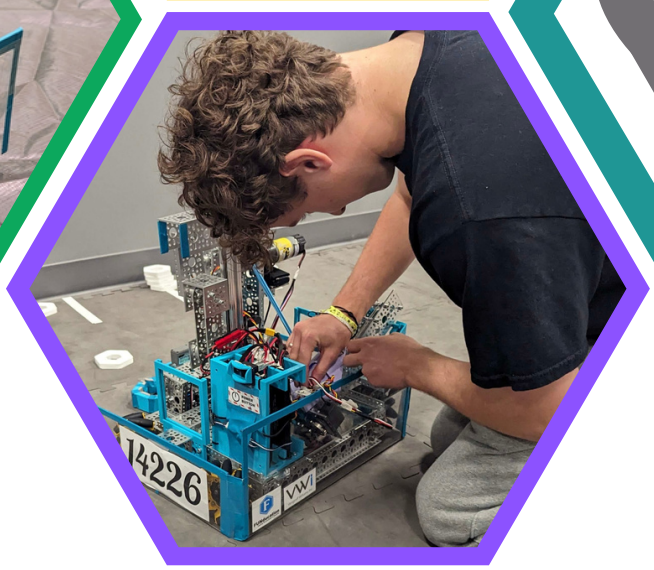
14226

Team Ironic

Engineering Portfolio







FIRST TECH CHALLENGE

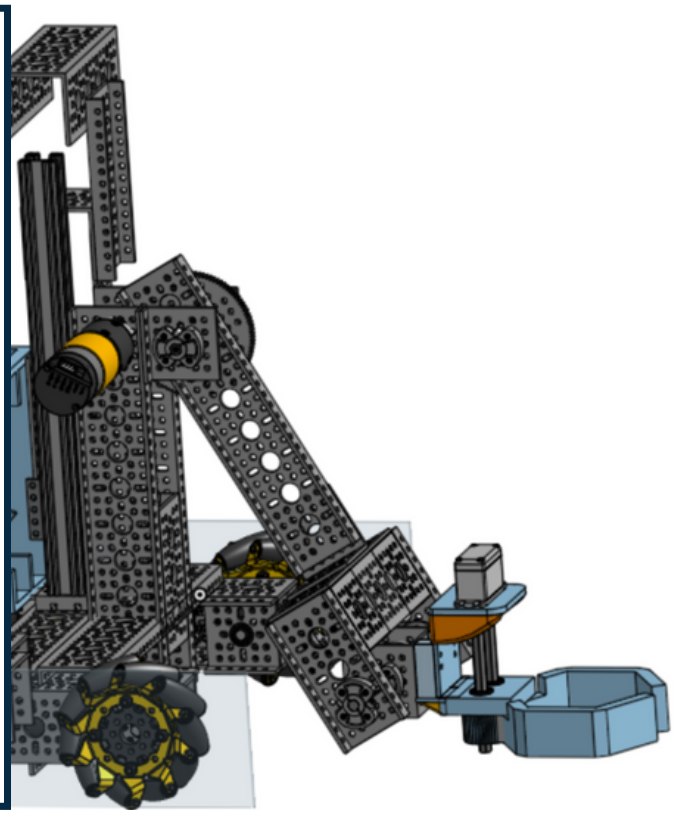
CENTERSTAGE

PRESENTED BY RTX

IRONIC ROBOTICS

Table Of Contents

Introduction	1
Team	2
Team Plan	3-4
Outreach	5-6
Design	7-10
Strategy	11
Programing	12-15



Who We Are

Ironic is a competitive robotics team from Manatee School for the Arts located in Palmetto, Florida. This team allows students 7-12th grade to learn through hands-on STEAM based programs as they work together to build a robot to compete in FIRST Tech Challenge throughout the school year. We have a total of 3 adult instructors and 7 hard-working team members who make whatever they imagine come to life.

Team Ironic

Team Members



Koby
Co-Captain
"Kendal"
Engineering / CAD



Reana
"eh?"
Engineering / CAD



Mary
"Better twin"
Coding/ Engineering



Jayden
Co-Captain
"K.I.S.S"
Programing / Driving



Ethan
"Ouch"
Engineering / Driving



Amberlynn
"Stitch"
Coding/ Engineering



Connor
"Zoro"
Engineering / Driving

Coaches



Mr. Foster
Coding Coach
5th season



Ms. McCollough
Engineering Coach
2nd season



Ciel
Alumni Mentor
4 seasons on team
1st season as mentor



Mr. Rote
Engineering Coach
4th season

Team Plan

Sponsorship

Our Current Team Sponsors:
Vengroff Williams: A leading business-solution provider, Vengroff Williams Inc. has been supporting us since the 2021-2022 season. VWi is a leading business-solution provider relied upon by more than 2,000 global clients to transform their businesses into leaner, more dynamic, agile and efficient operations. We thank Vengroff Williams for aiding in this season (and past seasons) of FTC.
FUNducation: FUNducation puts the fun in education by empowering communities and engaging kids in Science, Technology, Engineering, Arts, and Mathematics. With an end goal of launching, developing, and sustaining S.T.E.A.M within communities, FUNducation inspires local students. We thank FUNducation for getting us off the ground and continuing to help team Ironic.

