



TEAM X #23512

firstteamx@gmail.com

Engineering Notebook

Centerstage 2023-2024

Centerstage

Game Strategy (Our Approach to TeleOp)

Pixels - Main Scoring Element (22 - 58 pts):

We will pick up these hexagonal scoring elements called “Pixels”. These can be placed on a “Backdrop” to score 3 points. We can also move pixels into a taped off zone around the Backdrop, called the “Backstage”. This allows for us to score 1 point.

Suspending - Endgame Scoring Element (20 pts):

During “End Game”, the last 30 seconds of TeleOp, you may suspend your robot. This is basically when the robot hangs from the ground and grabs onto a bar in the truss. We decided to do this because of the large prize of 20 points.

Drone Launching - Endgame Scoring Element (10 - 30 pts):

During End Game, you may also launch a drone (paper airplane) over the field perimeters. There are three taped sections in which the drone must land. You can score 10, 20, or 30 points. The section closest to the field scores 30, and the section closest to the audience scores 10. Ultimately, we chose this due to the possible 30 points.

Objective	Points	Max	Min	Mechanism	Plan	Comments
Placing Pixels on Backdrop and Backstage						
- Pixel in Backstage	1	1	1	Claw	Y	Planning backdrop only
- Pixel in Backdrop	3	27	21	Claw	Y	9 pixels (3.5 cycles)
- Mosaic	10	20		Driver	Y	Planning a few
- Set Line Bonus	10	10		Driver	Y	Planning for 1st row
End Game						
- Suspension	20	20	20	Hook	Y	
Drone Launching						
- Landing Zone 1	30	30		Drone	Y	
- Landing Zone 2	20			Drone	Y	
- Landing Zone 3	10		10	Drone	Y	
Total (120s)		108	52			

Target
52 - 108



Game Strategy (Our Approach to Auto)

Preloaded Pixels

We preload 2 pixels in our XClaw, one yellow and one purple.

Spike Marks (Spike Mark - 20 pts; Corresponding ATag - 20 pts)

Our ML model with 98% efficacy helps us detect the correct spikemark to drop the purple pixel on. Our X-Bot then uses odometry and encoders with PID control to precisely navigate and drop a purple pixel on the spike mark with our custom Team Prop. Robot then navigates towards the backstage using April Tag detection to drop the yellow pixel at the corresponding location.

White Pixel Stacks (Backdrop - 8 pts; Backstage - 4 pts)

X-Bot picks up two pixels from the white pixel stacks, dropping one on the backdrop, and one on the backstage.

Parking (5 pts)

X-Bot finally parks by driving our robot into the backstage.

Auto
Target
33 - 58

Objective	Points	Max	Min	Mechanism	Plan	Comments
Placing Purple Pixel on Spike Mark						
- Team Prop is used	20	20	20	ML Model	Y	Preloaded
Placing Yellow Pixel on Corresponding April Tag						
- Team Prop is used	20	20		Claw & Cam	Y	Preloaded
Placing Pixels on Backdrop and Backstage						
- Pixel on Backdrop	10	10	5	Claw & Cam	Y	Yellow and White Pixel
- 1 Pixel in Backstage	3	3	3	Claw & Cam	Y	White Pixel
Parking Robot at end of Auto						
- Parking in Backstage	5	5	5	Odometry	Y	Easy Points
Bonus points if remaining Pixels in End of TeleOp						
- 1 to 3 points per pixel		7	4	Luck	Y	2 on backdrop + 1 backstage
Total (30 seconds)		58	33			

Total
Target
85 - 166



Game Strategy (Our Approach to TeleOp) - Qualifier 1

Pixels - Main Scoring Element (3 pts):

We will pick up these hexagonal scoring elements called “Pixels”. These can be placed on a “Backdrop”, which is an angled wall, to score 3 points. We can also move pixels into a taped off zone around the Backdrop, called the “Backstage”. This allows for us to score 1 point.

Suspending - Endgame Scoring Element (20 pts):

During “End Game”, the last 30 seconds of TeleOp, you may suspend your robot. This is basically when the robot hangs from the ground and grabs onto a bar in the truss. We decided to do this because of the large prize of 20 points.

Objective	Points	Max	Min	Mechanism	Plan	Comments
Placing Pixels on Backdrop and Backstage						
- Pixel in Backstage	1	1	1	Claw Mechanism	Y	Planning backdrop only
- Pixel in Backdrop	3	30	18	Claw Mechanism	Y	
- Mosaic	10	20	10	Driver Dependent	Y	Planning a few
- Set Bonus (3 max per alliance)	10	10		Driver Dependent	Y	Planning for 1st row
End Game						
- Suspension (per alliance)	20	20	20	Hook	Y	
- Parking in Backstage	5			Driver Dependent	N	We're suspending instead
Drone Launching						
- Landing Zone 1	30			Drone Mechanism	N	Not attempting in 1st Q
- Landing Zone 2	20			Drone Mechanism	N	Not attempting in 1st Q
- Landing Zone 3	10			Drone Mechanism	N	Not attempting
Total (120s)		81	49			

Target
49 - 81

Red = Not Doing



Game Strategy (Our Approach to Auto) - Qualifier 1

Preloaded Pixels

We will preload 2 pixels in our robot, one yellow and one purple.

Spike Marks (Spike Mark - 20 pts; Corresponding ATag - 20 pts)

We will drop a purple pixel on the spike mark with our Team Prop, scoring us 20 points. We will also drop a yellow pixel on the backdrop, aligned with the designated April Tag, scoring another 20 points. For every spike mark, there is one April Tag.

Parking (5 pts)

We will finally park by driving our robot into the backstage, and parking, disabling all movement until TeleOp.

Objective	Points	Max	Min	Mechanism	Plan	Comments
Placing Pixel on Spike Mark						
- White Pixel	10			Claw & Cam	N	Team Prop gives more points
- Team Prop is used	20	20	20	Claw & Cam	Y	Preloaded
Placing Pixel on Corresponding April Tag						
- White Pixel	10			Claw & Cam	N	Team Prop gives more points
- Team Prop is used	25	20		Claw & Cam	Y	Preloaded
Placing Pixels on Backdrop and Backstage						
- Pixel on Backdrop	5			Claw & Cam	N	Not attempting in 1st Q
- Pixel in Backstage	3			Claw & Cam	N	Not attempting in 1st Q
Parking Robot at end of Auto						
- Parking in Backstage	5	5	5	Camera	Y	Easy Points
Bonus points if remaining Pixels in End of TeleOp						
- 1 to 3 points per pixel		3		Luck	Y	
Total (30s)		53	25			

Target
25 - 53

Red = Not Doing



Robot Design Decision

We began with a drivetrain measuring 18in x 18in. However, at its maximum size, we didn't feel comfortable and decided to reduce it to 17in x 17in. After finalizing the drivetrain, our attention shifted towards developing a straightforward and efficient pixel **intake** and **release** mechanism. We ended up with way too many ideas, making it clear that we needed a more disciplined process to pinpoint the ultimate winner.



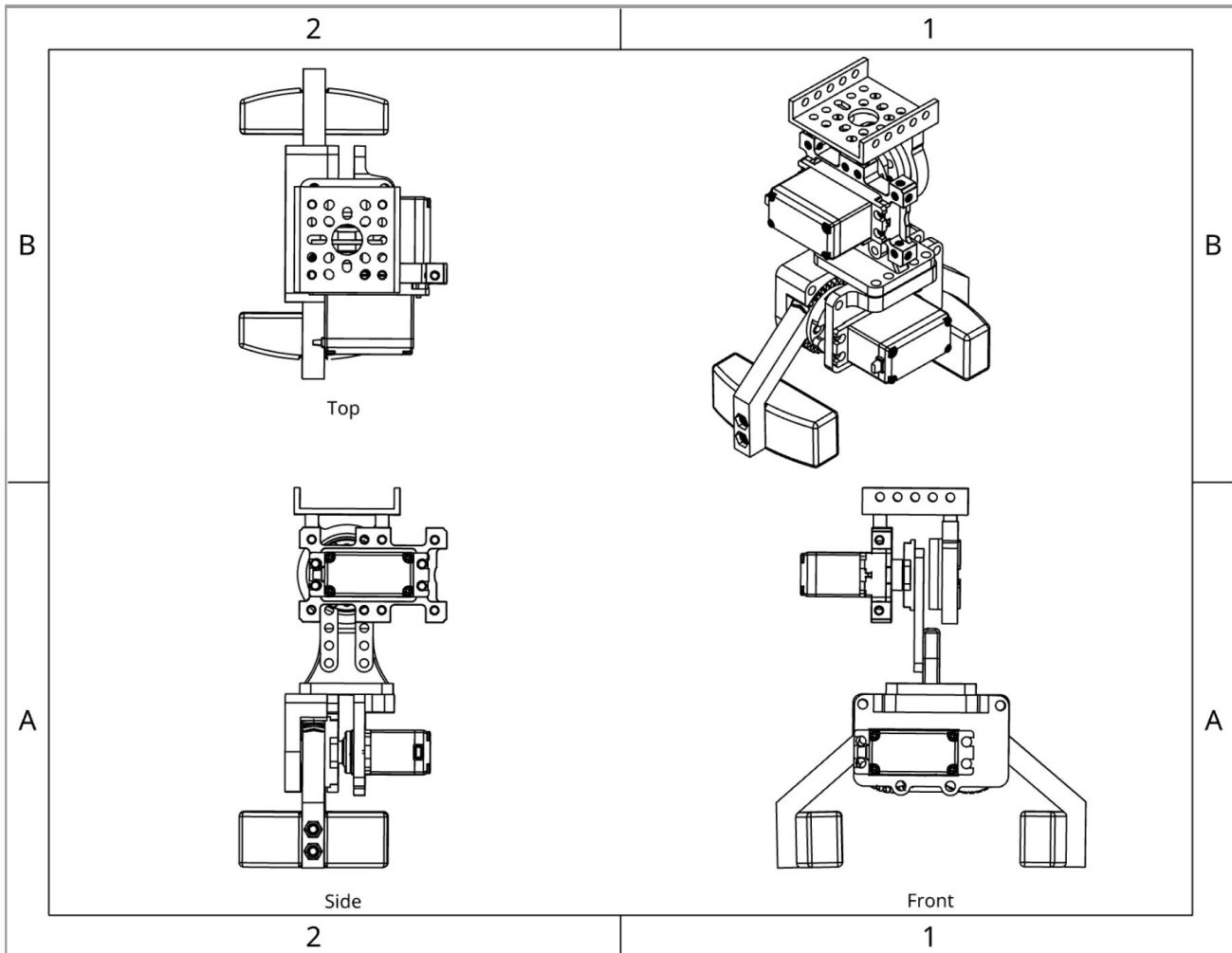
We decided to organize our designs into a chart, breaking down the pros and cons of each idea. This approach helped us eliminate emotional factors from the equation. Quickly, it became evident that "**Two Claws, One Arm**" design was the clear winner. Our choice leaned towards the mousetrap concept, employing distance sensors for pixel retrieval and pixel release.

Idea	Precision	Speed	Output	Intake	Risk	Rating	Rank (#)
<i>Articulated arm with claw</i>	6	6	8	7	5	32	2
<i>Snow Plower</i>	8	9	2	7	2	28	3
<i>Angled slides (Linear Actuator)</i>	6	7	7	2	6	28	3
<i>Spatula</i>	6	8	4	7	3	28	3
<i>Inverse Claw</i>	2	7	6	8	5	28	3
<i>Lead Screw Lift</i>	6	3	5	5	6	25	5
<i>Articulated arm with wheel</i>	4	6	2	5	7	24	6
<i>Two Claws, One Arm</i>	8	7	8	6	5	34	1



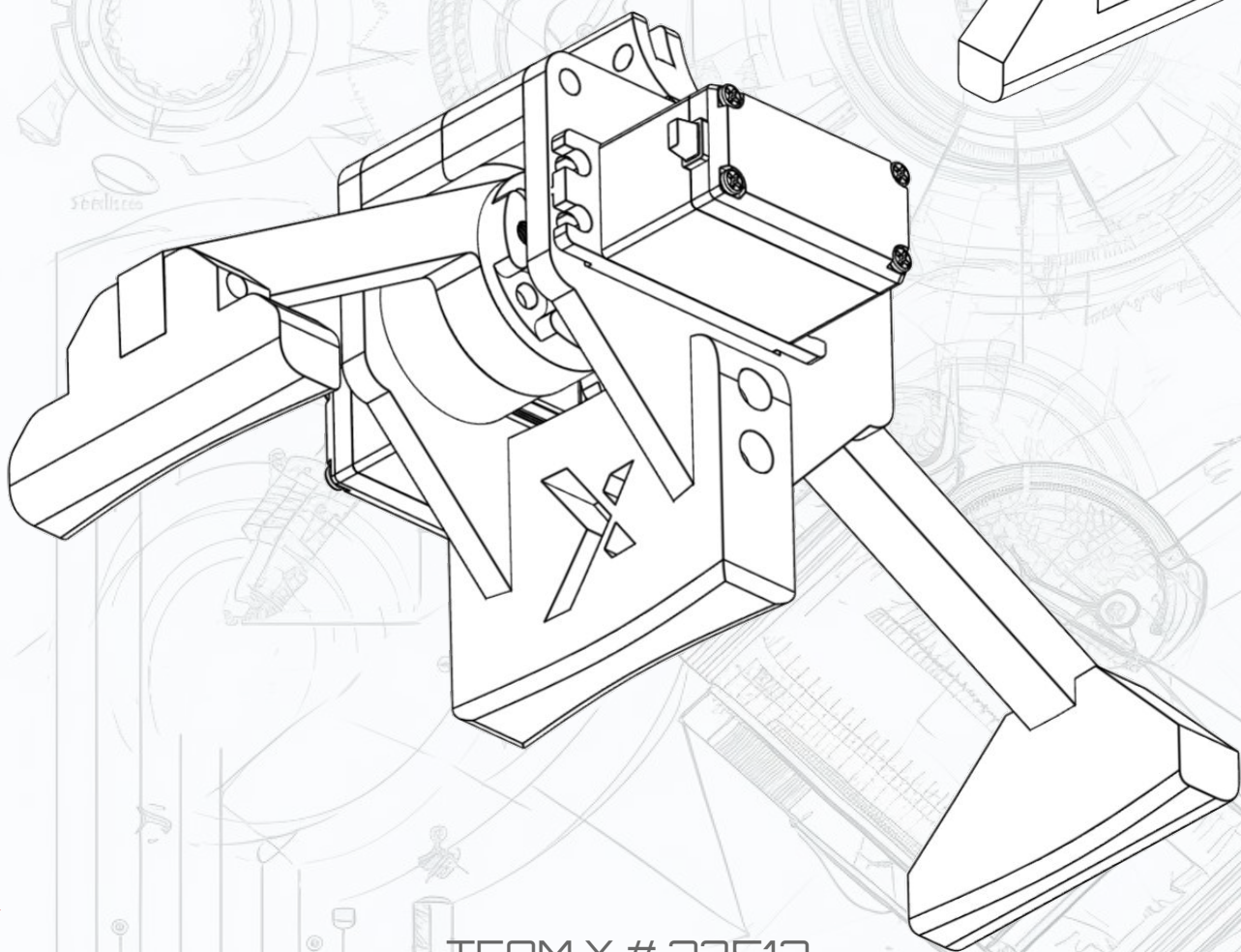
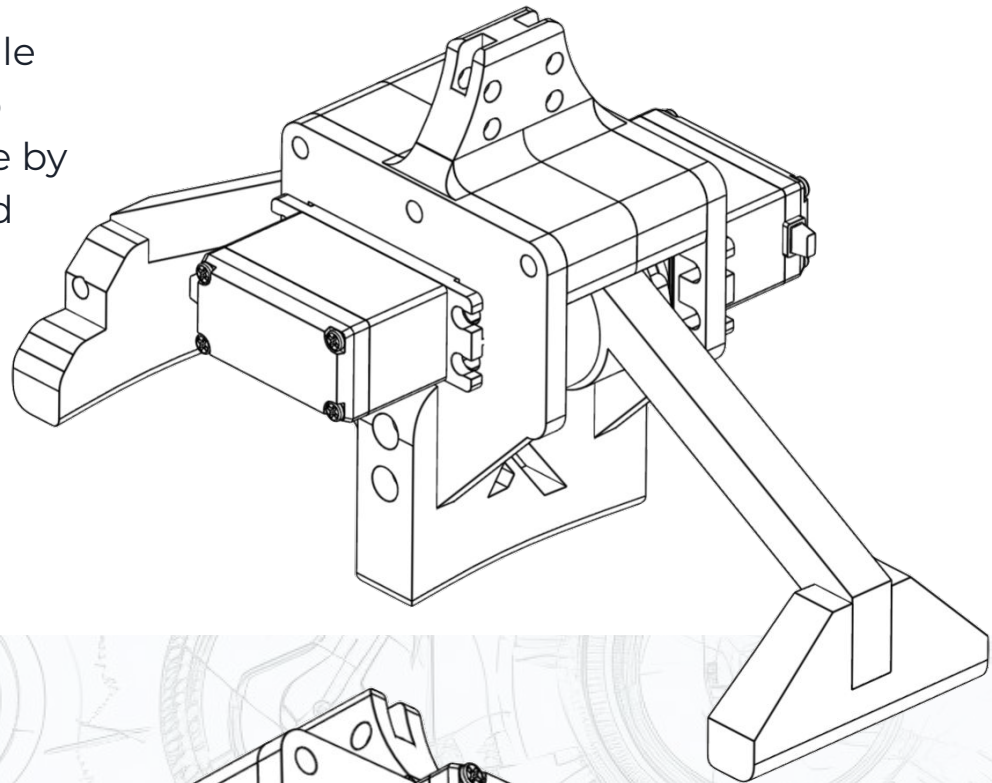
Loney Claw - 1st Iteration

First iteration of our claw design inspired by the looney claw

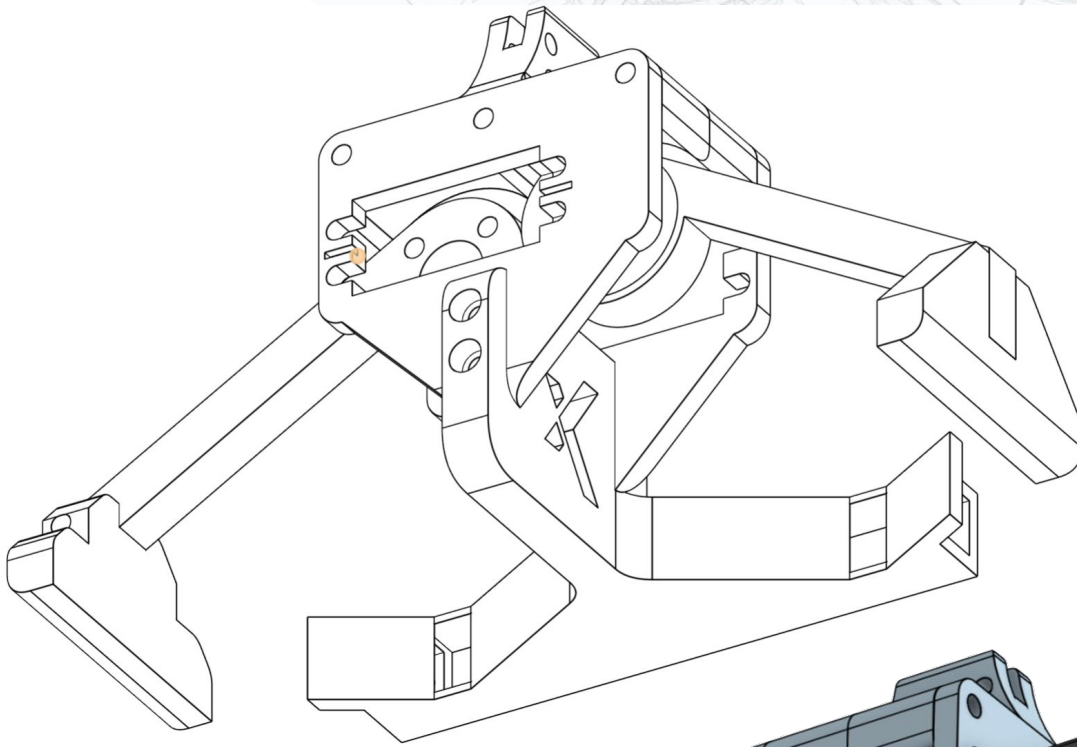


Dual Claw – 2nd Iteration

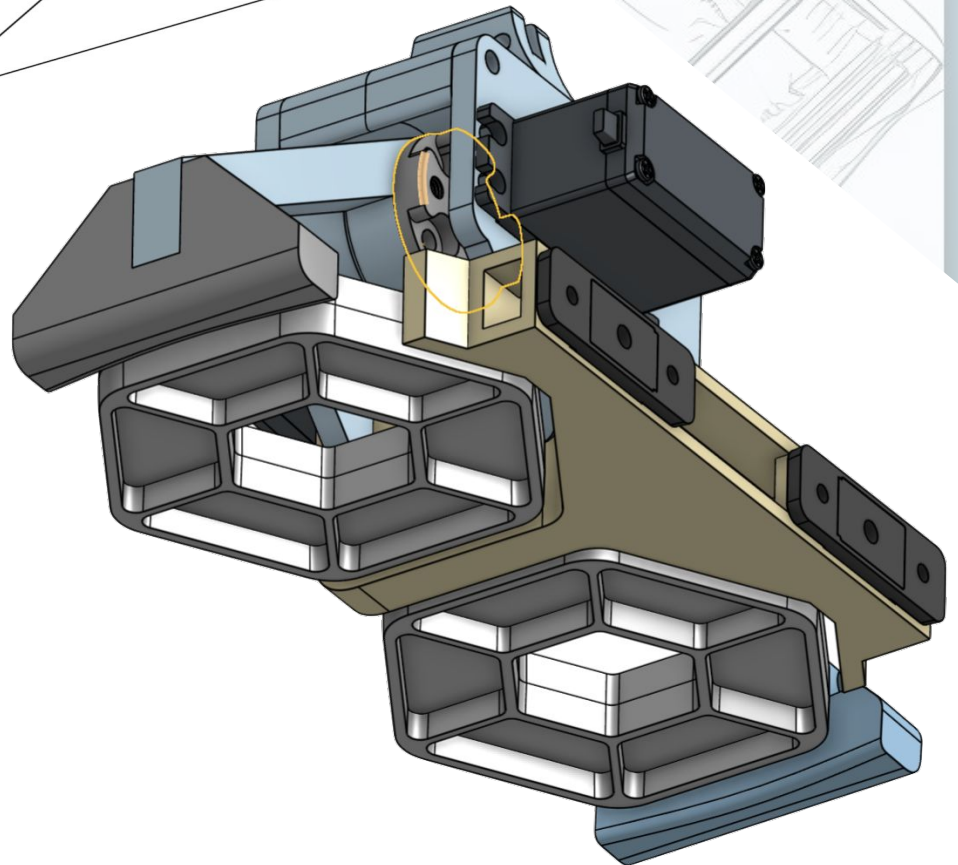
Converting single pixel intake into dual pixel intake by redesigning and adding an extra servo



Mousetrap Claw - 3rd Iteration

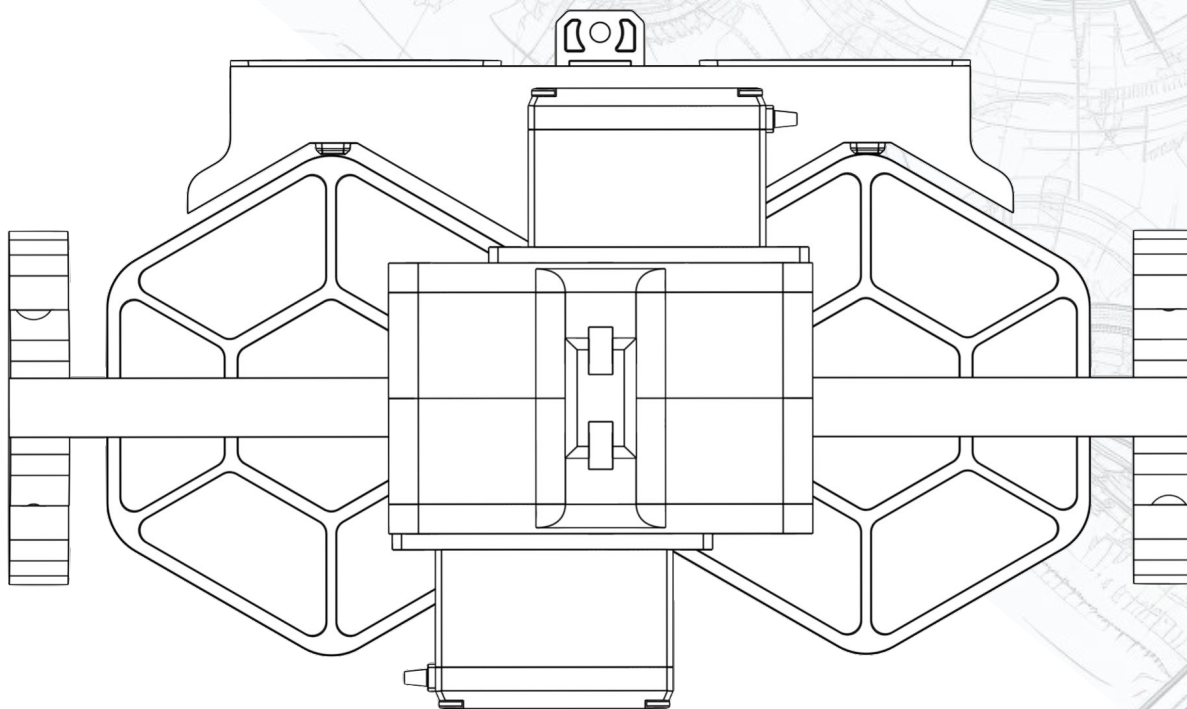


Added two Touch sensors to the Pixel V-guides to mimic the mousetrap function – Automatically closing the claw when a new pixel enters

