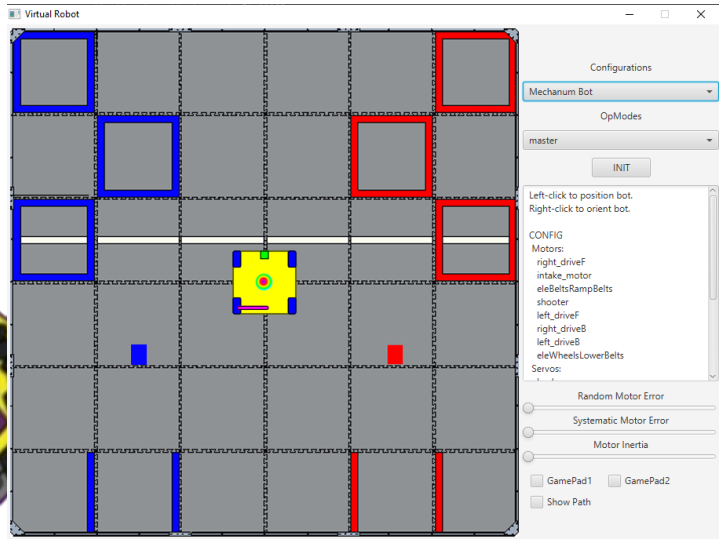


Driver enhancement and operation

•Mecanum drive code to efficiently maneuver and position
•Autonomous ring shooting using distance sensors and a state machine
•One button precision shooting positioning from up 2 feet away
•Automatic indexing upon intake using a senors and polycord motor manipulation
•Dual speed drivetrain modes
•Toggle button features to decrease driver error
•Two different drive modes; field based and robot based

Autonomous period movements

•Pid based turning that is quick and accurate within a degree
•Time based strafing with curve calculation for increased accuracy
•Encoder based forward and reverse movements to within 1in
•Vision target based positioning
i.e. (rotation, strafing, and forward and reverse positions)



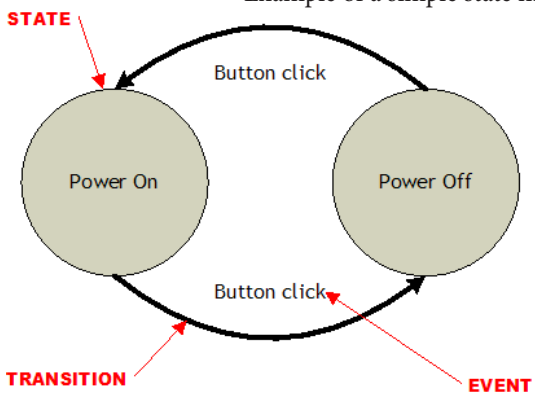
To test all of our drive systems we used a virtual test environment by team 8397 shown by this picture.

State Machines

State machines are a form of logic that allows us to store a state, and with that state, do a specific action. Our code contains two state machines:

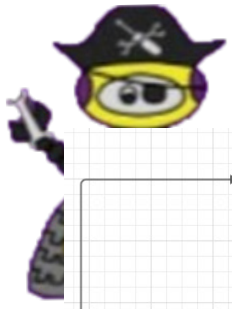
- Shooting
- Lining up

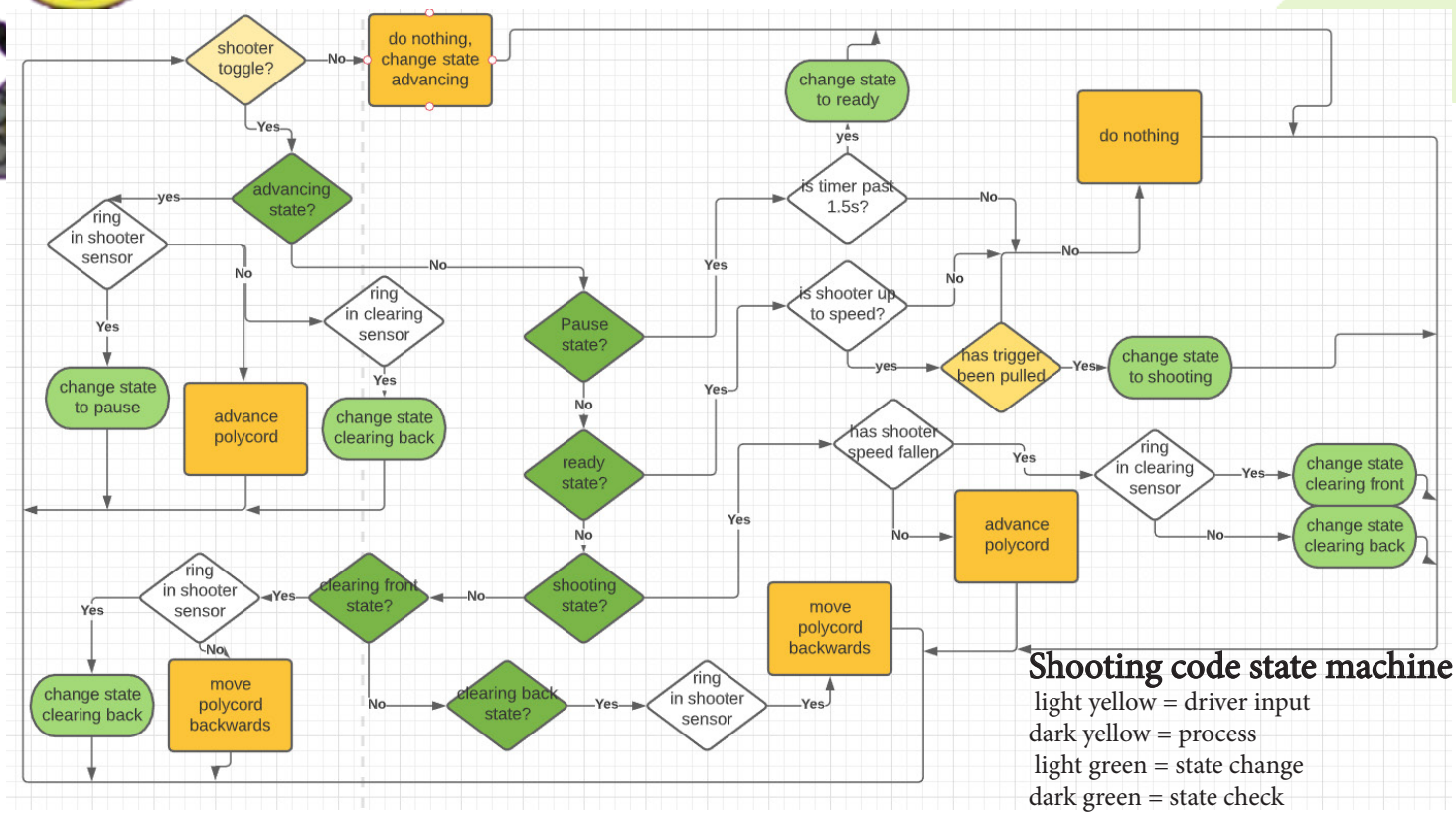
Example of a simple state machine



•**SHOOTING**: For our shooting we use the state machine to move the ring into a position, and track our shooter speed to only shoot when fully up to speed. The rings move by sampling two different distance sensors telling us whether a ring is out of position allowing us to dynamically reposition rings into a predetermined shooting position.

•**LINE UP**: For our line up we use the state machine to track which movements have been completed and transfer to the new state. At the end we set it so we cannot enter the next state until we press a button allowing us to easily use the line up whenever we want.





Shooting code state machine key:

light yellow = driver input
dark yellow = process
light green = state change
dark green = state check
white = decision

Programming

Autonomous

Our autonomous code is fairly simple with the goal being three high goals and navigation. While we have been running matches our autonomous has gone through some stages.

