Design Brief

Basic info

- Project Name: *FTC Freight Frenzy*
- Team: **Cryptic #20123**
- Organizer: *FIRST Tech Challenge*
- Deadline: **Jan 29th**

Defining Problem

Create a robot capable of delivering freight with accuracy, efficiency, and reliability into the designated locations

Background

Transportation has been evolving rapidly ever since the beginning of civilization. From horses to carts all the way to modern planes and rockets, the need to efficiently transport goods from one location to another rapidly increases. This year's FTC game Freight Frenzy ***inspires*** future engineers into tackling the problem of transportation.

Targets & Constraints

Targets:

- 7+ freights into Alliance SH
- Tilt Shared SH towards our side
- Traverse freely around the barriers
- High structural integrity
- Modular design for flexibility & low maintenance
- Scoring in high range **consistently**
- Ensure all our members understand the tasks and bring their perspectives to the table

Constraints:

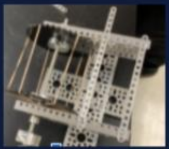
- Time limit - 30 seconds
- Obstacles on the field
- Physical robot constraints
- Rules and penalties in Game Manual 1&2
- Project period: 3 months
- Budget: $3,500
- Rookie members need to be trained with the appropriate skills they need

Timeline

9/2021

Team Cryptic is formed as a rookie FTC team and programming training sessions were held.

10/2021


Preliminary designing process begins and prototypes like the intake were tested.

11/2021

Design for custom side plates is finished and ordered. The plates have special features tailored for our robot

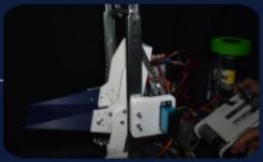
12/2021

The 6-wheel drivetrain is assembled and features a unique bevel gear chain system.

1/2021

Intake is mounted unto the robot and tele-op code is finished

1/2021


After several iterations, the final outtake system is mounted.

Future Plans

We will continue to improve our design and implement new ideas that we've learned throughout the season

Chassis

Brainstorming

While discussing through a series of in-person and virtual meetings, our target **criteria** for the chassis include:

- Traction: Have enough grip and **stability** to go over the pipe boundaries
- Power: Produce high amounts of **torque** to contain and transport freight efficiently
- Compact: Has open spaces to fit other components; will **not interfere** with other mechanisms

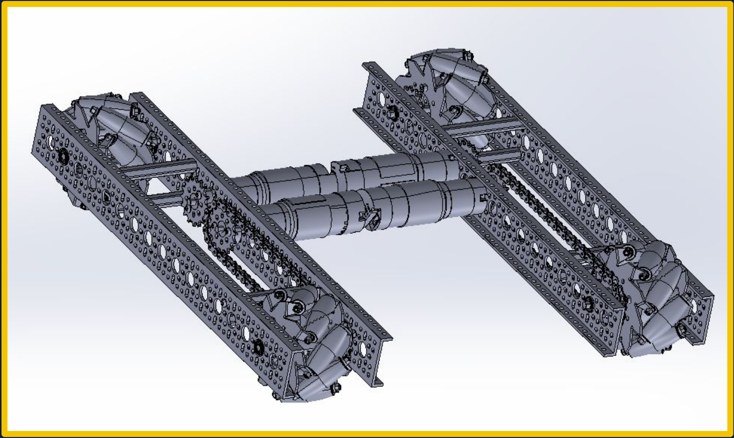
Preliminary Design

4-wheel Drivetrain
A simple popular design that consists of four mecanum wheels with a motor chained to each wheel. Some advantages of using this design are:

+ Uses less parts
+ Ability to strafe
+ Simple and straightforward

However:

- Can't pass through trench
- No space in middle



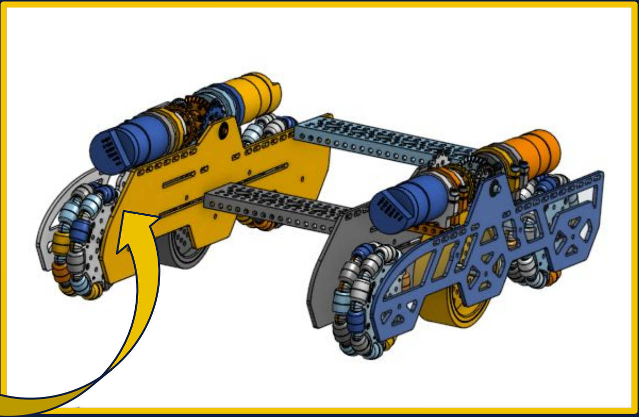
Final Design

6-wheel Drivetrain
A unique drivetrain with 2 rhino and 4 omni wheels all connected using a chain. Some advantages of this design are:

+ Large free space in center for other mechanisms
+ Can pass through barriers

However:

- Unable to strafe



Special Features of the design:

- Innovation miter gear/chain design for equal power throughout the three motors.
- Custom side plates to easily fit and assemble it
- Accessible REV Hub location out of the way of other mechanisms

*Because our strategy involves driving over the barriers and having mechanisms between our chassis, **this solution is the best choice** even if it is unable to strafe.*

Intake

Brainstorming

With mainly in-person meetings used to discuss the intake mechanism, we decided on some **criteria** that it should adhere to:

- Efficient: Should be able to intake one freight in under **two** seconds
- Compact: Take up **minimal** space for other components
- Reliable: Act as a **failsafe** to score points if the outtake malfunctions

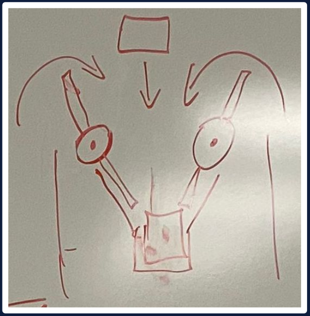
Preliminary Designs

Idea 1: Surgical Tube Funnel
Using two motors on either side with surgical tubes to **"funnel"** any game piece into the desired location. Some advantages of using this design are:

+ Intake any game piece

However:

- Not reliable

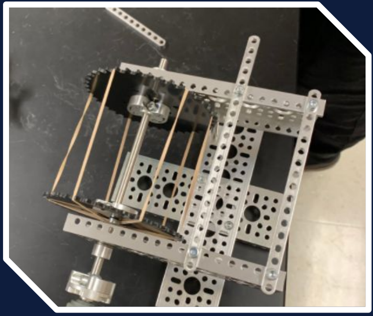


Idea 2: Rubber Band Roller
A unique drivetrain with 2 rhino and 4 omni wheels all connected using a chain. Some advantages of this design are:

+ Good traction

However:

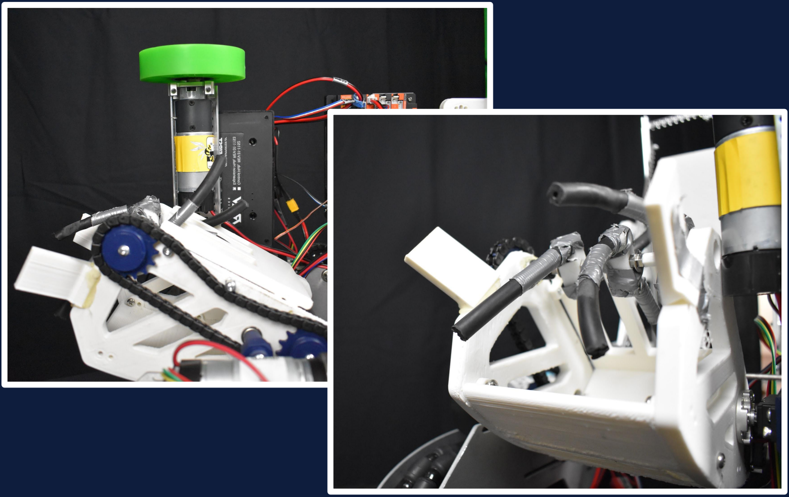
- Bands could snap
- No range



Problems Encountered

At first, our intake count not pick up anything at all. We tested **many** trials like increasing the power and changing flapper materials to **TPU** or **silicone**, but they did not help significantly. However, we realized angling the intake higher helps the intake **control** the freight better. This shows that the power of the intake does not affect it much, but instead, the **angle** is what really matters, because exerting power in the wrong direction will never pick the cube up.

