

210K



Kawaii Kittens

Calgary, Alberta, Canada

Digital Engineering Notebook



Date: Dec 26, 2025

Event Name: Ignite the Northwest

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: 210K

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

We have a unique pistonized hood puncher design, we have a linear slide system on our compression hood. Our hood incorporates a piston to extend and retract—going from not breaking the plane of the long goal—to instantly descoring the long tube to switch blocks into the control zone. It works by a piston mounted on an angle bar powering a screw mounted linear slide section of a hood.

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

Pages 291-295

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

As far as we are concerned, we are one of the few teams that currently incorporates a pistonized hood on a linear slide to decrease descore time and creatively work around the SG10 regulations and GG9 rule updates. We can use it in conjunction with our polycarbonate to effectively score an entire long goal of blocks, or descore only a few blocks from the opposite end of the goal.

Table of Contents

- > Tournaments/Scrimmages are underlined. Most robot versions will be aligned to a upcoming signature event due to their importance.
- > Page numbers correspond to those on the bottom corner of each page
- > All entries will be colour coded based on the engineering design process (see pages 14-15).

VRC Push Back was announced on May 8, 2025.

VØ - Game Analysis & Summer Robot Plans

Creation Date (MM/DD/YYYY)	Topic	pg.
PRESENT	Engineering Design Cycles (Gantt chart)	3-4
May 1, 2025	Quick-Reference Glossary	5-6
May 1, 2025	↳ Team Introduction	7
May 1-2, 2025	Notebook Formatting	8-14
May 2, 2025	Opening Thoughts	15
May 10, 2025	Game Introduction & Analysis	16-23

V1 - Mall of America Signature Event

Creation Date (MM/DD/YYYY)	Topic	pg.
	Drivetrain C1	24-56
May 11, 2025	↳ Informational Setup	25-28
May 11, 2025	↳ Identify the Problem - Drivetrain	29
May 11-12, 2025	↳ Drivetrain Brainstorming	30-31
May 12, 2025	↳ Selecting Drivetrain Type	32-33
May 12, 2025	↳ Identify the Problem - Drivetrain RPM	34

Table of Contents

<u>V1 - Mall of America Signature Event</u>		
Creation Date (MM/DD/YYYY)	Subject	pg.
May 12, 2025	↳ Drive RPM Brainstorming	35
May 12, 2025	↳ Selecting a Drive RPM	36-37
May 12, 2025	↳ Ball Bear Wheels	38-39
May 12, 2025	↳ Drivetrain Planning	40
May 15-17, 2025	↳ Building the Drivetrain	41-55
May 19, 2025	↳ Testing the Drivetrain	56
	Intake C1	57-90
May 22, 2025	↳ Picking up Game Elements from the Ground	58
May 22, 2025	↳ Brainstorming Intake Designs (based on past games)	59-60
May 23, 2025	↳ Choosing the Intake Roller Shape	61-62
May 25, 2025	↳ Planning our Intake	63
May 29-30, 2025	↳ Building the Bottom Stage Intake	64-75
June 1, 2025	↳ Moving the Elements Through our Robot	76
June 2, 2025	↳ Brainstorming Intake Systems	77
June 2, 2025	↳ Selecting an Intake System	78-79
June 7, 2025	↳ Planning our Intake System and Rollers	80
June 10, 2025	↳ Building the Intake System	81-90

Table of Contents

Creation Date (MM/DD/YYYY)	Subject	pg.
	Top Stage/Scoring C1	91-115
June 15, 2025	↳ How do we Score on Multiple Heights	92
June 17, 2025	↳ Brainstorming ways to Score on Different Heights	93
June 17, 2025	↳ Selecting a Design for Scoring	94
June 18, 2025	↳ Planning a Piston Lift	95
June 21, 2025	↳ Building the Piston Lift	96
June 23, 2025	↳ Planning Top Stage Intake	97
June 25-30, 2025	↳ Building the Top Stage Intake	98-105
June 30, 2025	↳ Brainstorming Block Transfer	106
July 1, 2025	↳ Planning Block Transfer	107
July 1, 2025	↳ Building Block Transfer	108-109
July 1, 2025	↳ Identify the Problem - Storing Blocks	110
July 1, 2025	↳ Brainstorming Methods of Storage	111
July 1, 2025	↳ Selecting a Design for Storage	112
July 1, 2025	↳ Planning our Trapdoor for Storage	113
July 1, 2025	↳ Building the Trapdoor	114-115

Table of Contents

Creation Date (MM/DD/YYYY)	Subject	pg.
	Matchloader C1.x	116-137
June 30, 2025	↳ Identify Challenges With Matchloading	117
July 2, 2025	↳ Brainstorming Matchloading Mechanisms	118-119
July 2, 2025	↳ Selecting a Matchloading Mechanism	120-121
July 3-4, 2025	↳ Building the Matchloading Mechanism (C1.0)	123-128
July 6, 2025	↳ Testing the Matchloader (C1.0)	129-130
July 6, 2025	↳ Building a New Matchloader Design (C1.1)	131
July 6, 2025	↳ Testing Matchloader (C1.1)	132-133
July 6, 2025	↳ Building Matchloader (C1.2)	134
July 6, 2025	↳ Testing Matchloader (C1.2)	135-136
	Intake Adjustments	137-142
July 7, 2025	↳ Testing Intake	137-138
July 7, 2025	↳ Issues With the Intake	139
July 7, 2025	↳ Brainstorming Designs	140
July 7, 2025	↳ Planning a new Intake Design	141
July 8, 2025	↳ Rebuilding Intake	142

Table of Contents

<u>V1 - Mall of America Signature Event</u>		
Creation Date (MM/DD/YYYY)	Subject	pg.
	Intake Tuning	143-151
July 13, 2025	↳ Issues with scoring	143
July 14-15, 2025	↳ Tuning Scoring	144-145
July 15, 2025	↳ Issues with the Trapdoor	146
July 15, 2025	↳ Brainstorming a Clutch	147
July 16, 2025	↳ Planning a Clutch and Ratchet Design	148
July 17, 2025	↳ Building the Clutch and Ratchet	149-150
July 17, 2025	↳ Pneumatics Layout	151
	Odometry	152-157
July 19, 2025	↳ Issues with Positioning Accuracy	152
July 19, 2025	↳ Researching Odometry Pod Designs	153
July 19, 2025	↳ Selecting Odometry Pod Design	154
July 21, 2025	↳ Planning Odometry Pods	155
July 21, 2025	↳ Building Odometry Pods	156
July 22, 2025	↳ Testing Odometry Pods	157
July 22, 2025	↳ Odometry Pods Clipping	158
July 22, 2025	↳ Piston Lift	159
July 23, 2025	↳ Odometry Pod Lift	160-161

Table of Contents

V1 - Mall of America Signature Event

Creation Date (MM/DD/YYYY)	Subject	pg.
	Skills and Strategy	162-168
July 24-26, 2025	↳ Analyzing the Skills Challenge Rules & Scoring	162-164
July 26-28, 2025	↳ Creating Paths & Practicing for Driver Skills	165-167
July 29, 2025	↳ Strategies and Win Conditions	168
July 29, 2025	Design Overview	169-170
	Mall of America Signature Event	171-176
July 29, 2025	↳ Planning & Goal-Setting for MOA Signature Event	171
Aug 1-2, 2025	↳ MOA Signature Event Matches	172-175
Aug 3, 2025	↳ MOA Results and Summary	176

V2 - Post-MOA Rebuild

Creation Date (MM/DD/YYYY)	Topic	pg.
	Post-MOA Considerations	177-181
Aug 3, 2025	↳ Identify the Problem - MOA Robot	178-180
Aug 14, 2025	↳ Selecting the Drivetrain	181

Table of Contents

<u>V2 - Post-MOA Rebuild</u>		
Creation Date (MM/DD/YYYY)	Topic	pg.
	Intake C2 and Framing	182-215
Aug 16, 2025	↳ Researching Intake Types	182-183
Aug 16, 2025	↳ Selecting an Intake Design	184
Aug 20, 2025	↳ Planning Our intake	185
Aug 26-31, 2025	↳ Building Structure	186-200
Aug 31-Sep 3, 2025	↳ Building Intake	201-208
Sept 4, 2025	Sept Game Manual Updates	209
Sept 4, 2025	↳ Identify the Problem - Ratchets	210
Sept 4, 2025	↳ Researching Ratchets	211-212
Sept 4, 2025	↳ Selecting Ratchet Design	213
Sept 5, 2025	↳ Building Ratchets	214-215
	Parking and Alignment	216-220
Sept 8, 2025	↳ Identify the Problem - Alignment and Parking	216
Sept 8, 2025	↳ Planning Designs using CAD	217
Sept 9, 2025	↳ Create Polycarbonate Pieces	218-219
Sept 9, 2025	↳ Attaching wheels	220
	Intake C2 - Hood and Hoard	221-239
Sept 9, 2025	↳ Blocking and Descore	221
Sept 9, 2025	↳ Researching Methods of Descore and Blocking	222

Table of Contents

<u>V2 - Post-MOA Rebuild</u>		
Creation Date (MM/DD/YYYY)	Topic	pg.
Sept 9, 2025	↳ Selecting a Hood Design	223
Sept 10, 2025	↳ Planning Our Hood	224
Sept 10, 2025	↳ Building Hood	225-228
Sept 11, 2025	↳ Planning Basket	229
Sept 11, 2025	↳ Building our Basket	230-231
Sept 11, 2025	Double Park	232
Sept 13-14, 2025	Highlanders Summit Game Analysis	233-237
Sept 16, 2025	↳ Building Tensioner	238
Sept 22, 2025	↳ Testing Intake System	239
	Tuning Intake C2	240-243
Sept 22, 2025	↳ Identify the Problem - Intake System Issues	240
Sept 22, 2025	↳ Planning Designs using CAD	241
Sept 24, 2025	↳ Hood Changes	242-243
	Intake C2 - Basket	244-250
Sept 25, 2025	↳ Basket Improvements	244-245
Sept 26, 2025	↳ Testing Basket	246
Sept 26, 2025	↳ Basket Issues	247

Table of Contents

<u>V2 - Post-MOA Rebuild</u>		
Creation Date (MM/DD/YYYY)	Topic	pg.
Sept 26, 2025	↳ Brainstorming Solutions for Basket Jamming	248
Sept 10, 2025	↳ Selecting a Solution for Jamming	249
Sept 10, 2025	↳ Planning an Agitator	250
	Polycarbonate	251-253
Sept 26, 2025	↳ Issues with Polycarbonate	251
Sept 26, 2025	↳ Redesigning Sleds	252
Sept 27, 2025	↳ Reattaching Sleds	253
	Matchloading C2	254-264
Sept 28, 2025	↳ Matchloading - How can we improve it?	254
Sept 28, 2025	↳ Researching Possible Designs	255-256
Sept 28, 2025	↳ Selecting a Matchloading Mechanism Design	257
Sept 28, 2025	↳ Planning our Matchloading Mechanism	258
Sept 29, 2025	↳ Building Matchloading Mechanism C2.0	259-260
Sept 30, 2025	↳ Testing Matchloading Mechanism C2.0	261
Sept 30, 2025	↳ Issues with Matchloading Mechanism C2.0	262
Oct 3, 2025	↳ Building Matchloading Mechanism C2.1	263
Oct 3, 2025	↳ Testing Matchloading Mechanism C2.1	264