Sustainability

Students learn through hands-on teaching while working with mentors and student team managers. The student team captains typically take the lead in teaching and instructing the newer members on the ropes of their respective sub-teams (i.e. engineering, coding, CAD, and driving). If any additional help or input is needed, our mentors are always glad to answer any questions the students need.

To keep our team going, we have a few ways to renew interest in robotics. Primarily, we will begin to let interested students join our off-season game, starting in February. During the off-season, all previous members will be exploring new interests, and team captains will teach members about their skills. We host a robotics club, where we invite middle and high schoolers to join and see the inner workings of a team. Another way we maintain interest in our team is by mentoring our sister school, HOLA Elementary's Lego League team that will begin next season. These students will work with basics in coding and constructing. We are hopeful that as those students graduate they will go into our middle school program.

Future Plans

Team Ironic's success within this season has been acknowledged by our school with a dedicated robotics space. We hope to use this area wisely in next year's season. This space will help optimize our time, as it will aid in our team's goal of bettering our organization and time management.

Besides plans on using our new space wisely, the team also has aims in upcoming fundraising. Each student has been given the opportunity to find five future donors. Whether it be a sponsor from a local business, or a contribution from friends and families, this search for more donors and sponsors helps us network within our community. Making connections is a huge part in gaining a better future for our team, and we are eager to get out into the community and gain from this opportunity.

Team Plan

Finances

Income	Amount
Sponsors	
Vengroff Williams	$1,000
FUNducation	$295
Misc.	
Students	$349.00
Carry Over from Previous Year	$3.28
Income Total	$1,648.28

Expenses	Amount
Registration Fees	
Season Registration & Control Kit	$602.18
League Fees	$100
Website Fees	
Domain Registration	$12
Web Hosting	$6
Parts	
Parts for Lift/Hang System	$183.60
High Torque Servo	$28.99
Servo Savers	$22.97
General Parts/Tools	$171.42
Misc.	
Full Game Set	$491.61
Expenses Total	$1,618.77

All team members are responsible for ensuring there are sufficient funds before ordering parts as well as documenting purchase orders. They also keep track of what they order and how much is spent, which is then audited by Team Ironic's coaches.

Contacting sponsors as well as reaching out to local businesses and persons in the community for sponsorship also is a team responsibility during the season.

Finishing last season with low finances, we are fortunate to have returning sponsors who aid in funding our robot, our team, and the necessary components needed to compete.

Schedule

1,985 WORK HOURS REACHED

Pre-season, the team primarily met on Tuesdays and Wednesdays, with the addition of Thursdays shortly after game launch. With the team coming in as needed on additional days to continue their progress.

The week before any league meet the team consistently met every day to prep for the competition ahead. We estimate that Team Ironic has spent in the realm of 1,985 work hours on the current seasons bot to include time spent drafting, programing, and 3D printing that took place outside of planned practice time.

Outreach

Outreach Events

First Lego League



Team Ironic uses its outreach opportunities in order to bring stem to younger audiences.

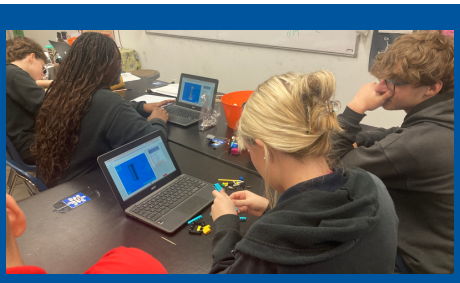
One such event was the FIRST Lego League where Team Ironic brought their bot to one competition to allow the Lego League teams to drive the bot as well as see the scale of robotics they could compete in if they continued with FIRST programs.

Through the consistency in our programing presets many kids were able to drive the bot and have a great time interacting with the team. When we were working with the kids at the First Lego League competition, we were able to discuss a lot about our bot and what we had to accomplish in the game. This included picking up pixels and placing them on the backdrop, hanging on scaffolding, and launching an airplane.

Many curious parents and children came to us asking questions about our robot, as well as to ask us if they could try driving our bot. Kids and their parents were learning about CAD, our bot, and FIRST Tech Challenge. This was a very successful outreach event, and Team Ironic was glad to be a part of it.