Final Design



Surgical Tube Sweeper
We used some of the principles of **idea 2**, but are using **surgical tubes** instead of rubber bands and it has a compartment that can tilt up or flip around to transfer the game piece into the deposit system. Some **advantages** of this design are:

+ High range from intake to outtake positions
+ Able to intake freight in **0.23** seconds
+ Simple to build and integrate with other mechanisms
+ Could become outtake
+ Reliable and consistent
+ One main piece, not too many separated parts

Outtake

Brainstorming

While discussing, we were inspired by past FIRST competition shooters and created some target **criteria**:

- Quick: Can deposit game piece into shipping hub in less than **three** seconds
- Accurate: Smallest chance of **error** when scoring points
- Compact: Doesn't **interfere** with other mechanisms

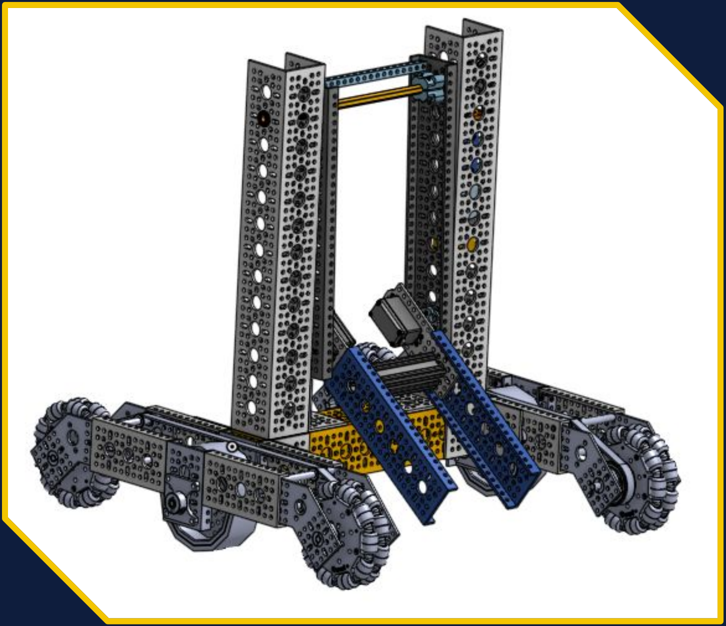
Preliminary Designs

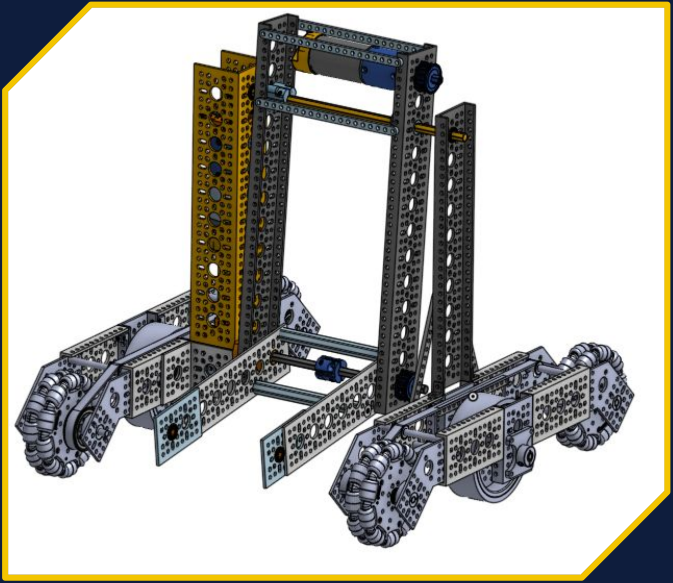
Idea 1: Rotating Arm
To test this design before ordering parts, we built a simple prototype using multiple U-channels and axles to demonstrate how the deposit would function and fix any complications that arose.. Some advantages of using this design are:

- + Intake on 1 side of robot, outtake on the other

However:

- - A lot of moving parts which leads to high chance of error
- - Too much strain on servos from weight of mechanism
- - Easily rock robot with high center of mass





Idea 2: Rotating Arm (Updated)
We wanted to develop the previous idea further and made many improvements. Some advantages of this design are:

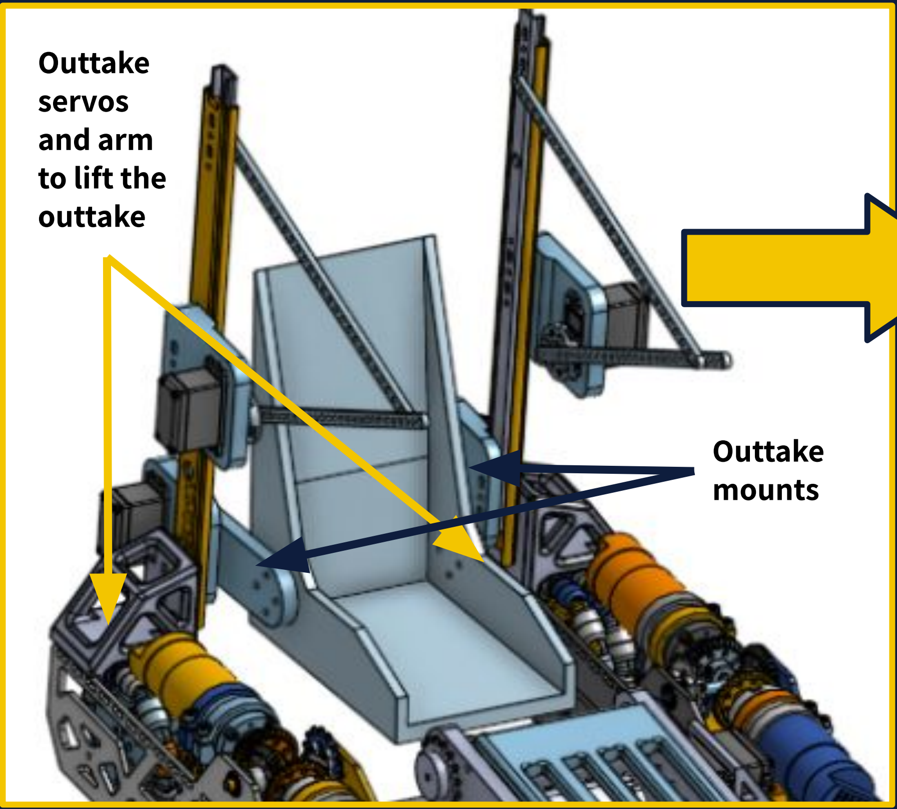
- + Move game piece vertically and horizontally
- + Easily move game piece to different heights

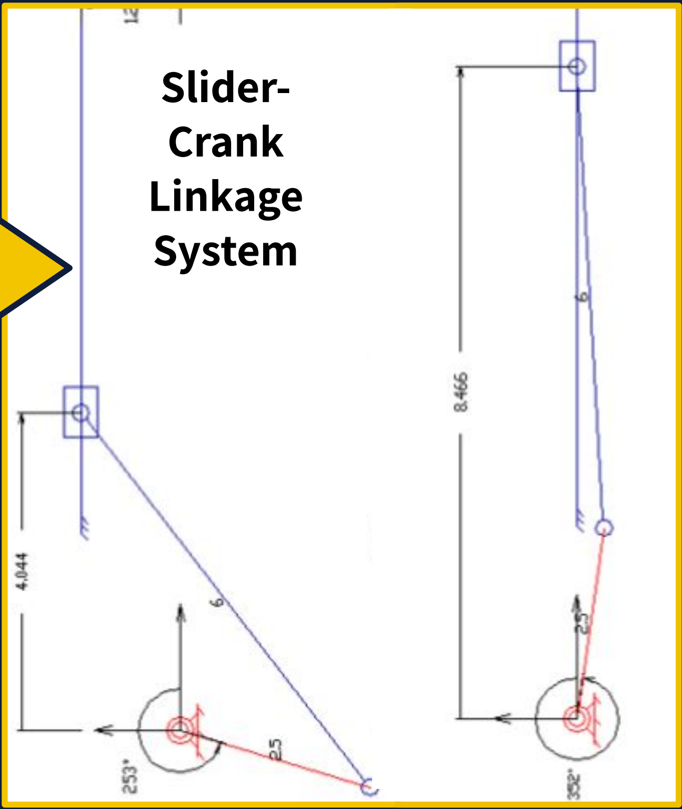
However:

- - A lot of moving parts which leads to high chance of error
- - High center of mass
- - Too much strain on servos from weight of mechanism
- - Difficult to wire the motor on the top of mechanism