League 1 Competition: No state machine for shooting, basic drive forward with no adaptive positioning
League 2 Competition: Automated shooting using state machine for more consistent shots, basic drive forward with no adaptive positioning
Inter-League Competition: Automated shooting using state machine for more consistent shots, adaptive positioning using camera based inputs

Strafing

Strafing was an issue for our team as encoder based movement did not work consistantly. this led us to utilizing a time based system. To increase the accuracy of our time based system we used a curve calculation using data gathered from tests to modify the time

```
public double strafeCurve(double distance){
    return -0.756*distance*distance+52.689*distance+148.74;
}
```

Curve Calculation

Forwards/Backwards

We use an encoder based code for our forward and backwards motions. We track the encoder position and shut the motors off upon reaching our target.

```
public void encoderDrive(double speed, double leftInches, double rightInches, double timeOutS)
    double leftTargetTicks = leftInches * COUNTS_PER_INCH;
    double rightTargetTicks = rightInches * COUNTS_PER_INCH;
    double startOffset = drive.getLeftTicks();
```

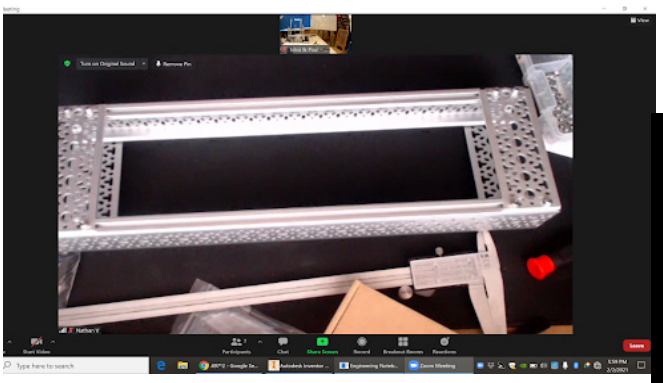
Calculation in encoder drive

Rotating

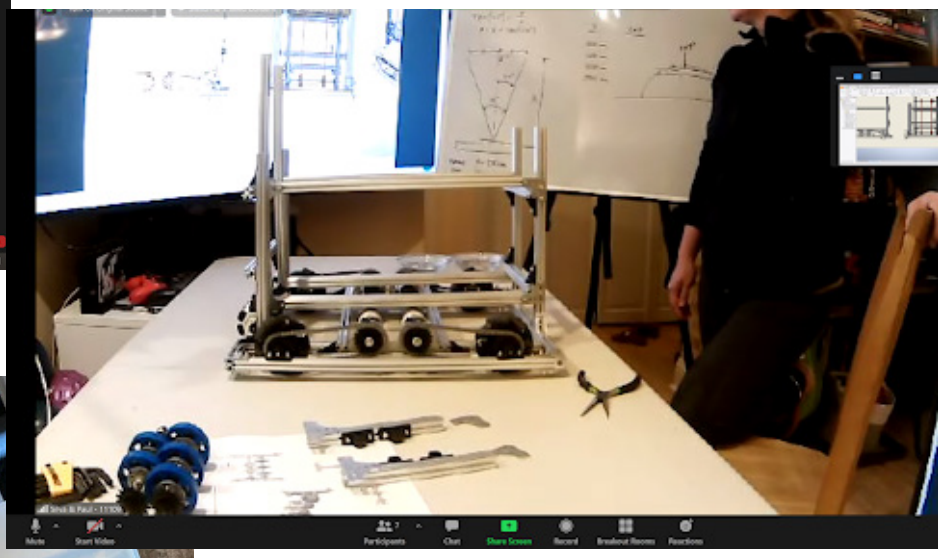
Our PID based turning during line up uses the information gathered from the webcam in tandem with the internal IMU to rotate to the exact heading we want within a degree.

```
if (Math.abs(87 - webCam.camHeading()) > 1) {
    IMU.rotate( targetAngle: 87 - webCam.camHeading(),  power: 0.4,  timeOutS: 2);
}
liningState = liningUpMode.distancing;
linningUpTime.reset();
```