Table of Contents

<u>V2 - Post-MOA Rebuild</u>		
Creation Date (MM/DD/YYYY)	Topic	pg.
	Intake C2 Tuning	265-278
Oct 4, 2025	↳ Intake Issues	265
Oct 4, 2025	↳ Fixing the Intake	266
Oct 5, 2025	↳ Tuning the Basket	267
Oct 5, 2025	↳ Testing Intake	268-269
Oct 5, 2025	↳ Planning Funnels	270
Oct 5, 2025	↳ Building Funnels	271
Oct 9, 2025	Game Manual Update	272
Oct 10, 2025	Brainstorming New Hood Designs	273
Oct 11, 2025	↳ Issues with Barrier Cross	274
Oct 11, 2025	↳ Building Support Wheels	275
Oct 14, 2025	↳ General Tuning	276
Oct 15, 2025	↳ Programming SAWP for Signitures and Locals	277-278
	WM Season Opener Tournament	279-283
Oct 17, 2025	WM Season Opener Tournament Goals	279
Oct 17, 2025	WM Season Opener Tournament Recap	280-283
	Intake C3	284-302
Oct 19, 2025	↳ Intake Issues	285

Table of Contents

<u>V2.1 - UC Berkley Signature Event + Locals</u>		
Creation Date (MM/DD/YYYY)	Topic	pg.
Oct 19, 2025	↳ New robot CAD	286
Oct 20-22, 2025	↳ Rebuilding our Intake	287-290
Oct 23, 2025	↳ Revisiting the SG10 & GG9 Rule Updates	291
Oct 23, 2025	↳ Brainstorming Hood Designs	292
Oct 23, 2025	↳ Planning our Hood Design	293-294
Oct 24, 2025	↳ Building our Hood	295
Oct 24, 2025	↳ Testing our Intake Efficiency	296
	Intake C4	297-302
Oct 24, 2025	↳ Intake Jamming Problems	297
Oct 24, 2025	↳ Brainstorming Jamming Solutions	298
Oct 24, 2025	↳ Selecting our Jamming Solution	299
Oct 25, 2025	↳ Planning Intake Jam Fixes	300
Oct 26, 2025	↳ Tuning our Intake	301-302
	Matchloader C3	303-309
Oct 29, 2025	↳ Identifying Previous Matchloader Issues	303
Oct 29, 2025	↳ Planning our new Matchloader	304
Oct 31, 2025	↳ Building New Matchloader	305
Nov 1, 2025	↳ Testing New Matchloader	306

Table of Contents

<u>V2.1 - UC Berkley Signature Event + Locals</u>		
Creation Date (MM/DD/YYYY)	Topic	pg.
Nov 2, 2025	↳ Planning our Polycarbonate	307
Nov 5, 2025	↳ Attaching Polycarbonate Pieces	308
Nov 5, 2025	↳ Testing Polycarbonate Aligners	309
Nov 8, 2025	↳ Identify Intake Issues	310
Nov 13, 2025	↳ Testing Intake Speed	311
Nov 12, 2025	↳ Double Park Improvements	312
Nov 13, 2025	↳ Double Park Time Testing	313
Nov 15, 2025	Removing our Double Park (strategy)	314
Nov 15, 2025	↳ Brainstorming Wing Placements	315
Nov 15-16, 2025	↳ Building our Wing	316-317
	One World @ UC Berkeley Signature Event	318-323
Nov 20, 2025	↳ One World Sig Autonomous Paths	318
Nov 21, 2025	↳ Planning for One World Signature Event	319
Nov 24, 2025	↳ One World @ UC Berk. Signature Recap	320-323
Nov 24, 2025	↳ Viable Autonomous Routines	324
Nov 26, 2025	↳ Wing Aligner Research	325
Nov 27, 2025	↳ Building our Wing Aligner	326

Table of Contents

<u>V3 - Ignite the Northwest</u>		
Creation Date (MM/DD/YYYY)	Topic	pg.
	STEMIA Local Tournament	327-337
Dec 12, 2025	↳ Planning for STEMIA Local	327
Dec 13, 2025	↳ STEMIA Local Results/Recap	328-332
Dec 20, 2025	↳ X'mas Holiday Competition Results/Recap	333-337
	Intake C5, Scoring C5	338-
Dec 15, 2025	↳ Previous Bot Issues	338-340
Dec 15, 2025	↳ Researching Funnels	341
Dec 15, 2025	↳ Selecting Funnels	342
Dec 16, 2025	↳ Researching Mid-Goal Scoring	343
Dec 16, 2025	↳ Selecting Mid-Goal Scoring	344
Dec 16, 2025	↳ Researching Mid-Goal Descoring	345
Dec 16, 2025	↳ Selecting Mid-Goal Descoring	346
Dec 16, 2025	↳ V3 CAD	347
Dec 17, 2025	↳ Drivetrain Framing	348-349
Dec 18-19, 2025	↳ Intake Structure	350-353
Dec 20, 2025	↳ Polishing Drivetrain	354
Dec 20, 2025	↳ Building Hood	355
Dec 20, 2025	↳ Building Intake	356-357
	Start of Coding Logbook	358

Table of Contents

The following table is for our **programming logbook**.

<u>V1 - Mall of America Signature Event</u>		
Creation Date (MM/DD/YYYY)	Topic	pg.
May 1, 2025	Programming Design Process Overview	2
May 1, 2025	Game Analysis	3-4
May 5-6, 2025	Selecting a Development Environment	5-6
May 10, 2025	Code Workflow - Cloud Storage	7-8
May 15-22, 2025	Determining a File Structure	9-14
May 23-27	Implementing our Codebase Structure	15-19
	Codebase C1	20-47
May 4, 2025	Drive Curve Theory	21-24
May 4, 2025	↳ Drive Curve Implementation	25
May 7, 2025	Raw Voltage Control Explanation	26
May 7, 2025	Driver Control Implementation	27
May 7, 2025	Past Evolution of Autonomous Movement	28
May 8-9, 2025	PID Movement Theory	29-35
May 9, 2025	↳ Proportional Controller	30-32

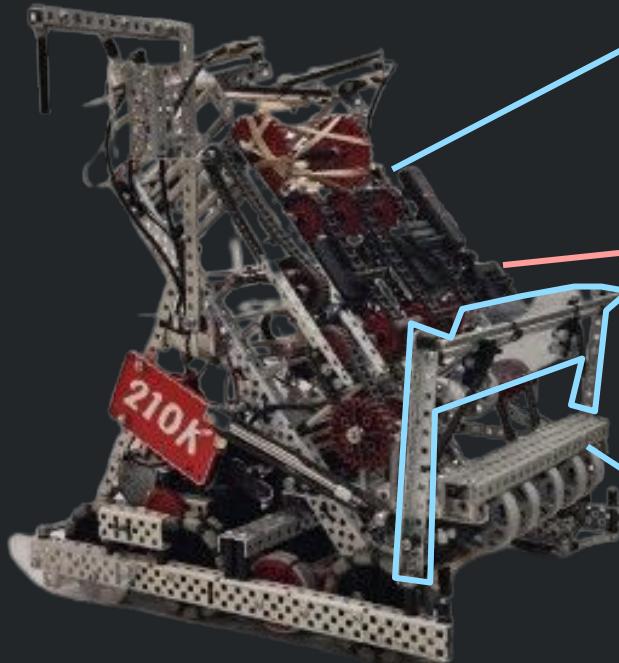
Table of Contents

V1 - Mall of America Signature Event

Creation Date (MM/DD/YYYY)	Topic	pg.
May 9, 2025	↳ Derivative Controller	33
May 9, 2025	↳ Integral Controller	34
May 9, 2025	PID Implementation & Tuning	35-37
May 9, 2025	Comparing Localization Methods	38
May 9-10, 2025	Odometry Theory Explanation	39-43
May 10, 2025	Explaining Multitasking Theory for VEX	44-45
May 13-15, 2025	Demoing our Autonomous Path Planner	46-47
May 20, 2025	Comparing Graphics Libraries	48
June 9-10, 2025	Demoing our LVGL Brain GUI	49-50
June 15-20, 2025	Investigation into Contactless Odometry [LAB]	51-53
July 15, 2025	Code Organization & Flowcharts	54
July 18, 2025	Explaining Distance Sensor Position Resets	55
July 20, 2025	Solo Autonomous WP Path @ MoA	56

Design Overview

Robot Iteration v2.1



1200 RPM Intake

Score blocks from storage into the long goal within 0.8 seconds

276-4852

Forward-facing distance sensor

Used in autonomous position resets and localization

Drop-down “tongue” matchloader

Quickly remove blocks from loaders (all 6 blocks in 1 sec) and snatch hard-to-reach blocks during autonomous

228-2500-230

VEX 6P chains

Less prone to snapping and more flexible around sharp bends than regular chain

276-4831

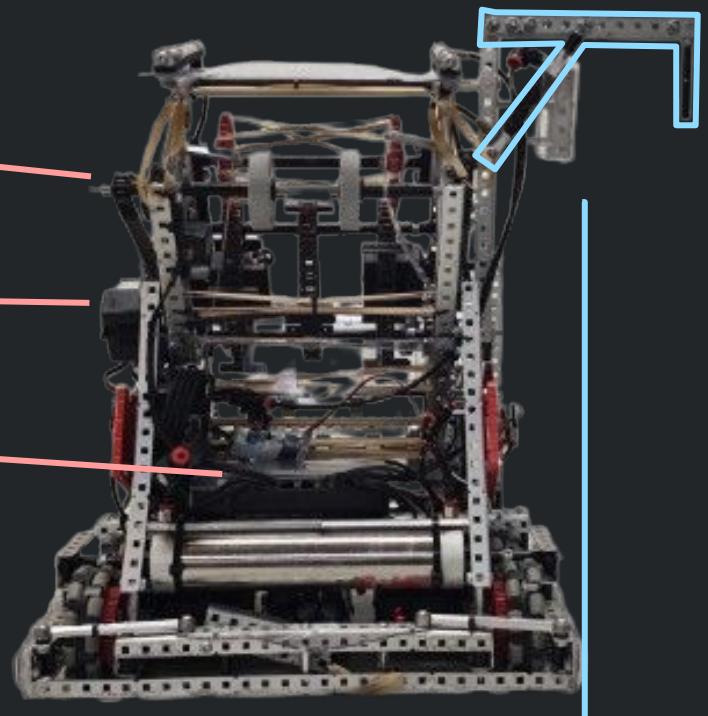
VEX V5 Radio

276-4810

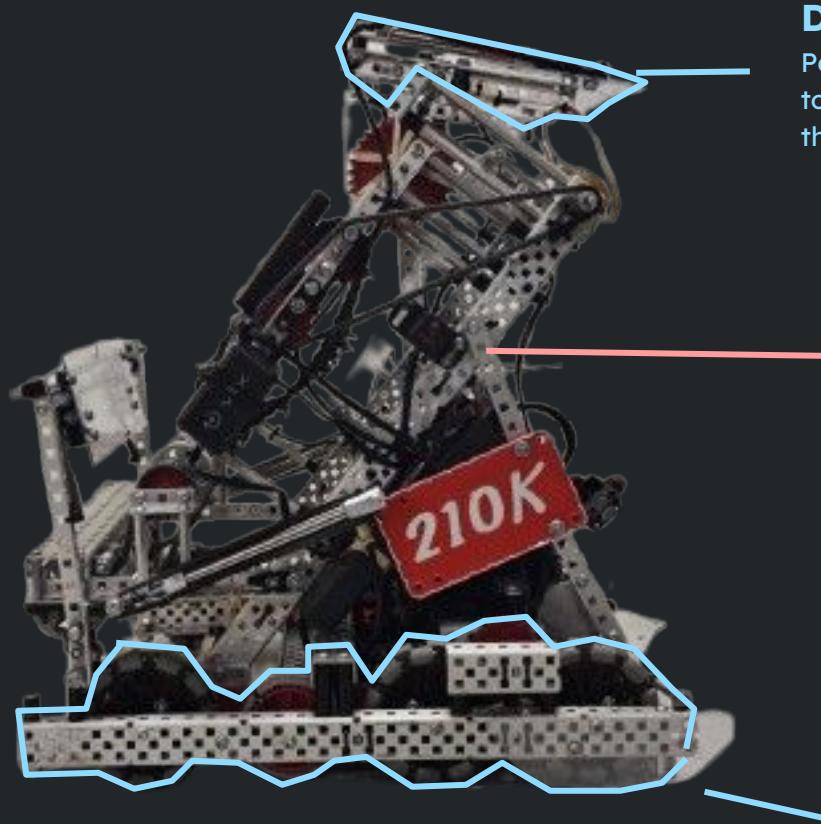
VEX V5 Brain

“Wing” descore mechanism

Piston-actuated lever that reaches into long goals and completely removes all blocks inside it.



