



Technical Binder



Droid Rage 3035

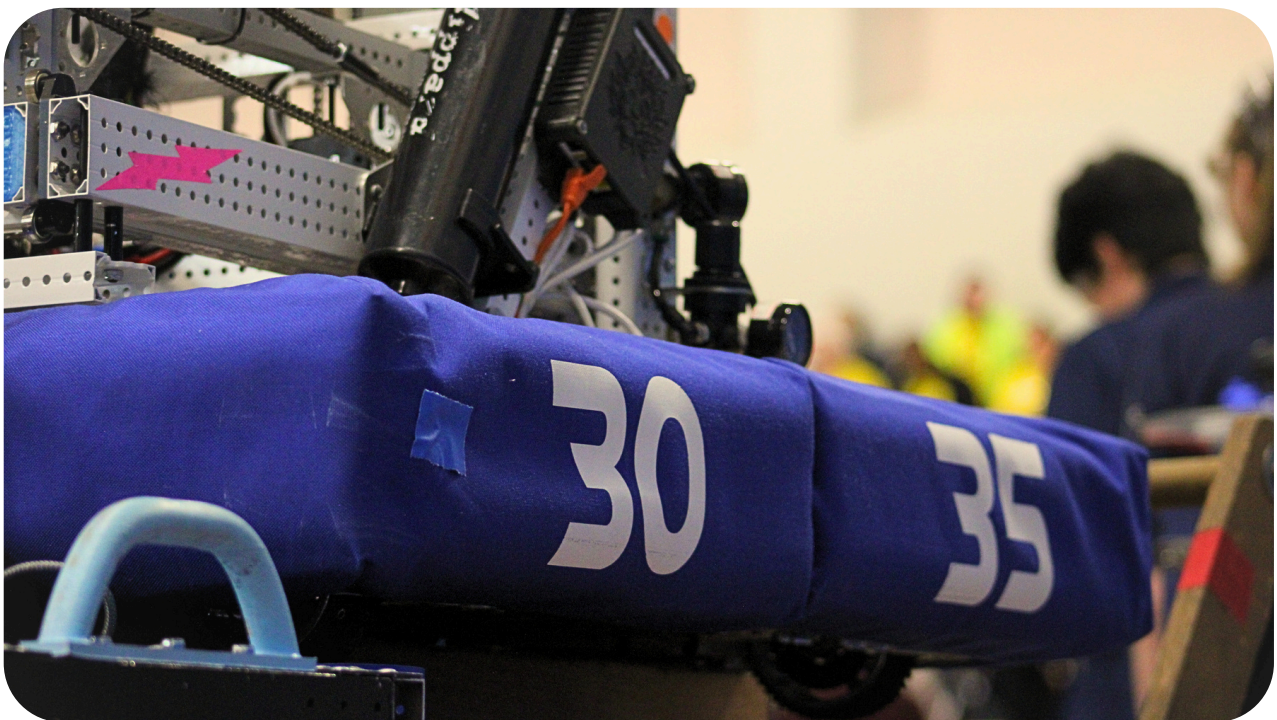
FRC 2023 | Charged Up

Overview

Our team's efforts began well before kickoff. Members organized the team structure, devised plans to overcome obstacles, taught each other new skills, and worked hard to make this Droid Rage's best season.

With the unveiling of the 2022-2023 game, our team began the design process including **robot design, prototyping, assembly, and deployment.** Throughout the season, our team has tirelessly pursued achieving the optimal design through various iterations until ultimately arriving at our current design.

This technical binder comprehensively details our game strategy, design decisions, and our 2022-23 robot: **EMERGENCE**



The robot name **EMERGENCE** is important to our team as our goal for this year was to revitalize Droid Rage 3035 like a Phoenix rising from the ashes with new energy to fuel this year's competition. The name also represents our strong work ethic. Failure is not the end but rather an opportunity to reinvent ourselves and become even better than before.

Table of Contents

Strategy.....	1
1.1 Game Analysis	1
1.2 Planning	1
1.3 Main Goal	1
1.4 Alliance Selection	2
1.5 Playoffs	3
Mechanical.....	2
2.1 Drive Train	4-5
2.2 Elevator	6-7
2.3 Prototype	8
2.4 Intake	9
2.5 Prototype	10
2.6 Custom Parts and CNC	11
Software.....	3
3.1 Mechanism Control	12-13
3.2 Safety and Driver Enhancements	14
3.3 Element Tracking	15



Strategy



Game Analysis

Planning

This year, we took an innovative approach to our team's strategy. We broke up members into small teams to brainstorm fresh and unique ideas.



Afterward, we came together to share our findings while reviewing the game manual page by page. Our goal was to design a robot that could navigate the challenges posed by 15 years of game and emerge victorious all while following the key principles of FIRST.

Main Goal

Our main goal in qualification matches is to **obtain as many ranking points as possible**, which will give us the flexibility to choose the type of alliance we want for playoffs.

To achieve this goal, we focus on the following:

- **Earn ranking points** - Specific tasks such as creating Links or earning the Sustainability Bonus can significantly improve our team's ranking position, so they are a top priority.
- **Score game pieces** - by placing game pieces on Nodes in the top and middle rows of the grid, we utilize the grid to create Links between adjacent Nodes in the same row. Getting 4-5 adjacent links earns us 1 Ranking Point.
- **Collaborate with our Alliance** - we work with our alliance partners to develop a cohesive strategy based on our team's robot abilities during the autonomous and driver controlled periods to win our matches. This earns us 2 Ranking Points.
- **Activation Bonus** - before a match, we confer with our alliance and depending on what autonomous each alliance has, EMERGENCE or another robot can run a Charging Station Auto-Balance Auto. This allows our alliance to earn 1 Ranking Point with only 2 robots balancing in End Game.

Game Analysis

Alliance Selection

Preparing for the event that our team becomes an Alliance captain, we formulated a plan to secure a strong and compatible alliance.

