PUSH BACK

HURRICANE ROBOTICS

ENGINEERING NOTEBOOK

MAY 2025 - JANUARY 2026

HEART LAKE SS

Brampton, Ontario

TABLE OF CONTENTS

Date: 02/23/2026 Event Name: Ontario HS Provincial V5RC Championships

Innovate Award Submission Information Form

Instructions for team: Please fill out all information, printing clearly. This form should be included either behind the front cover, or in a clearly marked section in your Engineering Notebook. Teams may only submit **one** aspect of their design to be considered for this award at each event. Submission of multiple aspects will nullify the team's consideration for this award.

Full Team Number: 296A

Brief description of the novel aspect of the team's design:

We developed a custom Augmented Monte Carlo Localization (aMCL) system written in Rust. This sensor-fusion algorithm acts as a probabilistic filter, combining high-frequency relative data from unpowered tracking wheels (Odometry) with absolute environmental data from four distance sensors (Likelihood Field). By continuously tracking a cloud of 384 virtual "particles," the robot can estimate its exact global position (x,y) on the field with millimetre precision. Crucially, the "Augmented" feature detects when the robot has been physically pushed off-course (the "kidnapped robot" problem) during defensive interactions and automatically injects random particles to instantaneously recover its true position without human intervention.

Identify the page numbers and/or the section(s) where documentation of the development of this aspect can be found:

You can find our design process for the Localization and aMCL code in the section PG.1.0:

- Robot Localization: Pages #180-181
- Odometry Implementation: Pages #182-183
- MCL Implementation: Pages #184-186

Explain why your submission is unique from other approaches to the problem it solves or task it performs:

The standard for VRC localization is pure odometry, which suffers from unbounded drift, or the VEX GPS, which offers subpar performance and fails when the field strips are occluded. Our aMCL implementation is unique because it brings university-level SLAM concepts to embedded V5 hardware. A standard particle filter is too computationally expensive for the V5 Brain's limited Cortex A9 processor. We overcame this by leveraging Rust's low-level memory control to implement a Structure of Arrays (SoA) data layout and writing custom SIMD (Single Instruction, Multiple Data) intrinsics. This allows us to parallelize the probability calculations of 384 particles within a 10ms control loop—a feat of optimization that offers the smoothness of dead-reckoning with the absolute truth of GPS, without the fragility of either. While other robots lose their position after a collision, ours mathematically recovers its state in real-time. We believe that MCL will be as big of an innovation to VEX as odometry has been abd we anticipate more teams to follow suit in the upcoming years.

NOTEBOOK INTRODUCTION

MEET THE TEAM

Laksan Mohan | 06/24/2025

Who Are We?

Hurricane Robotics, VRC Team 296A, is a group of passionate students from Heart Lake Secondary School, located in Brampton, Ontario. As a large and ambitious team featuring members with a variety of experience levels and skill sets, we combine our strengths to dominate in our second year in VEX Robotics!

Our mission as a team is to create a strong foundation and lasting legacy that sets up and empowers future Heart Lake students to embrace robotics, build on our successes and find their own success in STEM.

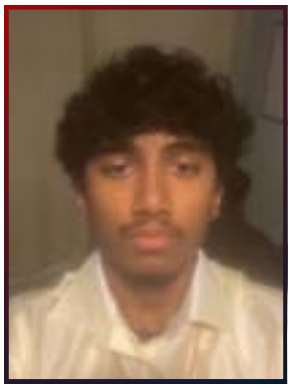
Our Team

Laksan Mohan | Captain / Builder / Notebooker

Hi! I'm Laksan and I've had a deep passion for robotics ever since I was introduced to LEGO Mindstorms EV3s when I was 7. Since then, I have led two EV3 robotics teams, competed in various tournaments and have been teaching Grade 5 students robotics at my old elementary school for the past 4 years.

Glae Alejo | Lead Programmer / Notebooker

Hey, my name is Glae Alejo and I'm a programmer for our Vex Robotics team. My first introduction to coding was the Scratch-like block coding for Lego EV3 Mindstorms in Grade 6 and ever since I've been passionate about robotics programming. My favourite part about robotics is the testing and problem-solving process in creating effective autonomous code.



Thikshaan Jeehavaren | Driver

Hi, my name is Thikshaan Jeevaharan, and I am the driver for our robotics team. I've always had a strong interest in how things work, and driving the robot during competitions is something I'm passionate about. I spent a lot of time practicing and learning how to control the robot smoothly and make quick decisions during matches. I work closely with my teammates to understand our strategy and help the team perform at our best. Being the driver is a role I take seriously, and it's one of the things I enjoy most about being part of the team.

Vabithan Gangatharan | Builder / Notebooker

I am Vabithan Gangatharan, a student at Heart Lake SS. I gained interest in Robotics from my peers and my father, I discovered a new version of myself - a version made just for this. Taking part in robotics has enhanced my understanding of this field, whether it be through conversations with those in the coding group, or by first hand experience of being able to build the robots and seeing how they work. Apart from my love for robotics and gaming, I love cooking.

Our Goals

For the current competition season, we aim to:

- **Qualify for Worlds**
  - We aim to place in the event finalists of regional competitions or win a judged award to secure a spot in Provincials.
  - In addition, to have a chance to qualify through Skills, we plan to develop a max point (119) skills routine.
  - Ideally, we want to qualify early into the competition season so we can spend more time planning for Provincials without worrying about having to qualify.
- **Achieve A Consistent Autonomous Routine**
  - By November, we will develop a reliable autonomous program that can consistently score 7 balls on both the long and middle goals in autonomous periods.
- **Develop Training Programs for New Members**
  - Create a series of workshops and resources to bring newer team members up to speed with building, coding, and competition strategy.
- **Secure $2000 in Sponsorships:**
  - Develop sponsorship proposals and presentations to secure funding for materials, competition fees, and travel expenses.
- **Host Two Fundraisers:**
  - Plan and execute two school fundraising events to generate revenue for the team.

Our Team Structure

This season, our team will be organized into two main sub-teams: mechanics and software. However, we encourage members to look into and try their hand in different tasks as it better shows where our talents and interests lie.

Team Meetings:

- We run an after-school COY where we meet every Wednesday and Friday from 2:15-4:15 PM as a team to work on the robot. In the opening minutes of each meeting, everyone goes over their progress in their specific roles and in their subteams
- Overall team timelines and goals will be set in this meeting
- Any member is welcome to discuss any matter about the team

Mechanics:

- Primary Team Members: Laksan, Vabithan
- This subteam focuses on the brainstorming, CAD, and construction of our team's robot
- Team members collaborate on a shared concept and goal, following the timeline set in the team meetings
- Members of the mechanics team also work on strategy to assess the game, effective ways to score, and how other teams are developing match strategies
- Mechanics operates on a flexible work schedule, where members work whenever they have time while keeping the others in the loop

Programming:

- Primary Team Members: Glae
- This subteam is tasked with creating the drive controls and autonomous routes and analyzing game strategies.
- Software members should work with mechanic members to learn the functions of the robots and to create drive controls to the driver's preference
- Software members should communicate with mechanic members any issues that arise with the functionality during programming

Notebooking:

- Primary Team Members: Laksan, Glae, Vabithan
- This subteam is responsible for documenting the entire engineering design process throughout the season
- Members record brainstorming ideas, design iterations, testing results, and reflections in the Engineering Notebook to ensure all stages of development are clearly communicated.
- The notebooking team also manages written reports, team bios, and visuals such as CAD renders and diagrams
- They work closely with the Mechanics team to stay updated on changes and ensure the documentation accurately represents the team's progress.
- Consistent communication and attention to detail are key parts of this subteam's workflow.

Drive:

- Primary Team Members: Thikshaan, Vabithan, Laksan
- Thikshaan is the main driver of 296A, and, if he is sick, Laksan steps in as the secondary driver
- The Drive subteam focuses on developing driver skills, refining match strategies, and maintaining consistent robot performance on the field
- Team members conduct regular driving practice sessions to improve precision, coordination, and adaptability during competition scenarios.
- This subteam also collaborates with Mechanics and Programming to fine-tune control schemes, adjust mechanisms for driver comfort, and ensure the robot performs reliably under match conditions
- Drive team members analyze match footage to identify areas for improvement and optimize both offensive and defensive gameplay strategies.

NOTEBOOK FORMATTING

Laksan Mohan | 06/25/2025

Goal: Set up consistent formatting and an efficient and easy to understand system for notebook entries.

Thank you so much for volunteering your time to judge. We have truly embraced the spirit of VEX and are a true example of what VEX can do for kids and the impacts you have by volunteering your time today!

We understand judging notebooks can be daunting. Below is a summary of how we format our notebook.

Notebook Layout:

We will be doing a fully electronic notebook this year. We have found it helps keep our notebook up-to-date, allowing multiple people to work at one time on the notebook as the design process is happening. Every entry is labelled under one of these 7 categories:

1. Project Management
   - Entries relating to the management of our team
2. Define The Problem
   - Entries relating to the 1st stage of our design process
3. Brainstorm & Research Solutions
   - Entries relating to the 2nd stage of our design process
4. Choose The Best Solution
   - Entries relating to the 3rd stage of our design process
5. Design & Build Prototype
   - Entries relating to the 4th stage of our design process
6. Test & Refine
   a. Entries relating to the 5th stage of our design process
7. Game Analysis & Strategy
   - Entries on scouting, research of other robots, game strategy, match analysis of our own matches (as well as Signature Event Finals), and game manual updates

General Page Set Up:

- Entry Title
  - Appears when anything in the entry bar changes or the topic being discussed changes
  - The title always appears at the top of the page; any space left over on the previous page will not be utilized
- Entry Bar
  - This always appears in the header/footer of every page
  - It includes the design number, the category the entry is on and the page number

- Entry Byline
  - This always appears after the title
  - It includes the author and date of the entry
- Goal
  - A short statement about goals for a page/meeting; it appears after the author and date
  - If a goal is not needed/obvious, it will not be mentioned
  - This always appears after the byline

Documentation Summary

Will be Documented	Will NOT be Documented
• Significant changes in the design or programming of the robot • Strategic discussion throughout the season, especially after tournaments and game manual updates • Testing and reflections about robot components • Robot research	• Minor changes in code, build, etc... • Cosmetic robot decorations or other elements that don't relate to the design process

DESIGN NUMBERING

Laksan Mohan | 06/25/2025

Goal: Lay out an easy-to understand system to help identify the current design cycle & specific robot version.

Requirements:

1. The numbering system must be clear and concise with no room for interpretation
2. Our system must differentiate between CAD models and physical robots
3. We need to be able to distinguish between major design changes
4. Differentiating minor changes is preferred but not necessary if it adds too much complexity

Our Numbering System:

C.0.0.0

CAD or Physical Robot?

The first and only letter is used to differentiate between CAD models (C) and a fully constructed robot (R).

Build/Design Number

The first number is used to identify whenever we completely switch designs in CAD (meaning we go through another design cycle) or fully rebuild the robot.

Major Iterations

The second number will be used to characterize major changes to the robot or cad, for example completely rebuilding or redesigning the lift system.

Minor Iterations

The third number, used only for the physical robot, represents minor iterations since the last major change (e.g., changing a wheel type or pivot point location). It is not used in CAD due to the extensive amount of small changes and constant tweaking.

Example: R.2.0.6

This would represent our second (2) build (R), with no major changes (0), and six minor changes (6).

Our Subsystem Numbering System:

We will be using a similar system to identify the current subsystem version.

I.0.0

Which Subsystem?

The first and only letter is used to differentiate between which subsystem I am referring to. The letter will be the first letter of the subsystem. For example, I for the intake and D for drivetrain. If the subsystem contains multiple stages, it will have a number next to the letter. For example, the first stage of the intake would be I1.

Build/Design Number

The first number will be used to characterize major changes or rebuilds to the subsystem, for example rebuilding the drivetrain.

Minor Iterations

The second number represents minor iterations (e.g., changing a wheel type or pivot point location).

Example: I3.1.6

This would represent our first (1) build of the intake stage #3 (I3) with six minor changes (6).

NOTEBOOK ACCOUNTABILITY

Laksan Mohan | 06/25/2025

Goal: Set up expectations for the notebooking that each of us does throughout the season.

To meet our team's goals of maintaining valuable documentation that would allow someone to be able to replicate our robot we have decided to create goals to ensure notebook accountability. Other than putting all technical elements into the notebook, these are our goals throughout this season:

1. The person that did the technical component does the notebooking on those components; additionally if multiple people work on the same component everyone does the notebooking.
2. Notebooking must be completed the same day that the technical component is completed
3. At least one member of the team must look at the documentation in its entirety within 24 hours after the notebooking is completed; additionally no edits should be made after the entry date

To maintain our accountability, if we break any of these goals we set for ourselves we will create a red notice explaining the goal that we broke. These notices will be put after the documentation and be dated. This is what one of those notices would look like:

Accountability notice: Vabithan Gangatharan completed the notebooking late (goal 2) on 8/16/2024

Conclusion:

Our team is setting goals for our notebooking to ensure our team is accountable to getting the documentation complete and done in a timely manner. Additionally we will have accountability notices when we don't achieve our accountability goals.

OUR DESIGN PROCESS

Laksan Mohan | 06/26/2025

Robotics involves countless hours of tinkering and iteration to develop innovative solutions for the challenges this year's competition brings. To give this process a more structured and consistent approach, we will be using a defined Engineering Design Process throughout the season. This method will guide us through every step, from brainstorming ideas to testing and refining our robot, ensuring that we work efficiently, make informed decisions, and keep track of each design cycle. Once we reach the end of the steps described, the process doesn't stop–we will repeat the design cycle to revisit various subsystems of our robot to develop more effective solutions and arrive at a final, working solution.

We will integrate this process into our engineering notebook entries by labelling the specific design cycle and stage the entry is addressing in the entry bar of each page.

Our Design Cycle

1

Define The Problem:
Defining the problem sets the foundation for all our design decisions, ensuring we understand the game's challenges and objectives. Without a clear understanding of what we're trying to solve, we risk losing focus and creating a robot that doesn't effectively address the game's challenges.

Implementation:

- Collect as much relevant information as possible about the problem
- Clearly identity and define the problem using relevant descriptions and pictures
- State any limitations, such as size restrictions, weight limits, budget, or materials
- Establish specific objectives for what we want to achieve in terms of performance, reliability, and strategy
- Establish a timeline and tasks to ensure timely progress

2

Brainstorm & Research Solutions:
By exploring a wide range of relevant ideas to the defined problem, we open up the possibility of discovering unique solutions that could give us a competitive edge. Additionally, with our limited resources, researching allows us to see what has been done before and learn from others' experiences to avoid wasting any time or money.

Implementation:
- Brainstorm solutions to the defined problem, even the wild or unconventional ideas, since they might lead to creative breakthroughs
- Create rough sketches or diagrams to visualize ideas
- Weight the benefits and drawbacks of each solution
- Collect testing data to prove the viability of solutions
- Consider and cite ideas from outside sources

3

Choose The Best Solution:
Review all the ideas generated during brainstorming and research and decide which one is the most viable to move forward with.

Implementation:
- Using a decision matrix, judge each possible solution on criteria, such as feasibility, cost, complexity, reliability, effectiveness, and how well they align with the game strategy
- Based on the analysis, choose the best solution
- Break down the chosen solution into tasks to optimize workflow
- Use a Gantt Chart to organize the timeline for completing tasks

4

Design & Build Prototype:
Building a prototype helps us uncover potential flaws, limitations, or unexpected challenges that we might not have noticed in the design phase. By identifying these issues early, we can make improvements and adjustments, saving time, effort, and resources in the long run.

Implementation:
- Create sketches and design a CAD model to plan the solution
- Record the steps taken and parts needed to build

5

Test & Refine:
Evaluate the performance of the prototype, validate the design and ensure that it performs as intended before the competition.

Implementation:
- Define a set process for testing the solution
- Thoroughly test all aspects of the solution, recording the results
- Analyze the testing data and look for specific problems or limitations in the design, such as mechanical weaknesses, programming glitches, or inefficiencies in the robot's operation
- Make adjustments and retest as needed
- Continue a cycle of testing, analyzing, refining, and retesting until we achieve the desired level of performance and reliability

Our Design Philosophies

Our design philosophies guide how we approach each challenge and ensure that we stay true to our values as a team. We believe that our work should be driven by these core principles:

- **Do It Right the First Time**
  - We believe in taking the time to plan and execute correctly from the start. Cutting corners leads to more issues down the road, so we aim for quality and attention to detail in everything we do, from initial design sketches to final builds. Even the smallest details can make a big difference in the performance of our robot.
- **Collaboration Over Competition**
  - We value teamwork and believe that the best solutions come from combining the ideas and skills of every team member. We foster an environment where everyone feels encouraged to share their thoughts, ask questions, and contribute to the robot's design.
- **Iterative Improvement**
  - We embrace the idea that the first solution is rarely the best one. Our design process is centred on testing, refining, and continuously improving our robot. We view each iteration as a chance to learn and get closer to creating a more efficient and reliable solution.
- **Fail Fast, Learn Faster**
  - We understand that failure is a natural part of innovation. Instead of fearing mistakes, we see them as opportunities for growth. By identifying what doesn't work early on, we can quickly pivot and find better solutions.
- **Creativity**
  - We encourage out-of-the-box thinking and creative problem-solving. By trying out strategies that may seem unusual, we are able to find novel strategies that give us a competitive edge while staying true to the game rules and constraints.
- **Sustainability for Future Teams**
  - We aim not only to build a competitive robot but also to lay the groundwork for future Heart Lake students. We thoroughly document our processes, successes, and lessons learned so that future teams can build on our foundation.

GAME INTRODUCTION + ANALYSIS

PUSH BACK INTRO

Laksan Mohan | 06/27/2025

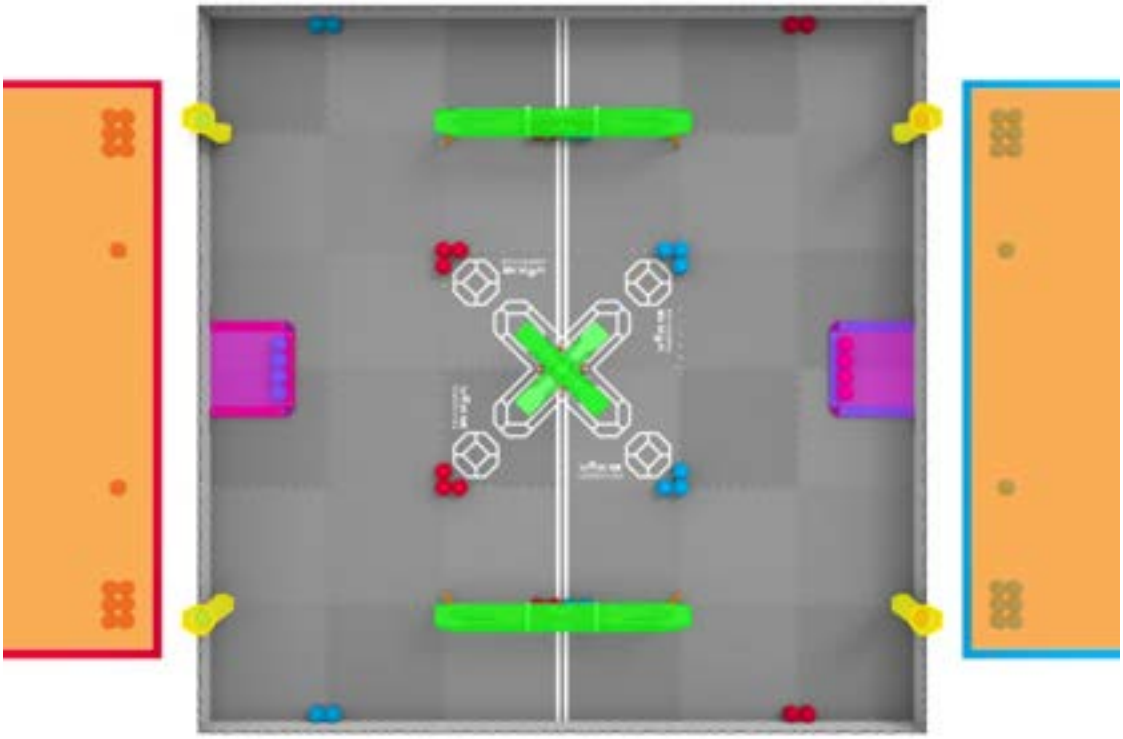


Figure #1: An overhead view of the V5RC Push Back Field, with Alliance Stations (orange), Loaders (yellow), Park Zones (pink), and Goals (green) highlighted. (Source: Game Manual)

Push Back is played on a 12'x12' square Field, set up as illustrated in the figures throughout. The game has 2 main components: Head-to-Head Matches and Skills. In these matches, two alliances–red and blue–both consisting of 2 teams each, compete in Matches consisting of a fifteen (15) second Autonomous Period followed by a one minute and forty-five second (1:45) Driver Controlled Period. Teams are trying to maximize the amount of win points they can get to increase their tournament ranking. The object of the game is to attain a higher score than the opposing Alliance by Scoring Blocks in Goals, Controlling zones within Goals, clearing Loaders, and Parking in defined zones at the end of the Match. During the Autonomous Period, robots move on their own and try to score as many points as possible along with the Autonomous Win Point (AWP); one of the three win points that can be attained in a match. During the Driver Controlled Period, drivers control their robots to get more points than the other alliance. If a team beats the other alliance or gets more points, they are awarded with 2 win points, if they tie both are awarded with 1 win points, and for the losing alliance, each team is awarded 0 win points. In skills, robots compete by themselves to maximize the points they can score in 1 minute matches. For skills, the team is ranked on their combined programming and driver skills scores.

GAME ELEMENTS

The field consists of:

- 88 Blocks, each colour has 44 blocks consisting of:
  - 2 Preloads, one per robot
  - 12 Match Loads
  - 18 that start the Match in predetermined locations on the Field
  - 12 that start the Match in Loaders
- 4 Loaders, 2 adjacent to each Alliance Station
- 3 Goals
  - 2 Long Goals
  - 1 Center Goal, consisting of an Upper goal and a Bottom Goal
- 2 Park Zones, one blue and one red

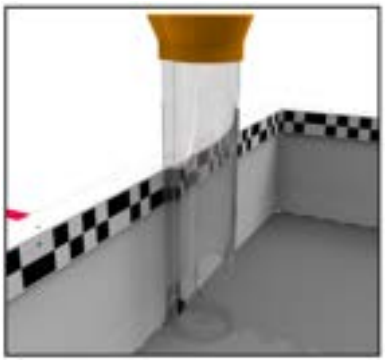
Source: Game Manual

Block:

- This is the primary scoring element of Push Back
- A blue or red 18-sided hollow plastic polygonal object with flat faces
- Weight: ~40 grams
- Each cross-section measures approximately 3.25” (82mm) between pairs of opposing flat faces, and 3.85” (98mm) between pairs of opposing corners

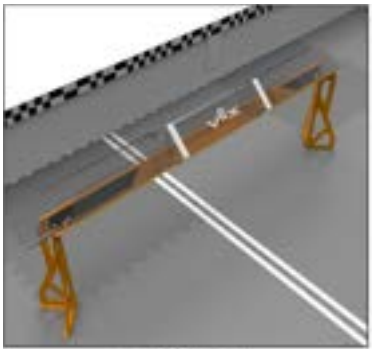
Loader:

- One of four 21.34” (542mm) tall plastic and rubber structures each attached to the Field Perimeter
- Robots may remove Blocks from Loaders during a Match
- Drive Team Members may add Match Load Blocks to Loaders during the Match
- Each Loader begins the Match containing **6 Blocks**



Source: Game Manual

GAME ELEMENTS



Source: Game Manual

Long Goal:

- Constructed out of plastic and metal components into which up to **15 Blocks** can be scored
- A Long Goal is 48.8” (1239mm) in length, with a 13.33” (339mm) enclosed center section
- The Control Zone for a Long Goal consists of the space between (but not including) the white tape lines (highlighted in green in Figure #2), and holds up to 3 Blocks.

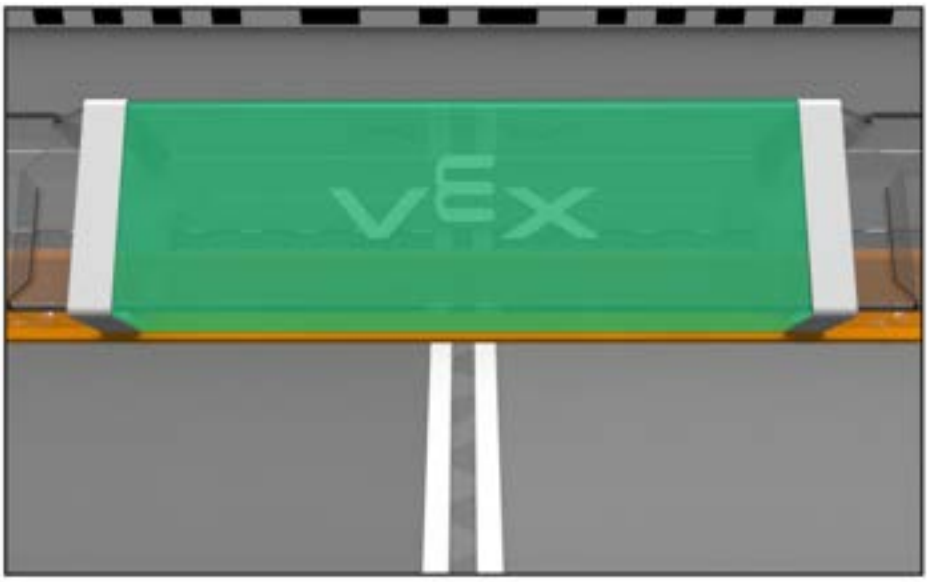
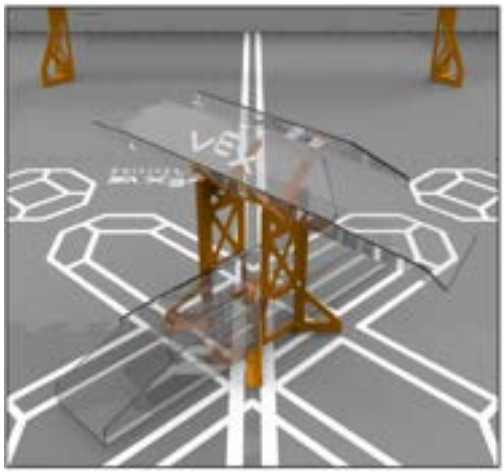


Figure #2: : The Control Zone (highlighted green) of a Long Goal consists of the volume between the white tape lines, as shown. (Source: Game Manual)

Center Goal:

- Each Center Goal is 22.6” (574mm) in length; holds up to **7 Blocks** per level
- The Control Zone for a Center Goal is the ENTIRE GOAL
- The Center Goal is spilt into 2 sections:
  - The Upper Goal
  - The Lower Goal
- From this point forward, the Upper Goal will be referred to as the Middle Goal and the Lower Goal will be referred to as the Bottom Goal



Source: Game Manual

GAME ELEMENTS

Source: Game Manual

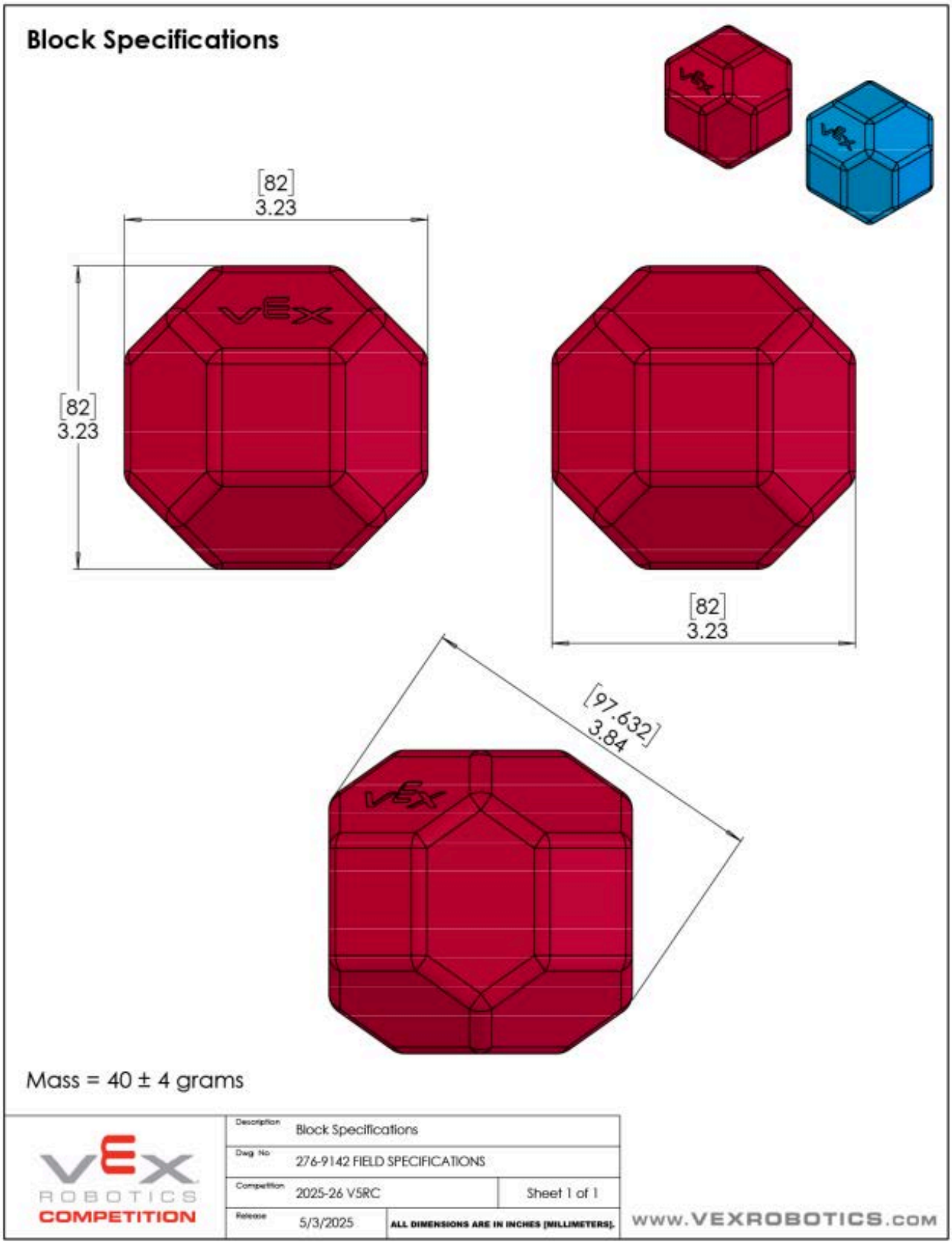
Park Zone:

- A Field Element that marks a location where Blocks begin a Match and Robots can be Parked at the end of the Match
- Park Zones are made of red or blue plastic extrusions and black plastic connectors
- Each Park Zone is 18.87“ (479mm) wide x 16.86” (428mm) deep.

Game Object Analysis

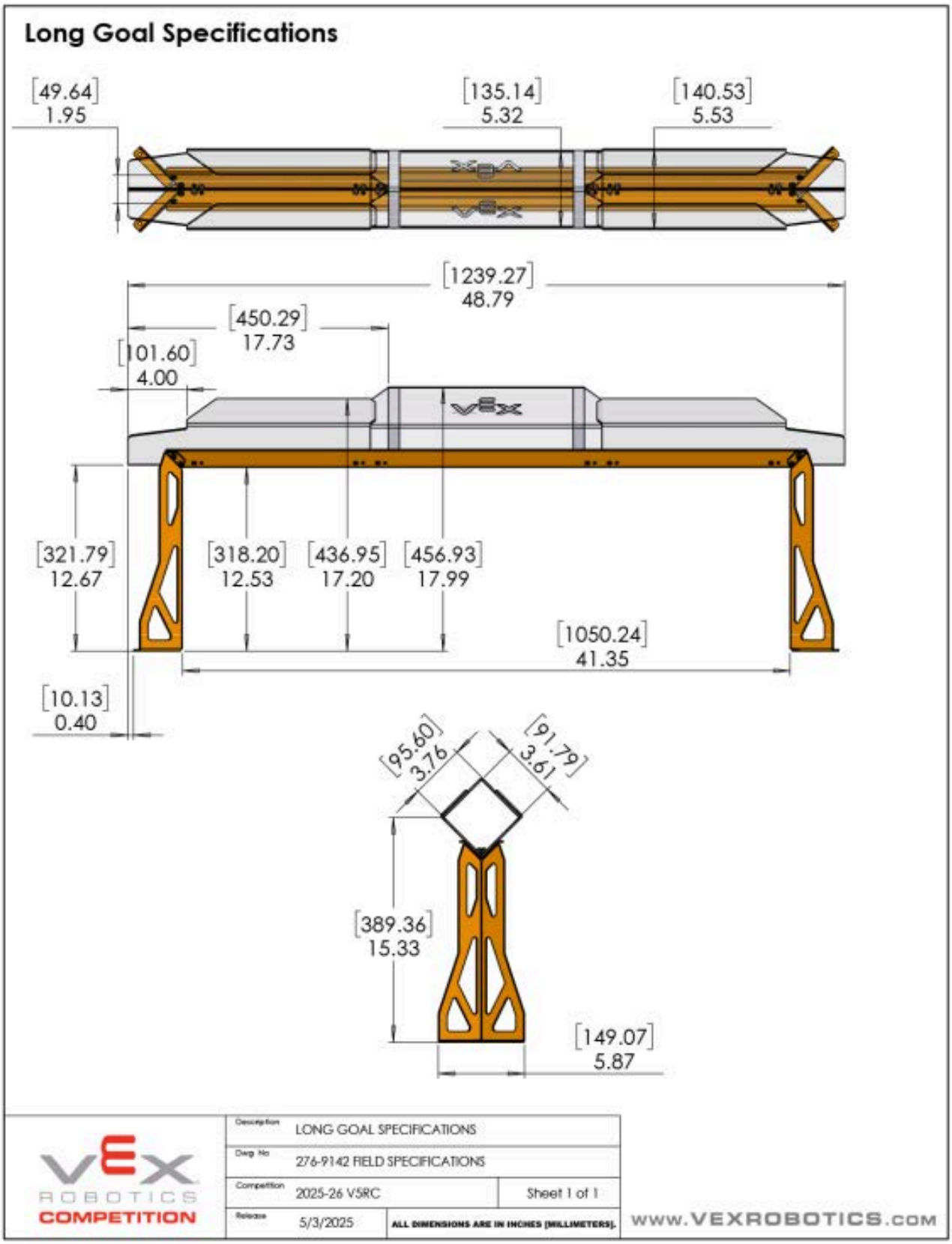
- Due to the tight dimensions between the blocks, match loaders and goals, accuracy and consistency is going to a very important aspect of this game
  - This will be a particular challenge with our autonomous routines
  - To help with this, we can implement aligners and use advanced systems to program our robot, such as Odometry, Monte-Carlo Localization (MCL) or Extended Kalman Filtering (EKF)
  - Additionally, when building, I will ensure I add some room for error (for example, making the intake wide enough that it spans 2-3 balls
- A Block is manufactured out of a smooth plastic that is slick
  - While match loading Blocks, the slick texture could be hard to grip, creating issues while attempting to load
  - A grippy material will be required to handle Blocks, due to the slick nature of the game object
- A Block is 3.25” in height at all times
  - This game object possess a unique geometry that remains the same height regardless of the orientation
  - This means I do not need to design the intake so it can pivot to different positions. Instead, I can opt for a floating intake
- Due to the different heights of the goals, I need to design an intake that can score at different levels
  - This may result in a pivoting hood design or different outtake stages
- The park zone is relatively high at 1"
  - This poses a tricky design challenge as I cannot mount cross-braces/elements too low to the ground
  - This may result in our design using sleds or lowered wheels to cross the park zone
- The match loader is open at both sides
  - Due to this, a design such as side rollers might work quite effectively to match load quickly
- Long Goals can hold up to 15 blocks
  - This means that when the goal is completely filled, the combined weight of the blocks is considerable
  - Thus, our outtake must be powerful to push out these blocks when scoring

EXACT DIMENSIONS



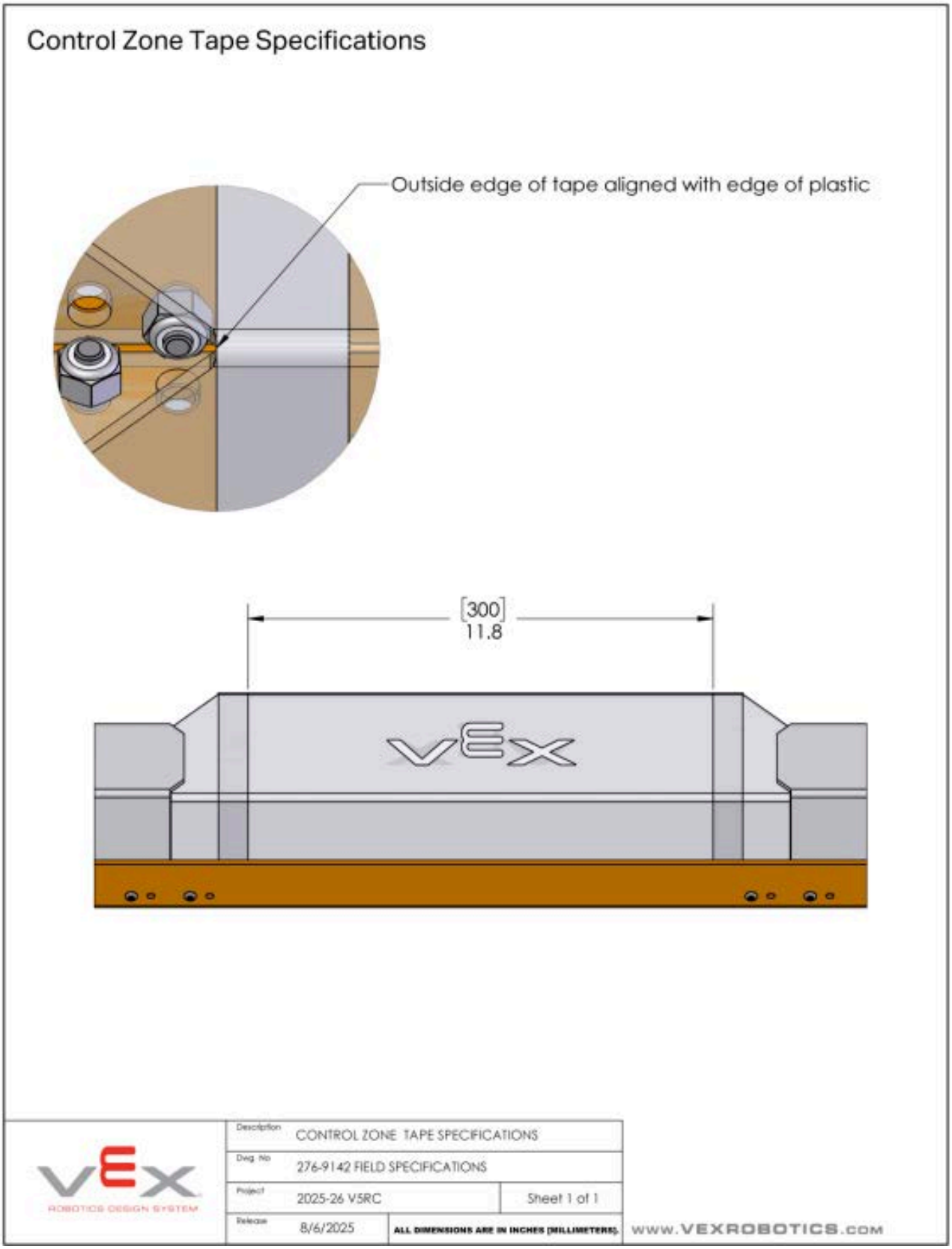
Source: Game Manual

EXACT DIMENSIONS



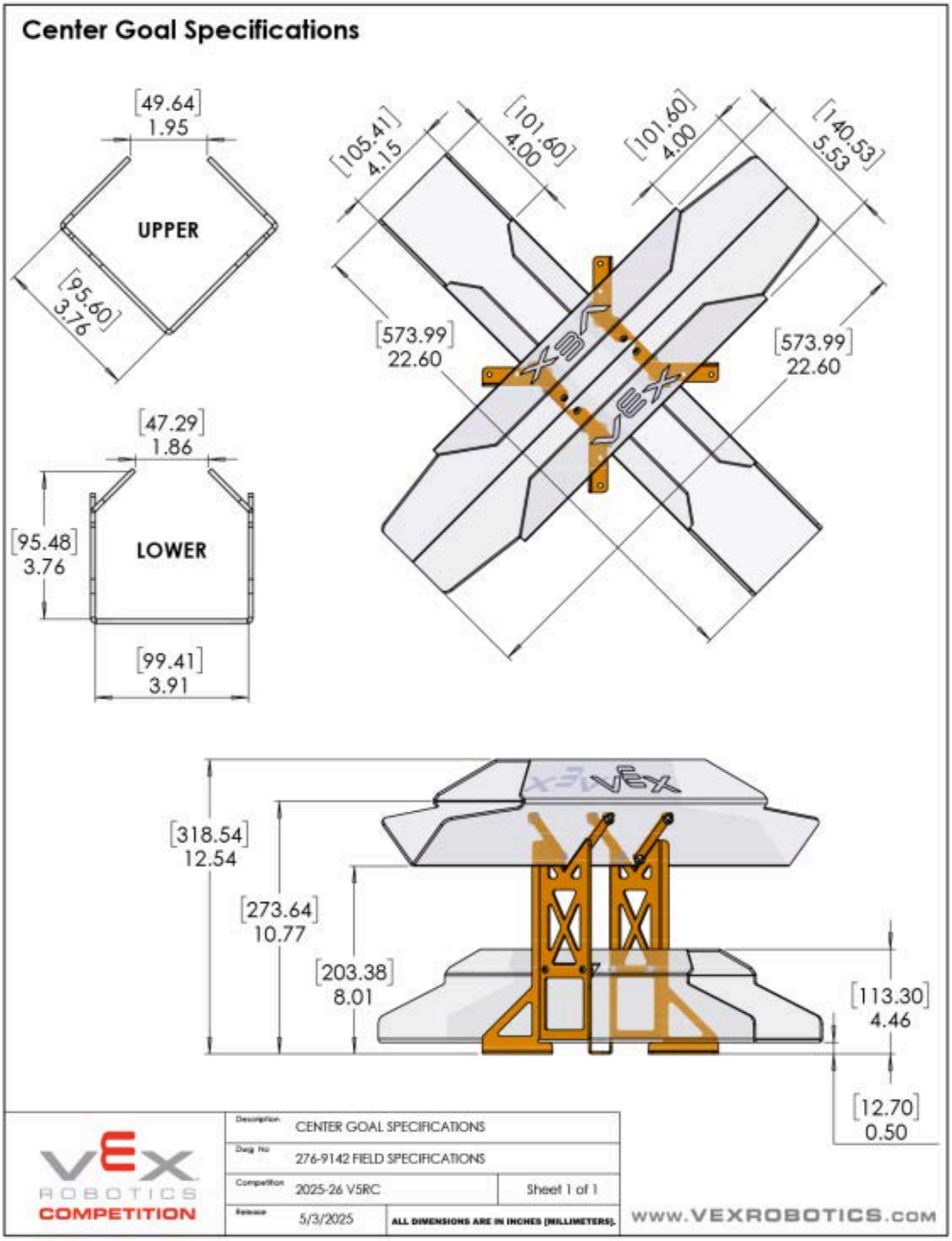
Source: Game Manual

EXACT DIMENSIONS



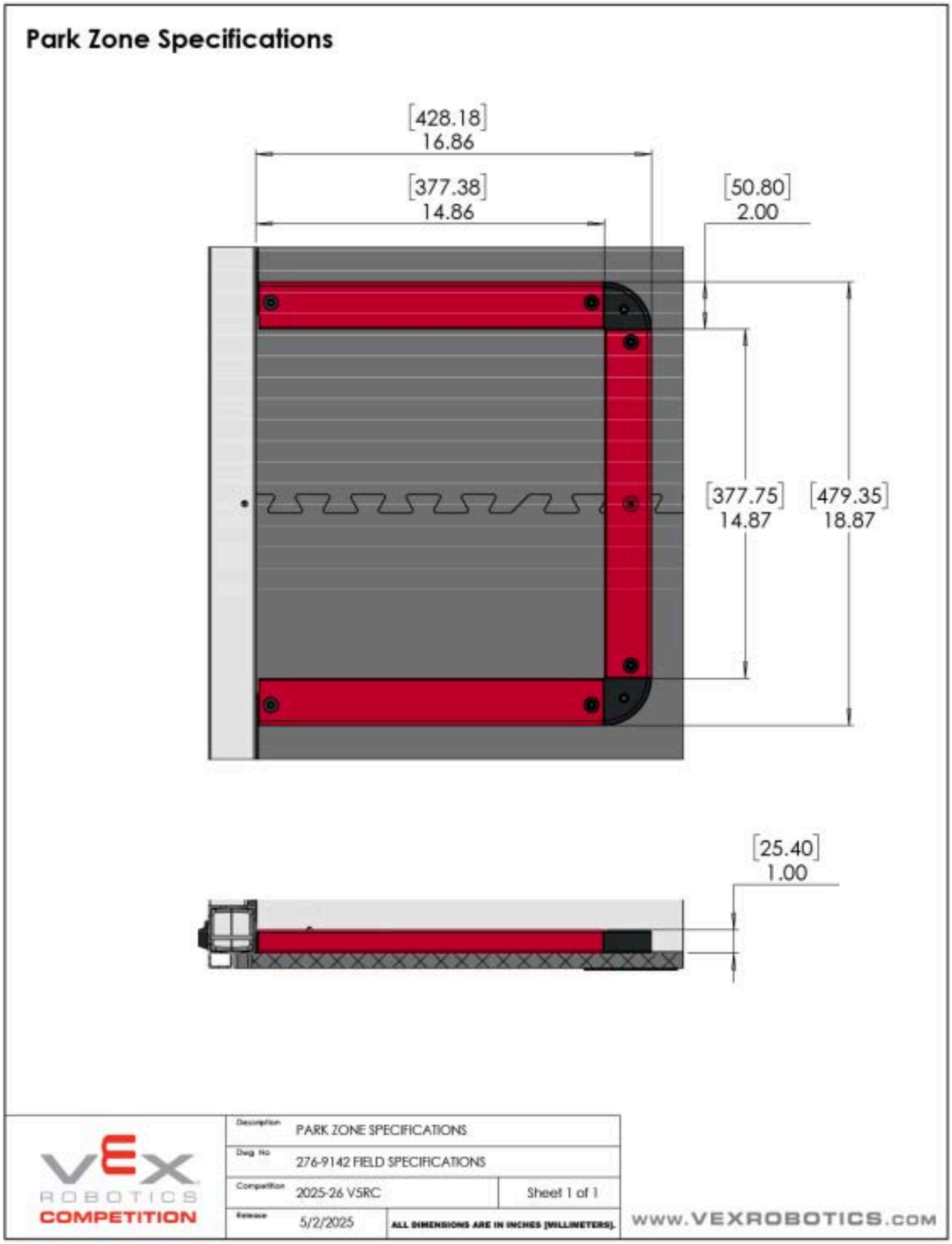
Source: Game Manual

EXACT DIMENSIONS



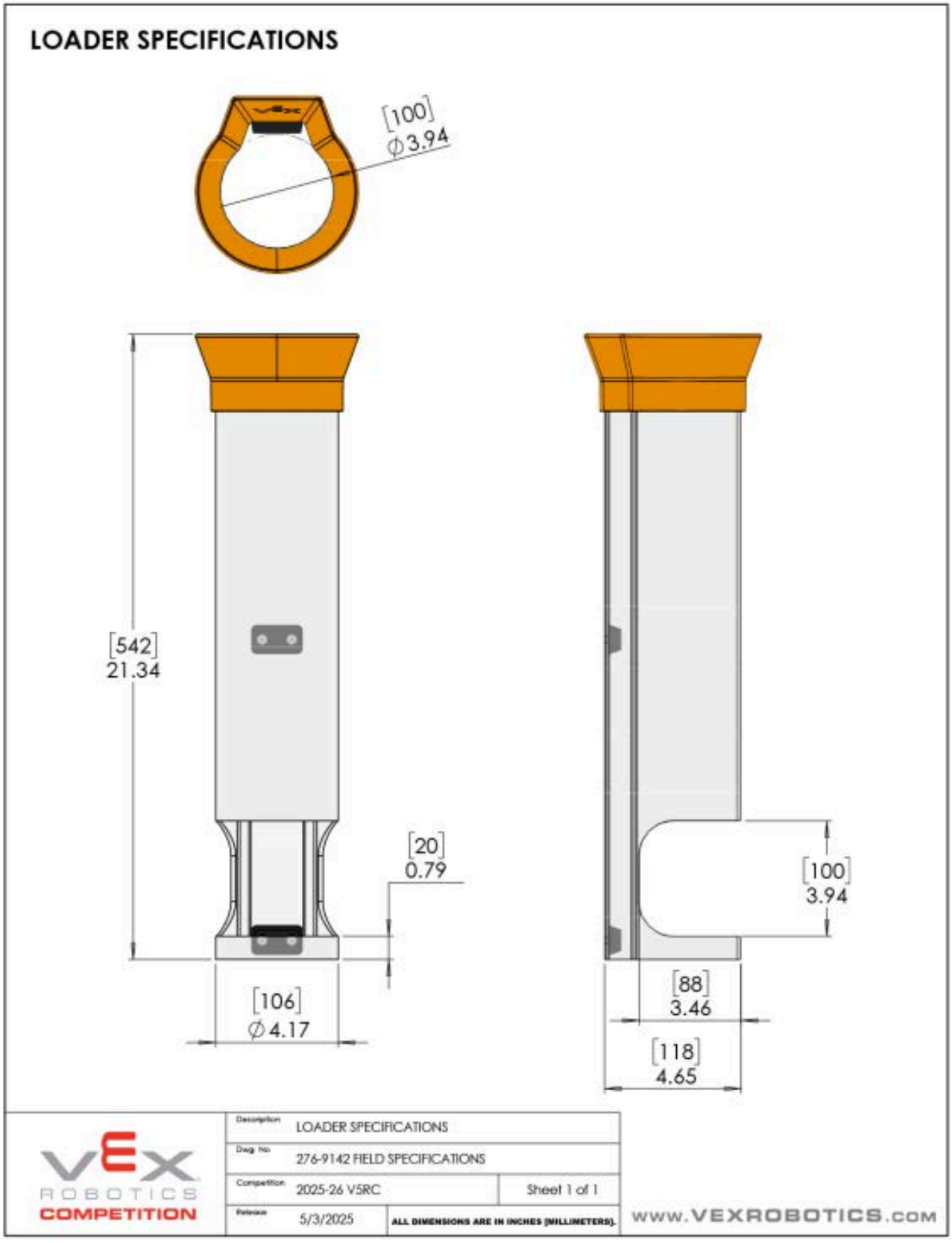
Source: Game Manual

EXACT DIMENSIONS



Source: Game Manual

EXACT DIMENSIONS



Source: Game Manual

ROBOT 1

DRIVETRAIN DESIGN BRIEF

Laksan Mohan | 07/01/2025

PROBLEM STATEMENT

The drivetrain is the core component that enables the robot to navigate the field and interact with game elements. Our team needs to develop a competitive drivetrain that is able to meet all constraints of the game manual. Our drivetrain must be capable of both fast field traversal (high top speed and quick acceleration) and high manoeuvrability (smooth turning) while maintaining the ability to resist and counter pushing from opposing robots. Additionally, the drivetrain must be precise enough to support autonomous code and serve as a compact and modular base to support scoring subsystems and mechanisms.