Design Overview



Descore “Hood”

Polycarbonate hood can extend laterally to descore blocks out the other end of the long goal with 24lbs of force

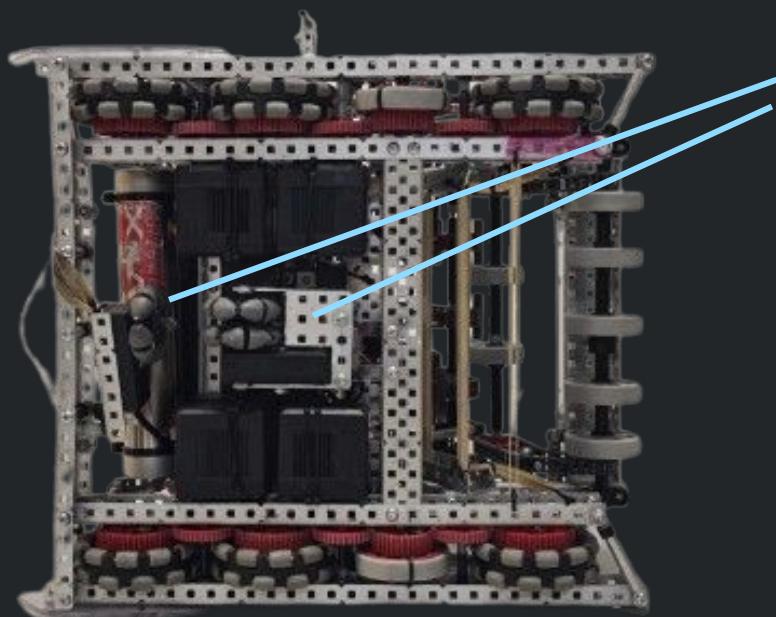
276-7043

Optical Sensor

Vерifies the colours of blocks as they pass by the intake to ensure we never score for the opposing alliance

3.25" 450RPM Drivetrain

Speedy enough to outmanoeuvre opponents while still having strong traction force and resist pinning.

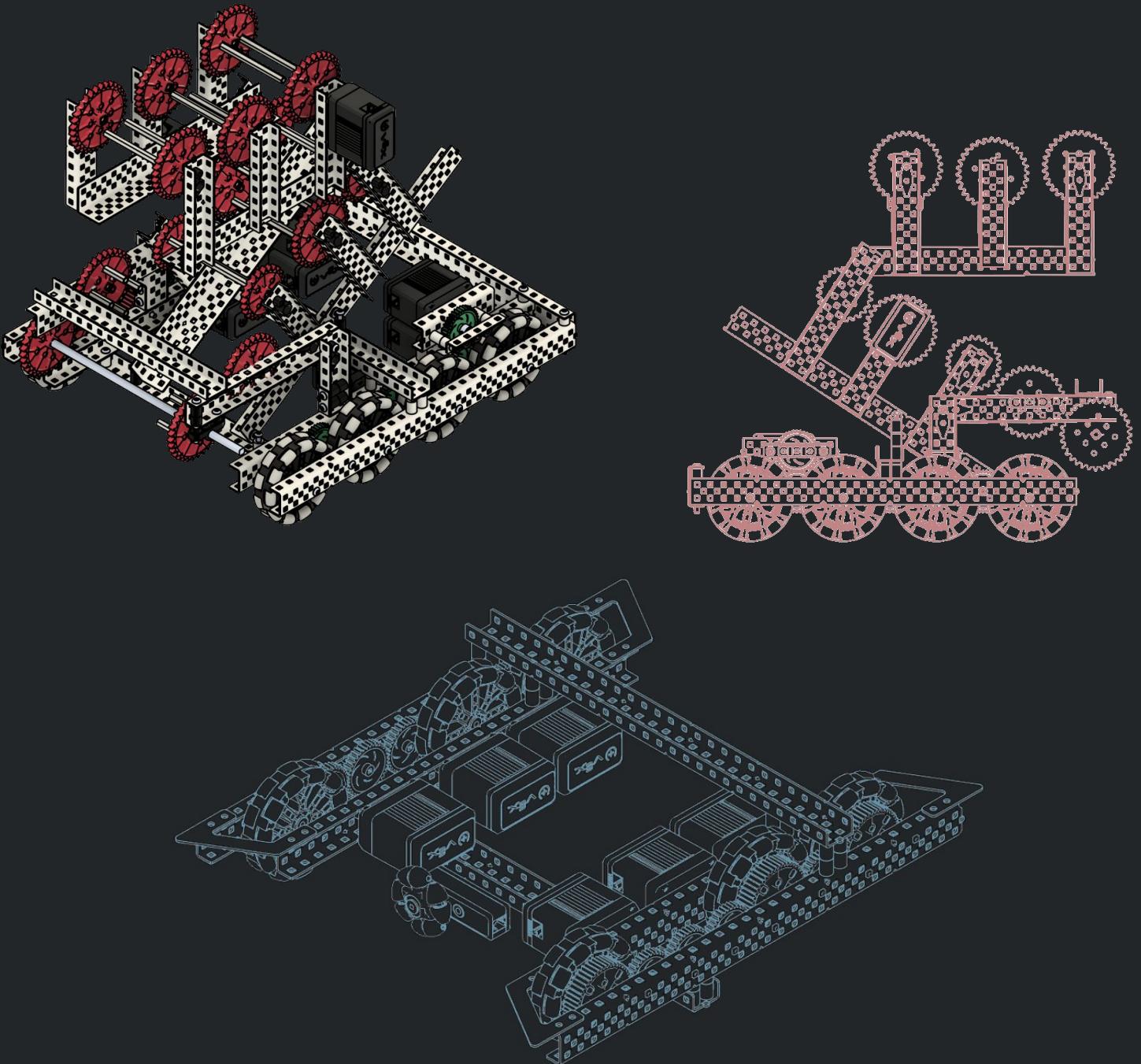


Dual-axis 2" Odometry Trackers

Tensioned free-spinning wheels that always touch the ground to measure the robot's travel laterally and horizontally. These pods automatically lift up when driving over the park barrier.

Digital Mechanical Assembly

Our initial design and prototyping process for each revision of our robot has been iteratively performed using 3D digital CAD software. This process has allowed us to **rapidly visualize ideas, share them with teammates, and help aid the fine-tuning process** of our final solution before starting construction.



Driver Assistance Overview

One main goal of our team this season is to **design with our driver** in mind. Given the extent to which the outcome of higher-level competitive matches depend on the skill of the driver more than the robot itself, it is imperative that our driver has the best tools and assistance macros at their disposal.

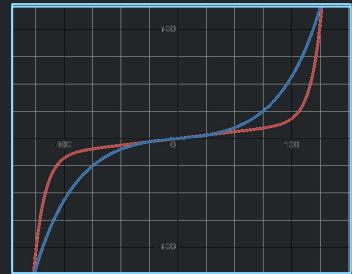
Colour sensor automatic sorting system for intake

Overrides for automatic systems

Button extensions for easy reach

Fully configurable chassis input curves for fine-tuned motion

Haptic feedback to notify of macro statuses

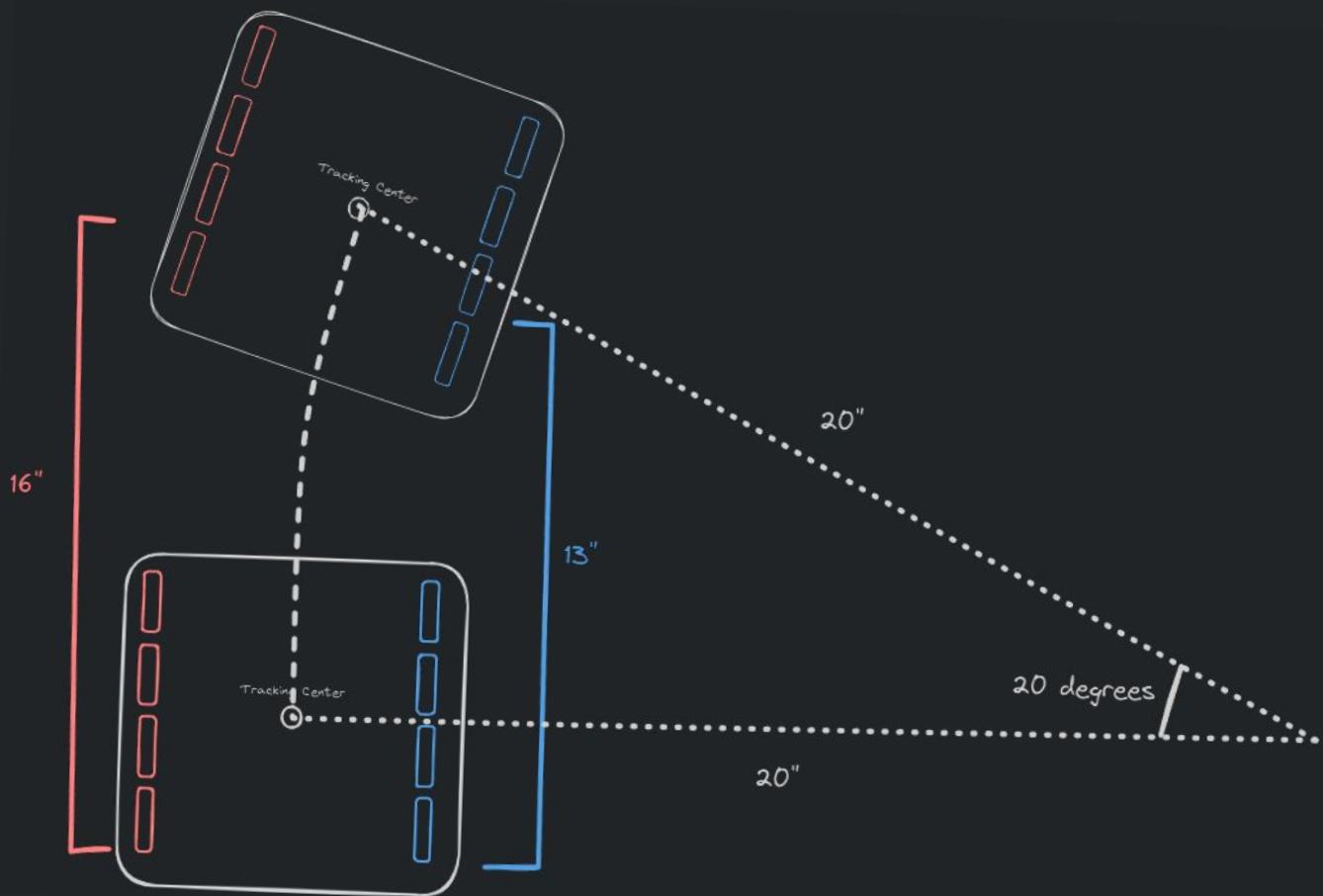


Autonomous Programming

A brief overview of our autonomous control systems.
Please see the programming logbook for details.