Scout other teams: Observe the matches of other teams and gather information about their strengths, weaknesses, and capabilities.

Analyze data: Using scouting information, we analyze average score, amount of nodes, and driver experience.



Evaluate compatibility: Determine which teams are compatible with our's in terms of robot design, communication style, and teamwork.

Discuss strategy: Meet with potential alliance partners to discuss strategy and identify which tasks each team is best suited to perform.

Make a shortlist: After evaluating potential alliance partners, we make a shortlist of teams that you would like to invite to join your alliance.

Formulate an invitation strategy: Determine the order in which we will extend invitations to teams on our shortlist, considering factors such as the likelihood of acceptance and rejection.

Extend invitations: Once we have formulated our invitation strategy, we extend invitations to teams on our shortlist.

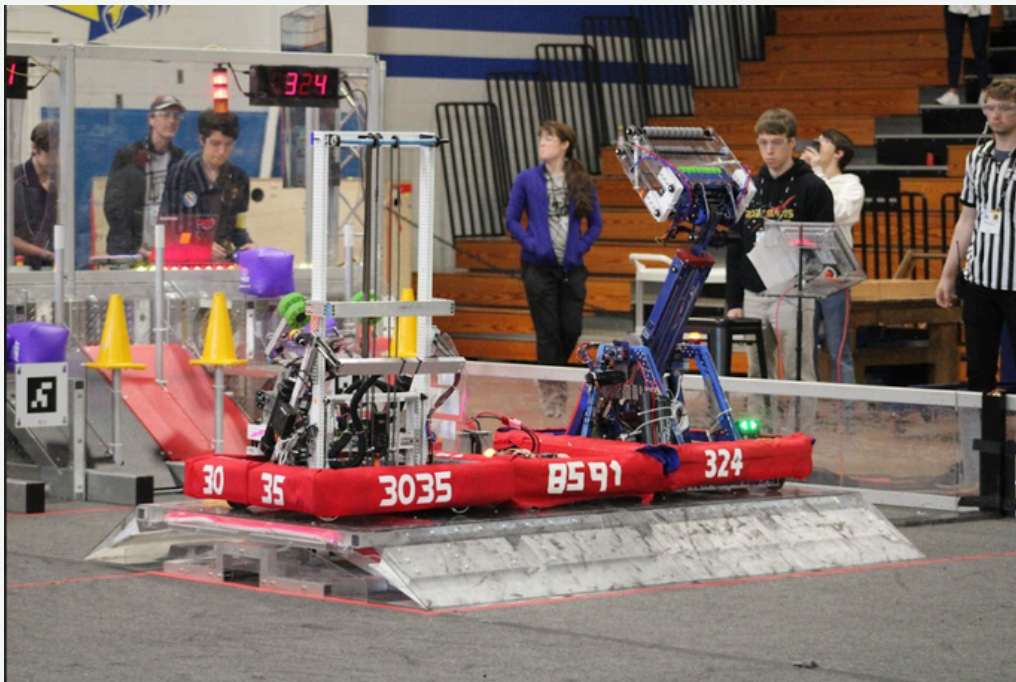


Finalize your alliance: Once we have secured alliance partners, we finalize our strategy and make any necessary adjustments. Practicing and working closely with our alliance partners to ensure that we are well-prepared for the competition.

