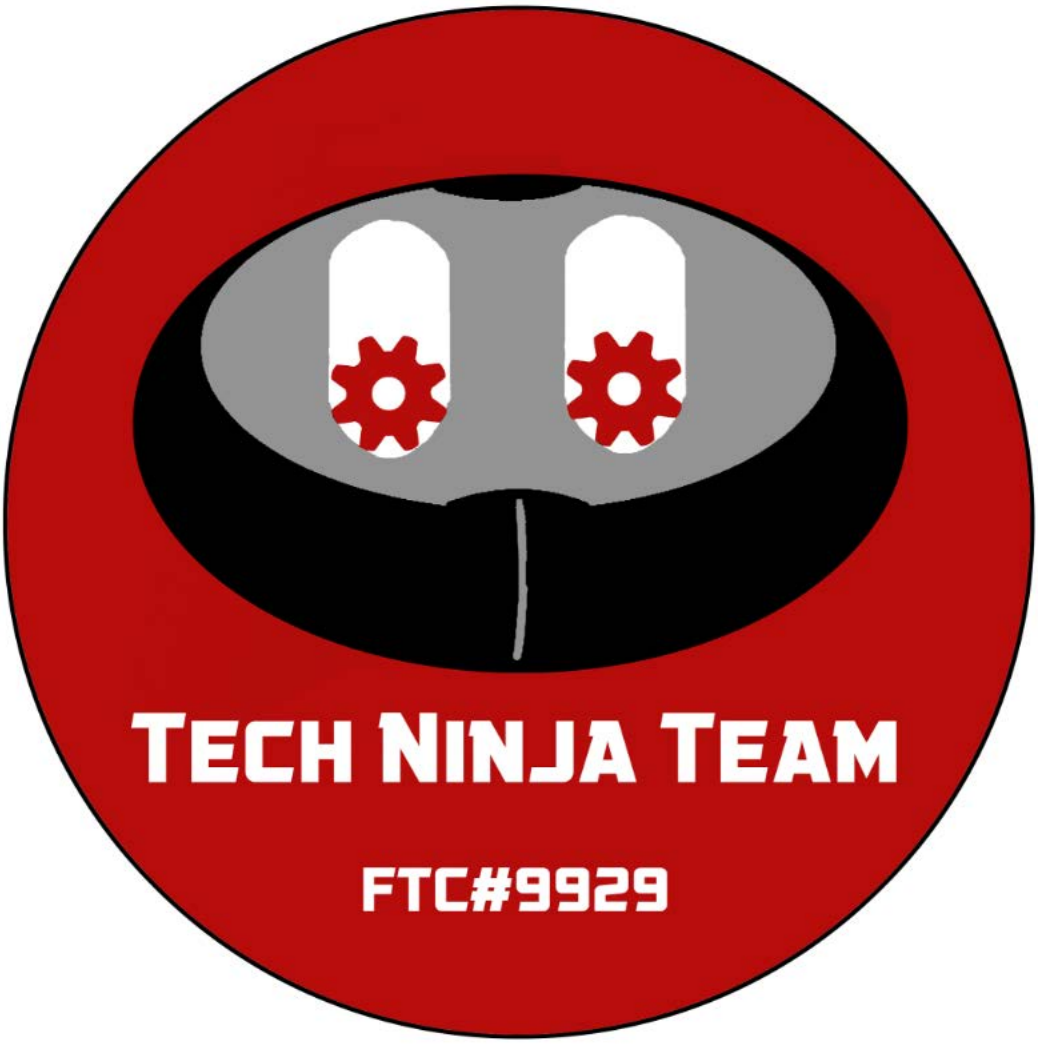


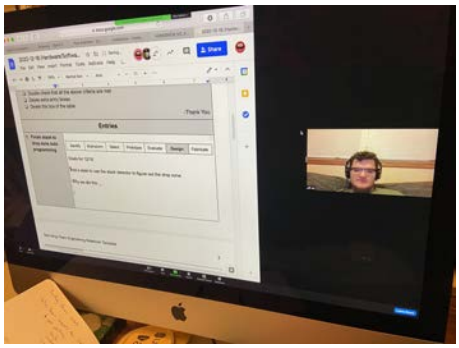
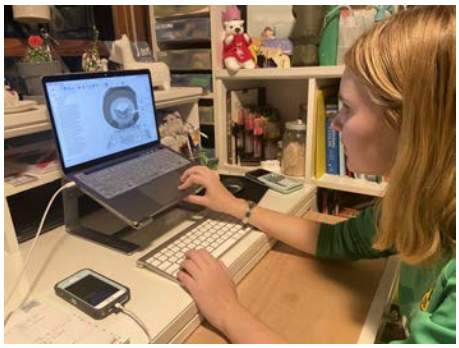
Engineering Portfolio 2020/2021

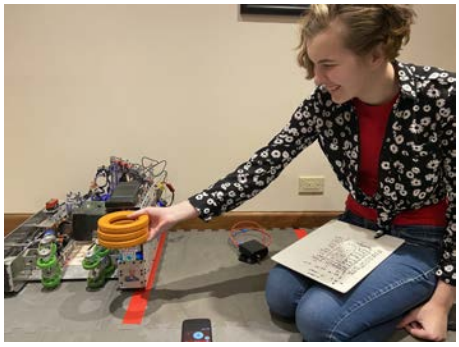
Team Overview - Passing on The Ninja Way

This is the sixth FTC season for the Tech Ninja Team. We are a team of thirteen (and one intern). Five of our team members are graduating seniors and will be off to college in the fall. Half of the team started in FIRST Lego League. This season we have worked on Ultimate Goal without leaving our hometowns of Flossmoor and Homewood. We've designed, prototyped, fabricated, programmed, and competed out of our workshop in a converted garage at the Homewood Science Center and from our living rooms, basements and family rooms via Zoom.

This year we focused on passing on the ninja way. Ultimate Goal is the last season for multiple team members as they will be graduating from the program as seniors. Passing on the ninja way means multiple things. Firstly, it means ensuring the next generations of ninjas are ready to take over the roles left by the seniors. Secondly, is that the skills we have learned over multiple seasons are passed onto the next generation. One way we are doing this is by having the senior programmers lead the software team meetings. Another is how the build team has all learned CAD which allowed them to work together through the design process over Zoom meetings to make the final robot.