Localization

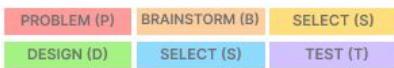
Using our custom IMU based drive odometry system we can accurately model changes in the position and orientation of the chassis in 2D space. Given a known starting position, we calculate the live absolute position of the chassis relative to its starting position. Wheel spin, friction differences, and unpredicted anomalies can lead to the predicted absolute state drifting away from the true state over extended paths such as the 60-second programming skills routine.



Engineering Design Cycles (Gantt Chart)

Management

DESIGN PROCESS STAGES



► Reiteration is shown through the multiple design cycles for each subsystem!

*Section sizes may be exaggerated to better show smaller regions.

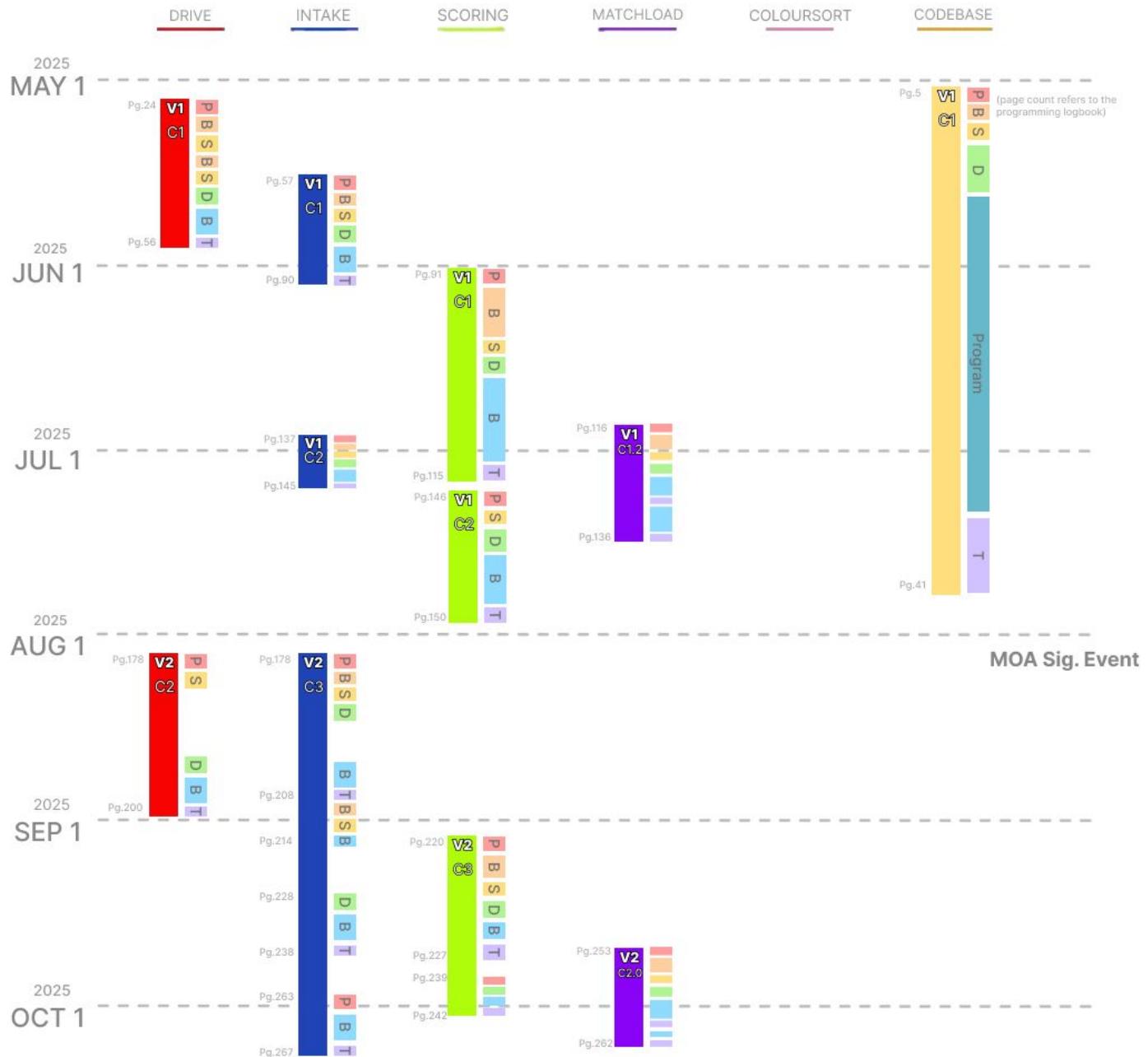
SUBSYSTEMS



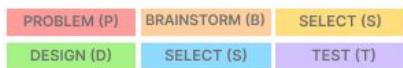
TAGS

V# is Robot Version #
C# is Design Cycle #

► Blurred sections indicate our future plans (to be determined)



DESIGN PROCESS STAGES

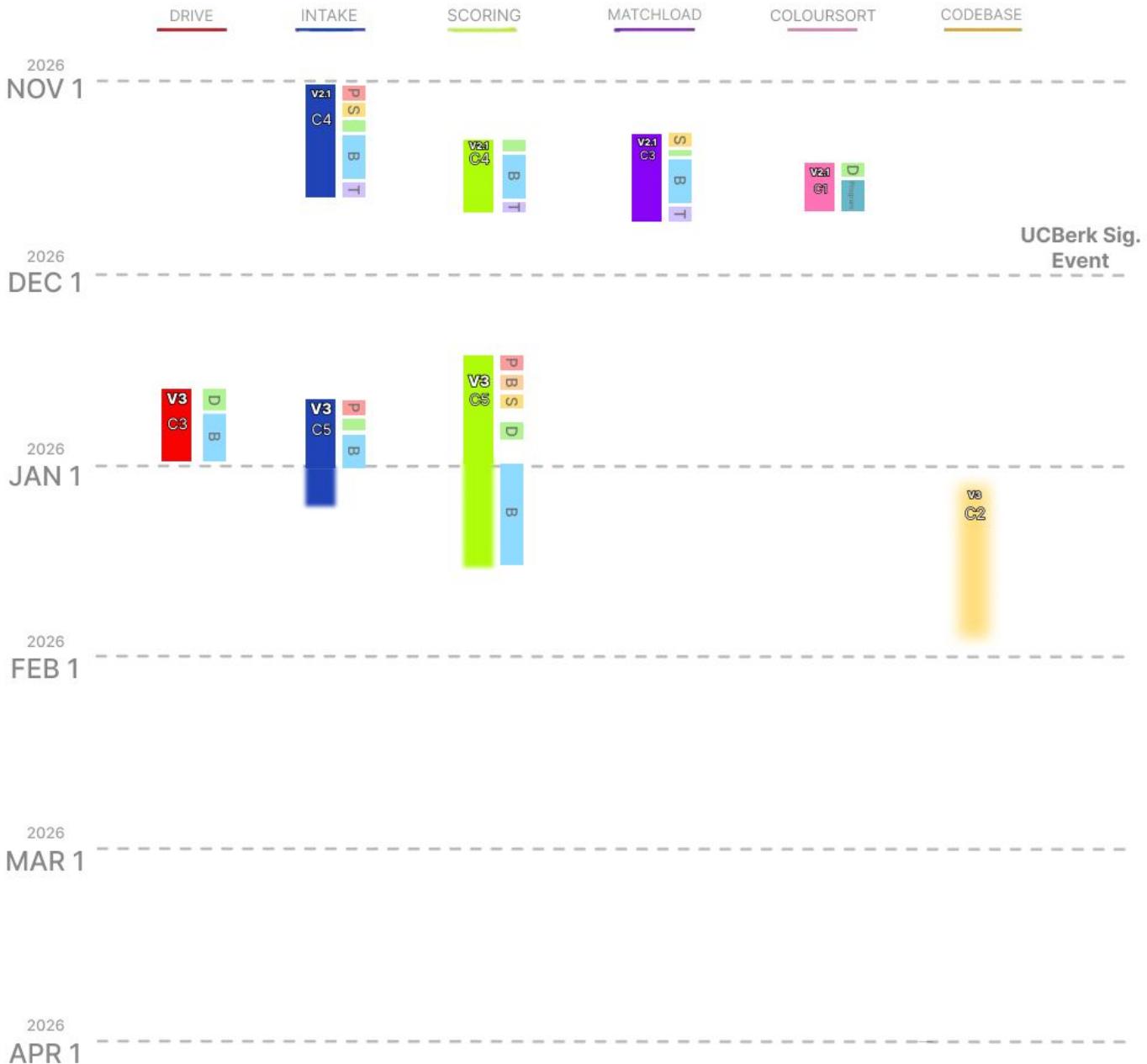


► Reiteration is shown through the multiple design cycles for each subsystem!

*Section sizes may be exaggerated to better show smaller regions.

*Section sizes may be exaggerated to better show smaller regions.

► Blurred sections indicate our future plans.



- ▶ Focus: To define non-standard terminology in robotics.

Date: May 1, 2025

AWP: Acronym for “autonomous win point” earned by completing a specified set of tasks in the autonomous period.

Arcade Drive: A driver control method where one joystick on the controller controls forward/backwards movement and the other controls turning.

Autonomous: Code designed to run during the autonomous period, or may reference the period itself in a match.

- The autonomous period lasts 15 seconds at the beginning of a match, when robots may only operate under the control of its internal code.
- **Auton:** A code routine that is run during the autonomous period. May be changed between qualification and elimination matches.

C: A low level programming language. Offers precise control over how code is executed and how information is stored.

- **C++:** A modern extension of C used in VEX and other robotics programs. It supports organizing code through “objects”.

CAD: Acronym for “computer aided design”. Used in engineering disciplines to plan out physical designs virtually. Used in VEX to digitally create a robot that can be used as a blueprint for physical construction.

CNC mill: A CNC is a digital cutting machine. It moves a high-speed rotating spindle along a pre-generated path, as directed by a CAD file, to cut out material. It is used to cut Lexan with precision and a smooth finish that cannot be achieved with manual cutting.

Drivetrain/DT: The “base of a robot”, consisting of its bottom wheels and the chassis/frame holding the base together.

DQ: Shorthand for disqualification

Field: A 12 ft x 12 ft square area robots compete in, defined by the floor and ~6 inch tall plastic walls. 24 inch x 24 inch foam mats line the floor of the field.

- ▶ **Focus:** To define non-standard terminology in robotics.

Date: May 1, 2025

Git: A software used to sync code between devices, keep track of changes across time and assign versions to software.

- **GitHub:** A platform hosted by Microsoft to publicly share Git code and store it.

H2H/Head to Head: The standard VRC match style between 2 alliances of 2 teams each.

Lexan: A transparent polycarbonate plastic material. Lexan is incredibly durable and resiliant to blunt force and direct impact, yet still flexes and cuts easily with a CNC mill.

Mech: A mechanism designed to do a specific action. For example, a MOGO mech describes an entire unit of pistons and metal for holding and manipulating a MOGO. They may be named off appearance, such as a “backpack mech” or a “grappling hook mech”.

Meta: Builds/strategies/play styles, which are considered optimal and standard at the current time. A meta is used by the majority of VEX teams, particularly in the North American region. These can change as time passes.