Legos In The Classroom



Team Ironic believes that everyone should have the chance to experience STEAM. That's why we love teaching others inside school about FIRST, so that they can learn about STEAM while at school. This allows gets the word out about our team to our school so others can join the our club, and possibly the team.

In the picture above, we let fellow students help build parts for the First Lego League. Even though they do not have hands on the bot itself, it puts a message out into the community that the First Lego League is open to everyone.

Outreach

Future Outreach Opportunities

As for future Outreach plans we are currently have plans in the making with multiple branches of the Manatee County Libraries, as well as with our sister school HOLA Elementary.

Rocky Bluff and Lakewood Ranch Library have STEAM events. One of these events is STEAM Saturday. For our outreach at these locations, team members would assist in teaching CAD software, coding, and other technology based labs to children and teens.

We have been in contact with our sister school, HOLA Elementary, about having a day where we are able to come and show the students our robot, introduce them to coding through block coding, and have engaging activities with the overall goal to introduce them to robotics and STEAM.

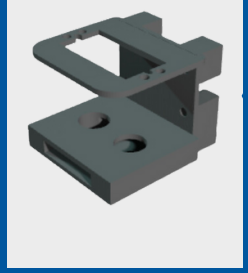
Outreach opportunities like these events allow us to contribute to our community as well as gain further insight into our own designs through onsite stress testing and debugging.

Many of the modifications we made to our designs on both the engineering and coding sides came from issues we ran into at outreach events. Many more of our team members were introduced to Team Ironic for the first time at outreach events just like these.

Design (C.A.D.)

CAD or Computer Aided Design has helped our team a lot this year. CAD is a software used for modeling 3D parts and designs. These designs act like a 3D blueprint that is then followed or printed. The use of CAD software like Onshape has allowed our team to take a step forward on all designs with the ability to suit our parts exactly to our needs.

One example of this is our gear box, which holds the claw and allows us to align to the floor for accurate pixel pickup. Through various iterations we were able to improve tensile and compressive strength along with minimizing the weight suspended on our lift arm and wrist joint.

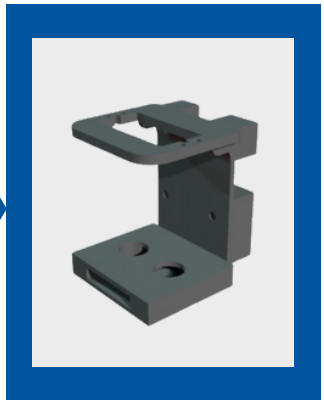


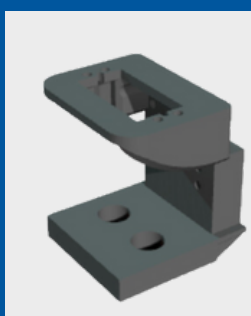
The original design of this gear box was built from goBILDA parts, specifically a servo mount connected to a 2-hole shallow U-channel with a 1-hole U-channel. This design proved too heavy for our wrist servo, which led to the burning out of several servos at meet 1. We had to come up with a solution.

The design to the left is the first CAD model of the gear box that was designed and outfitted to the bot, lowering the weight and further enabling further dexterity to the claw.

Design 1 proved to be a great basis but needed it improvements in terms of sizing and structure. The axles in use with the first CAD design caused strain between the servo resulting in breaks. To combat this, originally extra bearings where used as spacers.

This new design fixed these structural issues by lengthening the piece to allow for the needed space.





The final iteration was mostly designed with stress in mind as the last iterations proved to increase stability yet had issues with breaking if hit from certain angles.

To fix this, larger bridged supports were added to prevent the top mount breaking. In addition, we changed the back quad mount to a 2-mount in order to not press the top 2 pegs against our wrist mechanism.

Design - Lift

Our lift is made of 3 primary systems, the hook, the carriage, and the lead screw itself. They all move using a 1620-RPM planetary yellow jacket motor. Geared at 3:1 in favor of torque since it is a higher RPM motor.



Hook

Basic composite of goBILDA parts, taking advantage of a 4-hole U-channel to stay balanced when hanging as opposed to needing to center the entire mechanism on the bot.

Carriage

Made of 4 wheels on a goBILDA pattern plate to keep the carriage and the attached hook centered on the go-rail.

Lead Screw

A simple lead screw with a brass fitting inside of a go-rail, attached to a hyper hub with a 96 tooth gear attached.

Our lift mechanism, again proved how CAD can change our workflow of a team.