CONSTRAINTS

- Must fit within an 18” x 18” x 18” volume per R5
  - Can expand to 22" in all directions after the start of the game
- Must fit within our budget of $2000 for the full robot
- Can only be built with VRC-legal components per R17-R26, with some legal modifications per R27 and R28

OBJECTIVES

- Have a competitive top speed (60+ in/sec)
- Can accelerate to top speed within a second
- Can resist pushing from a similar-specced robot
- Can push a 20-pound robot
- Durability and reliability to withstand multiple high-contact matches without failures
- Movements are accurate to within a cm
- Can play both defensive and offensive roles
- Minimize motor drift and gear slop
- Is as compact as possible and serves as a good base for adding scoring subsystems and mechanisms onto
- Weighs under 18 pounds (ideally close to or less than 15 pounds)

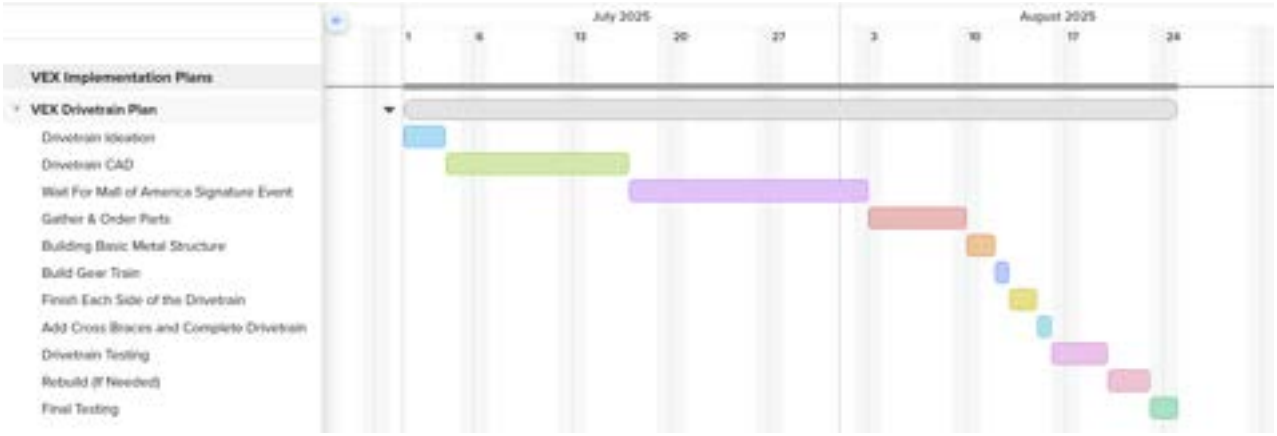
TIMELINE

As we wish to spend considerable time on building and fine-tuning the other mechanisms and making sure the drivetrain will be able to support them, we aim to have the drivetrain completed by the end of July 2025.

Additionally, we also wish to have the design for the other subsystems figured out by then. We will not finalize/build the rest of the robot until the Mall of America (MOA) VEX signature event takes place. This event is the first event of the VEX competitive season and serves as a great opportunity to see how match play affects the viability of certain designs.

By waiting for MOA to happen, we do delay our design process slightly, however, we gain the benefit of being able to refine/confirm our ideas before we fully implement them. As we are a school team with few resources, this ensures we are not wasting time and money.

To guide our design cycle of the drivetrain, we have included a timeline below:



DRIVETRAIN OPTIONS

Laksan Mohan | 07/03/2025

OPTION #1: TANK DRIVE

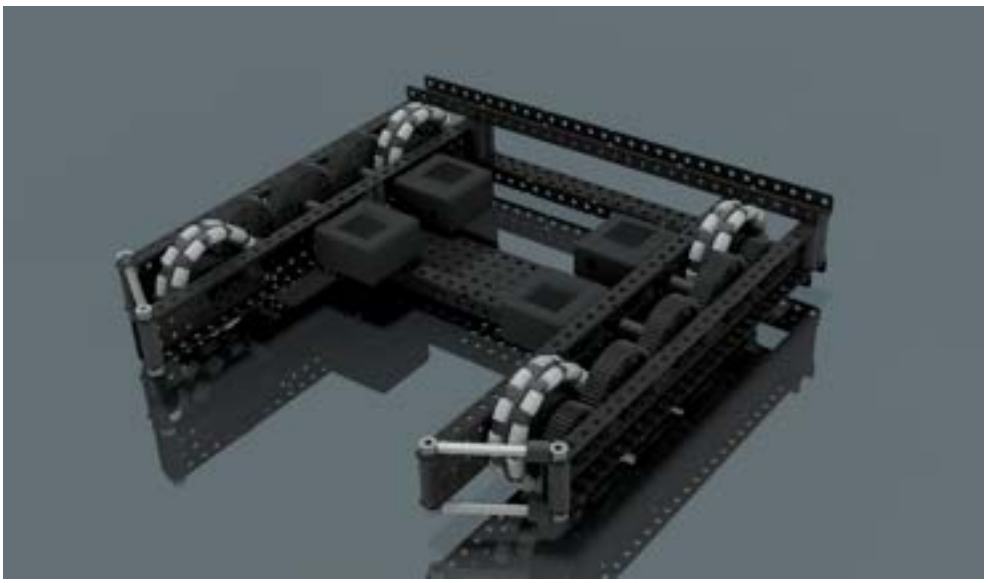


Figure #1: Sample Tank Drive by Xenon27 on the VEX Forum

The tank drive is the most common and simplest drivetrain in VEX. It typically uses 4 or more motors either geared or directly driven to two parallel sets of wheels (or treads) controlled independently. This drivetrain allows for very simple movement, where the robot can turn in place by spinning the two sides in opposite directions or move forward and backward by moving both sides in the same direction.

PROS:

- Simplicity
  - The design is straightforward, reliable and easy to build with basic VEX components
- Compact and Modular Design
  - The layout takes up very minimal space, providing great flexibility in the shape and size of our various subsystems due to the holes available
  - In addition, the layout itself works extremely well with the play style of Push Back, giving a nice open middle section to funnel blocks into and store

- Easy To Gear
  - Because each side of the drivetrain is mounted onto two long straight c-channels, there is a lot of room and flexibility to gear motors for torque or speed
- Simple Movement/Easy To Code
- Good At Going Over Obstacles (Beneficial For Park)
- Fast Turning

CONS:

- Inability To Strafe
  - The greatest flaw of a tank drive
  - Strafing is the ability of a robot to move sideways (left or right) without needing to rotate or change its forward-facing direction.
  - In a traditional drivetrain (like a tank drive), the robot can only move forwards, backwards, and rotate around its axis. Strafing requires a drivetrain that uses special or angled wheels (like mecanum or X-drive) to apply force perpendicular to the direction the wheels are spinning, allowing the robot to slide laterally.
  - Strafing allows robots to move quickly sideways, making them faster, more agile and able to navigate tight spaces or dodge obstacles. Specifically In tasks that require accuracy, such as scoring on a goal or aligning with a match loader, strafing lets the robot fine-tune its position without rotating.
- Easy To Get Pushed
  - This is a result of using Omni wheels, which can be mitigated by using traction wheels but results in less manoeuvrability (inability to drift)

OPTION #2: H-DRIVE

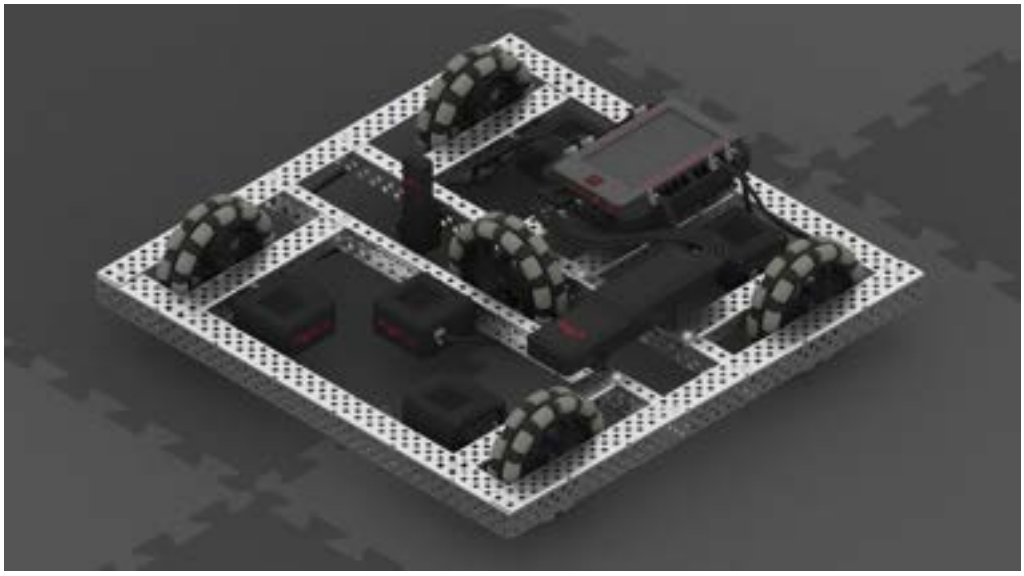


Figure #2: Sample H-Drive by VEX on the VEX Library

The H Drive is similar to a tank drive. The unique feature of an H-drive is the addition of a perpendicular wheel placed in the center of the robot, oriented sideways. This wheel is responsible for strafing. The arrangement of the wheels enables this drivetrain to be omni-directional. H-drives use omni-wheels, which have rollers on their circumference, allowing for smoother movement in any direction.

PROS:

- Able To Strafe
  - The H-drive allows for omnidirectional movement, meaning the robot can move forward, backward, sideways, and diagonally without needing to rotate. This gives it good maneuverability.
- Simpler Design Than Other Holonomic Drives
  - While it still provides omnidirectional movement, it is mechanically simpler than other holonomic drivetrains like mecanum or X-drive, as it only requires a single additional motor and wheel for strafing
- Easy To Gear
  - Because each component of the drivetrain is mounted onto two long straight c-channels, there is a lot of room and flexibility to gear motors for torque or speed
- Easy To Program

CONS:

- Takes Up A Lot Of Space
  - This design feature greatly changes the thought that needs to go into building it, having an extra wheel and motor in the middle takes up a large amount of space and creates more design constraints
- Limited Power On The Strafing Wheel
  - Using a single motor for strafing means it's under a lot of strain from having to push the weight of the robot by itself. This means it can't strafe with very much torque making it quite weak and slow.
- Loss Of A Motor
  - To accommodate this design, you must use an extra motor, taking away a motor from other possible design features (particularly from having a 6-motor drive or having a high rpm for the other various subsystems)
- Easy To Get Pushed
  - This is a result of using Omni wheels, while this issue is less prevalent than in a tank drive, it is still very much an issue
- Poor At Going Over Obstacles(Not Ideal For Park)

NOTE: A design mistake with the H Drive is having the fifth centre wheel on a different level than the other 4 wheels. When this design error happens, either the centre wheel or the drive wheels lifts the other off the ground.

OPTION #3: MECANUM DRIVE

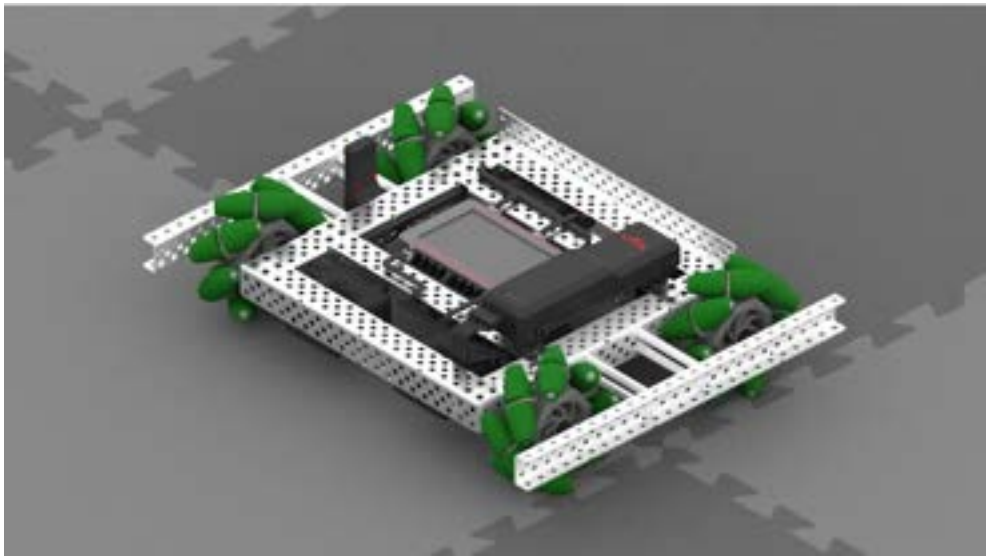


Figure #3: Sample Mecanum Drive by VEX on the VEX Library

The mecanum drive is built like a tank drive, except it uses mecanum wheels, which are specially designed with angled rollers on the circumference that allow the wheels to apply force at a diagonal. By controlling the direction and speed of each wheel, the robot can achieve omnidirectional movement.

PROS:

- Simplicity
  - Unlike more complex holonomic drivetrains (like X-drive), mecanum drivetrains still have a relatively simple structure (four wheels) and don't require special configurations other than using mecanum wheels
- Able To Strafe
  - Mecanum drives provide true holonomic motion, meaning the robot can move in any direction (forward, backward, sideways, diagonally) without turning or rotating. This gives the robot high manoeuvrability, especially in tight spaces or for precise positioning.
- Compact and Modular Design
  - The layout takes up minimal space, providing great flexibility in the shape and size of our various subsystems due to the holes available
  - In addition, the layout itself works extremely well with the play style of Push Back, giving a nice open middle section to funnel blocks into and store
- Good At Going Over Obstacles (Beneficial For Park)
- Very Resistant At Getting Pushed Laterally

CONS:

- Slow and Inability To Gear
  - Mecanum wheels require a good amount of torque to work. Therefore, you are unable to gear them to high levels and will, most likely, have them directly driven by a 200 RPM motor. This makes it extremely slow compared to most competition robots (350-450 RPM).
- Heavier and Wider Wheels
  - Mecanum wheels are heavier, which means the motors will be working harder and struggle to accelerate in comparison to other wheels
  - Mecanum wheels are wider, which leads to larger bases as opposed to the smaller, more agile bases that have become standard
- High Power Consumption and Overheating
  - Because each wheel has to be powered individually and synchronized for different movements, mecanum drives typically require more motors and power than simpler drivetrains like tank or H-drive, reducing battery life and making them prone to overheating.
  - They also experience much more friction than other wheels (regardless of the direction of movement) and therefore lose more energy
- Reduced Pushing Power
  - Mecanum wheels have lower traction compared to standard wheels because the rollers are designed to allow sliding motion. This means the robot is less effective in situations where pushing or pulling heavy loads (like pushing an opponent off a long goal or driving at full capacity) is required.
  - Also, its strafing power is very weak compared to other holonomic drives
- Complex Programming and Control
  - To achieve smooth omnidirectional movement, mecanum drives require advanced programming to synchronize the speed and direction of all four wheels. Proper motor control algorithms (like PID controllers) are often necessary to handle all possible movement combinations smoothly.

Note: A design mistake that can be made with the Mecanum drivetrain is not placing the Mecanum wheels in the correct orientation. When this design error happens, the drivetrain will not move sideways.

OPTION #4: X-DRIVE

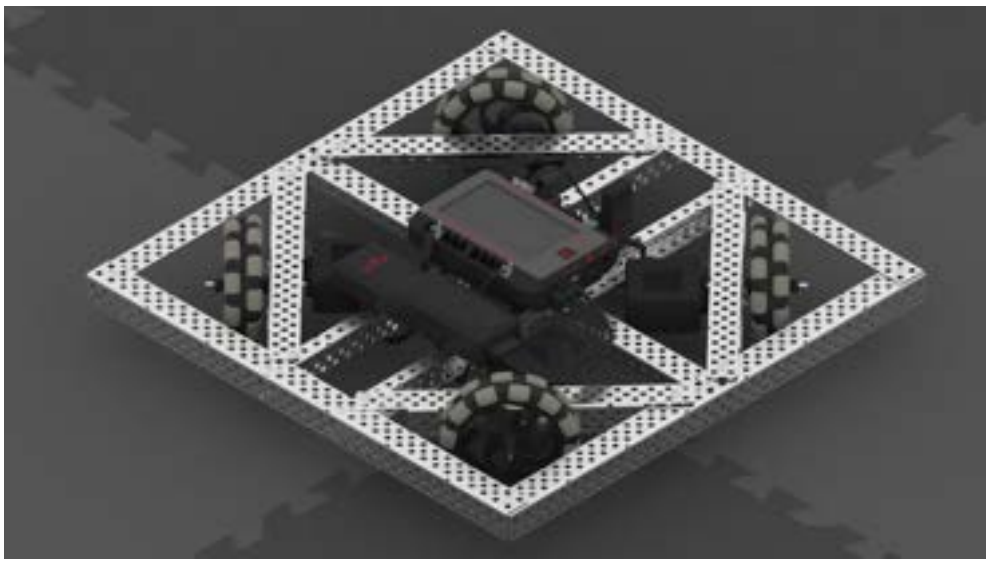


Figure #4: Sample X-Drive by VEX on the VEX Library

The X-Drive is the best holonomic drivetrain. This design can be assembled with four Omni wheels and four motors mounted in an “X” configuration, with each wheel placed at a 45-degree angle to the robot's frame. The layout of the wheels allows the drivetrain to move omnidirectionally by changing which motors are driving and with what speed.

To move forward, you would move each set of diagonal wheels in the same direction so that the net force of all 4 motors will be forward (view figure #5-left). This can be extrapolated to the other 4 cardinal directions for sideways and backward movement. For this to be possible, you must use omni wheels.

To move diagonally, you can move two diagonal motors in the same direction and ignore the other two so that the net force is going in the direction of the two diagonal motors (view figure #5-right).

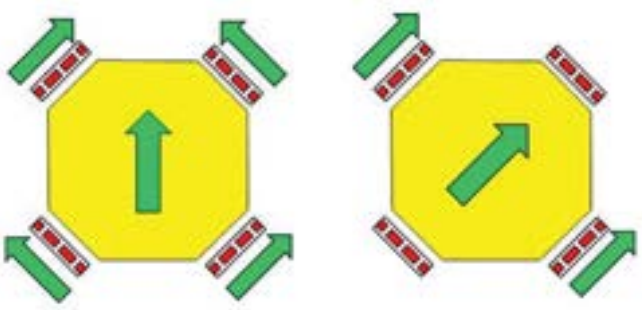
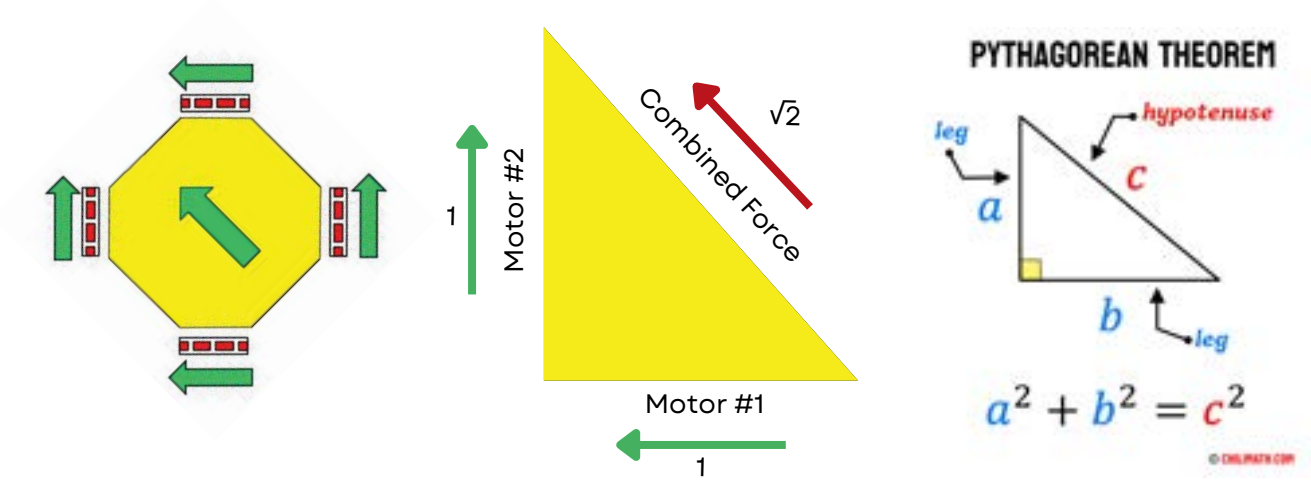


Figure #5: X-Drive Movement Vectors by technik3k on the VEX Forum

X-drives have faster movement than any other drive (assuming they have the same gearing) because the motors push at a 90° angle to each other. This concept works using the Pythagorean Theorem; since the force is going along the hypotenuse, the force of two perpendicular wheels is combined, resulting in a $\sqrt{2}$ (approximately 1.4x) speed increase (<u>note:</u> this consequently results in a loss of torque). This can be visualized in the figures below.



PROS:

- Able To Strafe
  - X-drives provide true holonomic motion, meaning the robot can move in any direction (forward, backward, sideways, diagonally) without turning or rotating. This gives it excellent manoeuvrability and makes it highly versatile during competitions where quick changes in direction are necessary.
  - This manoeuvrability makes it very forgiving for slight mistakes in driving and is very good for aligning with objects and goals
- Very Fast
- Okay At Going Over Obstacles (Beneficial For Park)

CONS:

- Hard To Gear
  - Because the wheel is angled, the only feasible way to gear x-drives is by building vertically and stacking the motor
- Easy To Get Pushed
  - Because X-drives have low torque and can only have 4 motors, they can easily be pushed by other drives. However, with a skilled driver and the drive's extra manoeuvrability, this can be counteracted.
- Hard To Build
  - An X-drive is more complex to construct compared to simpler drivetrains like tank or mecanum drives. Wheels need to be angled 45 degrees, which requires very precise mounting and excellent build quality to ensure smooth and accurate omnidirectional movement. Any slight misalignment can lead to extra friction or inefficiencies.

- Takes Up A Lot Of Space
  - Due to the wheel configuration, an X-drive tends to take up more space than other drivetrains. This limits the amount of space available for other components or mechanisms, especially in tight design constraints.
  - In particular, this is a major complication for this year’s game because an open middle section is necessary for building an intake subsystem. Additionally, the wheels of an x-drive angle towards the middle of the robot, imposing some difficult design constraints.
- Complex Programming
  - To control the robot's movement in all directions, X-drives require more complex programming than traditional drivetrains. The motors must be carefully synchronized to have smooth movement and handle all directional changes.

DRIVETRAIN CHOICE

Laksan Mohan | 07/05/2025

To decide on a drivetrain, we will be using a decision matrix. The following criteria were considered while we decided upon the design of our drivetrain:

- **Speed (X/10)**
  - In Pushback, scoring efficiency is directly tied to how quickly a robot can reach the match loaders, contest goals, and cycle blocks between goals.
  - Speed is also essential for quick offensive plays while avoiding defense.
  - The faster the drivetrain, the higher the score in this category.
- **Manoeuvrability (X/8)**
  - Precise movement and turning ability are key for aligning with the loaders and goals, maneuvering around opponents, and positioning in the park zone
  - High manoeuvrability can give an advantage when battling for position or placing blocks under defensive pressure
  - The more manoeuvrable a drivetrain is, the higher the score
  - Being able to strafe and turn quickly are major factors in this category
- **Match Flexibility (X/5)**
  - Pushback is a highly dynamic game with teams frequently switching between offensive scoring and defensive blocking
  - A flexible drivetrain allows our robot to adapt to either role as the match develops.
  - A drivetrain that balances speed, pushing strength, and precision earns more points here.
- **Ease of Integration (X/7)**
  - Due to the limited amount of space on our robot, we want to ensure that our choice of drivetrain does not restrict our options as we add more components to our robot design.
  - The easier it is to integrate new subsystems, the higher the score.
  - Having an open middle for an intake and space in the back for a long goal aligner is a big factor in this category.
- **Forwards Pushing Power (X/10)**
  - Pushback involves constant contesting of goals and zones, where head-to-head pushing battles often decide control
  - Strong forward pushing power allows our robot to displace opponents, play defense, and maintain control zones under pressure.
  - The greater the force the robot can push with, the higher the score.
- **Side Pushing Resistance (X/10)**
  - When scoring in long goals or protecting control zones, opponents will often attack from the side to disrupt our position
  - A drivetrain with strong lateral stability and traction resists being pushed
  - Stronger resistance earns a higher score.

To help make our decision, specifically in quantitative categories like speed, manoeuvrability, forwards pushing power, side pushing resistance, we need testing data to evaluate different designs. As we have limited time and budget to complete our robot, we were unable to conduct our own testing. Instead, we used data courtesy of Team 12401A and Engineering Girl on YouTube (https://youtu.be/Py14YTHCth0).

TEST #1: SPEED

Methodology:

- Each drivetrain would have a 10-pound weight attached to it.
- The drive would then complete 10 runs in a vex field.
- The robot would start with one edge against the side. When the program was executed, the robot would drive 108 inches, roughly the width of the field, and then stop when a sensor detected a line.
- Then the robot would turn around, be situated, then execute another run.
- The program would record the time from the beginning until the line was sensed.
- New batteries were used for each drive set of 10.
- This method took out the tedious and error-prone measuring of distance and made time as measured internally by the V5 brain the only variable.
- The runs were completed 10 times and then stored on an SD card then downloaded and analyzed.

Test Data:

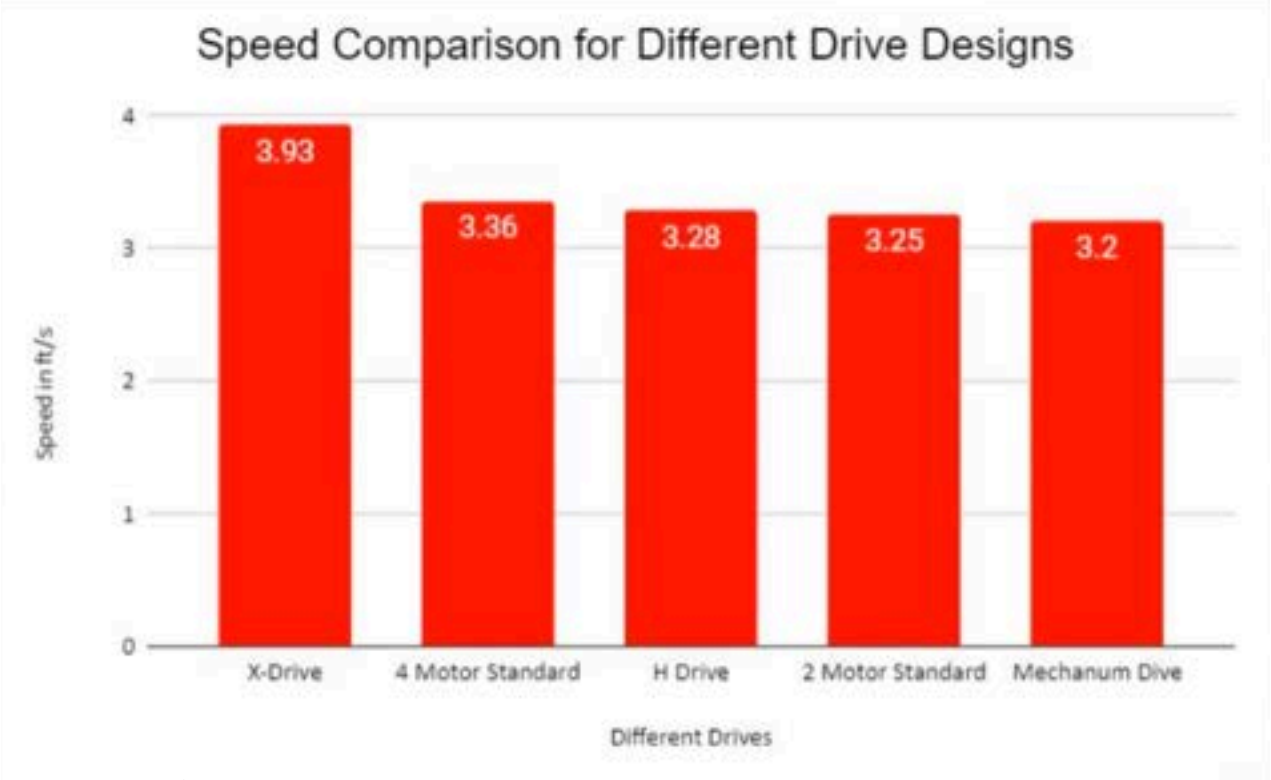


Figure #1: Courtesy of Team 12401A and Engineering Girl on YouTube

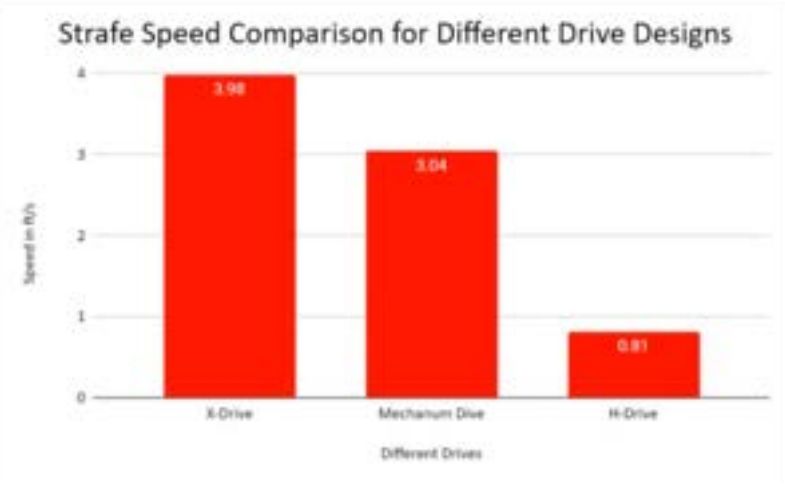


Figure #2: Courtesy of Team 12401A and Engineering Girl on YouTube

TEST #2: FORWARDS FORCE

To determine each drivetrain's ability to withstand head-on pushing, we looked at the data for the pushing force of each drivetrain type.

Methodology:

- A Nextech DFS-XM100 Force Gauge was attached to the field border structure.
- A bar was then rested on the base of the hook and the drive was then run with all motors at 100% for 10 seconds.
- The force gauge recorded the highest reading. 10 readings were taken for each drive design, being careful to recreate the starting position each time.
- New batteries were used for each set of 10 readings.
- All motors had a green cartridge providing factory-stated 1.05 newton-metres of torque.

Test Data:

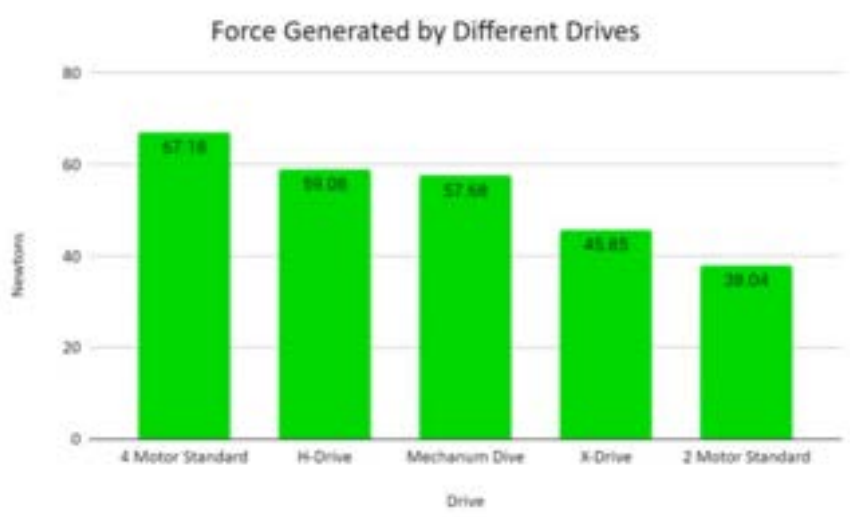


Figure #3: Courtesy of Team 12401A and Engineering Girl on YouTube

TEST #3: SIDEWAYS FORCE RESISTANCE

Using data from Purdue SigBots, each drivetrain (except for H-Drive) was tested for its ability to resist a push from the side, at a 90-degree angle from its forward direction.

Methodology:

- The chassis is placed onto the field
- The force gauge is hooked onto the middle of a chassis, along the left edge
- Force is applied to the end of the scale until the chassis begins to move in a direction perpendicular to its forward direction
- The maximum force is recorded and the force is removed
- The test is reset and reran until 5 tests are completed for each chassis type

Test Data:

	Test 1 (lbs)	Test 2 (lbs)	Test 3 (lbs)	Test 4 (lbs)	Test 5 (lbs)	Average (lbs)
Mecanum Drive	3.87	3.98	3.88	3.60	3.72	3.81
Tank Drive	4.76	4.60	4.81	4.90	4.70	4.75
X-Drive	1.54	1.23	1.39	1.49	1.35	1.40

Figure #4: Courtesy of Purdue SigBots

TEST #4: TURNING SPEED

In order to properly assess manoeuvrability, we looked at the data for the turning speed for each possible drivetrain.

Methodology:

- The test was conducted with a very basic step-turning program using an internal sensor
- The turn would be at 75% power for the first 30 degrees, 25% power for the next 30 degrees, and lastly 5% for the last 30 degrees
- The variables that were recorded were how long it took to turn 90 degrees and how much deviation from the starting point the frame moved in the x and y direction
- To be more specific, the robot turned 90 degrees four times and then recorded the time for each 90-degree turn
- After the four turns, the distance from where one corner of the drivetrain had started to where it ended was measured
- This process was repeated 10 times, so there was an average of the times of 40 90-degree turns and the x and y translation of 10 full 360-degree turns.

Test Data:

Drive	Time (ms)	Drift (cm)	Drift(in)
X-Drive	1683	1.61	0.63
Mechanum Drive	3362	2.95	1.16
H-Drive	1692	3.86	1.52
4 Motor Standard	1824	8.74	3.44

Figure #5: Courtesy of Team 12401A and Engineering Girl on YouTube

Using this data and the aforementioned criteria, we gave scores to each drivetrain solution in the following table:

	Tank Drive	H-Drive	Mecanum Drive	X-Drive
Speed (X/10)	9	8	6	10
Manoeuvrability (X/8)	4	7	6	8
Match Flexibility (X/5)	5	3	3	4
Ease of Integration (X/7)	7	3	5	3
Forward Pushing Power (X/10)	10	5	6	8
Side Pushing Resistance (X/10)	10	6	3	4
Total (X/50)	**45**	32	29	37

After considering each of the four solutions, our team has decided to move forward with a tank drive. This is because, despite its limited ability to strafe, it is fast, simple, easy to gear and program, small to give a lot of flexibility in the design of the other subsystems, has an open mid-section, and has the strongest forward pushing force and the best side resistance (using traction wheels)

DRIVETRAIN SPECS

Laksan Mohan | 07/08/2025

Although we've decided on building a tank drive, our work's not done yet. There are still a lot of decisions that need to be made on the specifics of the design.

DECISION #1: # OF MOTORS

For a tank drive, there are two viable options for the # of motors: a 4-motor drive or a 6-motor drive

4-Motor Drive:
In prior years, 4-motor drives have been a viable and competitive option. However, the current standard in competition is a 6-motor drive.

PROS:

- Leaves Motors for Other Functions
  - With only 4 motors on the drivetrain, we have 4 extra motors available for the different intake stages, putting less design constraints on us.
- Lower Power Consumption
  - Fewer motors mean lower overall power consumption, which can help conserve battery life during matches, allowing consistent performance throughout.
- Simpler to Control
  - A 4-motor drive system is generally easier to control and program.
- Lighter Weight
  - Fewer motors and less associated hardware (gears, wiring, etc.) reduce the overall weight of the robot by a good amount, improving our acceleration and agility.
- Slightly Easier to Build

CONS:

- Less Torque and Power
  - A 4-motor drivetrain provides less torque and power compared to a 6-motor system, meaning it is less effective in defence or in pushing battles with other robots
  - Especially in Push Back, which is a very contact-based game, this is a significant disadvantage.

- Reduced Speed and Acceleration
  - Our robot will most likely be bulky and heavy. With only 4 motors, our robot will be less effective and struggle to accelerate quickly, particularly during pushing matches.
- High Risk Of Overheating Motors
  - With only 4 motors, each motor will have additional stress put on it, resulting in a very high chance of the motors overheating during a match. This is a risk we cannot take and would be catastrophic.

6-Motor Drive:

Pros:

- Increased Power and Torque
  - With 6 motors, our drivetrain has significantly more power and torque, which helps us win pushing battles and allows the robot to handle heavy mechanisms and carry a lot of balls without losing speed.
- Faster Acceleration and Consistent Speed
  - More motors on the drivetrain mean faster acceleration, even with heavy mechanisms on board. Thus, we can maintain high speeds more easily when traversing the field or performing quick manoeuvres.
- Redundancy and Reliability
  - With 6 motors, even if one motor experiences issues during a match, the drivetrain can continue functioning with minimal performance loss. This adds reliability in matches where every component is critical.
- Lower Risk of Overheating
  - With 6 motors, each one has significantly less strain put on it, meaning there is a lower risk of overheating.

Cons:

- Less Available Motors for Other Mechanisms
  - By dedicating 6 motors to the drivetrain, we give ourselves a significant design constraint: we can only use a total of 2 motors for the rest of the mechanisms.
- Higher Power Consumption
  - A 6-motor drive system consumes more power, which leads to heavy battery drain, especially in back-to-back matches.
- More Weight
  - Additional motors add more weight to the robot, which could make the robot less agile, especially if not managed carefully.
- Slightly harder to build

Ultimately, we decided to go for a **6-motor drive** because we prioritized power, torque, and performance on the field. Push Back gameplay heavily relies on strong pushing power to contest goals and block zones, as well as the ability to resist being displaced by opposing robots. While 4 motors would be significantly easier, our design would be fundamentally weaker compared to other robots and, thus, fundamentally flawed. To overcome the motor constraint, we plan to hook up two of the intake stages to one motor via a gearbox or chains, use two 5.5W motors for the other stages, and use pneumatics for the other mechanisms.

DECISION #2: WHEEL TYPE AND

For a tank drive, there are two viable options for the wheel types: Omni and Traction wheels

Figure #1: Omni Wheels from VEX

Omni Wheels:

PROS:

- Omnidirectional Movement
  - Omni wheels have rollers around the circumference, allowing them to roll freely sideways while still providing forward and backward traction. This makes the robot more manoeuvrable.
- Faster, Smoother Turns
  - Omni wheels allow for smoother and faster turns since they have low lateral friction. Thus, the robot can pivot more easily, which improves overall agility and responsiveness on the field.

CONS:

- Wear and Tear
  - Omni wheels tend to wear out faster due to the extra stress on their rollers, especially in high-contact matches.

- Less Traction and Grip
  - Omni wheels provide less forward traction compared to traction wheels because the rollers reduce the contact surface. This, in addition to the fact omni wheels slide laterally, makes them less effective in pushing matches, and the robot is more likely to get pushed around by opponents.