PTO/Transmission: Stands for “power take off”. Refers to a mechanism used to share the motor power on a drivetrain with other components on a robot by sliding gears, as a way of working around the motor limit in VRC.

RPM: “Rotations per minute”. A standard measure of speed for a rotating roller or motor

Upset: An upset occurs in a competition, like sports or robotics, when the team popularly expected to win a match (the “favorite”) is defeated by an underdog whom is expected to lose.

Team Introduction

Management

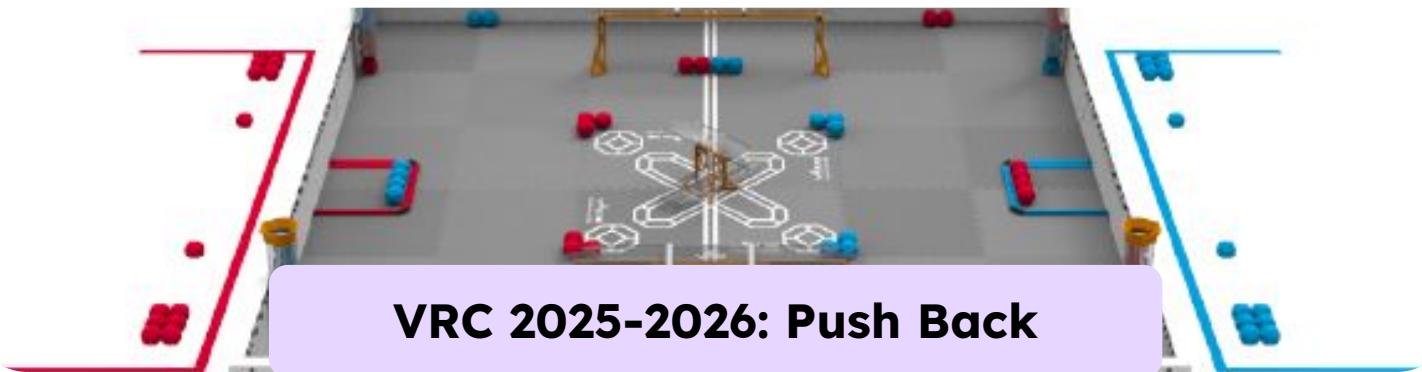
- Focus: Introduce our origins.

Date: May 1, 2024

We are a 4th year high school VEX team from the Western Mechatronics Robotics Club, located in Calgary, Alberta, Canada. We are the product of a merge between teams 2088S and 2088K, both of which have competed in the VEX World Championship. **210K** is composed of five members:

		
Daniel: (4th Year) <ul style="list-style-type: none">- Main Driver- Builder- Logbooker- Scout- Strategist	Brandon: (3rd Year) <ul style="list-style-type: none">- CAD Designer- Drive Team Sub- Programmer- Strategist- Logbooker	Joshua: (4th Year) <ul style="list-style-type: none">- Builder- Drive Team Member- Scout
		
Michael: (4th Year) <ul style="list-style-type: none">- Programmer- Logbooker- Scout	Bryan: (3rd year) <ul style="list-style-type: none">- Builder- Drive Team Member- Logbooker	

We are a dedicated high school robotics team with a strong passion for competing in tournaments, driven by the desire to achieve success and take pride in our accomplishments.



VRC 2025-2026: Push Back

Figure 1: Push Back field from the game manual

Overview: This logbook is made to document our design, building, strategy, and competitions across our entire Push Back competition season, along with our software development and thinking process.

Main Features of Notebook:

- Documentation of our engineering design process and development
- Documentation of team meetings
- Documentation of the robot design, build, and programming
- Documentation of the team's strategy and gameplay

Reasons for Format Design this season:

- Make the notebook more accessible

By using coloured pages, we aim to make it easier for reviewers to access the entries they may be most interested in.

- Make the notebook appearance clean & consistent

This notebook uses the “Theme” feature embedded into Google Slides which ensures that all pages always follow a particular theme layout.

- Enable the team to easily document all information

The organization style of the notebook also allows documentation by the logbookers to be a lot more time conserving and efficient.

Not all entries may appear in chronological order, and some entries may be grouped into a broader project whilst being written later than entries proceeding it from another topic.

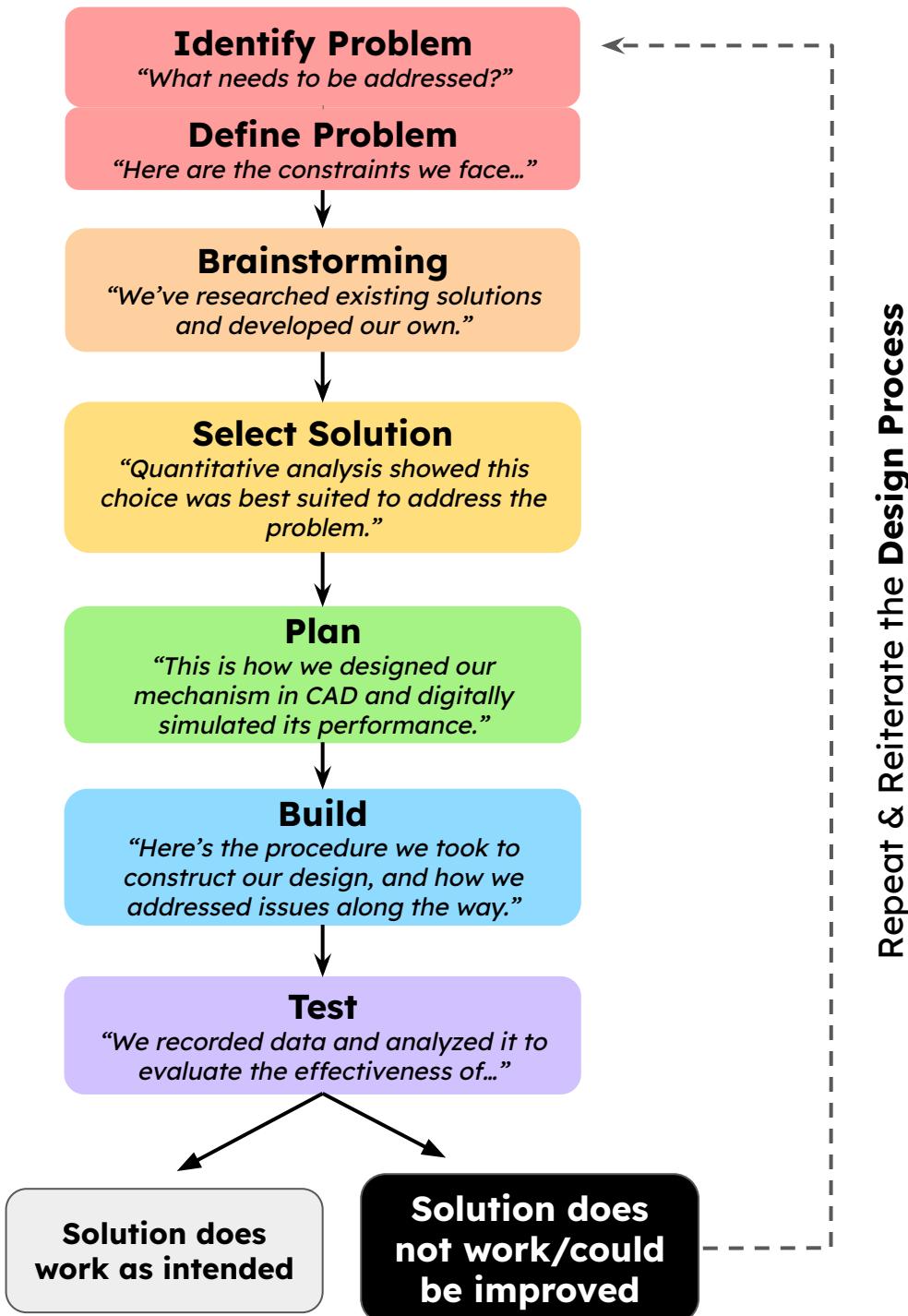
► Focus: To visualize our EDP.

Date: May 1, 2025

The following is a diagram of the **Engineering Design Process** that we will be using throughout the year, visualized with the colour coded categories for reasons discussed on the page beforehand.

While the system mostly goes the route of the arrows going down, at any time we will have to revert back stages or skip them depending on what is going on with our designs/subsystems.

Engineering Design Process Diagram



► Focus: To improve the readability of our logbook.

Date: May 1, 2025

This colour coding guide will help the further understanding of notebook organization and there are two main groups for the colour coding categories, the *Engineering Design Process* colours and *Miscellaneous* colours.

The Engineering Design Process colours are for robot design development entries that follow the baseline process used by most engineers, while the Miscellaneous colours are for development entries that do not fall into any of the 6 steps, but still add value and understanding to our notebook/process/organization.

1. Engineering Design Process Colours

Identify Problem

Entries that describe the problem or challenge we are facing and why it is an issue. Could be game/rule based or robot based.

Research/ Brainstorming

Entries that describe the discussion of design/subsystem options for the team and what they provide to the robot.

Select Solution

Entries that describe the final choices for robot design by matrices provided. This entry can always be brought up later.

Plan

Entries that describe the plan after choices have been made to implement the decided on designs/subsystems

Build

Entries that describe the building process, usually in steps or explanations.

Test

Entries that describe the data of testing of the subsystems to see if they worked as intended by the team in the brainstorming step.

2. Miscellaneous Colours

Strategy

Entries that describe the strategy of match play throughout the season; it can also describe changes of game strategy to rules.

Management

Entries that describe the team management, with part lists and organization in team member work.

Informational

Entries that are used for informational purposes that do not fall into any of the other categories.

Tournament Analysis

Entries that are used to document the outcome of tournament matches and skills attempts.

Skills

Entries that relate to planning and practicing for the skills challenge.

Coding

Entries that relate to our software development process and technical theory.

- ▶ Focus: To describe our design philosophy.

Date: May 1, 2025

Coming into this season as a 4th year team, we are following a design process that we've been continuously evolving upon since our first season and have found effective. Continuing from last year, we implement a highly collaborative and open approach to tackle our design process head-on:

- Identify design criteria/constraints to meet within a problem
- Research other unique solutions and implementations
- Use our solutions as the core foundation of our implementation whilst incorporating effective mechanisms from other designs
- Identify issues with other solutions to also improve ours

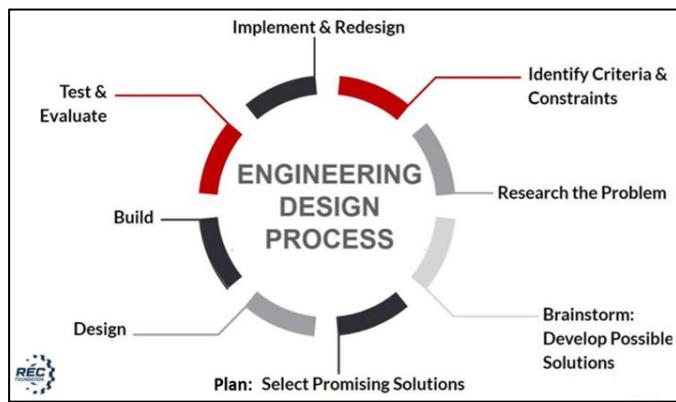


Figure 1: Loose inspiration for our personal EDP based on RECF resources:

<https://v5rc-kb.recf.org/hc/en-us/articles/9685581971095-VEX-V5-Robotics-Competition-Spectator-Primer-Part-1-The-Big-Picture-Structure-Season>

We are focused on **consistency and quality** (A worse-performing, but still operating subsystem is miles ahead of one that seems to be effective but functions horribly the majority of the time, which causes it to be inconsistent and could be match-affecting).