Game Analysis

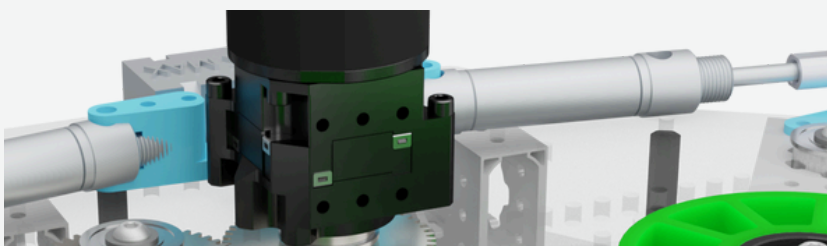
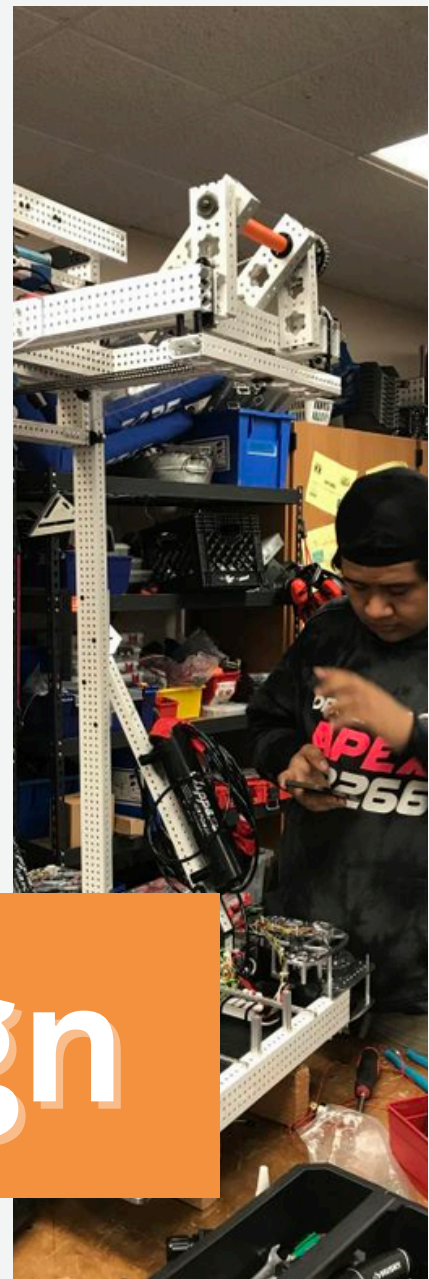
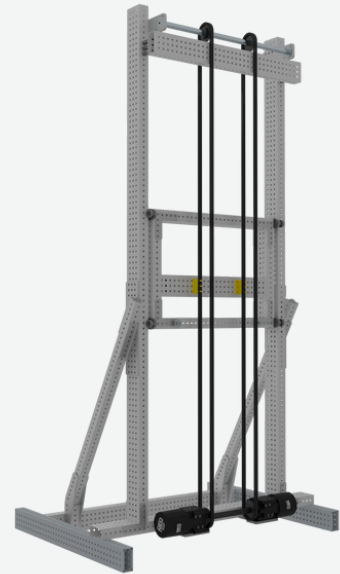
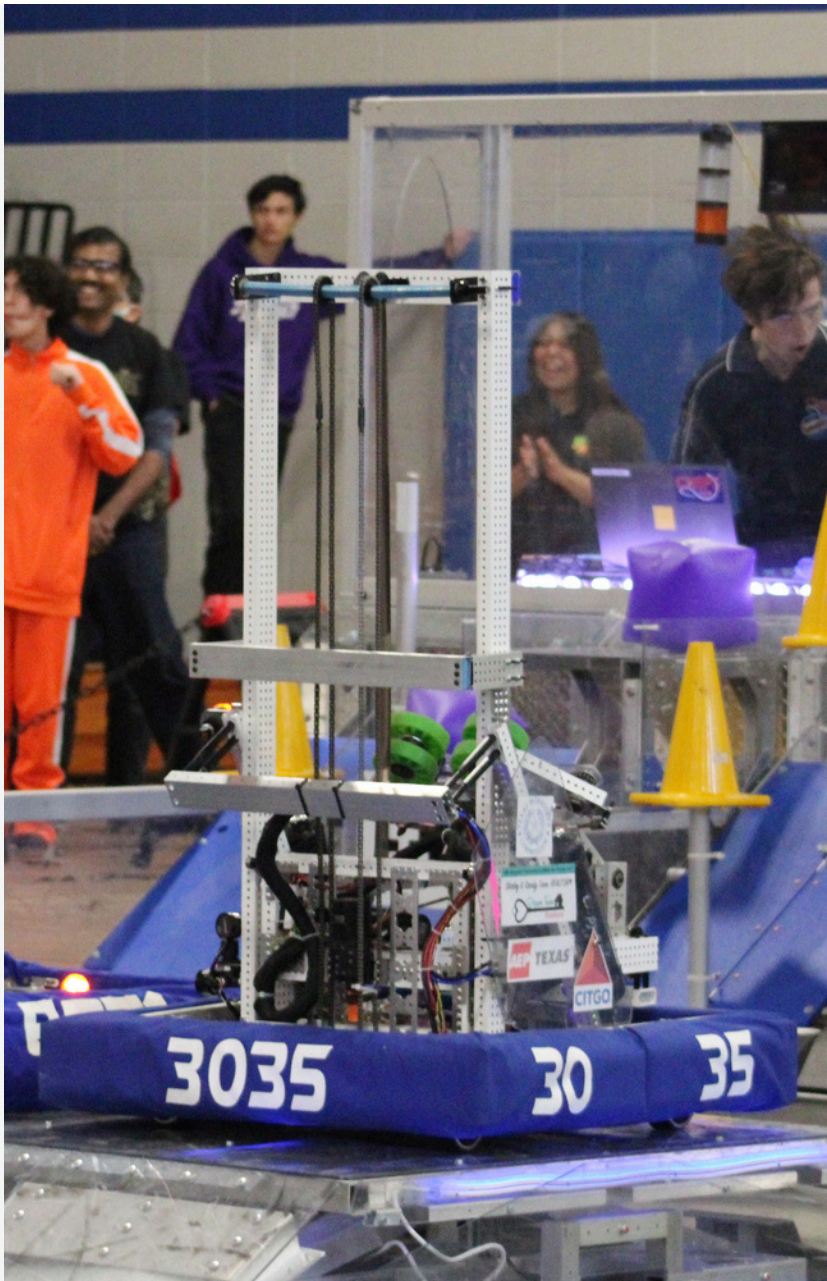
Playoffs

During playoffs, our main goal is to optimize our score to the fullest potential to win all our matches. To excel in CHARGED UP, we prepare for both the qualifying matches and playoffs. In playoffs, we prioritize node scoring to keep up with skilled teams, but also aim for links and attaining a balanced charge station for maximum points both during Auto and Teleop.



To increase our points, we do the following:

- Score Nodes in the high or middle girds during the autonomous period **(+4pts)** to score more points than when driver controlled **(+3pts)**.
-
- Create Links with our alliance **(+5pts)**.
-
- Dock and engage during the autonomous period **(+12pts)** to score more points than when driver controlled **(+10pts)**.

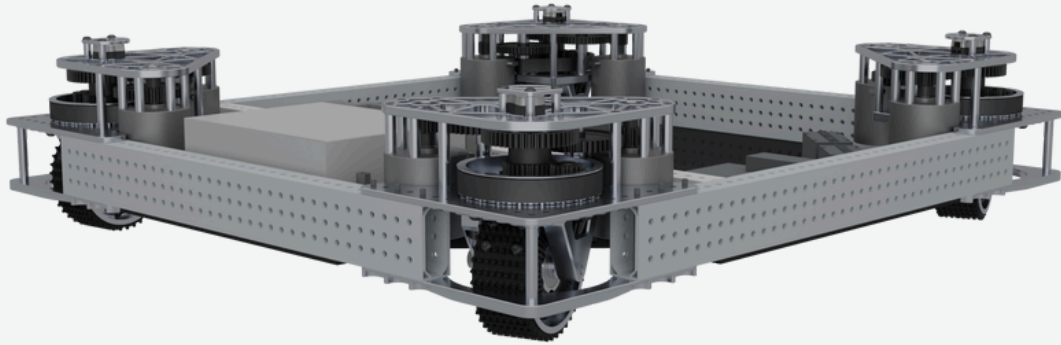


Robot Design



Drivetrain

EMERGENCE's innovative design is evident in our compact, yet reliable **26 x 28 drive train base** that can fit snugly alongside our alliance partners on the CHARGE STATION.