Figure #2: Traction Wheels from VEX

Traction Wheels:

PROS:

- High Traction and Grip
  - Traction wheels are designed with a rubberized surface that provides excellent grip on the field. This makes them ideal for pushing matches and resisting defence from other robots.
- Durability
  - Traction wheels are generally built to be sturdy and can handle a lot of wear and tear, making them durable over time in high-contact environments.

CONS:

- Reduced Maneuverability and Slower Turns
  - Compared to omni wheels, traction wheels make the robot less agile, particularly when turning. Since they grip the ground tightly, traction wheels cause the robot to turn slower or have difficulty making fast, sharp turns, which can be a major disadvantage in fast-paced matches where agility is key.

After considering both options, we decided to combine traction and omni wheels to create a balance between maneuverability and power. Our drivetrain will use an **8-wheel hybrid setup** (**4 omni wheels** at the corners, **2 traction wheels** in the middle, and an additional **2" flex wheel** on each side). The flex wheels fill the gap that would exist in a 6-wheel setup, helping the robot maintain constant ground contact and improving stability when crossing the park barrier. They also protect the drivetrain's gears from scraping against the park. This setup can offer the best of both worlds.

DECISION #3: ANTISTATIC OR OG WHEELS

In 2022, VEX released antistatic (abbreviated as AS) versions of all their wheels. Anti-static wheels have a high amount of traction and, unlike their predecessors, have a uniform width and tire shape. Although they are wider than the original wheels, this is actually an advantage rather than a detriment since the larger tire surface area increases the odds that the wheel will make contact with the field tiles, allowing the tire's traction to have a larger effect on the wheel's overall performance. Additionally, the width only comes out to be 1 inch, a compact wheel in comparison to the various wheels VEX Robotics has manufactured. Conveniently, the hubs of the wheels come with many mounting holes. Most importantly, these wheels have tires with special properties that prevent the accumulation of static, which decrease the odds of damage to electronics due to electrostatic discharge. For these advantages, we will be using **antistatic** versions of all wheels in our design.

DECISION #4: WHEEL SIZE

For each wheel type, VEX offers 3 sizes: 2.75” diameter, 3.25" diameter, 4" diameter. In deciding the type of wheels, we need to go over some key considerations:

- Control
  - Slop refers to the unintended movement, wiggle or “play” that occurs between mechanical components in, this case, the drivetrain. For us, slop is a result of gear backlash and internal motor components. Backlash is the space between the teeth of gears in a gearbox or drivetrain. When the motor changes direction or when torque is applied, the gears have to “catch up” to engage fully, resulting in a slight delay or imprecision in movement. Inside a motor, small gaps between internal components like gears, shafts, and bearings can contribute to slop. Over time, wear and tear exacerbate these gaps, leading to greater slop.
  - Slop is exacerbated the bigger the wheels are because the same percentage of slop is being applied, just over a greater circumference. For example, if the slop of a gear train is 15%, this results in 0.4125” of slop in 2.75” wheels and 0.6" of slop in 3" wheels. While this doesn’t seem like a lot, for precise autonomous routines (particularly for those using odometry), this inaccuracy could be extremely detrimental.
  - Thus, smaller wheels provide finer control over movement while larger wheels are more inaccurate.

- Speed
  - Larger wheels cover more distance per rotation because they have a greater circumference. For example, a drivetrain spinning at 600 RPM can travel 2400" with 4" inch wheels, as opposed to only 1650" with 2.75" wheels. As a result, larger wheels are faster and transverse the field more quickly, a key asset in Push Back.
- Stability
  - Smaller wheels have a lower profile, which can lower the robot's centre of gravity and improve stability.

Ultimately, we decided to choose **3.25" wheels** because they provide a middle ground, offering sufficient speed for moving across the field while maintaining better control compared to the larger 4" wheels. This balance makes the robot agile and responsive, making 3.25" wheels the perfect choice for a competitive, well-balanced robot design. In addition, there will be several low-hanging components on the drivetrain. With 2.75" wheels, these components would drag against the ground as the robot is moving and going over bumps, causing the robot to be less controllable.

DECISION #5: GEAR RATIO

When choosing the gear ratio for a drivetrain, there are two main factors to consider: the gear ratio's rotational speed (RPM) and torque. RPM and torque have an inverse relationship (when RPM is increased, you will lose torque). Ultimately, you need to find a middle ground that is fast enough to traverse the field and play defense while having strong torque for pushing battles. After testing and discussion, we identified 400 RPM and 450 RPM as the two most viable options. We are not considering any lower gear ratios because speed is a crucial advantage in this game. A faster drivetrain allows us to reach the goals faster, react quickly to shifting play, and apply controlled pushing power when needed. While lower RPM gear ratios would still be decently competitive and fast, the lack of speed would be a fundamental flaw in the design that would likely need to be fixed later on.

450 RPM Gear Ratio (76.6 in/sec):

To achieve this speed, the driving wheels (coloured blue in the diagram below) need to be powered by a 600 RPM (blue) motor cartridge geared down to 450 RPM using a 36:48 gear ratio. A 450 RPM drivetrain provides a higher top speed, allowing the robot to travel faster across the field. However, the higher RPM comes with trade-offs in terms of torque. At 450 RPM, the motors exert less torque, meaning the robot may struggle when pushing other robots or accelerating quickly. A key consideration is the controllability of the drivetrain. 450 RPM needs to be built very well and driven very skillfully to be viable.

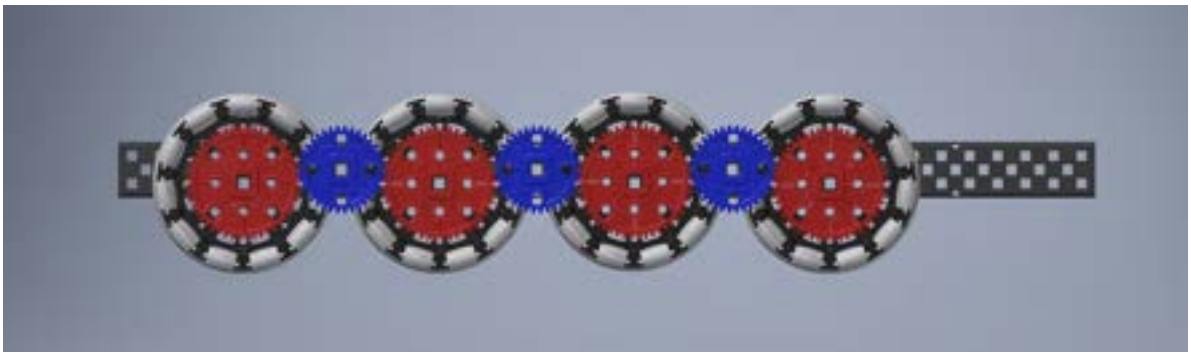


Figure #3: Sample 450 RPM Drivetrain layout by Xenon27 on the Vex Forum

400 RPM Gear Ratio (68.1 in/sec):

To achieve this speed, the driving wheels (coloured blue in the diagram below) need to be powered by a 600 RPM (blue) motor cartridge geared down to 400 RPM using a 48:72 gear ratio. The 400 RPM gear ratio offers a more balanced approach between speed and torque. While slightly slower than the 450 RPM configuration, 400 RPM still provides a high enough speed to traverse the field efficiently without being noticeably slower than competitors. The main advantage of the 400 RPM drivetrain is the increased torque compared to 450 RPM. This added torque improves the robot's ability to push other robots, handle obstacles, and accelerate more quickly. Most importantly, the lower speed makes it easier to control for our driver.

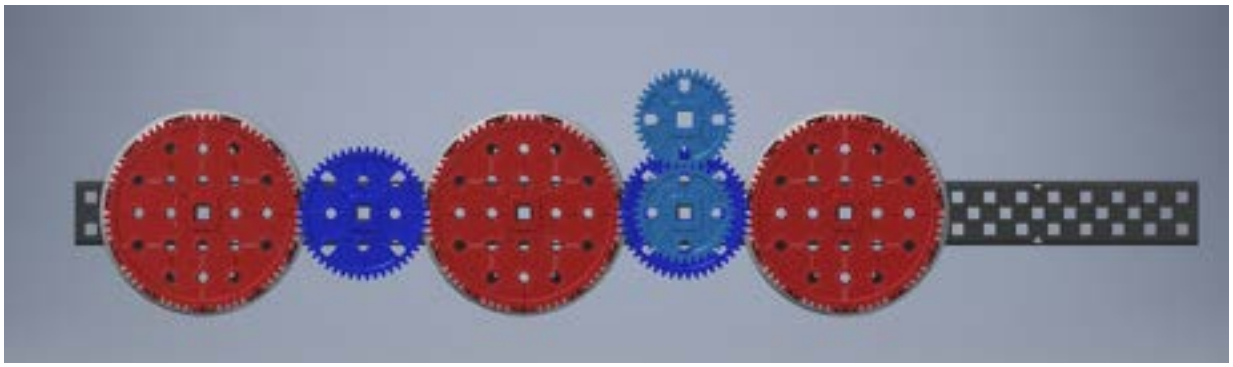


Figure #4: Sample 400 RPM Drivetrain layout by Xenon27 on the Vex Forum

Ultimately, we decided on the **450 RPM gear ratio** because speed is such a crucial factor in this game. Without speed, your design is fundamentally flawed. As this is our last year, we do not want to take this risk. The reason we settled on 450 RPM and nothing higher is because it would simply be too fast. On 3.25" wheels, 450 RPM is the highest ratio that is still viable. Anything over 450 RPM would burn out mid-match and be borederline uncontrollable. This decision does come with some sacrifices, however. In exhange for the speed, we have a slightly worse torque and a faster rate of overheating. Yet, with excellent build quality, these tradeoffs are worthwhile. Additionally, we ran a 400 RPM drivetrain last year and we noticed that the robot would sink into the mat and, due to the large size of the gears, pieces of the mat would end up in the gears and it was a pain to clean.

DECISION #6: SIZE OF THE DRIVETRAIN

In general, having a larger drivetrain is more beneficial as it gives:

- Increased Stability
  - A larger footprint increases the support base, making the robot more stable and reducing the risk of tipping over during sharp turns or rapid movements. This is especially helpful with our robot as it is very fast.
- More Space for Components
  - This is ideal as the intake will span the entire robot.
- Defensive Coverage
  - Defence will be a key element of Push Back gameplay. Thus, a larger drivetrain can help cover more space, blocking our opponents from reaching goals.

The starting size limit of a V5RC robot for Push Back is 18" by 18" by 18". This means the maximum size of our drivetrain can be 35 holes by 35 holes (17.5” by 17.5”). However, from our experience last year, you do not want the drivetrain to be at the max size. With a 35 hole by 30 hole drivetrain, we ran into issues with being too big and having trouble manoeuvring, especially in the corners. Another issue with a large base is the additional weight since every cross brace needs to be longer. Additionally, last year’s drivetrain limited the size of our other mechanisms and, as we want to be able to expand for a descore mech and a matchload mech, we must make the drivetrain as compact as possible while still being easy to build subsystems onto. When deciding on the smallest width we can do, we need to factor in the size of the air tank. To keep our center of gravity low, we want to place our air tank in between the two sides of the drivetrain. This is a lesson we learned last year, since we had our air tanks mounted high, which lead to the robot jerking up a lot. To fit the air tank, the drivetrain must be 27 holes long (13.5"). While we could make the width that, it is a bit harder of a number to fit all the components into. Therefore, we decided to go with a width of 30 holes (15"). To keep the base stable, we went for a square size. Hence, our chosen drivetrain size of **30 holes by 30 holes (15" by 15")**. This gives us a good 3" on each side to work with while having a nice big base to add other components onto. In addition, the wheelbase is 23 holes (11.5"), giving us a good buffer on each side to add standoff bracing.

DECISION #7: BUILD MATERIAL

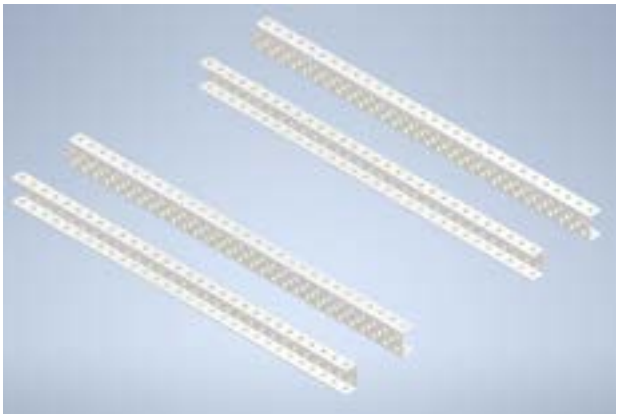
For our main structural elements, we chose to use C-channels as the primary building components. C-channels provide an excellent balance of strength, rigidity, and versatility, making them ideal for building a stable yet lightweight frame. Their open design allows for easy mounting of different

subsystems, while maintaining enough strength to handle impacts during pushing battles.

We decided to construct the robot entirely out of aluminum rather than steel. Aluminum offers is much lighter than steel (about 3x). Weight is a important considertion when building the robot as it allows for faster acceleration and more agile movement on the field. Since Pushback is a fast-paced, contact-heavy game that requires quick transitions between offense and defense, reducing weight improves both speed and efficiency. However, aluminum is easier to dent and bend. So, our design must fortify the parts of the robot that will be face the most impactsduring a match (the front and ends of the drivetrain c-channels, we will elaborate on our solution for this later). Additionally, aluminum is easier to machine, drill, and modify, which helps streamline our build process and allows for easier adjustments during testing.

DECISION #8: C-CHANNEL LAYOUT

When constructing the drivetrain, there are two main layout options for how the C-channels are oriented. As shown in the image blow, the inner C-channels should always face inward toward the center of the robot. This is better for mounting the motors and ensures that the drivetrain remains rigid and properly aligned.



C-Channel Configuration #1

C-Channel Configuration #2

For the outer C-channels, teams typically choose to face them either inward,]][ ]], or outward,][ ][. Inward-facing outer c-channels helps protect and cover the drivetrain axles, keeping them shielded from impacts and reducing the risk of field elements or other robots interfering with them. However, it requires careful planning and design, as accessing or modifying the drivetrain later is more difficult and requires partial or complete disassembly. Outward-facing outer c-channels is what most teams typically do. This layout exposes the axles, which makes repairs and adjustments easier throughout the season but leaves the drivetrain more vulnerable to collisions and contact damage.

After evaluating both options, we chose to have all C-channels facing inward. This configuration maximizes axle protection, structural integrity, and drivetrain durability, which are essential for Pushback's high-contact and defense-oriented gameplay. While it requires more forethought during assembly, the added protection and stability make it the optimal choice for our robot's long-term performance.

DECISION #9: VERTICAL MOTORS

For our drivetrain, we decided to mount the drive motors vertically instead of horizontally. This configuration was chosen to optimize internal space and improve subsystem integration. By mounting the motors vertically, we are able to save valuable space in the front of the robot, allowing us to design a lower and more efficient intake ramp that sits closer to the ground. This is especially important in Push Back, where fast and consistent block collection can make the difference in a match. Additionally, because the intake ramp naturally becomes steeper near the back of the robot, the vertical motors fit neatly within that slope, complementing the overall design instead of interfering with it.

To achieve this setup, we used 3-hole-wide c-channels for inner c-channels of the drivetrain. Unlike 2-wide c-channels, these do not have a center row of pre-drilled holes, so we needed to custom drill motor mounting holes to achieve the correct alignment. This process requires a high level of precision, as even a small misalignment can introduce friction or cause binding in the drivetrain. However, since these holes are drilled to exact size and tolerance, they provide a snug fit for the screws, which eliminates the need for bearing flats at the motor mounting points. This reduces unnecessary hardware, saves weight, and ensures a cleaner, more compact drivetrain assembly.

DECISION #10: SCREW-JOINTED WHEELS

For our drivetrain, we decided to use screw joints instead of mounting the wheels on drive shafts. During testing, we observed that the VEX brass circle inserts have tight tolerances when fitted onto screws, allowing for smooth, low-friction rotation. Compared to using shafts and bearing flats, this method significantly reduces play and friction in the drivetrain while also simplifying assembly and maintenance.

In our design, each non-powered wheel is directly screw-jointed by attaching a gear to the wheel, inserting brass inserts into the wheel holes, and running a screw through the assembly instead of a shaft. The screw acts as the rotational axle, while the brass inserts serve as the bearing surface.

This technique, known as screw jointing, allows for precise, low-friction movement and ensures consistent spacing between drivetrain components.

Additionally, we are using Keps nuts tightened against the faces of the C-channels to help lock in the spacing and create a boxed structure. This adds rigidity to the drivetrain, preventing the sides from flexing under load or during pushing battles. The result is a more durable and consistent drivetrain that maintains its alignment and performs reliably throughout the season.

DECISION #11: DROP-CENTERED WHEEL

To improve sideways pushing resistance and traction in our drivetrain, we designed our middle traction wheel to be dropped-center. This means that the middle wheel on each side of the drivetrain sits slightly lower than the front and back wheels, in our case, by approximately 1/16 inch. We achieved this by creating a custom polycarbonate gusset that offsets the mounting point of the center wheel.

On a hard surface, this configuration can make the drivetrain feel slightly “wobbly,” as the robot primarily balances on the center wheels. However, on the foam field tiles used in VEX, this effect is minimized. The foam compresses under the robot’s weight, allowing all six wheels to maintain consistent contact with the ground.

Importantly, the compression of the foam field surface gives the center wheels increased traction and stability, improving our robot’s ability to resist being pushed from the side. This is crucial in Push Back, as scoring and maintaining control of the long goals often depends on winning pushing battles.

DECISION #12: HS AXLE BRACING

In our drivetrain design, we chose to replace traditional C-channel cross braces with a custom high-strength (HS) axle brace. Instead of using bulky aluminum C-channels for cross-support, we are taking a steel HS axle and custom-drilling mounting holes into it, allowing it to function as a rigid, low-profile brace connecting both sides of the drivetrain.

This approach offers several key advantages. The HS axle is extremely strong and resistant to bending, providing excellent structural rigidity under load, especially during pushing battles or sudden impacts, which are common in Push Back. Because the axle has a smaller profile than a C-channel, it also saves internal space and keeps the drivetrain more compact,

allowing additional room for subsystems such as the intake or ramp. The reduced profile also helps simplify cable routing and keeps the overall structure cleaner.

However, this design does come with a trade-off: weight. HS axles are made of steel and are significantly heavier than aluminum C-channels. This means we must carefully manage the overall robot weight to stay within reasonable limits and maintain optimal mobility. Despite this, the strength, stiffness, and compact nature of the custom HS axle brace make it a worthwhile trade-off, providing a durable and efficient alternative to traditional cross bracing for our Push Back drivetrain.

DECISION #13: FRONT FUNNELS

The front of our drivetrain is designed with a funnel-shaped layout to improve alignment, block control, and overall driving efficiency. To achieve this, the inner c-channels are intentionally made shorter, creating a triangular taper at each front corner of the drivetrain. This design effectively forms a funnel that guides game elements toward the center of the robot while preventing blocks or field elements from catching on the edges during gameplay.

To ensure the c-channels remain rigid and don't bend inward under stress, we are reinforcing it with standoffs connecting the shortened inner C-channels to the outer frame. These standoffs maintain structural spacing and prevent deformation during pushing or collisions, ensuring that the wheels and drivetrain remain unobstructed. This also prevents debris or blocks from getting jammed between the wheels and the frame, maintaining smooth operation during matches. The design will be similar to the concept shown in the image below, but with the funnel shape flipped inward toward the robot's center.

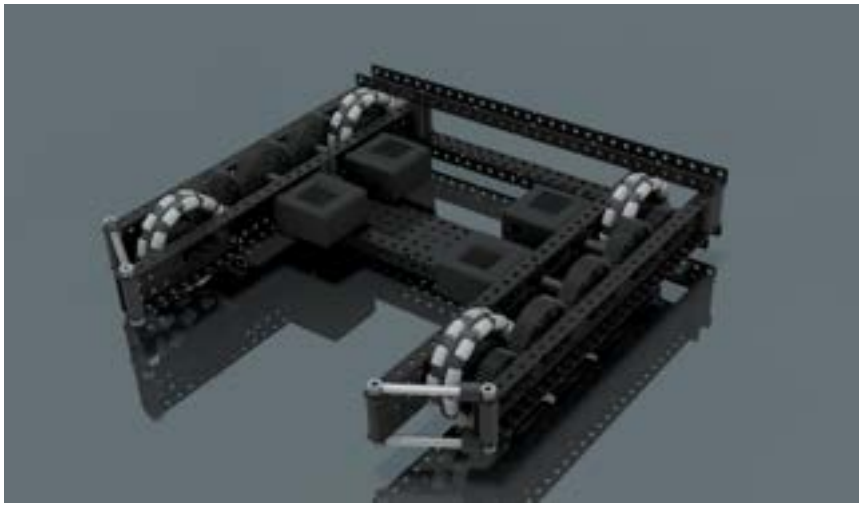


Figure #5: Sample Tank Drive Funnels by Xenon27 on the VEX Forum

SUMMARY OF DESIGN CHOICES

As a summary, the specs of our drivetrain are as follows:

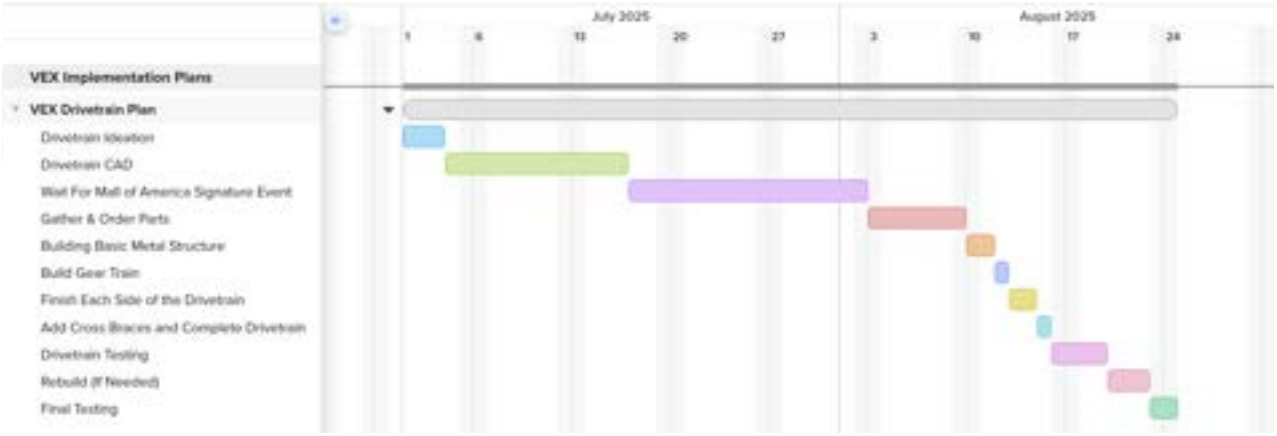
- 6 Motor Tank Drive
- 8 Screw-Jointed Anti-Static 3.25" Wheels
  - 4 Omni Wheels
  - 2 Traction Wheels
    - Drop Centered 1/16"
  - 2 2' Flex Wheels 30A
- 450 RPM Gear Ratio (76.6 in/sec)
- Size: 30 Holes by 30 Holes (15" by 15")
- Fully made of Aluminum C-Channels
- C-Channel Configuration #2: [[ ]]
- Vertically Mounted Motors
- HS Axle Braced
- Tapered Funnel Front

DRIVETRAIN IMPLEMENTATION PLAN

Laksan Mohan | 07/09/2025

As we wish to spend considerable time on building and fine-tuning the other mechanisms and making sure the drivetrain will be able to support them, we aim to have the drivetrain CAD completed by the end of July and have a finalized design built by the end of August.

To guide our construction of the drivetrain, we have included a timeline below:



PREROLLER DESIGN BRIEF

Laksan Mohan | 07/10/2025

The intake system is divided into three stages: the first stage (prerollers), the second stage, and the third stage. Prerollers are the initial contact point between the robot and the game elements, responsible for guiding and pulling objects from the field into the main intake system. Since they are required for every type of intake configuration, our first design priority is to develop an effective preroller setup before finalizing the remaining intake stages.

PROBLEM STATEMENT

The prerollers must be able to reliably collect and center game elements, minimize jamming, and maintain smooth feeding into the next stage. They should be low to the ground for optimal pickup, durable under repeated impact, and compact to fit within design and size constraints. A well-designed preroller system will ensure consistent, efficient performance throughout both autonomous and driver control periods, forming the foundation of a competitive intake system.

CONSTRAINTS

- Must fit within an 18” x 18” x 18” volume per R5
  - Can expand to 22" in all directions after the start of the game
- Must fit within our budget of $2000 for the full robot
- Can only be built with VRC-legal components per R17-R26, with some legal modifications per R27 and R28

OBJECTIVES

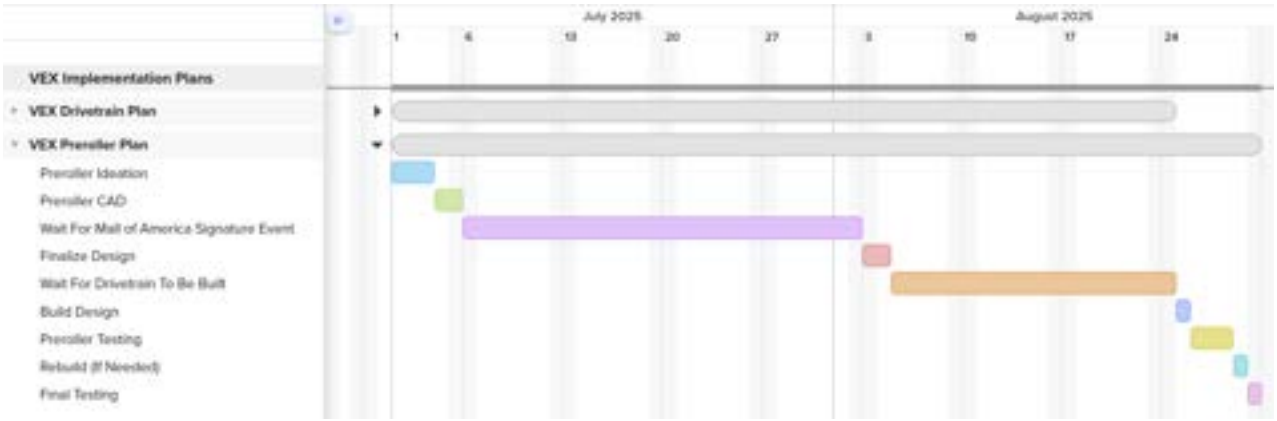
- Capable of intaking blocks at a minimum speed of 25+ in/sec.
- Can reliably intake blocks from various angles and orientations.
- Maintains a secure grip to prevent drops during rapid movement, turning, or collisions
- Designed to minimize jams and ensure smooth, uninterrupted operation
- Must be strong enough to withstand collisions and ramming at the front of the robot
- Can operate autonomously with consistent intake accuracy

- Compact design to integrate smoothly with other scoring mechanisms
- Weighs under a pound to maintain overall robot maneuverability and speed
- Adaptable to handle variations in game element placement

TIMELINE

We will be waiting until the Mall of America Signature Event is over to finalize the design and build the preroller. Until then, we will brainstorm ideas for, design, and CAD the preroller. Once Mall of America is over, assembly will be quick and final testing will take up the remainder of the time, as we have to tune the assembly to provide the right amount of friction and align to the right height.

To guide our design cycle of the preroller, we have included a timeline below:



PREROLLER OPTIONS

Laksan Mohan | 07/11/2025

For this game, there are only 2 viable types of prerollers: top-down prerollers and siide rollers.

OPTION #1: TOP-DOWN PREROLLER

The top-down preroller is designed to grip and collect blocks from above, pulling them into the robot's control for transfer into the scoring mechanism or storage area. This preroller uses rotating rollers positioned along the top of the robot's front to pull blocks inward as the robot approaches them.

Compression in this preroller can be adjusted by changing the roller material and the distance between the roller and the ground. However, the top-down roller is not suited for high-torque or high-compression setups. Since the rollers only contact the game object at a single point (from the top), excessive compression can lead to jamming between the roller and the field surface. Side roller prerollers handle these high-compression situations more effectively due to multiple contact points that better control and guide the block.

PROS:

- Simplicity
  - The design is straightforward, reliable, and easy to build with standard VEX components
  - It minimizes the number of moving parts, making it easier to maintain and tune during competitions
- Compact and Modular Design
  - Layout takes up very minimal space, providing great flexibility in the shape and size of our various subsystems due to the holes available

    - In addition, the layout itself works extremely well with the play style of Pushback, giving a nice open middle section to funnel blocks into and up onto a goal
- Consistent Block Control
    - Because the rollers make contact from above, this system can effectively control the block and maintain consistent compression, helping prevent jams or double prerollers
- Wide Fielding Range
    - These prerollers typically span most or all of the robot's front width, allowing for a broad collection area that can quickly acquire multiple blocks from the field

CONS:

- Limited Maneuverability
    - The preroller requires the robot to approach blocks head-on, limiting its ability to collect game elements from side angles or while strafing (if using an omni or X-drive)
    - This can slow cycle times when repositioning between blocks
- More Space Required
    - The top-down approach generally takes up more vertical space, which can limit the robot's design flexibility.
- Potential Height Limitation
    - Since blocks must fit beneath the top rollers, the preroller height must be carefully tuned
    - Too low and it can jam; too high and it may fail to grab blocks effectively
- Low Torque and Compression Control
    - Cannot handle heavy or tightly packed blocks without risk of jamming

OPTION #2: SIDE ROLLERS

Figure #1: Side Roller Prerollers by 94999E from VEX Change Up 2021

The side roller preroller is designed to collect game objects by pulling them inward from both sides of the robot. It consists of two separate rollers positioned parallel to each other, rotating inward to guide blocks toward the robot's center.

Because the two rollers are independent, compression can be finely tuned by adjusting the spacing between them. Increasing compression provides a stronger grip—ideal for securely holding or de-scoring blocks—but decreases fielding range. Conversely, decreasing compression widens the fielding range but reduces grip strength.

PROS:

- Versatile Collection
  - Effective for picking up objects from multiple angles, particularly useful in tight or cluttered field spaces
- Can Be Used For Match Loaders
  - Typically, side rollers are not an effective preroller system. However, this game has provided a unique advantage for side rollers as you are able to match load with them, without creating a seperate mechanism.
- High Torque Capability
  - Performs well in torque-based tasks like de-scoring or handling tightly packed blocks
- Tunability
  - Dual rollers provide adjustable compression, allowing fine-tuning for grip strength and performance.
- Compact Design
  - Occupies less vertical space, allowing for additional subsystems and an overall more modular robot build.
- Reliabile Object Control
  - Two points of contact ensure consistent grip on game elements, reducing slip during operation

CONS:

- Reduced Control on Smaller Objects
  - Smaller items (like blocks) can be more challenging to grip securely, especially on uneven surfaces.
- Requires Additional Stabilization
  - Often requires extra components to guide items accurately to the robot's core.
- Limited Fielding Range
  - Side rollers have a more limited collection width determined by the distance between the rollers and the grip of the roller material.

PREROLLER CHOICE

Laksan Mohan | 07/12/2025

To decide on an preroller, we will be using a decision matrix. The following criteria were considered while deciding on the design.

- Preroller Capture Power (X/15)
  - Strong capture power is essential for quickly and securely collecting blocks from the field.
  - A higher capture power allows for more consistent pickups, improving scoring efficiency and reducing time lost to misalignments or dropped game elements.
- Speed and Efficiency (X/10)
  - The preroller must operate faster than the drivetrain's approach speed to ensure smooth and uninterrupted collection during movement.
  - A fast and efficient preroller allows the robot to maintain momentum while gathering blocks, improving cycle times and scoring potential.
- Reliability and Durability (X/6)
  - Long-term reliability minimizes field breakdowns and allows all parts to last longer, maintaining a consistent performance level. The more dependable the preroller over time, the higher the score, as this is a significant factor for high-stress competition scenarios.
- Ease of Assembly and Maintenance (X/3)
  - Simple and modular construction makes it easier to assemble, adjust, or repair the preroller during practice or between matches.
  - While this is a lower-weighted criterion compared to performance-based factors, ease of maintenance is valuable for ensuring quick recovery in the event of mechanical issues.
- Versatility and Adaptability in all Scenarios (X/10)
  - A good preroller should be able to handle points coming in from weird angles as the robot scrambles to pick up as many points as possible. This means error correction capabilities and a general sturdiness that allows it to take in haphazardly aligned points. The more adaptable the preroller, the higher it will score in this category.
- Weight and Portability (X/3)
  - A lightweight design allows for other subsystems to be reinforced and improved, as well as improving the overall handling of the robot.
- Cost Effectiveness (X/3)
  - As we were operating on a severely limited budget, we also wanted to find designs that wouldn't eat up too much of our budget, allowing for the purchase of other parts.

Analysis

Let's take a closer look at what the advantages and disadvantages of each option are in more detail.

Top-Down	Side Rollers
Friction is primarily dependent on the weight of the preroller assembly pressing the roller onto the blocks. **Advantages:** • Only requires one motor, making it power-efficient • Simple and compact design that fits easily into limited space • Can span the full robot width, allowing for wide fielding range • Much better at taking in points from weird angles • Relatively low-cost and easy to assemble with standard VEX components **Disadvantages** • Performs best when approaching from the front, less effective when attacking from the sides • Not ideal for high-torque or high-compression scenarios due to risk of jamming • Less precise control when dealing with stationary or tightly grouped game objects • Relies heavily on correct roller-to-ground distance for consistent pickup performance	Friction is primarily dependent on the fine-tuning of roller spacing, material, and hinge alignment for consistent compression. **Advantages:** • Provides high torque output due to the use of two motors • Can be tuned for strong compression, ideal for de-scoring or handling tightly packed blocks • Offers precise control when picking up stationary or well-aligned game objects • Compression can be easily adjusted for different game object sizes or grip strengths • Can reach further than top-down by moving wheels forward **Disadvantages:** • Requires precise adjustment of spacing and hinge alignment to function reliably • Without properly tensioned rubber bands or compliant wheels, block grip decreases significantly • Limited fielding range, must be closely aligned with game objects to intake effectively • Tends to deflect rather than align points coming in from side angles, must be facing directly head on • More complex and time-consuming to build than top-down systems

Decision Matrix

	Top-Down	Side Rollers
Capture Power (X/15)	12	15
Speed and Efficiency (X/10)	9	6
Reliability and Durability (X/8)	6	3
Ease of Assembly (X/3)	3	2
Versatility (X/10)	8	10
Weight (X/3)	2	1
Cost Effectiveness (X/3)	3	2
Total (X/52)	**43**	40

After evaluating each preroller design using the decision matrix, our team has decided to move forward with the **top-down preroller intake**. This design was chosen for its simplicity, efficiency, and consistency in collecting blocks. The top-down roller offers a wide fielding range, allowing for easier pickup from multiple angles without requiring perfect alignment, which is especially valuable during fast-paced gameplay. It is also lightweight, easy to assemble, and integrates cleanly with other intake stages, leaving open space for additional mechanisms. While it does not provide as much torque as side rollers, its reliability, speed, and low maintenance requirements make it the best overall option for our robot's intake system.

PREROLLER SPECS

Laksan Mohan | 07/13/2025

Although we've decided on building a top down roller, our work's not done yet. There are still a few decisions that need to be made on the specifics of the design.

DECISION #1: HINGE HEIGHT

For our intake, we must decide on how much to allow it to hinge, and how high to put it above the ground.

To determine how high to hang our intake, we can do some trial and error to determine which amount works best. Through testing, we found that the optimal height of the preroller is 2.5" (approximately 80% of the height of a standard Push Back block).

This height ensures that the roller applies enough downward force to maintain contact and compression with the block while avoiding excessive drag on the field tiles. Placing the preroller slightly above the midpoint of the block prevents jamming and allows for a smooth rolling motion when the robot approaches from any angle.

This configuration also helps guide blocks naturally into the rest of the intake system without requiring additional alignment mechanisms, maintaining both speed and simplicity.

DECISION #2: POWER TRANSMISSION

Next, we must decide how the intake will be powered, with the two main options being a dedicated motor, or piggybacking off of the intake motor.

While having a dedicated motor would provide significantly higher amounts of torque, this wouldn't be an extremely high priority due to the fact that the robot driving forwards would assist in taking in the block, and so the rollers are there to supplement the robots momentum, instead of supersede it. Additionally, we are operating on a strict power limit of 88 watts, and as we are already using 6 motors for the drivetrain, and plan to use 2 motors for the intake, this would conflict with other features. As a result, we are forced to rely on one of the intake motors providing energy to the preroller.

To do this, we will mechanically link the preroller to the rest of the intake

through a chain-driven system

DECISION #3: ROLLER CHOICE

We decided to use VEX Flex Wheels for the preroller. Flex wheels provide an excellent balance of grip, compliance, and durability, ideal for the hard plastic blocks used in Push Back. Their rubber-like material deforms slightly under pressure, allowing the preroller to maintain consistent compression across different block positions.

For this design, we selected 2" diameter 30A Flex Wheels, as they are lightweight and soft enough to conform to the shape of the blocks without damaging them or the field tiles. The smaller diameter also allows the intake to sit lower to the ground, improving pickup consistency and reaction time.

DECISION #4: SHAFT TYPE

The rollers will be mounted on a High Strength (HS) Axle. While standard axles would reduce weight, the HS axle provides significantly greater resistance to twisting and bending, an essential feature for a subsystem that experiences repeated impacts and high rotational torque.

Using HS axles also ensures a secure and long-lasting connection between the roller and the sprockets, reducing slippage under load and preventing shaft warping over time. The axles will be supported on both sides with HS bearing flats to minimize deflection and vibration.

DECISION #5: BUILD STRUCTURE

The preroller assembly is constructed using 2 hole aluminum C-channels for the side supports. Aluminum was chosen for its strength-to-weight ratio and ease of modification. The channel design allows for slotted mounting holes, which provide minor adjustability in the vertical position of the preroller during testing. This flexibility allows us to fine-tune compression levels and optimize intake performance without rebuilding the structure.

DECISION #6: SIZE

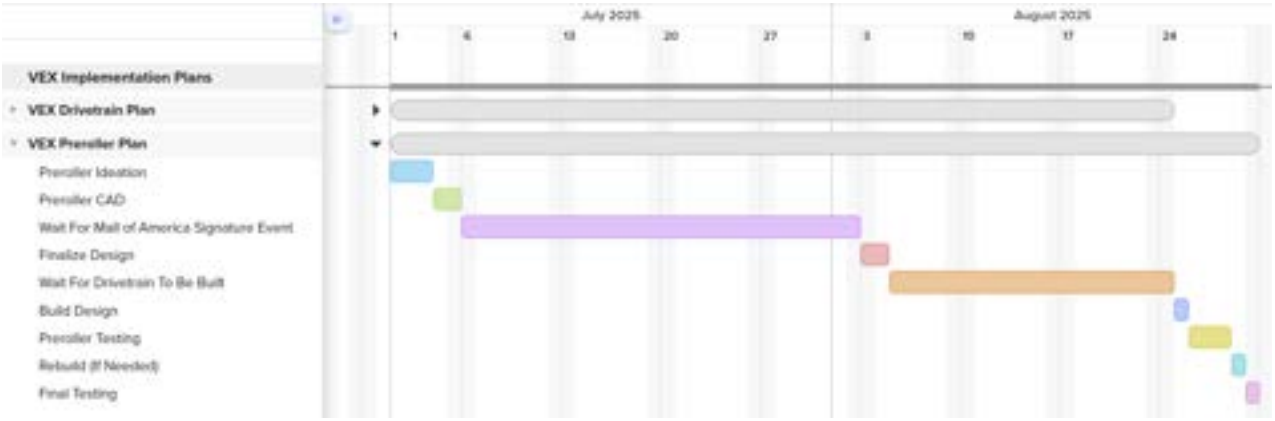
When finalizing the dimensions of the intake, we decided to have the preroller and intake assembly span 11.5" across the front of the robot. By not extending the intake across the full 15" width of the drivetrain, we intentionally left space on both sides of the robot to mount a potential match loader mechanism. Additionally, maintaining this reduced width helps keep the intake structurally rigid and lightweight while simplifying alignment between the preroller and subsequent intake stages.

PREROLLER IMPLEMENTATION PLAN

Laksan Mohan | 07/13/2025

We aim to have a completed CAD done by July at the same time as the Drivetrain CAD. Then, after the Mall of America Signature Event has happened, we will finalize our design and start building once the drivetrain is done.

To guide our construction of the preroller, we have included a timeline below:



INTAKE DESIGN BRIEF

Laksan Mohan | 07/14/2025

Now that we've determined our preroller, we will design the rest of the intake. We plan to have our intake function as a combined intake and scoring subsystem, rather than two separate mechanisms. By integrating both functions into one continuous intake system that spans the entire robot, we can streamline the transfer process, improve alignment consistency and reliability, reduce weight and mechanical complexity, and maximize scoring efficiency.

PROBLEM STATEMENT

The intake system must collect game elements rapidly and consistently from the field, guiding them into the robot for efficient scoring. It should be optimized for speed, reliability, and adaptability, capable of operating under varying field conditions and maintaining performance even during intense match play. The intake must balance compactness and durability while integrating seamlessly with the scoring mechanisms.

CONSTRAINTS

- Must fit within an 18" x 18" x 18" volume per R5
  - Can expand to 22" in all directions after the start of the game
- Must fit within our budget of $2000 for the full robot
- Can only be built with VRC-legal components per R17-R26, with some legal modifications per R27 and R28

OBJECTIVES

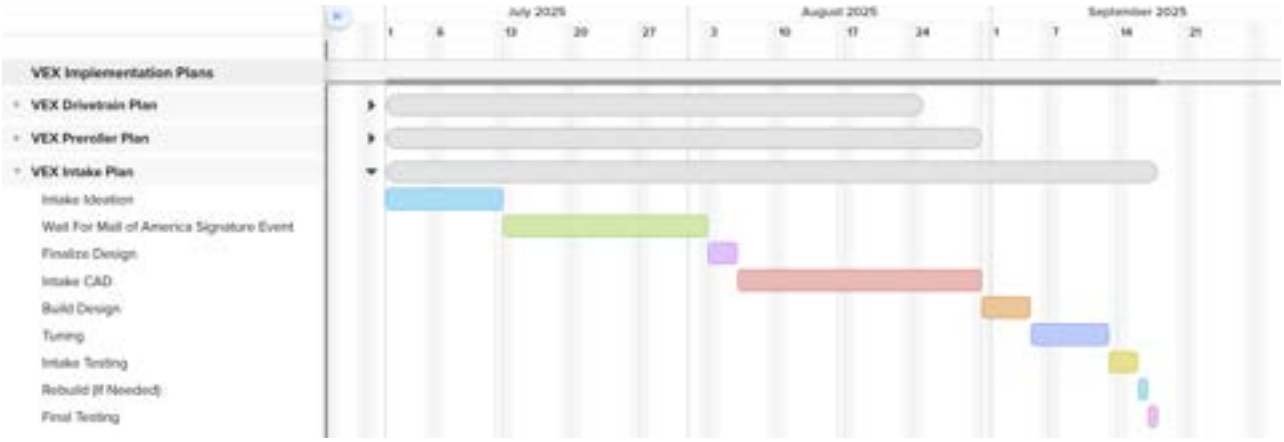
- Must be structurally rigid and resistant to deformation under repeated impacts.
- Must not interfere with drivetrain movement or obstruct scoring mechanisms.
- The design must ensure centered and stable game element intake even at full robot speed.
- Designed to allow easy maintenance and replacement of rollers and spacers between matches.
- Can intake from multiple angles and recover misaligned elements effectively.

- Incorporates anti-jamming features such as indexing and controlled compression spacing.
- Maintains high efficiency during autonomous routines, with precise and repeatable performance.
- Designed to be modular, allowing quick adjustments for prototype testing or mid-season upgrades.
- Constructed with aluminum and polycarbonate components for durability without excess weight.
- Can handle continuous cycles for 2+ minutes without overheating or loss of performance.
- Integrates sensor feedback for colour sorting.

TIMELINE

We will be waiting until the Mall of America Signature Event is over to finalize the design and build the intake. Until then, we will brainstorm ideas for, design, and CAD the intake. Once Mall of America is over, assembly will be quick. However, final testing will take a while as we have to tune the assembly to provide the right amount of compression to have no jamming.

To guide our design cycle of the preroller, we have included a timeline below:



INTAKE OPTIONS

Laksan Mohan | 07/15/2025

OPTION #1: HOPPER INTAKE

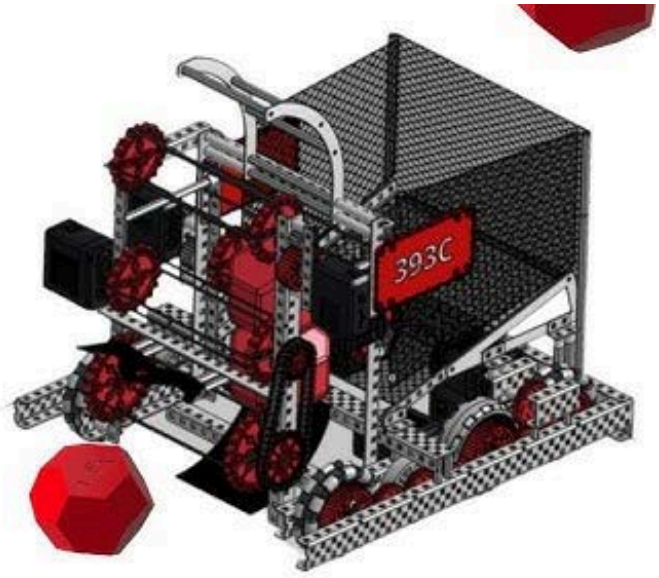


Figure #1: Sample Hopper Bot CAD by 393C

The hopper intake uses a front-mounted preroller that funnels game elements into a large containment “basket” or hopper inside the robot. Once collected, the robot drives to the goal and outtakes from that basket into the goal. The design relies on volume rather than speed, prioritizing capacity and simplicity.