Outtake

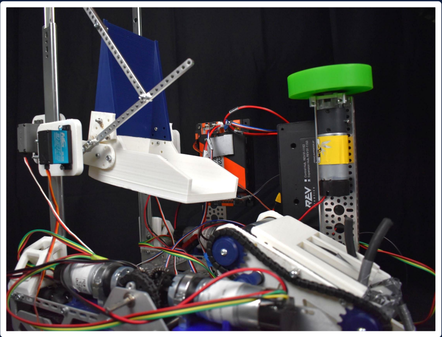
Final Design

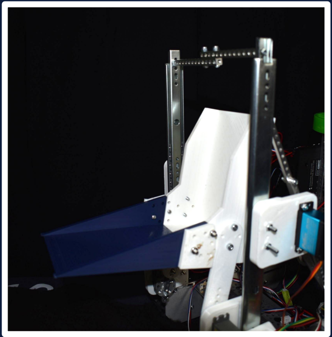


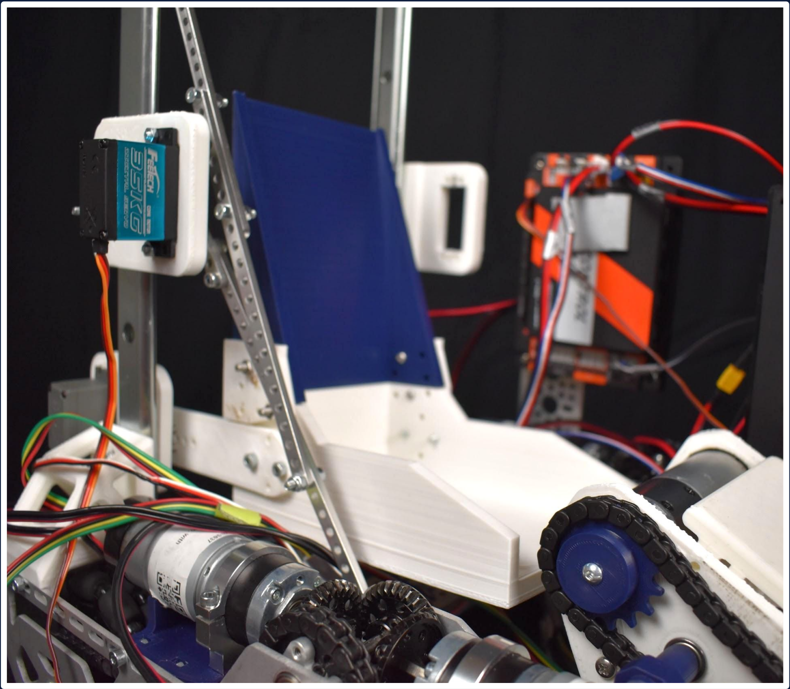


We modified the design of the deposit system because:

- *We wanted to be able to reach the 14 in mark to dump elements into the shipping hub*
- *It would be more efficient and faster to have two separate mechanisms work at the same time*







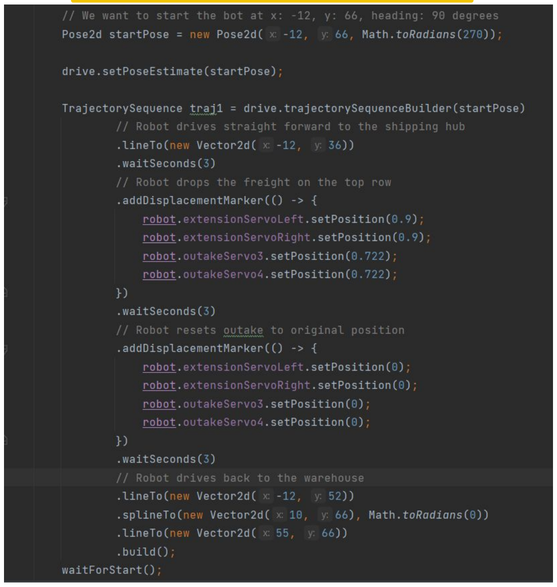
Because of the problems with the previous design, we are now using a design with slides and a storage container that will rotate 90 degrees to deposit the game piece. Some **benefits** of using this design are:

+ Much more **compact**
+ Fluid motion with little to **no strain** on servos
+ Move game piece to **different** horizontal heigh
+ Able to **accurately** position