At the end of the day being a ninja is about upholding our values: acting now - and iterating, communicating effectively, expecting and embracing change, and taking risks without panicking. For us the pandemic was an unexpected challenge, but we feel we are stronger as a team because of how it encouraged us to really live up to our values, and stretch ourselves in new directions that we might not otherwise have.

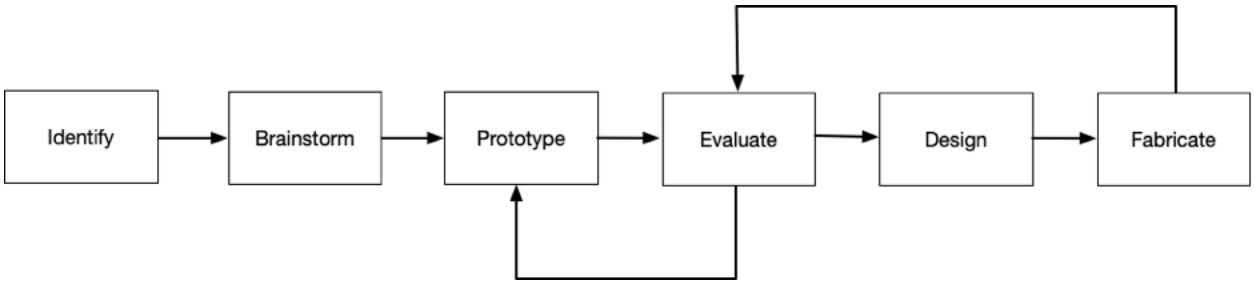




How We Work

Like many FTC teams, we have a build team, a software and controls team and drive teams. Team members choose a sub-team specialization based on their interests and skills, but sometimes because of need or particular interest they may work on parts of the robot that are not their “usual” team.

We follow this engineering design process:



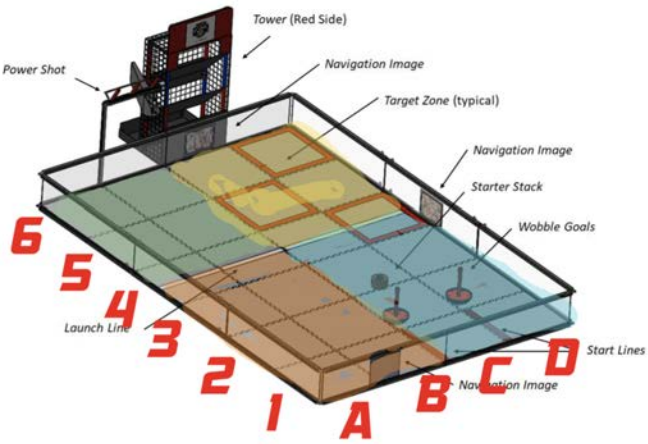
- Identify – identify the problem to be solved
- Brainstorm – brainstorm solutions
- Prototype – quickly build physical or mathematical models to evaluate the brainstormed solutions
- Evaluate – run experiments to see if the solution works
- Design – design a solution to use on the robot based on the prototype(s) and the evaluation
- Fabricate – machine, assemble or program the solution that was designed. Evaluate whether it meets the requirement.

Strategy for Ultimate Goal

We met during kick-off weekend, watched the reveal video with the rest of the Illinois FTC teams on the state kickoff, and then set to work learning the rules of the game so that we could come up with a strategy for playing it.

Auto Strategy

1. We started figuring out our possible strategies by finding all possible solutions to the autonomous challenge. (all starting positions, penalties, routes, etc.)
2. We then determined the max auto score, from that we chose a reasonable auto score goal of 56 points. We wanted to achieve this by placing the wobble in the target zone (15 pts), shooting 3 high goals (36 pts), parking (5 pts).
3. To polish our strategy we anticipated possible issues by listing errors in previous years' strategies in order to avoid them and we did a tabletop simulation to see how our strategy might look in the real world.



Tele-op Strategy

- We determined our strategy options for tele-op in a similar way to our auto strategy, we started by listing all possibilities for scoring.
- We determined that we would try to shoot rings for the entirety of tele-op, with a "cycle time" of 13 seconds (worst case), giving us 9 cycles per tele-op. We discussed giving up 1 "cycle" to move the remaining wobble goal past the scoring line to make it eligible for scoring in end game.

End Game Strategy

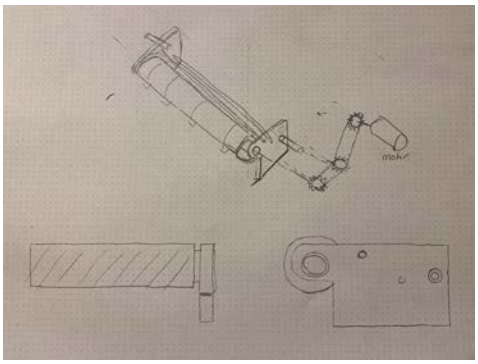
- We knew we wanted to use the wobble goals in end game, so we focused our discussion on that

Overall Strategy

- For Auto we ultimately decided to move the wobble goal, shoot three rings at the high goal, and park
- For Tele-Op we decided to focus on quick shooting strategies to optimize the number of rings we could score, this heavily influenced the design of both the launcher and the intake and helped the programming team create a state machine that ensured those two mechanisms worked well together.
- We wanted to focus on scoring the wobble goals during end game, which led us to design an efficient and simple wobble goal mechanism to help us complete this task.

Robot Design and Fabrication for Ultimate Goal

Intake Design and Evolution

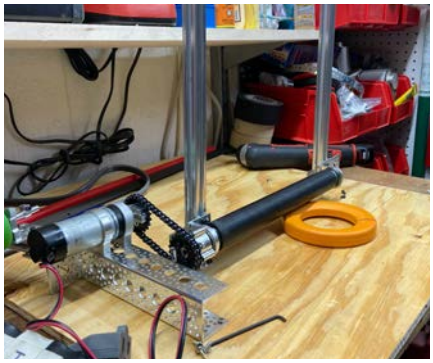


Design and Evolution

- The intake started as a single stage roller intake and was developed into a 3 stage compliant wheel intake
- The intake was designed to use only one motor with pulleys.
- The intake was created to act like the rollers in a vacuum cleaner to sweep the rings into the robot

Testing

- We tested many speed, angles, and heights of the intake rollers to optimize speed and efficiency
- We tested both a long surgical tubing covered roller and compliant wheels, we decided on compliant wheels since they could give a little and grab the rings more securely and at more angles.