With four independently powered wheels, our robot's holonomic swerve drive boasts unparalleled omnidirectional movement capabilities compared to a standard differential drive train.

EMERGENCE provides drivers with superb control, maneuverability, traction, and response time through various driver enhancements. Our mobility also allows our robot to turn perfectly in place, further increasing its **agility** and **efficiency** on the playing field.

Compact Size

- Fits easily on the CHARGE STATION alongside our alliance
- Maximizes scoring potential through easier maneuverability around obstacles
- Effortless to position robot close to the charging station and charge for Activation Bonus

Four Independently-Powered Wheels

- Provides exceptional mobility and maneuverability on the playing field
- Enables robot to move quickly and effectively
- Allows for more efficient scoring of points

Endgame Strategy

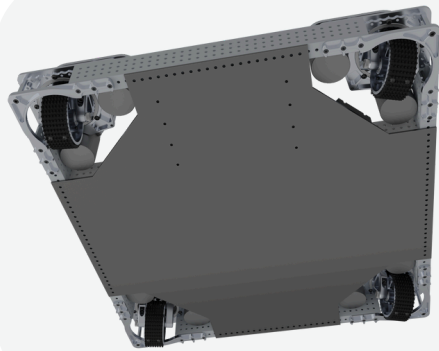
- Scores additional points during final seconds of the match
- Use as an opportunity to win a close match
- Ascend onto the charging station swiftly

Drivetrain

The team's design process for **EMERGENCE** incorporated the MK4i Swerve Module. We also developed a backup design with the AndyMark AM14U5 6 Wheel Drop Center Drive. Our design ensures consistency and redundancy.

By considering various options and designing numerous prototypes, the team created a design that could adapt to any circumstance with optimal performance.

Rectangular Base

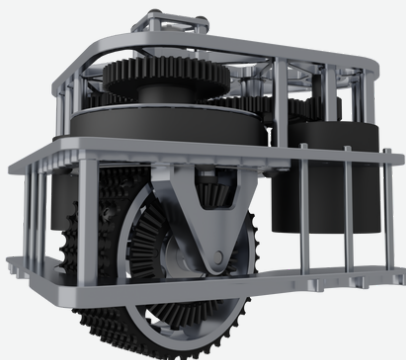


- 728 square inch rectangular base
- 28 inch x 26 inch perimeter
- 20 pound steel base plate to lower center of gravity
- 2 inch ground clearance
- Battery opposite of intake to distribute weight evenly



Neo Brushless Motors

- Lightweight(0.938 pounds each)
- Empirical Free Speed: 5676 RPM
- 4 Neo motors comparable to 6 mini CIM motors
- Cost efficient at ~\$48 per motor



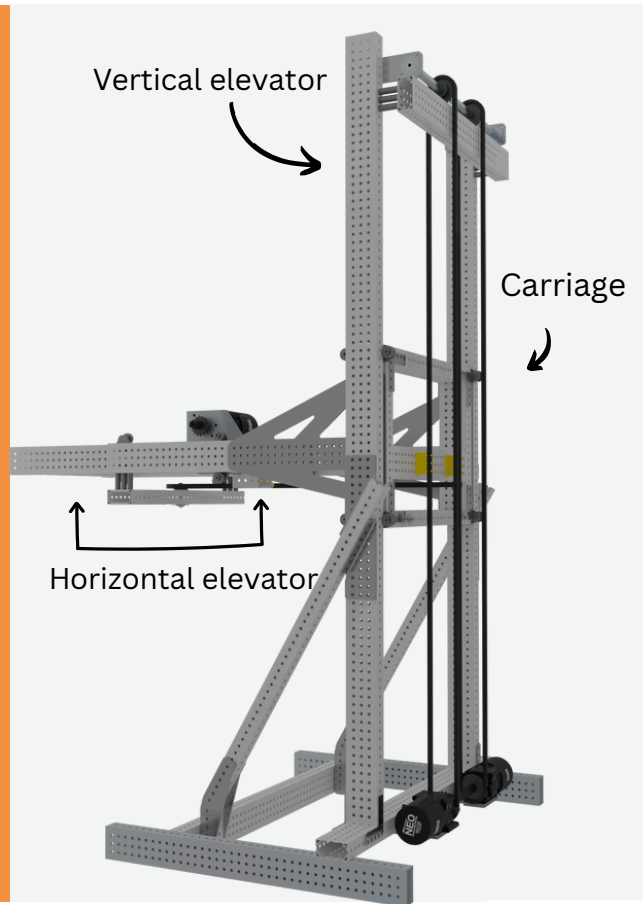
Swerve Drive

- 4-wheel swerve drive
- MK4i custom swerve module
- 8 NEO brushless motors with 4 driving and 4 steering motors.

Elevator

EMERGENCE's elevator system was designed with the goal of cycling as precisely and quickly as possible. To achieve this, we utilized two modified Thrifty Bot 2-stage elevator kits for both horizontal and vertical linear movement.

This design decision brings dependable linear movement for scoring, while ensuring structural integrity.



Final Iteration

Our team made several modifications to our elevator design based on testing in the workshop and competitions.

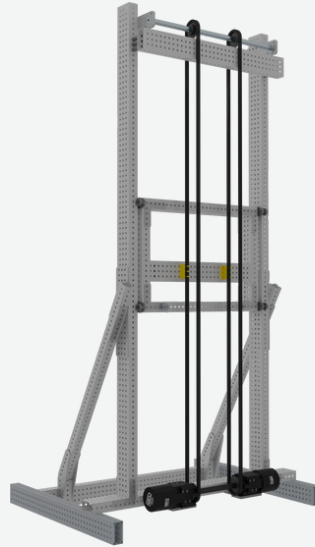
We added a 2 x 1 inch structure to the carriage at our first event, and added additional structure to the elevator after our second event to prevent bending. These changes improved the elevator's performance, durability, and reliability.