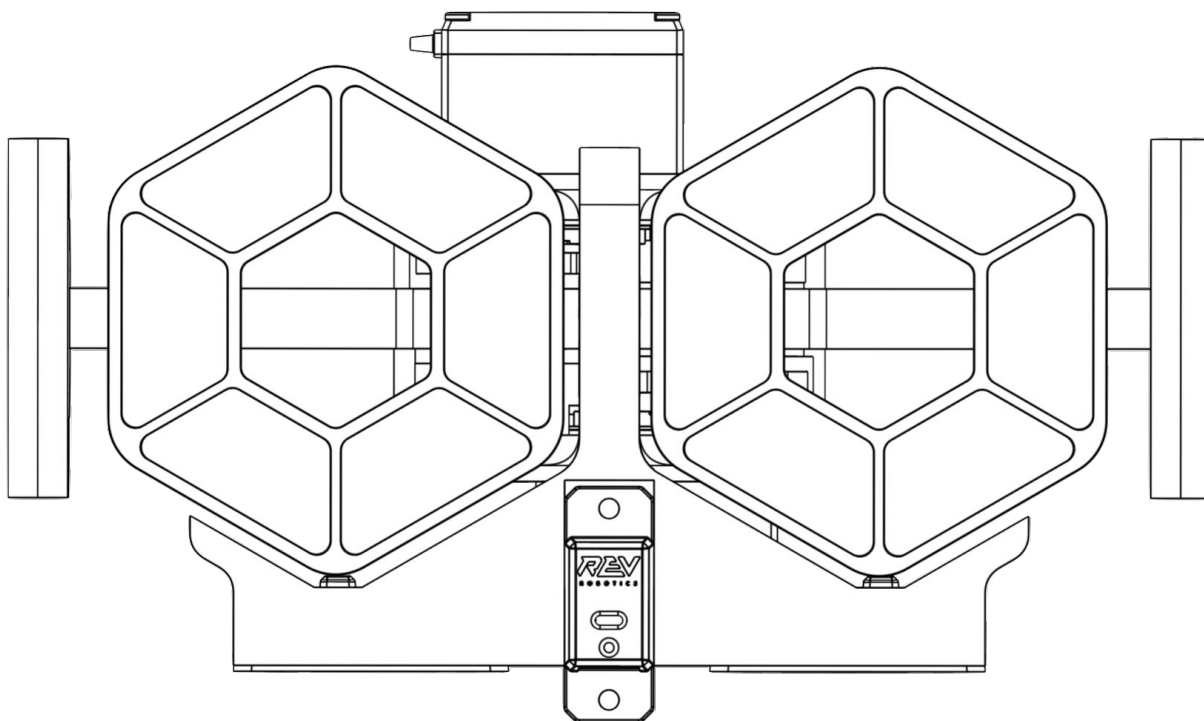


Mousetrap Claw - 3rd Iteration

TOP

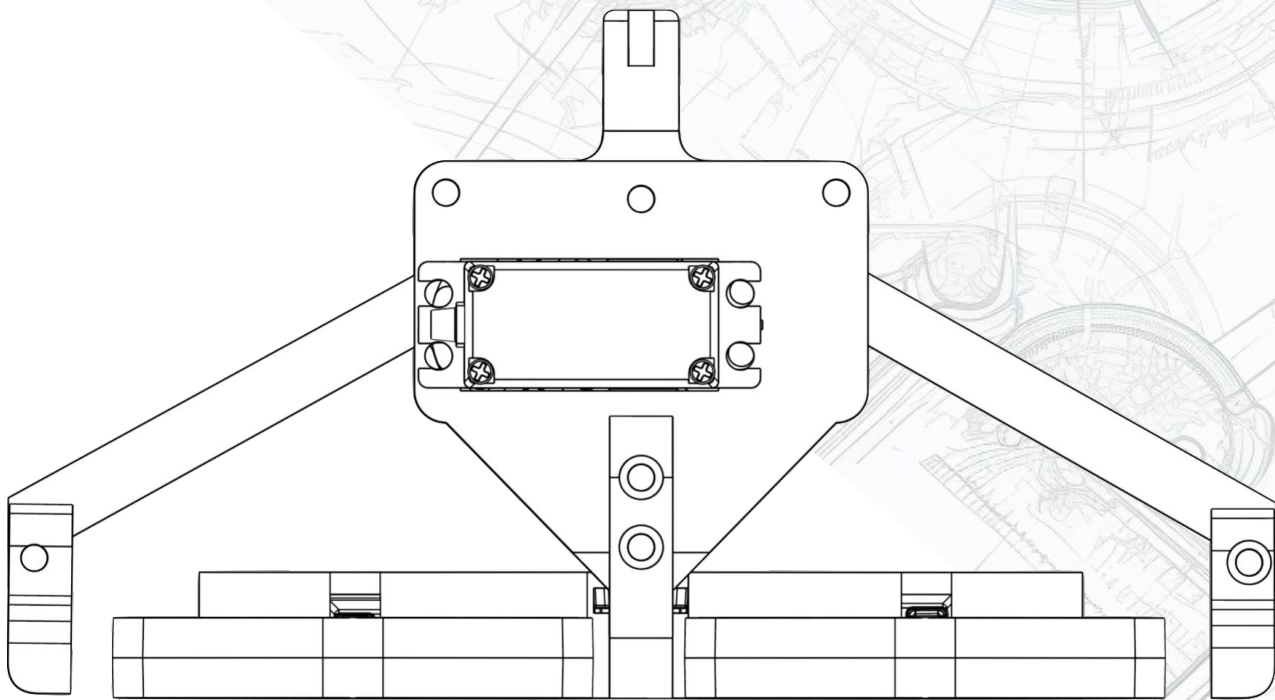


BOTTOM

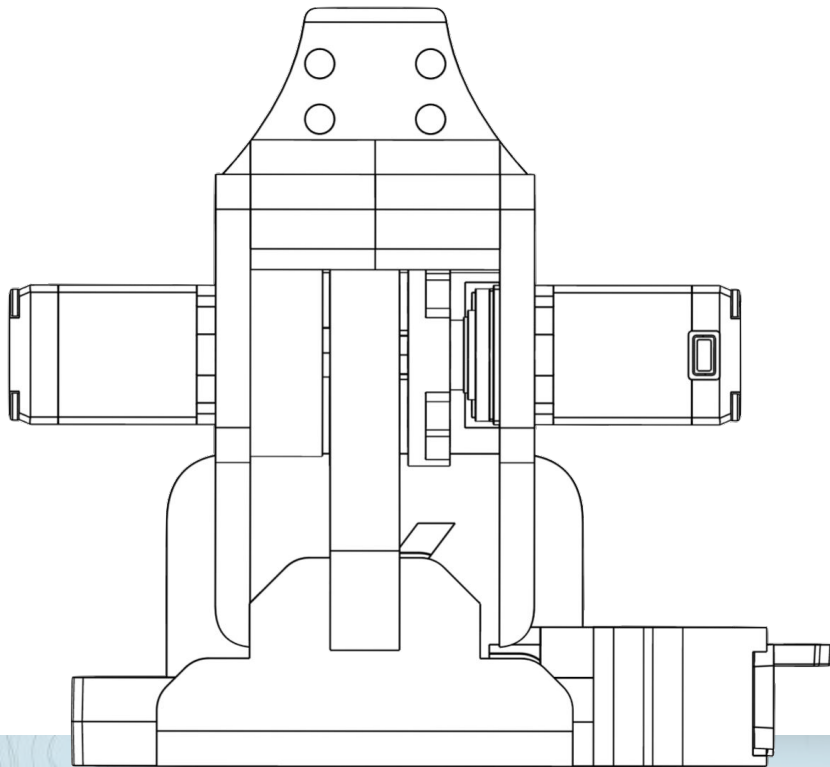


Mousetrap Claw - 3rd Iteration

FRONT

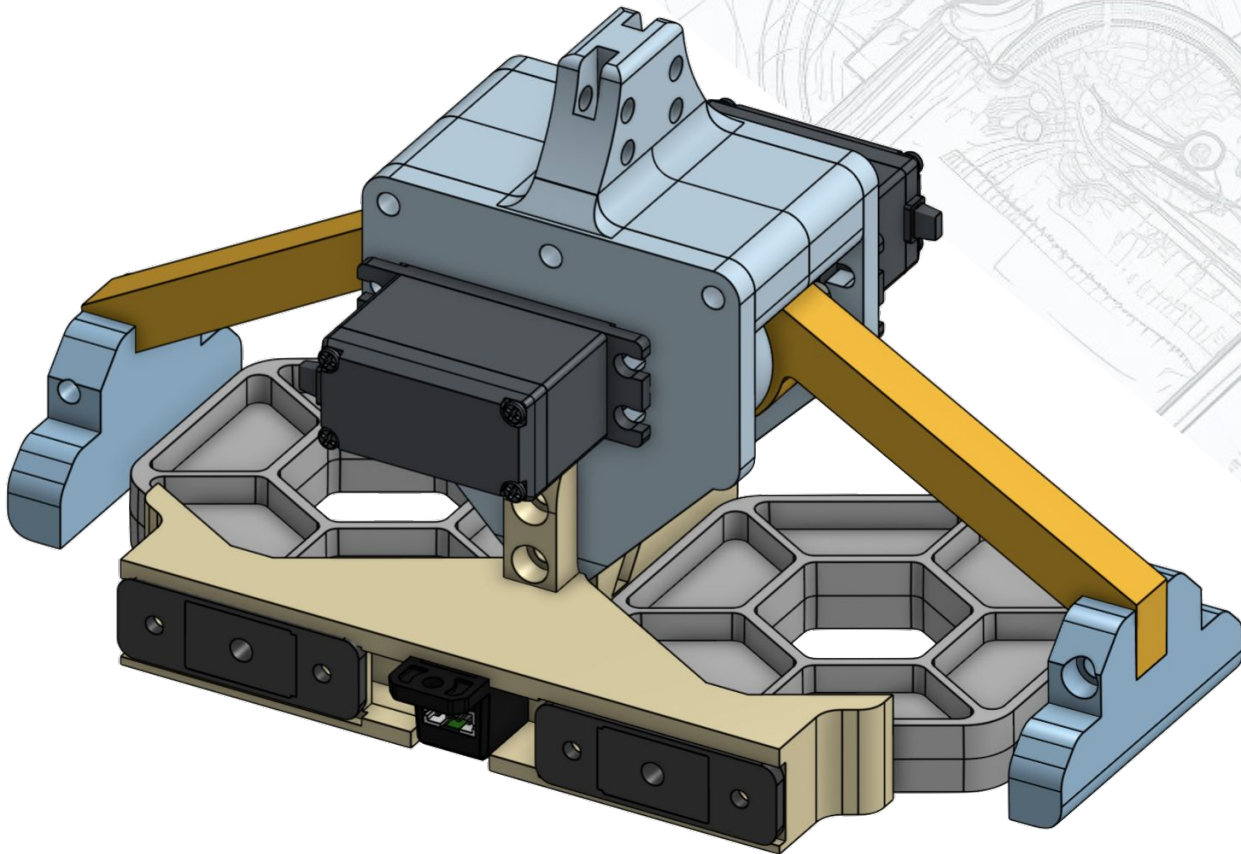


SIDE



Mousetrap Claw - 4th Iteration

Added distance sensor to our mousetrap design to automatically release the pixels on the backboard



Mousetrap Claw - 5th Iteration

After several iterations, our team ended up with this amazing **mousetrap claw**, which is sleek and delivers high performance via it's core design and automation (self-activated)

Pixel alignment`

The custom design automatically rotate pixels into correct orientation

Two claws in one

Design allows the Robot to carry two pixels at a time

Independent control

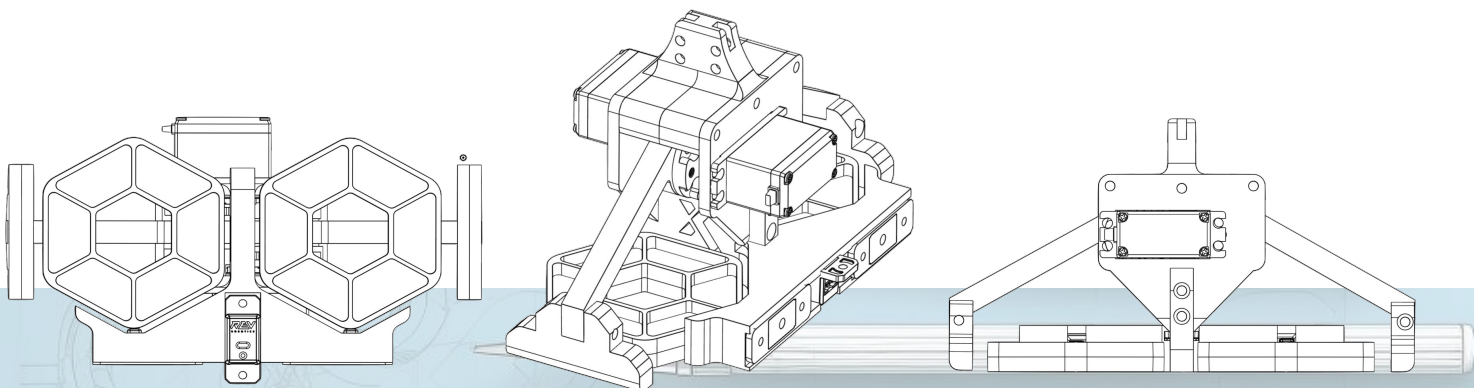
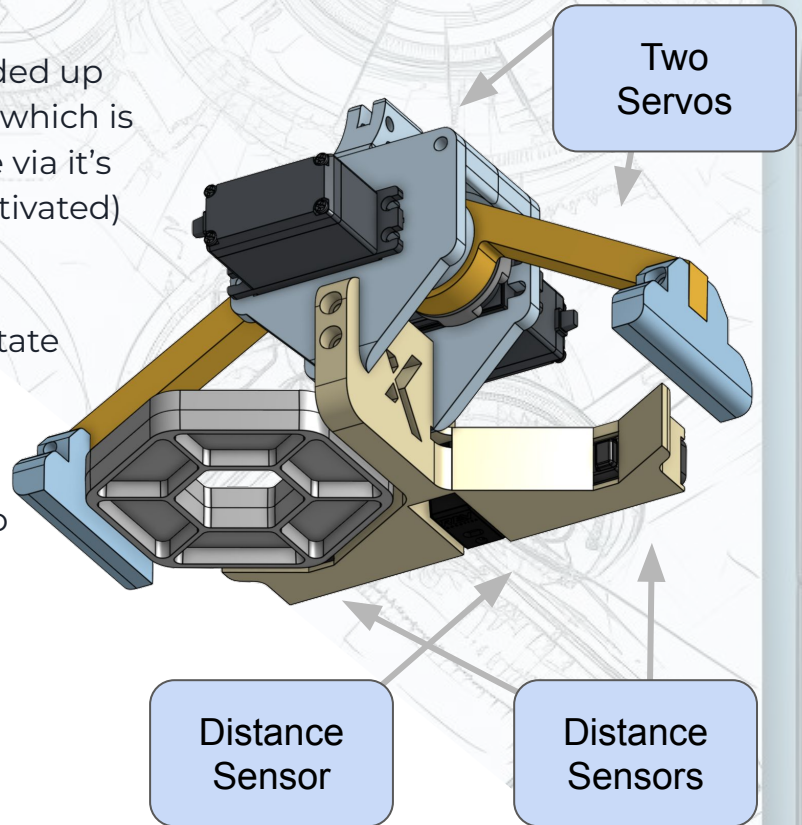
Use of two servo motors to allow for independent control of the claws

The Mousetrap

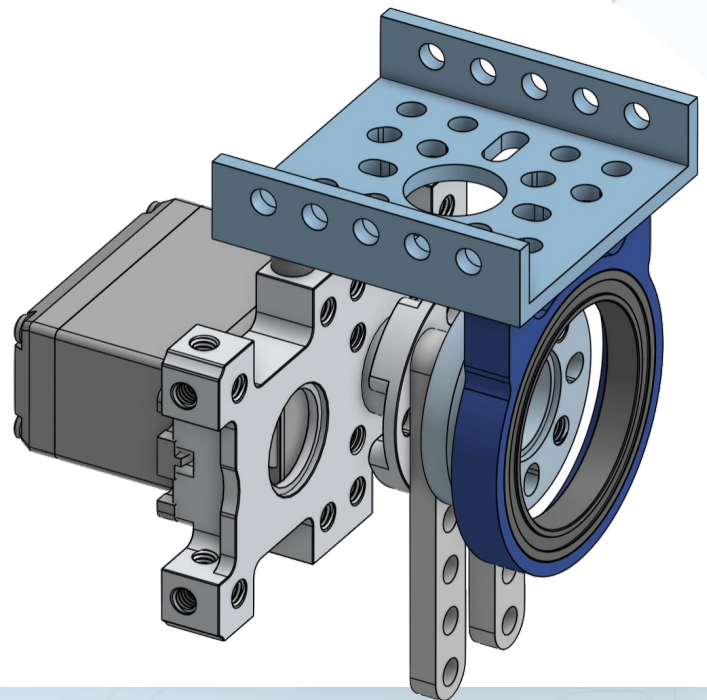
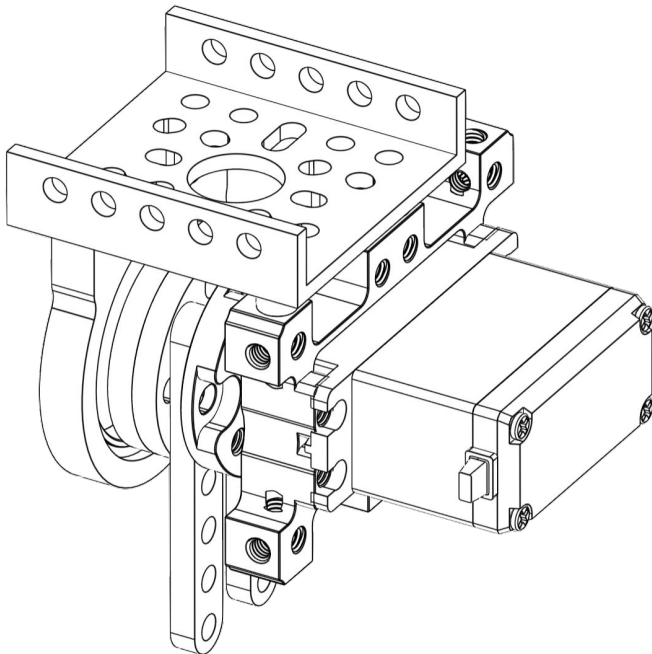
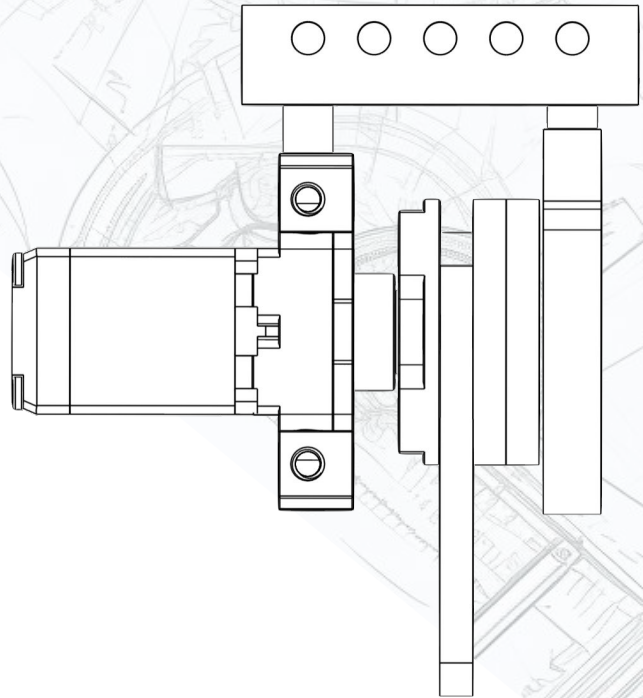
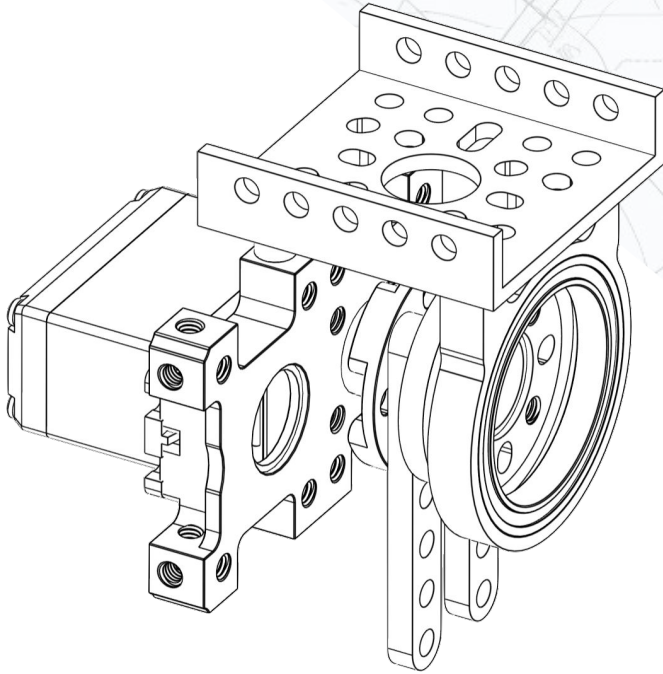
Our desire to reduce human error lead us to design what we call the mousetrap. Pixels are grabbed independently and automatically as they enter the space and trigger the **distance sensors**, eliminating the need for a human operator to take any action

Automated dispensing

The claw uses **distance sensors** to automatically pick pixels and automatically drop the pixels on the back board, enabling high degree of precision.



Wrist Design



X Claw Design - Latest Design

After several iterations, our team ended up with this amazing **X claw**, which is sleek and delivers high performance via it's core design and automation.