Fabrication

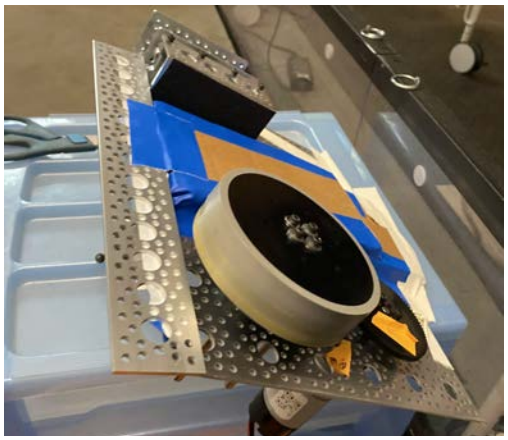
- We designed the final mechanism completely in CAD and used the design to fabricate a couple prototypes and the final mechanism
- We fabricated this part both at home and in the workshop

Final Mechanism

- The final mechanism was implemented on the robot and tweaked to work with the other mechanisms
- The intake guides the rings up a ramp into the hopper after sweeping them off the field.

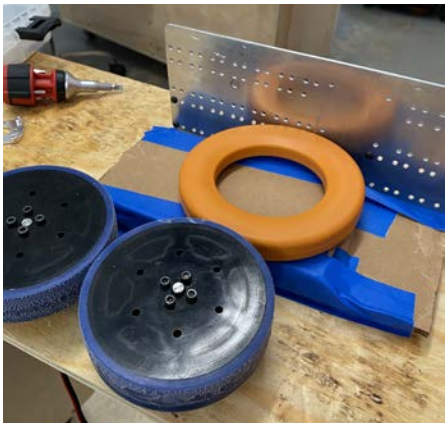
Launcher

Launcher Design and Evolution



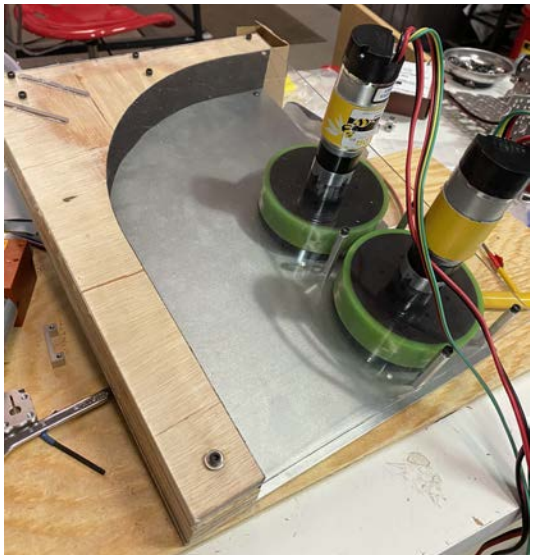
Design and Evolution

- We started by building prototypes for a wheeled launcher and a spring launcher
- After coming up with a base for the wheeled launcher, we started experimenting and trying to improve the design



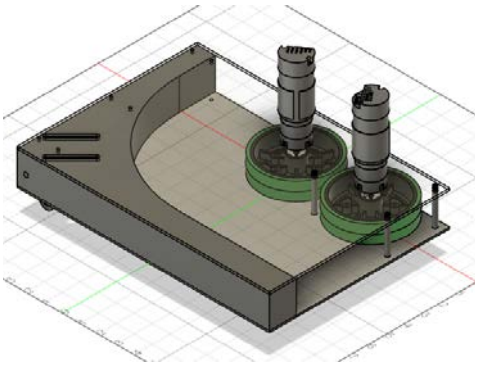
Testing

- After testing the one wheel on the wheeled launcher, we realized to reach the tower, we need two wheels
- We considered if two wheels or a curved wall would get us greater distance -- realized we could use both



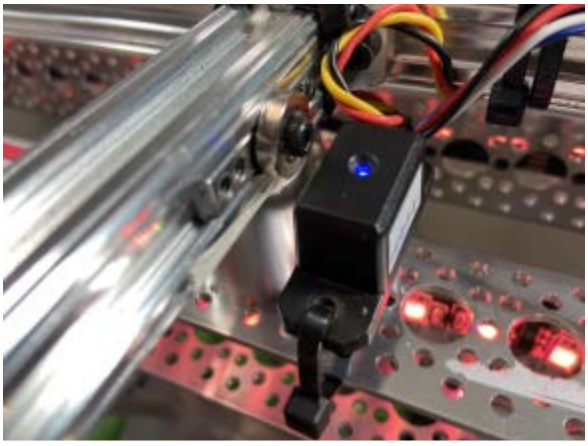
Final Mechanism

- We designed the final mechanism completely in CAD and used the design to construct the final mechanism



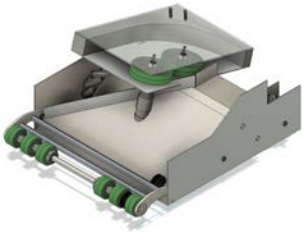
Wobble Goal

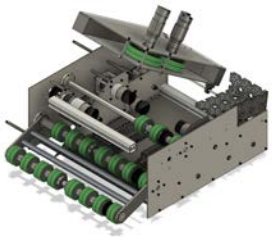
Based on our strategy analysis - we determined that we should have a very simple wobble goal mechanism. We settled on a swinging arm, driven by a 256:1 reduction to both have controllable speed, force, and not require feed-forward to hold a wobble goal in any position. The motor was put onto a set of bevel gears which connects to a piece of shaft. The shaft connects to another piece of shaft which has the swinging arm on it. We decided on a swinging arm because it puts less strain on the servo which controls how the arm grips onto the wobble goal. We also added limit switches and hard stops to make sure it operated consistently, and was able to be moved easily during autonomous.