PROS:

- VERY High Capacity
  - This is a MAJOR advantage, as it can hold 20+ blocks
  - This means we can focus on collecting balls at the start of matches and then score all at once while our oponnent is refilling their intake, quickly swinging the match in our favour
- Excellent for Defense
  - The solid basket structure makes the robot physically resilient and capable of pushing back opponents without losing collected elements.
- Simplified Control System
  - Fewer active subsystems mean less code complexity, leading to greater reliability in competition.

- Easy To Gear
  - Because the rollers are mounted onto two long straight c-channels, there is a lot of room and flexibility to gear motors for torque or speed

CONS:

- High Risk of Jamming
  - Because the hopper outtakes from the bottom of the basket, the weight of all the collected balls causes an tight compression zone. As a result, there is a lot of risk of jamming.
  - To fix this, complex mechanism need to be built and tuned (such as agitators and funnels), requiring a considerable amount of time to achieve consistency
- Slower Scoring Cycle
  - The robot must first collect several blocks before driving to score, making it less efficient for single-object scoring or quick cycles.
- Bulkier Structure
  - The large basket takes up considerable internal volume, limiting space for other subsystems or mechanisms.
- Heavier Design
  - Additional metal bracing and panels increase total weight, slightly reducing acceleration and agility.
- Reduced Autonomous Flexibility
  - Requires longer pathing and timing to fill and empty the hopper effectively during autonomous periods.
- Center of Gravity Shift
  - When filled, the basket's weight can raise the robot's center of mass, potentially affecting stability when turning at high speed.

OPTION #2: C-CURVE INTAKE

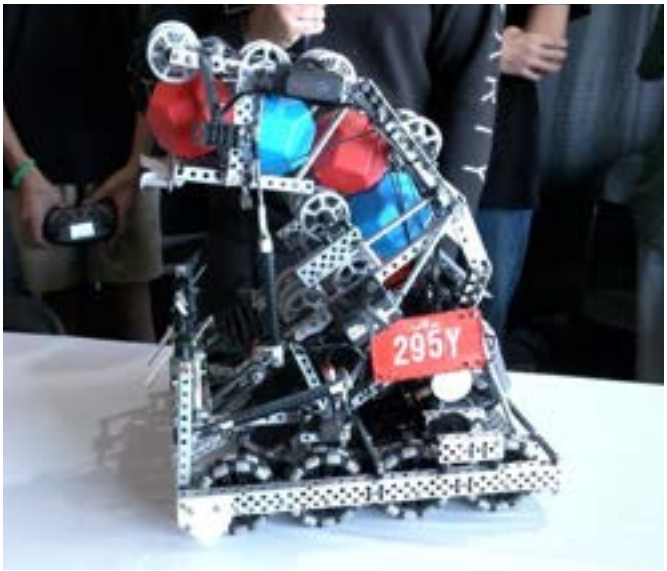


Figure #2: Sample C-Curve Bot by 295Y Party

The C-Curve intake utilizes a curved conveyor intake system that forms a “C”-shaped path through the robot, allowing game elements to be collected, lifted, and scored in one continuous motion. The intake starts with a set of front prerollers that pull blocks inward from the field and immediately guide them upward along a curved backplate or roller channel. As the blocks travel through the curve, they are compressed lightly between opposing rollers or a polycarbonate guide surface, keeping them centered, stable, and under full control at all times.

PROS:

- Extremely fast scoring:
  - This design eliminates the need for a separate handoff between the intake and scoring subsystems, creating a unified mechanism that both collects and scores
  - Once the triballs reach the top of the C-curve, they are immediately positioned at the correct height and orientation for scoring into any of the goal zones
  - Because the motion path is continuous, this intake achieves exceptionally fast cycle times (it can collect and score blocks within seconds of contact).
- Compact and Integrated
  - Combines intake and scoring functions into a single mechanism, freeing up internal space for drivetrain stability or sensors.
- Smooth Control
  - Continuous roller contact keeps elements centered and prevents bouncing or loss of grip.
- Versatile Scoring Range
  - The curved geometry can be tuned to reach all goal heights, from low to high zones, through a pneumatic hood.
- Efficient for Autonomous Routines
  - Minimal transition delay enables precise and repeatable scoring paths.
- Stable Under Pressure
  - The roller compression ensures reliable intake even during pushing or defensive maneuvers.
- Easy to Align
  - The intake naturally centers blocks as they move along the curve, simplifying driver control.

CONS:

- Complex to Design and Tune
  - The curved geometry requires careful CAD modeling to maintain correct spacing and roller compression.
- Higher Maintenance Demands

    - Multiple rollers, belts, and axles increase potential wear points and make repairs more time-consuming.
- Heavier than Linear Systems
    - Extra rollers and support structures can add weight if not optimized
- Limited Visibility Inside the Curve
    - Can make diagnosing jams or friction issues more difficult during early testing.
- Requires Precise Calibration
    - Small inconsistencies in roller height or speed can disrupt the smooth feed motion.

OPTION #3: S-SHAPED INTAKE

Figure #3: Sample S Intake by Team 11811A Hula Hooligans

The S-Shaped intake features a unique front-to-back intake path that follows an “S”-shaped curve through the robot. Game elements are collected from the front rollers, guided upward through a controlled curve, and then redirected toward the back of the robot where they are stored or immediately scored. Unlike hopper or C-shaped systems, this layout allows the bot to intake from one side and score from the opposite, creating a highly efficient flow of game elements through the chassis.

PROS:

- Front-To-Back Intake Flow
    - This front-to-back motion offers a major strategic advantage in Push Back: the robot can match load directly from the match loaders and then simply drive backward a few feet to score without needing to turn or reposition.
    - This reduces both cycle time and driver workload while keeping the robot aligned with the goal at all times.

- Stable Control Path
  - The continuous S-curve keeps blocks compressed and centered, reducing jamming and ensuring reliable feeding.
- Compact and Efficient Use of Space
  - Vertical stacking and curvature allow the system to fit multiple rollers within the 18” frame
  - This design has the 2nd biggest capacity
- Excellent Scoring Accuracy:
  - Because the scoring exit is aligned to the rear, blocks are consistently released from the same height and angle.
- Fast Cycle Times
  - Because the blocks flow through one fixed path, it can be tuned to be extremely consistent and have fast scoring cycles

CONS:

- Challenging to Build and Tune
  - The alternating roller heights and angles require precise alignment in CAD and physical assembly
  - Out of every design, this is the most difficult to build and tune
- Difficult Maintenance
  - Limited internal visibility makes troubleshooting jams or uneven compression more time-consuming.

INTAKE CHOICE

Laksan Mohan | 07/16/2025

To decide on an intake, we will be using a decision matrix. The following criteria were considered while deciding on the design.

- Scoring Speed and Cycle Efficiency (X/15)
  - The intake must enable fast and fluid transitions from pickup to scoring, minimizing wasted motion and downtime between cycles.
  - Higher-scoring designs will intake and score blocks faster than the drivetrain's travel speed, allowing the robot to maintain momentum and complete multiple scoring cycles within a single match.
- Control and Consistency of Game Elements (X/10)
  - Precise and reliable handling of blocks is critical to prevent drops, jams, or misalignments.
  - A high-scoring design ensures consistent compression and feed rate across all stages of the intake, maintaining full control over elements during rapid movement or collisions.
- Reliability and Durability (X/8)
  - Durability is vital for consistent match performance across long competitions.
  - An intake that resists jamming, misalignment, and mechanical fatigue will score higher in this category, as it reduces the risk of performance loss during key matches and autonomous routines.
- Ease of Assembly and Maintenance (X/2)
  - Intakes that are simpler to build, adjust, and repair between matches save valuable time during events, allowing for quick pit turnaround and testing cycles.
  - An accessible design that allows easy roller or belt replacement will score higher.
- Integration with Other Subsystems (X/10)
  - Because our design goal is to combine intake and scoring into one continuous mechanism, an intake that naturally integrates with our scoring system and overall robot layout will be rated higher.
  - The better the design aligns with chassis space, weight distribution, and scoring geometry, the more it contributes to overall robot efficiency.
- Adaptability and Field Versatility (X/7)
  - The intake should perform reliably from multiple angles, handle uneven field conditions, and recover from misaligned pickups.
  - Higher scores are given to designs that can intake blocks during motion, operate under defense, and function effectively from both

sides of the field.

- Weight and Center of Gravity Impact (X/3)
  - Lightweight intakes reduce strain on the drivetrain and maintain better robot balance, especially during pushing or rapid acceleration.
  - Compact, weight-efficient systems score higher as they improve overall maneuverability without compromising strength.
- Scoring Versatility (X/10)
  - The ability to score on all goal heights—low, mid, and high—is essential for strategic flexibility.
  - An intake that can consistently deliver block to any scoring target without repositioning will score higher.

Analysis

Let's take a closer look at what the advantages and disadvantages of each option are in more detail.

Hopper	C-Curve	S-Shaped
The Hopper intake focuses on collecting and storing multiple blocks in a large chamber before releasing them all at once to score. **Advantages:** • High storage capacity • Simple and reliable build • Easy to maintain • Great for defensive play • Consistent bulk scoring **Disadvantages** • Slow scoring cycles • Large and heavy structure • Less precise scoring • Inefficient for single blocks	The C-Curve intake features a continuous roller path shaped like the letter "C." Game elements are pulled in from the front and guided upward through a curved channel, where they are compressed and transported directly into the scoring position. **Advantages:** • Very fast intake and scoring cycle • Smooth, continuous motion with no handoff • Compact and space-efficient design • Consistent control and compression • Can score at multiple goal heights	The S-Shape intake uses an "S"-curved roller path that moves nlocks from the front of the robot to the back. This front-to-back intake flow allows the robot to collect from one side and score from the opposite side, enabling fast match-loading cycles. **Advantages:** • Fast and simple scoring (just drive backward) • Smooth front-to-back flow • Consistent scoring position • Compact and stable design • Good control and alignment

	Disadvantages: • Complex to design and tune • Harder to repair or adjust internally • Slightly heavier than other options • Sensitive to roller alignment	**Disadvantages:** • Harder to CAD and assemble • Slightly slower than C-curve • More friction points • Shifts weight toward the back

Decision Matrix

	Hopper	C-Curve	S-Shaped
Scoring Speed and Cycle Efficiency (X/15)	6	14	13
Control and Consistency of Game Elements (X/10)	7	9	8
Reliability and Durability (X/8)	8	6	6
Ease of Assembly and Maintenance (X/2)	1	2	2
Integration with Other Subsystems (X/10)	9	7	10
Adaptability and Field Versatility (X/7)	7	6	7
Weight and Center of Gravity Impact (X/3)	1	2	2
Scoring Versatility (X/10)	6	6	10
Total (X/65)	44	52	**58**

After evaluating each intake design using the decision matrix, our team has decided to move forward with the **S-shaped intake system**. This design was chosen for its strong balance between speed, capacity, control, and integration. The front-to-back intake path allows the robot to match load and then simply drive backward to score, reducing driver workload and improving cycle efficiency. Its smooth, curved flow keeps blocks stable and aligned throughout the entire intake process, ensuring consistent performance during both autonomous and driver control periods. The S-shaped intake also fits well within the robot's frame and weight limits while maintaining a compact, modular structure that integrates seamlessly with the scoring mechanism. Although it is more complex to tune than simpler systems, its versatility, reliability, and efficient scoring motion make it the best overall choice for our robot's combined intake and scoring subsystem.

INTAKE SPECS

Laksan Mohan | 07/17/2025

Although we've decided on building a s-shaped intake, our work's not done yet. There are still a few decisions that need to be made on the specifics of the design.

DECISION #1: CAPACITY

After careful consierdation and game analysis, we decided that we want to hold a maximum capacity of 10 blocks. This capacity was determined based on optimal space usage within the chassis, ensuring that the intake can store a full field cycle without causing jams or excess friction. Additionally, as we have learned through early-season match analysis, fast cycle times are better than high capacity. Typically, you only need to hold 6-7 balls at a time. However, we added extra capacity as it allows up to not refill as often and, when full, can score the majority of a long goal.

DECISION #2: ROLLER TYPE

Next, we need to decide on what rollers we will use to push the block through the intake. To ensure optimal compression and control through the S-shaped path, the intake uses a combination of 30A 2" Flex Wheels and rubber band rollers, each chosen for their specific mechanical properties.

- Front Rollers (Entry):
  - The front rollers, including the preroller and the next internal roller, uses 30A 2" Flex Wheels. These soft, compliant rollers guide the blocks smoothly into the intake while minimizing bounce and over-compression. Their low durometer rating allows them to deform slightly under pressure, providing strong grip without excessive drag.
- Mid Rollers (Curved Section):
  - The middle rollers use rubber band rollers. These rollers compress more aggressively, making them ideal for maintaining traction around the intake's curved geometry. The extra compliance makes tuning around the S-curve much easier and ensures that blocks remain in contact even when slightly misaligned.
- Rear Rollers (Scoring Exit):
  - The final set of rollers at the back of the robot returns to 30A 2" Flex Wheels. Here, the focus is on precision and control. The flex wheels gently guide the blocks into the scoring area without sudden release or bounce, improving accuracy during scoring.

This hybrid configuration provides the best balance between compression, control, and adjustability, with flex wheels managing precision and rubber band rollers providing adaptability in the high-friction zones of the curve.

DECISION #3: POWER TRANSMISSION

The intake will be powered by three dedicated intake motors, linking rollers through a chain-driven system to synchronize speeds along the curve. Chain drive was chosen for its high torque capacity and minimal slippage under load.

The bottom rollers will be driven by a single 11W motor, spinning at 600 RPM. The preroller assembly will remain mechanically connected to this drive system, ensuring consistent rotational velocity throughout the entire intake path. By sharing power between the preroller and internal rollers, we reduce the number of motors needed while maintaining strong intake performance. The middle rollers will be driven by a 5.5W motor (spinning at 600 RPM) and the rear roller will be powered by another 5.5W motor (900 RPM). We had to do this so the rear roller could be reversed while the rest of the intake spins (allowing us to score on the middle goall) and to fit within the 88W motor limit. The rear roller is geared at a higher RPM as if it was spinning slower than the rest of the intake, it would cause a bottleneck point where blocks get jammed. We chose the rest of the intake to spin at 600 RPM because it was fastest speed to easily gear to and it will allow us to score a full long goal in under 5 seconds.

DECISION #4: SHAFT TYPE

Unlike the preroller, which uses High Strength (HS) axles, the main intake system will use Low Strength (LS) axles throughout.

While LS axles are weaker and more prone to twisting under high torque, they were chosen intentionally for several key reasons:

- **Ease of Construction:** LS axles are much easier to build around. The HS bearing flats and collars are bulky, requiring extra clearance and often custom-drilled holes to mount properly, which would make the intake more complex and time-consuming to assemble.
- **Weight Reduction:** The S-intake uses a large number of rollers and axles, meaning using HS axles throughout would significantly increase the robot's total weight. LS axles keep the structure lighter without sacrificing necessary rigidity for this application.
- **Sufficient Strength for Purpose:** Because the intake rollers are not exposed to high torque loads, LS axles provide adequate performance while simplifying design and reducing stress on the frame.

DECISION #5: STRUCTURAL DESIGN

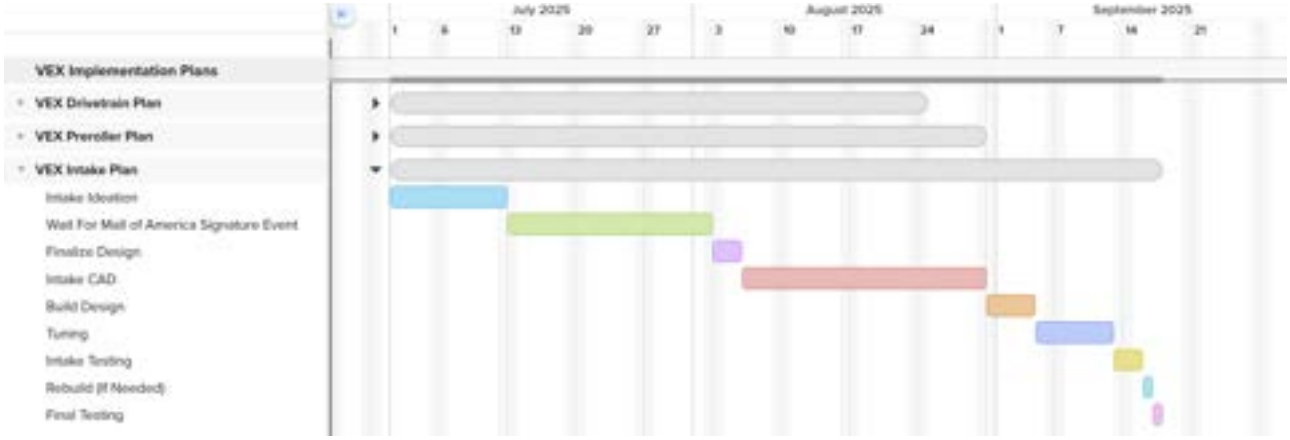
The intake will be constructed primarily from 2 aluminum C-Channels that are mounted in an X shape. This was done as it provides the greatest amount of mounting holes and flexibility for us to tune and get the S-Shape right. The ramps will be made from 1x1 aluminum L-channels with mesh in the middle, chosen for their excellent strength-to-weight ratio and ease of adjustment. The mesh works as a great ramp that will bend to allow the balls to easily flow through. Bracing will be kept minimal to maintain low weight while ensuring rigidity against torsional forces caused by repeated impacts.

INTAKE IMPLEMENTATION PLAN

Laksan Mohan | 07/18/2025

Because the intake is a point where our design could vary drastically, we will be waiitng until the Mall of America Signature Event to happen before we CAD or build. Then, after the Mall of America Signature Event has happened, we will finalize our design and start the CAD and building once the drivetrain and preroller are done.

To guide our construction of the intake, we have included a timeline below:



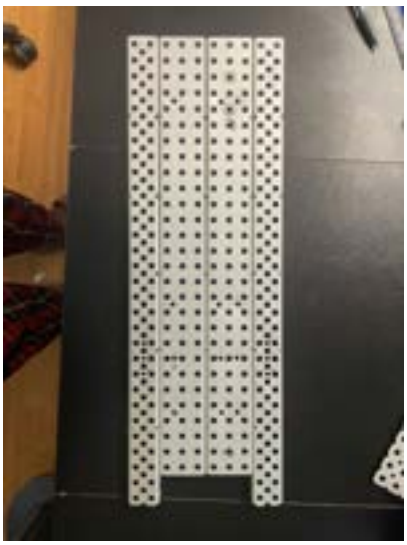
DRIVETRAIN BUILD LOG

Laksan Mohan | 08/17/2025

This section covers our entire build process from August 10th, 2025 to August 20th, 2025

TEAM MEMBERS INVOLVED

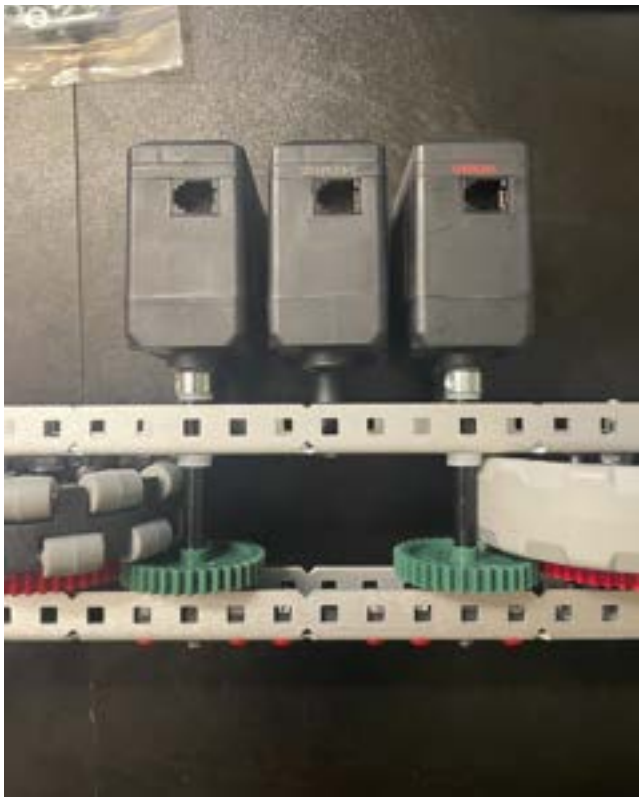
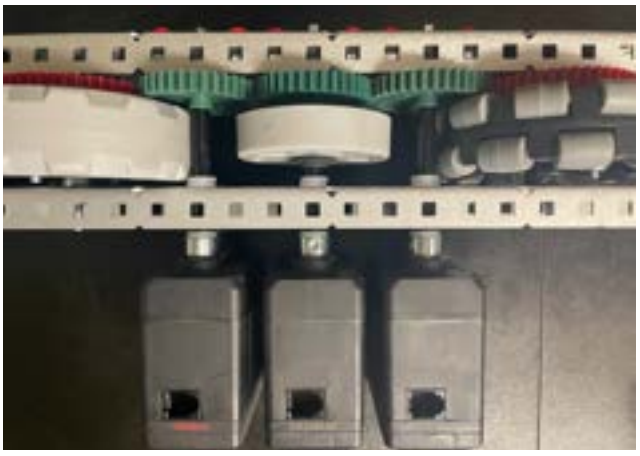
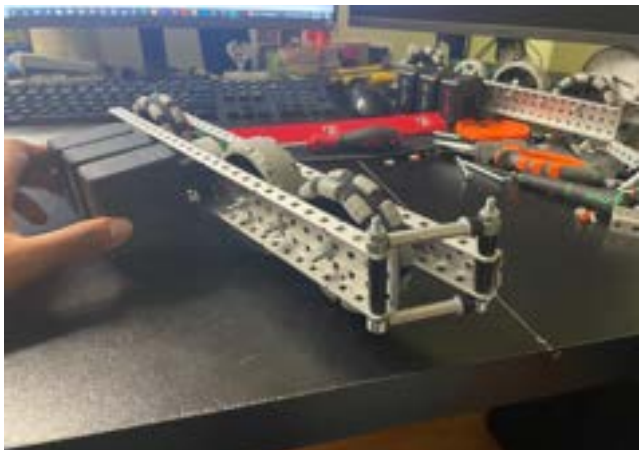
- Laksan
- Vabithan
- Thikshaan

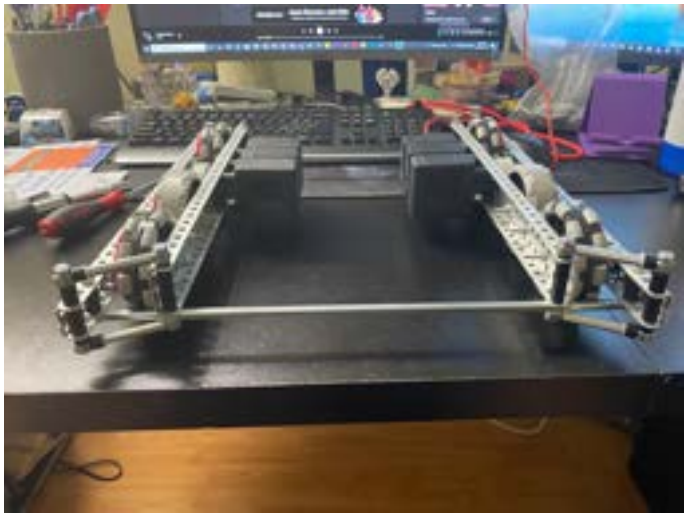


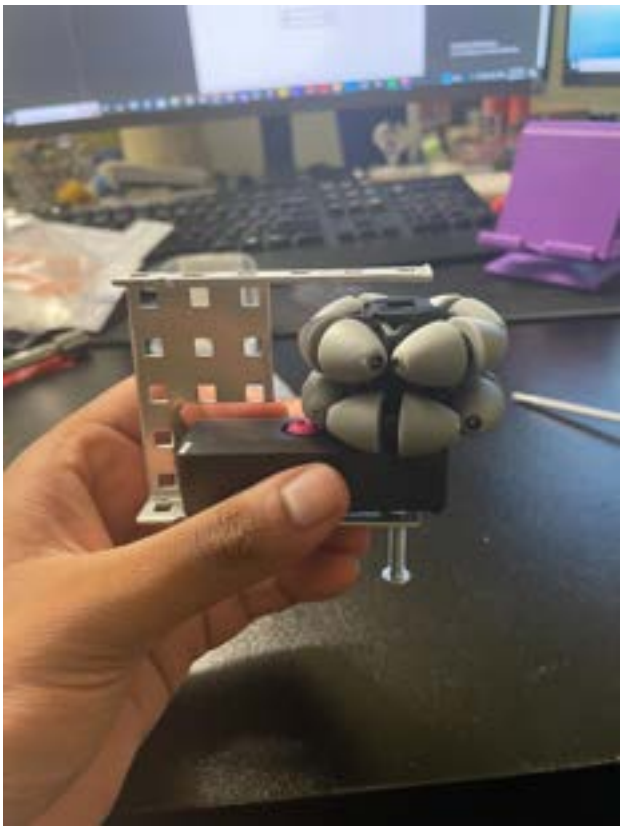


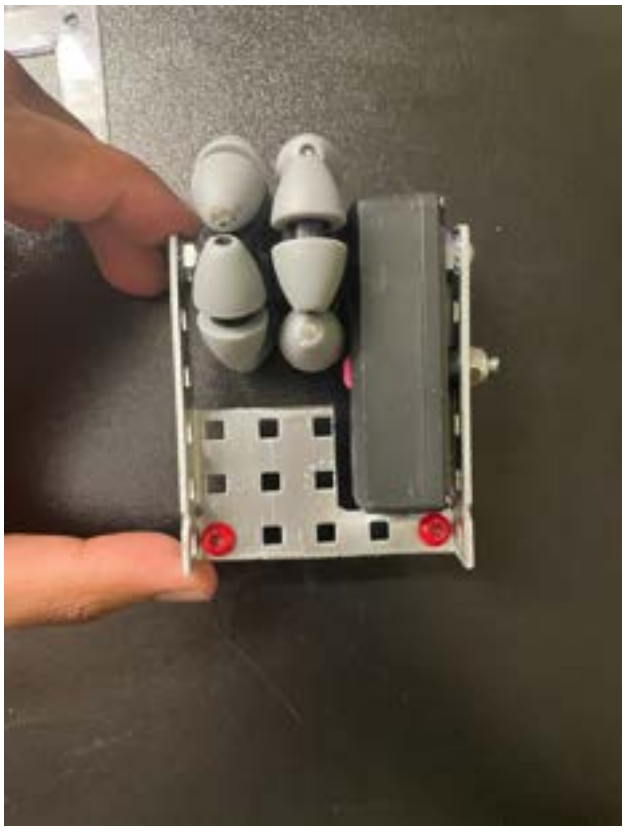


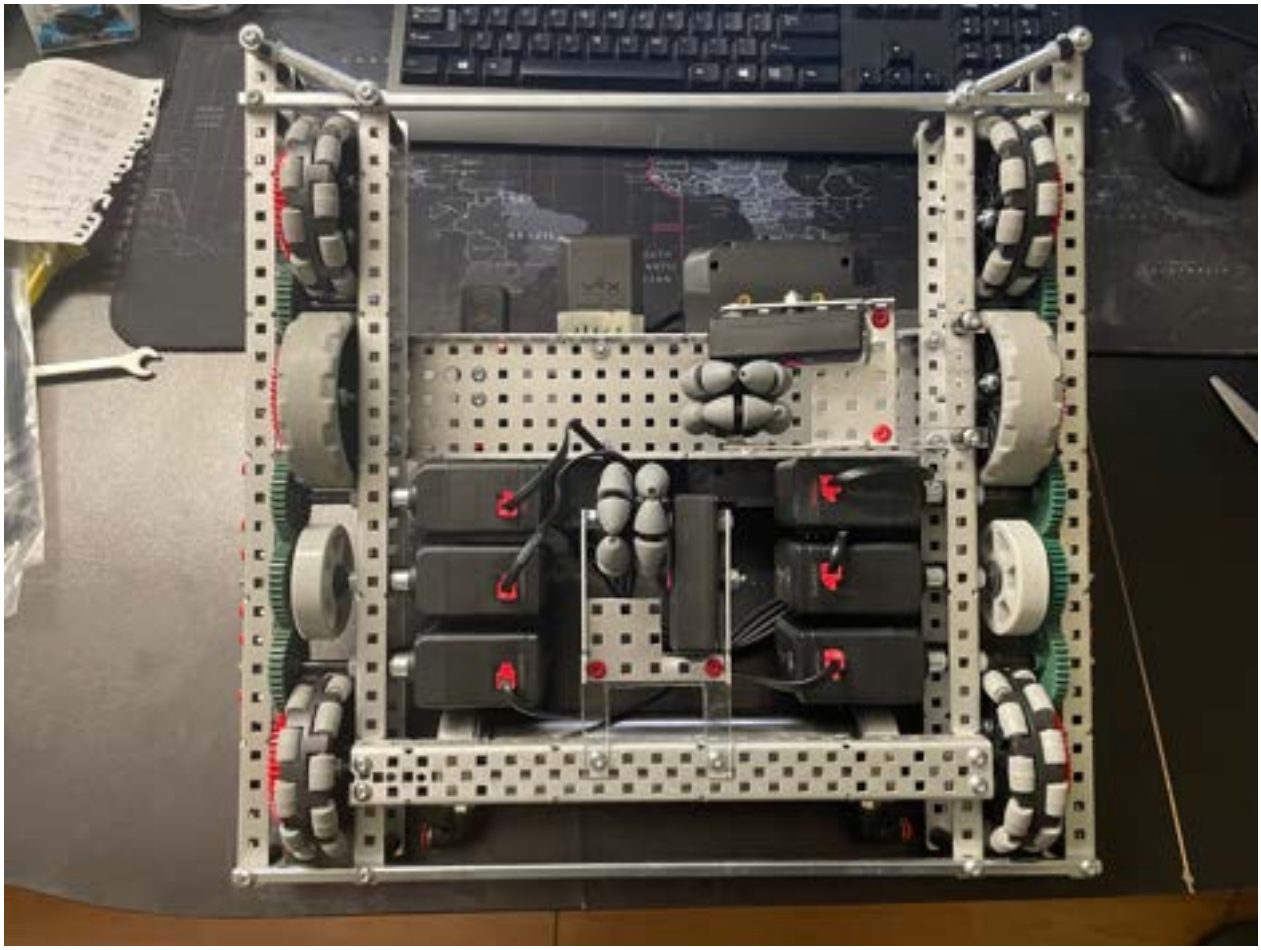


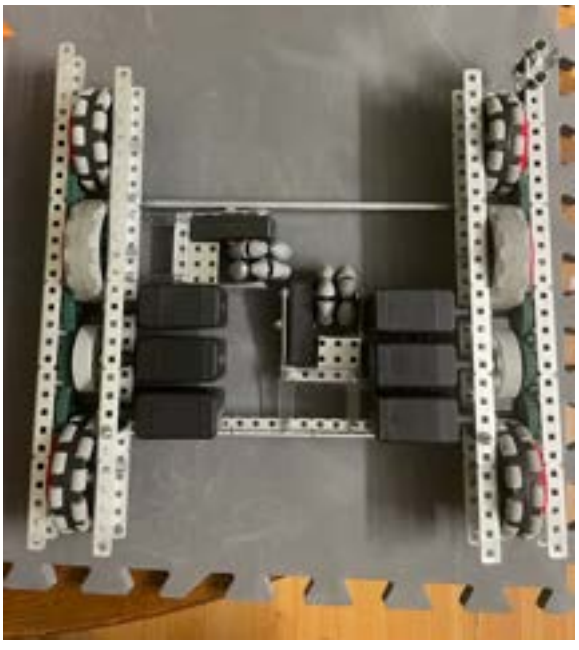




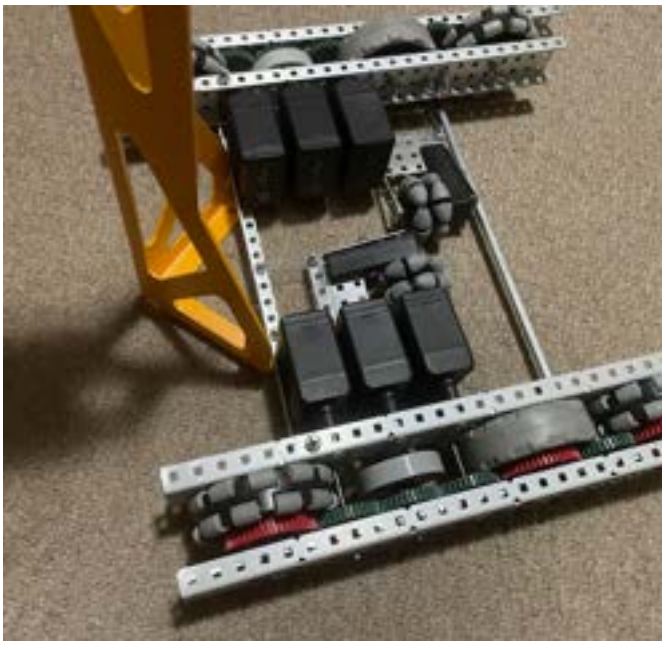












DRIVETRAIN TESTS

Laksan Mohan | 08/24/2025

Now that our drivetrain is built, we will come up with some tests and methodology. Once our full robot is complete, we will conduct these tests and record the results in a separate entry.

TEST #1: SPEED

With this test, we hope to measure the maximum speed of the drivetrain.

Methodology:

- The drive will complete 10 runs in a VEX field.
- The robot will start with one edge against the side. We will a code to assist us with this test.
- When the program is executed, the robot will drive 108 inches, roughly the width of the field, and then stop when a sensor detects a line.
- Then the robot will turn around, be situated, then execute another run.
- The program will record the time from the beginning until the line is sensed.
- New batteries will be used for each drive set of 10.
- This method takes out the tedious and error-prone measuring of distance and makes time as measured internally by the V5 brain the only variable.
- The runs will be completed 10 times, then stored on an SD card and downloaded and analyzed.

TEST #2: FORWARDS FORCE

With this test, we hope to measure the drivetrain's ability to push other robots.

Methodology:

- A force gauge from our school's science department will be attached to the field border structure.
- A bar will then be rested on the base of the hook and the drive will then run with all motors at 100% for 10 seconds.
- The force gauge records the highest reading. 10 readings will be taken, being careful to recreate the starting position each time.
- New batteries will be used for each set of 10 readings.

TEST #3: SIDEWAYS FORCE RESISTANCE

With this test, we hope to measure the drive's ability to resist a push from the side, at a 90-degree angle from its forward direction.

Methodology:

- The chassis is placed onto the field
- The force gauge is hooked onto the middle of a chassis, along the left edge
- Force is applied to the end of the scale until the chassis begins to move in a direction perpendicular to its forward direction
- The maximum force is recorded and the force is removed
- The test is reset and reran until 5 tests are completed.

TEST #4: TURNING SPEED

With this test, we hope to evaluate how well the drivetrain can turn and manoeuvre.

Methodology:

- The test will be conducted with a very basic step-turning program using an internal sensor
- The turn will be at 75% power for the first 30 degrees, 25% power for the next 30 degrees, and lastly 5% for the last 30 degrees
- The variables that will be recorded are how long it took to turn 90 degrees and how much deviation from the starting point the frame moved in the x and y direction
- To be more specific, the robot will turn 90 degrees four times and then record the time for each 90-degree turn
- After the four turns, the distance from where one corner of the drivetrain had started to where it ended will be measured
- This process will be repeated 10 times, so there was an average of the times of 40 90-degree turns and the x and y translation of 10 full 360-degree turns.

TEST #5: ENDURANCE

With this test, we hope to measure how long the drivetrain performs under continuous operation.

Methodology:

- Run the full robot continuously while picking up game pieces
- Record motor temperatures on the brain during the full test
- After 10 minutes or when the battery dies (whatever comes first), stop the robot and check for any signs of overheating or wear

PREROLLER BUILD LOG

Laksan Mohan | 08/25/2025

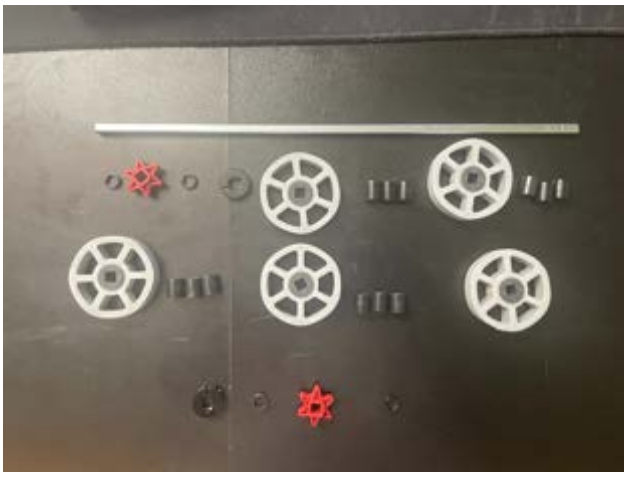
This section covers our entire build process on August 24th, 2025

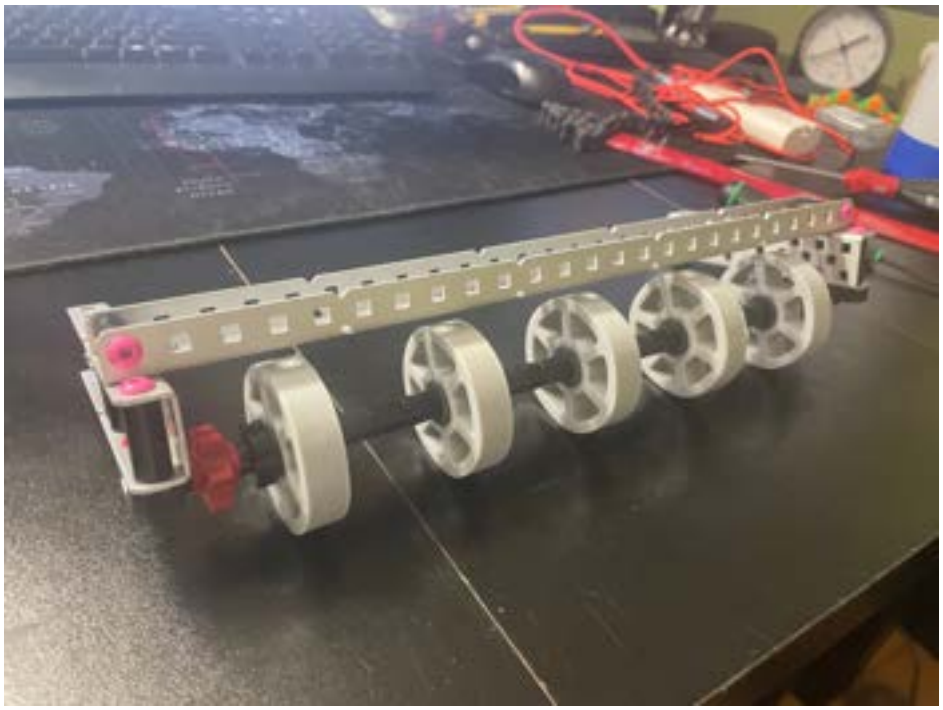
TEAM MEMBERS INVOLVED

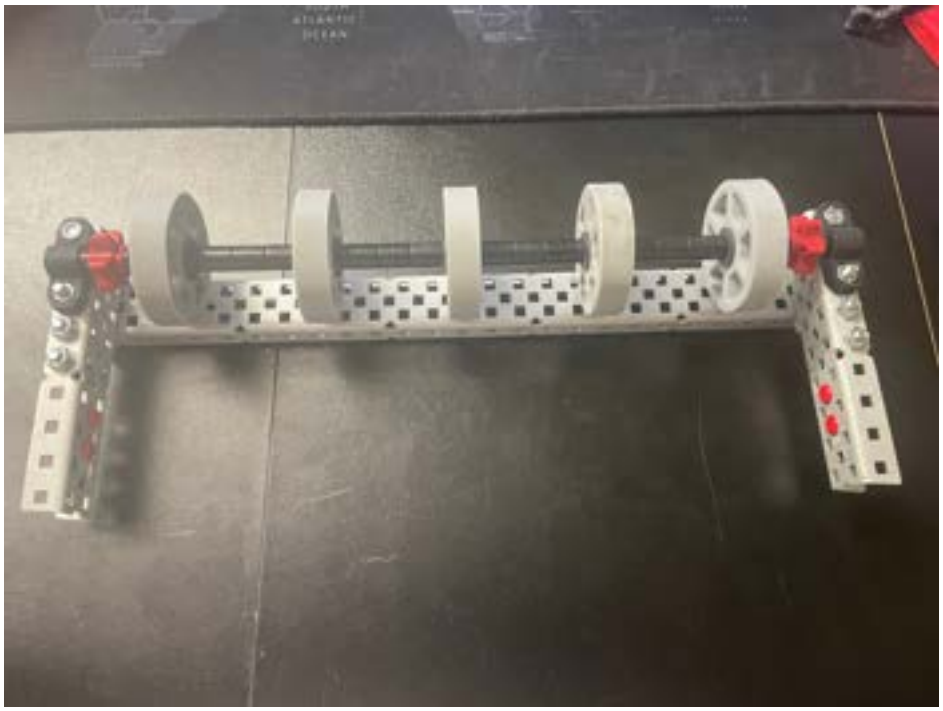
- Laksan
- Vabithan











PREROLLER TESTS

Laksan Mohan | 08/26/2025

Now that our preroller is built, we will come up with some tests and methodology. Once our full robot is complete, we will conduct these tests and record the results in a separate entry.

TEST #1: INTAKE SPEED

Objective
To measure how quickly the preroller can intake a game piece from rest into the second stage of the intake.

Methodology:
- The robot will be placed with a block directly in front of the preroller, aligned to the center of the roller width.
- Using the V5 Brain timer, the time between the start of roller motion and when the block reaches the secondary intake rollers will be recorded.
- This test will be repeated 10 times to find the average intake time.
- The test will be performed at 50%, 75%, and 100% roller power to determine speed consistency across power levels.

TEST #2: ALIGNMENT AND CENTERING

Objective:
To determine how effectively the preroller can center misaligned blocks and feed them straight into the intake.

Methodology:
- Blocks will be placed 2", 3", and 4" offset from the intake centerline.
- The preroller will be activated at 75% speed.
- Observations will be made on whether the block self-centers or jams.
- This will be repeated 5 times per offset distance.
- Success rate (%) will be calculated as the number of times the block was successfully centered divided by total trials.

TEST #3: ALIGNMENT AND CENTERING

Objective:
To evaluate how well the roller material grips and maintains contact with blocks under different surface conditions.

Methodology:

- Tests will be conducted with clean, dusty, and slightly oily blocks.
- The preroller will be run at a constant speed and the time for the block to be drawn fully into the intake will be recorded.
- Any slippage or bouncing will be noted.
- Each condition will be tested 5 times.

TEST #4: CHAIN RELIABILITY

Objective:
To test the stability and durability of the chain powering the preroller when subjected to continuous intake operation.

Methodology:

- The preroller will be run continuously for 10 minutes at 100% power.
- The intake system will repeatedly pick up and eject blocks during the test.
- Chain tension, temperature, and slippage will be checked every 2 minutes.
- Any derailment or loss of motion will be recorded.

TEST #5: HEIGHT EFFECTIVENESS

Objective:
To confirm that the preroller height (set to 80% of the block height) allows for consistent block pickup and transition.

Methodology:

- A range of 5 preroller heights will be tested (70%, 75%, 80%, 85%, and 90% of block height).
- For each height, 10 intake runs will be performed.
- Success will be determined by whether the block enters smoothly without being pushed or stuck.
- Results will be graphed to find the optimal height-performance relationship.

TEST #6: MULTIPLE BLOCK HANDLING

Objective:
To evaluate the preroller's ability to handle back-to-back blocks efficiently.

Methodology:

- Two and three blocks will be lined up directly in front of the intake.
- The preroller will be run continuously, and whether each block is picked up cleanly without interference will be recorded.

- Any double-feeding or jams will be noted.
- 5 trials will be performed for each case.