Pixel alignment

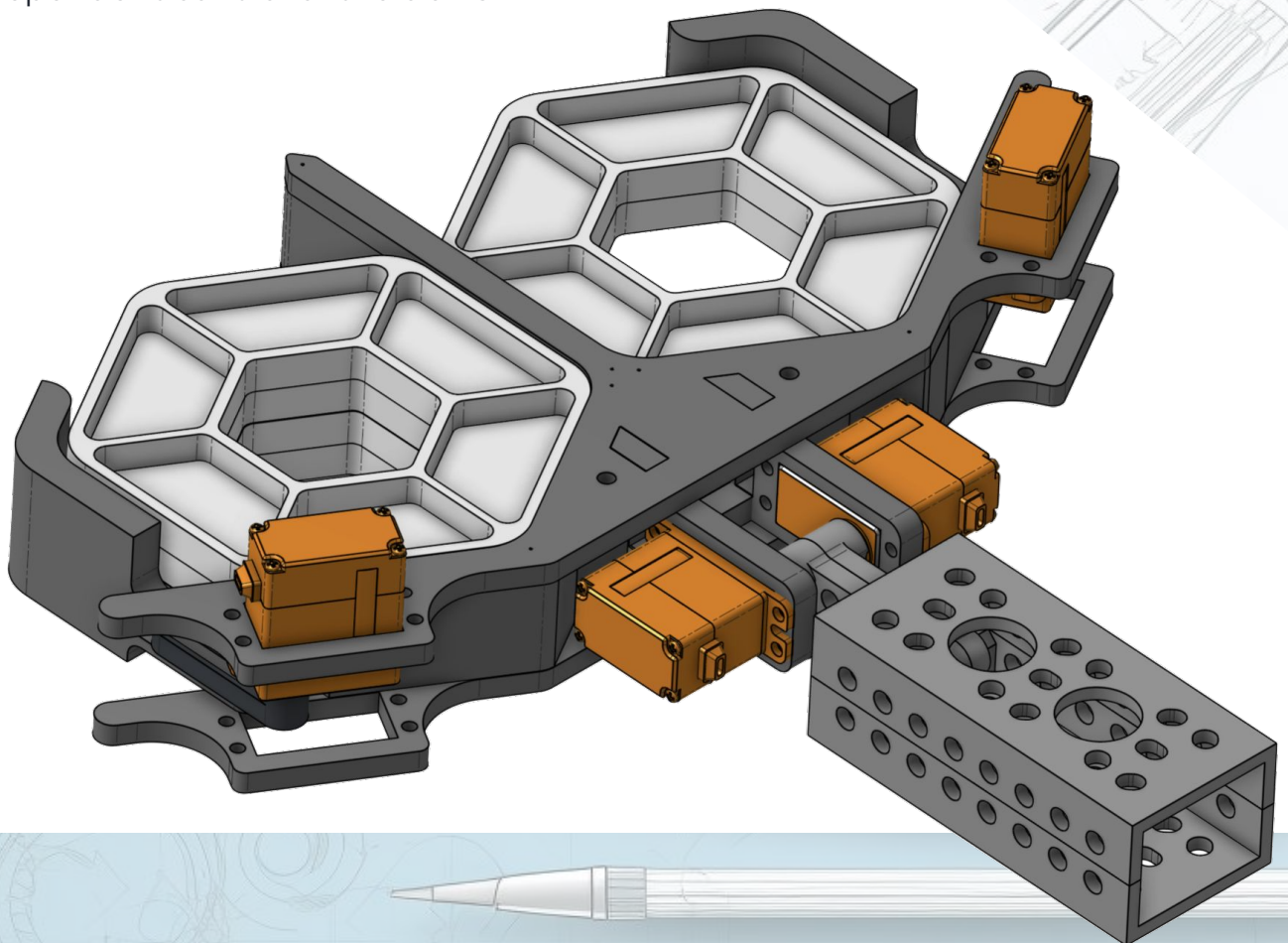
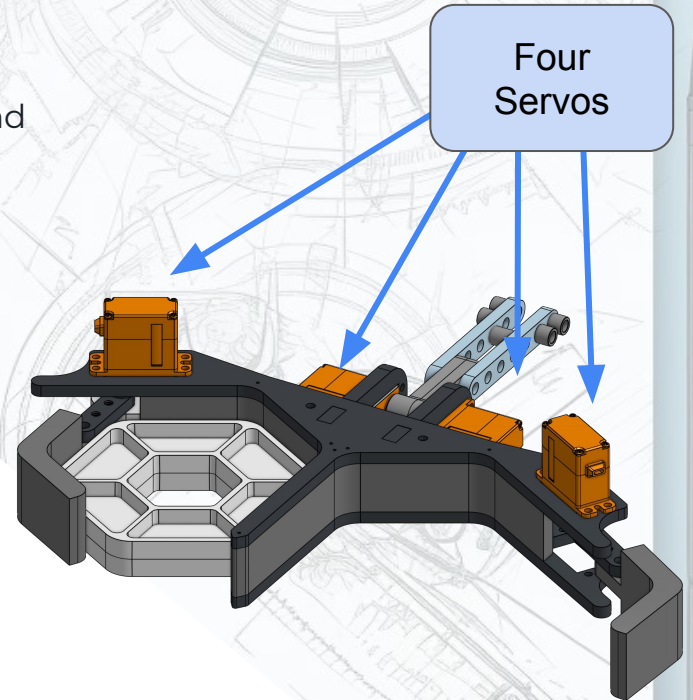
The custom design automatically rotate pixels into correct orientation

Two claws in one

Design allows the Robot to carry two pixels at a time

Independent control

Use of two servo motors to allow for independent control of the claws

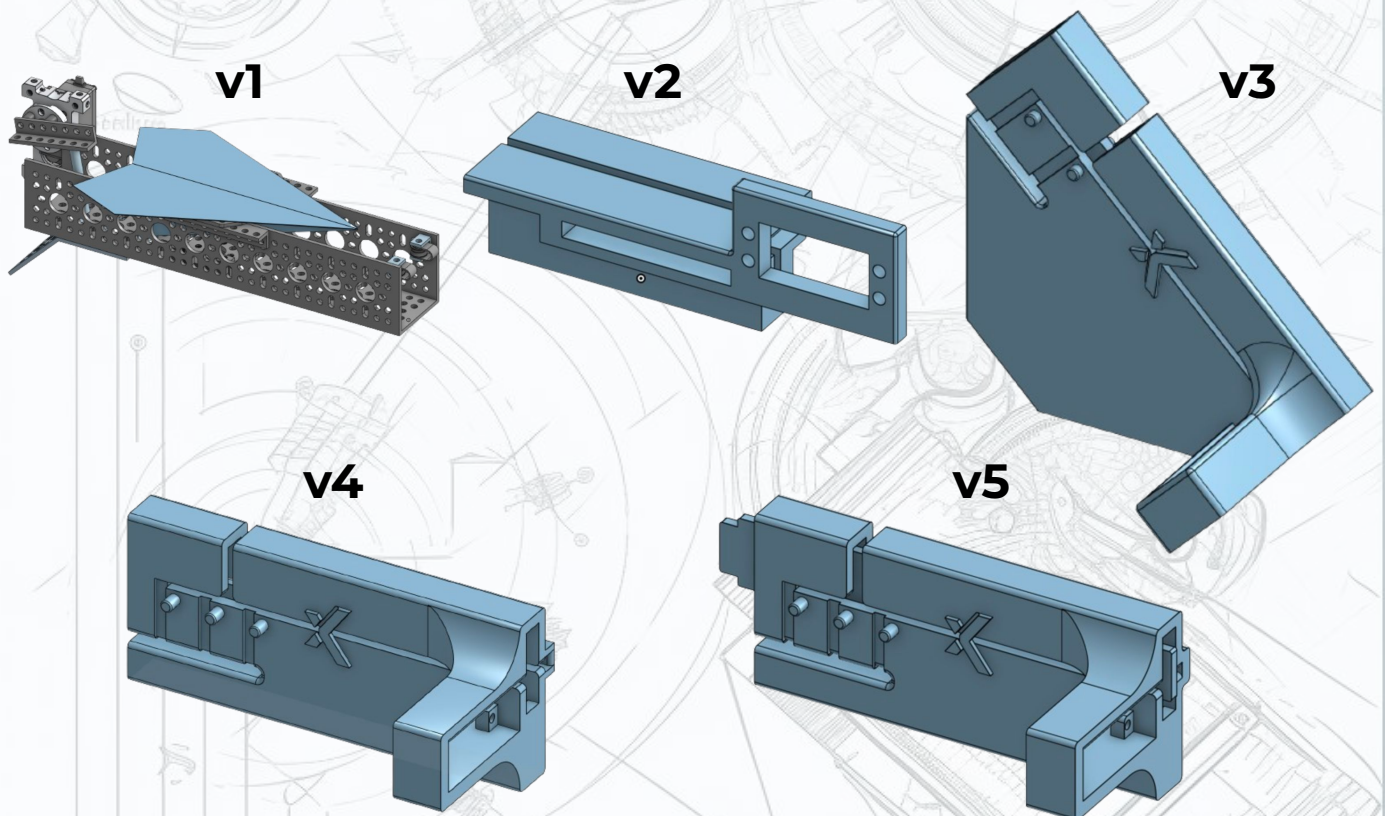


X Launcher Design

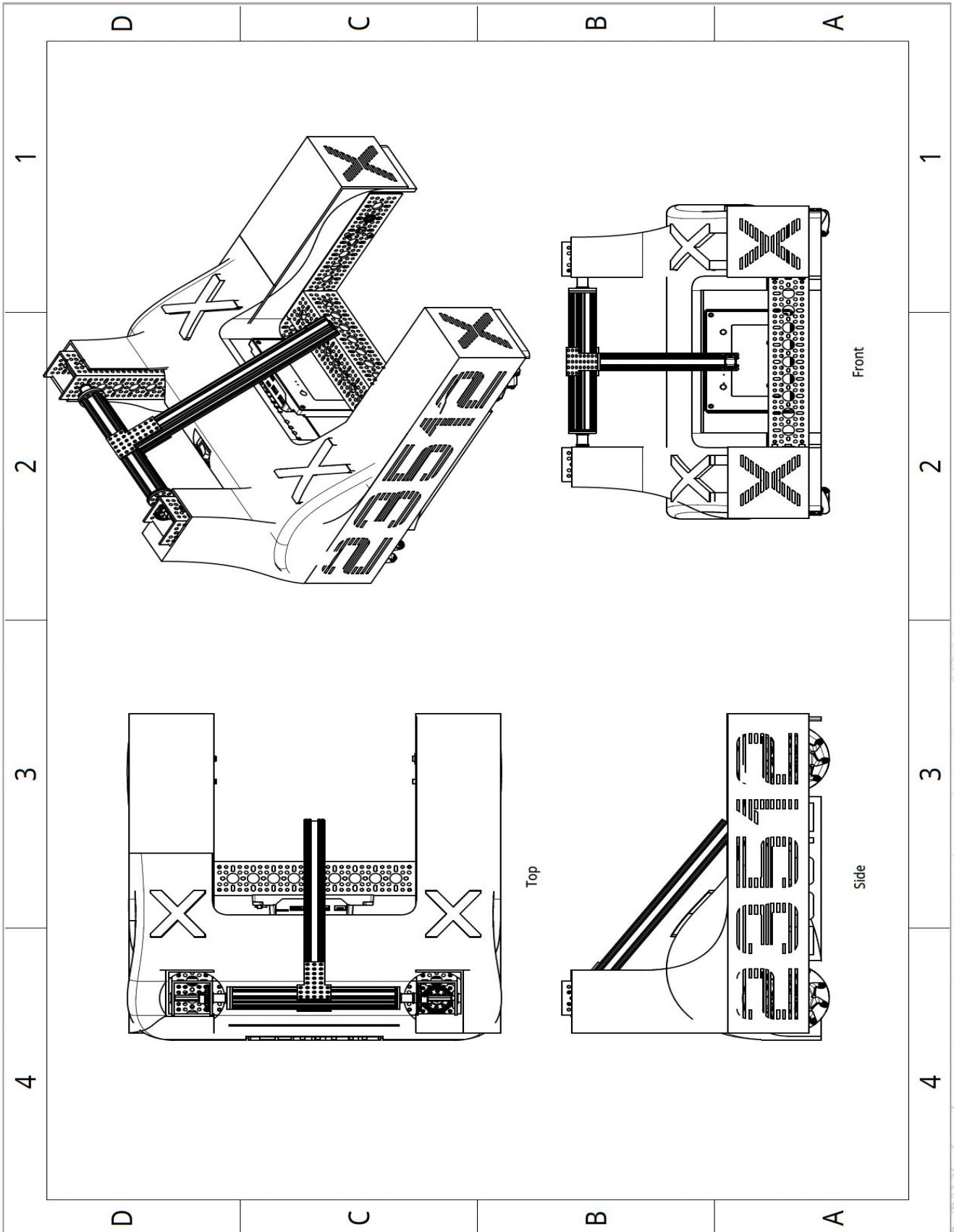
After experimenting with five different versions, our team settled on the **X launcher** as our final choice. This innovative mechanism incorporates various tension settings to hold back a rubber band, along with a servo that releases it. This allows us to consistently get 20 to 30 points.

Our development process can be traced through the five iterations we undertook, showcasing the evolution of our design over time. Initially, we utilized goBILDA channels, but gradually transitioned to a more customized fabrication approach, eventually relying entirely on 3D printing.

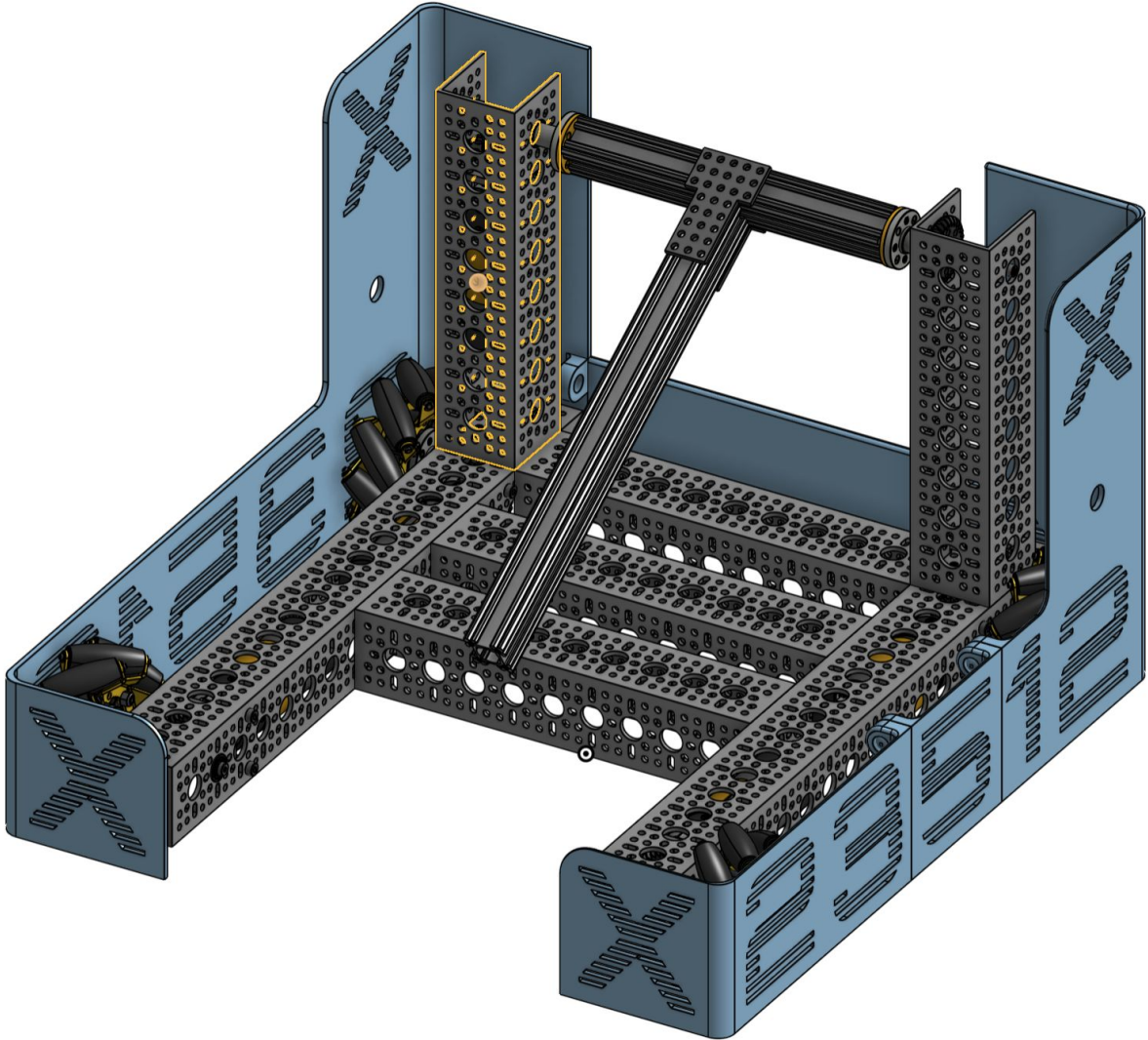
Our latest design also tackles the most common problem encountered during gameplay: the drone falling off the robot. We achieved this by eliminating the need to expose the drone's wings and instead storing the entire drone securely inside the launcher itself.



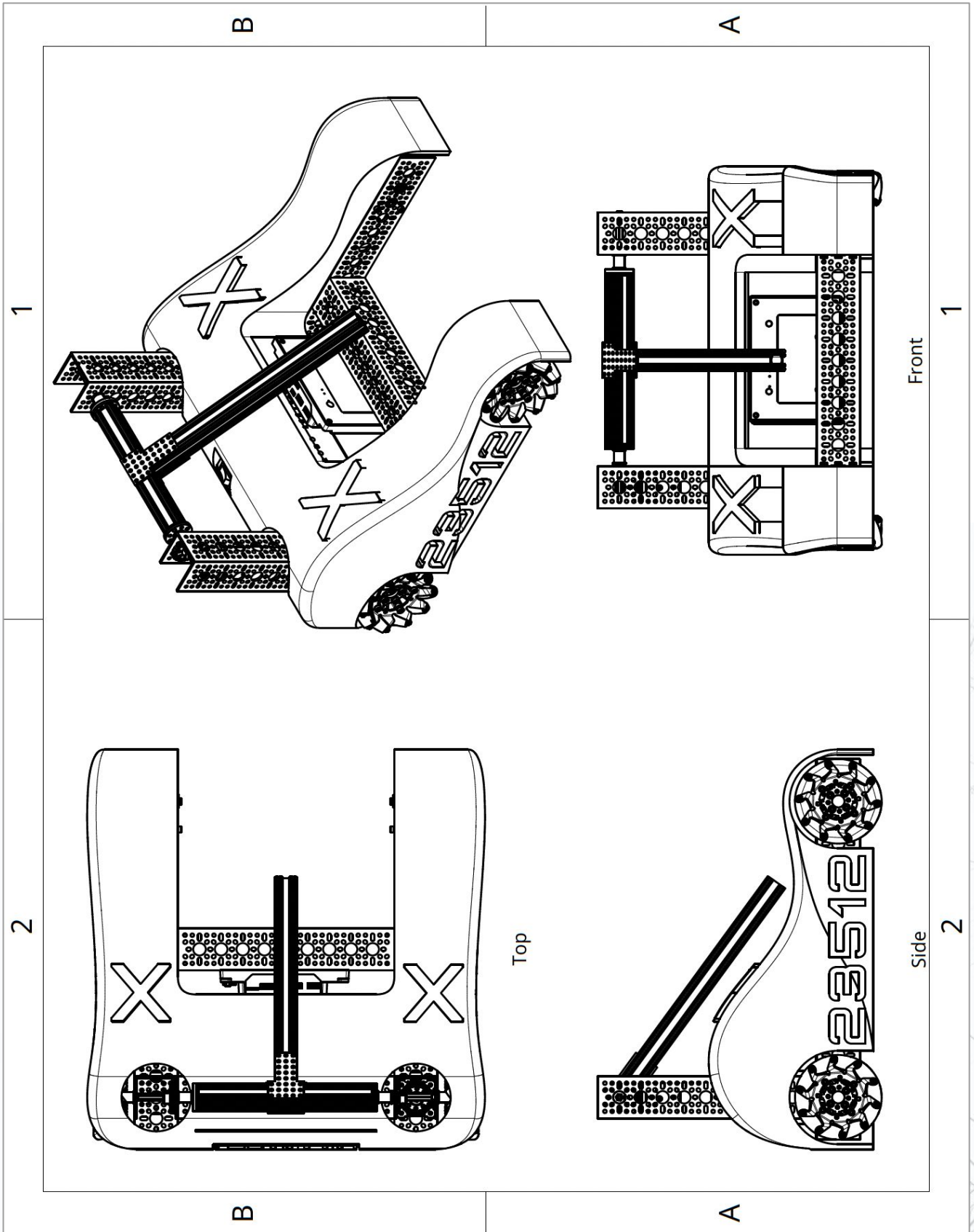
XBot Iteration 1



XBot Iteration 1



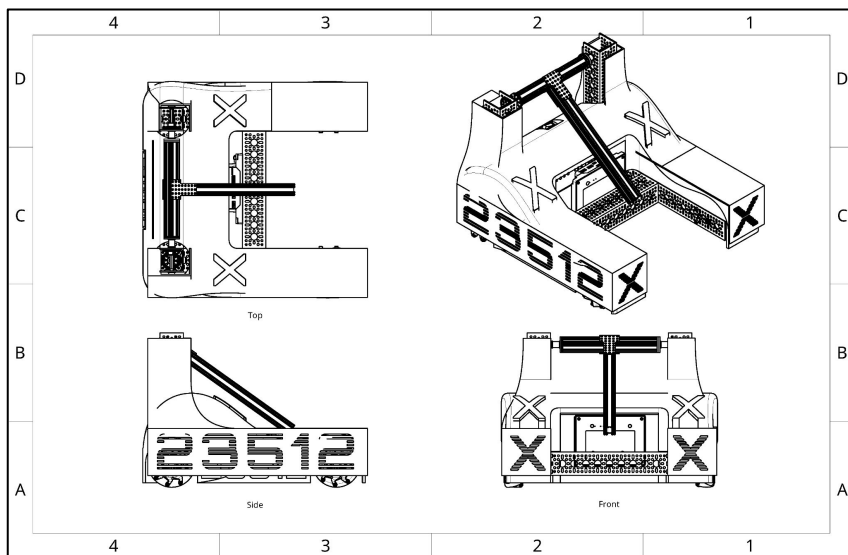
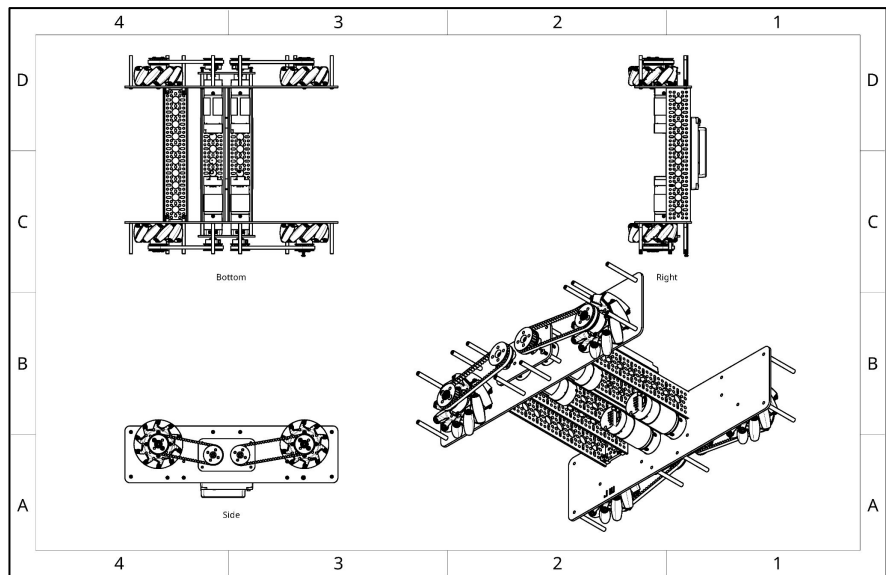
Batboy Iteration 2



Robot Design & Performance

Design is crucial because it allows for people to prototype their ideas without physically buying parts and building it. This is very convenient because it saves time and money, and reduces the excess amount of iterations. Many rookie teams decide to free-build their robots to save time. However, design can save you time by eliminating unnecessary iterations from the building process. Ultimately, design is necessary for teams to get the most out of the least.

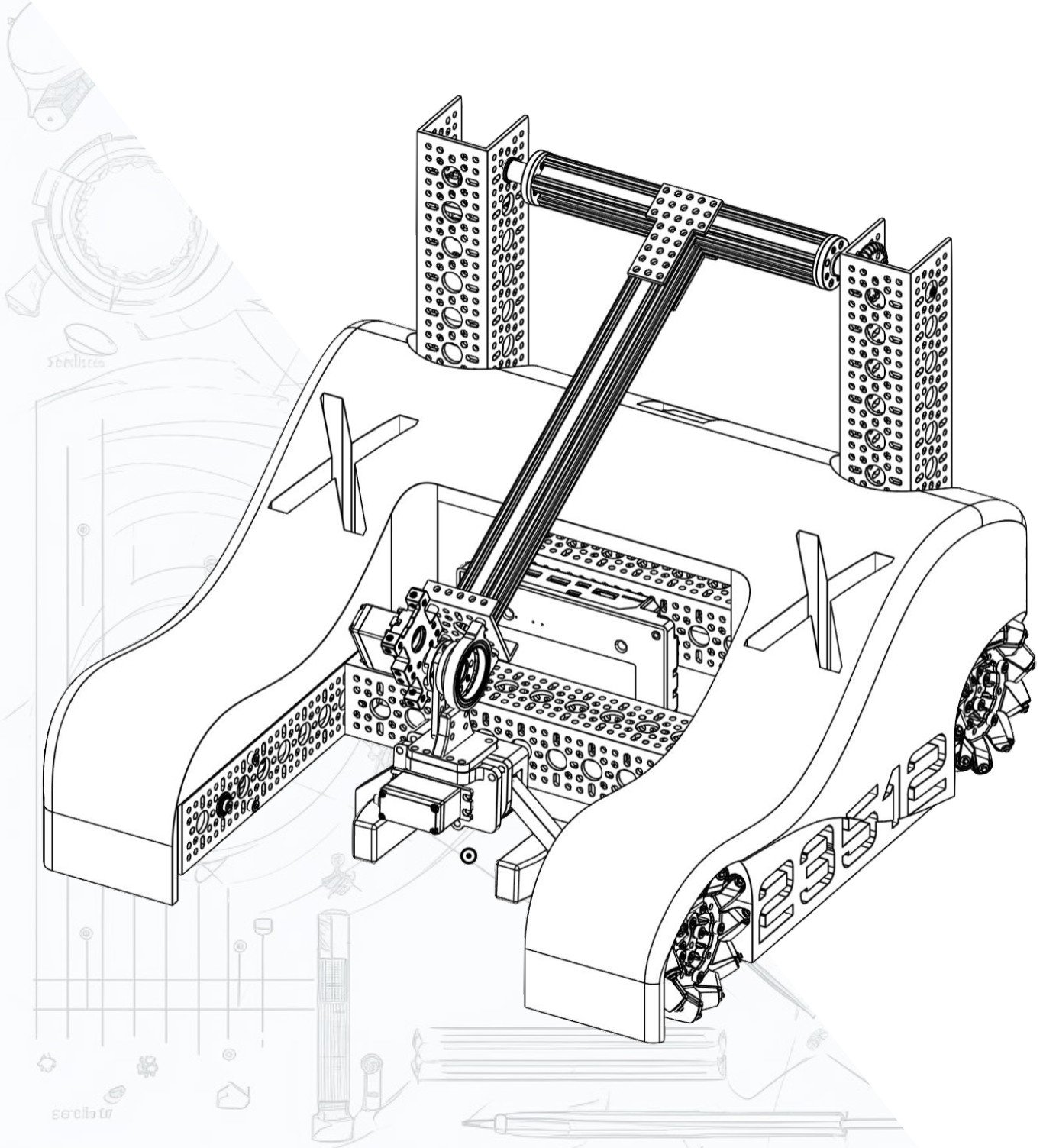
The image on the right is a schematic of our first drivetrain concept. It is a parallel plate drivetrain inspired by Tristan. We thought of using this, but the amount of time and resources needed, as a rookie team, was not ideal.