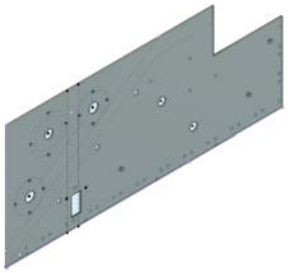


Wobble Goal Limit Switch and Hard Stop

Drivebase





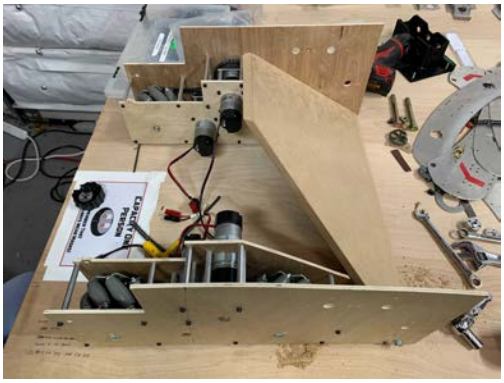


Much of the drivebase prototyping happened in CAD over Zoom sessions. The design evolved to hold the intake and launcher together, and later the side plates were extended to become structure for the wobble goal mechanism to give it more strength and stability, rather than using channel and fasteners.

CNC

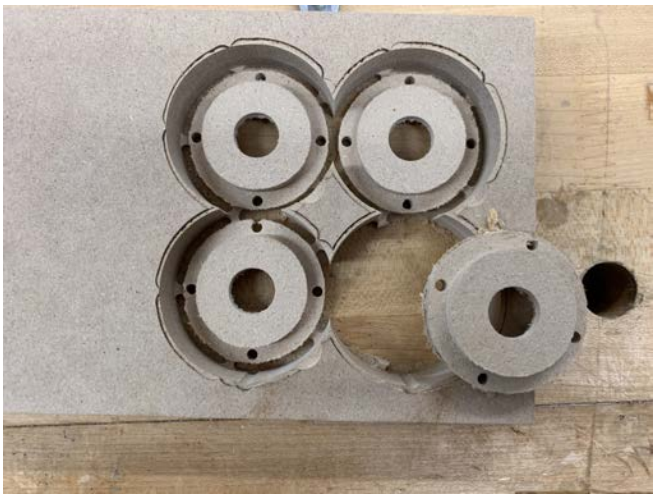
Sheet goods are vital in our robot design and a great deal of our machining time consists of them. We needed six aluminum plates for the drive base (two pairs of side plates, a belly pan, and a back plate for stability), another one for the launcher, an acrylic plate for the top of the launcher, and two polycarbonate plates for the intake. We spent a lot of time learning how to better use our CNC machine this season.

The most useful prototyping sheet material is 1/8” plywood. It is approximately the thickness of other sheet goods, is relatively soft, and produces small, thin chips. It can be easily modified with extra holes and cuts after initially machined, making it excellent for prototyping and gathering practical feedback on our designs. We cut our initial drive base part designs out of plywood for the aforementioned reasons and gathered feedback and information on where to put new holes, what geometry needed to change, and what



holes to move or remove.

We ended up cutting a lot more aluminum than we have in prior seasons - and learned a lot more about feeds, speeds and the hardness and heat-treatability of aluminum grades - 5052 is soft and tended to gum up our cutters.

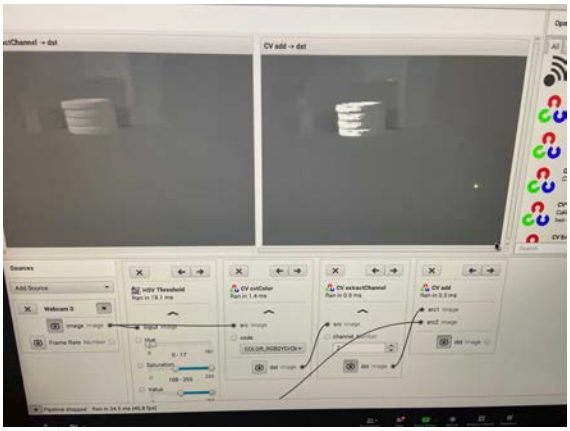


This season marked another first, where we machined thicker parts - bearing housings for the intake. We first machined them out of MDF as a prototype, and later out of 6061 aluminum. 6061 is special because it is harder than the 5052 we used for sheet goods, and it is heat treatable. This means it cuts into nice, small chips, and we can cut it much faster and more efficiently than 5052. We could make more aggressive cuts, clear larger amounts of material, and cut much cleaner in 6061 than in 5052 aluminum. Although 6061 is the more expensive of the two, we are considering using sheets of it for next season due to the ease of manufacturing it.