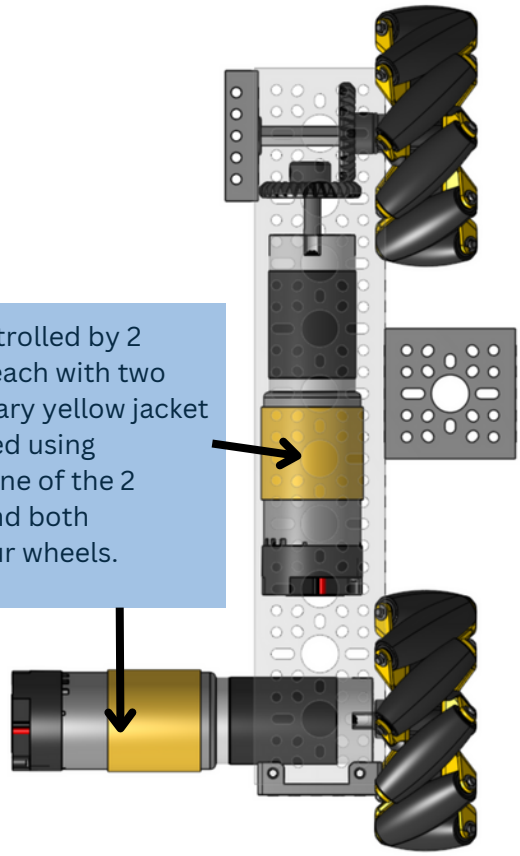
This design was put together in Onshape, an online CAD workshop using the GoBilda Library. Once built in Onshape the parts were able to be directly ordered using the catalog of parts used.

The build process of the physical parts was far faster than anything else we have built at this point due to the CAD design.

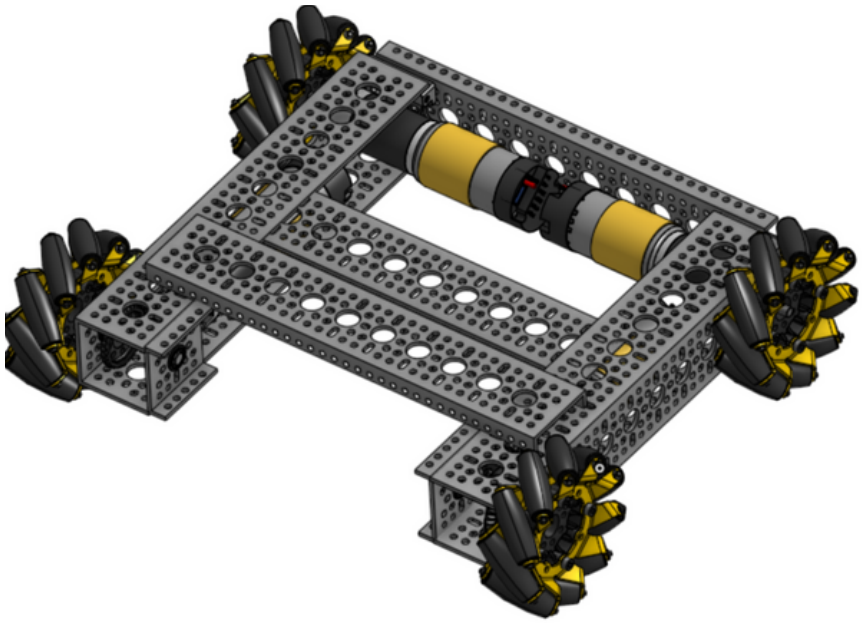
The CAD design could be taken apart and laid out in step by step instructions, like LEGOs. This enabled one team member to design the build while a completely different team member built it in their absence.

Design - Chassis

The chassis is the main body of our bot. We designed it to be a square so that it could fit in the boundaries while also giving us the maximum space. When we were placing our control hub and our expansion hub on the bot we were able to fit them in a spot without over-crowding. Even though everything fits, the main problem is the cables getting knocked loose or getting damaged. We have attempted to fix this with the use of foam tape around sharp corners, as well as making cables more secure. Originally the control hub and the expansion hub were on two separate layers of plexiglass however we had to scrap that idea when we realized the hanging mechanism and the airplane launcher wouldn't fit, so we printed a shelving unit to make them more compact.



The robot is controlled by 2 mirrored sides, each with two 320-RPM planetary yellow jacket motors controlled using encoders, with one of the 2 bevels geared and both connecting to our wheels.