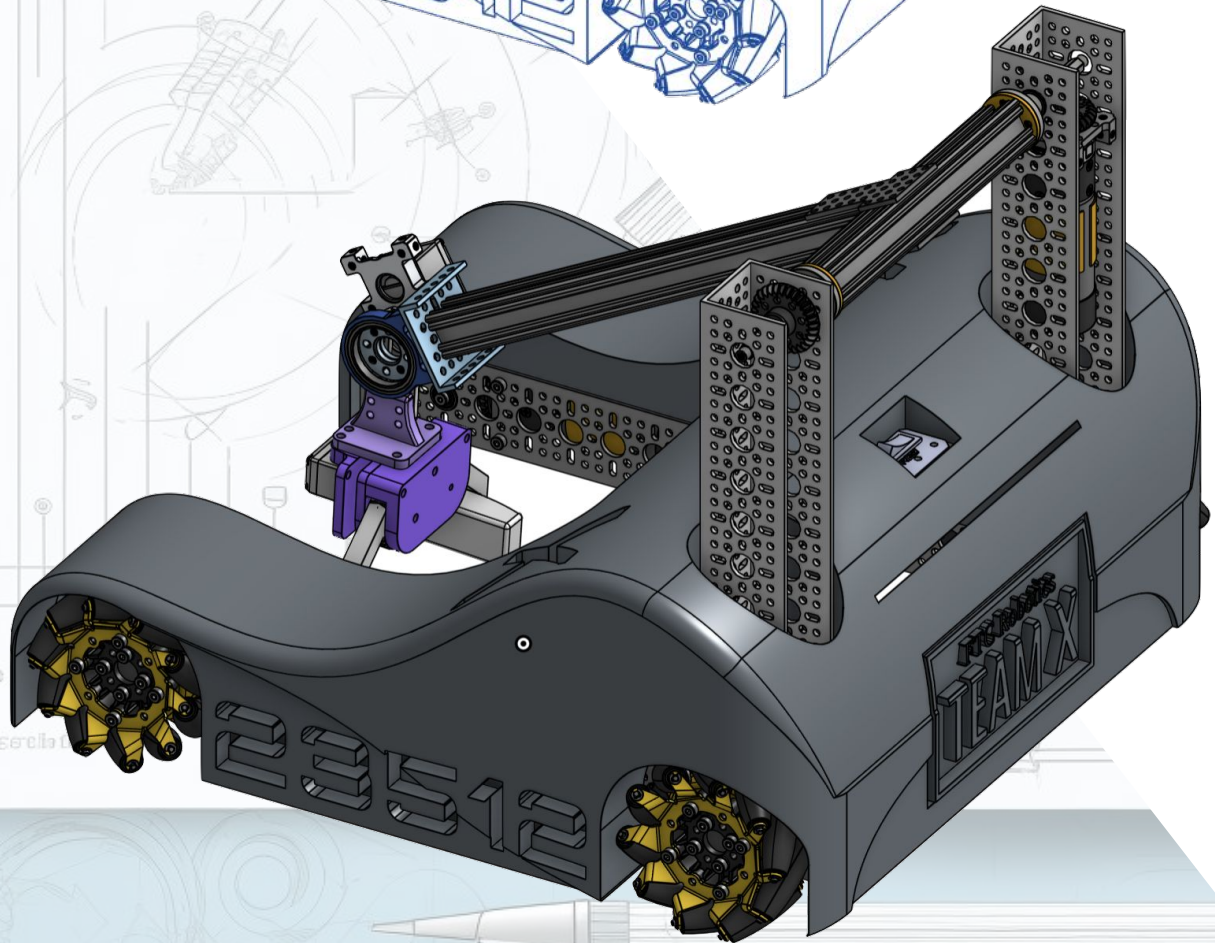
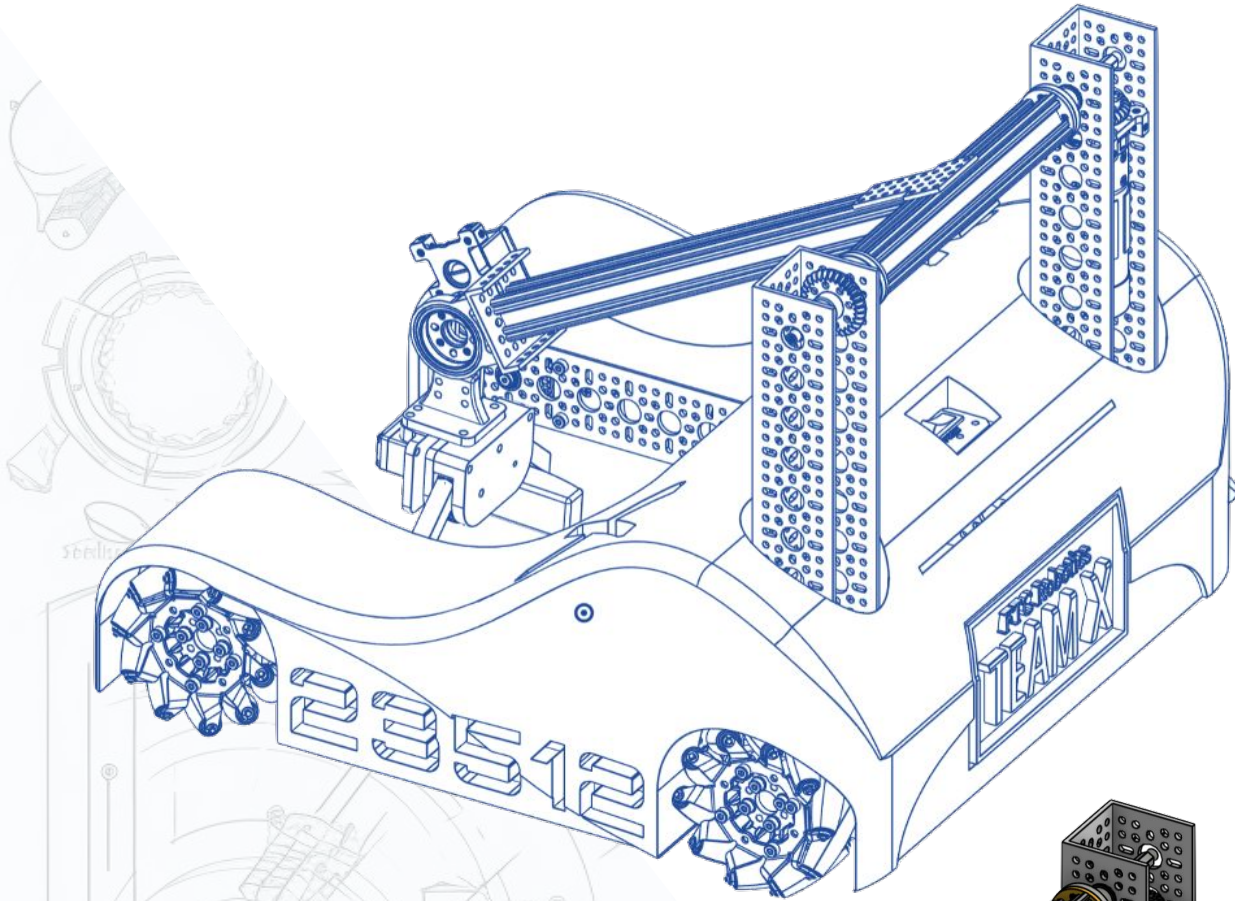
The image on the left was our second rendition of the drivetrain. We wanted to go for form and function. This design was a sleek way to conceal our robot's internals and manage our cables, while displaying our team number. This design was then further enhanced into our final design.



Batboy - Qualifier 1



Batboy - Qualifier 1



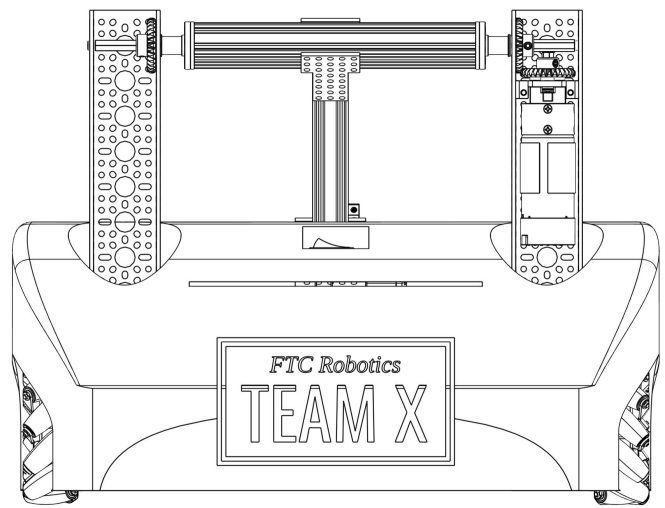
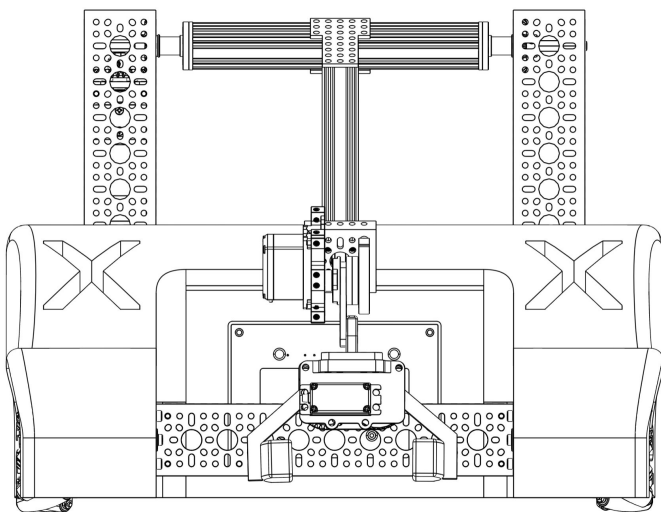
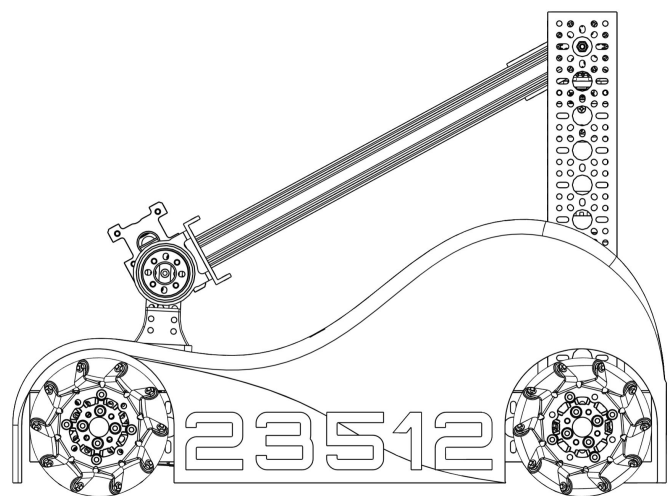
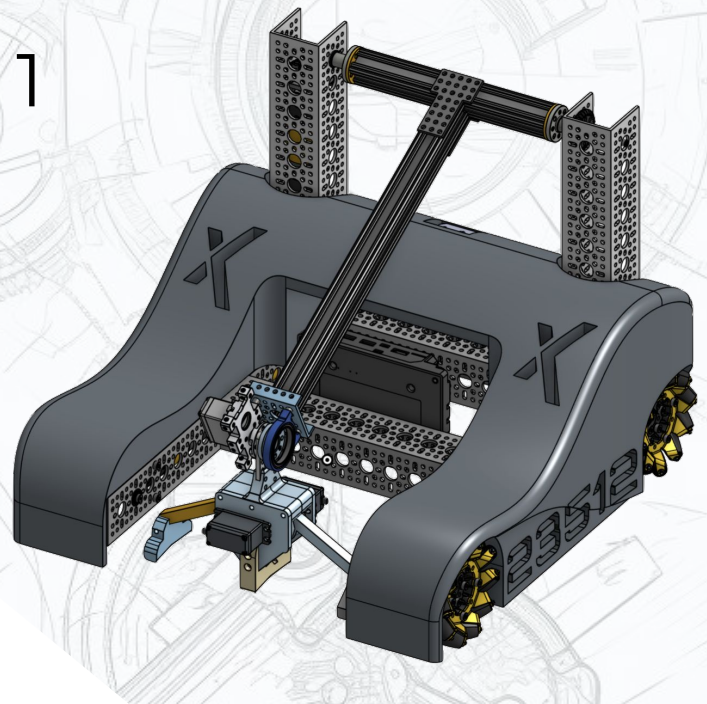
Batboy - Qualifier 1

This is our ultimate robot creation, embodying the essence of simplicity, elegance, and top-notch performance.

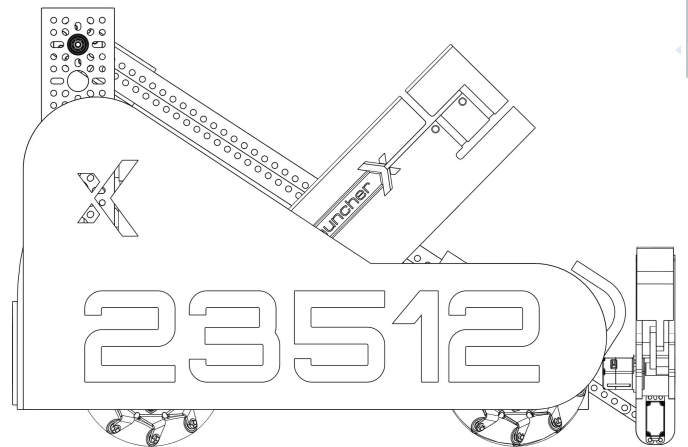
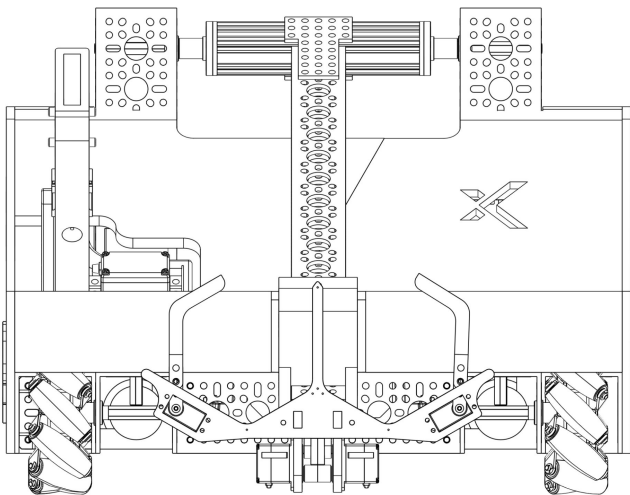
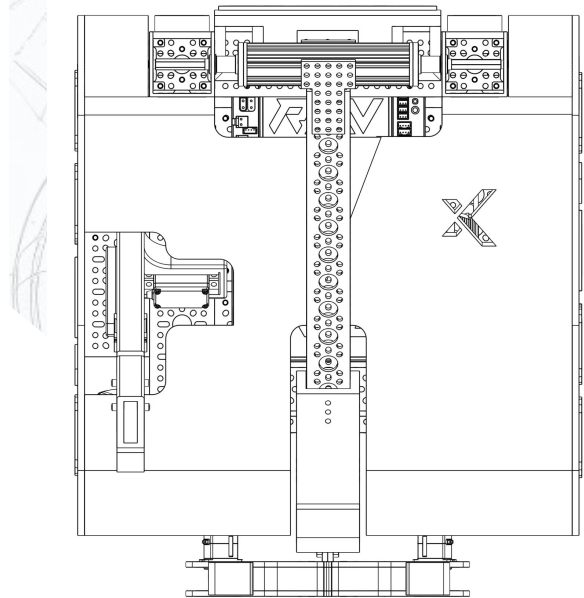
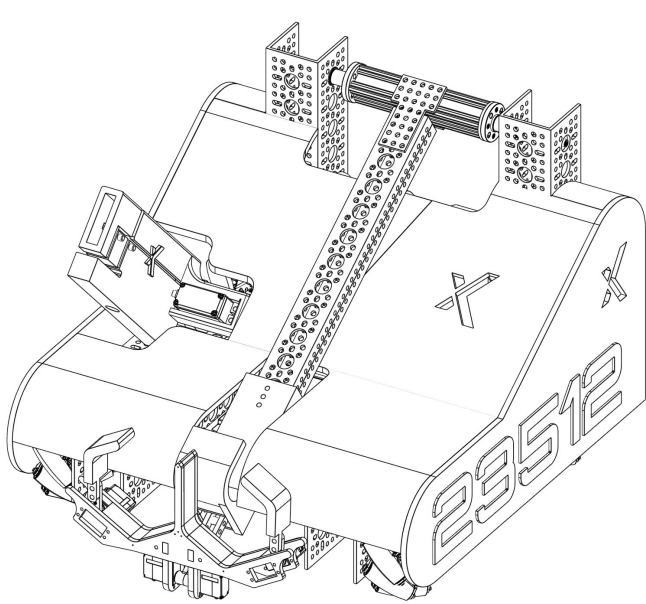
Drawing inspiration from the human body, this marvel employs basic components—a nimble **Arm**, a versatile **Wrist**, and a formidable two finger **Claw**.

Its *straightforward, lightweight, swift, and predictable* design ensures exceptional performance. The sleek and modern aesthetic sets it apart as a daring innovation that pushes the design boundaries of an FTC Robot!

Note: The design can easily be enhanced to incorporate an **Elbow**, increasing robot's vertical reach dramatically, all while keeping its height under 14 inches



Batman CAD Designs (Latest)



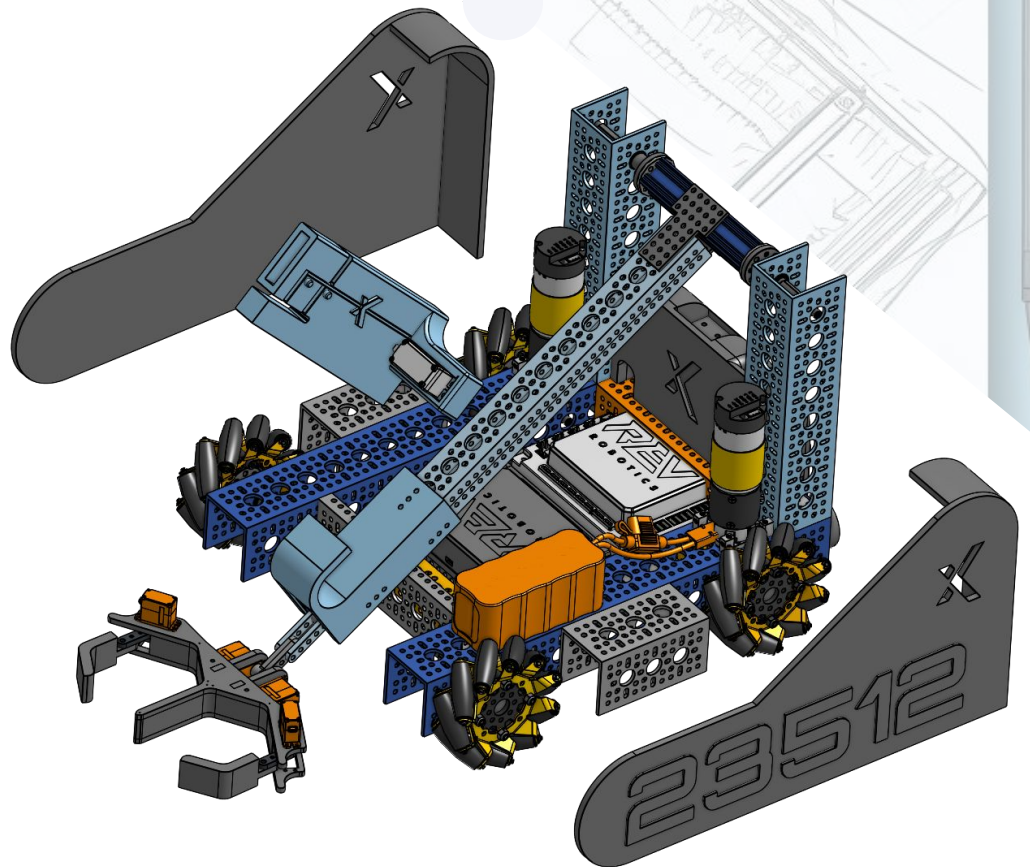
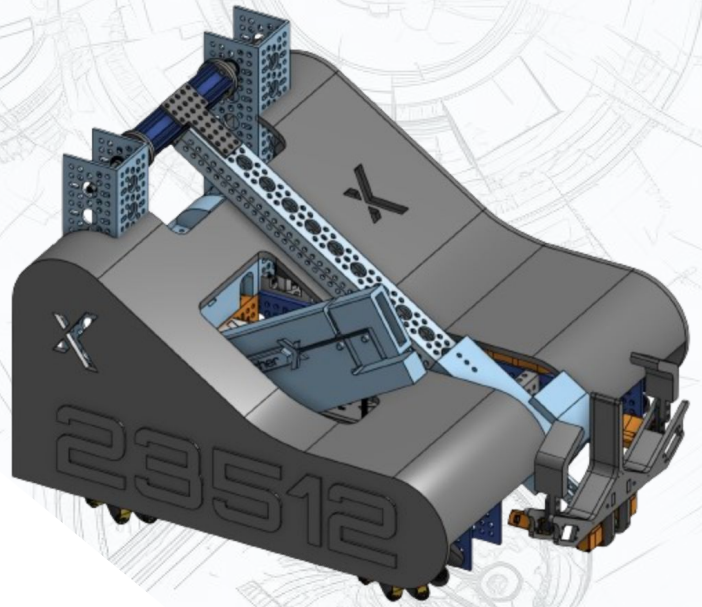
Batman

This is our ultimate robot creation, embodying the essence of simplicity, elegance, and top-notch performance. Drawing inspiration from the human body, this marvel employs basic components — a nimble **Arm**, a versatile **Wrist**, and a formidable two finger **Claw**.

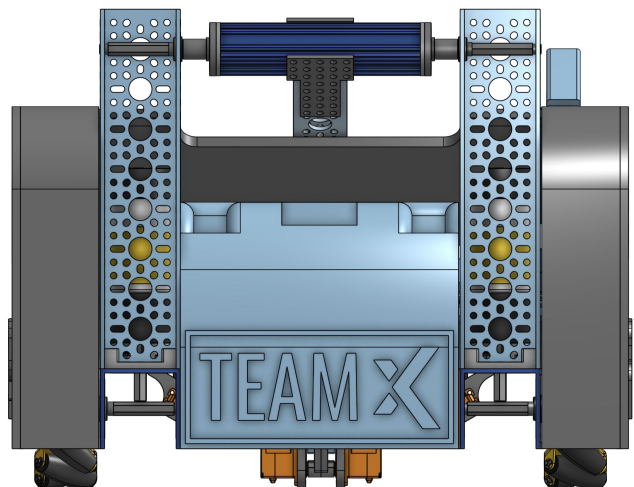
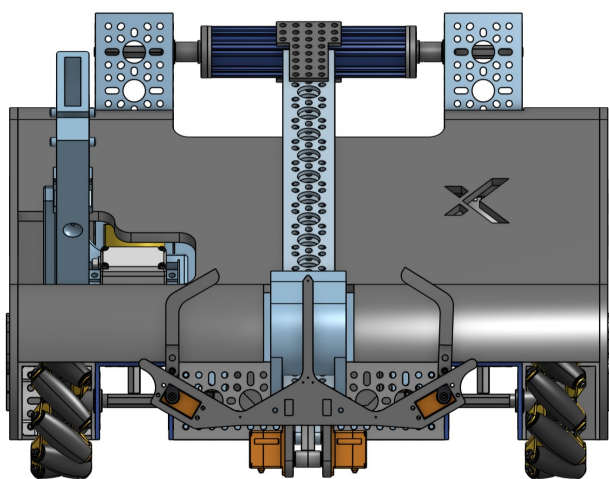
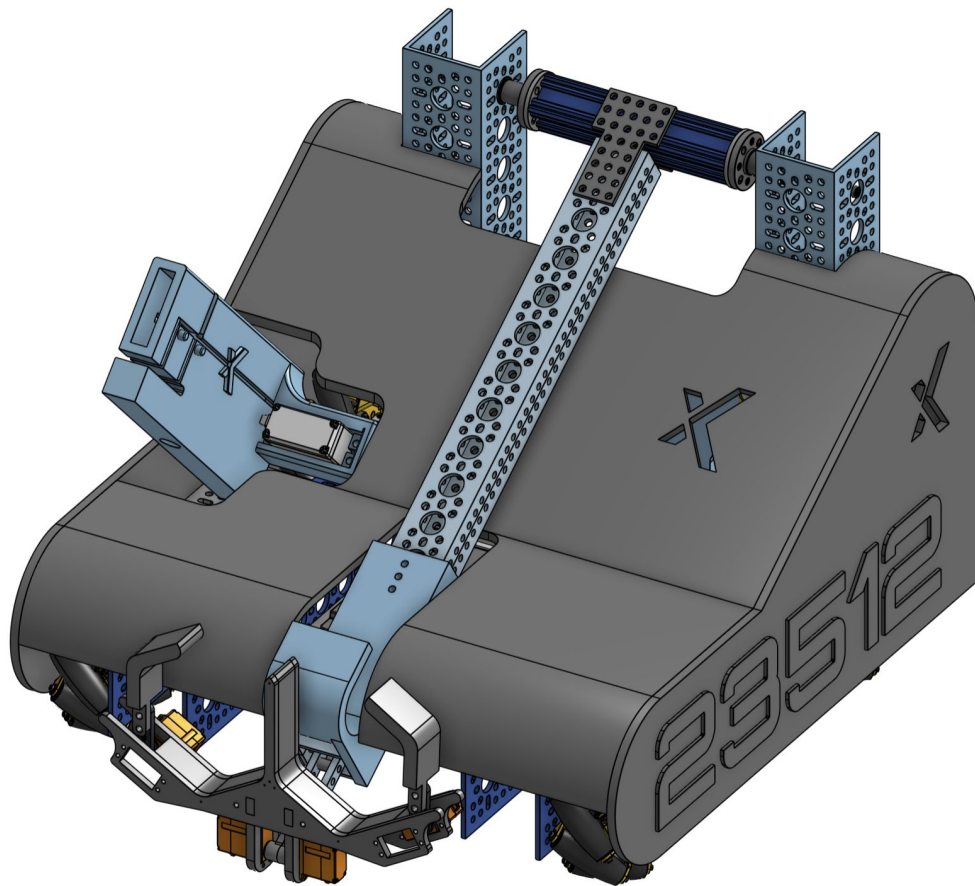
X-Bot's straightforward and lightweight design enables exceptional performance.