Autonomous

Our Autonomous Paths

Path One Code



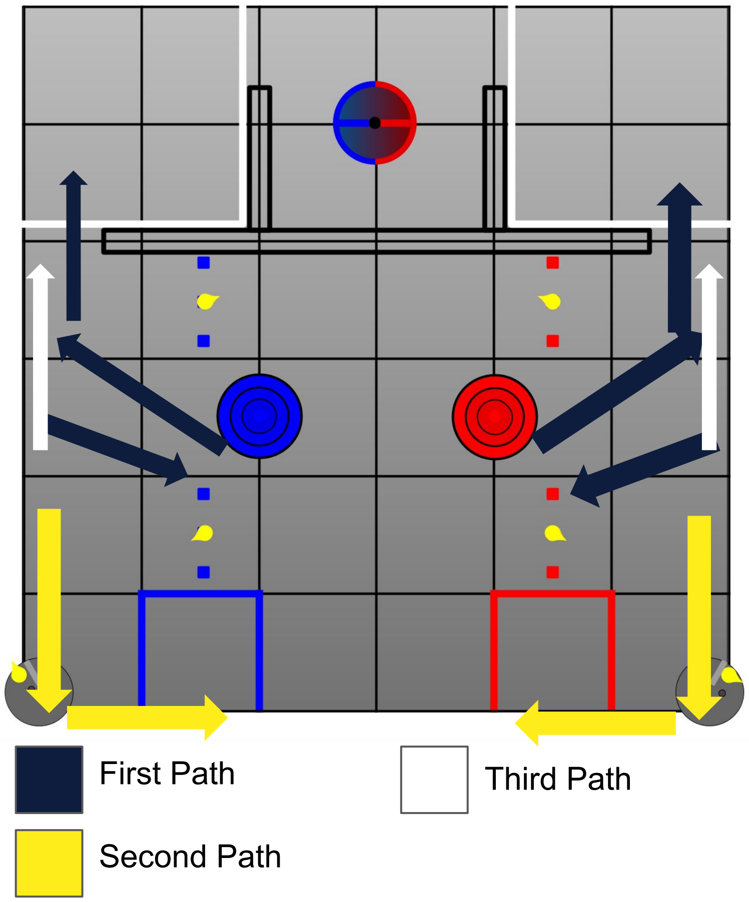
```
// We want to start the bot at x: -12, y: 66, heading: 90 degrees
Pose2d startPose = new Pose2d( x: -12,  y: 66, Math.toRadians(270));

drive.setPoseEstimate(startPose);

TrajectorySequence traj1 = drive.trajectorySequenceBuilder(startPose)
        // Robot drives straight forward to the shipping hub
        .lineTo(new Vector2d( x: -12,  y: 36))
        .waitSeconds(3)
        // Robot drops the freight on the top row
        .addDisplacementMarker(() -> {
            robot.extensionServoLeft.setPosition(0.9);
            robot.extensionServoRight.setPosition(0.9);
            robot.outakeServo3.setPosition(0.722);
            robot.outakeServo4.setPosition(0.722);
        })
        .waitSeconds(3)
        // Robot resets outake to original position
        .addDisplacementMarker(() -> {
            robot.extensionServoLeft.setPosition(0);
            robot.extensionServoRight.setPosition(0);
            robot.outakeServo3.setPosition(0);
            robot.outakeServo4.setPosition(0);
        })
        .waitSeconds(3)
        // Robot drives back to the warehouse
        .lineTo(new Vector2d( x: -12,  y: 52))
        .splineTo(new Vector2d( x: 10,  y: 66), Math.toRadians(0))
        .lineTo(new Vector2d( x: 55,  y: 66))
        .build();
waitForStart();
```

We programmed **three** main autonomous paths. These paths utilize **displacement**, line to, **trajectories**, and **waitsecond methods**. Our first path goes to the shipping hub which **drops the freight off** on the top level. The path also **returns** to the shipping container to maximize points in this autonomous path. Our second path **goes straight** to the duck wheel, **spins** the duck wheel, and then **parks** in the alliance storage unit

Our last path drives directly into the warehouse, which guarantees us some points while risking **no penalties**. It was important for us to program **multiple paths**. If one of our paths interfered with an alliances shipping hub, we could use a **separate path**. This allows us to **maximize** our potential points in the competitions.


First Path
Third Path
Second Path