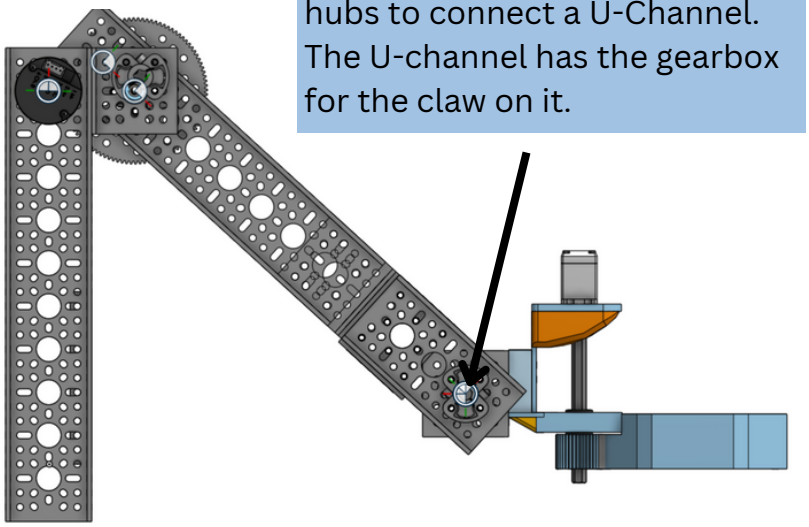


The full bot measures at 441.96mm wide, 365.76mm long, and 450.85mm tall. These measurements do not allow us to pass under the field elements with out use of the gate, but the foot print of our chassis leaves room for all of our needed elements for this years game.

Design - Arm

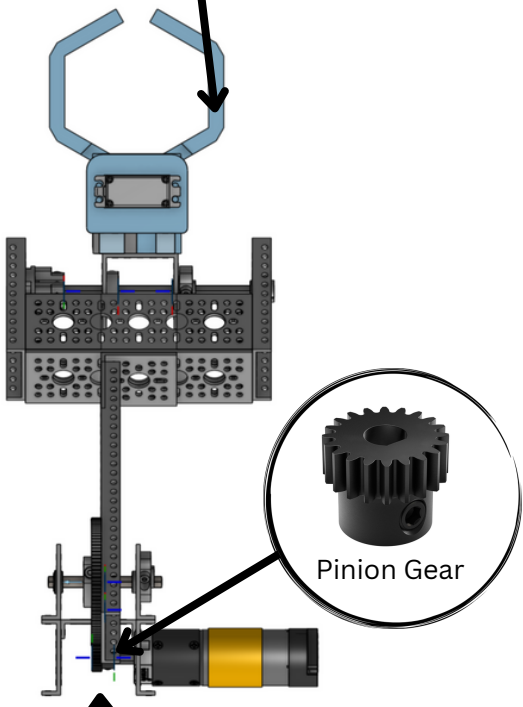
The Wrist

The wrist has a servo that is used to turn the axle which has hubs to connect a U-Channel. The U-channel has the gearbox for the claw on it.



The Claw

The claw is attached to the wrist by two axles going through two holes with bearings (with one axle attached to the servo). Both claw parts have a gearing mechanism to open and close the claw with only one servo attached. The claw can pick up two pixels at once.



In the beginning of the season we had the arm bend at two points using a chain system. We found that the chain idea was not a long term solution due to issues like the chain slipping and the arm extension being limited.

We then started to shift our attention towards a design with one point of rotation, which you see in the model above. Due to this change we were able to scrap the troublesome chain for a normal motor and gear shaft to lift the arm. This design has been what the team has been using since.

The design of the wrist has been minimally changed with a primary focus on weight. This was done through switching out axle locks and experimentation with different servo mounts.

The Elbow

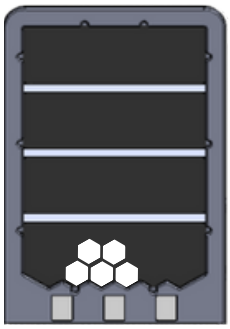
The elbow has a yellow jacket motor which has a pinion gear on its axle. The pinion gear turns a larger gear that allows the arm to move up and down.

Strategy - Driver Controlled

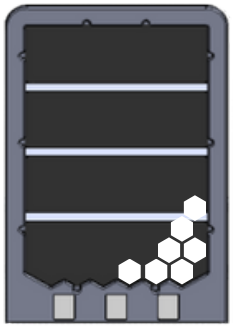
Meet #1 / The Start

At Meet #1 the drive team's strategy was to grab 2 white pixels at the same time and just try to place as many on the board as possible, then in the last few seconds of endgame we would park on either side of the board as we had not yet completed designing our hanging system or plane launcher. One other note is that due to our bot's size we had to move back and forth through the gate using the bot's arm to lift it rather than through scaffolding (this strategy does not change throughout Meets 1-3 however we do improve its speed).

We were able to follow this plan somewhat consistently during our matches, though still faced technical failures at this meet. Ending this meet we felt practice on precision and speed was needed.



Example of projected plan of Meet 1.



Example of projected plan of Meet 2.