Sleek and modern aesthetics sets our robot apart and pushes the design boundaries.

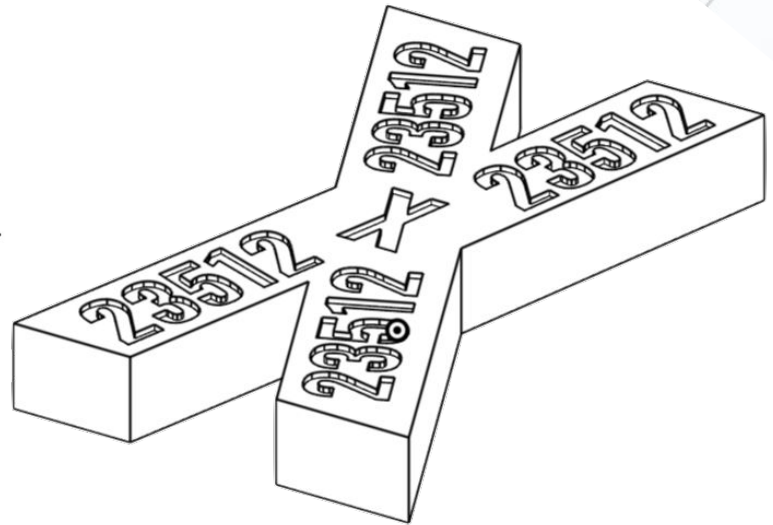
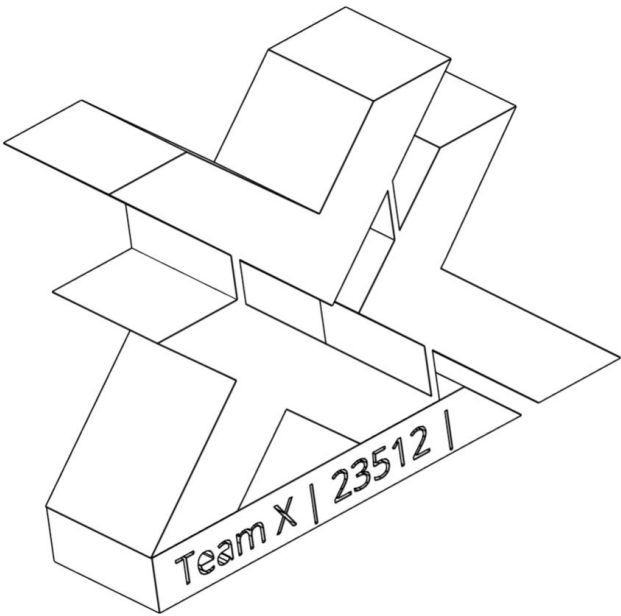
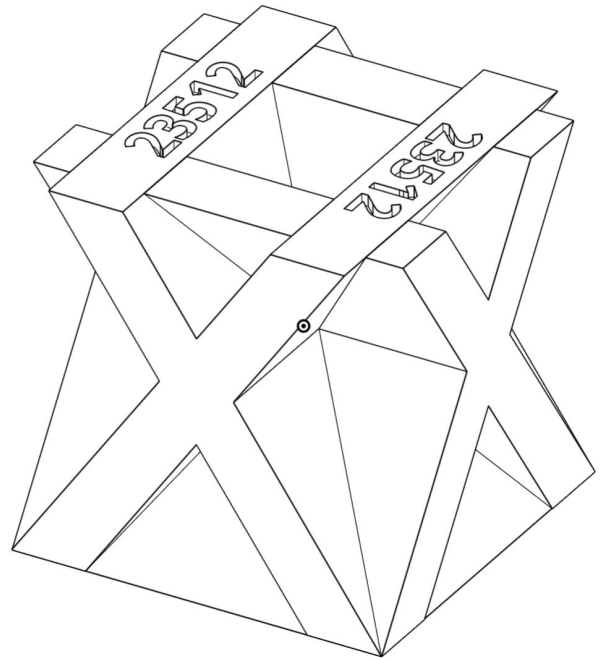
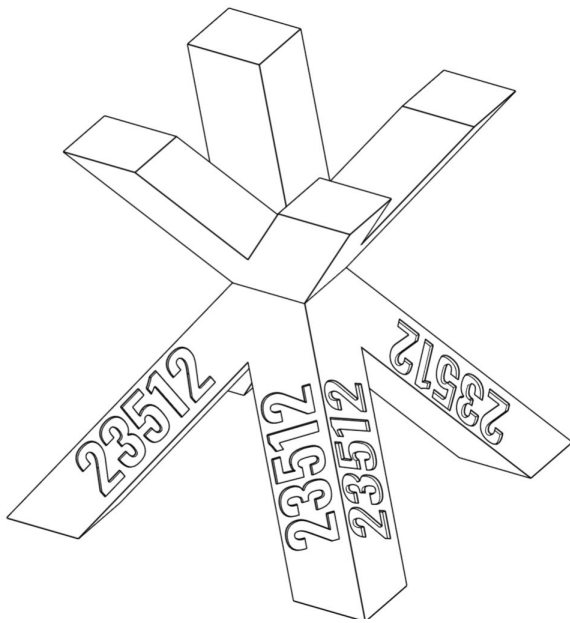
X-Bot measures 14 by 14 inches and stands at a height of 12 inches. Its small size facilitates easier robot maneuvering for the driver, while its odometry system continuously tracks the robot's position on the field at all times.



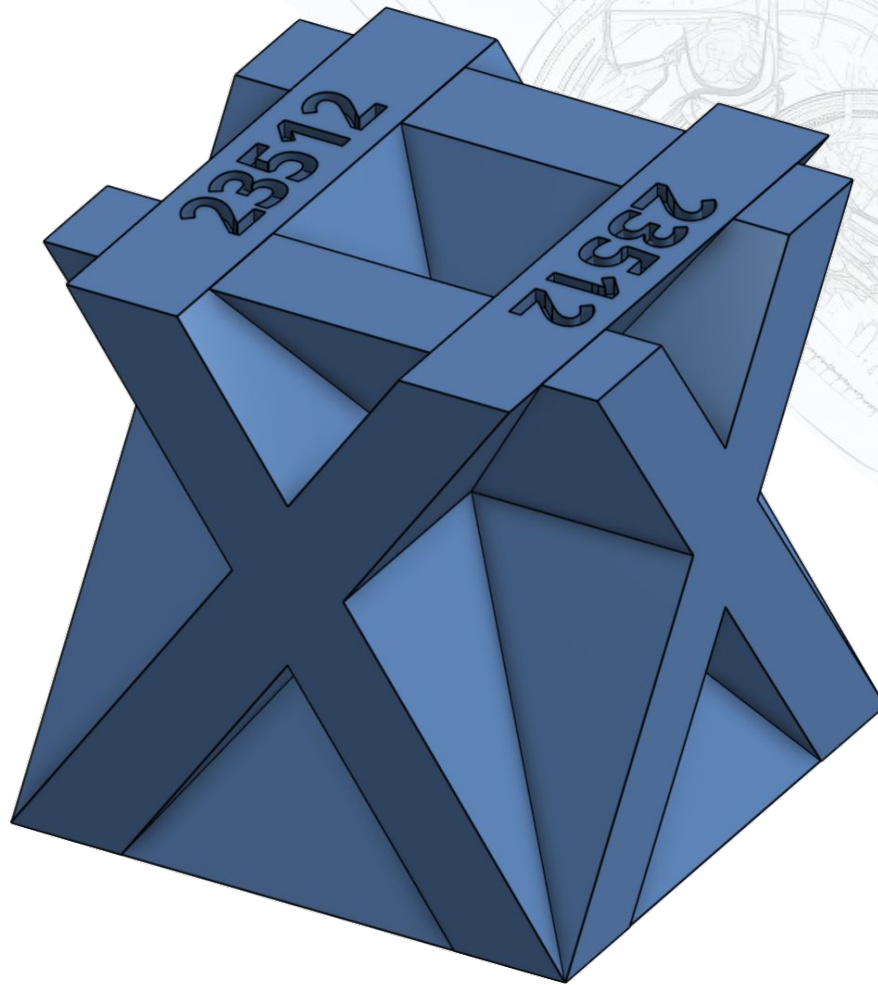
Batman HD Designs



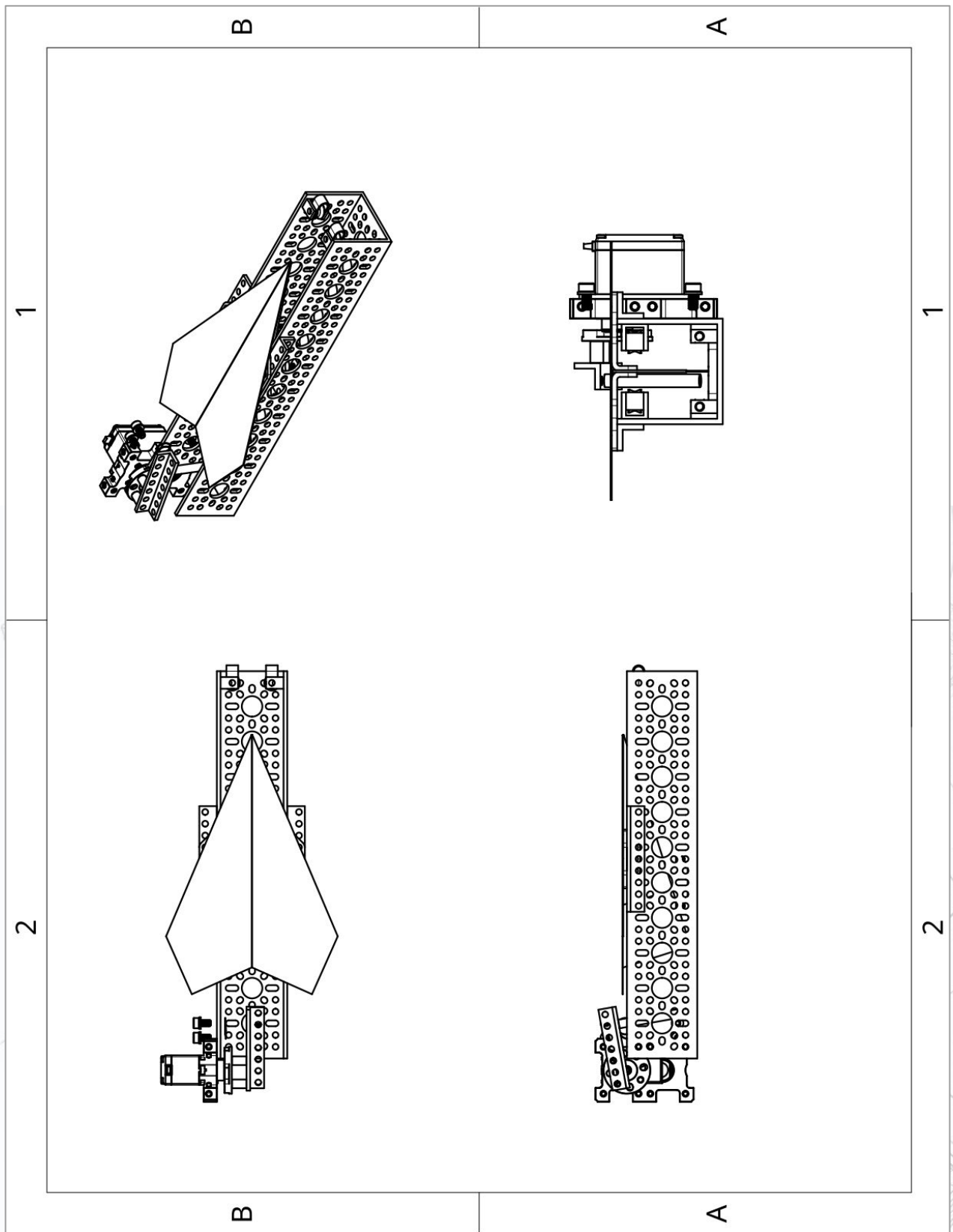
TeamX Prop - Design ideas



Custom Team Prop



Drone Launcher - 1st Iteration



Software Design

Programming Decisions

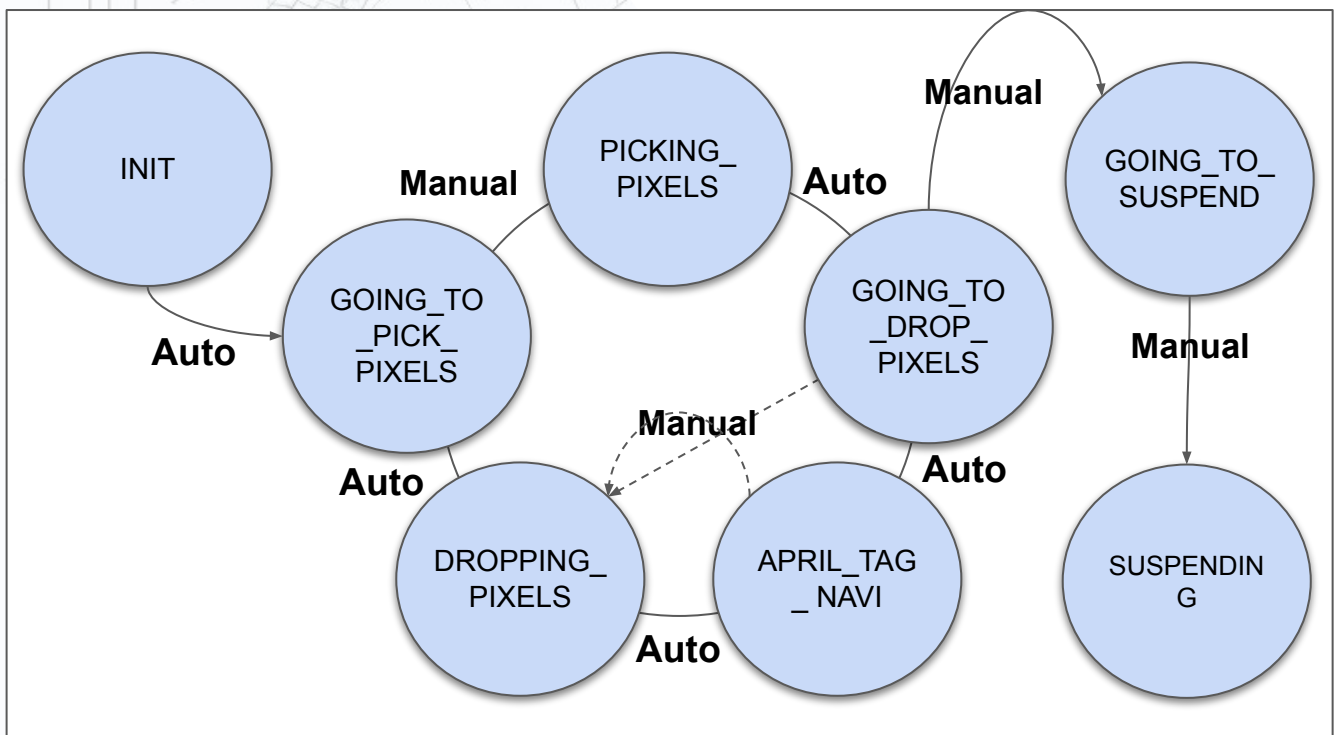
Development of Java software is done on Android Studio that allows us to program the robot during autonomous and control the robot during Tele-op.

Autonomous Goals

Our main goal this year is to deposit Pixels on the backdrop, deliver Pixels, and park as consistently as possible. To achieve this consistency, we needed an accurate way to track our movement and make sure we are always reaching our desired location every single time. This led us to use drive motor encoders to track our forward/backward motion and the built in IMU to track how much we turn.

State Machine

We decided to use a State Machine (using **Game Modes**) for designing and implementing complex logic and achieve clear, maintainable, and predictable robot behavior. State machine enhances *readability*, *modularity*, and *testability* of our code, making it easier to develop and maintain high-quality working Robot.



Software Programming and AI

Going to Pick Pixels

Robot opened claw faces ground and arm moves upwards. Robot speed is set to max power allowing us to travel the field in under 5 seconds. This

Picking Pixels

Robot slows down to decrease precision and allows us to auto grab or manually grab pixels

Going to Drop Pixels

Robot increases speed to minimize time took to travel the field while allowing us to look for an april tag. We used apriltags to help us automatically move the arm upwards and drop the pixel on the backdrop.

AprilTag Navigation

When the apriltag is detected, it takes full control of the robot and changes gamemode to dropping pixels

Dropping Pixels

The robot will only be able to strafe side to side and will allow the robots arm to move upwards to the backdrop and drop the pixels

Going to Suspend

The robot will move the arm upwards getting ready to hang

Suspending

The robot will move towards the bar and align the robot based on the bar. The robot will then retract the arm allowing it to suspend.



Game Mode

Code Blame 65 lines (57 loc) · 2.84 KB

```
1 package org.firstinspires.ftc.teamcode;
2
3 // INIT -> GOING_TO_PICK_PIXELS ->
4 // PICKING_PIXELS -> GOING_TO_DROP_PIXELS ->
5 // DROPPING_PIXELS -> GOING_TO_PICK_PIXELS
6 public enum GameMode {
7     NONE,
8     // Manual Game Initialized, Waiting for Play
9     // ENTER MODE == Hit the INIT button on Driver Station
10    // EXIT MODE == Hit the Play button on Driver Station
11    INIT,
12
13    // User driving Robot toward pixels, Robot is in motion
14    // Arm back side, wrist down, claw open, clearance from ground > 0
15    // ARM = AUTO, WRIST = AUTO, CLAWS = AUTO
16    // ENTER MODE == AUTO (after pixel drop) or Initial Play Button,
17    // EXIT MODE == USER ACTION changes game mode to PICKING_PIXELS
18    // Operator 1 = DRIVER, Operator 2 = ACTION BUTTON (engage at the right time)
19    GOING_TO_PICK_PIXELS,
20
21    // User picking pixels, Robot is already on top of pixels, ARM lowers down to ground level
22    // Arm back side, wrist down, claw manually operated, clearance from ground = 0
23    // ARM = AUTO, WRIST = AUTO, CLAWS = OPEN or CLOSE
24    // ENTER MODE = USER ACTION, -- lowers the arm closer to ground to pick pixels
25    // EXIT MODE == USER ACTION changes game mode to GOING_TO_DROP_PIXELS
26    // Operator 1 = DRIVER (Light engagement), Operator 2 = CLAWS and ACTION BUTTON (engage at the right time)
27    PICKING_PIXELS,
28
29    // User driving Robot toward back board, Robot is in motion
30    // Arm back side, wrist facing board, claw holding pixels
31    // WRIST = AUTO, CLAW = CLOSED (holding pixels)
32    // ENTER MODE = USER ACTION, -- Once pixels are picked
33    // EXIT MODE == USER ACTION changes game mode to DROPPING_PIXELS
34    // Operator 1 = DRIVER, Operator 2 = Move ARM to right height and ACTION BUTTON
35    GOING_TO_DROP_PIXELS,
36
37    // Detected April Tag and Arm turns towards back board, and drops pixel
38    // Arm back side, wrist facing board, claw open
39    // ARM = AUTO, WRIST = AUTO, CLAWS = OPEN
40    // ENTER MODE = AUTO NAVIGATION
41    // EXIT MODE == AUTO -- right when pixels drop
42    // Operator 1 = DRIVER (Light engagement), Operator 2 = ARM CONTROL, OPEN CLAWS
43    APRIL_TAG_NAVIGATION,
44
45    // Arm back side, wrist facing board, claw open
46    // ARM = MANUAL, WRIST = NONE, CLAWS = OPEN
47    // ENTER MODE = USER ACTION,
48    // EXIT MODE == AUTO -- right when pixels drop
49    // Operator 1 = DRIVER (Light engagement), Operator 2 = ARM CONTROL, OPEN CLAWS
50    DROPPING_PIXELS,
51
52    // Mode should only be activated when Robot is in position
53    // Moves ARM high up to hang the Robot
54    // ARM = AUTO, WRIST = AUTO, CLAW = AUTO
55    // ENTER MODE = USER ACTION,
56    // EXIT MODE == AUTO -- When Robot arm reaches the right position
57    // Operator 1 = Operate ARM
58    GOING_TO_HANG,
```



enums

SpikeMark

Code Blame 14 lines (11 loc) · 252 Bytes

```
1 package org.firstinspires.ftc.teamcode;
2
3 public enum SpikeMark {
4     LEFT(1),
5     CENTER(2),
6     RIGHT(3);
7     private final int value;
8
9     SpikeMark(final int newValue) {
10         value = newValue;
11     }
12
13     public int getValue() { return value; }
14 }
```

Parking

Code Blame 13 lines (10 loc) · 233 Bytes

```
1 package org.firstinspires.ftc.teamcode;
2
3 public enum Parking {
4     LEFT(1),
5     RIGHT(2);
6     private final int value;
7
8     Parking(final int newValue) {
9         value = newValue;
10    }
11
12    public int getValue() { return value; }
13 }
```

Alliance

Code Blame 13 lines (10 loc) · 233 Bytes

```
1 package org.firstinspires.ftc.teamcode;
2
3 public enum Alliance {
4     RED(1),
5     BLUE(2);
6     private final int value;
7
8     Alliance(final int newValue){
9         value = newValue;
10    }
11
12    public int getValue() { return value; }
13 }
```

DistanceFromBackdrop

Code Blame 13 lines (10 loc) · 257 Bytes

```
1 package org.firstinspires.ftc.teamcode;
2
3 public enum DistanceFromBackdrop {
4     NEAR(1),
5     FAR(2);
6     private final int value;
7
8     DistanceFromBackdrop(final int newValue) {
9         value = newValue;
10    }
11
12    public int getValue() { return value; }
13 }
```



autonomousPlay()