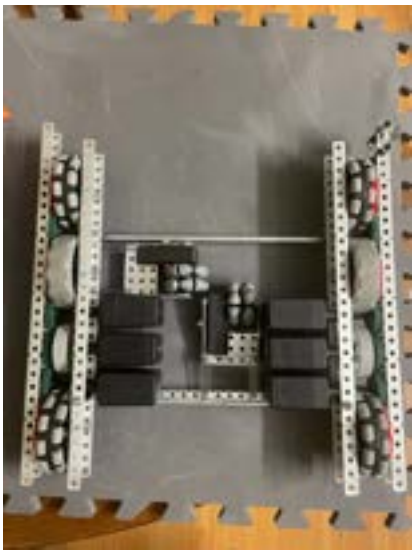
INTAKE BUILD LOG

Laksan Mohan | 08/30/2025

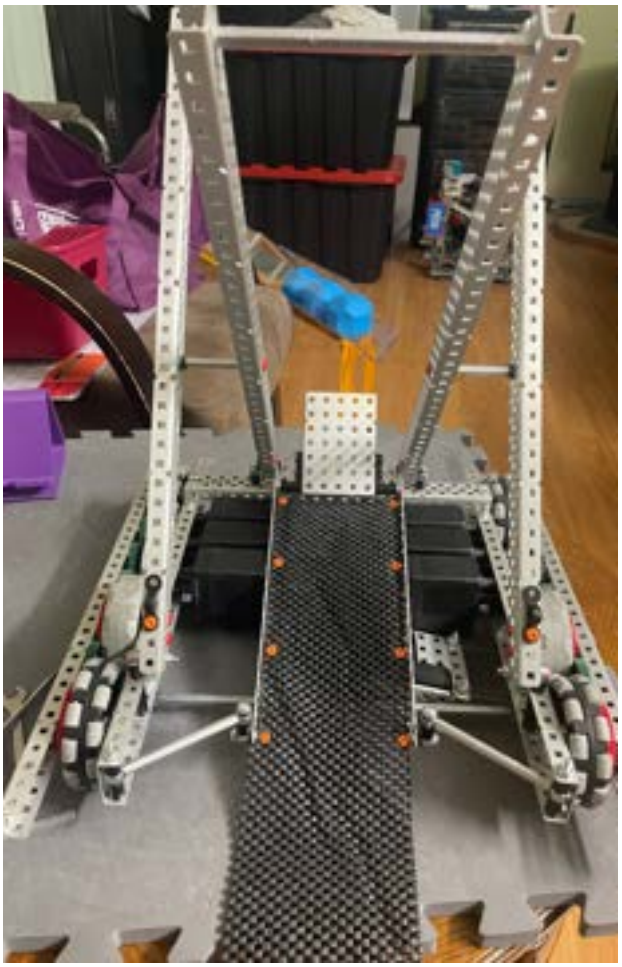
This section covers our entire build process from August 30th, 2025 to September 15th, 2025

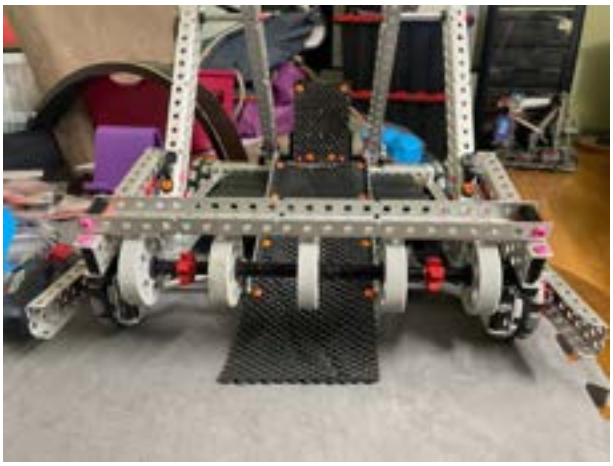
TEAM MEMBERS INVOLVED

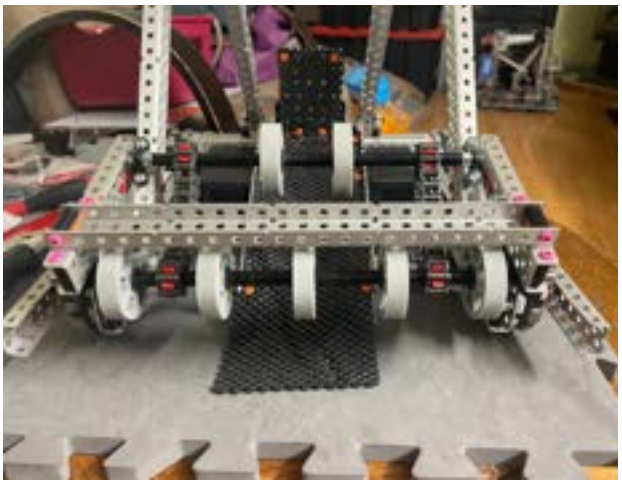
- Laksan
- Vabithan
- Thikshaan

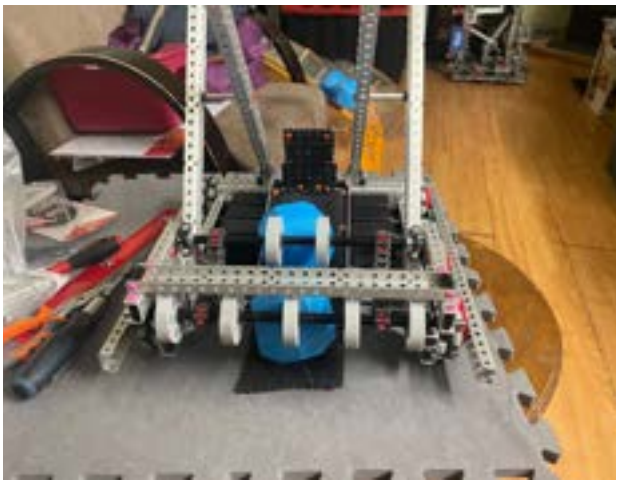


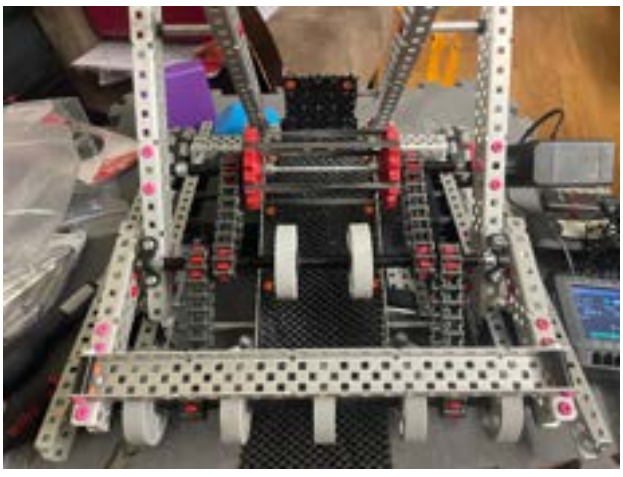


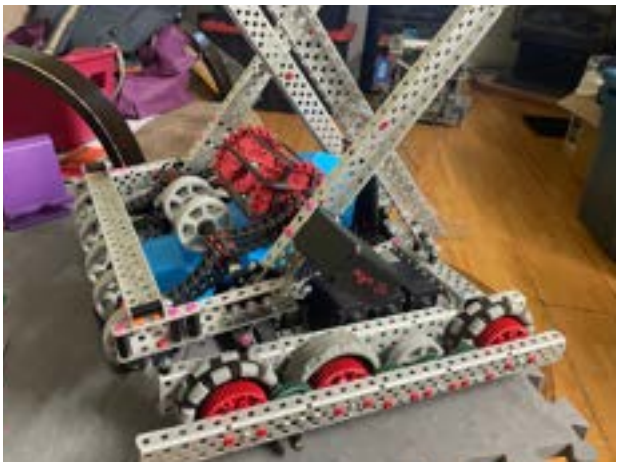


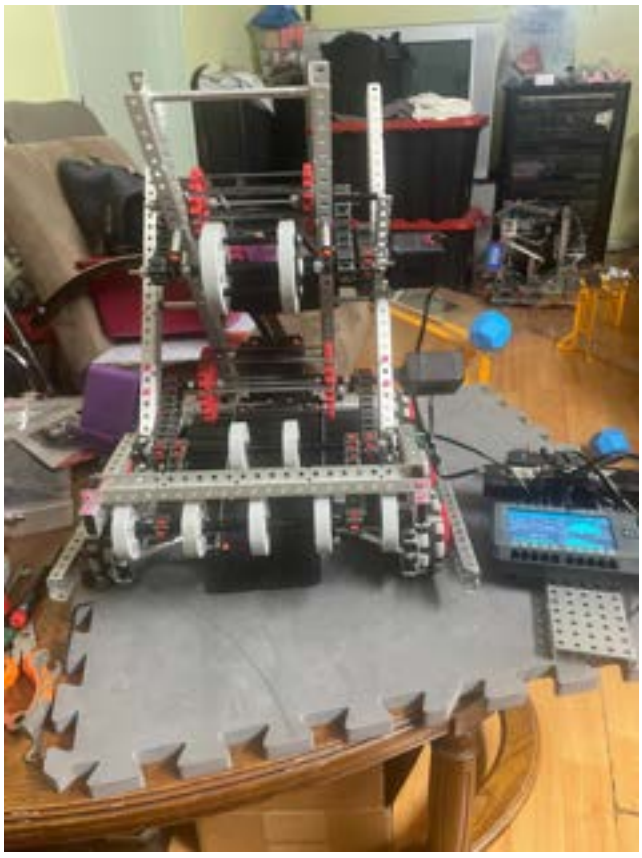


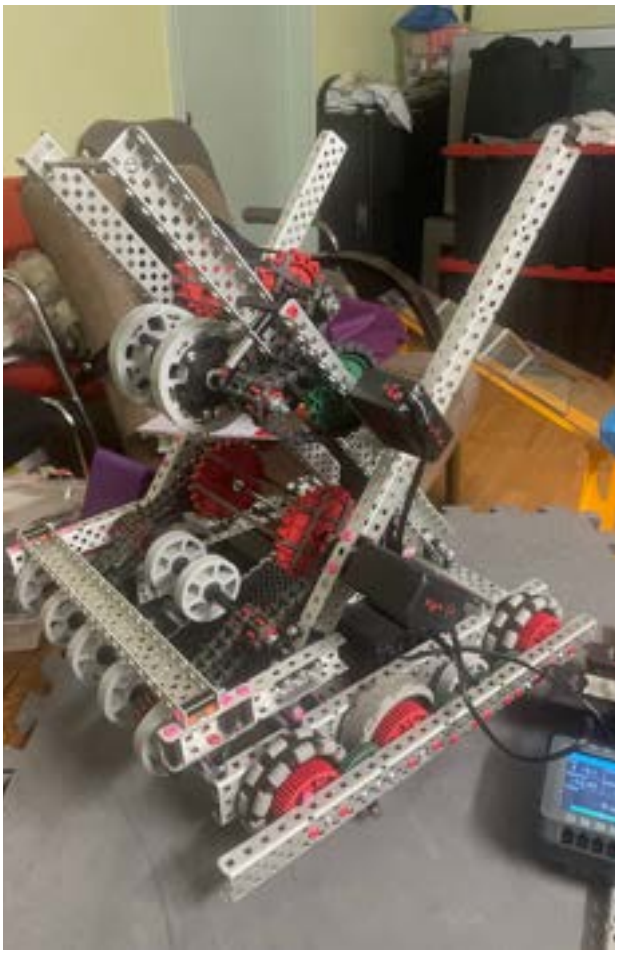


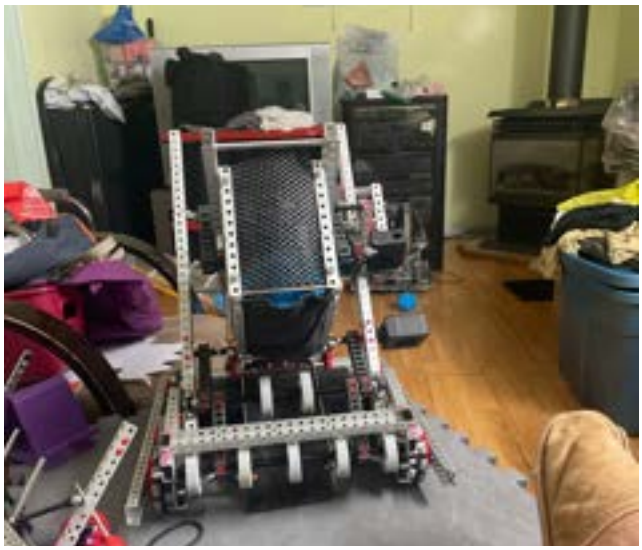


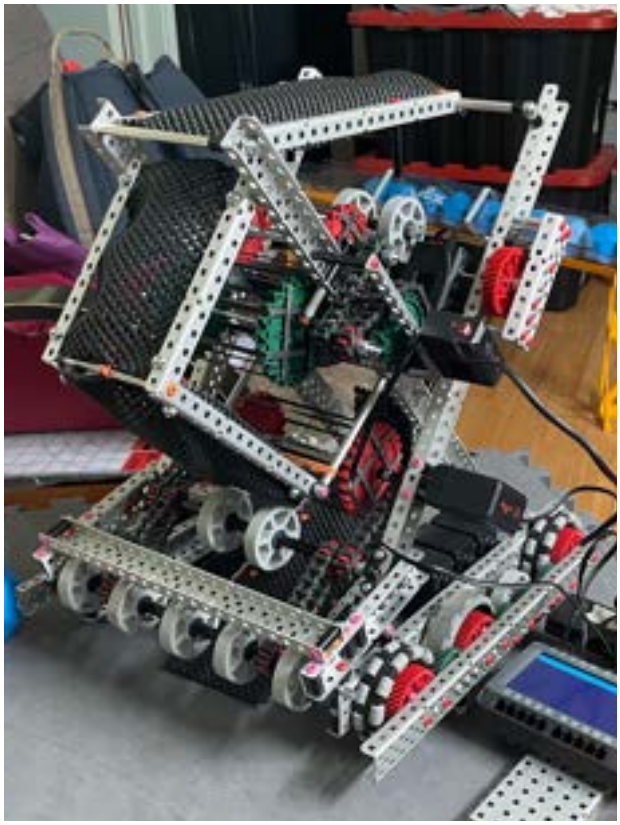


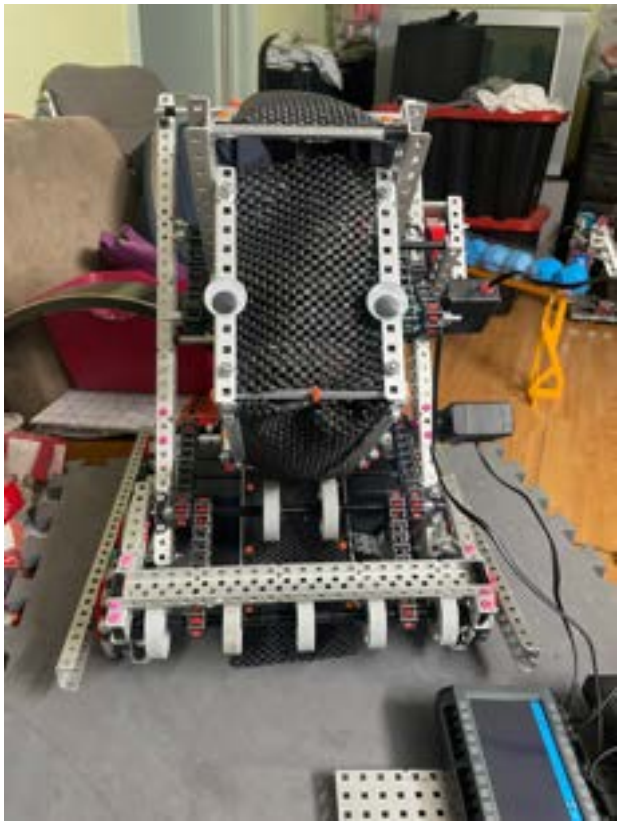


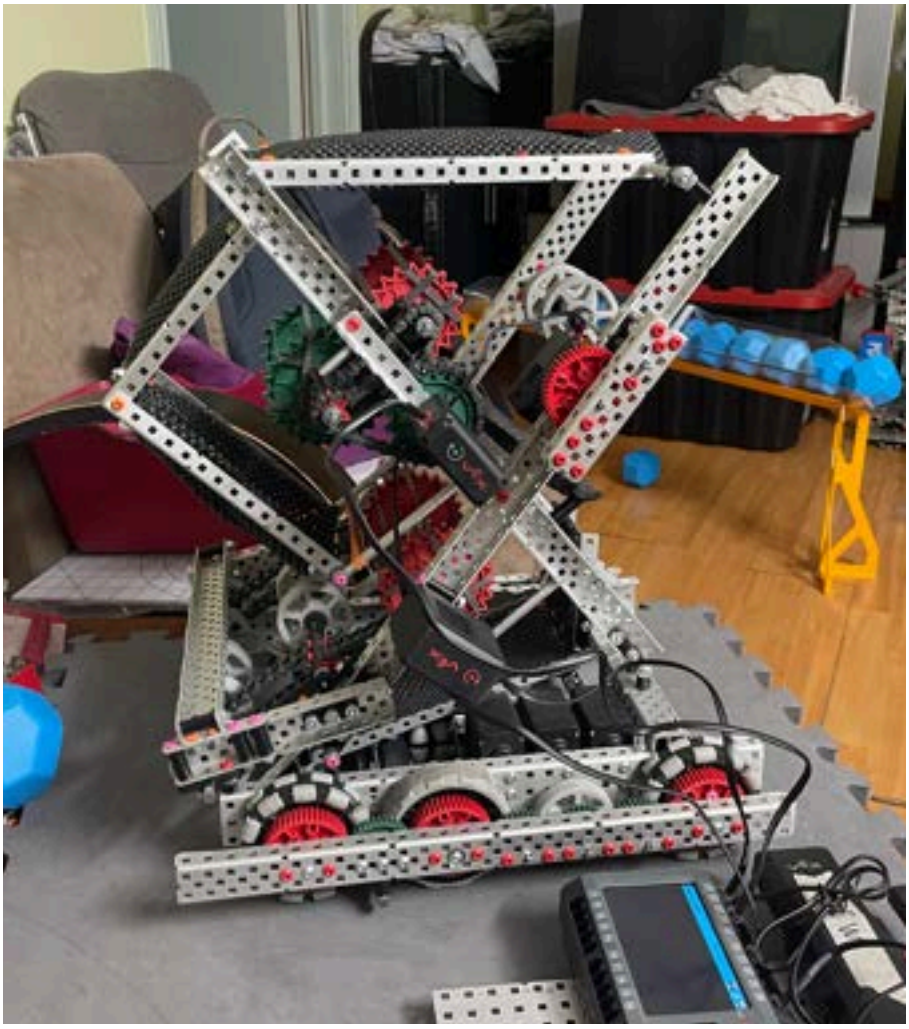


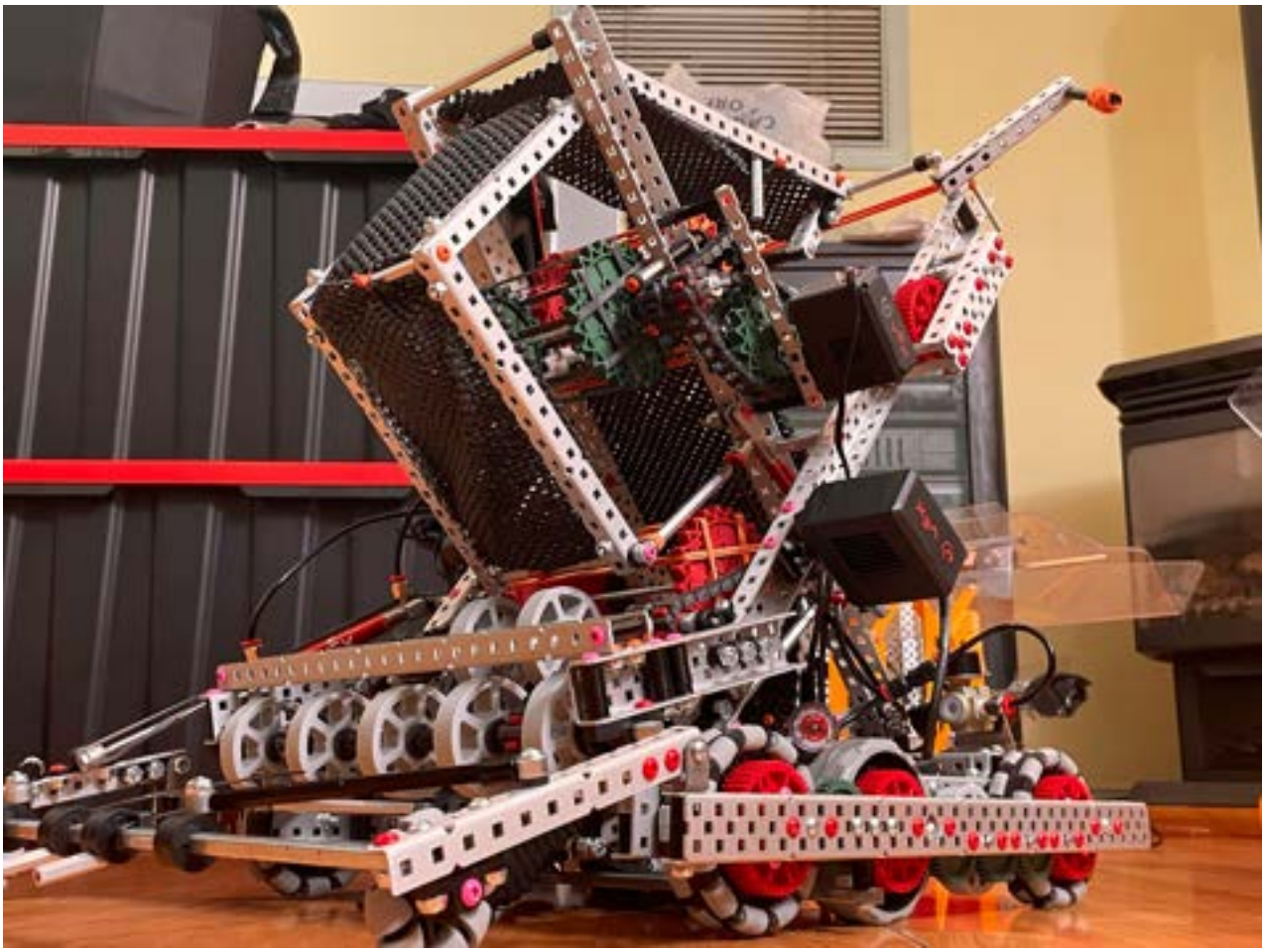












INTAKE TESTS

Laksan Mohan | 09/18/2025

Now that our intake is built, we will come up with some tests and methodology. Once our full robot is complete, we will conduct these tests and record the results in a separate entry.

TEST #1: CYCLE TIME

Objective
Measure how many balls the intake can move from the front entry to the back scoring position per unit time (balls / minute).

Methodology:

- Place 10 balls spaced evenly in front of the intake entry.
- Run the intake system at standard autonomous speed and let the robot feed and score (or deposit at the back) continuously.
- Record the time from the moment the first ball crosses the intake plane to the moment the last ball reaches the back scoring location.
- Repeat 3 runs and calculate average balls/minute and standard deviation.
- Note any stalls or slower sections in the S-curve.

Test Metrics: balls per minute, variance, and any pauses.

TEST #2: SINGLE-BALL TRANSFER RELIABILITY

Objective:
Confirm a single ball reliably travels through the entire S-path and arrives at the back scoring location without reorientation or jamming.

Methodology:

- Place one ball centered at the intake opening.
- Start a recording on your phone and run the intake at 75% power and observe travel through two S turns until scoring position.
- Use the video to mark time to travel and note orientation changes and contact points.
- Repeat 15 times.
- Calculate success rate (arrives without jam or manual reset).

Test Metrics: success rate (%), average transit time, common failure points.

TEST #3: MULTI-BALL PACKING

Objective:
Evaluate behavior when balls enter the intake back-to-back and measure spacing control (to avoid double-feeds or jams).

Methodology:
- Feed 2, 3, and 5 balls into the intake at short intervals (0.5 s, 1.0 s, 1.5 s gaps).
- Run the intake and record occurrences of double-feeding, bridging, or queue collapse inside the S path.
- For each spacing, do 5 trials.
- Note the minimum safe spacing that yields ≥90% success.

Test Metrics: minimum spacing (s), failure rate per spacing.

TEST #4: JAM RECOVERY & BACKDRIVE

Objective:
Test how the system responds to a jam and whether backdrive or reverse can clear it reliably.

Methodology:
- Intentionally induce a jam by inserting a ball at an angle or by using a slightly deformed ball.
- Observe whether sensors detect stall/overcurrent.
- Execute programmed jam-recovery routines (stop → reverse X ms → forward) and/or manual intervention.
- Perform 10 induced jams and record recovery success and required intervention (automatic vs manual).

Test Metrics: automatic recovery success rate (%) and average recovery time.

TEST #5: SCORING CONSISTENCY

Objective:
Verify the ball arrives at the back in a consistent orientation/position for scoring mechanism to accept it without extra manipulation.

Methodology:
- Run 20 single-ball transfers.
- At the scoring exit, measure the ball's position relative to the goal

(centerline and depth ± mm) and note any rotational orientation if relevant.
- Log how many required follow-up correction (small nudge) to score.

Test Metrics: % within tolerance, mean offset, number of corrections needed.

TEST #6: SENSOR & DETECTION ACCURACY

Objective:
Validate intake sensors (optical sensor) correctly detect ball presence, transit, and jams in the S-path.

Methodology:
- Install sensors at entry, mid-S, and exit positions.
- Pass a ball at normal speed 20 times and log sensor triggers vs actual ball position (using video for ground truth).
- Intentionally create partial passes and borderline cases to test false positives/negatives.
- Record latency between real crossing and sensor reading.

Test Metrics: detection accuracy (%), false positive/negative rates, latency (ms).

TEST #7: SPEED VS. RETENTION

Objective:
Find the best motor power settings (entry rollers vs mid vs exit) that maximize throughput while preventing ball slip or ejection.

Methodology:
- Test 9 combinations of entry/mid/exit roller powers (low/med/high; e.g., 50/75/100%).
- For each combination, run 10 balls sequentially and observe slip, change of spacing, or ejection.
- Rate each combo on throughput and retention quality.

Test Metrics: ranked power profiles, recommended nominal settings.

TEST #8: HEAT & ENDURANCE

Objective:
Assess thermal behavior and mechanical wear of rollers, motors, and any belts/chains during extended operation.

Methodology:

- Run the intake continuously for 20 minutes while continuously feeding balls at a steady rate.
- Log motor temperatures via the brain at 2-minute intervals and inspect rollers and guides at 5-minute intervals for wear/looseness.
- Stop sooner if motors exceed safe temp thresholds.
- Repeat once after a cool-down to check repeatability.

Test Metrics: temperature curve, any component degradation, safe run-time limit.

TEST #9: NOISE, VIBRATION & STRUCTURAL LOAD

Objective:
Detect sources of unacceptable vibration or noise that might affect accuracy or longevity.

Methodology:

- Run intake at nominal speed with and without balls.
- Use smartphone accelerometer app attached to chassis to log vibration at 3 locations (front, mid-S, back).
- Listen and record audible noise; note any rattles or resonance.
- Identify component frequencies that coincide with vibrational peaks.

Test Metrics: vibration amplitude (g), problematic frequencies, recommended damping fixes.

TEST #10: INTERFERENCE & FIELD-ROBUSTNESS

Objective:
Measure how external interactions (hits from other robots, accidental pushes) affect ball retention and flow in the S-path.

Methodology:

- While running the intake, apply lateral pushes to the chassis equal to light and medium contact (simulated by dropping a 1 pound weight).
- Also run a scenario where another robot lightly nudges the back of the robot while feeding.
- Record any spillage, misfeeds, or jams; test recovery.

Test Metrics: resilience score (no fail / minor fail / major fail) per contact severity.

TEST #11: MATERIAL & SURFACE VARIATION

Objective:
Test intake performance across balls of slightly different surface textures or wear (slick, scuffed, slightly dented).

Methodology:
- Acquire samples of balls representing expected variation.
- Run 10 transfers for each ball type and log slippage, transit time, and jams.
- Note which surfaces require design changes (roller surface, pressure, or geometry).

Test Metrics: success rate per ball type, suggested material adjustments.

MATCHLOADER DESIGN BRIEF

Laksan Mohan | 10/01/2025

The matchloader must be able to reliably interface with the match load station and guide game elements into the robot's intake system in a consistent and repeatable manner. It must ensure that blocks enter the robot in a predictable position and orientation, minimize jamming caused by angled approaches, and reduce variability during both match play and skills runs. A well-designed matchloader will allow the intake system to be tuned aggressively for speed while maintaining alignment and consistency throughout repeated matchloading cycles.

PROBLEM STATEMENT

In Push Back, the match load station is a critical scoring resource. However, inconsistent approach angles and impact forces can cause blocks to enter the robot unevenly, resulting in jams, reduced intake speed, and loss of alignment. The matchloader must correct for these inconsistencies by mechanically enforcing repeatable contact with the match load station, ensuring fast, reliable block transfer without disturbing the robot's heading or downstream scoring routines.

CONSTRAINTS

- Must fit within an 18" x 18" x 18" volume per R5
  - Can expand to 22" in all directions after the start of the game
- Must fit within our budget of $2000 for the full robot
- Can only be built with VRC-legal components per R17-R26, with some legal modifications per R27 and R28
- Must integrate with the existing intake and preroller systems
- Must withstand repeated high-speed impacts with the match load station

OBJECTIVES

- Consistently guide blocks straight into the intake system
- Allow matchloading of all available blocks in minimal time
- Minimize jamming caused by angled or imperfect approaches
- Mechanically enforce repeatable robot positioning at the match load station

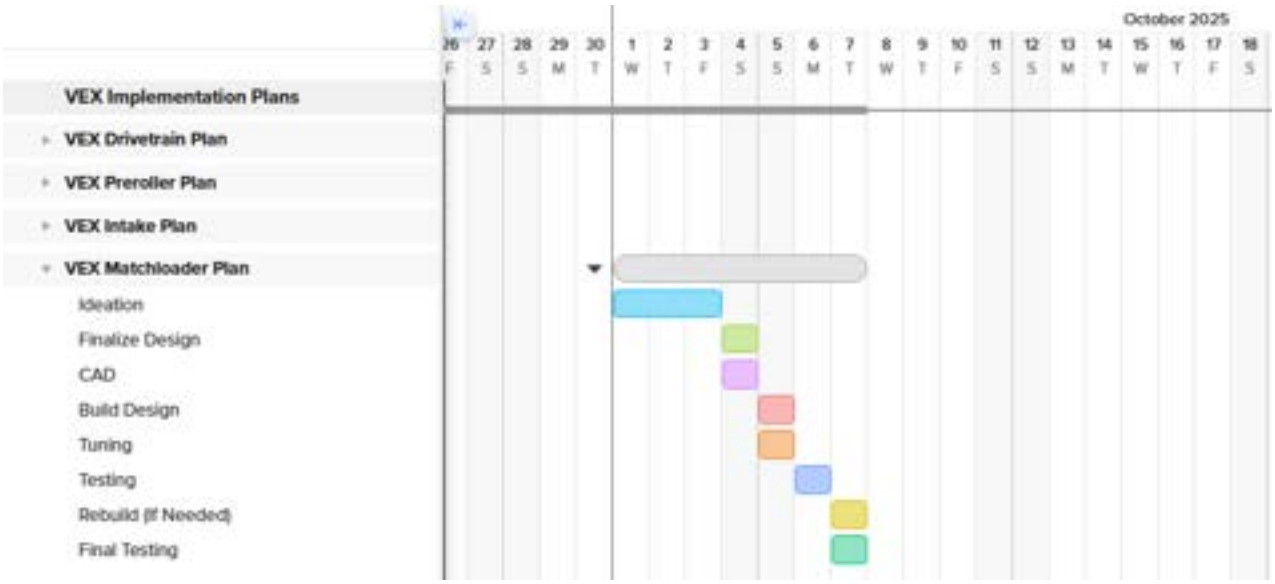
- Enable higher intake and preroller speeds through improved consistency
- Maintain robot heading during entry and exit from the match load station
- Support reliable operation during both driver control and skills runs

The matchloader serves as the interface between the robot and the match load station, ensuring that blocks are transferred into the intake system efficiently and predictably. Because matchloading is often performed under time pressure and imperfect alignment, the matchloader must be designed to reduce reliance on driver precision by mechanically correcting positioning and guiding block entry. This subsystem plays a critical role in improving cycle time, intake reliability, and overall scoring consistency.

TIMELINE

The matchloader design will be finalized after entire intake has been completed. In the interim, the team will brainstorm concepts, analyze proven designs, and CAD potential solutions. Once the final design is selected, construction and integration will proceed rapidly, followed by testing and tuning to ensure optimal interaction with the intake and preroller systems.

To guide our design cycle of the matchloader, we have included a timeline below:



MATCHLOADER OPTIONS

Laksan Mohan | 10/03/2025

OPTION #1: SIDE ROLLER INTAKE

Figure #1: Side Roller Prerollers by 94999E from VEX Change Up 2021

The side roller intake uses two powered rollers mounted on either side of the intake opening to actively pull blocks from the match load station into the robot. As the robot enters the match load station, the rollers grip the block and guide it inward, functioning similarly to a traditional intake but optimized for matchloading.

PROS:

- Very effective and fast
  - Powered rollers aggressively pull blocks into the robot, allowing for rapid matchloading cycles.
- High tolerance to minor misalignment
  - With sufficient compliance, the rollers can correct for small approach angle errors.
- Well-understood design
  - Similar mechanisms are commonly used, making it easier to design and tune.

CONS:

- Requires additional motors
  - Adds weight, power consumption, and wiring complexity.
- Increased mechanical and control complexity
  - Roller speeds must be carefully tuned to avoid jams or uneven feeding.
- Potential reliability concerns late in matches
  - Motors are subject to heat, brownouts, and wear over long matches.

OPTION #2: LITTLE WILL MECH

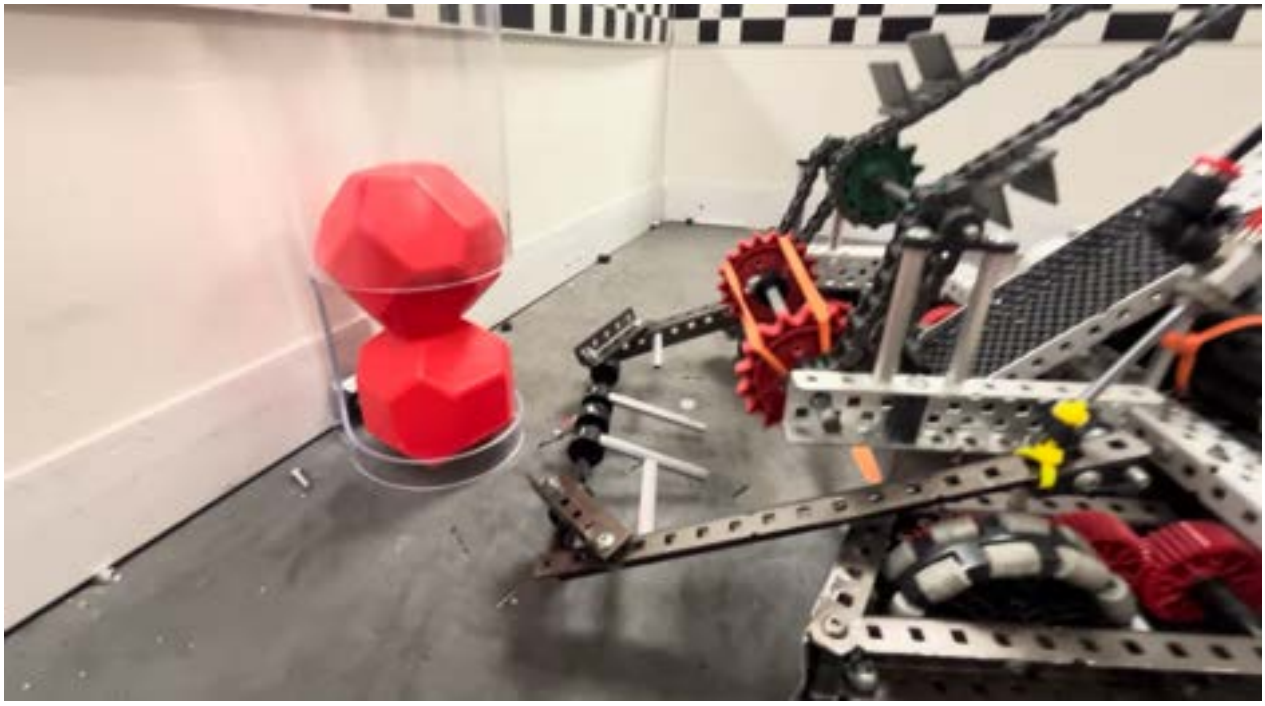


Figure #2: Little Will Mech by Little Will from Team 44252A

Inspired by Team 44252A, the Little Will Mech uses a pneumatic drop-down mechanism with angled standoffs that extend into the match load station. These standoffs create a ramp that causes blocks to slide downward and into the intake as the robot drives forward. The system relies on geometry and gravity rather than powered rollers.

PROS:

- Passive design (no motors required)
  - Reduces weight, power usage, and electrical complexity.
- Simple and lightweight
  - Fewer moving parts make the mechanism easier to build and maintain.
- Effective
  - Clean contact with the match loader results in smooth block transfer when aligned properly.

CONS:

- Inconsistent performance
  - If the robot approaches at an angle, blocks may drop unpredictably or fail to enter the intake.
- Slower than powered solutions
  - Relies on gravity and robot motion rather than active pulling, resulting in slightly slower matchloading times.
- Lower forgiveness under match pressure
  - Small errors in approach can result in failed matchloads.
- Risk of ejecting blocks from the match load station
  - Because the mechanism relies on angled standoffs and gravity, improper contact or excessive speed can cause blocks to be forced upward or outward. This may result in blocks being launched out of the match load station, which is a minor rule violation and can escalate to disqualification if repeated.

OPTION #3: PIVOTING TRAY

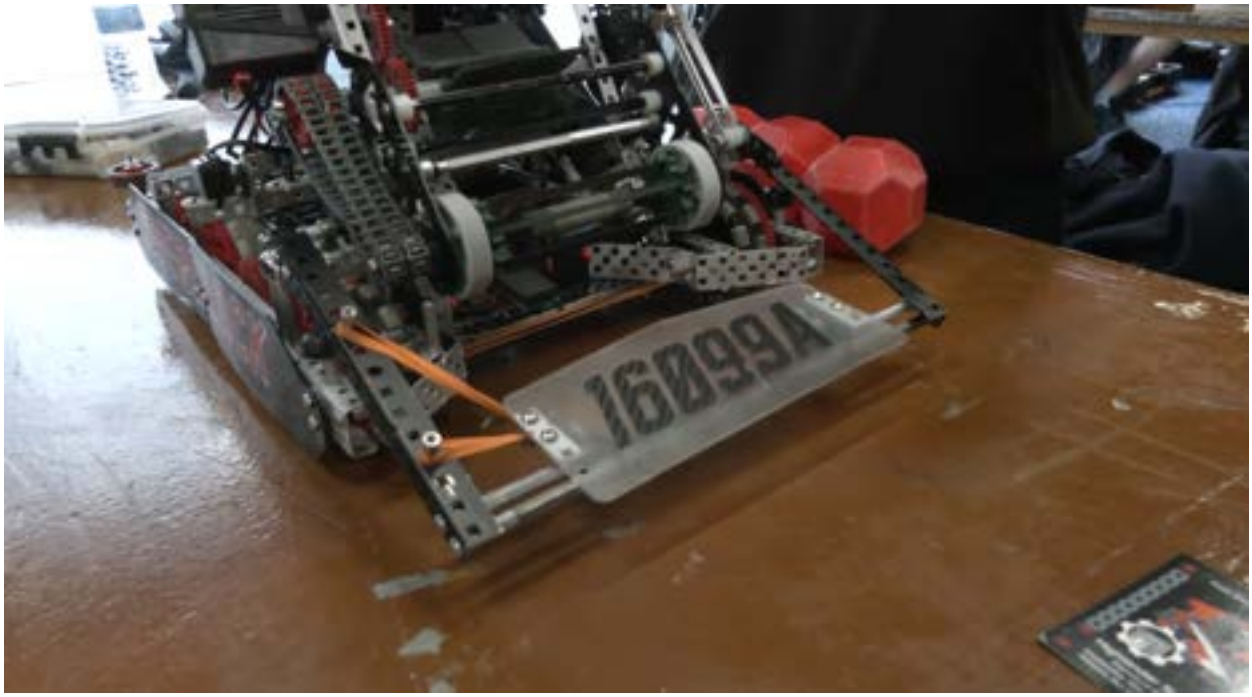


Figure #3: Pivoting Tray Matchloader by Overclock Team 16099A

Inspired by Team 16099A, the pivoting tray uses a flat plastic tray mounted on a pivot that rotates upward to match the angle of the match load station. When the robot enters the station, the tray aligns itself with the loader, forming a smooth, continuous ramp that allows blocks to slide directly into the intake regardless of approach angle.

PROS:

- Extremely fast matchloading
  - The smooth ramp enables blocks to enter the intake almost instantly.

- Works from nearly any approach angle
  - The pivoting motion self-corrects alignment, making the system highly forgiving.
- Highly consistent block entry
  - Blocks enter the robot in the same position each time, enabling aggressive intake tuning.
- Low-actuation system
  - Requires minimal actuation compared to full roller-based designs.

CONS:

- Requires precise geometry and tuning
  - The tray angle and pivot location must closely match field tolerances to ensure consistent performance.
- Increased mechanical complexity
  - Additional moving components introduce more potential wear points compared to fully static designs.
- Structural rigidity is critical
  - Any flex in the tray can reduce consistency and reliability during matchloading.
- Rare risk of block ejection
  - While significantly reduced compared to the Little Will Mech, improper tuning or excessive approach speed can very rarely cause blocks to exit the match load station. However, the guided tray geometry minimizes this risk in nearly all scenarios.

MATCHLOADER CHOICE

Laksan Mohan | 10/03/2025

To decide on an matchloader, we will be using a decision matrix. The following criteria were considered while deciding on the design.

- Matchloading Speed (X/15)
  - A fast matchloader is essential for offensive dominance.
  - The faster blocks can be transferred into the robot, the less time opponents have to interfere or respond defensively.
  - Designs that enable near-instant block transfer score higher in this category.
- Consistency & Repeatability (X/15)
  - Consistency is critical for both match play and skills runs.
  - The matchloader must ensure that blocks enter the robot in the same position and orientation every time, regardless of minor approach errors.
  - Highly repeatable systems score higher.
- Angle Tolerance (X/10)
  - During real matches, the robot cannot always approach the match load station perfectly straight.
  - A strong design must function reliably even when approaching from severe angles.
  - Designs that self-correct alignment score higher.
- Rules Safety & Compliance (X/6)
  - Any mechanism that risks ejecting blocks from the match load station can result in minor violations or disqualification if repeated.
  - Designs that minimize this risk score higher.
- Mechanical Simplicity & Reliability (X/4)
  - Simpler mechanisms with fewer failure points are easier to maintain and less likely to break during competition.
  - While performance is prioritized, reliability remains critical.
- Weight & Integration (X/3)
  - A lightweight and compact matchloader allows for better weight distribution and easier integration with the intake and drivetrain systems.
- Cost Effectiveness (X/2)
  - As we were operating on a severely limited budget, we also wanted to find designs that wouldn’t eat up too much of our budget, allowing for the purchase of other parts.

Design Comparison

Option 1: Side Roller Intake

- Strengths: Very fast and aggressive intake using powered rollers
- Weaknesses: Requires motors, increases complexity, and adds electrical and tuning overhead

Option 2: Little Will Mech

- Strengths: Lightweight, passive, and simple
- Weaknesses: Inconsistent at angles and carries a significant risk of ejecting blocks from the match load station, which can lead to rule violations or disqualification

Option 3: Pivoting Tray

- Strengths: Extremely fast, highly angle-tolerant, and very consistent
- Weaknesses: Requires precise geometry and careful structural reinforcement

Decision Matrix

	Side Rollers	Little Will Mech	Pivoting Tray
Matchloading Speed (X/15)	12	10	14
Consistency (X/15)	11	9	15
Angle Tolerance (X/10)	7	3	10
Rules Safety (X/6)	6	2	5
Simplicity & Reliability (X/4)	3	4	3
Weight & Integration (X/3)	1	2	2
Cost Effectiveness (X/2)	1	2	2
Total (X/55)	41	32	**51**

After evaluating each matchloader design using the decision matrix, our team has decided to move forward with the **Pivoting Tray matchloader**. This design was selected for its **exceptional speed, high consistency, and superior angle tolerance**, which are critical for both match play and skills runs.

The pivoting tray allows blocks to enter the robot in a predictable and repeatable manner, enabling aggressive intake tuning while maintaining reliability. Additionally, its guided geometry significantly reduces the risk of ejecting blocks from the match load station, improving rules compliance compared to other passive designs. While it requires careful tuning and structural reinforcement, its performance advantages far outweigh the added complexity, making it the best overall solution for our robot's matchloader system.

MATCHLOADER SPECS

Laksan Mohan | 10/04/2025

The objective of the match loader mechanism is to reliably extract Match Load Blocks from the Match Loader Station and immediately hand them off to the intake system, while withstanding repeated high-force impacts with the field structure. Because Match Loads are introduced dynamically and often under defensive pressure, the mechanism must be robust, height-consistent, and tolerant of imperfect driver alignment.

DECISION #1: MATCHLOADER LENGTH

The matchloader is designed to be **11 holes long**, which was selected as the optimal length through onfield testing.

This length is critical because it allows pre-roller wheels to make contact with the Blocks immediately as they exit the Match Loader Station opening. By ensuring early roller engagement, Blocks are stabilized and guided into the intake before they can bounce, rotate, or fall unpredictably. This reduces cycle time and minimizes jams caused by blocks landing flat or skewed.

The 11-hole length also avoids unnecessary overextension, keeping the mechanism compact while fully covering the effective depth of the Match Loader opening.