- **Initial Drivetrain:** The drivetrain is the robot's core, and it has to work flawlessly before other subsystems, code, and decorations can be added.
- **Robot Tuning/Debugging:** Last season, we were building our robot right up to Worlds. This did not leave enough time for adequate testing, leading to preventable problems with our elevation. This season, we want more time before tournaments where we do not construct any new subsystems.

- **Focus:** Detailing how this notebook will be formatted, including the design process tags.

Date: May 1, 2025

C-Channel Materials Comparison

Test

► **Focus:** Compare the pros and cons of different C-Channel materials for our robot chassis.

Date: May 14, 2025

Drivetrain Frame & Chassis:

There are two main types of metal available for C-Channels. We wanted to qualitatively weigh the features of each to decide the primary material for our chassis.

Material Type	Pros	Cons
Aluminum 	<ol style="list-style-type: none"> 1. Lorem ipsum dolor sit amet consectetur adipiscing elit. Blandit quis suspendisse aliquet nisi sodales consequat magna. 2. Sem placerat in id cursus mi pretium tellus. Finibus facilisis dapibus etiam interdum tortor ligula congue. Sed diam urna tempor pulvinar vivamus fringilla lacus. 3. Porta elementum a enim euismod quam justo lectus. 	<ol style="list-style-type: none"> 1. Nisl malesuada lacina integer nunc posuere ut hendrerit. Imperdiet mollis nullam volutpat porttitor ullamcorper rutrum gravida. 2. Ad litora torquent per conubia nostra inceptos himenaeos. Ornare sagittis vehicula praesent dui felis venenatis ultrices.
Steel 	<ol style="list-style-type: none"> 1. Dis parturient montes nasceruntur ridiculus mus donec rhoncus. 2. Potenti ultricies habitant morbi senectus netus suscipit auctor. 	<ol style="list-style-type: none"> 1. Maximus eget fermentum odio phasellus non purus est. Platea dictumst lorem ipsum dolor sit amet consectetur. 2. Dictum risus blandit quis suspendisse aliquet nisi sodales. Vitae pellentesque sem placerat in id cursus mi.

Figure 1: 35-hole long C-Channel (VEX KB)

Figure 2: Assorted steel parts (Purdue Sigbots)

Authored by: Michael

Witnessed by: Brandon

Page 17

This flag at the top shows which stage of the **Engineering Design Process** this page focuses on

The **problem** we're currently facing is identified at the top to remind readers and ourselves exactly what we are currently doing.

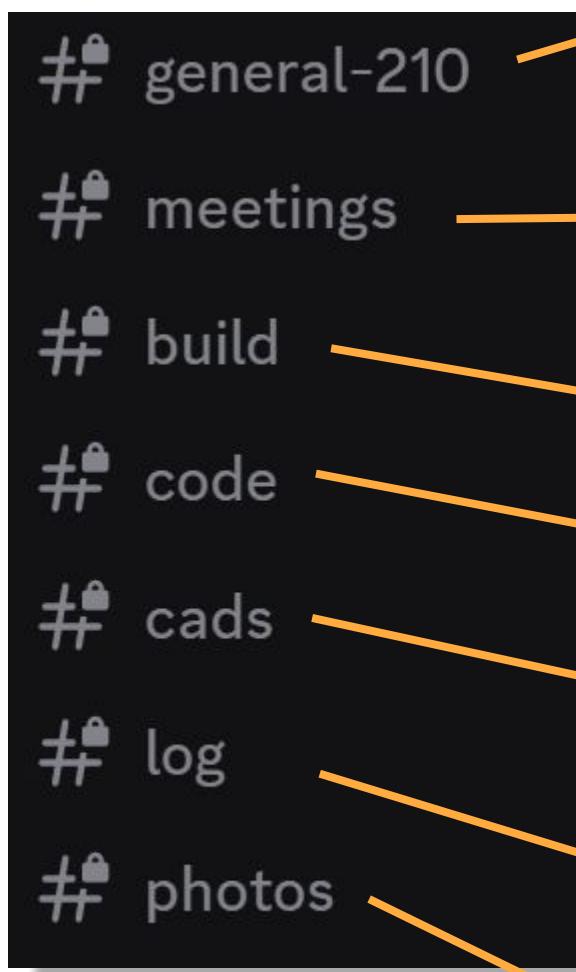
Subheading(s) categorize which **subsystem** we're addressing on this page

Signing off on pages ensures accountability and streamlines communication when members back-read logs

- ▶ Focus: Optimizing efficient communication

Date: May 2, 2025

Discord is one of our teams main **communication** when it is **difficult to meet up in meetings**. We aim to **stay organized** by using **various messaging channels** to ensure everything and **everyone** is aware where to look for a **specific issue/topic**.



General: Used for messaging timings to come in, setting up meetings, everything not robot related

Meetings: When meeting up, it tends to be difficult to remember everything, thus every meeting, we designate one member to put little notes in this channel

Build: Where our builders speak to build issues

Code: where we speak to code issues and possible auton routes

Cads: Where our Computer Aided Designs are put to ensure they can always be found

Log: Logbook notes/if we need to assign someone to a set log due to time issues

Photos: Where all the robot photos are stored so we can put them into logbook

► Focus: Meta information about our programming logbook.

Date: May 2, 2025

This logbook will primarily track the development of our robot's control software in accordance with our Engineering Design Process, featuring much of the mistakes, setbacks, and successes that we experience in small, separated chunks. Much of the technical explanation and mathematical theory behind our design flow will be omitted for the sake of length in this logbook and adhering to a chronological order.

- If you are looking at a physical/digital logbook right now, **you will have also received an additional copy of the programming logbook at the end of this engineering logbook.**

Project: Drive Curve cont. Informational

The basic gist of this system is that the drive curve allows us to **modify the sensitivity of the controls**, allowing us to adapt the robot to our driver, not the other way around. Since most drivers (like ours) often use controllers the most by playing video games, and video games allow players to change the sensitivity, the same should be possible in robotics, so that our driver can be as proficient with the controls as possible based on their muscle memory.

Standard Drive Mapping v.s. **Our Curve**

This equation relies on Euler's number, which is commonly used in other applications involving exponential growth. The exponential function e^x always grows at a rate (or derivative) of e^x . This can be changed with the t value, which determines how "aggressive" the curve is.

$x = \text{joystick analog input, between } 0 \text{ and } 127$

$t = \text{curve constant, set as variable "drive_curve_constant"}$ in the example code; this can be changed to your driver's liking

$$y = \left(e^{-\left(\frac{x}{10}\right)} + e^{\left(\frac{(abs(x)-127)}{10}\right)} \cdot \left(1 - e^{-\left(\frac{x}{10}\right)}\right) \right)_0$$

(powf(2.718, -(drive_curve_scale / 10)) + powf(2.718, (fabs(joy_stick_position) - 127) / 10) * (1 - powf(2.718, -(drive_curve_scale / 10)))) * joy_stick_position

Author: Michael Date: May 4, 2025 Page 10

Project: Comparing Graphics Libraries Select Solution

We've been looking into adding support for **autonomous selectors**, **debugging interfaces** and **telemetry visualizations** into our codebase. We envision these features to be graphically displayed on the V5 Screen, with the possibility for inputting actions using the touchscreen or the V5 controller.

In our research, we found **3** major graphics libraries that can be used to display text, pictures, etc on the brain:

Library	Pros	Cons
LLEMU (PROS)	<ul style="list-style-type: none"> Extremely simple to implement with minimal code required for basic text display Very lightweight with minimal processing overhead Perfect for simple debugging output and basic status displays during development 	<ul style="list-style-type: none"> No graphics capabilities beyond basic text characters Cannot display images, custom fonts, or any visual elements beyond text Appears outdated and unprofessional compared to modern interface expectations
LVGL (not vex-specific)	<ul style="list-style-type: none"> Professional-grade GUI framework with modern, polished visual appearance Comprehensive widget library Active community support with regular updates 	<ul style="list-style-type: none"> Large learning curve requiring familiarity with object-oriented concepts Higher memory usage that may impact performance May be overkill for simple status displays or basic debugging
pros::screen (PROS)	<ul style="list-style-type: none"> Direct pixel-level control allowing for completely custom graphics and layouts Flexibility to create unique interfaces tailored specifically to our needs 	<ul style="list-style-type: none"> Requires significant programming effort to build even basic GUI elements from scratch Time-intensive development process, especially for complex multi-view interfaces

Author: Michael Date: May 20, 2025 Page 35

- ▶ **Focus:** Reflect on our previous season's experiences.

Date: May 2, 2025

Throughout the 2024-25 season High Stakes, we have learned and experienced a lot as a team. Though we had vastly improved as a team, winning eleven awards and performing consistently at competitions, we were unable to qualify for the World Championships despite reaching the finals of our world championship qualifying tournament. This year, **five of our six members** will be returning to 210K. We feel that we will need to balance each members' skill set and improve our team efficiency to complete tasks.

Many important things we learned from last season include:

1. **Match Analysis:** Rewatching matches and **analyzing** them is incredibly beneficial to improve driving skill and strategy. This helps our team improve on strategies during the match as well as analyze different situations and how to respond accordingly.
2. **Having a plan:** During the 2024-25 season, we often lacked **time management** and **plans** to complete our goals. This year we look to improve our overall efficiency and time management, making use of the time we spend at our facility and not wasting it especially after losing a member of our team.
3. **Innovation and taking charge:** Throughout the 2024-25 season, we often followed popular designs and not pushing innovation on our own. Although we did innovate a “descore” mechanism, it was ultimately too late in the season and it had little impact on the overall community. This year, we look to push innovation and design different prototypes instead of following a general trend.

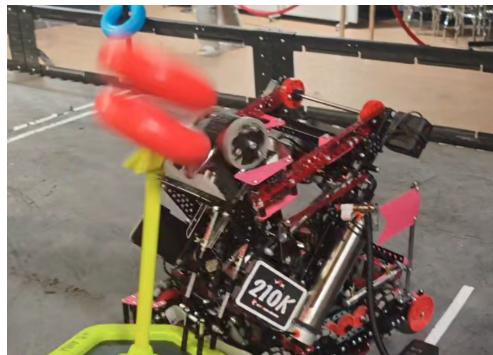


Figure 1: Our multi-ring descore mechanism from last season. This mechanism was developed after our provincial championship and never saw regional play.

Section A - α

Game Introduction

Push Back - Analysis

Identify Problem

► **Problem:** Analyze the game layout.

Date: May 10, 2025

Note: All game manual review in this section is based on the initial release of the game manual. Future changes in the game manual will be analyzed in a follow-up page. (v0.1).

Push Back: is a game with three main elements to interact with:

1. Tubes/Goals
 - a. In order to score points, we must place the main game element, a block, into a tube
2. Match loading tubes
3. Parking zone

Objectives

1. The main game element of push back are “blocks.” There are **88 blocks** around in a match of push back, 44 red and 44 blue. Blocks must be placed into different tubes to be counted for points. Each block is worth **3 points** when scored in a tube
2. There are **4 tubes** on the field
 - a. There are two **long tubes** that can each hold up to **15 blocks**. These long goals also have a **control zone** that can hold 3 blocks, having a majority of blocks in this zone will give a **control bonus**, yielding **10 points**.
 - b. There are **two central tubes** at the center of the field. These goals are on different elevations, posing a challenge to score on. The upper central goal can **hold 7 blocks**. Scoring majority on this goal yields **8 bonus points**. The lower goal can also hold 7 blocks, scoring majority on this goal will give **6 bonus points**.
3. There are **4 matchloading tubes** that can be used to introduce new blocks into the field. Each matchloading tube starts with and can hold up to **6 blocks**. These vertical tubes have an opening at the bottom allowing for access to blocks.
4. A parking zone can be used to park a robot at the end of the match. If there is a single robot in the parking zone, an extra **8 points** are awarded. If both robots are able to fit in the parking zone, an extra **30 points** are awarded.