Software for Ultimate Goal

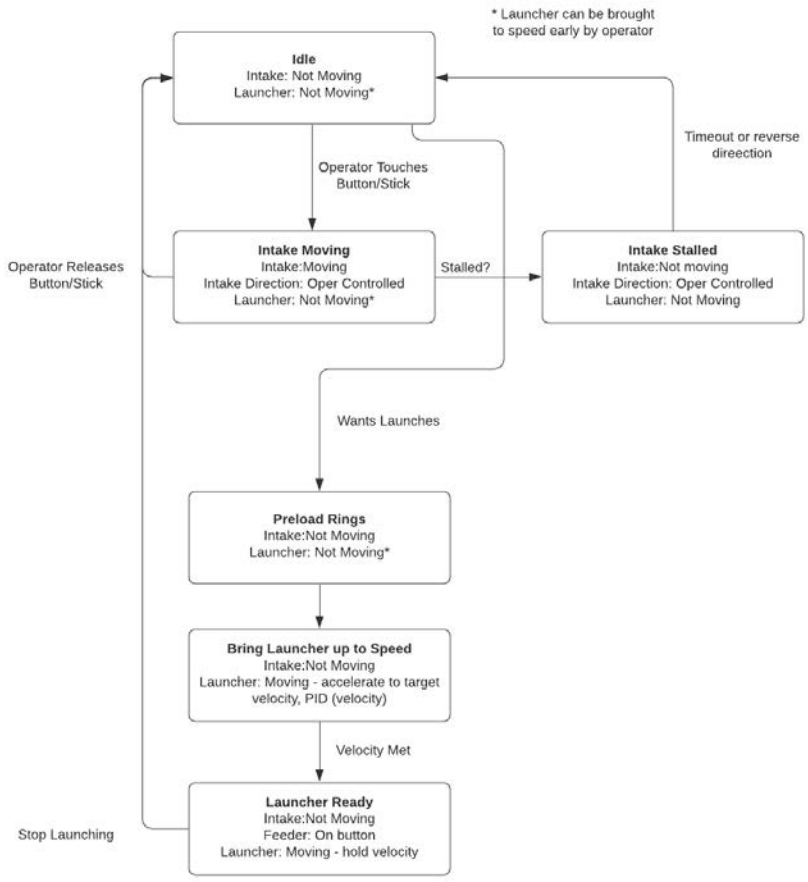
Vision system

Our vision system is built as an OpenCV pipeline and used in the autonomous period to detect the starter stack size. The software team designed the system using GRIP. GRIP made it possible for the team to build the stack detector before we had a robot as well as let everyone on the software team participate in building it, because we could run it on a laptop and share it via Zoom while we were prototyping. Our detector is capable of detecting the ring stack in under 500ms. It does this by using pixel measurement of the bounding box formed from detecting the color orange. The height measurement of this box is then used to make a decision of how many rings are in the stack seen by the camera.



State machines

Our team utilizes state machines in both autonomous and tele-op. We use them because they allow us to automate a series of tasks, which are broken into states.



The state machines used in tele-op also allow switching between complete driver control (open loop) to automated assistance (closed loop) for mechanisms that require multiple steps to operate. This automation allows us to streamline a set of multiple, repetitive tasks to one button, or prevent errors if the human operators were to ask for things to happen at the wrong time or in the wrong order.

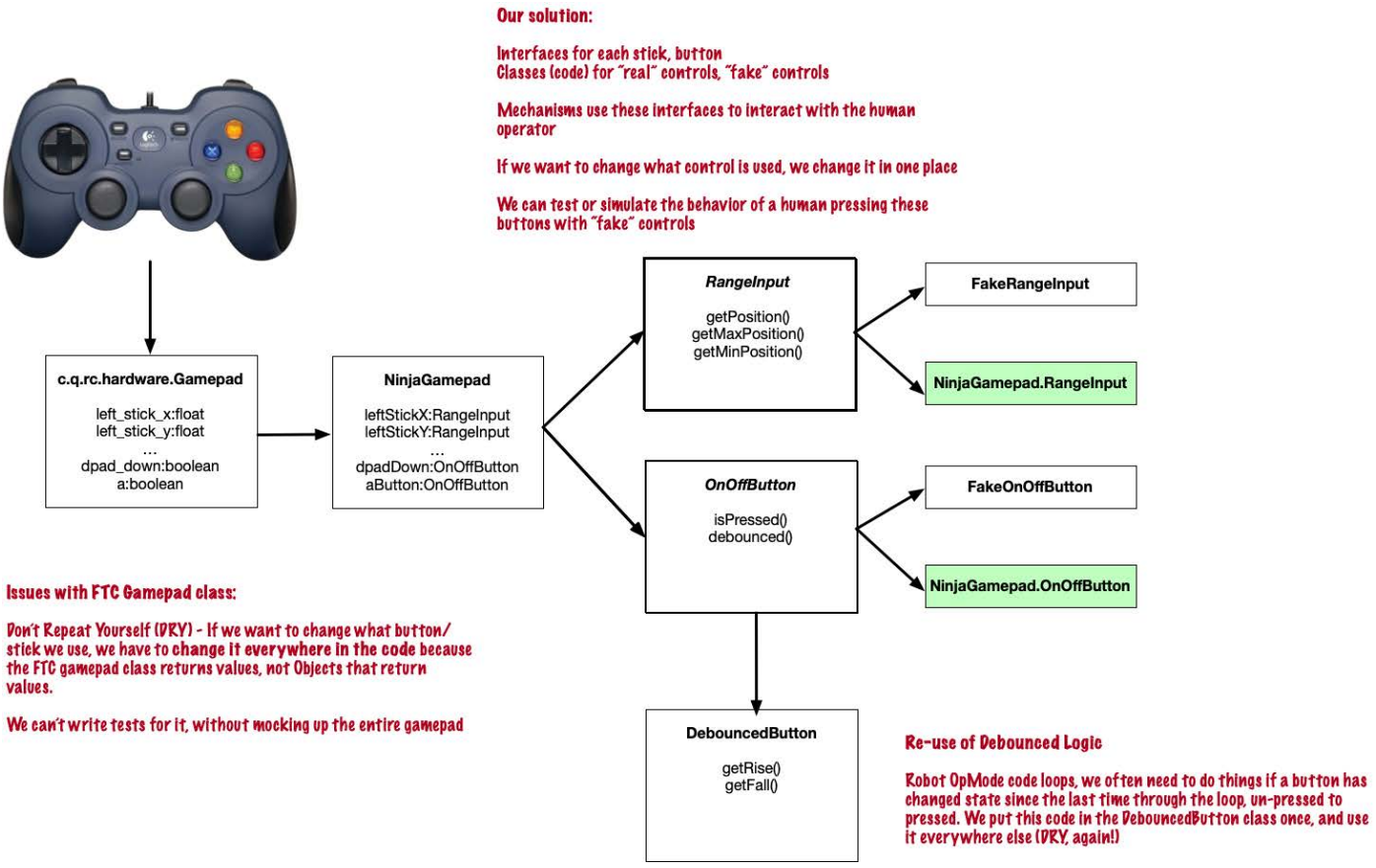
An example of such a set of tasks this season is when the drive team transitions from intaking rings, to launching them. Our code first makes sure the hopper holding the rings tilts back to be aligned with the launcher, starts the launcher wheels spinning, and uses the encoders to measure their angular velocity. The state machine will not let the operator launch rings until all of the steps have happened, and the launcher wheels are at the required velocity. The state machine also sets the color and pattern of the LEDs on the robot to let the drive team know what state the launcher is in.