DECISION #2: PIVOT POINT AND MOUNTING METHOD

The match loader pivots using **piston mounts from the legacy VEX pneumatic kit**, repurposed as the rotational mounting points for the mechanism. These mounts were selected because they are exceptionally compact, low-profile, and structurally rigid, making them ideal for use as a pivot in a space-constrained subsystem.

Using these mounts allows the pivot point to sit very close to the robot frame, reducing unnecessary offset and minimizing the moment arm acting on the structure during deployment and retraction. This directly lowers stress on the mounting hardware when the match loader is pressed into the Match Loader Station or experiences defensive contact.

Additionally, the small footprint of the piston mounts simplifies packaging and prevents interference with nearby mechanisms, enabling a clean and mechanically efficient pivot design.

DECISION #3: OVERALL WIDTH

Because the pivot solution is so compact, the match loader is able to span the **full width of the robot: 30 holes (15 inches)**. This full-width design provides several advantages:

- It maximizes alignment tolerance with the Match Loader Station, making it easier for drivers to engage the station even when approaching at slight angles.
- It ensures consistent block interaction across the entire opening, reducing the likelihood of Blocks slipping past the tray edges.
- Despite the increased width, the match loader does not interfere with any other robot subsystems, due to careful placement of the pivot point and internal packaging of the intake, drivetrain, and structural elements.

The full-width configuration improves robustness and ease of use without introducing mechanical conflicts elsewhere on the robot.

DECISION #4: STRUCTURAL REINFORCEMENT

To ensure durability, the match loader uses **standoff-based triangular bracing** along the sides of the tray. This triangulated structure greatly improves resistance to bending and torsional loads compared to a flat or cantilevered design.

The front of the match loader is subjected to repeated forces from:

- Aggressive driver alignment into the Match Loader Station
- Incidental defensive contact
- Continuous compressive loading during block transfer

Triangular bracing distributes these loads across the structure, preventing localized bending and maintaining alignment over the course of a tournament.

DECISION #5: GROUND REFERENCE AND HEIGHT CONTROL

Two 2-inch omni wheels are mounted, one on each side of the match loader. These wheels act as a mechanical height reference, ensuring consistent deployment geometry.

When deployed, the omni wheels maintain a constant height off the ground, positioning the tray just above the lip of the Match Loader Station opening. This height allows Blocks to fall cleanly onto the tray without catching or rebounding, while still allowing the pre-rollers to immediately capture and guide the Blocks inward.

DECISION #6: HS AXLE BRACE AND IMPACT RESISTANCE

A high-strength axle spans the entire front of the match loader, serving as both a structural brace and a mounting point for the tray plastic.

This axle was selected over standoffs because the front of the match loader experiences constant compressive forces when the robot rams into the Match Loader Station. Testing and analysis showed that standoffs would bend or deflect under these conditions, while a high-strength axle provides the necessary bending stiffness to maintain alignment.

By tying both sides of the mechanism together, the axle ensures even load distribution and long-term structural reliability.

DESIGN SUMMARY AND ADVANTAGES

This match loader design provides the following key advantages:

- Compact and robust pivot using pneumatic piston mounts
- Full-width (15”) coverage without subsystem interference
- Immediate block control through early pre-roller contact
- Consistent height control via omni wheels
- High resistance to bending and fatigue from repeated impacts

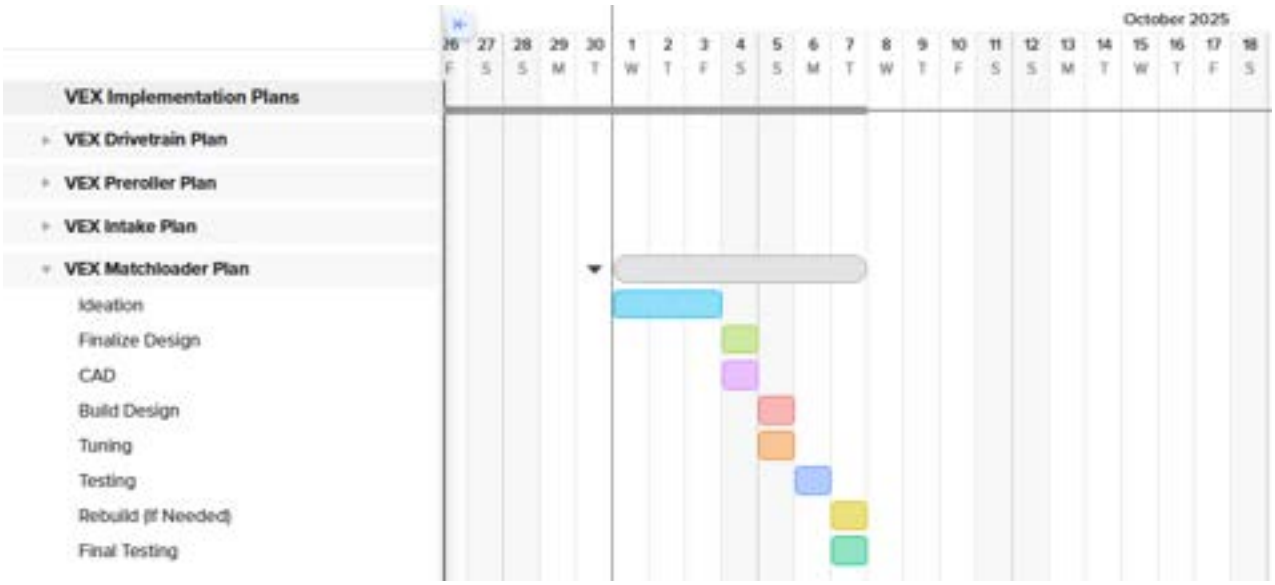
Overall, the match loader is optimized for fast, repeatable match-load cycles, supporting an aggressive offensive Push Back strategy while maintaining mechanical reliability across an entire competition season.

MATCHLOADER IMPLEMENTATION PLAN

Laksan Mohan | 10/04/2025

Our first competition is fast approaching, so we hope to complete this mechanism by the end of next week.

To guide our construction of the matchloader, we have included a timeline below:



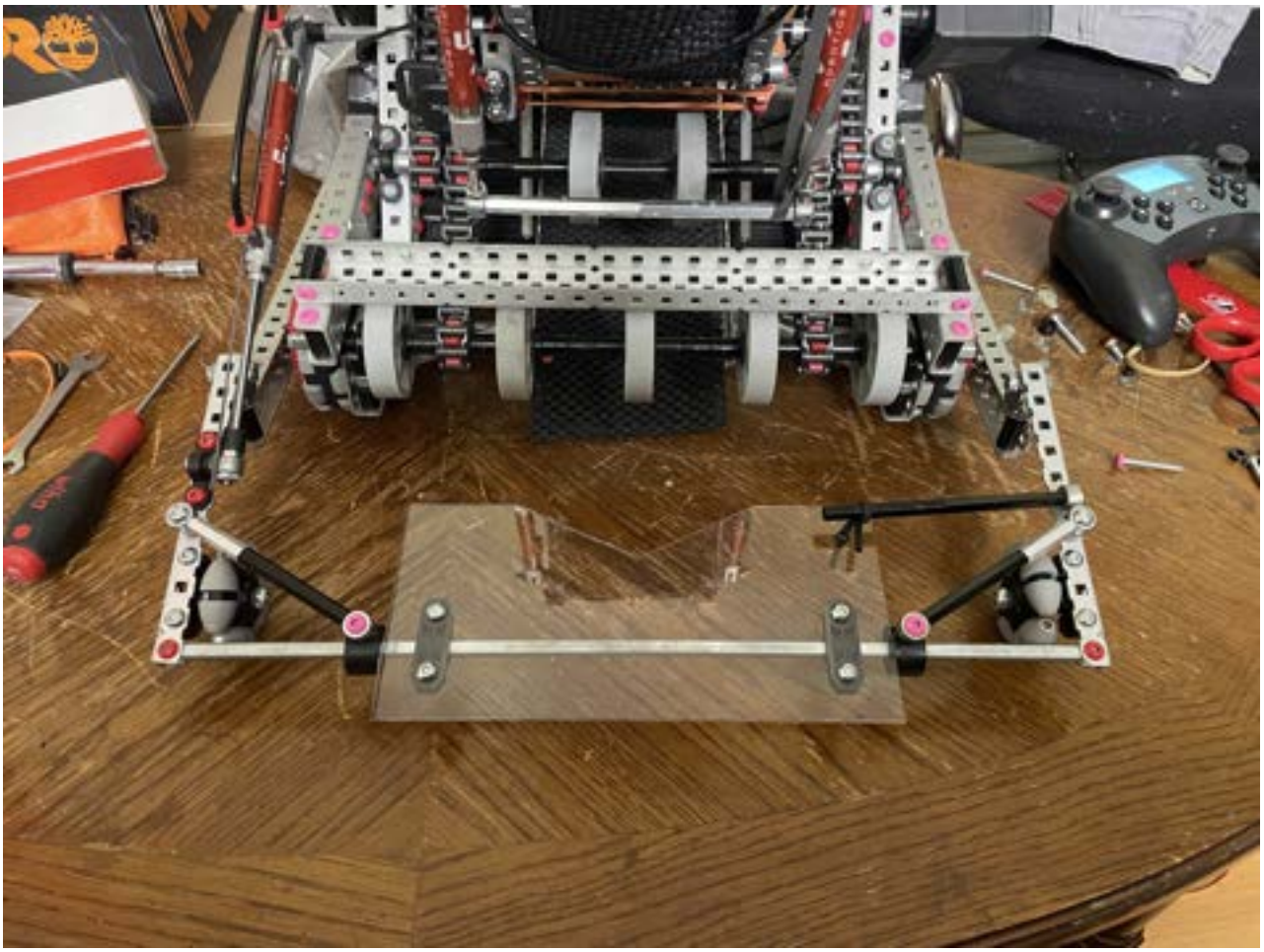
MATCHLOADER BUILD LOG

Laksan Mohan | 10/06/2025

This section covers our entire build process from October 5th, 2025 to October 6th, 2025

TEAM MEMBERS INVOLVED

- Laksan



Note on Documentation:
For this build log, we only took one photograph because the majority of the construction work occurred during short, time-compressed working sessions where the priority was staying on schedule and maintaining safety. We recognize that more visual documentation would have been ideal and will plan for a clearer photo-capture process in future builds.

MATCHLOADER TESTS

Laksan Mohan | 10/08/2025

Now that our matchloader is built, we will come up with some tests and methodology. Once our full robot is complete, we will conduct these tests and record the results in a separate entry.

TEST #1: MATCH LOAD TRANSFER RELIABILITY

Objective
Confirm that Match Load Blocks consistently fall onto the match loader tray and are successfully handed off to the intake without jamming, bouncing out, or misalignment.

Methodology:
- Place the robot against the Match Loader Station at a standard driver alignment angle.
- Introduce one Match Load Block at a time from the station.
- Run the pre-rollers and intake at full speed.
- Observe whether the block:
  - Lands cleanly on the tray
  - Is immediately captured by the pre-rollers
  - Transfers fully into the intake system
- Repeat for 20 trials.

Test Metrics:
- Successful transfer rate (%)
- Number of jams or block ejections
- Average time from block release to intake capture

TEST #2: ALIGNMENT TOLERANCE

Objective
Evaluate how tolerant the full-width (15”) match loader is to imperfect driver alignment.

Methodology:
- Approach the Match Loader Station at three angles:
  - Head-on
  - ±5° offset

    - ±10° offset
- For each angle, perform 5 match load attempts.
- Record whether blocks successfully land on the tray and enter the intake.

Test Metrics:

- Success rate per approach angle
- Maximum misalignment angle with ≥90% success

TEST #3: HEIGHT CONSISTENCY & GROUND REFERENCE

Objective

Verify that the omni-wheel ground reference maintains the correct deployment height across varying field conditions.

Methodology:

- Deploy the match loader on:
    - New tiles
    - Slightly worn tiles
    - Tile seams
- Observe the vertical position of the tray relative to the Match Loader Station lip.
- Run 10 match load cycles per surface condition.

Test Metrics:

- Tray height variation (qualitative or measured)
- Transfer success rate across surfaces

TEST #4: STRUCTURAL IMPACT & FATIGUE TEST

Objective

Ensure the match loader structure withstands repeated ramming into the Match Loader Station without deformation or loss of alignment.

Methodology:

- Drive the robot into the Match Loader Station at normal match speed.
- Deploy the match loader and hold contact pressure for 3 seconds.
- Retract and repeat for 30 cycles.
- Inspect:
    - Front high-strength axle
    - Pivot mounts (pneumatic piston mounts)
    - Triangular standoff bracing

Test Metrics:

- Visible bending or deflection (yes/no)
- Change in deployment smoothness
- Hardware loosening or wear

TEST #5: PIVOT SMOOTHNESS & DEPLOYMENT RELIABILITY

Objective

Confirm that the compact piston-mount pivot provides smooth, repeatable deployment without binding.

Methodology:

- Deploy and retract the match loader 25 consecutive times.
- Perform the test both:
  - Stationary
  - While moving
- Observe motion consistency and actuator load (if applicable).

Test Metrics:

- Deployment success rate
- Any binding or asymmetrical motion
- Change in deployment time over repetitions

TEST #6: CYCLE TIME CONTRIBUTION

Objective

Measure how much the match loader improves overall match load cycle time.

Methodology:

- Time the sequence from:
  - Initial contact with Match Loader Station
  - To block fully entering the intake
- Perform 10 trials and compute the average.
- Compare against a baseline run with slower or partial alignment.

Test Metrics:

- Average match load cycle time (s)
- Variance across trials

TEST #7: INTERFERENCE & SUBSYSTEM COMPATIBILITY

Objective
Verify that the full-width match loader does not interfere with other robot subsystems during operation.

Methodology:
- Deploy the match loader while:
  - Driving
  - Running the intake
  - Turning in place
- Observe for collisions, wiring strain, or motion restriction.

Test Metrics:
- Interference observed (yes/no)
- Clearance margins (qualitative)

TEST #8: DEFENSIVE CONTACT ROBUSTNESS

Objective
Assess match loader performance under light defensive contact.

Methodology:
- While the robot is pressed into the Match Loader Station, apply lateral force to the chassis (simulated by a light push).
- Attempt a match load during contact.
- Repeat 10 times.

Test Metrics:
- Transfer success under contact (%)
- Any block loss or misfeed

SUMMARY & VALIDATION CRITERIA

The match loader will be considered **competition-ready** if it meets the following benchmarks:
- ≥95% match load transfer success rate
- No permanent structural deformation after impact testing
- Smooth, repeatable deployment using piston-mount pivots
- No interference with other subsystems
- Reliable performance across alignment and surface variations

These tests validate that the match loader is robust, driver-tolerant, and optimized for high-speed Push Back gameplay.

ALIGNER DESIGN PROCESS

Laksan Mohan | 10/15/2025

PROBLEM STATEMENT

In VEX Push Back, accurately and repeatedly aligning the robot to the Long Goal is critical for both scoring and descoring. The Long Goal opening is narrow relative to robot width, and even small angular misalignments can result in:

- Blocks missing the goal opening
- Increased cycle time due to driver correction
- Reduced effectiveness when attempting to descore opponent Blocks

Under match conditions, especially while under defensive pressure, relying solely on driver precision is unreliable. Therefore, a passive mechanical solution was needed to self-align the robot to the Long Goal using field geometry.

CONSTRAINTS

- Must fit within an 18” x 18” x 18” volume per R5
  - Can expand to 22" in all directions after the start of the game
- Must fit within our budget of $2000 for the full robot
- Can only be built with VRC-legal components per R17-R26, with some legal modifications per R27 and R28

CRITERIA

- **Self-centering:** Automatically correct small angular and lateral misalignments
- **Geometry-matched:** Conform precisely to the Long Goal opening
- **Passive operation:** Require no motors, sensors, or driver input
- **Compact and lightweight:** Avoid interference with other subsystems
- **Robust:** Withstand repeated contact with field elements

CONCEPT DEVELOPMENT

Several alignment concepts were evaluated:

1. **Flat bumper-style aligner**
   - Simple but provides no self-centering ability

2. **Curved funnel aligner**
    - Self-centering but difficult to manufacture accurately and inconsistently engages the goal edges
3. **Trapezoidal geometry aligner**
    - Uses angled faces to guide the robot into alignment
    - Matches the rectangular opening of the Long Goal

After evaluation, the **trapezoidal aligner** was selected as the optimal solution.

FINAL DESIGN DESCRIPTION

The final aligner uses a trapezoidal shape designed to slot precisely into the geometry of the Long Goal opening.

As the robot approaches the goal:

- The angled sides of the trapezoid contact the vertical edges of the Long Goal
- Any angular or lateral misalignment generates a corrective force
- The robot is passively guided into a centered and square position

Once fully seated, the aligner prevents further lateral motion, holding the robot in an optimal orientation for both scoring and descoring actions.
The trapezoidal geometry was chosen specifically because it:

- Converts forward driving force into lateral correction
- Provides deterministic alignment rather than relying on friction
- Engages consistently even when the robot approaches at slight angles

DESIGN JUSTIFICATION

This aligner design provides several key advantages:

- **Repeatability:** Alignment accuracy is determined by geometry rather than driver skill
- **Speed:** Reduces time spent correcting position during cycles
- **Reliability:** No moving parts, motors, or sensors that could fail
- **Defensive robustness:** Maintains alignment even when lightly contacted by another robot

Because the aligner is purely passive, it adds no software complexity and does not consume additional power.

INTEGRATION WITH THE ROBOT

The aligner is positioned at the front of the robot so it engages the Long Goal before blocks are released or removed. Its compact profile ensures it

does not interfere with:

- Intake operation
- Match loader deployment
- Turning radius or drivetrain motion

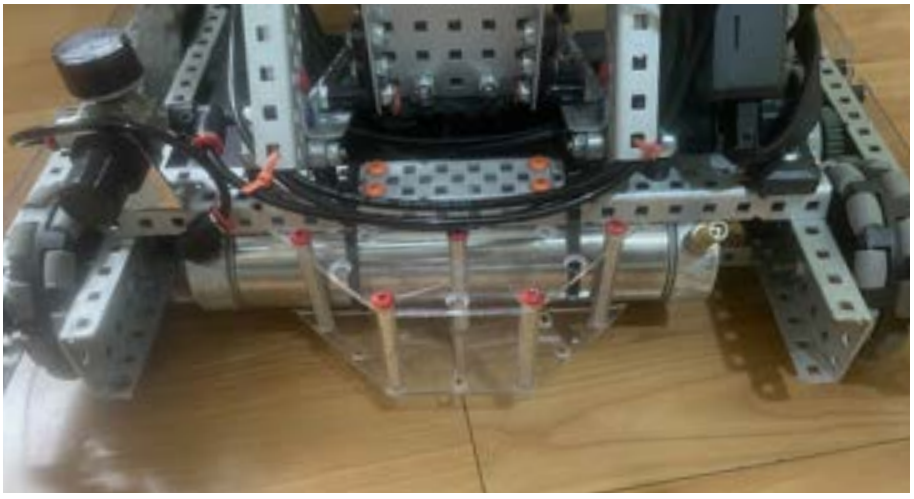
The design also allows quick disengagement by simply reversing the drivetrain into the long goal, ensuring smooth transitions between scoring, descoring, and navigation.

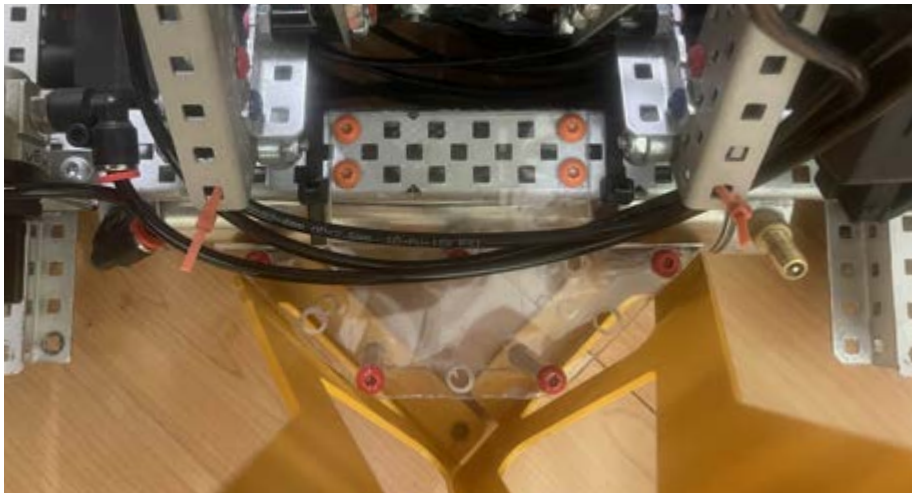
DESIGN IMPLICATIONS AND STRATEGIC IMPACT

By reducing alignment error, the trapezoidal aligner:

- Increases scoring consistency in the Long Goal
- Enables faster and more reliable descoring attempts
- Lowers cognitive load on the driver during high-pressure matches

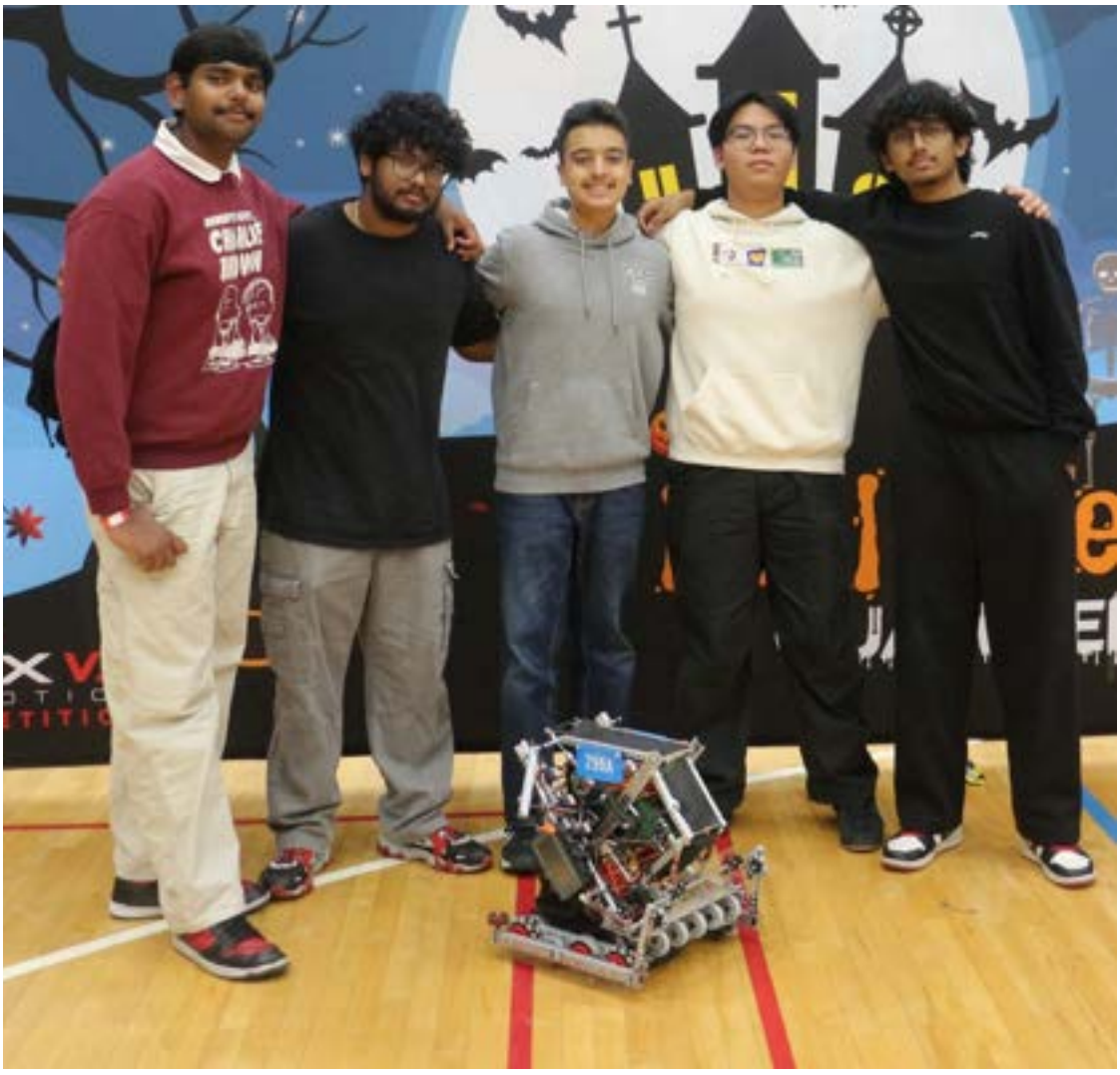
Overall, this aligner transforms a difficult, precision-based task into a repeatable mechanical interaction, improving cycle time, consistency, and match performance.





CAUTION TAPE QUALIFER COMPETITION REFLECTION

Laksan Mohan | 11/07/2025



The **Caution Tape Qualifier** on **November 1st** was our first competition of the season and one of the most competitive events we expect to face before Provincials. With **63 teams** in attendance, our primary objectives were to test our early-season build under real match conditions, evaluate autonomous consistency, and benchmark our performance against high-level teams.

Overall, we finished qualification rounds **ranked 16th out of 63**, **winning 4 of our 6 matches**. Our autonomous routine performed reliably throughout the day and successfully scored in nearly every match, validating both our code structure and tuning approach. We also posted a **combined driver +**

programming skills score of 58, **placing 11th** at the event. These results confirmed that our competitive baseline is strong and that our robot is capable of performing at a high level when everything functions as intended.

However, the event also highlighted several weaknesses that we need to address. In one of our qualification losses, our matchloader string snapped mid-match, leaving us unable to capitalize on our offensive strategy. In another loss, we were out-positioned after losing autonomous, and our robot spent most of the match being guarded and physically pushed, severely limiting our cycle efficiency. These matches demonstrated the need for stronger defensive counter-strategies and more robust subsystem durability.

During eliminations, we partnered with **Stratford Robotics (29295A)**. Although our strategy going into the match was sound, their intake overheated, significantly limiting alliance scoring potential. We were ultimately eliminated in Round of 16. While this was disappointing, we learned the importance of proactive alliance communication, equipment checks, and building contingency strategies around partner limitations.

A major learning moment from this event came from the judged awards. Despite submitting a notebook, we did not receive recognition. This showed us that while our build process is thoughtful, our documentation did not yet clearly demonstrate iterative engineering, decision-making rationale, or testing validation to the standard judges expect. This feedback reinforced that documentation needs to be treated as an engineering task, not an afterthought. For our next event, we will plan to polish up our notebook and keep it up to date.

KEY STRENGTHS OBSERVED

- Autonomous routine was consistent and reliable throughout the day.
- Robot performed competitively against high-caliber teams.
- Skills score (58) placed 11th and validates robot scoring potential.
- Match strategy and communication during qualifiers were generally effective.

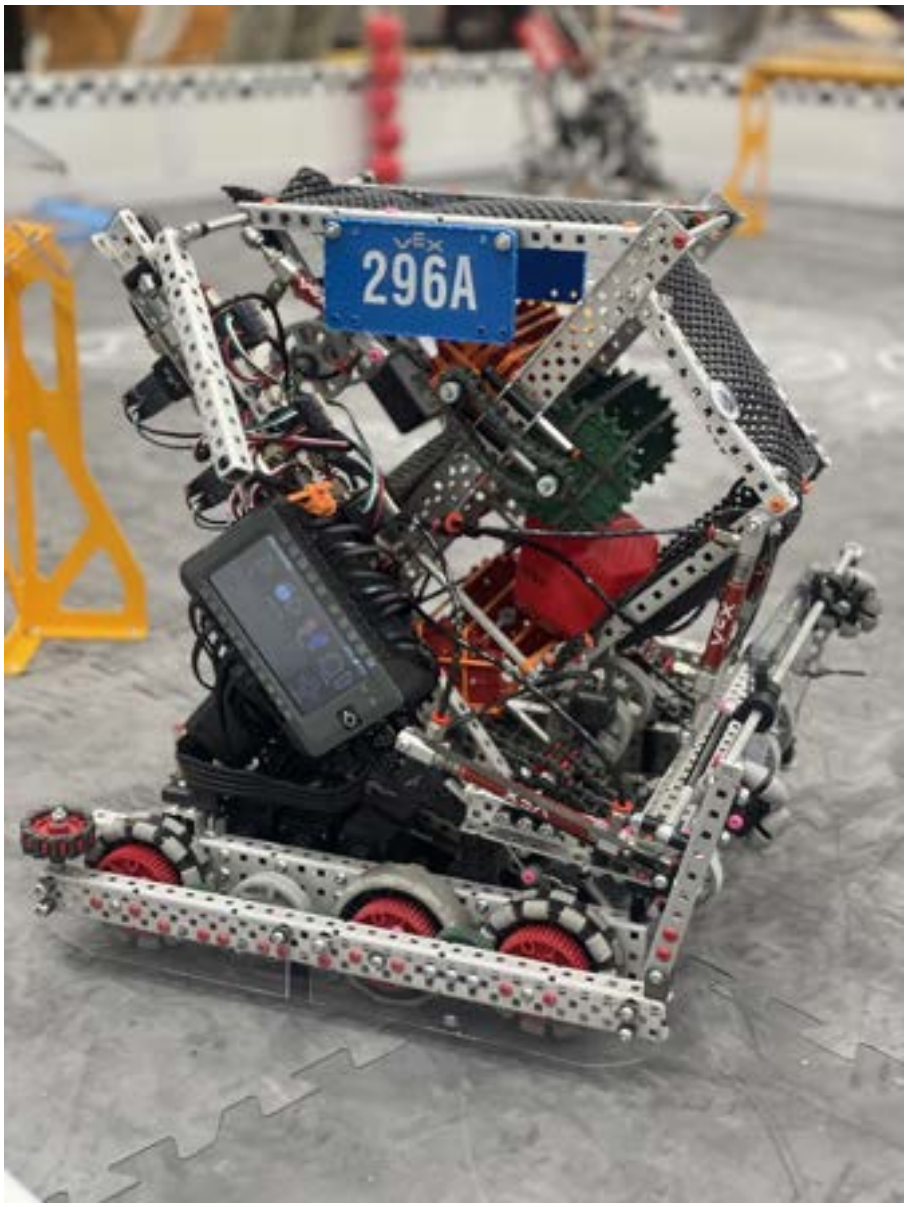
AREAS REQUIRING IMPROVEMENT

- Mechanical reliability, especially matchloader durability components.
- Ability to play through heavy defense and maintain scoring productivity.
- Alliance adaptability when partners experience mechanical failures.
- Notebook clarity, depth of engineering justification, and iteration tracking.

ACTION ITEMS MOVING FORWARD

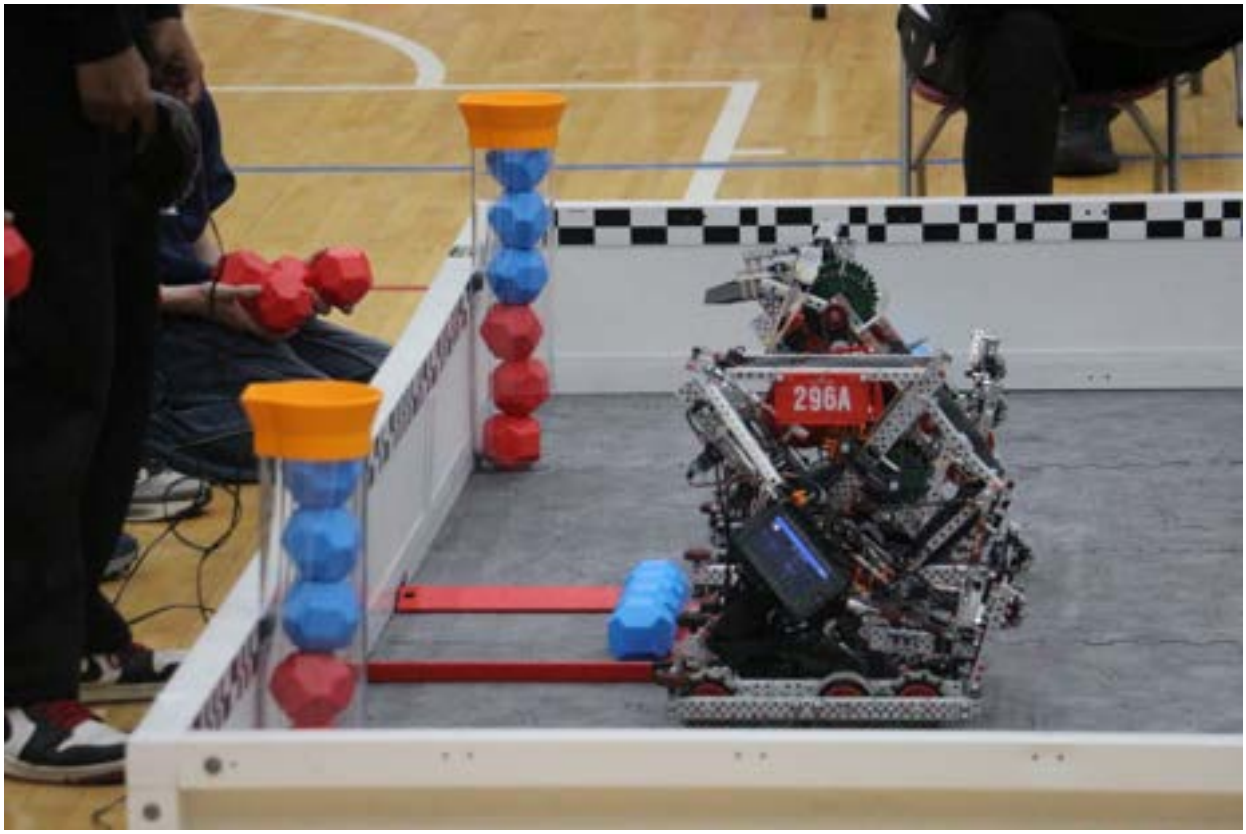
1. Reinforce and redesign matchloader components to prevent future failures.
2. Develop defensive counter-strategies and driver practice focused on escaping pins and pressure.
3. Implement pre-match mechanical checklist protocols with alliance partners.
4. Revamp our engineering notebook to better document testing, iterations, and design rationale.
5. Improve combined skills score to at least 100.

In summary, the **Caution Tape Qualifier** confirmed that our robot has strong competitive potential, but also exposed gaps that must be addressed if we want to contend at Provincials. The experience motivated us to refine both our robot and our documentation practices, and it provided a clear roadmap for improvement going into the remainder of the season.



Back 296A Match List

Q 3 10:02 AM	296A 1104V	45	73	29295A 2486M
Q 17 10:49 AM	82000B 296A	21	94	82855Z 16610Z
Q 40 12:36 PM	3150A 61119D	57	59	14749K 296A
Q 50 1:04 PM	715A 1140X	38	105	7614A 296A
Q 71 2:09 PM	55580A 17924M	26	118	296A 252A
Q 87 3:01 PM	58676R 296A	91	15	95690A 2055A
R16 8-1 4:29 PM	55288A 2055A	50	36	296A 29295A



ST. CATHERINES QUALIFER COMPETITION REFLECTION

Laksan Mohan | 11/29/2025

The **St. Catharines Qualifier** on **November 22nd** represented our second major event of the season and an opportunity to test whether the improvements made after the Caution Tape event would translate into competitive results. This competition ultimately became one of our strongest performances to date and provided several critical learning moments for both technical development and tournament strategy.

We completed qualification rounds with a **6–1 record**, **ranking 6th overall**. Our driver, Shaan, performed consistently at a high level across every match, showing strong field awareness and efficient cycling under pressure. A highlight was **Qualification Match 37**, where our alliance partner did not arrive. Despite effectively playing 1-vs-2 and being defended throughout the

match, we still secured a win by **60 points**. This demonstrated that our robot can operate independently and still meaningfully control the pace of a match.

Our single loss in qualifications resulted from an unusual chain of issues. Our alliance partner's robot was powered off at the start, leaving us alone. At the same time, one of our odometry pods had bent without our noticing, which caused our autonomous routine to mis-track and lose the autonomous bonus. During driver control, the opposing alliance boxed us in repeatedly. They exceeded the 3-second pinning limit multiple times, but it was not enforced. Even so, we continued scoring and nearly completed a comeback but ran out of time before we could score our filled robot. From this match, we learned the importance of frequent mechanical inspections, clear rules communication, and remaining composed even when officiating does not go our way.

In eliminations, we formed an alliance with **Discobots Venom (1104V)** and felt confident in our strategy. However, during autonomous our controller disconnected momentarily, causing the robot to continue rolling forward after the command ended and cross the autonomous line. Even though we went on to outscore the opposing alliance by 80 points during driver control, we were disqualified after the match. We later learned that eliminations apply a strict zero-tolerance policy for minor violations, regardless of intent. Our appeal was denied because it was submitted after the official review window. Although frustrating, this reinforced the necessity of checking controller connectivity before every match and understanding the procedural timelines for ruling challenges.

Our performance in Skills was a major highlight. Shaan achieved a **62 in driver skills**, while Glae posted a **47 in autonomous skills**, resulting in a **combined total of 102**. This nearly doubled our previous competition score, placed us more than **36 points** ahead of the next highest team, and ranked us within the **Top 200 teams worldwide**. This validated the effort invested in repeatable autonomous scoring and disciplined driver practice.

At the awards ceremony, we received both the **Innovate Award** and the **Robot Skills Champion Award**. The Innovate Award recognized the uniqueness of our robot design and the quality of documentation in our notebook. In contrast to our experience at the earlier qualifier, this confirmed that our revisions to documentation, especially around design justification, iteration tracking, and testing, had significantly improved.

KEY STRENGTHS OBSERVED

- Consistent match execution and strategic decision-making during qualifications.
- Demonstrated ability to win matches even under disadvantage conditions.
- Major improvement in Skills performance, ranking top-tier internationally.
- Engineering notebook and design process now competitive at the judged level.

AREAS REQUIRING IMPROVEMENT

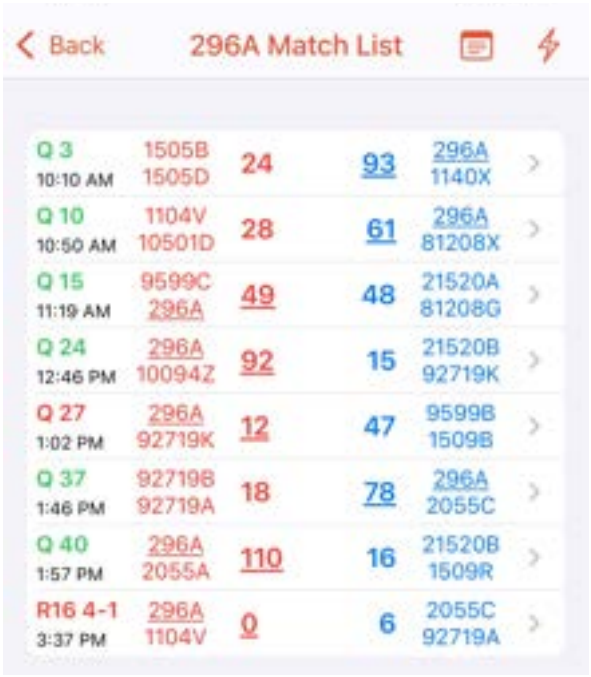
- Mechanical robustness of delicate components (e.g., odometry pods).
- Controller connectivity reliability and pre-match verification procedures.
- Tactics for responding to aggressive defense and box-outs.
- Deeper understanding of playoff-specific rule enforcement and appeals.

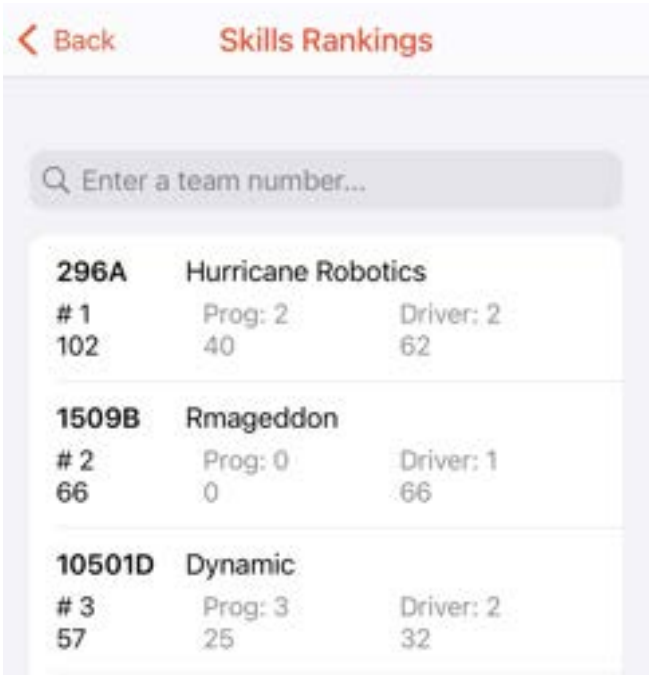
ACTION ITEMS MOVING FORWARD

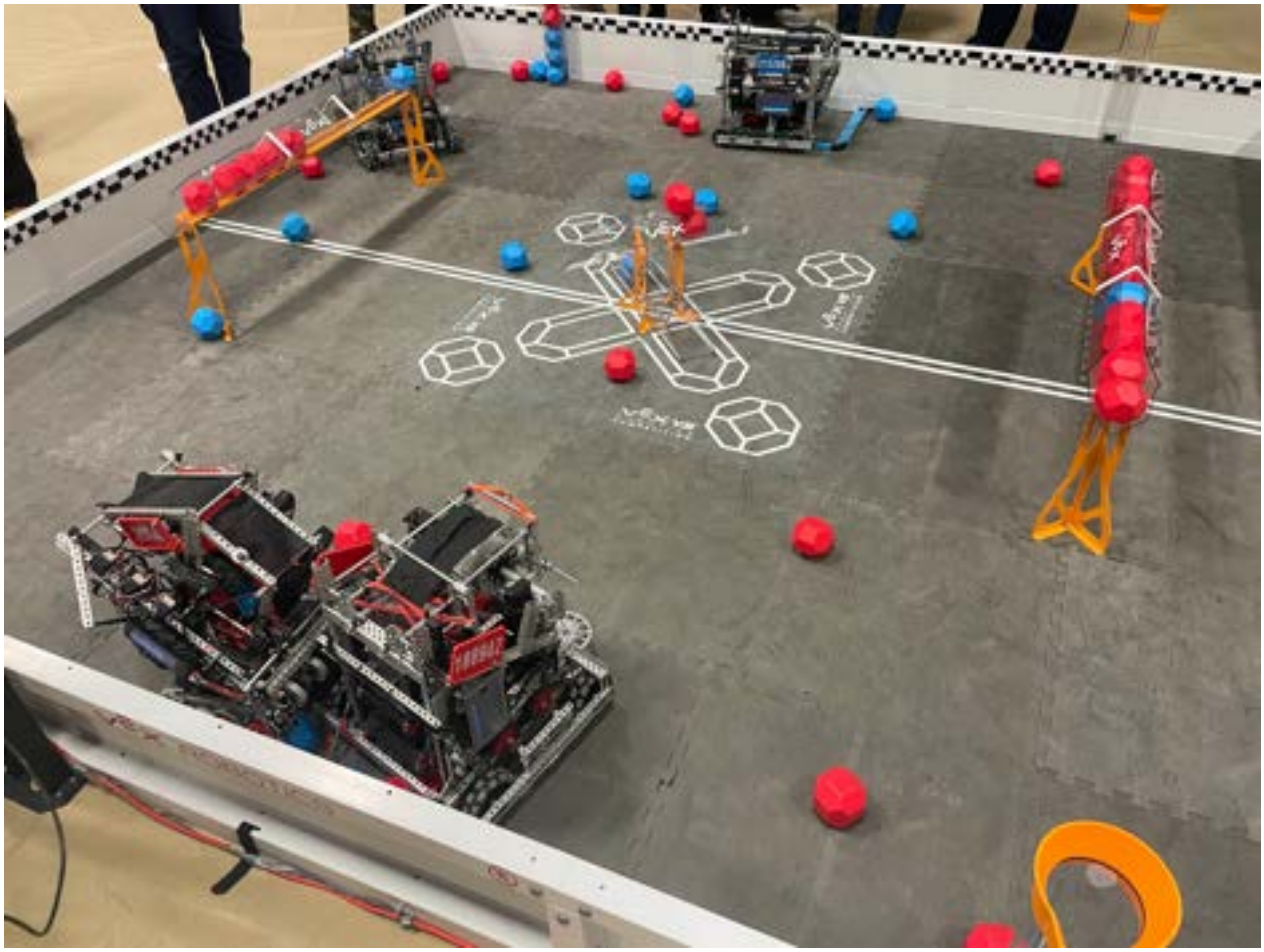
1. Implement redundant inspections before every match, including odometry alignment and controller checks.
2. Add anti-defense strategies to driver practice, including path-escape drills and fake-out cycles.
3. Review elimination-round rule interpretations to avoid preventable disqualifications.
4. Begin a controlled rebuild focused on raising our scoring ceiling for Provincials and Worlds qualification.

Overall, St. Catharines was a **highly successful competition**. Our tournament performance, skills ranking, and judged awards validated the trajectory of our robot and our engineering process. At the same time, the disqualification experience highlighted limitations in both hardware robustness and procedural awareness. Moving forward, we will pause competition and begin a full rebuild with the goal of achieving a **200+ Skills score at Provincials**. The remainder of this notebook will document the design objectives, iterations, and testing process for the next generation of our robot.