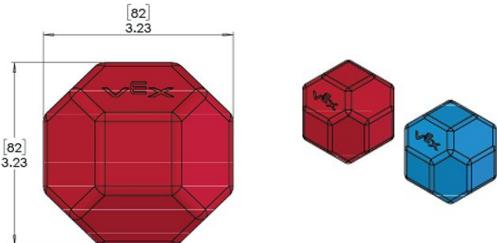


Figure 1: Block Specifications.
(VEX game manual)

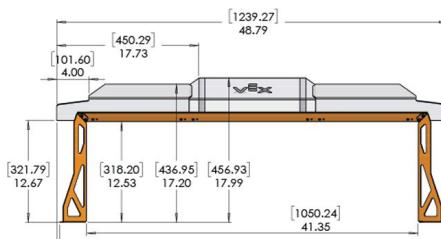


Figure 2: Side view of a long tube
(VEX, game manual)

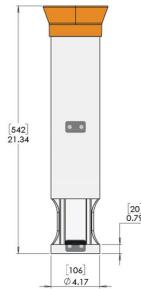


Figure 3: Matchloading tube
(VEX, game manual)

- **Problem:** Differentiate the different goal sizes

Date: May 10, 2025

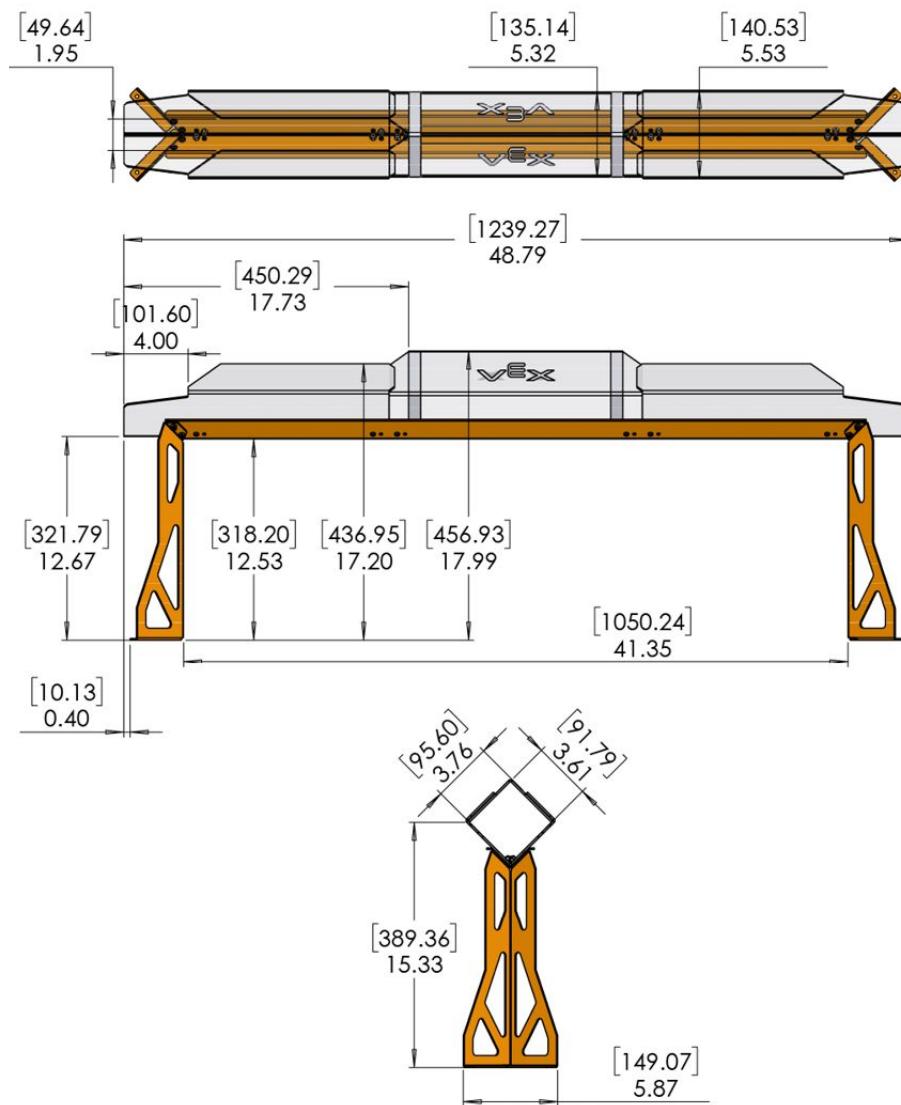
Goals:

A Field Element that is constructed out of plastic and metal components into which Blocks can be Scored.

- 3 Types of goals: Long Goal, and Center Goal (Upper and Lower)

Long Goals:

- Each Long Goal is 48.8" (1239mm) in length, with a 13.33" (339mm) enclosed center section.
- Scores up to 15 blocks, with a maximum of 55 points (including 10 point Control Bonus)
 - 2 Long Goals on a V5 Field

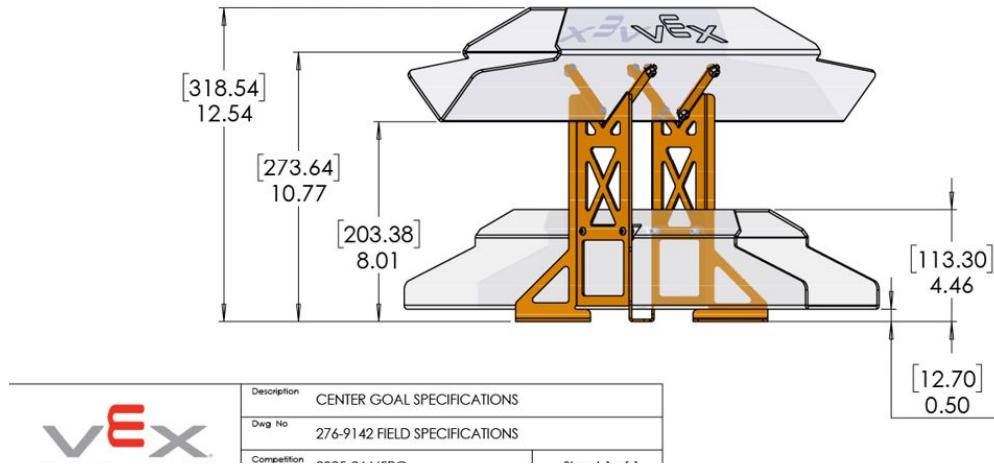
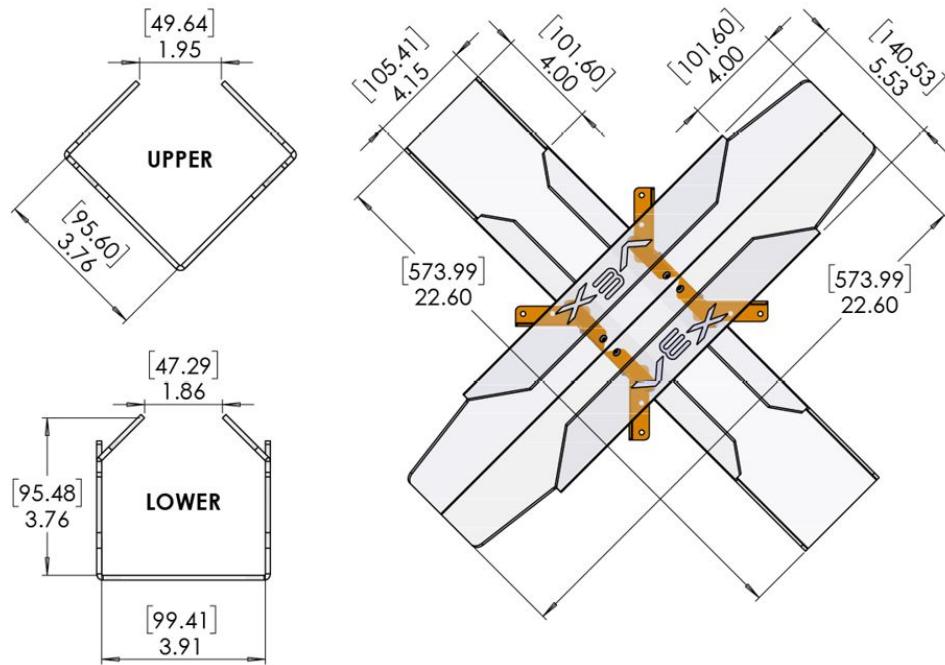


► Problem: Identify Mid Goal Sizes

Date: May 10, 2025

Center Goals (Lower & Upper):

- Each Center Goal is 22.6" (574mm) in length.
- In the middle of the Field
- Scores up to 7 blocks per goal, with a maximum of 56 points
 - Control Bonus: 8 Points for Upper and 6 Points for Lower



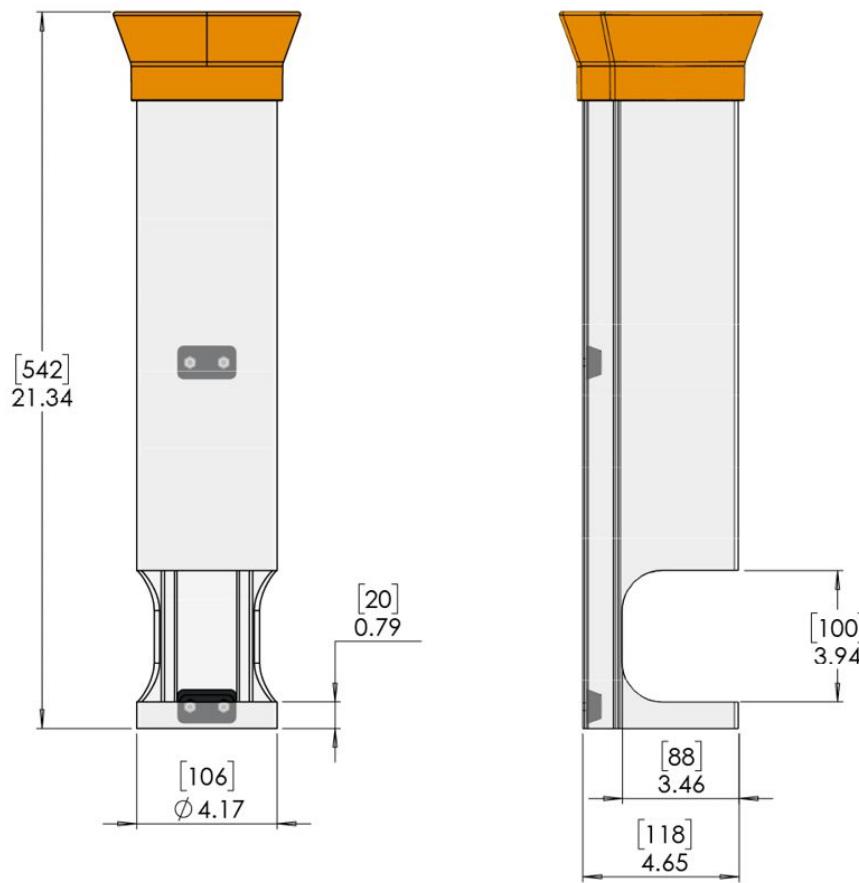
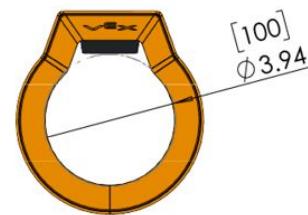
► Problem: Identifying Matchloader tube

Date: May 10, 2025

Loaders:

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, and Drive Team Members may add Match Load Blocks to Loaders during the Match.