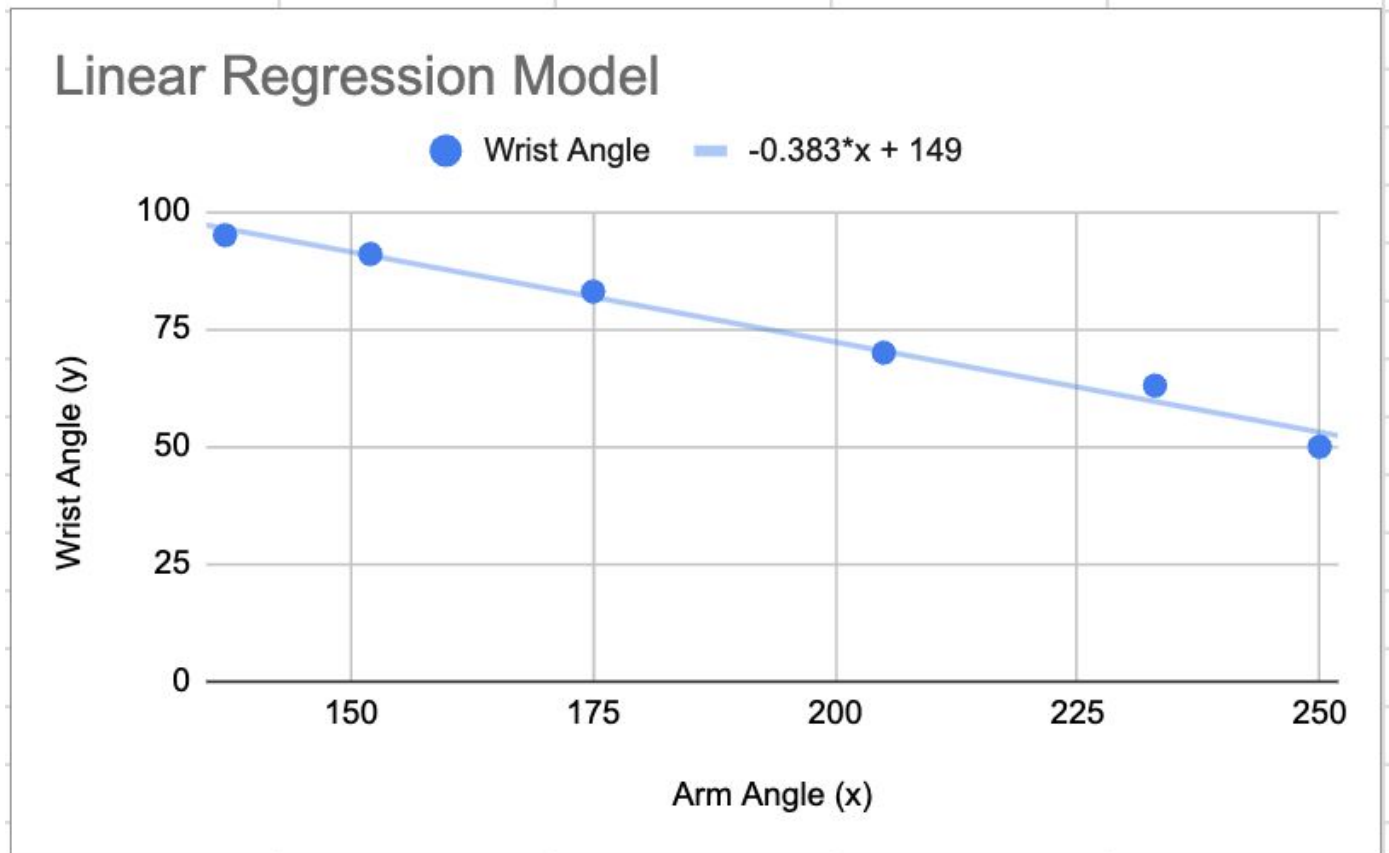
```
void autonomousPlay(Alliance alliance, DistanceFromBackdrop distanceFromBackdrop, Parking parking) {
    if (!teamPropDetectionCompleted) {
        detectTeamPropAndSwitchCameraToAprilTag();
    } else {
        telemetry.addData("SpikeMark", spikeMark + ", confidence" + detectionConfidence);
        telemetry.addData(">", "Robot Heading = %4.0f", getHeading());

        if (!spikeMarkPixelDropped) {
            switch (spikeMark) {
                case LEFT:
                    leftSpikeMark(alliance, spikeMark, distanceFromBackdrop, parking);
                    break;
                case RIGHT:
                    rightSpikeMark(alliance, spikeMark, distanceFromBackdrop, parking);
                    break;
                case CENTER:
                    centerSpikeMark(alliance, spikeMark, distanceFromBackdrop, parking);
            }
            telemetry.addData("Back Side", "Ready for April Tag Nav");
            spikeMarkPixelDroppedGetReadyForATagNav();
        }
        if (!aTagPixelDropped) {
            if (!arrivedAtBackDropTagPosition) {
                telemetry.addData("Looking for April Tag", desiredTagId);
                aprilTagNavMoveToDesiredTagPosition(alliance);
            } else {
                telemetry.addData("Arrived", "Dropping Pixel now");
                dropYellowPixel();
                //Park Now
                telemetry.addData("Parking", "");
                parkRobot(alliance, parking, spikeMark);
            }
        }
        telemetry.update();
    }
}
```



Linear Regression

Arm Angle	Wrist Angle	wrist angle	
137	95	0.95	
152	91	0.91	
175	83	0.83	
205	70	0.7	
233	63	0.63	
250	50	0.5	

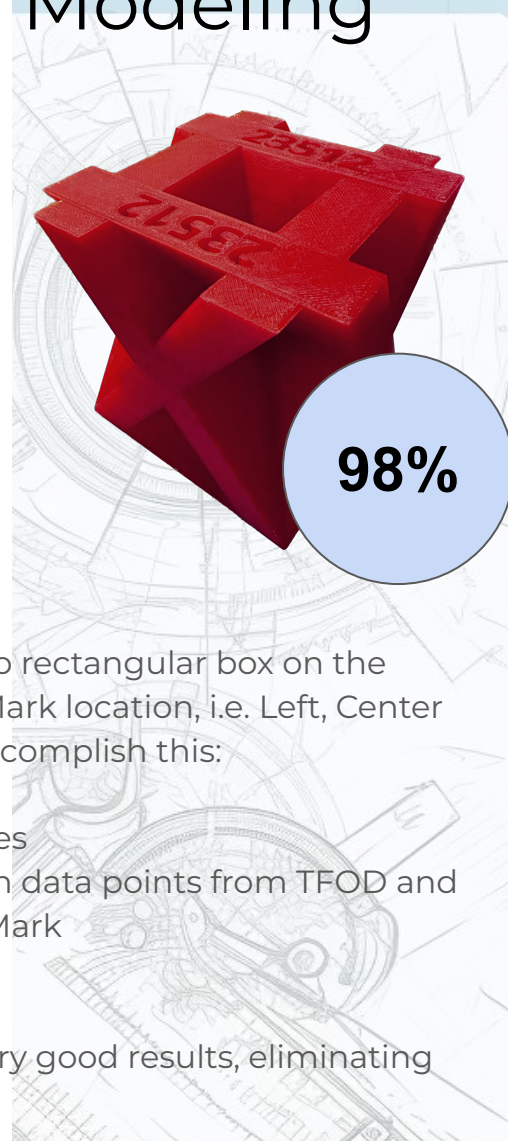


Machine Learning and Modeling

Machine Learning (Tensorflow)

We decided to use a custom Team X prop to replace the white pixel on Stripe Marks. Using custom prop meant we had to train our own Tensorflow Object Detection (TFOD) interface model as opposed to using the standard model provided by FTC to detect a white pixel.

Our custom model is able to identify the Team Prop with over **98% efficacy**



Statistical Modelling

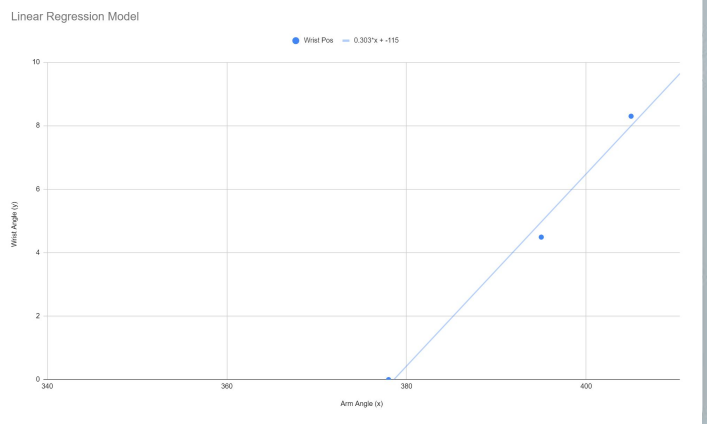
TFOD gives us the location and size of the Team Prop rectangular box on the image. However, our goal is to figure out the Spike Mark location, i.e. Left, Center or Right. We explored three different strategies to accomplish this:

1. Mapping image coordinates to field coordinates
2. Machine Learning model using 100s of location data points from TFOD and labelling them for Left, Center or Right Spike Mark
3. Statistical modeling

We discovered that statistical modeling delivered very good results, eliminating the need for more complex solutions.

Linear Regression

In order to eliminate user error and improve speed, we decided to move the Wrist automatically based on the Arm movement. To calculate the Wrist Angle based on the Arm angle, we used linear regression (as shown in the chart on the right)



Odometry and Road Runner



Road Runner

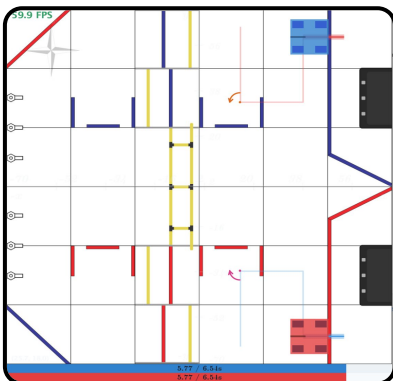
We use Road Runner, a motion planning library, written for the FTC robotics competition. This helped us in designing complex path following while maintaining control of velocity & acceleration during the autonomous time period.

3-Wheel Odometry and Encoders

We use 3-wheel odometry configuration to keep record of X-Bot's position and heading (Pose) at any time. Using two parallel odometry wheels to calculate heading is more accurate and higher performance as compared to relying in control hub's IMU.

Dead Wheels

A small unpowered wheel that tracks the distance the robot has traveled through the encoder attached to the wheel's axle. Generally, odometry wheels are sprung so that the wheel is in contact with the floor tiles at all times to ensure accuracy.



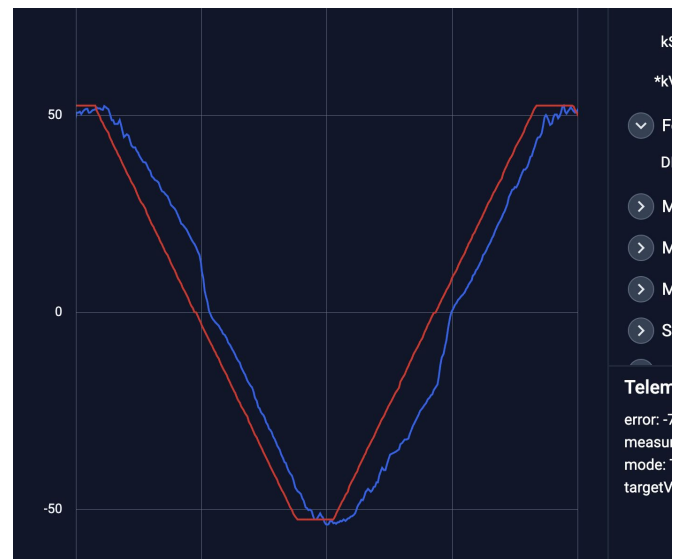
Meep Meep

We use Meep Meep to build and test trajectories before coding our Auto. It allows us to simulate the robot's movements without requiring a physical field. This sped up our autonomous programming process as we are able to determine the most efficient paths.



Feedforward Tuning

Tuning the feedforward controller for accurate following is necessary for accurate path following. Poor tuning of the feedforward controller result in errors later along the line. The goal is for the **poseVelocity** line to match the **targetVelocity** line as can be seen in the following charts:



In the above velocity vs. voltage graph, **kV** is the slope and **kStatic** is the y-intercept. To find **kV** and **kStatic**, the robot executes a quasi-static ramp test where the voltage is slowly ramped up to minimize acceleration. Throughout this procedure, the velocity and voltage are recorded. Goal is to optimize **kV** (Velocity), **kA** (Acceleration) and **kStatic** values such that both graphs overlaps

Follower PID Tuning

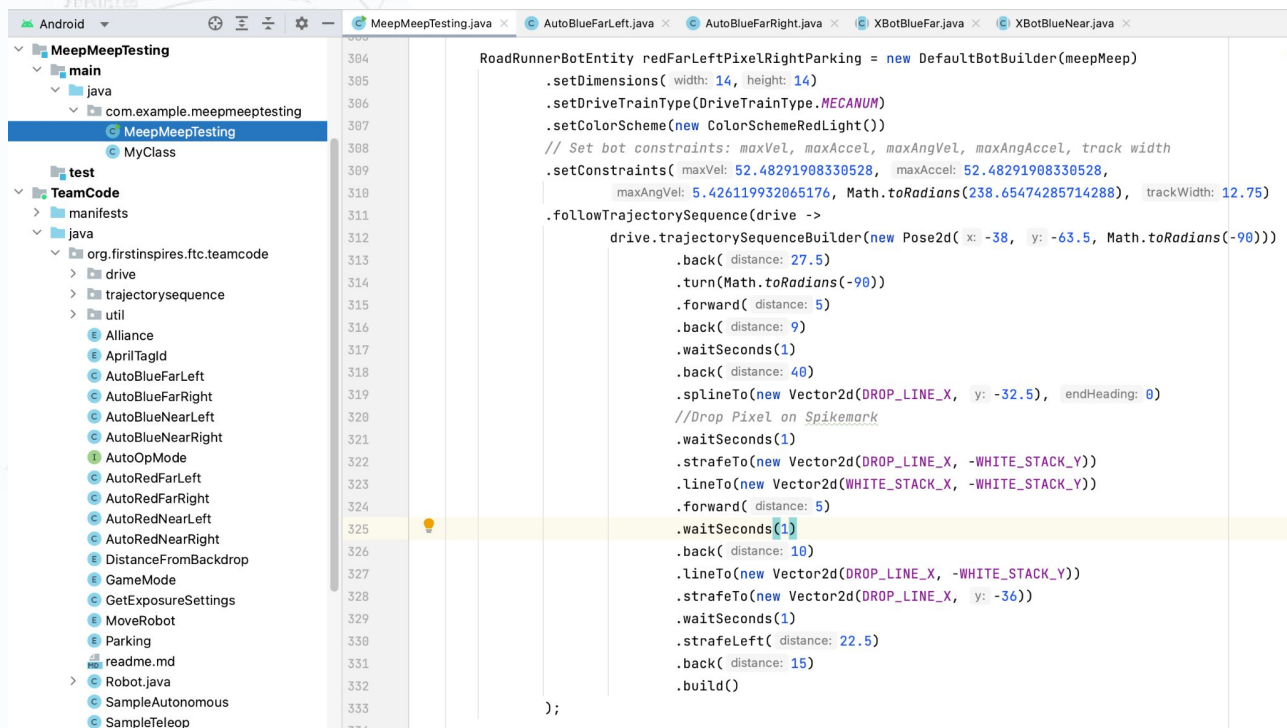
We tune two PID's in this step, the heading PID and the translational (x/y) PID. This enables closed-loop feedback control to ensure accurate path following.

Follower PID tuning involves adjusting the proportional, integral, and derivative gains of a PID controller used in trajectory following. The goal is to achieve precise path tracking by balancing responsiveness, stability, and accuracy while minimizing overshoot and oscillations through experimentation and optimization of the controller's parameters.



MeepMeep Testing

MeepMeep is a trajectory visualizer designed specifically for the Road Runner trajectory sequence API. We use MeepMeep to build and test trajectories before coding our Auto. It allows us to simulate the robot's movements without requiring a physical field. This sped up our autonomous programming process as we are able to determine the most efficient paths.

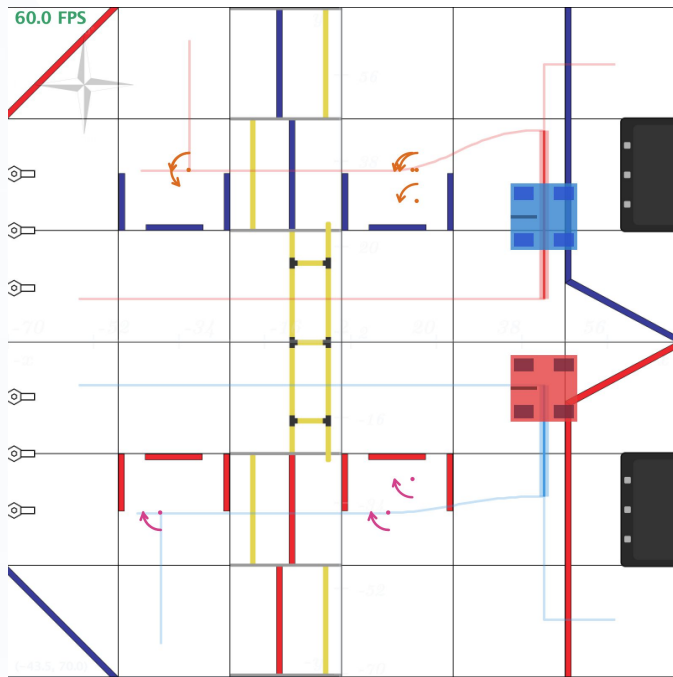


```
304 RoadRunnerBotEntity redFarLeftPixelRightParking = new DefaultBotBuilder(meepMeep)
305     .setDimensions( width: 14, height: 14)
306     .setDriveTrainType(DriveTrainType.MECANUM)
307     .setColorScheme(new ColorSchemeRedLight())
308     // Set bot constraints: maxVel, maxAccel, maxAngVel, maxAngAccel, track width
309     .setConstraints( maxVel: 52.48291908330528, maxAccel: 52.48291908330528,
310                    maxAngVel: 5.426119932065176, Math.toRadians(238.65474285714288), trackWidth: 12.75)
311     .followTrajectorySequence(drive ->
312         drive.trajectorySequenceBuilder(new Pose2d( x: -38, y: -63.5, Math.toRadians(-90)))
313             .back( distance: 27.5)
314             .turn(Math.toRadians(-90))
315             .forward( distance: 5)
316             .back( distance: 9)
317             .waitSeconds(1)
318             .back( distance: 40)
319             .splineTo(new Vector2d(DROP_LINE_X, y: -32.5), endHeading: 0)
320             //Drop Pixel on Spikemark
321             .waitSeconds(1)
322             .strafeTo(new Vector2d(DROP_LINE_X, -WHITE_STACK_Y))
323             .lineTo(new Vector2d(WHITE_STACK_X, -WHITE_STACK_Y))
324             .forward( distance: 5)
325             .waitSeconds(1)
326             .back( distance: 10)
327             .lineTo(new Vector2d(DROP_LINE_X, -WHITE_STACK_Y))
328             .strafeTo(new Vector2d(DROP_LINE_X, y: -36))
329             .waitSeconds(1)
330             .strafeLeft( distance: 22.5)
331             .back( distance: 15)
332             .build()
333     );
```

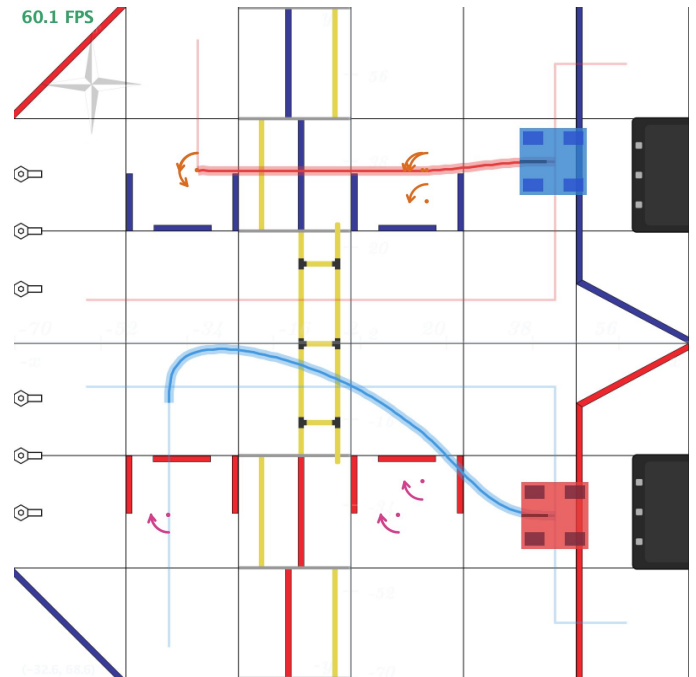
Above screenshot shows sample trajectory sequence for red Alliance with Team Prop on Left Spike Mark and Robot Parking on right side of the backboard.



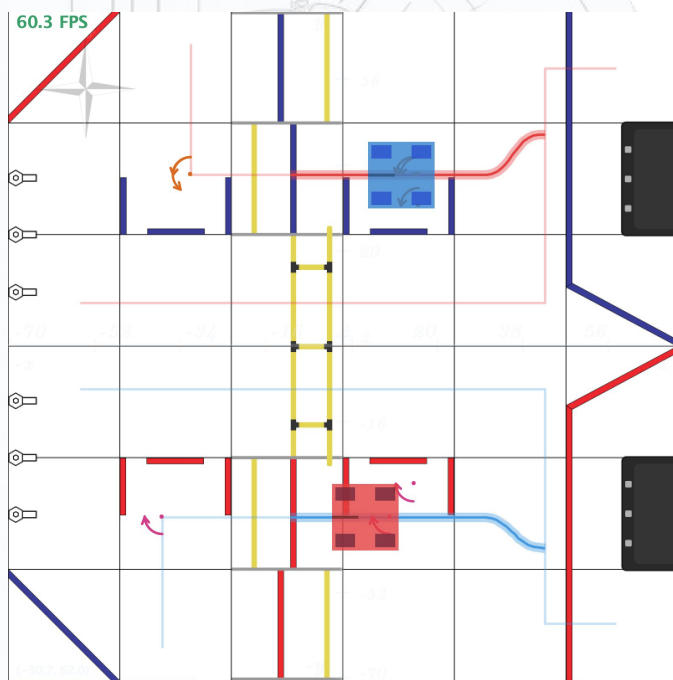
Autonomous Routes



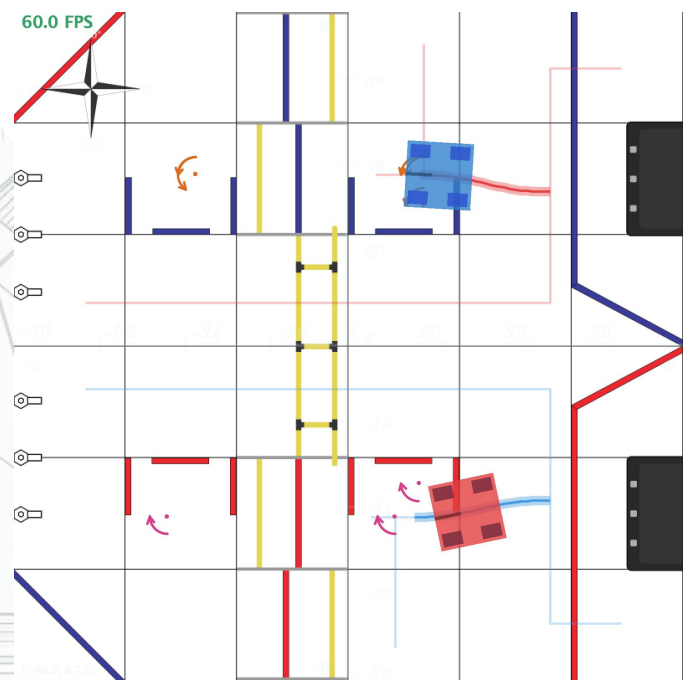
FAR: Blue: Right, Red: Left



FAR: Blue & Red: Center



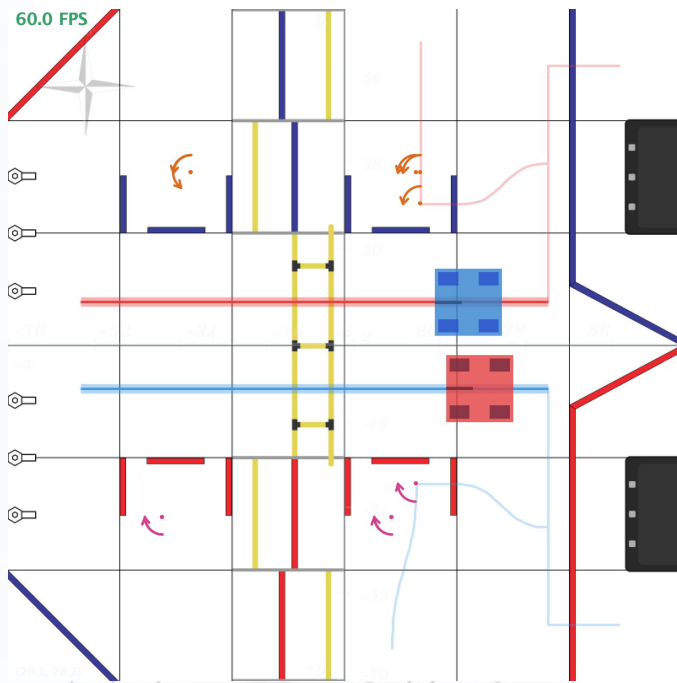
FAR: Blue: Left, Red: Right



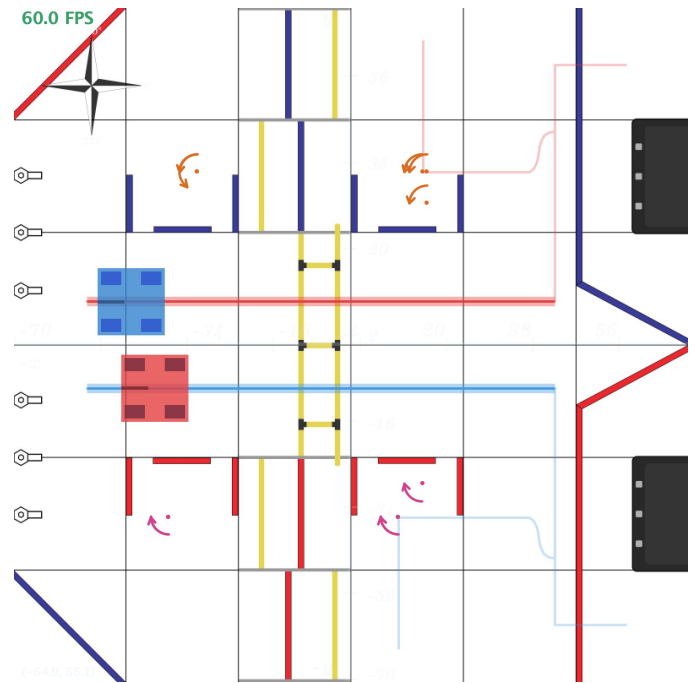
NEAR: Blue: Right, Red: Left



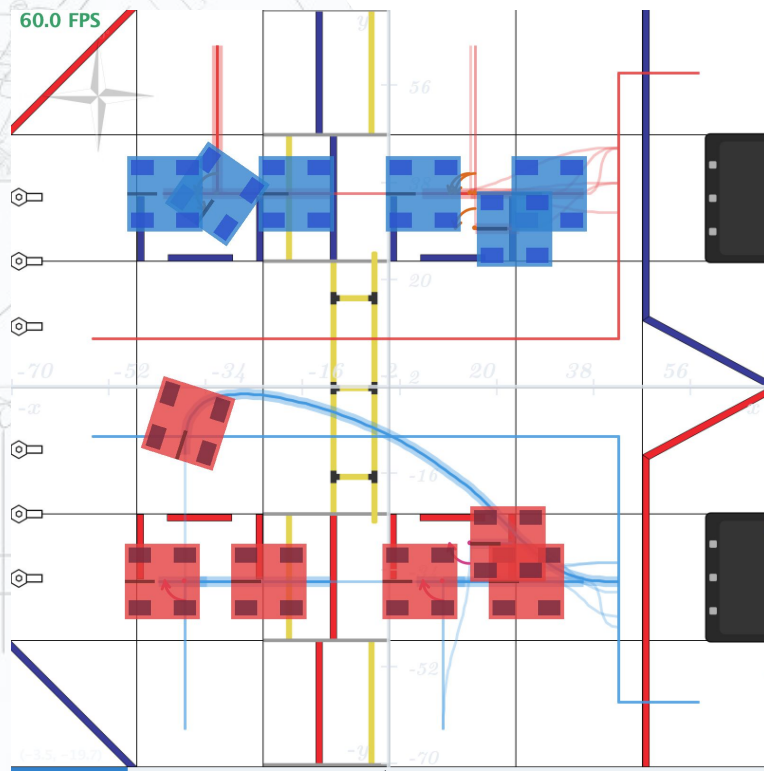
Autonomous Routes



NEAR: Blue & Red: Center



NEAR: Blue: Left, Red: Right



All Auto routes at once



Challenges and Solutions

Design Challenges

Since this is our rookie year, we didn't have a robot from previous year to start with. We began exploring different ideas for the robot design and came up with multiple options. To remove the emotional element from decision making, team decided to use a **scoring system** to pick the winner.