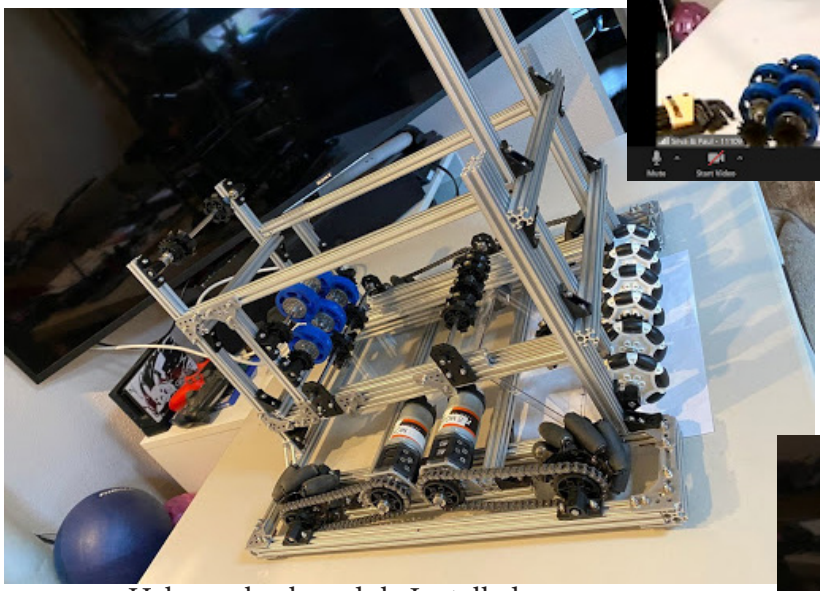
Adaptive Turn Calculation



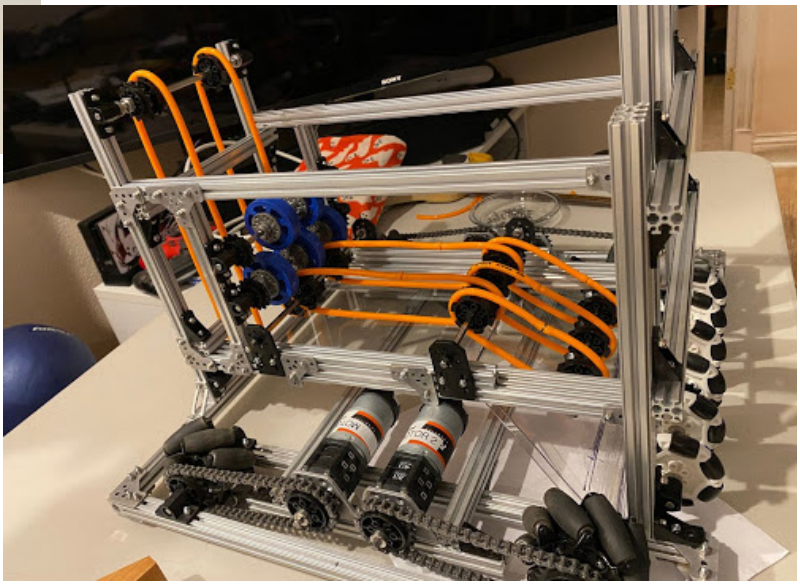
Shooter Box Creation



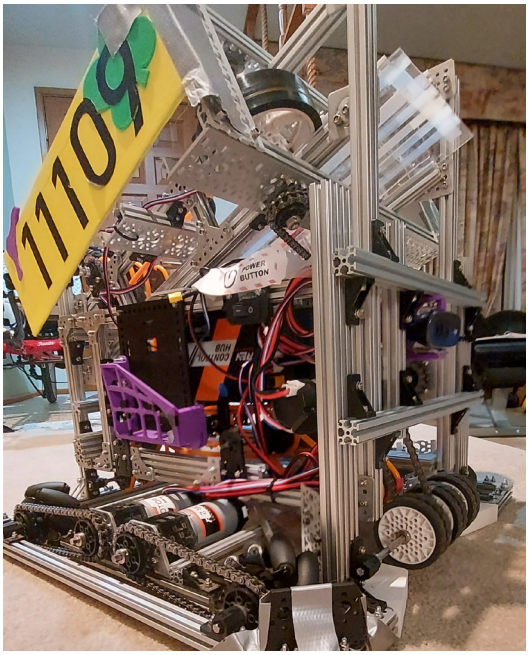
Braeden Building over zoom collaboration



Helper wheel module Installed



Lower Polycord DONE!



Up to Date Robot