In the final iteration, we incorporated all of these modifications into a refined and optimized design, resulting in an elevator optimized for competitive success.

Elevator

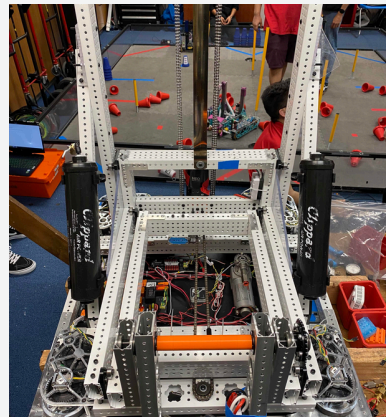
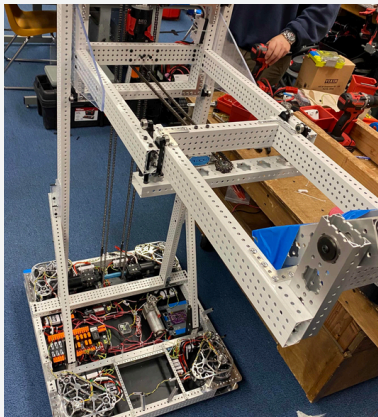
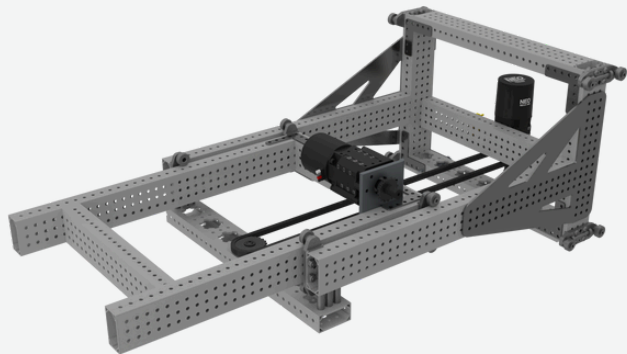
Vertical Elevator

- 48 x 26 inch perimeter
- Constant force spring to strengthen the vertical carriage and lessen strain on the motor
- 34 inch vertical reach
- 2 Neo Brushless Motors



Horizontal Elevator

- 18 x 15 inch perimeter
- Extends horizontally 16 inches from 18.5 inch base
- Mounted on vertical carriage
- 2 Neo Brushless Motors



Elevator: Iteration

Telescoping Arm

As we designed our robot for the game, we considered incorporating a telescoping arm connected by sprocket and chain. While this design had some advantages, such as a familiar look and eight points of contact with the drivetrain, we also identified several drawbacks that came with it.



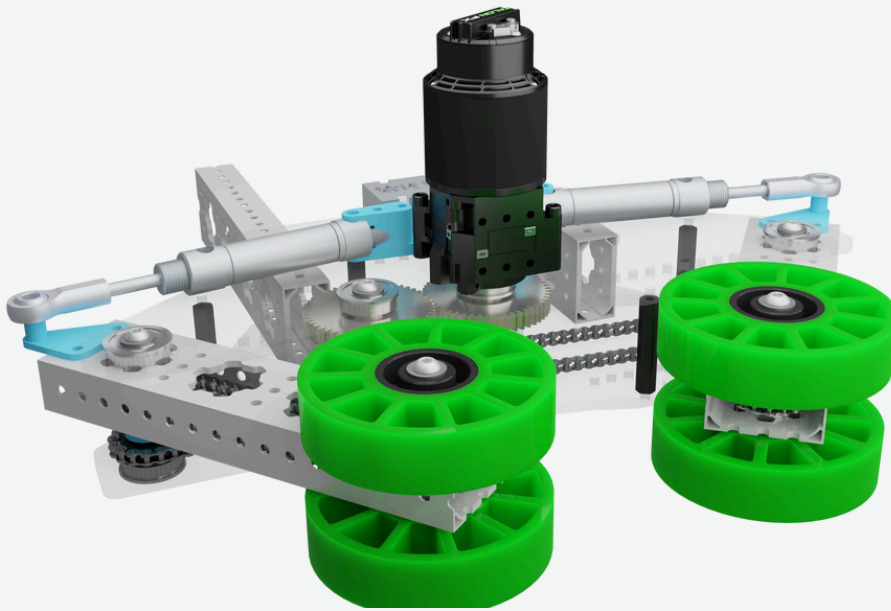
Pros

- Reaches high row for scoring
- Easy to build
- Cost effective

Cons

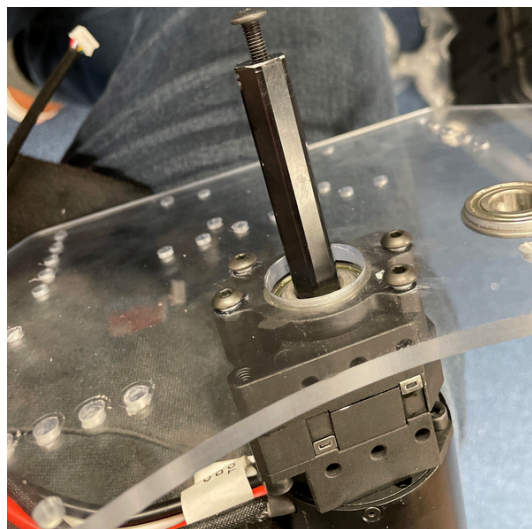
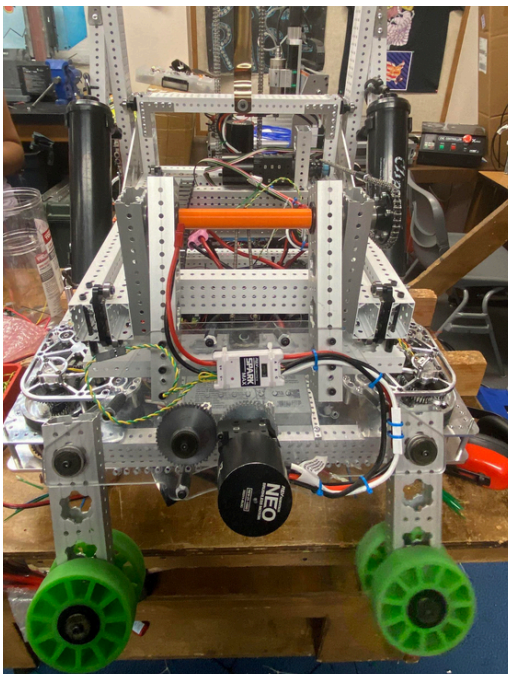
- Springs prone to breakage
- High Center of Gravity
- Difficut to Attach Intake