In addition to making tasks easier and safer to complete, the computer's reaction time is a lot faster than a human's. This allows us to steal back milliseconds throughout the match, which can add up to more time to allow the team to score more points.

But sometimes the sensors our code depends on break or become miscalibrated. When this happens the driver can take full control of the robot using the “unsafe” buttons, which temporarily turn off the limits and safeties we use to automate our code. This means a faulty sensor, like an encoder in the launcher failing, or limit switch in the wobble goal mechanism doesn’t mean that the mechanism it is attached to is inoperable (both things have happened, more than once!).

Drive Team Control Flexibility

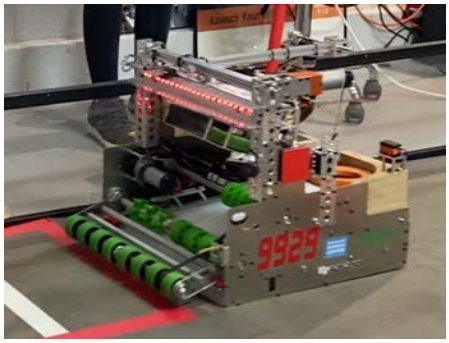
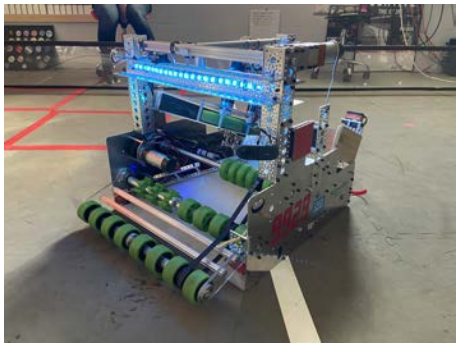
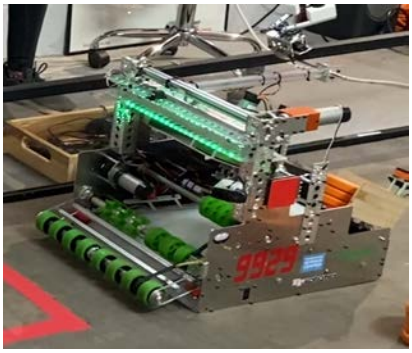
We have added to the FTC SDK's driver controls with our own set of classes. We have classes, not just for the gamepad, but for the buttons and sticks on the gamepad, as well as ways to convert between the two types of controls. Rather than referencing the gamepad in our code, our mechanisms and state machines reference the button or joystick objects. This makes it simple to change what button does what in the code, as where it is set is only a singular line of code. This simplicity is useful for optimizing the gamepad for the drivers and the tasks they must perform during a match. These classes are part of the TntFtcCore library which we share publicly for use by other teams (see the Outreach section).



Signalling the Drive Team with LEDs on the Robot

To better communicate the state of the robot with the drive team this year, we added a LED signalling system. The set of LED lights, as shown below, inform the driver of the different states the launcher is in. The red light which pulses is the signal to the driver that the launcher is idle (it is also one of our team colors). The green and aqua lights signals to the driver that the launcher is up to speed and what angle the launcher is aiming. This allows our drive team to know what the robot is doing - without looking at the driver controller phone.

It also helps with debugging because it allows us to see what the robot thinks is happening based on the software in real time as opposed to looking back at telemetry and logs. Although telemetry is still helpful, the signaling system allows us to narrow down where to look for bugs before we even look there.

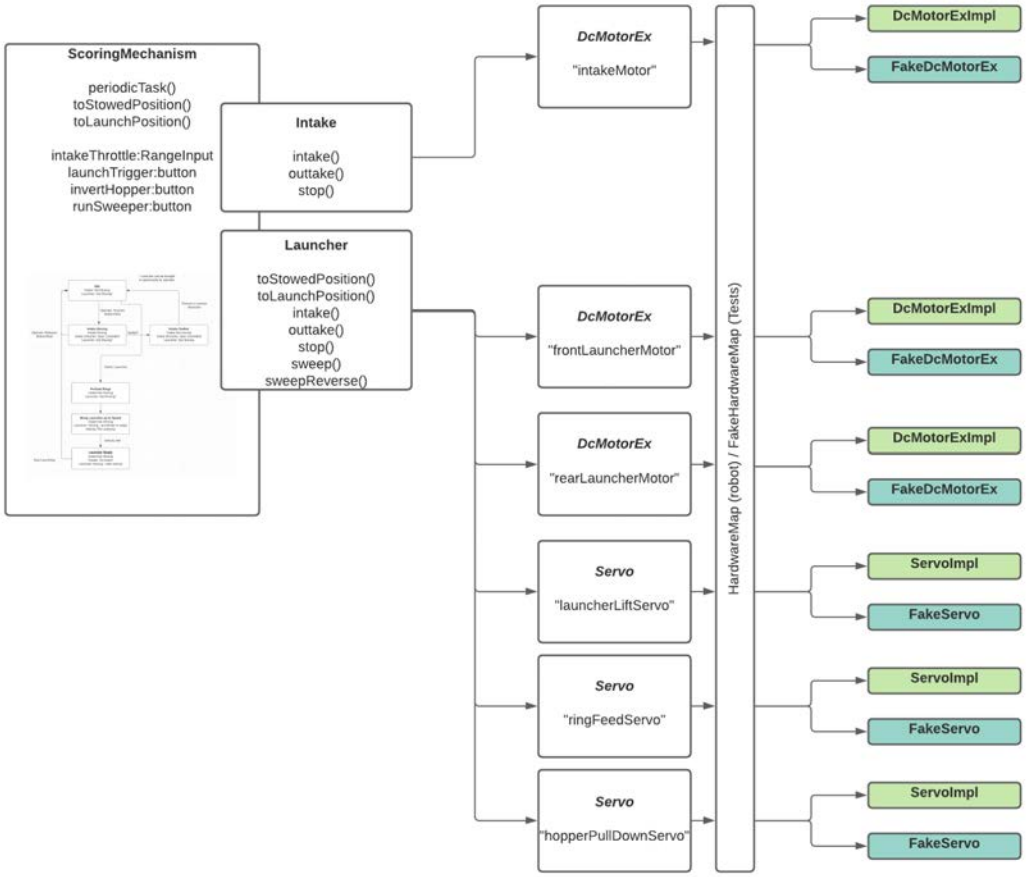
		
Launcher Idle	Launcher @ Speed - Powershot Angle	Launcher @ Speed High Goal Angle

Unit testing

Unit testing is our method of testing our robot code when the software team doesn't have direct access to the robot. It allows us to run the code in a simulated environment. This is done through the use of fakes, which is code that mimics the input and output of real life robot parts such as motors and servos. We then program these fakes to go to certain positions or states to check in the output from our code is what is expected. This allows us to find structure and variable related bugs before we can run the robot with the code.