Challenges with Intake / Picking Pickel

We looked at different ways to take in pixels, checking out ideas online. But we rejected most of them because they were either too complex, not practical, or didn't work well. We liked the looney claw design, but didn't like the fact that it could only grab one pixel at a time. So, we made our own **Mousetrap Claw** using CAD that could auto-grab and auto-release multiple pixels at once.

Software Challenges

As we wrote more code and added more sensors and actions to the Robot, things got pretty complicated. Then, we discovered **State Machines**, which reduce complexity and help achieve clear, maintainable, and predictable robot behavior. We added **Game Modes** and reworked the program using a State Machine. This made the code easier to read, split into smaller parts, and test, making it simpler to build and take care of a well-functioning Robot.

Too many user actions / running out of buttons on gamepads

Adding more features to the robot in TeleOp mode meant needing more buttons on the gamepad for users to control everything. But we soon realized we were running out of buttons on the controller. To solve for this challenge, we decided to **automate** as many **user actions** as possible.

Human errors combined with lack of time

Having too many controls in TeleOp mode created another problem: the human operator had to remember all of them while still operating the robot efficiently in a competitive setting. Sure, with practice, users could get faster, but the risk was too much. Fortunately, **automating the process** of picking and dropping pixels, along with **auto-drive** capabilities, not only decreased human errors but also greatly boosted the robot's performance.

Machine Learning

The screenshot shows the FIRST Machine Learning Toolchain interface. A modal window titled "Upload Video File" is open, allowing a user to upload a video. The modal includes a "Choose File" button, a text input field for the description (containing "Team X Prop"), and an "Upload" button. The background interface shows the "FIRST TECH CHALLENGE" logo, navigation links for "Videos", "Datasets", and "Models", and buttons for "Upload Video", "Produce Dataset", and "Delete Videos". A notification at the top right states "Exporting 'My Movie' was successful." The footer includes "WITH SUPPORT FROM Google Cloud" and copyright information for 2021 FIRST.

The screenshot shows the FIRST Machine Learning Toolchain interface with a table of uploaded videos. The table has columns for "Date Uploaded", "Description", "Video Filename", "File Size", "Dimensions", "Duration", "Frames per Second", "Number of Frames" (subdivided into "In Video", "Extracted", "Labeled", and "Excluded"). A single row is visible, representing the video "My Movie.mp4" uploaded on 11/22/2023 at 12:17:24 PM with a description of "Team X Prop". The interface also shows the "FIRST TECH CHALLENGE" logo, navigation links, and buttons for "Upload Video", "Produce Dataset", and "Delete Videos".

Date Uploaded	Description	Video Filename	File Size	Dimensions	Duration	Frames per Second	Number of Frames			
							In Video	Extracted	Labeled	Excluded
11/22/2023, 12:17:24 PM	Team X Prop	My Movie.mp4	42,468,543					0	0	0

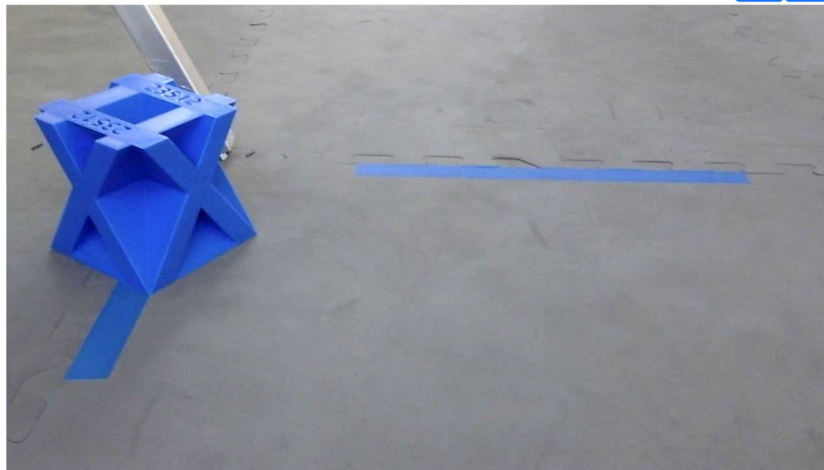


Machine Learning

Video: Team X Prop



838 Frames



X1	Y1	X2	Y2	Label	

Frame 228



Ignore this frame

Tracking with OpenCV™

Algorithm: MedianFlow

[What's this?](#)

Start Tracking



Could not track all objects. Paused.

Ignored frames: 0

Unlabeled frames: 611

Find Ignored Frames

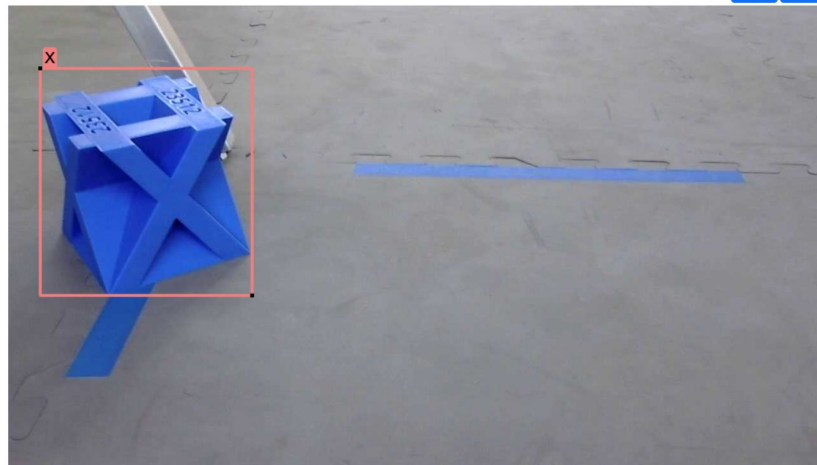
Find Unlabeled Frames



Video: Team X Prop



838 Frames



X1	Y1	X2	Y2	Label	
50	98	381	452	X	

Frame 201



Ignore this frame

Tracking with OpenCV™

Algorithm: MedianFlow

[What's this?](#)

Start Tracking



Ignored frames: 0

Unlabeled frames: 128

Find Ignored Frames

Find Unlabeled Frames



TEAM X # 23512

Machine Learning

The screenshot shows the 'FIRST Machine Learning Toolchain - v2.1.4' interface. A 'Produce Dataset' dialog box is open, displaying the following information:

- Percentage of Frames for Training: 90 %
- Percentage of Frames for Evaluation: 10 %
- Description: TEAM X PROP DATASET
- Progress: Video frames processed: 0 of 838

The background interface includes a 'Student Access' section, an 'Uploading Video' section, and a table of uploaded videos. The table has columns for Date Uploaded, Description, Video Filename, File Size, Dimensions, Duration, Second, In Video, Extracted, Labeled, and Excluded. One video is listed: 'My Movie.mp4' with a file size of 42,468,543 and dimensions of 1280 x 720.

The screenshot shows the 'FIRST Machine Learning Toolchain - v2.1.4' interface. A 'Start Training' dialog box is open, displaying the following information:

- Starting Model: SSD MobileNet v2 320x320
- Number of Training Steps: 4000
- Advanced: Maximum Training Time: 600 Minutes. Your team has 600 minutes of training time remaining.
- Description: TEAM X PROP MODEL
- Text: Your selected dataset contains 754 training images. With the selected model, 32 images will be processed during each step. This is called the batch size. It will take 24 steps to perform one full cycle through your training data. This is called an epoch. Training for 4000 steps will perform 169 epochs.
- Text: This training job will take approximately 67 minutes, but will be stopped if it runs longer than 600 minutes.

The background interface includes a 'Student Access' section, an 'Uploading Video' section, and a table of produced models. The table has columns for Date Produced, Description, and Evaluation. One model is listed: 'TEAM X PROP MODEL' with a date produced of 11/22/2023, 1:17:56 PM. The evaluation table shows a Total of 84, Negative of 10, and Labels of X.



Machine Learning

ftc-ml.firstinspires.org

FIRST Machine Learning Toolchain - v2.1.4

FIRST TECH CHALLENGE Resources Help/Feedback Hello, Rajeev FTC Team 23512

Student Access
Instructions for granting students access to the system can be found [here](#)

Uploading Video
Instructions for uploading video can be found over on [ftc-docs](#)

Videos Datasets Models

More Training Download Model Stop Training Delete Models Remaining training time: 0 minutes

<input type="checkbox"/>	Date Created	Description	Starting Model	Steps Requested	Job State	Steps Completed	Training Time
<input type="checkbox"/>	11/22/2023, 1:22:30 PM	TEAM X PROP MODEL	SSD MobileNet v2 320x320	4,000	RUNNING	0	1:49

Model: TEAM X PROP MODEL

Stop Training Refresh Interval:

Details Graphs Images

Date Created	11/22/2023, 1:22:30 PM	Training		Evaluation	
Original Model	SSD MobileNet v2 320x320	Total Frames	754	Total Frames	84
Added Datasets	TEAM X PROP DATASET	Negative Frames	118	Negative Frames	10
		Label Counts	Label Count X 636	Label Counts	Label Count X 74
		Steps Requested	4,000	Job State	RUNNING
		Job State	RUNNING		
		Steps Completed	1,100		
		Training Time	17:16		



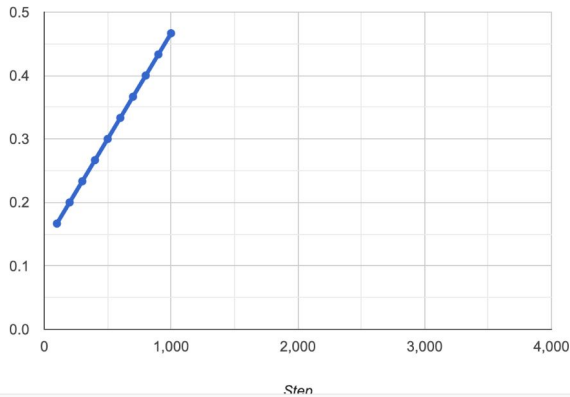
Machine Learning

Model: TEAM X PROP MODEL

⊗ Stop Training Refresh Interval:

Training Metrics

learning_rate



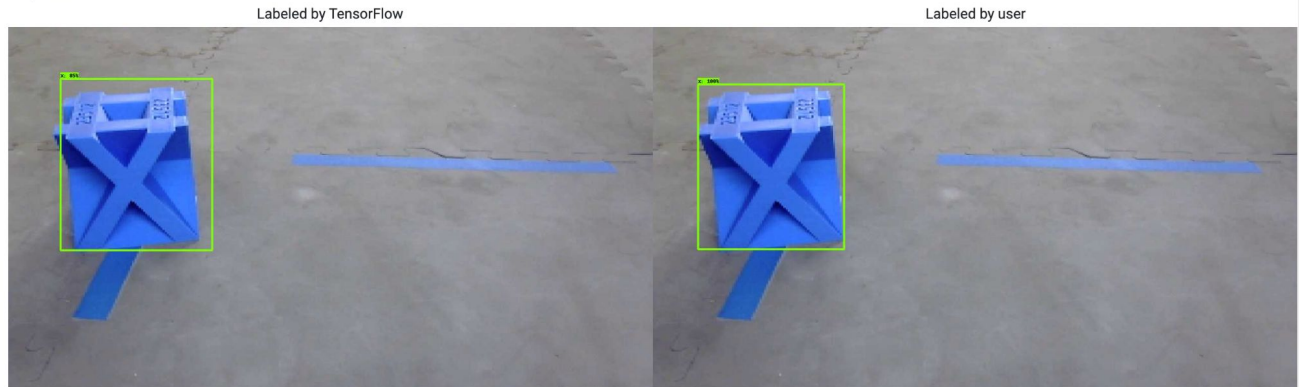
Model: TEAM X PROP MODEL

⊗ Stop Training Refresh Interval:

Page 4 of 9

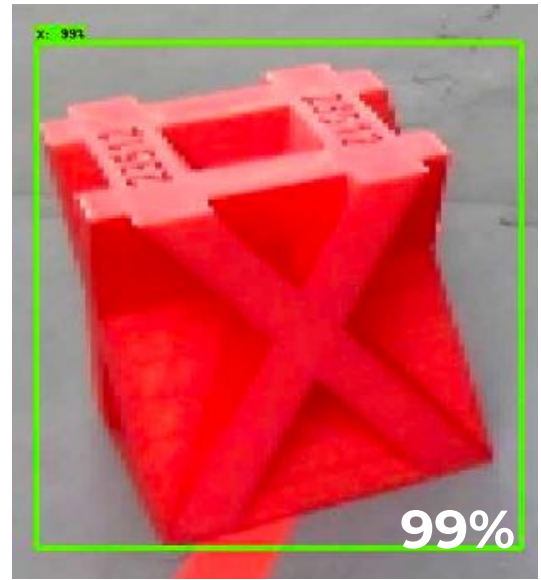
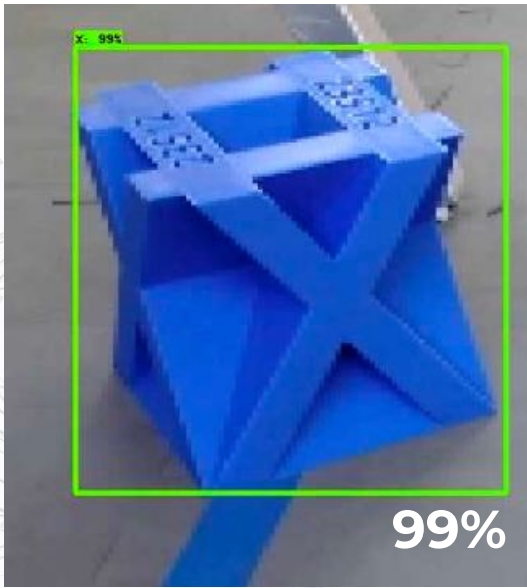
eval_side_by_side_30_0

Step: 800



TEAM X # 23512

Machine Learning



Model: TEAM X PROP MODEL

Refresh Inte

[Details](#) [Graphs](#) [Images](#)

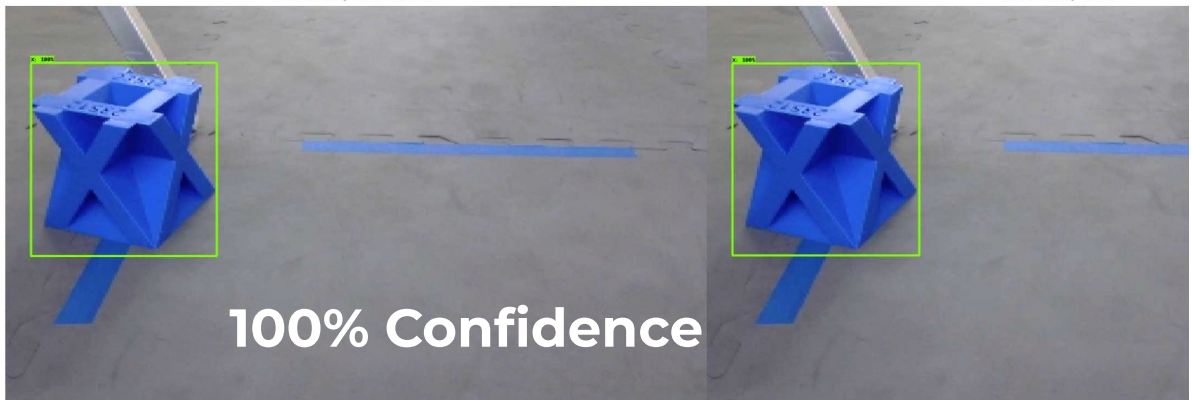
Page 1 of 9

eval_side_by_side_9_0

Step: 4,000



Labeled by TensorFlow

Labeled by user



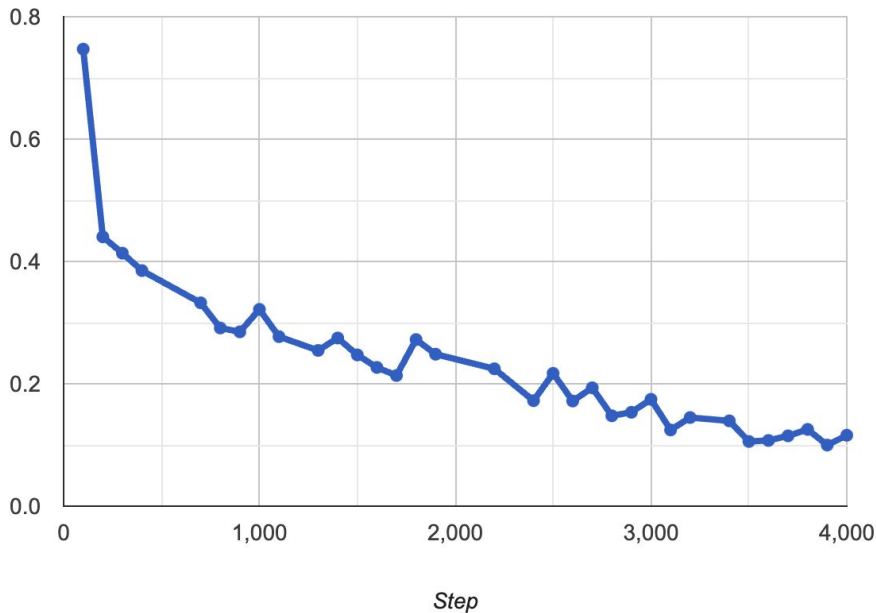
Machine Learning

Model: TEAM X PROP MODEL

⊗ Stop Training Refresh Interval:  

Details		Training		Evaluation	
Date Created	11/22/2023, 1:22:30 PM	Total Frames	754	Total Frames	84
Original Model	SSD MobileNet v2 320x320	Negative Frames	118	Negative Frames	10
Added Datasets	TEAM X PROP DATASET	Label Counts	Label Count X 636	Label Counts	Label Count X 74
		Steps Requested	4,000	Job State	STOPPING
		Job State	SUCCEEDED		
		Steps Completed	4,000		
		Training Time	56:37		

Loss/total_loss



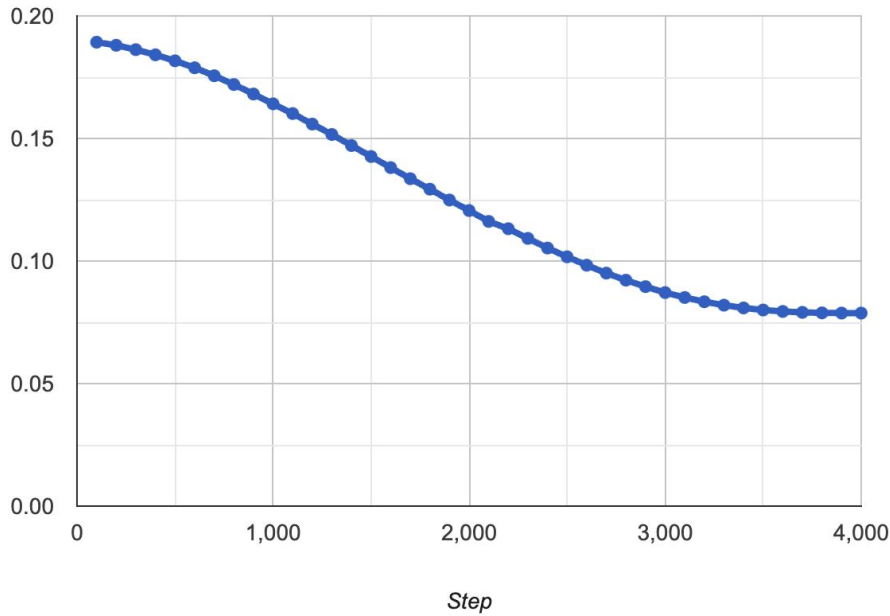
A low loss indicates that the model is performing well on the training data.

It suggests that the predictions are close to the actual values, and the model is learning the underlying patterns in the data.



Machine Learning - Evaluation Metrics

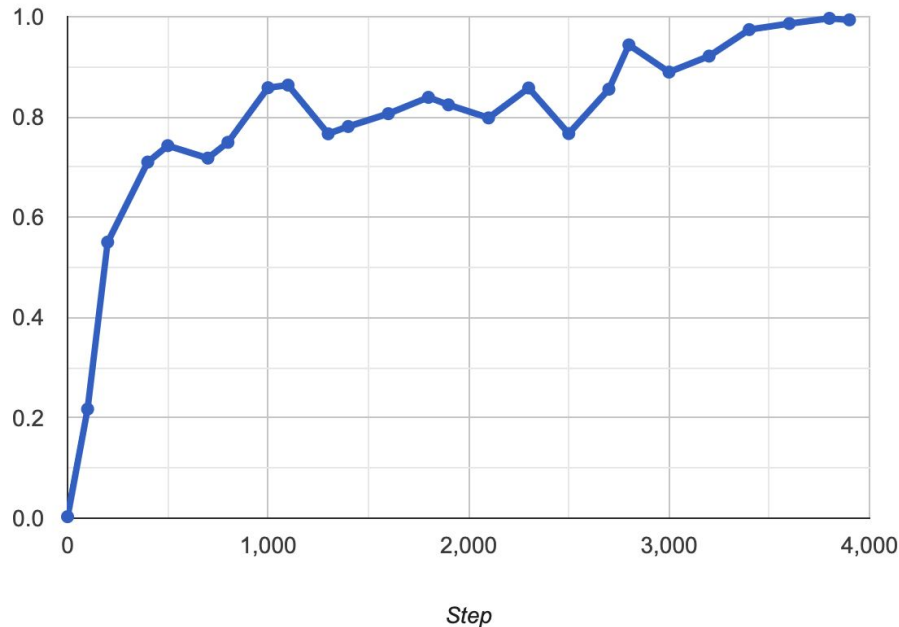
Loss/regularization_loss



Regularization is a technique used to prevent overfitting, where a model performs well on the training data but fails to generalize to new, unseen data.

Low numbers indicates that the primary loss term dominates, and the model is focusing more on fitting the training data.