Meet #2 / The Hang

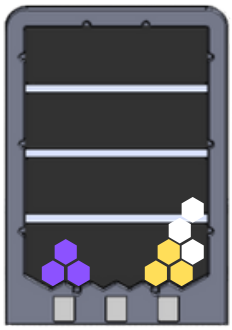
With the addition of our new hanging system and additional drivers practice, we could now consistently grab 2 pixels and place them simultaneously on the board with more precision, a shift in strategy which opened up more opportunities for points.

We would grab 2 pixels at a time, placing all the pixels to one side of the board, aiming to get them above the first scoring line for the points bonus. Once in end-game we would aim to score 1-2 more times on the board and with ~10 seconds left we would attempt to hang. We did notice that when attempting the hang it would sometimes fail due to the gears used becoming misaligned. However, we later fix that issue.

Meet #3 / The Plane and Mosaic

At Meet #3 the drive team had almost perfected our placement method for pixels so rather than going for quantity we opted to try to create 2 mosaics utilizing pixels scored in autonomous, then we would attempt to stack pixels to get over the score lines by using mosaic nearest to the edge of the board for height.

For our endgame portion, tweaks were made to the hanging system which improved consistently and speed. This, combined with our drone launcher making its debut, increased our projected endgame points from 20 points somewhat consistently to 20 - 50 points consistently, with our only issue in end game being occasional miss fires of our plane due to a small flaw in the endurance of our plane launcher. Finally, we started to swap out our batteries in between each match to maintain consistency.



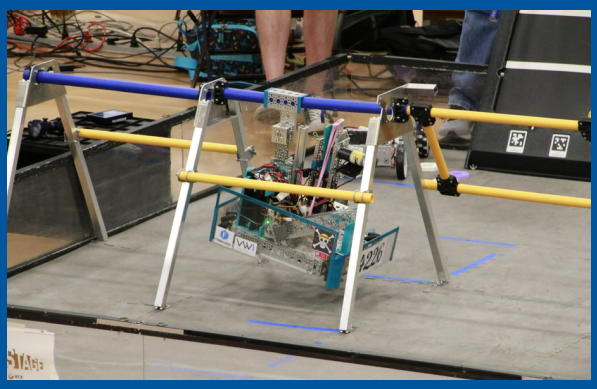
Example of projected plan of Meet 3.

Programming - Learning Java

In this season of FTC, our team took on some new members, a few of which were interested in the programming aspect of robotics. These new members were each first taught the basics of Java (the language we use for our bot). At first, our newer programmers were a bit confused by the code. However, after some time spent learning they were ready to work on programming specific functions for the bot this season.

The first function programmed by the newer programmers was the hanger subsystem and it's TeleOp function. This code went through two models. For the first of these models, we looked at programing DC motors. Specifically programming for our lead screw mechanism to move up and down. This part of the programming process was pretty simple even for the new programmers.





The second stage of this model dealt with the hanging system's code. We removed the flipping mechanism's code and instead opted for a automatic stop for the leadscrew, which perfectly lines up with the truss allowing us to more efficiently execute the suspension.

However the knowledge we gained from that portion of code was later used in the drone launcher's system in order to launch and reset the drone launcher.

After we make a change to the code, we push the changes onto the robot so we can test it! Once we're all done, we push it to GitHub.

```
@Override
public void extend(double speed) {
    if (gallows.getCurrentPosition() >= gallowsMaxHeight)
        {
            gallows.setPower(0);
            fullyExtended = true;
        }
    else {
            gallows.setPower(speed);
        }
}
```

Programming - Autonomous

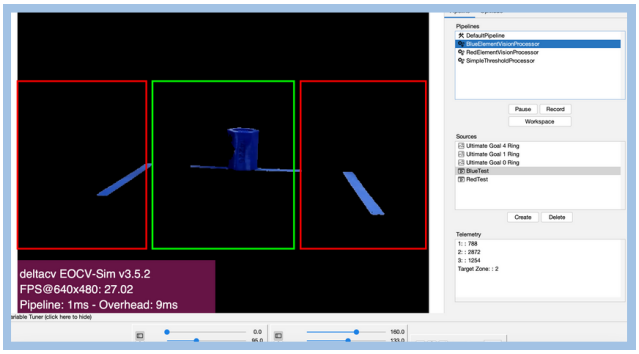
During this season, we've learned a lot about running our robot autonomously. One of the main things we've looked into was RoadRunner. RoadRunner is a library that gives you a bunch of functions to make your autonomous easier to manage.

This system works based off of a coordinate plane; so to easily visualize it, we've also looked into MeepMeep. MeepMeep allows you to see the functions from RoadRunner on a virtual field hosted on your device. This is great because it allows people to work on autonomous paths without having to use the bot.