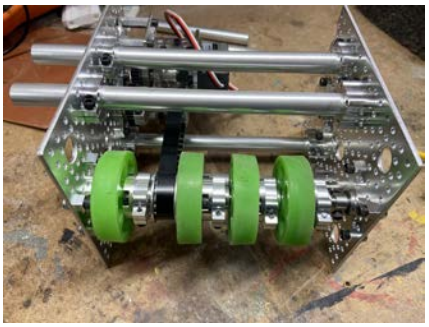
```
@Test
public void happyPath(){
    Drivebase drivebase = new Drivebase(UltimateGoalTestConstants.HARDWARE_MAP);
    drivebase.driveCartesian( xPower: 1,  yPower: 0 , rotationPower: 0,  inverted: false);

    // What does strafing look like?
    Assert.assertEquals(leftFront.getPower(), -leftRear.getPower(),  delta: 0.01);
    Assert.assertEquals(rightFront.getPower(), -rightRear.getPower(),  delta: 0.01);

    // What does turning look like?
    drivebase.driveCartesian( xPower: 0,  yPower: 0 , rotationPower: 1,  inverted: false);
    Assert.assertEquals(leftFront.getPower(), leftRear.getPower(),  delta: 0.01);
    Assert.assertEquals(rightFront.getPower(), rightRear.getPower(),  delta: 0.01);
    Assert.assertTrue( condition: rightFront.getPower() < 0);
    Assert.assertTrue( condition: leftFront.getPower() > 0);

    // What does driving forward look like?
    drivebase.driveCartesian( xPower: 0,  yPower: 1 , rotationPower: 0,  inverted: false);
    Assert.assertEquals(leftFront.getPower(), leftRear.getPower(),  delta: 0.01);
    Assert.assertEquals(rightFront.getPower(), rightRear.getPower(),  delta: 0.01);
    Assert.assertTrue( condition: leftFront.getPower() > 0);
}
```

This also allows us to make code that is much safer by the time it even reaches the robot - for example, this season we had the test shown above that makes sure our mecanum math makes the robot move correctly before our drivebase had even been built.

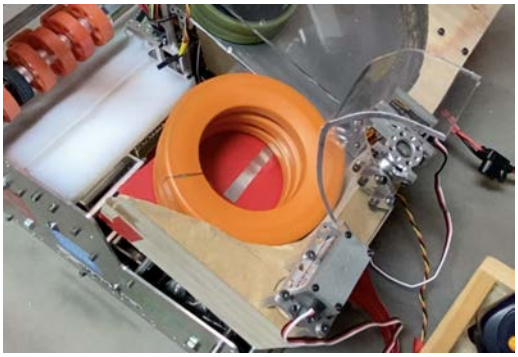


Changes Made for the State Tournament

We replaced the “sweeper” for a fourth stage on the intake. Using a fourth stage eliminates the need for operator input to push the rings into the hopper. It also prevents the rings from getting flipped out of the intake. We run the fourth stage with a continuous rotation servo into a gear train to get the correct speed and driven by a belt to leave room for the rings to clear the mechanism. The software team was able to synchronize the fourth stage to the rest of the intake to make it even easier for the drive team to intake rings.

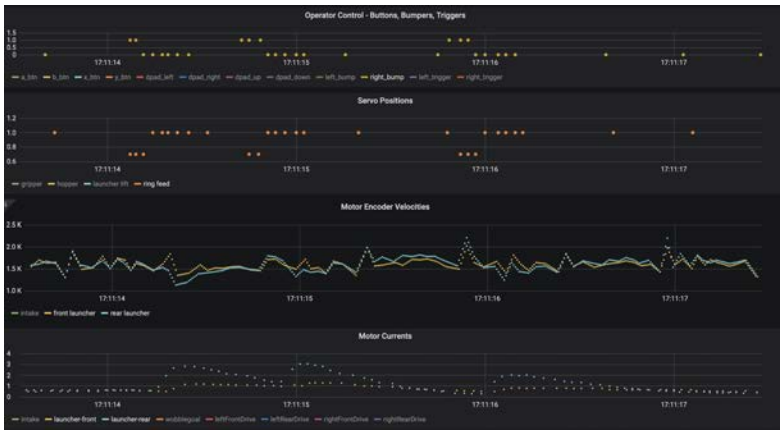


Software changes for State include adding the code for the new ring hold down mechanism, automating the wobble goal arm, and adding launch buttons to the driver gamepad.



The ring hold down arm was added to the launcher mechanism to better line up the rings to the launcher wheels. This was to improve consistency of the launcher between rings. We used real-time performance metrics to get the best delay time between rings. We also used a state machine to change which position the arm of this mechanism was in. It goes between up (zero rings), three rings, two rings, one ring, and back to up in that order automatically. This means that it cannot go to the next number before being in the state of the number before.

We also automated the extension of the wobble goal arm during tele-op. To do this we extended portions of our wobble goal code from auto into an open/closed loop state machine for tele-op that is tied to a button on the gamepad. We also added exits back into the open loop part of the wobble state machine when the operator moves the sticks.