296A
INNOVATE AWARD
TEAM 296A
ENGINEERING NOTEBOOK


Back
296A Match List
Q 3 10:10 AM 1505B 1505D 24 93 296A 1140X
Q 10 10:50 AM 1104V 10501D 28 61 296A 81208X
Q 15 11:19 AM 9599C 296A 49 48 21520A 81208G
Q 24 12:46 PM 296A 10094Z 92 15 21520B 92719K
Q 27 1:02 PM 296A 92719K 12 47 9599B 1509B
Q 37 1:46 PM 92719B 92719A 18 78 296A 2055C
Q 40 1:57 PM 296A 2055A 110 16 21520B 1509R
R16 4-1 3:37 PM 296A 1104V 0 6 2055C 92719A


Back
Skills Rankings
Enter a team number...
296A Hurricane Robotics
1 102 Prog: 2 40 Driver: 2 62
1509B Rmageddon
2 66 Prog: 0 0 Driver: 1 66
10501D Dynamic
3 57 Prog: 3 25 Driver: 2 32




INNOVATE AWARD
TEAM 296A


VEX
ROBOTICS
COMPETITION
VEX
ROBOTICS
COMPETITION
ROBOT SKILLS
CHAMPION
2025-2026


VEX
INNOVATE AWARD
TEAM 296A
HURRICANE ROBOTICS


296A
INNOVATE AWARD
TEAM 296A
ENGINEERING NOTEBOOK

ROBOT 2

DRIVETRAIN CHANGELOG

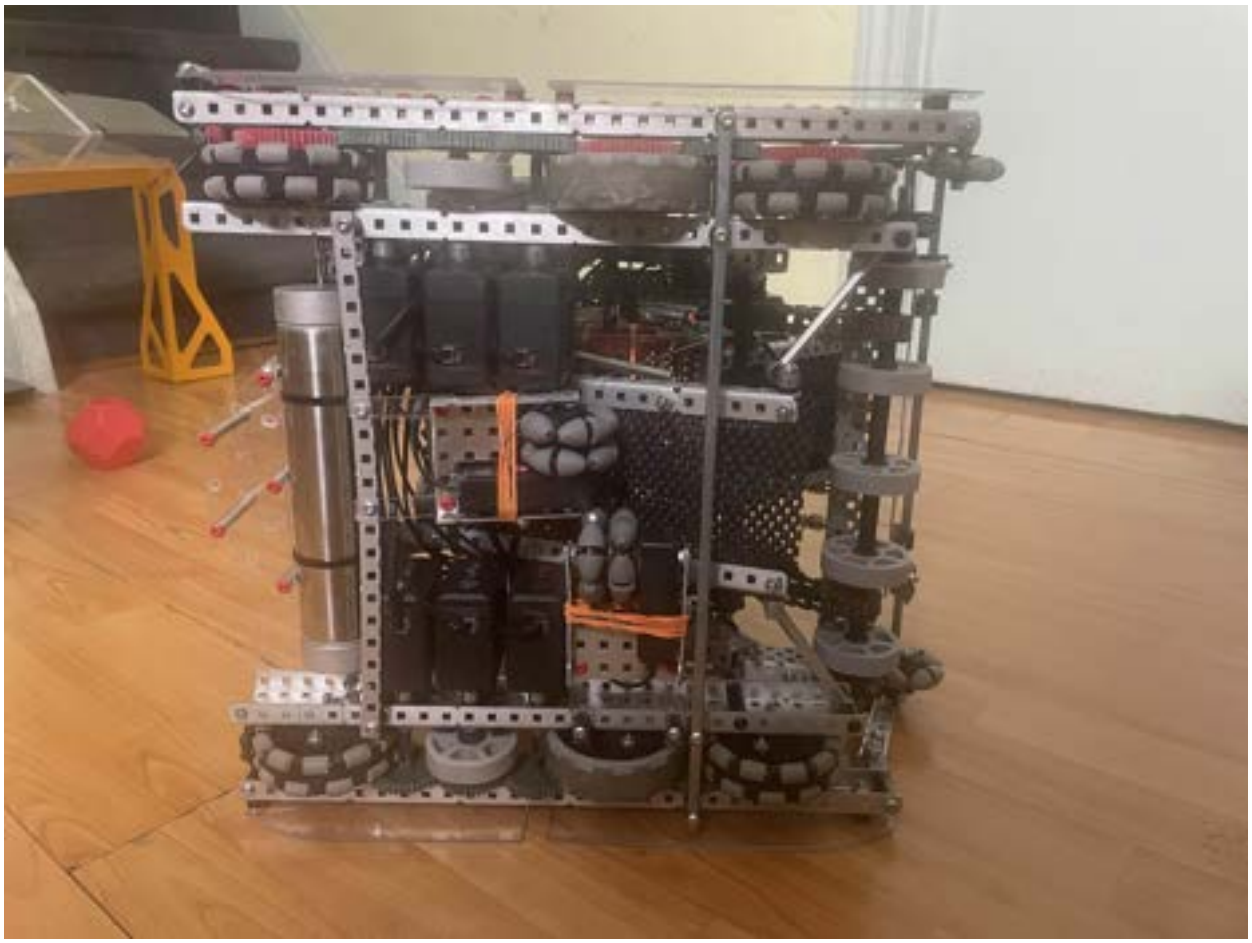
Laksan Mohan | 11/30/2025

BACKGROUND & EVALUATION

After extensive on-field testing, match play, and skills runs with our initial drivetrain configuration (D.1.0), we identified several performance limitations that negatively affected consistency, robustness under defense, and reliability during skills runs. These issues became especially apparent during high-contact match scenarios and precise autonomous/skills navigation near field boundaries.

Following the Engineering Design Process, we documented these failures, redefined the problem, and implemented targeted design changes to address each shortcoming.

CURRENT DRIVETRAIN



ISSUE #1: EXCESSIVE DRIVETRAIN WIDTH

Observed Problem (Test & Refine):
Our original drivetrain measured **30 holes (15”) wide**, which provided excellent stability and general maneuverability. However, during real match conditions, we observed that this width became a liability in **tight field spaces**, particularly the alleyway between the long goal and the field wall.

During both match play and skills runs:

- The robot frequently **made contact with the wall** when driving near long goals.
- Opposing teams were able to **exploit our width**, pinning or redirecting our robot during defensive matches.
- In skills runs, even a **minor alignment error** would cause the robot to collide with the wall, often ruining the entire run due to lost positioning.

Redefined Problem:
The drivetrain must be narrow enough to navigate tight field geometry consistently, while still being large enough to house odometry pods, pneumatics, and internal components without compromising structural integrity.

Implemented Change (Design & Build):
We reduced the drivetrain width to 27 holes (13.5”).

Justification:

- This narrower footprint significantly improves clearance near walls and goals.
- The reduced width makes the robot harder to bully or pin during defense.
- Using an **odd number of holes** provides a center mounting point, simplifying construction, symmetry, and future modifications.
- The internal volume remains sufficient to house odometry pods and other subsystems cleanly.

ISSUE #2: DROP-CENTERED TRACTION WHEEL IMPLEMENTATION

Observed Problem (Test & Refine):
Our previous drivetrain utilized a drop-centered traction wheel concept intended to improve turning and traction balance. While the idea was sound in theory, real-world testing revealed that:

- The performance gains were **negligible**.

- The design added **unnecessary weight**.
- The build complexity increased significantly, making repairs and adjustments more difficult.

Redefined Problem:
The drivetrain must prioritize simplicity, reliability, and serviceability over marginal theoretical performance gains.

Implemented Change (Design & Build):
We removed the drop-centered traction wheel concept entirely in the new drivetrain iteration.

Justification:
- Eliminating this feature reduces weight and mechanical complexity.
- A simpler drivetrain improves reliability across long competition days.
- The performance tradeoff was minimal, making this change clearly beneficial.

ISSUE #3: POOR PARK BARRIER TRAVERSAL

Observed Problem (Test & Refine):
A major flaw identified during skills runs was our robot's **inability to reliably cross the park barrier**. The wheels were mounted too high relative to the chassis, causing:
- Frequent high-centering on the park barrier.
- Occasional complete stalls that would end a skills run prematurely.

Redefined Problem:
The drivetrain must reliably traverse the park barrier without risking getting stuck or requiring perfect approach angles.

Implemented Change (Design & Build):
We lowered the wheels to the bottom row of the drivetrain C-channels.

Justification:
- This lowers the effective ride height and improves ground contact.
- The robot can now consistently clear the park barrier at various angles.
- Reliability during skills runs is significantly improved.

ISSUE #4: MOTOR LAYOUT & CENTER OF GRAVITY

Observed Problem (Test & Refine):
Lowering the wheels eliminated the possibility of **vertically stacked motors**, forcing a reevaluation of our motor layout.

Redefined Problem:
We must maintain a 6-motor drivetrain while optimizing internal space and keeping the center of gravity low.

Implemented Change (Design & Build):
We adopted a **front-1 / rear-2 motor configuration per side**, allowing:

- One motor in the front
- Two motors in the back

This layout creates space in the center of the drivetrain for an **air tank**, with an additional air tank mounted toward the rear.

Justification:

- Pneumatics are positioned low and centrally, improving stability.
- A lower center of gravity enhances pushing power and reduces tipping risk.
- Internal packaging is cleaner and more modular.

ISSUE #5: WHEEL ORIENTATION & MAINTAINABILITY

Observed Problem (Test & Refine):
In the previous design, the wheels were oriented facing inward, which:

- Made maintenance and repairs more difficult.
- Restricted access when implementing drivetrain changes or adding subsystems.

Implemented Change (Design & Build):
We reoriented the wheels to face outward.

Justification:

- Improves accessibility during repairs.
- Makes future drivetrain and subsystem modifications easier.
- Reduces rebuild time during competitions.

ISSUE #6: FLEX WHEEL EFFECTIVENESS

Observed Problem (Test & Refine):
Our drivetrain included a flex wheel intended to assist with park traversal. In practice:

- The flex wheel **did not consistently contact the park barrier**.
- It provided no meaningful advantage during testing or matches.

Implemented Change (Design & Build):
We removed the flex wheel from the drivetrain.

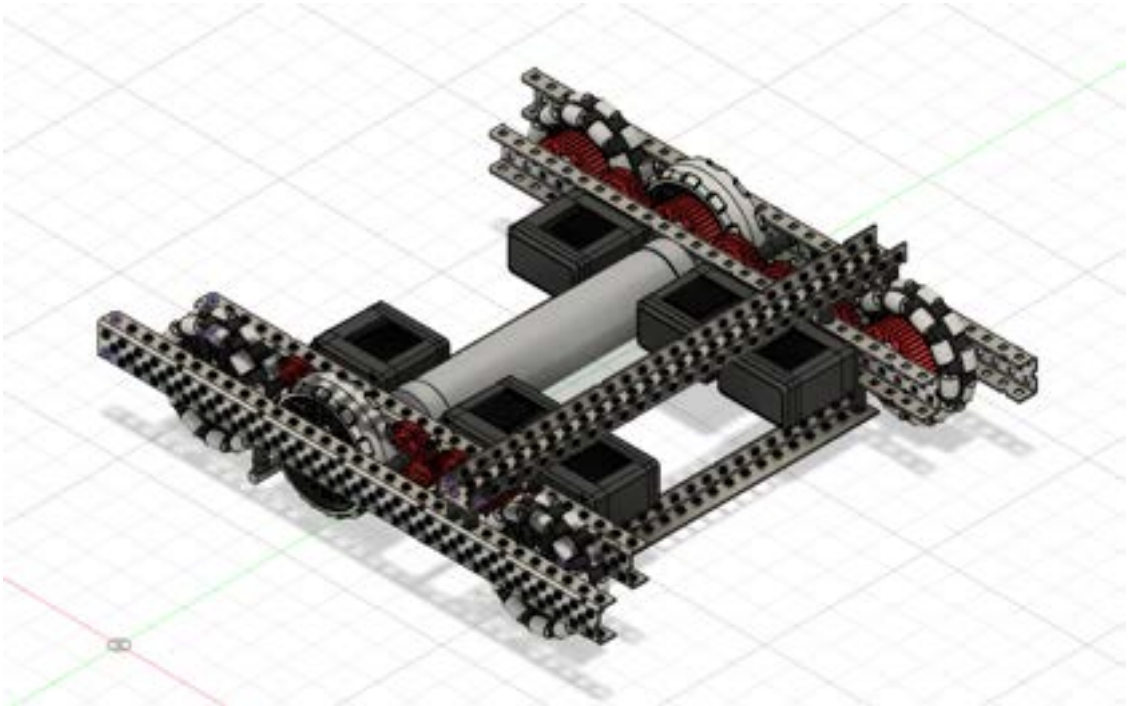
Justification:
- Eliminates unnecessary components and weight.
- Simplifies drivetrain construction.
- No loss in functionality after wheel lowering.

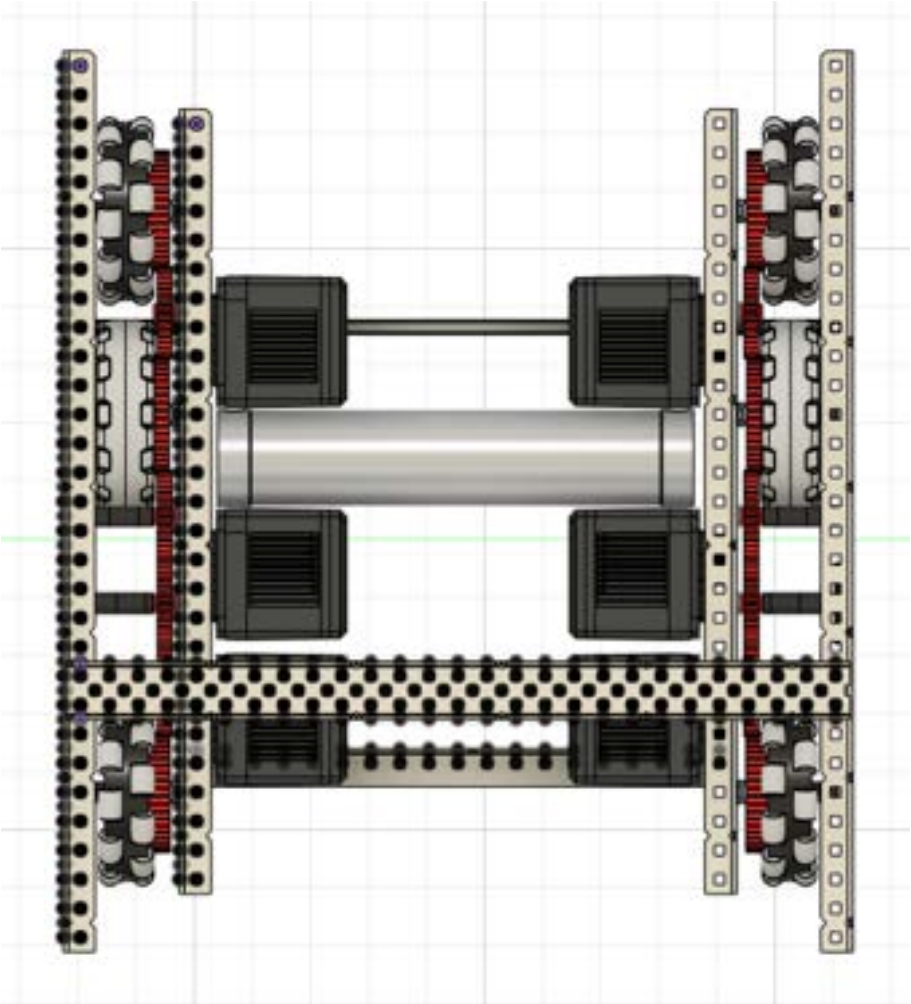
FINAL DRIVETRAIN CONFIGURATION (POST-ITERATION)

Despite these structural changes, we retained several proven drivetrain elements:
- **450 RPM drivetrain** (36:48 gear ratio)
- **4 × 3.25" omni wheels** mounted at the corners for maneuverability
- **2 × 3.25" traction wheels,** positioned directly behind the front omni wheels

Rationale:
Placing the traction wheels toward the front improves resistance to being pushed, especially when contesting or holding position on long goals, while the omni wheels preserve turning smoothness and driver control.

NEW DRIVETRAIN





REFLECTION

This drivetrain iteration demonstrates our commitment to iterative improvement through real testing. Rather than relying solely on theory or CAD assumptions, we allowed match data and skills performance to guide our redesign decisions. Each change directly addresses a documented failure mode, resulting in a drivetrain that is narrower, more reliable, easier to maintain, and better suited to Push Back gameplay.

INTAKE CHANGELOG

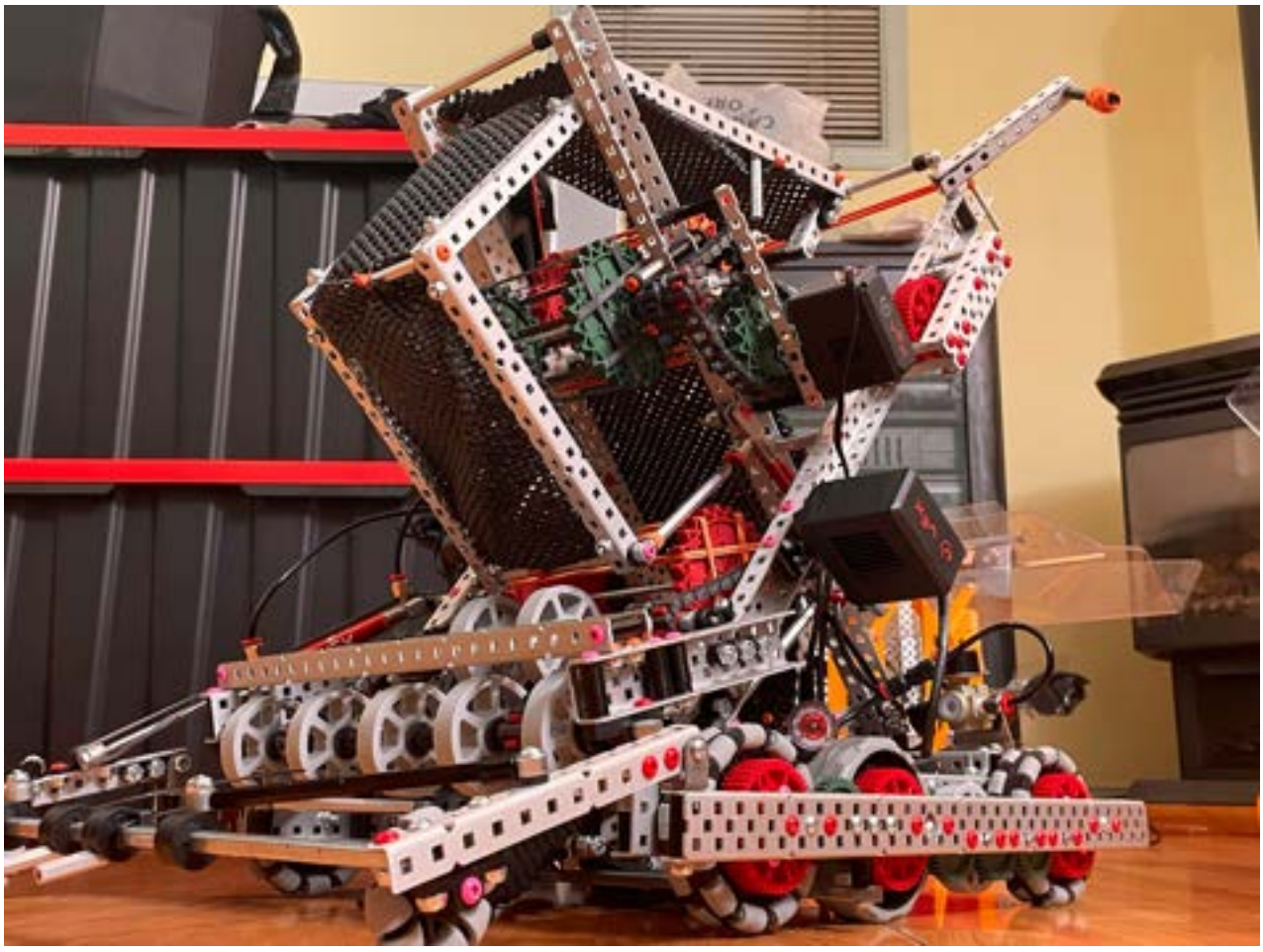
Laksan Mohan | 12/02/2025

BACKGROUND & EVALUATION

Following extensive testing of our original intake system (I.1.0) during match play, driver practice, and skills runs, we determined that the intake was **highly reliable and thermally efficient**. Throughout all testing, the intake **never overheated**, even under continuous operation. However, as the competitive meta of Push Back evolved and matches became increasingly offense-focused, we identified that raw intake speed, rather than reliability, had become the primary limiting factor.

Using the Engineering Design Process, we analyzed performance data, redefined our objectives, and implemented a series of changes aimed at maximizing intake throughput without sacrificing consistency or durability.

CURRENT INTAKE



ISSUE #1: INTAKE THROUGHPUT LIMITATION

Observed Problem (Test & Refine):
While the intake consistently collected and scored blocks, it lacked the speed necessary to dominate offensive interactions. Through timing analysis, we observed that:

- The intake was unable to score **7 blocks in under 1.5 seconds**, which is increasingly required to beat defensive reactions.
- Opposing robots often had sufficient time to contest or disrupt scoring before we could finish a cycle.

Redefined Problem:
To remain competitive in an offense-heavy game, the intake must be capable of **ultra-fast block acquisition and scoring**, allowing us to complete cycles before opponents can effectively respond.

Design Objective:

- Score **7 blocks in under 1.5 seconds**
- Preserve thermal safety and reliability
- Maintain controllability for the driver

ISSUE #2: MATCHLOAD EFFICIENCY & PREROLLER PERFORMANCE

Observed Problem (Test & Refine):
Our original matchload system was fast but not fully optimized. While effective, we identified that:

- The preroller did not aggressively funnel blocks into the intake.
- Minor misalignments during match loading could slow block entry.

Redefined Problem:
The preroller must aggressively guide blocks into the intake regardless of slight driver misalignment and must do so at extreme speed.

Implemented Change (Design & Build):
We redesigned the preroller wheel layout to:

- **Two 2” 40A flex wheels** on the outside
- **Two 1.625” 30A flex wheels** in the center
- A **2.5” gap** between the inner wheels

Justification:
This configuration creates a passive funneling effect. Blocks naturally move toward the path of least resistance, meaning:

- Blocks are drawn toward the softer, more compliant 30A flex wheels.
- The open center gap guides blocks directly into the intake path.

This funneling effect dramatically improves:

- Match loading consistency
- Bottom and middle goal scoring alignment
- Intake forgiveness during high-speed approaches

With this change, we project matchloading all **6 blocks in under 1 second.**

ISSUE #3: ROLLER SPEED UNIFORMITY

Observed Problem (Test & Refine):
In the original intake design, **all rollers spun at 600 RPM**. While simple and reliable, this configuration limited intake acceleration and caused occasional block stall between stages.

Redefined Problem:
The intake requires a **progressive speed gradient** to continuously accelerate blocks as they move inward, preventing stalls and maximizing throughput.

Implemented Change (Design & Build):
We introduced a staged RPM configuration:

- **Preroller (Stage 1):** 1800 RPM
- **Second Roller (Stage 1):** 900 RPM
- **Third Roller (Stage 1):** 600 RPM
- **Second Stage:** 600 RPM
- **Third Stage:** 800 RPM

Justification:

- Higher RPM at the preroller rapidly captures blocks.
- Gradually decreasing RPM prevents jamming while maintaining momentum.
- The speed gradient ensures blocks are always being pulled forward, never pushed backward.

This change dramatically increases intake speed while maintaining smooth block transfer between stages.

ISSUE #4: INTAKE STRUCTURE SIZE & WEIGHT

Observed Problem (Test & Refine):
Our original intake was intentionally designed to be as large as possible, with a capacity of **11 blocks**, under the assumption that higher capacity would improve scoring efficiency. However, testing revealed several drawbacks:

- The extended intake protrusion prevented blocks from being picked up from under the long goal.
- This limited both autonomous routines and match play strategies.
- The larger structure added unnecessary weight and complexity.

Redefined Problem:
The intake must balance capacity with **strategic flexibility**, compactness, and weight efficiency.

Implemented Change (Design & Build):
We simplified and compacted the intake structure, drawing inspiration from a proven design by **Team 38535X**.

Key changes:

- Reduced intake capacity from **11 blocks to 8–9 blocks**
- Shortened the intake footprint
- Removed unnecessary structural elements

Justification:

- Enables blocks to be placed under the long goal again, expanding autonomous and match strategies.
- Reduces weight, improving acceleration and handling.
- Makes the intake more compact and easier to integrate with other subsystems.

FINAL INTAKE CONFIGURATION (POST-ITERATION)

After iteration, the intake now features:

- A high-speed, staged RPM roller system
- An aggressively funneling preroller layout
- Reduced structural weight and footprint
- Optimized capacity aligned with game strategy

Despite these aggressive performance upgrades, the intake maintains its original strengths of thermal reliability and consistency.

REFLECTION

This intake redesign exemplifies our team's philosophy of i**terative improvement driven by real match data**. Rather than optimizing for theoretical capacity, we optimized for **speed, strategic flexibility, and offensive dominance**. By redefining our objectives around real gameplay demands, we created an intake capable of executing ultra-fast scoring cycles while remaining reliable and driver-friendly.

ALIGNER CHANGELOG

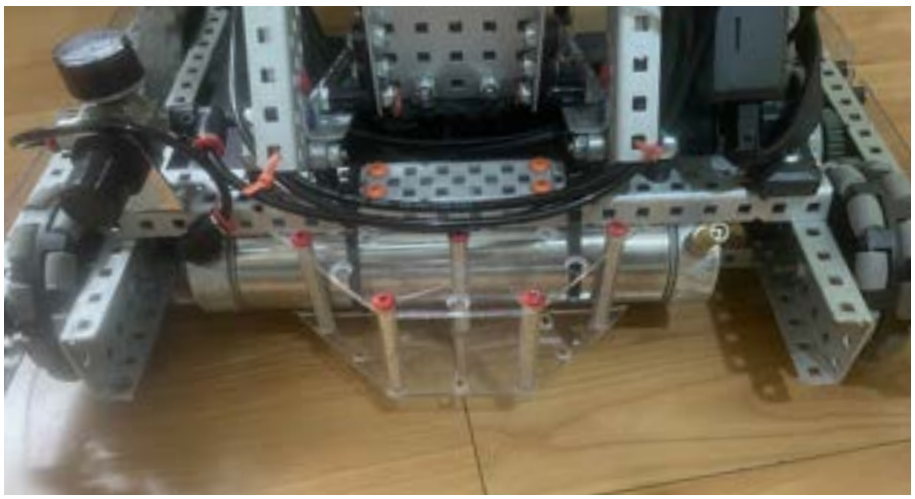
Laksan Mohan | 12/05/2025

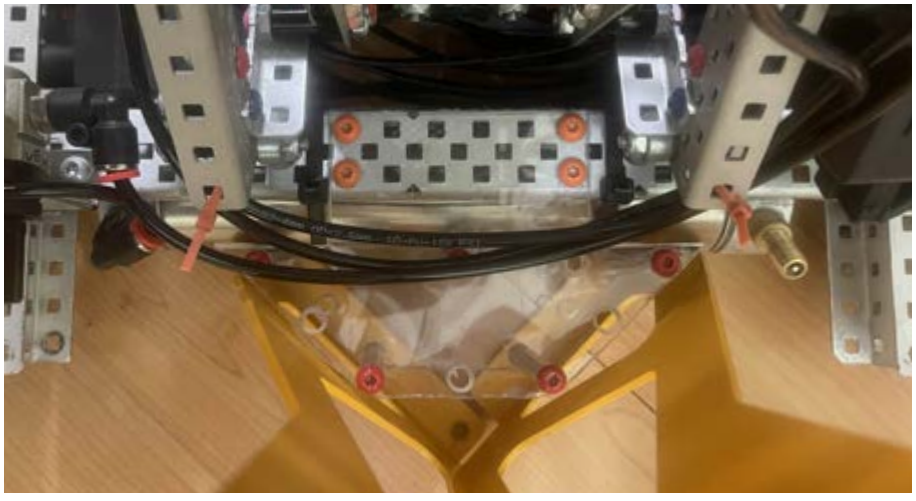
BACKGROUND & EVALUATION

Our long goal aligner (A.1.0) performed reliably under ideal conditions and successfully aligned the robot when approaching the long goal head-on. During early testing, the mechanism demonstrated strong consistency when the robot was already well positioned and parallel to the goal.

However, as we transitioned into **skills runs and real match scenarios**, we discovered a critical design flaw that significantly limited the aligner's effectiveness under non-ideal approaches. Following our Engineering Design Process, we analyzed this failure, redefined the alignment problem, and implemented a redesigned solution optimized for real-world gameplay conditions.

CURRENT ALIGNER





ISSUE #1: TRAPEZOIDAL ALIGNER GEOMETRY

Observed Problem (Test & Refine):
Our original aligner used a **trapezoidal geometry**, rather than the more traditional triangular or curved aligner shape. This decision was made during the design phase because the trapezoidal profile fit cleanly within the long goal opening and appeared effective in CAD.

However, field testing revealed several major issues:

- The **large flat front edge** required extremely precise alignment.
- If the robot was misaligned by more than **~2 inches**, the aligner would fail to engage.
- Approaches at angles greater than **~40 degrees** resulted in the robot catching or bouncing off the goal instead of self-correcting.
- In skills runs, this failure often ruined the entire run, as small approach errors could not be recovered.

Redefined Problem:
The aligner must be able to **self-correct alignment errors**, regardless of approach angle or lateral offset, and must function reliably in high-speed, imperfect driving conditions.

ISSUE #2: SHARP VERTICES AS FAILURE POINTS

Observed Problem (Test & Refine):
The trapezoidal design introduced **distinct corners and vertices**. During testing, these vertices frequently became:

- Points where the robot could **catch or bind** on the goal.
- Locations that prevented smooth sliding into alignment.
- Failure points during angled or off-center approaches.

Redefined Problem:
The aligner must eliminate sharp vertices, as corners act as mechanical traps rather than alignment aids.

IMPLEMENTED CHANGE #1: ELLIPTICAL ALIGNER GEOMETRY

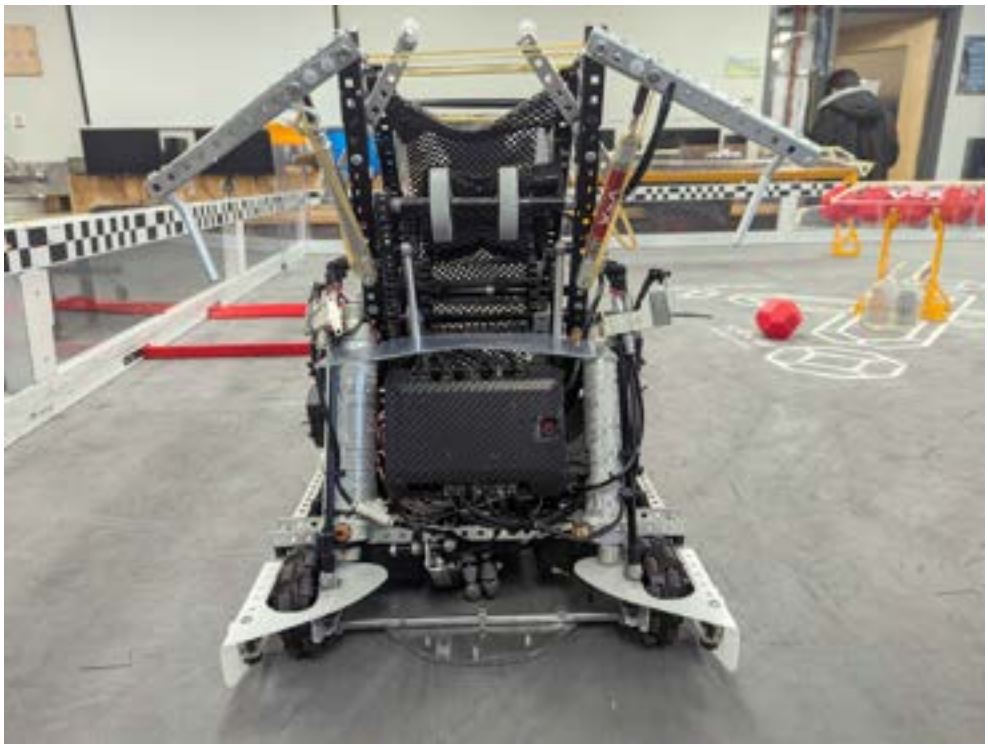
Design & Build Solution:
We redesigned the aligner to feature a **fully elliptical, curved profile with no corners**.

Justification:
- A continuous curve allows the robot to **slide naturally into alignment**.
- Removing vertices eliminates snag points and binding.
- Curved geometry redirects lateral forces into corrective motion rather than resistance.

This change dramatically increases the aligner's ability to correct for misalignment, even during aggressive or high-speed approaches.

IMPLEMENTED CHANGE #2: COMBINED SLED / ALIGNER DESIGN

Design & Build Solution:
We implemented **combination sled-aligner** components, inspired by a proven design used by **Team 502G**.



Justification:
- The sled component allows the robot to glide over the park barrier smoothly.
- Integration of sleds and aligners reduces plastic count and complexity.
- The curved sled shape complements the elliptical aligner, further reducing the chance of catching.

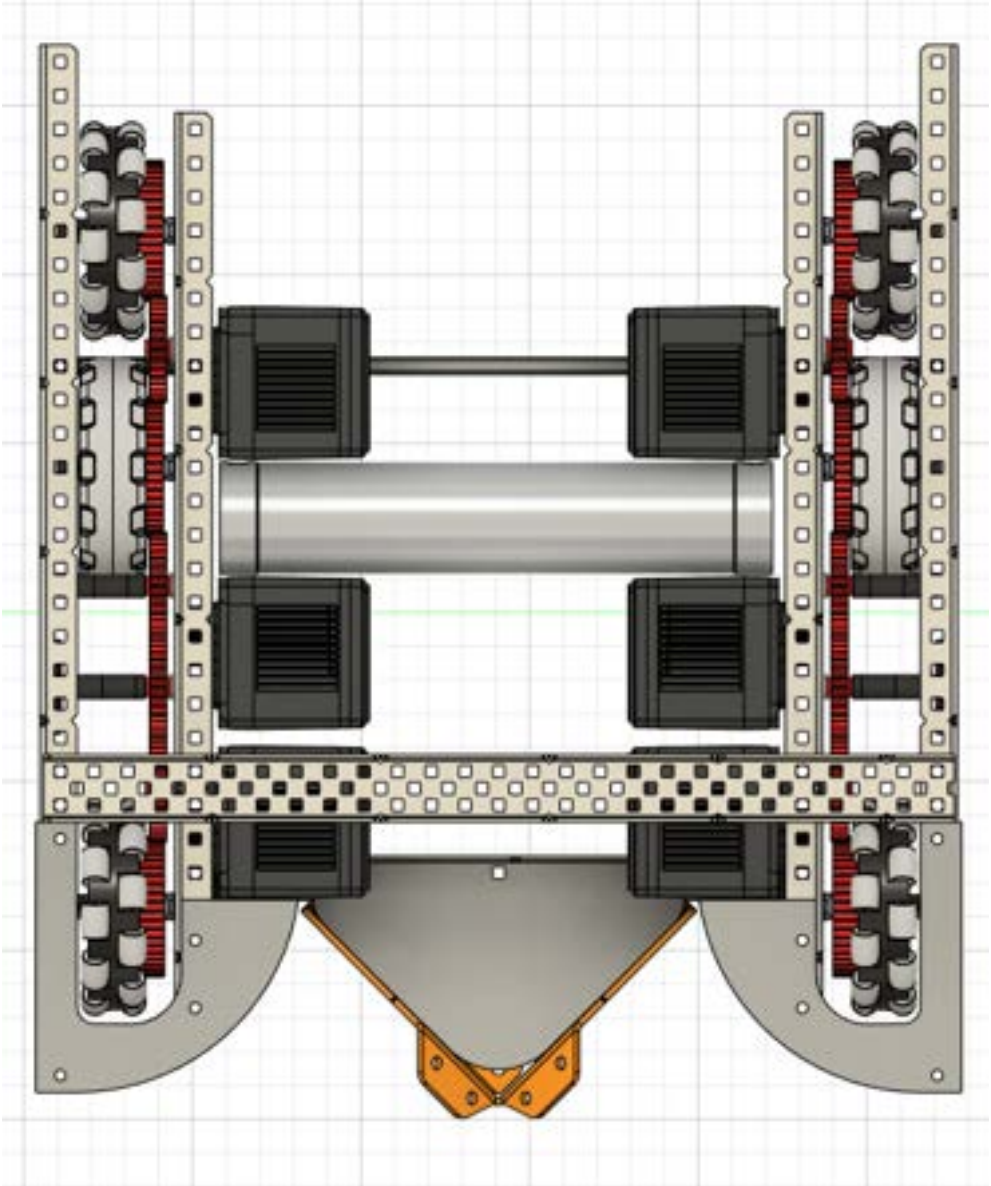
This hybrid design improves both **alignment reliability** and **drivetrain survivability** during repeated high-speed impacts with the long goal.

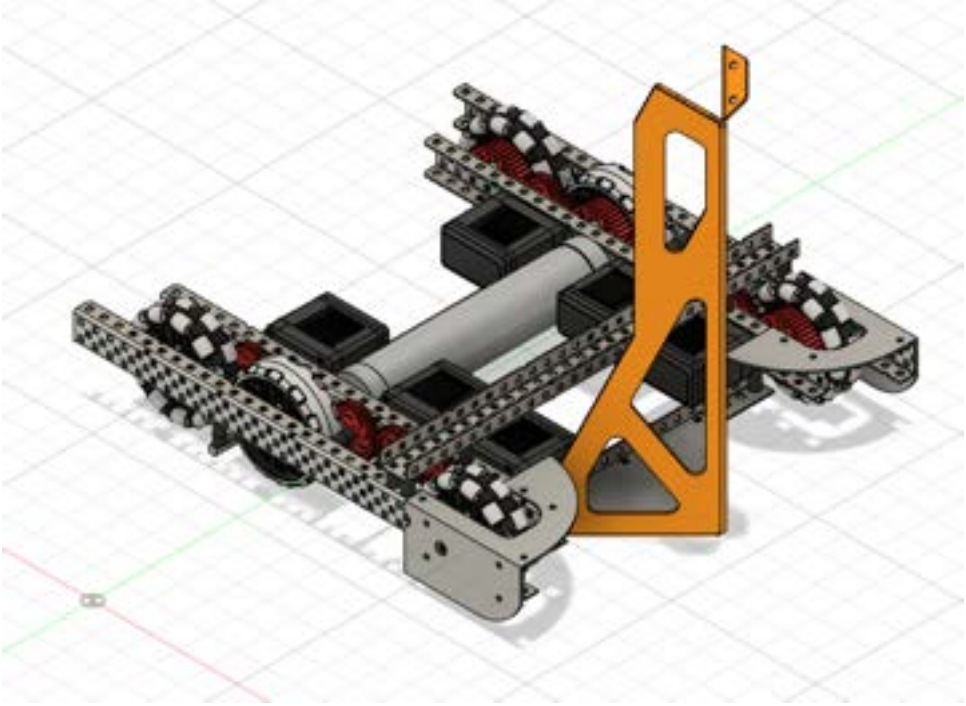
FINAL ALIGNER PERFORMANCE (POST-ITERATION)

With the elliptical geometry and sled integration:

- The robot can align from **any approach angle**, including extreme angles and near-perpendicular approaches.
- The aligner effectively supports **360-degree alignment**, allowing the robot to self-correct regardless of orientation.
- Skills consistency is significantly improved, as minor driving errors no longer result in catastrophic failures.

NEW ALIGNERS





REFLECTION

This aligner redesign highlights the importance of designing for **real match conditions rather than idealized CAD scenarios**. While the trapezoidal aligner appeared effective in theory, field testing revealed its limitations under imperfect alignment. By eliminating corners, adopting a fully curved geometry, and integrating sled functionality, we transformed a fragile mechanism into a **robust, angle-agnostic alignment system** that significantly improves both match and skills performance.

MATCHLOADER CHANGELOG

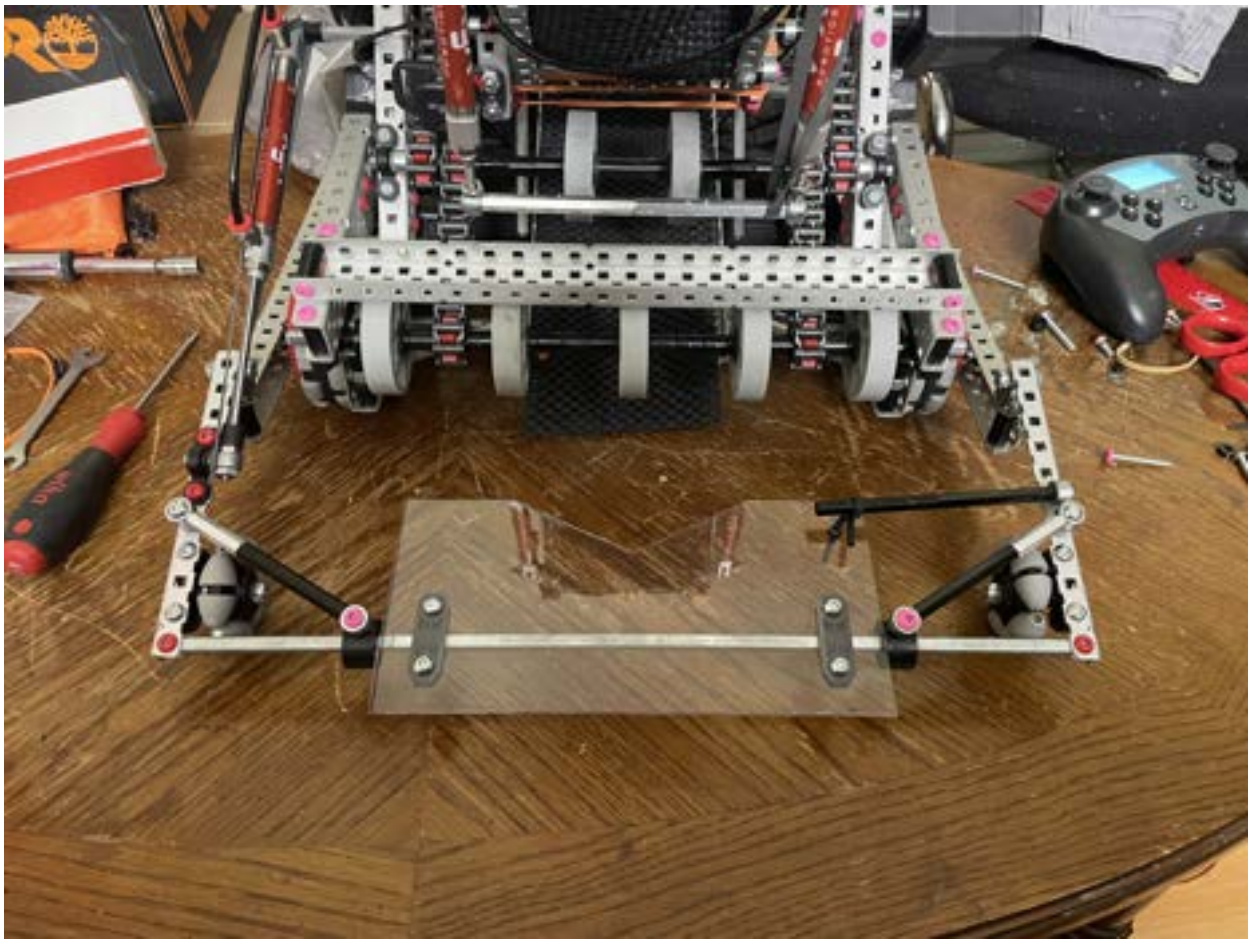
Laksan Mohan | 12/07/2025

BACKGROUND & EVALUATION

Our original matchloader interface (M.1.0) performed reliably under ideal conditions and allowed us to remove blocks from the match loader at competitive speeds. In controlled testing, the system consistently transferred blocks into the robot without mechanical failures.

However, as we increased cycle speed and began executing full skills routes and high-speed match play, we identified repeatable inconsistencies caused by approach angle variation. Following the Engineering Design Process, we analyzed these failures and implemented a redesign focused on repeatability and alignment.

CURRENT MATCHLOADER



ISSUE #1: ANGLE-SENSITIVE MATCHLOADING

Observed Problem (Test & Refine):
During testing, we observed that if the robot contacted the match loader at even a slight angle:

- Blocks would **enter the robot inconsistently** instead of dropping straight in.
- Blocks could **jam or stall** at the intake entrance, thus increasing the time at the matchload station

Redefined Problem:
To achieve maximum intake speed, the matchloading interaction must be geometrically repeatable, ensuring blocks enter the robot in the same position and orientation every time.