We also have a system that allows us to park in different locations based off of what our alliance prefers.

```
waitForStart();

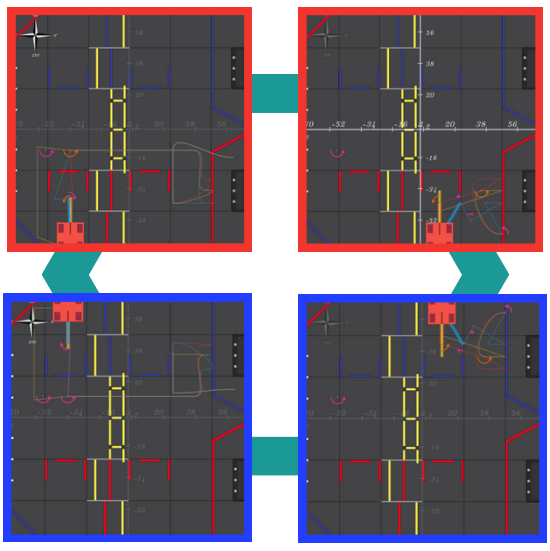
switch (blueElementVisionProcessor.getTargetZone()) {
    case 1:
        drive.followTrajectorySequence(blueOffStage1);
        //drive.followTrajectorySequence(waitDebugger);
        drive.followTrajectorySequence(mainPath);
        drive.followTrajectorySequence(place1);
        arm.moveToPosition( height: 0);
        if (parkInCorner == true)
            drive.followTrajectorySequence(parkCorner);
        if (parkInCorner == false)
            drive.followTrajectorySequence(parkMiddle);
        break;
    case 2:
        drive.followTrajectorySequence(blueOffStage2);
        //drive.followTrajectorySequence(waitDebugger);
        drive.followTrajectorySequence(mainPath);
        drive.followTrajectorySequence(place2);
        if (parkInCorner == true)
            drive.followTrajectorySequence(parkCorner);
        if (parkInCorner == false)
            drive.followTrajectorySequence(parkMiddle);
        break;
    case 3:
        drive.followTrajectorySequence(blueOffStage3);
        //drive.followTrajectorySequence(waitDebugger);
        drive.followTrajectorySequence(mainPath);
        drive.followTrajectorySequence(place3);
        arm.moveToPosition( height: 0);
        if (parkInCorner == true)
            drive.followTrajectorySequence(parkCorner);
        if (parkInCorner == false)
            drive.followTrajectorySequence(parkMiddle);
        break;
    default:
        telemetry.log().add("Couldn't detect zone, defaulted to zone 1.");
        drive.followTrajectorySequence(blueOffStage1);
        drive.followTrajectorySequence(mainPath);
        drive.followTrajectorySequence(place1);
        arm.moveToPosition( height: 0);
        if (parkInCorner == true)
            drive.followTrajectorySequence(parkCorner);
        if (parkInCorner == false)
            drive.followTrajectorySequence(parkMiddle);
}
```



This year we used EOCV for our visual detection. We learned that with EOCV you can change the color of video pixels. This is very useful for the detection because we can make the detection based off of how much of one color we see.

These are the paths for the autonomous sequences, diagrammed using MeepMeep. For each autonomous sequence, there are three trajectories.

Red: Zone 1
Yellow: Zone 2
Blue: Zone 3



Programming - Driver Controlled

In this season of FTC our programmers worked towards trying to lessen the manual control of the bot during TeleOp in order to increase the efficiency of the driver's scoring. We did this through the use of preset buttons and commands.

Before these presets were made the driver manually controlled the arm and wrist rotation using only the joysticks which proved to be difficult and slow. Ideally, the presets would automate specific repeated movements made in-game.

We established these known repeated positions through practice as: set pixel stack height, grabbing from ground, and moving the arm behind the bot to go under the gate. Values applying to these positions were pulled from the telemetry data obtained through our OpMode specifically for testing this new system in our code.