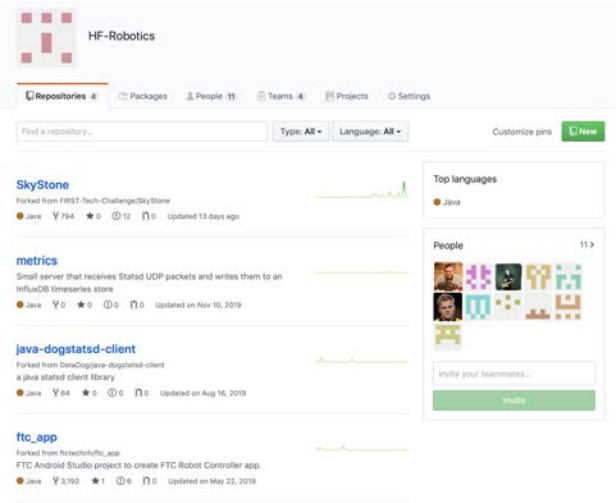
In addition we added the capability for the driver to launch rings alongside the operator.

This was with the intent to cut down on the in mach delay between lining up the robot up with the goal and the actual launching of the rings to the goal. We did this using the AnyButton class from our library of code, which allows us to tie the same functionality to multiple buttons with ease. In the end, this change was only 8 lines of code and required no changes to the state machine or other code related to running the launcher but has made a big difference in our ability to score rings quickly.

Outreach for 2020 - 2021

Our workshop is hosted by the Homewood Science Center. Fun fact - the Science Center used to be a funeral home, and our workshop used to be where the hearses were parked! From March 2020-September 2020, team members helped fill STEM learning kits for the science center. During that time, Homewood Science Center passed out 11,506 PopUp SCIENCE @Home hands-on learning kits to kids in our community to help keep science learning alive during the pandemic.

Young learners excited to explore PopUp SCIENCE @Home STEM activity kits.



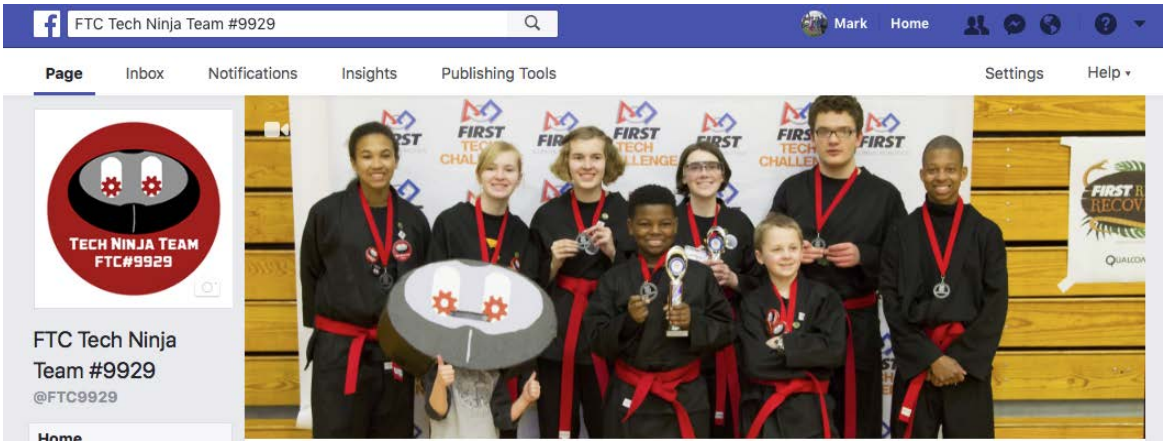
GitHub – We share all our robot code with the world as we write it on GitHub

It is also on GitHub where we publish TntFtcCore, which is a publicly available library of code available for anyone who wants to use it. It was made as a legacy project by the senior programmers. It consists of software produced over the multiple years the senior programmers have participated in FTC. This software is designed to make advanced tasks more reliable and easier to program. It already has 100+ downloads per month

Website

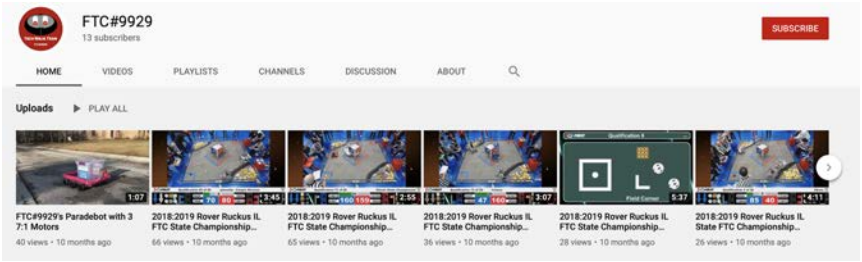
The team has its own independent website. It features pictures of events and meets, information about both the FTC FIRST program and our team, some blog entries of things we have learned, and even past seasons' Engineering Notebooks to serve as examples to new teams. We use the website to reach out to potential sponsors and the community.

Facebook – We have our own Facebook Page, used more to reach local audiences (upcoming events, news). We have 150+ followers that are primarily within the state of Illinois or are family and friends of the team members.



Twitter – We are @FTC9929 on Twitter. Here is where we tend to keep up with other FTC teams, sharing our successes (and experiments that don't quite work out). We have over 450 followers, and we enjoy seeing how teams around the world are having fun with robots and STEM.

YouTube – We have many videos including footage of previous matches and tech tips such as "Friends Don't Let Friends Use KEP Nuts"



Sustainability

This season we were fortunate to once again be awarded a grant from Schneider Electric.

We have a lot of costs that Schneider helped cover:

- Power Tools ($400)
- Robot parts and spares ($1400)
- Competition Registration ($275)
- Sanitizing supplies!

The Homewood Science Center has been invaluable to us for many seasons. They provide us with space to work and have introduced us to our two sponsors. We give back by volunteering at their events, which in turn helps us with STEM outreach. The Homewood Science Center allowed us to maintain access to our shop throughout the pandemic. Being able to use the shop has been great because it allowed us to have access to our tools and have enough space for social distancing.

The Tech Ninja Team tends to recruit new members through our sister FLL teams. Gregory is a new team member this year - and he has joined the software sub-team. Gregory has participated for many years in FLL.

We also have an “Intern” role, where potential team members who do not yet meet the age requirements for FTC, but have interest in what we’re doing, or plan on joining next season and want to get a jumpstart can participate.

Having the intern role also helps new members gain the same skills that old members have learned over multiple seasons as well as become acclimated to the team environment as a whole before they compete.

We have one intern this year, Leith, who has been working with the software team to learn more about how we write code and work together.