DetectionBoxes_Precision/mAP



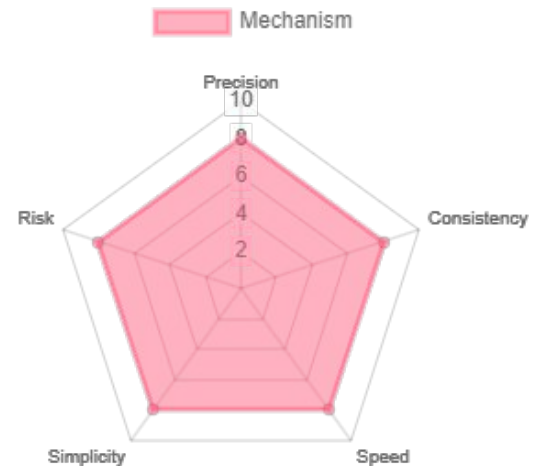
98%

mAP = Mean Average Precision



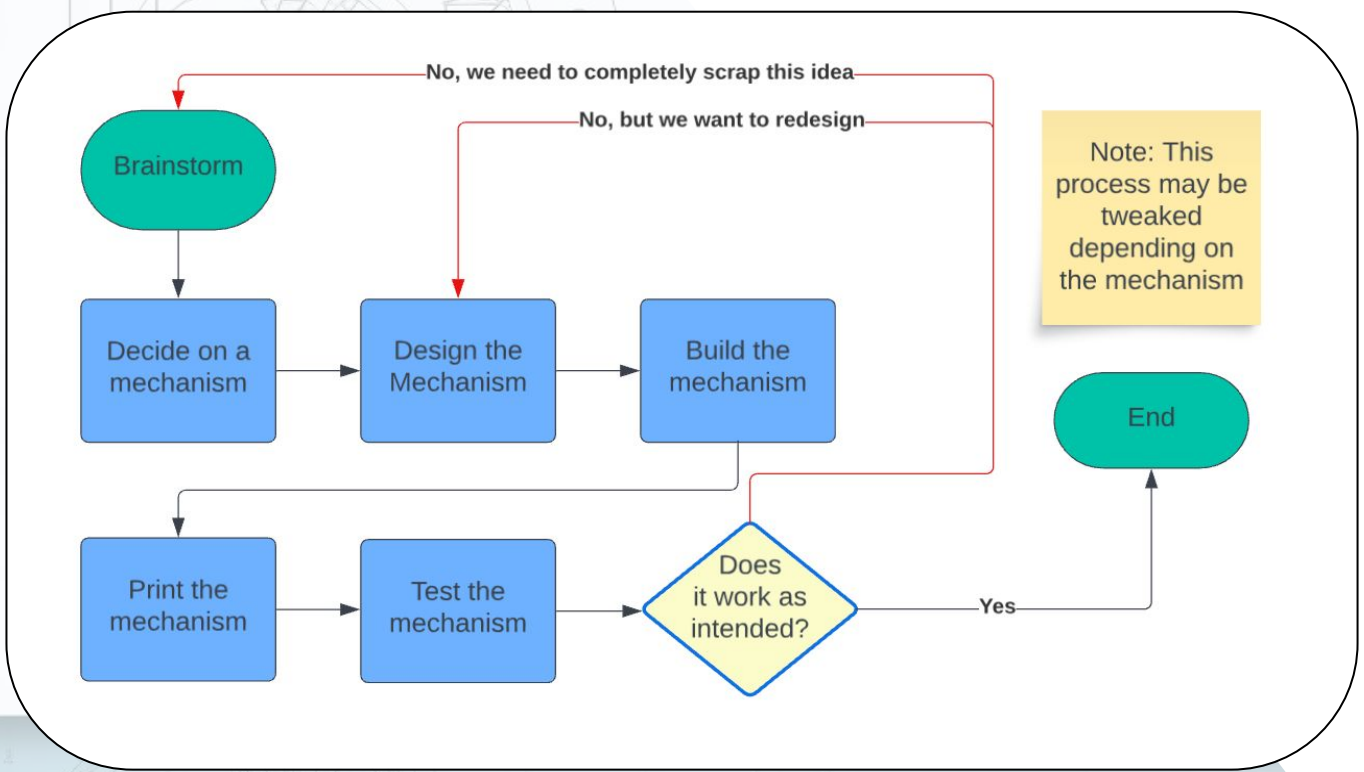
Testing and Validation

Testing and validation play a big role in the world of robotics, ensuring that systems meet their intended goals and perform well. In FTC, effective testing and validation procedures are fundamental to guaranteeing the success and dependability of robotic parts, whether they are designed as a drivetrain, claw, arm, or virtually anything else.



Our Process of Testing Goes as Follows:

To begin with, we generated ideas through brainstorming. Specifically, we explored various approaches to constructing the claw responsible for grabbing the pixel. Subsequently, we crafted the mechanism using Onshape, a CAD software widely used in FTC. Once the design was complete, we utilized our impressive 3D printer to bring it to life. Following the printing, we tested the mechanism with the software and iterated on it if necessary.



Finances

Expense	Paid
Team Registration with FIRST	\$ 295.00
Field Set (Perimeter and Soft Tiles)	\$ 1,295.00
FTC Starter Kit + Strafer + Tools	\$ 1,147.31
FIRST Control and Comm Set 5 & REV Control Hub Set	\$ 654.83
Motors, Batteries, Chargers from GoBilda	\$ 475.67
FIRST CenterStage 2023-2024 Season	\$ 564.92
Power and Surge	\$ 39.34
Dividers, Inserts, Latch	\$ 95.85
Rev Robotics Expansion Hub and wires	\$ 392.48
TPU 3D	\$ 21.92
Pulleys and Parts	\$ 191.38
Clamping Hub and Spacer	\$ 40.96
Bars, Bearings and more	\$ 276.30
2 ZOSKAY Servo Motors	\$ 63.92
2 Axon MINI Servo Motors	\$ 213.95
Dividers	\$ 81.54
Bearings Servo blocks, Gears	\$ 376.98
Polycarbonate Filament for Hooks and Side Plate	\$ 98.10
Tapes for Field Set	\$ 33.03
Channels, Gears, Clamps	\$ 647.12
Go Bilda order -- rods, etc	\$ 269.70
Rev Battery and Switch	\$ 50.54
More wires and 2 motors	\$ 120.82
Amazon printer filament for chasie	\$ 30.84
GoBilda Wires	\$ 25.60
Rev controllers and sensors	\$ 83.16
Another Logi Camera	\$ 21.50
Black Filament for Robot Chassis	\$ 49.59
GoBilda Motors (high torque)	\$ 137.95
Distance Sensors	\$ 61.17
goBILDA goBars	\$ 92.13
goBILDA 117 motors	\$ 82.83
goBILDA Order for Robot enhancements post 1st Q	\$ 618.28
Axon Micro+ (new Claw)	\$ 348.56



Outreach Activities

Outreach and sharing our wealth of knowledge with the community is ingrained in our team's DNA. To fulfill this mission, we have a dedicated team member committed to the cause. Our efforts extend beyond in-person events, as we actively utilize online platforms to educate, mentor, and host free Coding Camps, promoting awareness in STEAM, Robotics, and FIRST. We also openly share our CAD models, robot design iterations, and code with other FTC teams.

MakerFaire

Vallejo, CA

Joined Playing at Learning to help spread STEAM and FIRST awareness

Helping FLL Teams

Local

Help bootstrap new FLL teams, donate legos to local teams

(Currently Looking for FLL Teams)

Teaching CAD

Online,
Periodic

Open office hours, Discord Meetings

Open Source CAD Models

OnShape,
FTC Discord

X Launcher, X Claw, Decoy Pipe End

Host Scrimmages

Dublin, CA

Team X base

(Planned for Jan)

Coding Camp for Kids

Online

Python, HTML, CSS

(Planned for Spring)



FIRST @ Maker Faire

For the past four years, our involvement in outreach events as an FLL team has been substantial. However, our debut as an FTC team at the Maker Faire in Vallejo marked a significant milestone. We teamed up with **Playing at Learning** in educating **thousands of kids and parents** about the wonders of **STEAM, FIRST** and **Robotics**

Our exhibit was a hub of creativity, with kids designing their own unique lapel pins. The atmosphere was vibrant and engaging, and the joy on the children's faces was truly priceless. We also distributed our team pins, fostering a sense of community among the attendees.

The star of our showcase was undeniably our Robot, stealing the spotlight with its impressive *mecanum* wheels that allowed it to effortlessly strafe sideways. Spectators were captivated by its cool maneuvers, and we seized the opportunity to showcase internal workings of its arm, wrist, and claw.

The sheer number of kids expressing interest in robotics was astonishing!

In retrospect, our first outreach experience was nothing short of amazing. The dates, Sunday, October 22nd, is etched in our memories as a day filled with fun, learning, and the joy of sharing our passion for robotics with everyone at the Maker Faire.



OUTREACH EVENT

SUN 10/22/2023

Attendees

Arnav	Ronav	Samar	Yajat	Coach 1	Coach 2
✓	✓	✓	✓	✓	✓

Time Commenced: 8:00 AM
Time Adjourned: 6:00 PM
Location: Maker Faire in Vallejo

Team's Reflection

ARNAV // The Maker Faire was really fun and exciting for me. It opened up a new part of me that I have never seen before. The maker faire was a very interesting place to look at but at the same time, I was able to inform people about robotics and FIRST. We also had many things that people were enthusiastic about. One activity that we had was the table where you could make your own lapel pin for free. These kids were very creative and came up with many amazing designs. We also had the option for people to look at our robot and the way it worked. As I was showing off the robot, the most asked question was "How do these Mecanum wheels work?" Me and my teammates were explaining how they helped the robot move in a sideways direction and the people were fascinated. A couple of parents walked up to me and started watching as I drove it around and asked a couple of questions. I even directed them to a sign-up sheet that gives them more information about FIRST and the robotics involved for kids. My favorite part of this event was probably explaining robotics and helping kids with their pins. It really amazed me how many people were interested in robotics. This experience really helped me realize the fact that people might not care about robotics but when they see smaller things like this, it sparks innovation and curiosity in them.



OUTREACH EVENT

SUN 10/22/2023

RONAV // Going to the Maker Faire in Vallejo with my robotics team, "Team X," was a blast. There were four of us, and we shared a booth with "Robo Racers #16481" from West Dublin. We're based in East Dublin, so they're pretty close to us. Our half of the table was a hub of creativity. We stamped custom pins for kids, and they loved it. We also handed out team pins, making everyone feel like a part of our community.

The star of our show was our robot. We showed off its cool moves with Mecanum wheels that let it strafe sideways. People were wowed by this. We also explained how our robot's arm worked with its shoulder, wrist, and claw. It was like showing off a cool gadget to friends.

But that's not all. The Maker Faire was a blast of innovation. We saw a humongous robot that looked like it came straight out of Real Steel. There were robots from Star Wars, mini Transformers, BattleBots, lego, drone racing, humanoids, etc. I even got jumpscared by a skeleton that jumped out of a trash can when exiting the restroom. One couple even built a recreation of Boston Dynamics "Spot", a robot dog.

Ultimately, it was an amazing experience, especially for our first outreach. I'll never forget that day, Sunday, October 22nd. It was all about fun, learning, and sharing our love for robotics with everyone there. From 8:00 to 6:00, all ten hours were worth it.

SAMAR // The Maker Faire was an unforgettable experience. From learning about new robots to getting to teach others about FIRST, it was everlasting fun. We had multiple activities that kids could interact in.

One of our activities was our "Make Your Own Pin" station. Kids got to express their creativity and show others. I noticed a very visible pattern. The younger the kids got, the more open they would be to using more colors. Maybe that shows that they have more creative freedom. These small realizations were cool things I got to notice.



The most common question I got about our robot was about our Mecanum wheels. I explained how the wheels granted the advantage to strafe easily when they move in an inward motion. It was really nice to see people join along as I explained and contributed to the conversation. One small group of 5 year olds came in to talk about how their RC cars went “twisty twisty” and worked kind of like ours. Slowly, it shifted to a conversation about FIRST and its many benefits.

It was amazing to see how many kids were interested in robotics. I got the chance to talk to the parents first hand and suggest and point them where to sign up. It goes to show those little things like being able to talk and show the audience great things is what made me really happy. That led to the realization that even the smaller things are helpful.

YAJAT // It was a dark and rainy morning, but my mood was as bright and sunny as could be. I was going to Maker Faire! It was an opportunity of potential greatness for me, and looking back, I could not ask for anything better. Among the traffic and business of Makerfaire, we managed to claim a booth and set up our custom pin-maker and a little mat to drive our robot around. After what felt like an eternity, visitors finally started showing up, and a surprising amount of visitors came to our booth. Our custom pin-maker was an instant success, and it even brought people to be interested in our robot, robotics, and FIRST. That was actually one of my favorite parts. Getting people interested in our robot while I drove it around. I still remember how excited I was that other people actually cared about our team and FIRST. Talking to them, explaining the parts of the robot, and answering all their questions was easily the best part of Maker Faire. We took shifts managing our booth, and when it wasn't my shift, I explored the other booths and sections or enjoyed the food trucks. Going to Maker Faire was like a gift, and definitely one that I enjoyed and squeezed every last bit of joy from. When it was time to go back, I immediately missed answering peoples questions and explaining the robot.

Outreach: Makerfaire



Outreach: FLL Team





X-Bot vs SPOT



TEAM X # 23512

Mentors & Industry Experts

Mentors & industry experts are like helpful guides. Their experiential knowledge becomes a guiding light and helps our team avoid problems and do better. It's like having a map in a confusing place—mentors help us in many ways.

We're getting support from two experienced FTC teams who are helping us avoid common problems. We were told that design is the most important element of robot performance. We are lucky to have one of the world's best designers (Shivang) mentoring our team.

SHIVANG PATWA, Chief of Design, Cowbell

Design Mentor

Shivang brings years of experience in creating meaningful experiences in both digital and physical spaces. He has a unique blend of creative intelligence and a pragmatic attitude.

FTC Team **Mastermindz #18466**, Dublin

Team Mentor

Since this is our 1st year in FTC, the Mastermindz team has provided invaluable guidance, particularly in mentoring us through the fundamentals, from basic building blocks to interesting design ideas. Special thanks to Yash from Mastermindz.

FTC Team **Voyager 6+ #12869**, Dublin

Team Mentor

Team Voyager 6+ played a crucial role in our journey by not only mentoring us but also lending a helping hand with spare parts, like sensors, when needed. Special thanks to Aarsh from Voyager 6+.

Sponsorships & Grants



Control Award Submission Form

Team # 23512	Team Name: TEAM X
---------------------	--------------------------

Summary:

Despite being a rookie team, we prioritized **computer-aided-design** (CAD) over free-building our bot from the outset. After refining our design to near perfection, we spent significant time building and enhancing our **machine-learning** (ML) model. We achieved an accuracy rate of over **98%** in detecting our custom team prop, surpassing the default ML model provided by FTC, which has only 90% efficacy to detect the white pixel.

Realizing the importance of precision in robot positioning and movement, we made the decision to integrate **odometry** and **encoders** into our X-Bot. This enhancement significantly improved our robot's drive and positioning capabilities. By utilizing MeepMeep, we embraced a **test-driven development** (TDD) methodology for building and testing trajectory sequences before coding them in our Autonomous mode. This approach not only resulted in a bot that performs exceptionally well but also boasts an impressive aesthetic.

Since this is our first year in FTC, none of our progress would have been possible without the incredible support of FTC community (including FTC Discord), as well as our mentor teams, and coaches.

Autonomous Objectives

- Using the custom **TensorFlow Object Detection** (TFOD) ML model, detects the TeamX Prop position on the Spikemark and precisely navigate (using **odometry**) to place the purple pixel.
- Using **April Tags**, navigate to the corresponding backdrop location (matching April-Tag ID to the spike mark) and drop the yellow pixel.
- Self-drive to collect two white pixels from the central stack and return them to the backdrop.
- Depending on the alliance's preference, park either to the right or the left of the backdrop.
- Maintain full control of all actions all the time.

Sensors used

- 3-dead wheel **odometry** to keep track of robot position (x, y, heading) at all times.
- Instead of relying on Gyro in the inefficient Inertial Measurement Unit (IMU) in the control hub, we use two-parallel dead wheels and **encoders** to calculate precise heading.
- **April Tags** (Computer Vision) are used for Auto Drive in both autonomous and driver controlled modes
- **Tensorflow** Object Detection (TFOD) is used to identify custom Team Prop
- Four controlled **servos** to enable precise movement of Wrist and Fingers in the **X-Claw**

Key algorithms

- **ML model** (built using TFOD) to recognize custom TeamX Prop placed on Stripe Marks
- **Statistical model** to convert TFOD output (position and size) to Spike Mark location (Left, Center or Right) and corresponding April Tag IDs
- **Linear regression model** to calculate Wrist position based on Arm position.
- **State Machine (using Game Modes)** to implement complex logic and achieve clear, maintainable, and predictable robot behavior.
- **Proportional, Integral, and Derivative** (PID) controller to adjust the gains used in trajectory following. This enables closed-loop feedback control to ensure accurate path following.

Control Award Submission Form

Team # 23512	Team Name: TEAM X
---------------------	--------------------------

Driver controlled enhancements:

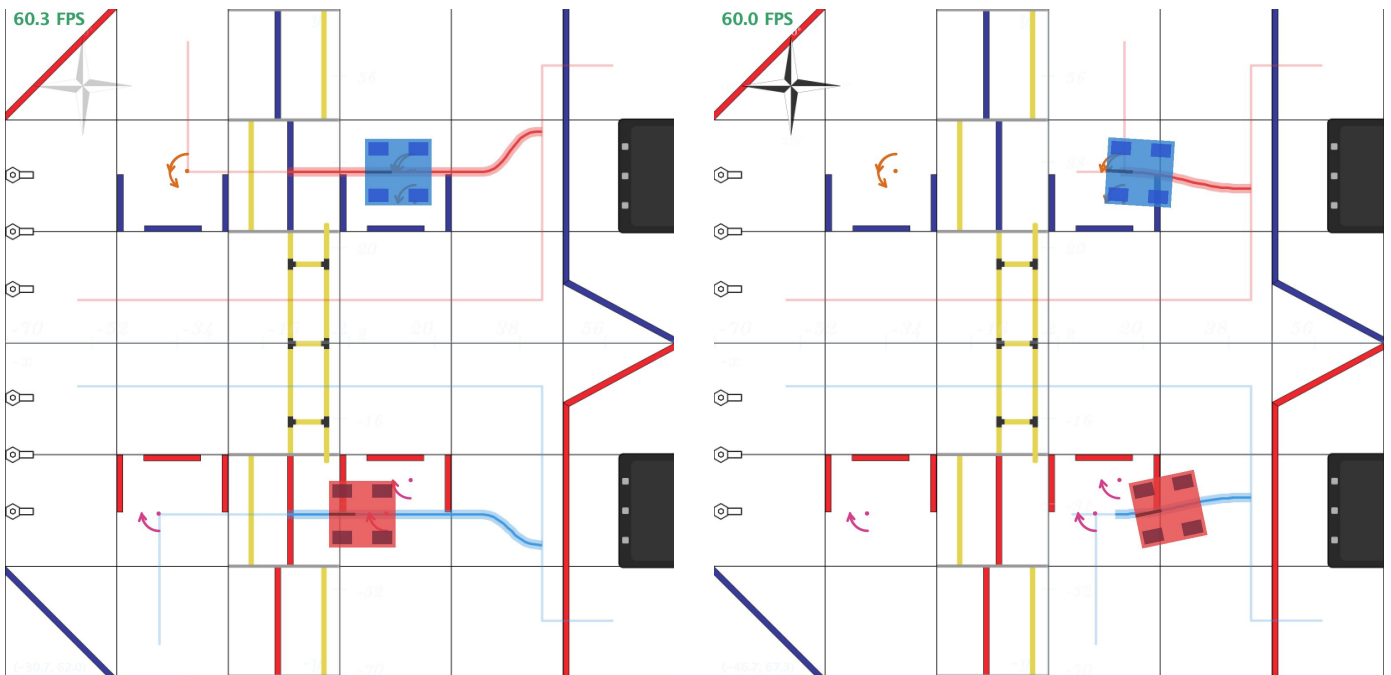
- **Autonomous driving** even in TeleOp mode (using **odometry**) to help eliminate human errors, and improve Robot performance. Example: Lock robot's heading when the human player is driving in a straight line or strafing, thereby eliminating possible shifts.
- Use **State-Machine** and **Game Modes** based on various conditions to automatically move **Arm, Wrist, Claws** or get in-and-out of april-tag based navigation

Engineering portfolio references

- Refer to *Machine Learning and Modeling (Page 10)* to learn about **Tensorflow** Object Detection (TFOD) and how it is used to recognize custom Team Prop
- Refer to **Odometry** (Page 11) to learn about how Robot keeps track of its position on the field at all times.
- Refer to *Machine Learning and Modeling (Page 10)* to learn more about how **Linear regression** is used to calculate Wrist position (Y) based on Arm Position (X)
- Refer to *Software Design (Page 9)* to learn more about how **State Machine** is used
- Refer to **X-Claw Design** to see how X-bot uses 4 micro servos to move **wrist** and **fingers**

Autonomous program diagram(s)

- Diagrams below show four examples of autonomous routes. First picture shows X-bot placed farther away from the backboard whereas the second picture shows X-bot closer to the backboard



- XBot Blue Alliance Left Pixel
- XBot Red Alliance Right Pixel

- XBot Blue Alliance Right Pixel
- XBot Red Alliance Left Pixel

Control Award Submission Form

Team # 23512	Team Name: TEAM X
---------------------	--------------------------

Autonomous objectives

- *Autonomous in **AutoOp***
 - Identify the Spike Mark with our TeamX Prop using custom TensorFlow Object Detection (TFOD). Our custom ML model efficacy is over 98%
 - Place purple pixel on identified spike mark (Auto Drive using Encoders and Odometry – not using IMU)
 - Drop the yellow pixel on the corresponding location on the back board (Auto Drive using AprilTags)
 - Pick 2 white pixels and drop on backboard (Odometry allows precise navigation at all times)
 - Park in the parking area (using Odometry)
- *Autonomous in **TeleOp***
 - Change Game Modes automatically based on game conditions
 - Turn Auto Drive on/off using April Tags Detection

Sensors used

- 3-wheel odometry to keep track of robot position (x, y, heading) at all times.
- April Tags (Camera Vision) are also used for Auto Drive
- Tensorflow Object Detection (TFOD) is used to identify custom Team Prop

Key algorithms

These algorithms make our Robot unique:

- **Machine learning algorithm** (built using TFOD) is used to recognize custom TeamX Prop placed on Stripe Marks
- **Statistical algorithm** is used to convert TFOD output (position and size) into identifying the Spike Mark location (Left, Center or Right)
- **Linear regression** model to calculate Wrist position based on Arm position. We decided to use a State **Machine** (using **Game Modes**) for designing and implementing complex logic and achieve clear, maintainable, and predictable robot behavior.

FTC Team **Glitch** (#23836)

We have been privileged to support Team Glitch, the Lane Middle School 7th-grade rookie team, consisting of fifteen members. We've had the opportunity to assist them in **organizing their tasks, addressing code issues, and testing their mechanisms.**

We also provided them advice on their **drone launcher** and shared our **CAD designs** with them, which we later open-sourced. We also helped them with their league meets, where we offered guidance in **troubleshooting robot issues.**

It's been a rewarding experience to mentor Team Glitch, and we're grateful for the chance to contribute to their growth and development in robotics.



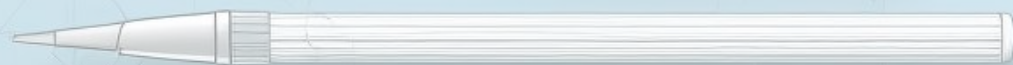
FTC Team **Rapid Fire** (#25442)

We are mentoring Team Rapid Fire, a new FTC team with seven members, for the 2024-25 season. We're helping them with challenges like **scheduling and communication** in meetings, establishing a basic understanding of **FTC rules**, and assigning roles within the team.

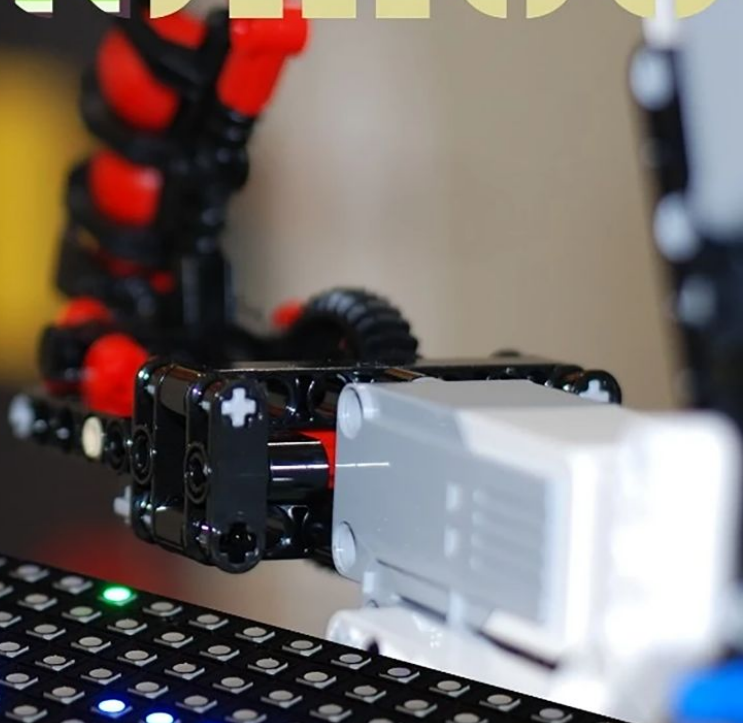
We're also helping them with a communication platform, providing detailed explanations of FTC basics and competition rules, and outlining various team roles. Hopefully, our guidance is helpful to Rapid Fire as they navigate FTC's complexities and prepare for a successful season.



TEAM X # 23512



Robot Challenge



Picnic Area

EMERALD GLEN PARK, DUBLIN



organized by

TEAM



Flappy Bird by