These positions were then laid out together to form actions that could then be programmed into our TeleOp mode and assigned to specific button inputs, In some cases, assigning multiple preset positions to a single button which cycles through them.

```
float elbowControl = gamepad2.right_stick_y;
double wristPos = gamepad2.left_stick_y;

if (currentGamepad2.b && !previousGamepad2.b) {
    armState = ArmCalculationTest.ArmState.BACK_POS;
}
if (currentGamepad2.x && !previousGamepad2.x) {
    armState = ArmCalculationTest.ArmState.GROUND_POS;
}

switch (armState) {
    case GROUND_POS:

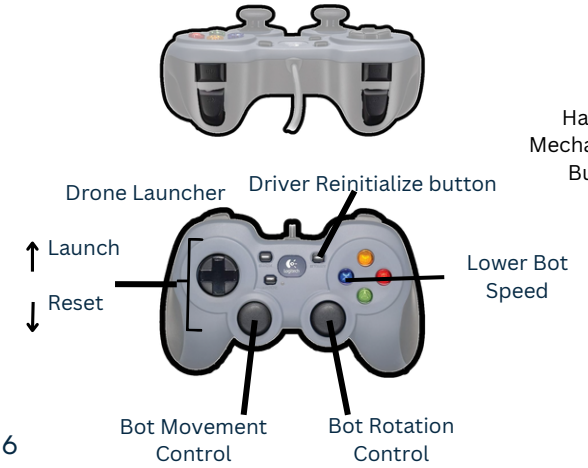
        arm.moveToPosition( height: 0);
        wrist.setWristPos(wristGroundPos);

        if (currentGamepad2.a && !previousGamepad2.a) {
            armState = ArmCalculationTest.ArmState.BACKDROP_1;
        }
        if (currentGamepad2.right_stick_y != 0) {
            armState = ArmCalculationTest.ArmState.MANUAL;
        }
        if (currentGamepad2.y && !previousGamepad2.y) {
            armState = ArmCalculationTest.ArmState.PICKUP_POS_3;
        }
        break;
    case BACKDROP_1:

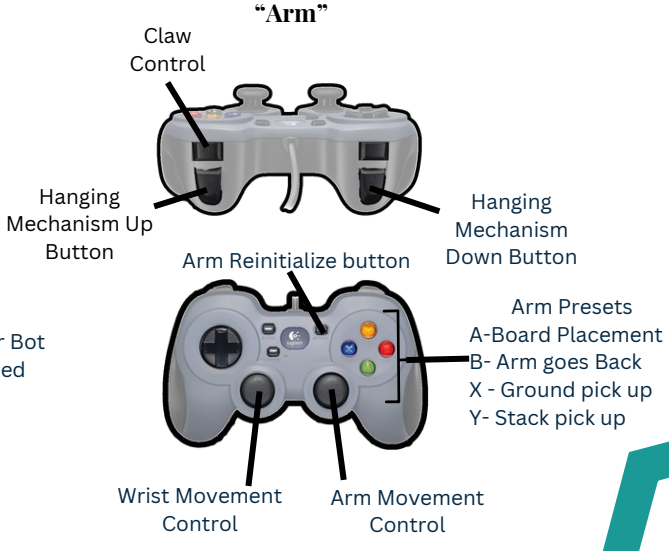
        arm.moveToPosition( height: 6.25);
        wrist.setWristPos(wristStandStill);

        if (currentGamepad2.a && !previousGamepad2.a) {
            armState = ArmCalculationTest.ArmState.BACKDROP_2;
        }
        if (currentGamepad2.y && !previousGamepad2.y) {
            armState = ArmCalculationTest.ArmState.PICKUP_POS_3;
        }
```

Gamepad 1
"Chassis"



Gamepad 2
"Arm"



Programming - Driver Controlled

There are also very important buttons on the controllers, those being the start buttons. The one on the the chassis controller resets the robot **heading**. The **heading** is the direction that the robot is facing. Our robot uses field-centric driving; which is dependent on the heading that we start with, so this button is to help us fix any issues from initialization.
The button on the arm controller is to reset the arm **offset**. The arm position and arm encoder values depend on when the robot is initialized, so the arm will always have an **offset**. This button allows us to fix that issue resetting that offset so that the arm presets work correctly.

```
public void resetHeading() {
    imu.resetYaw();
}
```

Reset in the chassis subsystem.

```
public void resetOffset() {
    offset = 0;
    elbow.setMode(DcMotor.RunMode.STOP_AND_RESET_ENCODER);
    elbow.setMode(DcMotor.RunMode.RUN_USING_ENCODER);
}
```

Reset in the arm subsystem.

```
if (currentGamepad1.options && !previousGamepad1.options)
    chassis.resetHeading();

if (currentGamepad2.options && !previousGamepad2.options)
    arm.resetOffset();
```

The button assignments in TeleOp (options is the start button).

```
6 usages
private SampleMecanumDrive chassis;
15 usages
private ArmSubsystem arm;
4 usages
private IClawSubsystem claw;
11 usages
private IWristSubsystem wrist;
5 usages
private IHangerSubsystem hanger;
3 usages
private IAirplaneLauncherSubsystem launcher;
```

This year, we took a more organized approach to the code of the systems on the robot. Instead of defining them all in one place, we made them all different subsystems. Each subsystem consists of an interface and a class. Methods are made and the subsystem is privately initialized and made part of the hardware map. After that programmers can make that subsystem accessible from whichever class you like!

This approach to code organization also contributes towards team stability, as new programmers who are just joining the team can look at the code easier. This approach will definitely be carried to our other seasons.