Intake



EMERGENCE uses a custom poly-carbonate plate design for durability and impact resistance. The intake used to be powered by a Neo Motor but, we switched to a Falcon 500 Motor for extra power and more accurate encoder resolution. The design allows EMERGENCE to accurately shoot elements into the high nodes, intake swiftly, and shoot at a distance.

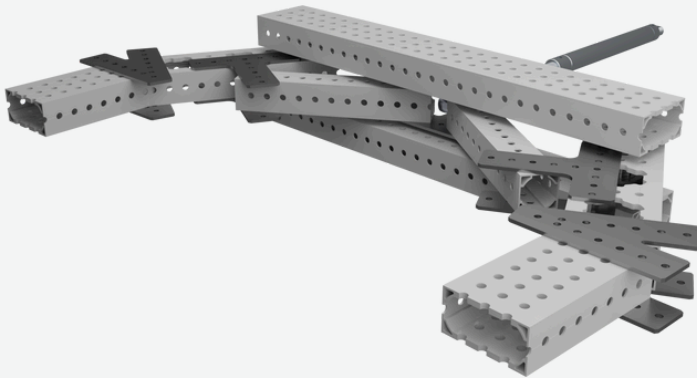
- **Pneumatic claw actuated intake** with 2 small pistons to shift claw positioning and 1 / 8 inch thick poly-carbonate plate
- **Falcon 500 motor** with a 4:1 Gear Ratio
- **Grasps Cones/Cubes** with same mechanism
- 245 degrees of freedom



Intake: Iterations

Passive Intake

In our pursuit of creating an innovative robot design, we experimented with a **Passive Intake System**. However, during testing, we found that this approach would not intake quickly enough and presented significant challenges with alignment, impacting the robot's accuracy while scoring.

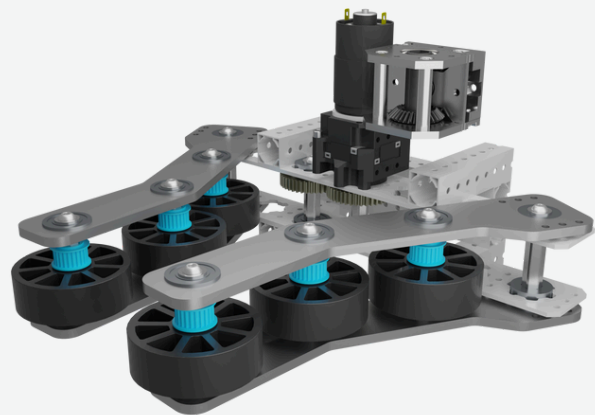


- Constructed using varying sized aluminum
- Intake's claw powered by pneumatic pistons

Active Intake

As we iterated through prototypes, our active intake design encountered several issues, including a limited range of vertical movement and a small intake base, would hinder our ability to score.

- Base constructed using 2 x 1 inch box tubing
- Custom cut poly-carbonate plates for claw
- 8 grip wheels powered by a Falcon 500 Motor



Fueled by our team's unwavering passion for excellence, we embarked on a mission to elevate our Intake system to new heights. We meticulously re-engineered every aspect of the design identifying areas for improvement and refining our approach, ensuring that we incorporated all we learned into our final design.

3d Printed Parts & CNC

3D printing gives us a competitive edge in the arena by allowing us to create custom parts **quickly** and **economically**. 3D printing allows us to adapt to changing circumstances, create multiple iterations, and push the boundaries of what's possible in design and fabrication.

Maxtube Endcaps



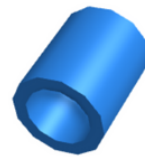
- Support structure
- Varying lengths
- Located from drive train to Elevator

Piston Mounts



- Support structure
- Coincides with plate
- Catered to our needs

Spacers



- Support structure
- Varying lengths
- Essential to Vertical Elevator

This year, Droid Rage Robotics has purchased a **CNC machining**, allowing us to create durable and cost-effective custom-made parts. This advanced technology allows us to create iterations and bring innovative designs to life .

Custom Gusseted Angle Plate

- 150 degree angle Bracket
- 1 / 8 inch Aluminum Plate
- Designed to grant another point of contact for vertical elevator
- Essential to Vertical Elevator Structural integrity

Custom Horizontal Elevator Plate

- 1 / 8 inch aluminum plate essential to elevator
- Designed to have multiple points of contact between Vertical Elevator ,carriage and Horizontal Elevator
- Pocketed to not have excess weight