ISSUE #2: LACK OF A DEDICATED MATCHLOAD ALIGNMENT FEATURE

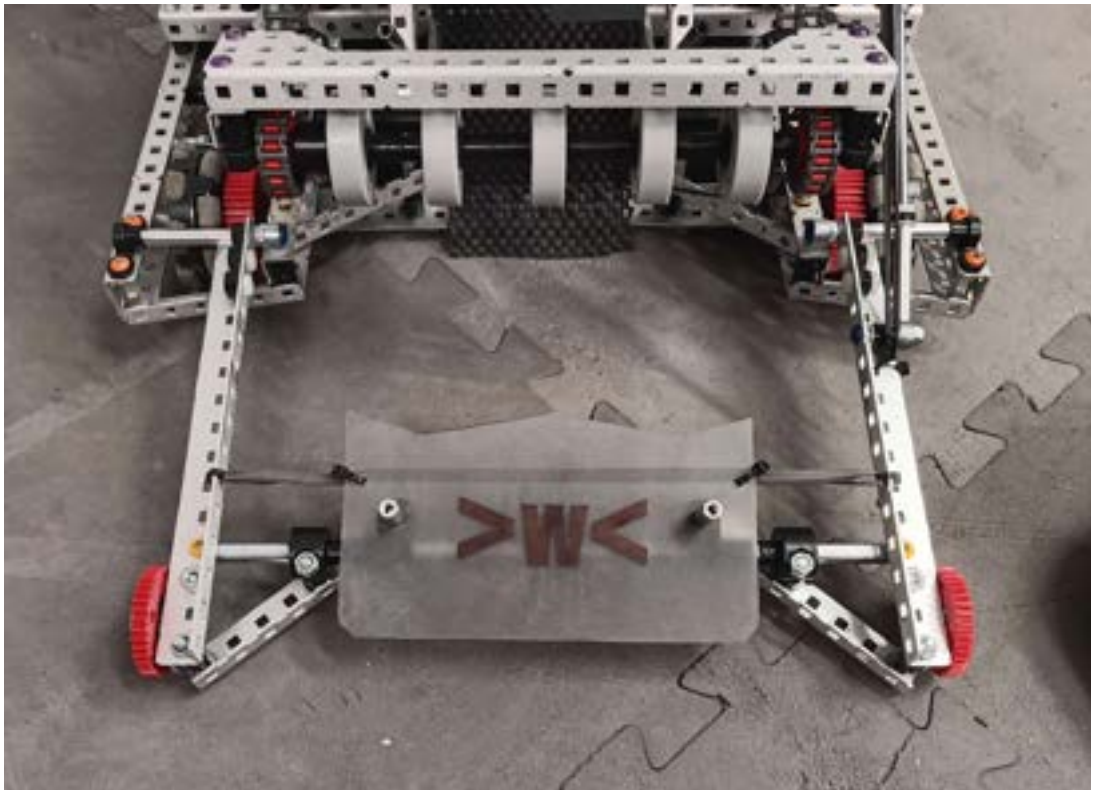
Observed Problem (Test & Refine):
Our original design relied solely on driver precision to align with the match loader. While workable, this approach:

- Introduced variability between runs.
- Caused inconsistency during skills when small errors compounded over time.
- Limited how aggressively we could tune intake and preroller speeds.

Redefined Problem:
The robot must mechanically enforce consistent alignment with the match loader rather than relying on driver or autonomous precision alone.

IMPLEMENTED CHANGE #1: MATCHLOAD ALIGNER INTEGRATION

Design & Build Solution:
We implemented a **dedicated matchload aligner**, inspired by a proven design used by **Team 3332A**.



Justification:

- The aligner forces the robot to contact the match loader at a **consistent position and angle**.
- Blocks now drop **vertically and predictably** into the intake path.
- The repeatable interaction allows us to **tune the intake and preroller to higher speeds** without risking jams.

This change significantly improves both speed and reliability during matchloading.

ISSUE #3: SKILLS ROUTE DISRUPTION FROM IMPACT JERK

Observed Problem (Test & Refine):
During skills runs, we noticed that impacting the match loader often introduced **a jerking motion**, causing:

- The robot to leave the match loader at a slight angle.
- Small heading errors that compounded later in the route.
- Occasional failure to correctly back into the long goal, ruining the entire skills run.

Redefined Problem:
The matchloading interaction must not disturb the robot's heading or introduce angular error that propagates into later autonomous movements

IMPLEMENTED CHANGE #2: CONTROLLED CONTACT & EXIT GEOMETRY

Design & Build Solution:
The matchload aligner also functions to:

- Absorb and redirect impact forces.
- Guide the robot into a predictable exit orientation after loading.

Justification:

- The robot will now leave the match loader square and centered.
- Autonomous and driver skills routines will become significantly more repeatable.
- Heading consistency improves long-goal alignment reliability later in the run.

FINAL MATCHLOADER PERFORMANCE (POST-ITERATION)

After implementing the matchload aligner:

- Matchloading is **more consistent regardless of approach angle.**
- Intake and preroller systems can be tuned more aggressively for speed.
- Skills runs are significantly more reliable, with fewer cascading alignment failures.
- Overall cycle time is reduced while consistency is increased.

REFLECTION

This redesign reinforces the principle that **speed is only valuable when it is repeatable**. By mechanically enforcing alignment with the match loader, we eliminated variability that previously limited intake tuning and skills reliability. Drawing inspiration from Team 3332A and adapting it to our robot allowed us to transform a good matchloading system into a high-speed, competition-consistent solution.

MINOR CHANGES

Laksan Mohan | 12/10/2025

BACKGROUND & EVALUATION

In addition to major subsystem redesigns, we identified several smaller but critical improvements through extended match play, skills runs, and late-match testing. While none of these changes required a full subsystem overhaul, each addressed a specific reliability, usability, or strategic limitation observed during competition simulation.

Following the Engineering Design Process, we documented these issues, evaluated their competitive impact, and implemented targeted refinements to improve overall robot performance and consistency.

ISSUE #1: DOUBLE PARK PNEUMATIC RELIABILITY

Observed Problem (Test & Refine):
Our robot successfully demonstrated a **working double park mechanism**, but testing revealed that it was **not competitively reliable**. Due to the robot's mass (**~17 lbs**), the double park required:

- **~50 PSI or higher** to actuate reliably.
- Near-full pneumatic pressure late into a match, which was rarely achievable after extended scoring and wing use.

As a result, double park success was inconsistent, especially near the end of long matches.

Redefined Problem:
The double park system must remain reliable even at r**educed pneumatic pressure** late in the match.

Implemented Change (Design & Build):

- Added **dual pneumatic air tanks** to increase total stored air volume.
- Began an active **weight reduction effort** across the robot to lower the force required for double parking.

Justification:

- Increased air volume reduces pressure drop during a match.

- Lower robot mass reduces required actuation force, improving double park reliability at lower PSI.

ISSUE #2: DOUBLE WING COMPLEXITY & USABILITY

Observed Problem (Test & Refine):
Our previous **double wing design** offered strong theoretical advantages in match play. However, driver testing showed that:

- Shorter wings were significantly harder to deploy and control.
- The added complexity reduced consistency under time pressure.
- Driver reaction time suffered, particularly in fast defensive situations.

Redefined Problem:
The wing system must prioritize **speed, simplicity, and driver consistency** over theoretical flexibility.

Implemented Change (Design & Build):
We transitioned from a double wing system to a single, longer wing.

Justification:

- A longer wing is easier to deploy accurately.
- Reduces driver cognitive load.
- Improves consistency in both match play and skills.

ISSUE #3: WING HEIGHT CONSISTENCY FOR DESCORING

Observed Problem (Test & Refine):
We observed that maintaining the correct wing height for descoring required continuous driver input, which was inconsistent under match pressure.

Redefined Problem:
The wing must mechanically hold its optimal height for descoring without constant adjustment.

Implemented Change (Design & Build):
We added a **latching mechanism** to the wing system.

Justification:

- Ensures the wing remains at the correct height.

- Improves descoring reliability.
- Reduces driver workload during critical match moments.

ISSUE #4: WING DEPLOYMENT SPEED & ALIGNMENT

Observed Problem (Test & Refine):
In match play, effective wing usage must occur **in under one second**. We observed that:

- Precise wing positioning under pressure was difficult.
- Slight misalignment reduced effectiveness or caused missed interactions.

Redefined Problem:
Wing deployment must be fast, intuitive, and mechanically guided to ensure consistent positioning.

Implemented Change (Design & Build):
We added a **wing alignment guide**, allowing the driver to deploy the wing quickly without needing precise manual alignment.

Justification:

- Enables sub-second wing deployment.
- Improves consistency across different drivers and match situations.
- Reduces mechanical misalignment and wasted motion.

ISSUE #5: MIDDLE GOAL ALIGNMENT IN SKILLS

Observed Problem (Test & Refine):
During skills runs, we identified that aligning with **the upper middle goal** was slower and less consistent than desired.

Redefined Problem:
The robot must align with the upper middle goal quickly and repeatably to optimize skills scoring.

Implemented Change (Design & Build):
We added a **dedicated middle goal aligner**.

Justification:

- Improves consistency during skills.

- Reduces reliance on perfect driver or autonomous positioning.
- Allows more aggressive skills routing.

ISSUE #6: STRUCTURAL PLASTIC MATERIAL SELECTION

Observed Problem (Test & Refine):
We originally used **polycarbonate** for all plastic components. However, testing revealed that:

- Polycarbonate lacked sufficient rigidity in certain applications.
- It is not laser-cuttable, limiting manufacturing precision and iteration speed.

Redefined Problem:
We require a plastic material that is rigid, precise, and easy to manufacture consistently.

Implemented Change (Design & Build):
We switched from polycarbonate to **Delrin** for applicable components.

Justification:

- Delrin is laser-cuttable, enabling faster iteration.
- Increased rigidity improves mechanical consistency.
- Cleaner manufacturing improves repeatability.

ISSUE #7: COMMUNICATION RELIABILITY & REDUNDANCY

Observed Problem (Test & Refine):
A single radio presents a single point of failure. If it disconnects during a match, the robot becomes completely disabled.

Redefined Problem:
The robot must remain operational even in the event of a radio failure.

Implemented Change (Design & Build):
We installed **two radios** on the robot.

Justification:

- Provides redundancy in case of radio disconnection.
- Improves match reliability.
- Reduces the risk of being fully disabled during competition.

REFLECTION

While classified as “minor,” these changes collectively have a **major impact on competitive reliability**. By improving pneumatic consistency, simplifying driver interactions, enhancing alignment, upgrading materials, and adding communication redundancy, we significantly increased the robot’s robustness under real competition conditions. These refinements demonstrate our commitment to addressing small failure points before they become match-losing issues.

ROBOT 2 IMPLEMENTATION PLAN

Laksan Mohan | 12/20/2025

Initially, we planned to have a completed robot by the end of winter break. However, we did not anticipate potential shipping delays. In particular, many of the holiday days fell on weekdays, which significantly slowed order processing and delivery timelines. As a result, we now anticipate receiving our remaining parts near the end of winter break, on **January 2nd, 2026**.

Due to these delays, our original implementation plan has been **revised and restructured**. To guide the construction of the redesign and ensure it is completed on schedule, we have included a detailed timeline below:

PROGRAMMING

LIBRARY CHOICE

Glae Alejo | 05/14/2025

Choosing the right programming library is one of, if not the most critical decision in VEX Robotics development. The library decides what programming language we will use to code the robot, along with the software tools, language features and what paradigm we must think in to code in the language properly. It also affects development speed, determining how quickly we can implement and test new autonomous routines and macros. A poor choice here could cost us matches due to crashes, while the right choice provides a solid foundation for success throughout the new season.

OPTIONS EVALUATED

Choices	Language	Pros	Cons
Vexide	**Rust**	• Excellent compile-time safety catches errors before runtime • Built-in async executor • Built-in SIMD API for performance-critical math • Memory safety without garbage collection • Rich standard library • Fast API updates (able to reverse-engineer new VEX software)	• Steeper learning curve for Rust • Smaller VEX community compared to PROS • Longer compile times especially for clean release builds

OPTIONS EVALUATED

Choices	Language	Pros	Cons
PROS	**C/C++**	• Mature and widely used in VEX community • Extensive documentation • Large community support • Fast compile times • We used PROS last year, so we have more experience	• Risk of runtime errors (segfaults, buffer overflows) • No built-in async/await • More boilerplate code • Slow to update API, must wait for VEX SDK updates
VEXcode V5 C++	**C++**	• Official VEX solution • Large community support • Fast API updates	• Same C++-specific disadvantages as PROS above • Limited API documentation

FINAL CHOICE: VEXIDE

We chose Vexide primarily for its compile-time safety guarantees. Rust's type system and borrow checker catch entire classes of bugs at compile time rather than during a match. This dramatically reduces runtime panics and speeds up our iteration cycle since we catch errors before deploying to the robot. The built-in async executor is equally critical, allowing us to write concurrent code for sensors, motors, and autonomous routines in a clean, composable way. Performance was another key factor: Rust's zero-cost abstractions and SIMD support give us C-level speed with high-level ergonomics (think Python). The modern tooling around Cargo provides excellent dependency management, allowing us to use other Rust libraries in our code quickly. While the learning curve is steeper and compile times are longer, the reduction in debugging time and the prevention of match-losing runtime crashes make this tradeoff worthwhile.

CODE EDITOR CHOICE

Glae Alejo | 05/15/2025

The code editor is our primary interface for development work. The right editor choice significantly impacts how quickly we can write, navigate and refactor code, how easily new members can start contributing code, and how smoothly the software runs on team hardware. Since we spend hours in the editor during development and competition prep, choosing one that maximizes efficiency while minimizing friction is essential.

OPTIONS EVALUATED

Choices	Pros	Cons
Zed	• Extremely fast performance (built in Rust) • Native, lightweight application • Clean, minimal UI reduces distractions • Fast startup time (<1 second) • Built-in LSP support (rust-analyzer) • Low memory footprint • Modern design philosophy	• Relatively new (less mature) • Smaller extension ecosystem • Fewer advanced features than VS Code • Limited plugin development • No legacy configuration support

OPTIONS EVALUATED

Choices	Pros	Cons
VS Code	• Massive extension marketplace • Excellent Rust support (rust-analyzer) • Integrated debugging • GitLens and other productivity tools • Industry standard • Remote development support	• Electron-based (higher resource usage) • Slower startup time • Can become sluggish with many extensions • Higher memory consumption (~300-500MB) • UI lag
RustRover/ CLion	• Professional-grade IDE from JetBrains • Advanced refactoring tools • Excellent code analysis • Integrated everything (database, version control) • Smart code completion • Purpose-built for Rust/C	• Very heavy resource usage (>2GB RAM) • Slow startup (10+ seconds) • Probably overkill for VEX • Complex UI

FINAL CHOICE: ZED

We selected Zed for its performance characteristics and simplicity. As a native application built in Rust, it starts in under a second and maintains responsive editing even during intensive compile cycles, which is crucial when we are iterating rapidly during meetings. The clean, minimal UI reduces cognitive overhead and lets the team focus on code rather than managing IDE complexity. Zed's out-of-the-box rust-analyzer integration works flawlessly with Vexide. Most importantly, it strikes the right balance between accessibility and power—simple enough that new team members can be productive immediately, but capable enough for advanced development work. VS Code's sluggishness and memory bloat would slow us down during competitions, while RustRover/CLion's heavyweight IDE approach is overkill for our needs.

VERSION CONTROL SYSTEM CHOICE

Glae Alejo | 05/16/2025

Version control is essential for managing code across a team and throughout a competition season. The right VCS enables seamless collaboration, comprehensive history automatically keeping track of changes and allowing us to backtrack if needed, and insurance in case something goes terribly wrong with the code or the laptop we have the code on. Without proper version control, we risk losing work and having multiple conflicting versions of code.

OPTIONS EVALUATED

Choices	Architecture	Pros	Cons
Git	**Distributed**	• Industry standard • Massive community and resources • Excellent branching and merging • GitHub integration (it's in the name) • Fast local operations • Powerful history rewriting	• Complex command set • Steep learning curve initially • Easy to make mistakes (force push, etc.)

OPTIONS EVALUATED

Choices	Architecture	Pros	Cons
Apache Subversion (SVN)	**Centralized**	• Simple mental model (linear history) • Better binary file handling • Atomic commits • Easier to learn • Fine-grained access control • Good for large binary assets	• Requires server connection for most operations • Slower branching/merging • Declining community support
Jujutsu (jj)	**Distributed**	• Modern VCS design (learned from Git's mistakes) • Every operation is recorded • Easy to undo anything • Better merge conflict resolution • Simpler mental model • Git interoperability	• Very new (experimental) • Tiny community

FINAL CHOICE: GIT

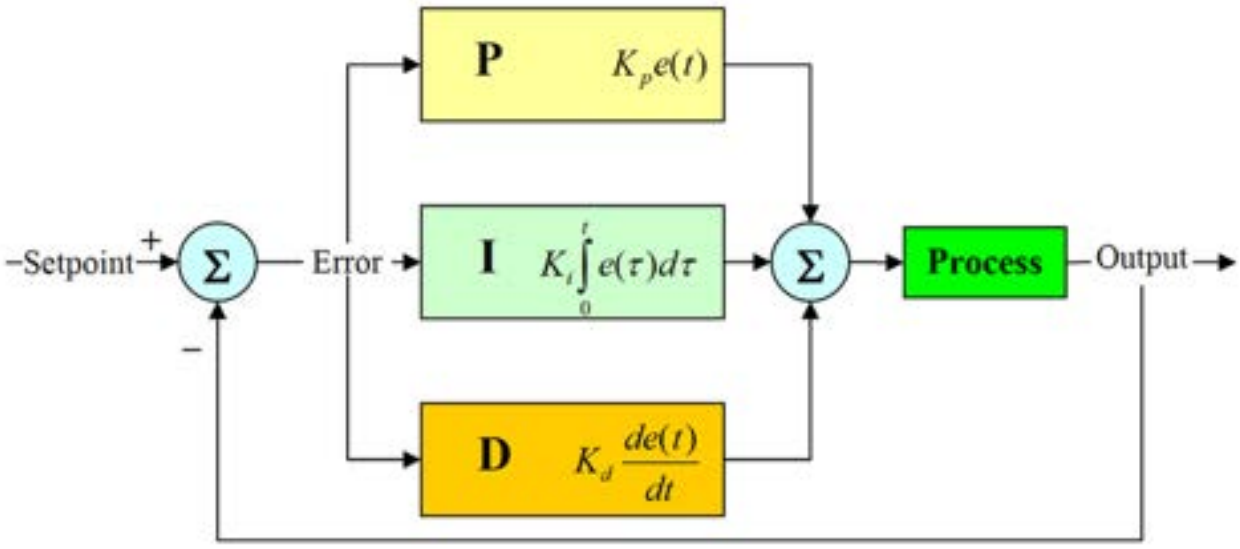
Git was the clear choice despite its complexity because of its ubiquity and distributed architecture. Every team member having a full repository clone means we have automatic backups and can work offline during competitions or travel when internet access is unreliable. GitHub's ecosystem gives us free hosting and GitHub Actions to automatically test code. While Git's learning curve is steep and the command set is confusing initially, the abundance of tutorials, Stack Overflow answers, and existing team familiarity make this investment worthwhile.

FEEDBACK CONTROL

Glae Alejo | 5/17/2025

Feedback control is a method where the controller continuously measures the actual output of a system and compares it to the desired setpoint. The difference between these values (the error) is used to calculate the action needed to correct the system output. This creates a closed-loop system that can react to disturbances and correct for deviations from the target. Some real-world applications include temperature regulation (fridge, oven, thermostat) and cruise control in cars to maintain a certain speed. In robotics we use feedback control to manage things like the robot's position, heading and velocity.

The most common feedback controller, and the one we will use is the Proportional, Integral, Derivative (PID) controller. In order to minimize the amount of fancy math written, I will explain through a Rust implementation of the PID controller which we use for linear drive movements.



CODE

```
pub struct Pid {
 pub kp: f64, // Proportional gain
 pub ki: f64, // Integral gain
 pub kd: f64, // Derivative gain
 pub integral_limit: f64, // Anti-windup limit
 pub min_error: f64, // Deadband threshold
 pub integral_decay: f64, // Integral decay factor
 prev_error: Option<f64>, // Previous error for derivative
 integral: f64, // Accumulated integral
 disabled: bool,
}
impl Pid {
    pub fn next(&mut self, error: f64, dt: f64) -> f64 {
        if self.disabled || error.abs() < self.min_error {
            self.reset();
            return 0.0;
        }

        if dt == 0.0 {
            self.prev_error = Some(error);
            return self.kp * error + self.ki * self.integral;
        }

        // Integral with decay (prevents windup)
        self.integral *= self.integral_decay;
        self.integral += error * dt;
        if self.integral_limit > 0.0 {
            self.integral = self
                .integral
                .clamp(-self.integral_limit, self.integral_limit);
        }

        // Derivative calculation
        let derivative = match self.prev_error {
            Some(prev) => (error - prev) / dt,
            None => 0.0,
        };

        self.prev_error = Some(error);

        // PID output: u(t) = Kp*e(t) + Ki*∫e(t)dt + Kd*de(t)/dt
        // Sum of P, I, D terms
        (self.kp * error) + (self.ki * self.integral) + (self.kd *
derivative)
    }
}
```

COMPONENT BREAKDOWN

Proportional (P): self.kp * error

The proportional term responds proportionally to the current error. Larger errors produce larger corrections. This provides the primary driving force toward the target but cannot eliminate steady-state error alone since the output becomes zero when error is zero. Steady-state error is the persistent offset that remains when the system settles. For instance, if the robot needs to reach exactly 48 inches but consistently stops at 47.5 inches due to friction, that 0.5 inch difference is steady-state error. With only proportional control, as the error approaches zero, the output voltage becomes too small to overcome friction, leaving the robot stuck slightly away from the target.

Integral (I): self.ki * self.integral

The integral term accumulates error over time, which eliminates steady-state error by continuing to increase output even when the proportional term is small. Our implementation includes a decay factor self.integral *= self.integral_decay to prevent windup, where the integral grows too large and causes overshoot. The integral is also clamped to a maximum value to further prevent excessive accumulation.

Derivative (D): self.kd * derivative

The derivative term responds to the rate of change of error, calculated as (error - prev_error) / dt. This provides damping to reduce overshoot and oscillation. When the error is decreasing rapidly (high rate of change), the derivative term applies a counter-force to prevent the system from overshooting the target.

Tuning:

To tune the three constant terms, we use this method which proved reliable and fast in the past, based on this StackExchange discussion: https://robotics.stackexchange.com/questions/167/what-are-good-strategies-for-tuning-pid-loops

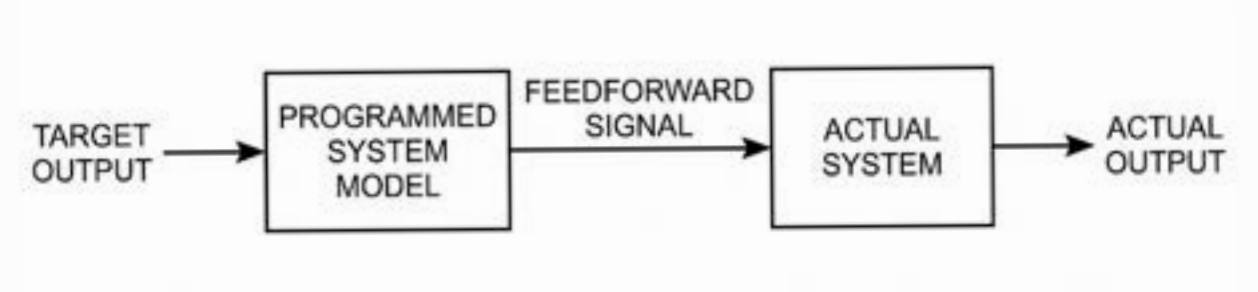
1. Start with 0 for ki, kd and a small number for kp.
2. Increase kp until you get consistent oscillation around the target
3. Increase kd until oscillations stop
4. Repeat steps 2 and 3 until increasing kd does not stop the oscillations
5. Set kp, kd to the last stable values

Only add ki if you see a steady-state error (start small)

FEEDFORWARD CONTROL

Glae Alejo | 5/19/2025

Feedforward control predicts what control input is needed based on a model of the system, without waiting to measure the error. It acts on knowledge of the system dynamics and the desired trajectory, providing open-loop control that anticipates what will be needed. Feedforward control is proactive, acting before errors occur by predicting what the system will need. In order to predict the system, a model of the system is required, that is, understanding of the system's physics such as mass, friction, and motor characteristics. Feedforward provides fast response since there is no feedback delay from waiting to measure the actual output. However, feedforward cannot correct for disturbances or errors in the system model, such as motor overheating or unexpected surface friction. We combine feedforward with feedback control to cover the weaknesses of both, using feedforward for speed and feedback for accuracy.



In practice, our robot code implements feedforward control through 2D motion profiling and trapezoidal motion profiling. The system calculates the required velocities and voltages ahead of time based on the desired path, given real-world constraints (in code, this shows up as constant gains) on acceleration, voltage and the curvature at certain spots on the path (the robot can only turn so fast). These required velocities and voltages are then sent to our RAMSETE controller or PTP controller, which uses these values to drive the motors while also correcting for any errors between where the robot actually is and where it should be on the path with the separate feedback component.

PATH FOLLOWING CHOICE

Glae Alejo | 05/19/2025

Path planning decides where the robot should go and at what speed, while path following ensures the robot actually tracks that plan on the field. We will focus on our path following solution for now. Good path following minimizes time-to-goal without sacrificing stability or accuracy. The path following solution determines how smooth the robot's motion is, how it handles curvature and constraints as provided by feedforward constants and the generated trajectory, and how robust it is to disturbances like motor overheating or battery voltage fluctuation.

The terms below may cause confusion, but are simple once explained.

Time Parameterization
This refers to the process of assigning a timestamp and velocity to every point along the geometric path. While motion profiles are normally generated based on distance steps (calculating velocities for every inch of travel), the result is a "schedule" that dictates exactly when the robot should arrive at each point. This converts a static shape into a dynamic trajectory.

Geometric Tracking
These are algorithms that calculate control outputs based purely on spatial relationships–such as the distance and angle to a specific "lookahead point"–without considering the time schedule or the robot's velocity. Pure Pursuit is a prime example; it only cares about where the robot is, not when it gets there. Similarly, PID control proportionally responds entirely based on error, the robot's current deviation from the goal point.

Nonlinear Control
This describes a feedback system where the corrective response is not strictly proportional to the error but changes dynamically based on the robot's current state. In our case, the controller gains automatically adjust based on the robot's linear and angular velocities. This allows the controller to react aggressively when moving slowly to correct errors, but dampen its response when moving at high speeds to prevent oscillation.

OPTIONS EVALUATED

Approach	Type	Pros	Cons
Pure Pursuit	**Geometric Path follower**	• Widely used in VEX • Continuous lookahead smooths tracking • Conceptually simple	• Can cut corners on sharp turns • Performance depends on lookahead distance selection • Limited tuning • No time parameterization
RAMSETE Controller	**Nonlinear tracking (differential drive)**	• Strong convergence properties on trajectories • Handles curvature and robot kinematics well • Good at high-speed tracking	• Requires time-parameterized trajectories • More complex math and tuning • Assumes reasonably accurate robot model
Point-to-point + Dual PID (Distance, Heading)	**Geometric + PID**	• Simple to implement and reason about • Robust with well-tuned PIDs • Works with sparse waypoints • Easy to constrain voltage and add settle conditions	• Slow at corners, since it needs to turn before moving to the next point • No time parameterization

FINAL CHOICE: PTP, RAMSETE

We are standardizing on a hybrid of Point-to-point (PTP) traversal using PID and RAMSETE.

- **Why PTP + PID**: It provides reliable stopping behavior with settle conditions and stall protection, and is the fastest path to robust straight and short segment motions. Drive distance error and heading error are controlled independently with tuned PIDs, and we clamp voltages and enforce timeouts for safety. This yields predictable, repeatable point-to-point motion with minimal overhead.
- **Why RAMSETE**: For curved, time-parameterized trajectories at higher speeds, RAMSETE's nonlinear feedback law provides better stability and convergence than pure geometric followers, particularly under model errors and disturbances. It uses a trajectory with known pose, velocity, and curvature over time, improving path fidelity and speed while maintaining control.

We did not select **Pure Pursuit** as a primary method because it can cut corners on sharp turns, is sensitive to lookahead tuning across different speeds, and does not enforce a time-parameterized schedule, which we want for consistent, high-speed autonomous routines.

PATH GENERATION CHOICE

Glae Alejo | 05/20/2025

To run a path following algorithm like RAMSETE, we first need to generate the path geometry. The robot needs a continuous mathematical curve to follow, rather than just a list of discrete waypoints. The choice of spline generation algorithm dictates how smooth the path is and, crucially, how much control we have over the robot's heading at specific waypoints. We need a solution that allows us to define specific entry and exit angles for game-specific interactions (like aligning with a goal).

OPTIONS EVALUATED

Approach	Description	Pros	Cons
Bezier Curves	**Uses "control points" that pull the curve toward them to define shape**	• Very smooth curvature (C2 continuity, meaning there is little risk of sudden jumps in acceleration)	• Control points are unintuitive to tune; hard to enforce specific heading constraints at waypoints
Standard Catmull-Rom	**Passes through all points automatically; curvature is determined by previous/next points**	• Easy to implement; no extra "control points" needed	• Rigid; cannot manually define a start/end heading (the curve dictates the angle)

OPTIONS EVALUATED

Approach	Description	Pros	Cons
Cubic Hermite Splines	**Defined by positions (x,y) and velocity vectors (dx,dy) (tangents) at each point**	• Full control; we can define the exact position AND heading at every waypoint	• More complex math

FINAL CHOICE: HYBRID HERMITE SPLINE

We selected **Hermite Splines** as our primary path generation method. The ability to define arbitrary tangent vectors allows us to enforce specific headings at critical points–for example, ensuring the robot is perfectly straight when approaching the match loader, regardless of the curve leading up to it.

To mitigate the tedium of defining tangents for every intermediate point, we implemented a hybrid system. If a specific heading on a waypoint is not defined in the autonomous routine, our code defaults to a **Centripetal Catmull-Rom** heuristic to auto-calculate the optimal tangent based on the surrounding points. This gives us the best of both worlds: the ease of use of a Catmull-Rom spline for general movement, with the strict control of a Hermite spline when precise alignment is required.

MOTION PROFILE CHOICE

Glae Alejo | 05/21/2025

Once the path geometry is generated, we must determine the velocity at every point along that curve. This is Motion Profiling. A path is just a line; a trajectory is a path with a schedule. We need a profiler that maximizes speed while respecting the physical limits of the robot. If we simply drive based on distance, we neglect the significant energy required to rotate the robot mass. We need a solution that accounts for both translation and rotation to generate accurate feedforward values.

OPTIONS EVALUATED

Approach	Description	Pros	Cons
Bezier Curves	**Uses "control points" that pull the curve toward them to define shape**	• Very smooth curvature (C2 continuity, meaning there is little risk of sudden jumps in acceleration)	• Control points are unintuitive to tune; hard to enforce specific heading constraints at waypoints
Standard Catmull-Rom	**Passes through all points automatically; curvature is determined by previous/next points**	• Easy to implement; no extra "control points" needed	• Rigid; cannot manually define a start/end heading (the curve dictates the angle)

OPTIONS EVALUATED

Approach	Description	Pros	Cons
Cubic Hermite Splines	**Defined by positions (x,y) and velocity vectors (dx,dy) (tangents) at each point**	• Full control; we can define the exact position AND heading at every waypoint	• More complex math

FINAL CHOICE: HYBRID HERMITE SPLINE

We selected **Hermite Splines** as our primary path generation method. The ability to define arbitrary tangent vectors allows us to enforce specific headings at critical points—for example, ensuring the robot is perfectly straight when approaching the match loader, regardless of the curve leading up to it.

To mitigate the tedium of defining tangents for every intermediate point, we implemented a hybrid system. If a specific heading on a waypoint is not defined in the autonomous routine, our code defaults to a **Centripetal Catmull-Rom** heuristic to auto-calculate the optimal tangent based on the surrounding points. This gives us the best of both worlds: the ease of use of a Catmull-Rom spline for general movement, with the strict control of a Hermite spline when precise alignment is required.

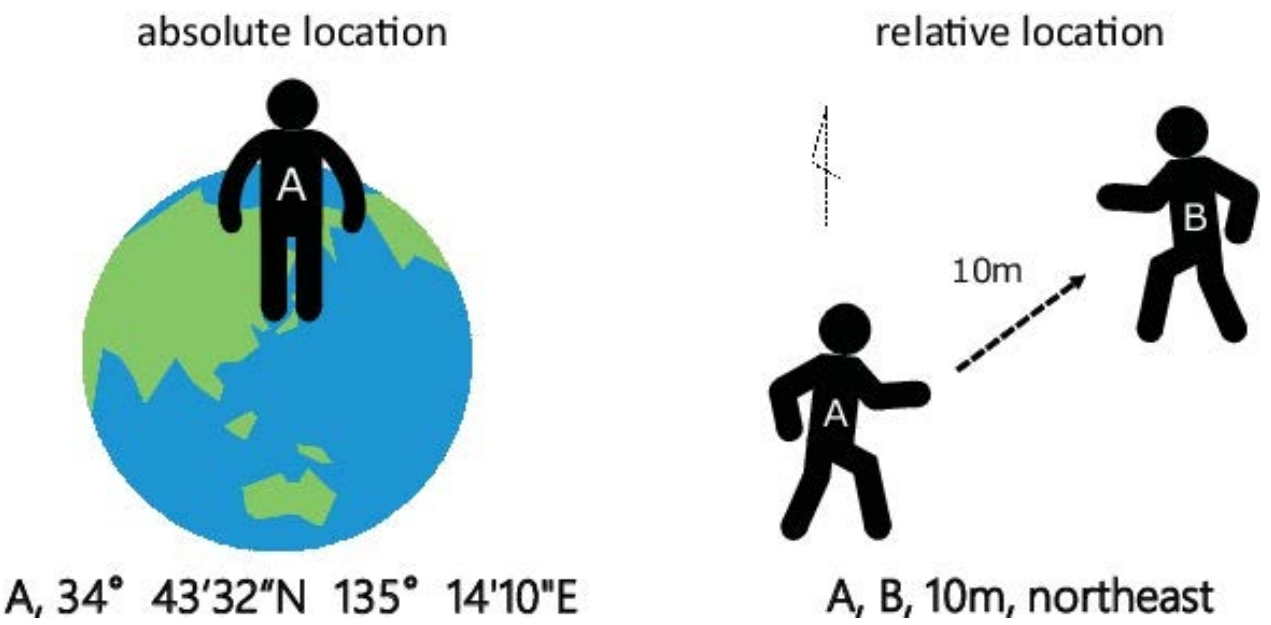
ROBOT LOCALIZATION

Glae Alejo | 5/24/2025

Path planning and motion control are useless if the robot does not know where it is on the field. This is the problem of Localization: determining the robot's global pose (x, y, theta) relative to the field's coordinate system. In VEX, we do not have access to global GPS or external motion capture systems; the robot must determine its location using only onboard sensors. To solve this effectively, we must understand the two fundamental categories of positioning systems available to us and use them to cover their respective weaknesses: Relative and Absolute.

Relative Positioning (Dead Reckoning)
Relative positioning calculates the robot's current position based on its previous position and the measured change in motion. In our setup, this is handled by our Odometry system, which utilizes unpowered tracking wheels and the V5 Inertial Sensor. The primary advantage of relative positioning is speed and smoothness. Our encoders provide data at a very high frequency with negligible latency, allowing the control loop to react instantly to robot movement. However, the critical flaw is unbounded error accumulation. Because we calculate position by integrating velocity over time, any minor sensor noise or wheel slip is added to the total. If the robot slips 1/4" every second, after 30 seconds, it is off by 7 1/2". This error never corrects itself; it only grows.

Absolute Positioning
Absolute positioning determines the robot's location by referencing fixed, known landmarks in the environment. In real life, the Global Positioning System (GPS) would be the most prevalent example of an absolute positioning system, which uses a constellation of satellites to triangulate a receiver's position on Earth. In the context of a VEX field, the most reliable landmarks are the field perimeter walls. We use distance sensors to measure the gap between the robot and the walls to triangulate our position. The advantage here is bounded error. An absolute measurement does not depend on where the robot was five seconds ago; it only cares about what the sensors see right now. This allows us to "reset" or correct the drift generated by the relative system. However, absolute sensors are often slower, noisier, and can be confused by opaque obstructions like field elements blocking the view of the walls.



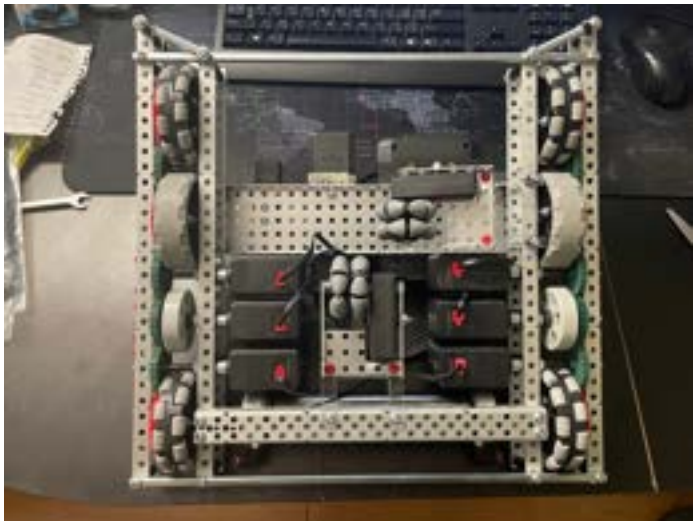
The Solution: Sensor Fusion
Relying on relative positioning alone guarantees drift. Relying on absolute positioning alone results in noisy data. Our solution is a Sensor Fusion pipeline that treats the different physical quantities differently based on sensor reliability.
For **Heading (theta)**, we trust the V5 Inertial Sensor implicitly. It is accurate enough that we do not need to estimate rotation using particles.
For **Position (x, y)**, we implement an **Augmented Monte Carlo Localization** (aMCL) filter (https://cs.gettysburg.edu/~tneller/cs371/17sp/mcl/2017/dmy/index.html):

- **Prediction**: We use the high-frequency Odometry data from our unpowered wheels connected to rotation sensors to move a cloud of 384 virtual "particles" around the internal map.
- **Correction**: We use the four distance sensors to check the distance to the field perimeter. Particles that match the sensor readings are kept; particles that don't are discarded.
- **Injection**: If the filter detects it is lost (using the Augmented MCL "slow/fast" weight average method), it automatically injects random particles to re-find the robot.
- **Final Estimate**: The final estimated pose of the robot is determined based on the average position of all 384 particles.

ODOMETRY IMPLEMENTATION

Glae Alejo | 5/25/2025

Our Odometry system is the "Prediction" step of our localization pipeline. It runs every 10ms and calculates the relative change in position. We use a standard arc-tracking algorithm, but with a critical correction for tracking wheel placement.



Wheel Configuration
We use two unpowered 2" omni-wheels: one parallel to the drive (measuring forward/backward motion) and one perpendicular (measuring strafe). Since these wheels are not at the center of rotation, they register movement simply by the robot turning.

The Algorithm

1. **XY Delta:** We calculate the raw distance traveled by the parallel (dp) and perpendicular (ds) wheels.
2. **Heading Delta:** We calculate the change in heading (dtheta) using the VEX V5 Inertial Sensor.
3. **Offset Correction:** We mathematically remove the "ghost movement" caused by the wheel offsets.
4. **Global Rotation:** We rotate this local vector by the robot's average heading to map it to the field coordinate system.

CODE

```
    pub fn update(
        &mut self,
        parallel_revs: Position,
        perpendicular_revs: Position,
        imu_heading_rad: f64,
    ) {
        let par = parallel_revs;
        let perp = perpendicular_revs;
        let p = par.as_revolutions();
        let q = perp.as_revolutions();

        let dp = (p - self.prev_parallel_revs) *
self.tracking_wheels.wheel_diameter * PI;
        let ds = (q - self.prev_perpendicular_revs) *
self.tracking_wheels.wheel_diameter * PI;

        self.prev_parallel_revs = p;
        self.prev_perpendicular_revs = q;

        let theta = self.normalize_angle(imu_heading_rad);
        let dtheta = self.normalize_angle(theta -
self.pose.heading);
        // compensate rotation-induced wheel travel
        let dx_robot = dp - dtheta *
self.tracking_wheels.perpendicular_offset;
        let dy_robot = ds + dtheta *
self.tracking_wheels.parallel_offset;

        self.pose.x += dx_robot * self.pose.heading.cos() -
dy_robot * self.pose.heading.sin();
        self.pose.y += dx_robot * self.pose.heading.sin() +
dy_robot * self.pose.heading.cos();
        self.pose.heading = theta;
    }

}
```

MCL IMPLEMENTATION

Glae Alejo | 5/26/2025

Our Monte Carlo Localization (MCL) algorithm is a highly optimized Particle Filter designed to run within the constrained cycle time of the V5 brain. While standard particle filters are computationally expensive, we achieved real-time performance by leveraging Rust's low-level memory control and SIMD (Single Instruction, Multiple Data) intrinsics.

Data Structure: Structure of Arrays (SoA)
Standard implementations use an Array of Structs (each particle is an object with x, y, theta). We inverted this to use a Structure of Arrays. We store all X coordinates in one contiguous vector, all Ys in another, and all weights in a third. This memory layout is cache-friendly and allows us to load four floating-point values into a single CPU register at once. We use the core::simd::f32x4 type to process four particles simultaneously for operations like summation and weight normalization.

Step 1: The Motion Model (Prediction)
Every control loop (10ms), we take the relative displacement (dx, dy) from the Odometry. We apply this displacement to every single one of the N=384 particles. Crucially, we add proportional Gaussian noise (MOTION_NOISE_ALPHA) to each particle's movement. If the robot moves 10 inches, we might randomly add +/- 0.5 inches of error to the particles. This artificial "blurring" represents the growing uncertainty of our dead reckoning.

Step 2: The Sensor Model (Likelihood Field)
We utilize a "Likelihood Field" model for our four distance sensors. We only trigger this step if the robot has traveled at least 2 inches to prevent sensor noise from destabilizing the filter while stationary.

- Raycasting: For every particle, we calculate the theoretical distance to the nearest field wall based on the particle's hypothetical position and the sensor's mounting angle.
- Scoring: We compare the theoretical distance to the actual sensor reading. We map this error to a probability using a Gaussian distribution centred at zero error.
  - Perfect match = High Weight.
  - Large discrepancy = Near-Zero Weight.

Step 3: Augmented Resampling (The "Kidnapped Robot")
Standard MCL fails if the robot is picked up and moved (the "kidnapped robot" problem), as all particles remain clustered at the old location. We implemented Augmented MCL, which tracks two averages of the particle weights:

w_slow: Long-term average (decays slowly).
w_fast: Short-term average (decays quickly).

If w_fast drops significantly below w_slow, it indicates that none of our particles explain the current sensor data well. The algorithm calculates an injection probability P = max(0, 1 - w_fast / w_slow). Based on this probability, we inject completely random particles scattered across the field during the resampling step, allowing the filter to instantaneously "re-find" the true position.

Step 4: Stochastic Universal Sampling
To create the next generation of particles, we perform resampling using a prefix-sum (cumulative weight) array. We use Stochastic Universal Sampling (SUS), which is akin to spinning a roulette wheel with N equally spaced pointers. This prevents "particle deprivation" (losing diversity) better than standard roulette wheel selection.

CODE

```
// Structure of Arrays (SoA) layout for cache efficiency and SIMD
struct Particles {
    x: Vec<f32>,
    y: Vec<f32>,
    theta: Vec<f32>,
    weight: Vec<f32>,
}

// SIMD Optimization: Summing 4 particles per CPU instruction
fn sum_particle_components(particles: &Particles) -> (f32, f32, f32) {
    let mut sx_simd = f32x4::splat(0.0);
    let mut idx = 0;
    while idx + SIMD_WIDTH <= N {
        sx_simd += f32x4::from_slice(&particles.x[idx..idx + SIMD_WIDTH]);
        // ... (repeat for y and theta)
        idx += SIMD_WIDTH;
    }
    // ... (reduce sum)
}

// The Likelihood Field: Scoring particles based on sensor data
fn update_weight(&mut self, beams: &[Beam], field_half: f32, stddev: f32) {
    for i in 0..self.len() {
        let mut log_w = 0.0;
        for b in beams {
            // Where would the wall be if the robot was at this particle?
            let origin = self.expected_point(i, b);
            let expected_dist = Self::distance_to_wall(origin, field_half);
            let error = (b.distance - expected_dist).abs();

            // Gaussian probability: High error = Low weight
            let p_hit = Self::gaussian(error, stddev);
            let p = Z_HIT * p_hit + Z_RAND;
            log_w += p.ln(); // Sum log-probabilities for numerical stability
        }
        self.weight[i] = log_w;
    }
    // ... (exponentiate and normalize)
}

// Augmented Resampling with Stochastic Universal Sampling
fn resample_step(&mut self, beams: &[Beam]) {
    // ... (prefix sum calculation) ...

    // Injection Logic: If short-term avg < long-term avg, we are lost.
    let random_prob = (1.0 - self.w_fast / self.w_slow).max(0.0);

    for j in 0..N {
        // Random Injection for recovery
        if next_f32(&mut self.rng) < random_prob {
            self.resampled_particles.inject_random(FIELD_HALF);
            continue;
        }

        // Standard Low-Variance Resampling
        let target = (u0 + (j as f32) * INV_N) * total_weight;
        while self.cumulative_weights[i] < target { i += 1; }
        self.resampled_particles.push_copy_from(&self.particles, i);
    }
}
```

HURRICANE ROBOTICS