Software



Subsystems

Scoring Mechanisms

Horizontal
Elevator

Intake

Vertical Elevator

Pivot

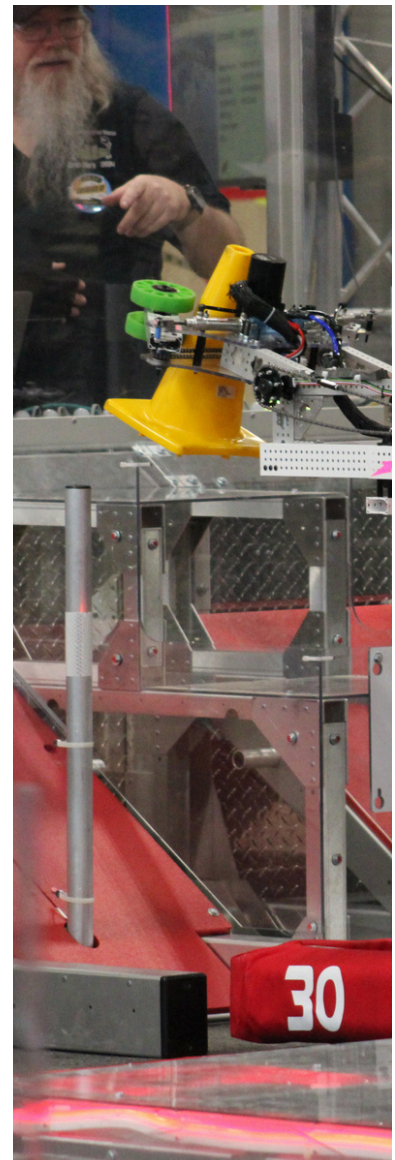
Drive

Swerve Drive

Other

Light

Element
Tracking



Mechanism Control

Trapezoid Motion Profiling

Using trapezoid motion profiling, EMERGENCE generates trajectories for various mechanisms to follow. The robot calculates a state for every 0.02 seconds along the path of the trajectory along with a **target position, velocity, and acceleration** at each point. These numbers are fed into our feedforward control equations.

How does it work?

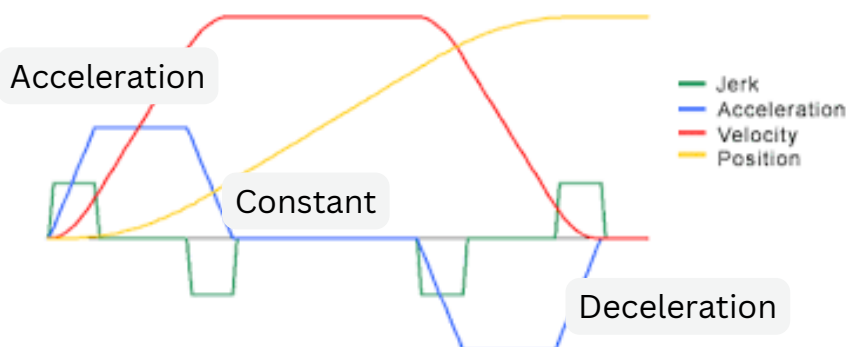
Rather than going at maximum velocity throughout trajectory, mechanisms are designed to follow a **trajectory** to match their motion characteristics. In the trajectory, the speed **slowly ramps up** until the max velocity is reached, then at the end of the trajectory, it drops back down until the mechanism reaches the target position. This helps mechanisms move smoothly.

Trapezoid Profile Code Example

```
// Set trapezoid profile constraints
constraints = new TrapezoidProfile.Constraints(
    maxVelocity: 8, // Radians per second
    maxAcceleration: 16 //Radians per second per second
);

public void setTarget(double positionRadians, double velocityRadiansPerSecond) {
    // Immediately and safely return from function if position is outside of range
    if (positionRadians < Constants.MIN_POSITION) return;
    if (positionRadians > Constants.MAX_POSITION) return;

    // Set goal and current trapezoid profile state
    state = new TrapezoidProfile.State(getEncoderPosition(), getEncoderVelocity());
    goal = new TrapezoidProfile.State(positionRadians, velocityRadiansPerSecond);
}
```



We use Path Planner to generate trajectories for our drive train. Through path planner, our team can draw paths and make them **automatically match** the alliance color.

Mechanism Control

Feedforward Control

By using feedforward control on our robot, we can model the behavior of any system to intelligently **guess the ideal output without feedback** from sensors.

Feedforward Code Example

```
// Create new arm feedforward using static constant, gravity constant,  
// velocity constant, and acceleration constant.  
feedforward = new ArmFeedforward(ks: 0.079284, kg: 0.12603, kv: 2.3793, ka: 0.07)
```

Feedforward equations for our robot

Pivot - $ks \cdot \text{Sign} + kg \cdot \cos(\text{Angle}) + kv \cdot \text{Velocity} + ka \cdot \text{Acceleration}$

Vertical Elevator - $ks \cdot \text{Sign} + kg + kv \cdot \text{Velocity} + ka \cdot \text{Acceleration}$

Horizontal Elevator - $ks \cdot \text{Sign} + kv \cdot \text{Velocity} + ka \cdot \text{Acceleration}$

Intake - $ks \cdot \text{Sign} + kv \cdot \text{Velocity}$

Drive - $ks \cdot \text{Sign} + kv \cdot \text{Velocity} + ka \cdot \text{Acceleration}$

All mechanisms utilize voltage compensation to improve consistency and reliability.

Symbols beginning with k represent a constant.

Sign can be +1, -1, or 0.

Proportional Feedback Control

Since feedforward control does not account for disturbances, we use **PID** (Proportional Integral Derivative) controllers to **account for disturbances using data input from sensors**. Since we have feedforward control to do most of the work, our feedback control only needs a **proportional** constant for accurate movement as shown in the sample below.

Feedback Code Example

```
// Set the proportional correction and tolerance constants.  
controller.setPID(kp: 5, ki: 0, kd: 0);  
controller.setTolerance(positionTolerance: 0.10);
```

- headingError * kp
- positionError * kp

Kp weights for each mechanism are **multiplied** by the error from the sensor whether it be a **gyroscope, drive train encoders, an absolute encoder, or a relative encoder**.

Despite primarily controlling mechanisms through buttons, mechanisms can be controlled manually using sticks to set the desired position and velocity.

Safety and Driver Enhancements

Keeping mechanisms safe

Absolute Encoder on the Pivot

- Even under mechanism failure, our pivot position is always localized accurately

Manual Soft Limits on all Mechanisms

- Prevents mechanism from moving more than physically possible to **prevent breaking mechanisms**.



SafeCanSparkMax and SafeTalonFX

- Abstraction of our motor controllers to have a consistent interface to control motors.
 - Safety through a **software switch**
 - Automatic **logged output**



```
// Safe Motor Abstract class
// This type is implemented differently for different motors
public abstract class SafeMotor {
```

Implementation in Code

```
    // Using ShuffleboardValue<T>, we can easily log motor output, and
    // ensure motor does not run when disabled.
    protected final ShuffleboardValue<Boolean> isEnabled;
    protected final ShuffleboardValue<Double> outputWriter;

    public SafeMotor(ShuffleboardValue<Boolean> isEnabled, ShuffleboardValue<Double> outputWriter) {
        this.isEnabled = isEnabled;
        this.outputWriter = outputWriter;
    }

    // Methods to be overridden by SafeCanSparkMax and SafeTalonFX
    public abstract void setPower(double power);
    public abstract void setVoltage(double outputVolts);
    public abstract void setInverted(boolean isInverted);
    public abstract void setIdleMode(IdleMode mode);
};
```

Driver controller vibration

When the **velocity error** of the intake is too high, the controller vibrates to indicate that the robot has possession of an element.

Field Oriented Drive

Driver's control sticks are rotated by the gyroscope angle to make up on the stick always drive away regardless of bot orientation.

Input Squaring

All stick inputs are squared ($x \cdot |x|$) to allow for finer control at lower speeds.

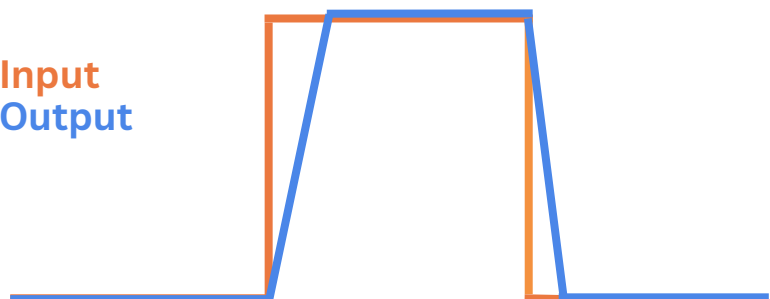
Slew Rate Limiter

Prevents quick rates of changes for the swerve drive to avoid tipping

Slow Mode

Through a button, the driver can slow the drivetrain for finer control

Input
Output



Slew Rate Limiter Example
Diagram

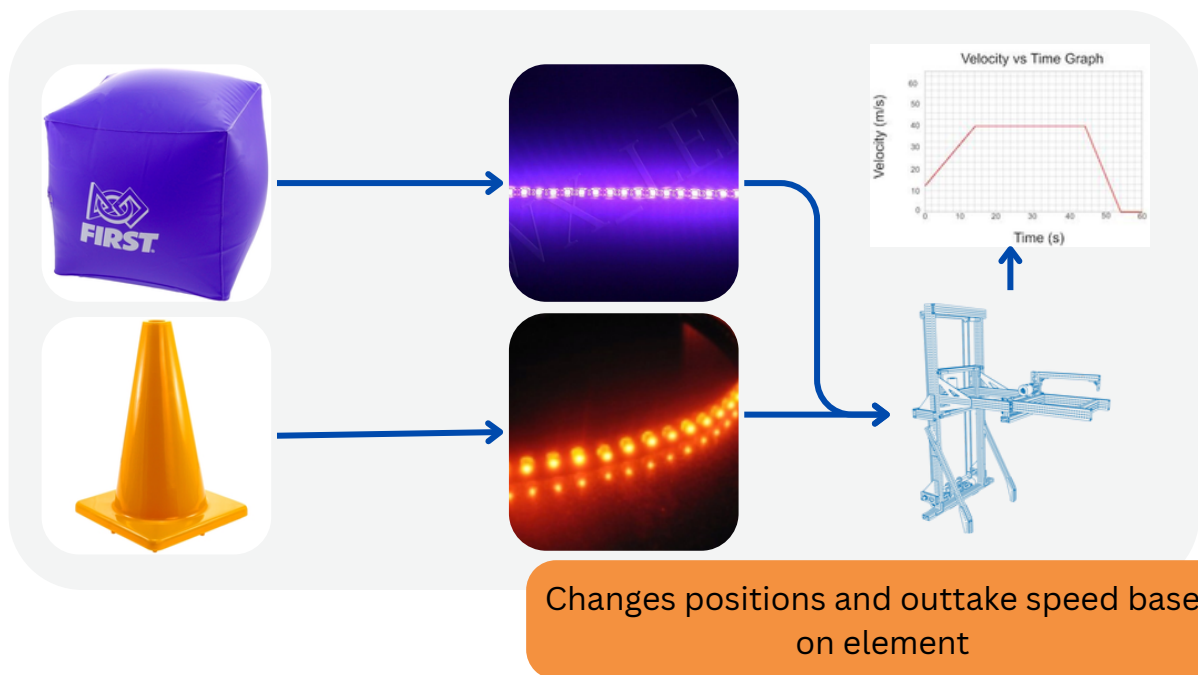
Element Tracking

Element Tracking changes behavior of robot

- Drivers **always alert** of current element.
 - **Lights** change between yellow and purple.
 - Box on screen changes color.
- Elevator and Pivot move based on the tracked element.
 - **Driver errors reduced** since half the number of buttons are needed.
 - Lessens the number of decisions from drivers
- Intake speed changes based on the tracked element.
 - Allows outtaking from different positions including shooting.

Simplified code

- Element tracking is **encapsulated** in code to prevent accidental misuse.
- Our **SuppliedCommand** class allows us to change mechanisms based on robot state
- Our **ShuffleboardValue<T>** allows us to easily log data onto shuffleboard and get values from Shuffleboard



Element Tracking Code Example

```
// Enum to represent
// all possible elements
public enum Element {
    CONE,
    CUBE
}

// Setter to automatically write data
public static synchronized void set(Element element) {
    TrackedElement.element = element;
    elementWriter.write(element.name());
    isCubeWriter.write(isCube());
}

public static synchronized Element get() {
    return TrackedElement.element;
}
```