- When the match starts there are three Blocks of your color on the bottom, and 3 of the opposing alliance's Blocks
- Has a 6 block capacity



Push Back - Analysis

Identify Problem

► Problem: Analyze the game layout.

Date: May 10, 2025

Each game will be played with two alliances, One “red”, one “blue”, both composed of **two teams** will compete in a 12ft x 12ft arena.

The center of the field will have two goals.

One upper goal and one lower goal.

The field will also have long goals on each side of the field. Each alliance’s driver station will have two matchloading tubes.

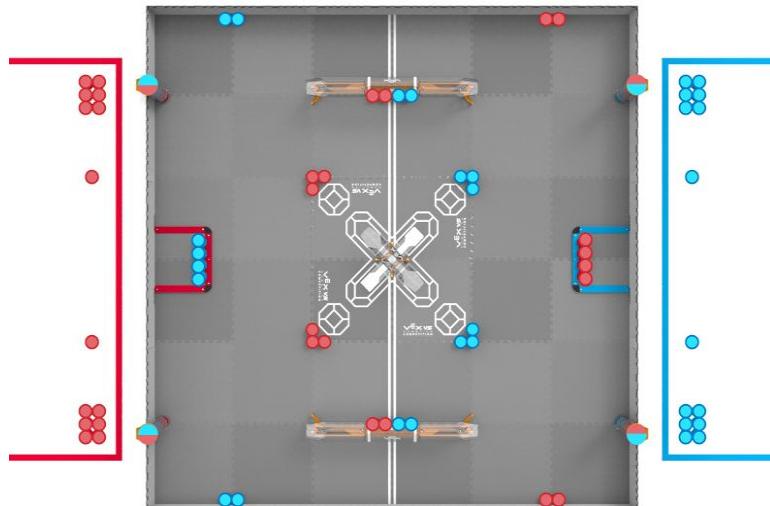


Figure 1: Field diagram, (VEX, game manual)

All robots must fit within a **18” x 18” x 18”** volume.

- Robots may expand up to **22” x 22” x 22”** in volume.

There is no possession limit, meaning a single robot can hold **unlimited** blocks.

Autonomous

- An **autonomous win point** is awarded to an alliance who can score 7 blocks of their colour into 3 different goals. They must also remove 3 blocks of their colour from matchloading tubes. Both robots must not contact the parking barrier.
 - Although difficult, these tasks could be completed by a single robot thus meaning **Solo AWP** is achievable
- **SG7** states that we may not interact with tiles, robots or scoring elements on the opposite side of the autonomous line. Meaning there may be a risk factor when creating different autonomous paths.
- The **autonomous bonus** is worth **10 points**. In the case of an autonomous tie, the points will be split.

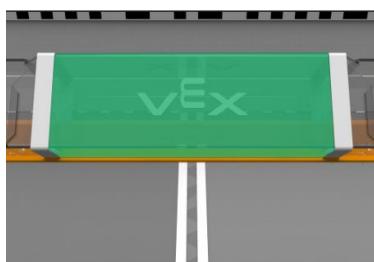


Figure 3: Control zone of a long goal (VEX, game manual)

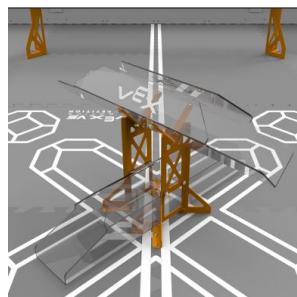


Figure 2: Center goals (VEX, game manual)



Figure 4: Parking barrier (VEX, game manual)

- ▶ Problem: Identify similarities with past games.

Date: May 10, 2025

Blocks

Taking a first look at the push back game elements we can determine that the **block** holds **properties** similar to a **ball**.

However, due to its varying shapes on different sides, its movement across the field is **unpredictable**. Some similar game elements include **VRC Over Under triballs** where the game element could roll, but due to its triangular shape it was very random and hard to control. Blocks **DO NOT** count as field elements



Figure 1: Diagram of a block (VEX Forums)



Figure 2: Diagram of Tribal (VEX Forums)

Match Loading Tubes

Matchloading tubes are similar to **VRC Change Up** scoring goals where the game object is accessible near the **bottom of the tube** when force is applied. Game objects can also be put into both tubes through a hole in the top of the tube.



Figure 3: Diagram of Change Up Goal (VEX Forums)



Figure 4: Diagram of Push Back Tube (VEX Forums)

Scoring Tubes

Scoring tubes are most similar to **VRC Sack Attack** scoring goals where both hold a similar **sharp edged shape**. Both tubes use **similar plastic material**, meaning they hold similar properties such as having **low friction**.

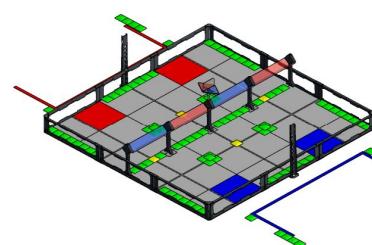


Figure 5: Diagram of VRC Sack Attack Field Layout (VEX Archives)

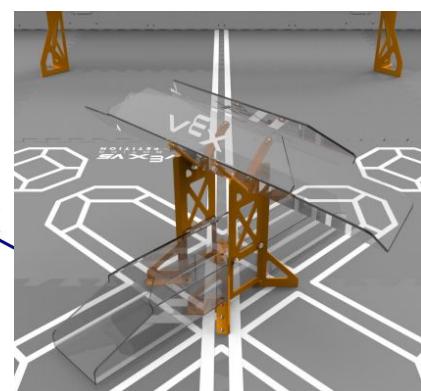


Figure 6: Diagram of Push Back Center Scoring Goals (Game Manual)

► Problem: Important Scoring Rules

Date: May 10, 2025

Scoring Rules:

<SC1>: All Scoring statuses are evaluated after the Match ends. Scores are calculated five (5) seconds after the Match ends, or once all Blocks, Field Elements, and Robots on the Field come to rest, whichever comes first.

Example: A Block which slowly falls out of a Goal at five (5) seconds would not be considered Scored.

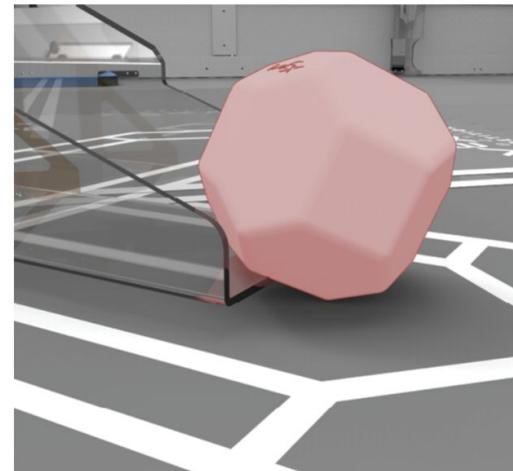


Figure SC2-1: This Block is touching the Floor, and would not be considered Scored.

<SC2>: A Block is considered Scored if it meets all of the following criteria:
The Block is in contact with the inside surface(s) of the plastic trough of a Goal.
The Block is not in contact with a Robot of the same color as that Block.
The Block is not in contact with the Floor.

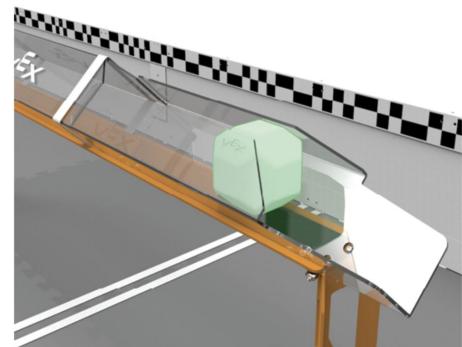


Figure SC2-2: This Block is contacting the inside surface of the Goal, and not touching a Robot of the same color. It is considered Scored.

<SC3>: A Control Zone is considered Controlled by an Alliance if a majority of the Blocks Scored in that Control Zone are the same color as the Alliance.

For Long Goals, a Scored Block is considered Scored in the Control Zone if it is entirely contained within that Control Zone.

A Block must be considered Scored in a Goal to also be considered Scored in a Control Zone.

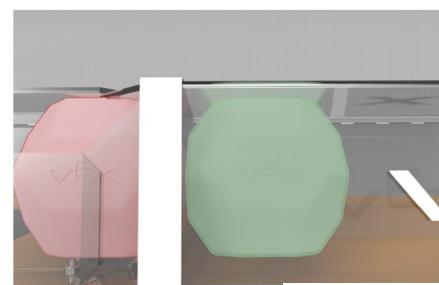


Figure SC3-2: The (green) Block on the right is fully within the boundaries of the Control Zone, and would be considered as Scored.

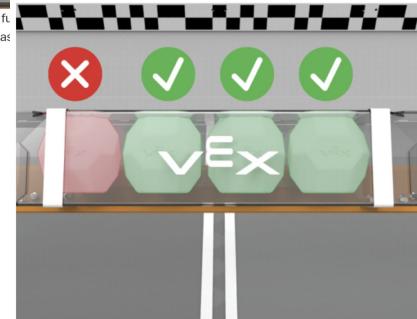


Figure SC3-1: The left-most (red) Block is not fully within the tape lines marking the boundaries of the Control Zone, and therefore would not be considered in determining which Alliance is in control of the Goal. The three other (green) Blocks are fully within the tape lines, and would be considered Scored.

Version 1.0: Mall of America Signature Event

Drivetrain

- ▶ Focus: Analyze material properties to optimize weight and structural integrity.

Date: May 10, 2025

Weight over the past few years have been a key concern in maximizing the potential of both the robot as well as the drivetrain. As such, we have implemented multiple ways to save weight including:

- **Minimizing** the amount of **C-channel** used
 - Switching to **angle bars** or **halfcut 3 wide** channels
- Maximizing the use of our **polycarbonate**
 - Polycarbonate is **very versatile** and can be used to save weight if implemented correctly
- Using **nylon and aluminum screws**
 - Although screws many not seem very heavy, the use of many **steel screws** throughout a robot will **stack up over time**
- Using **low strength axles** when high strength axles are not necessary

These methods we use allow us to aim for a overall robot weight around **11-12 lbs.**

This also lets us keep an ideal acceleration and torque. Anything thing to keep in mind is to keep a **low center of gravity (COG)**. This allows us to be untippable which is beneficial.

Additionally, having a **lot of weight** will lead to **quicker burnout** which will harm the longevity of our motors.

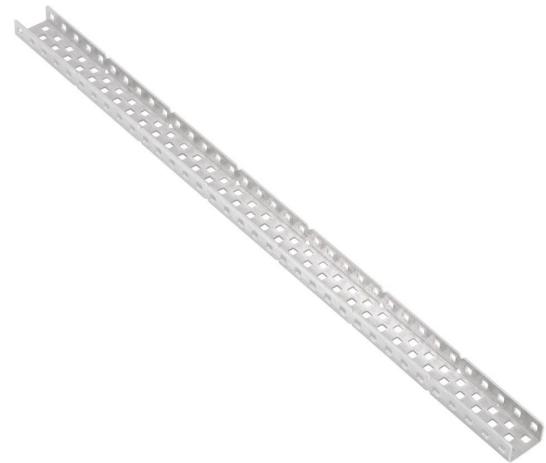


Figure 1: C-Channel

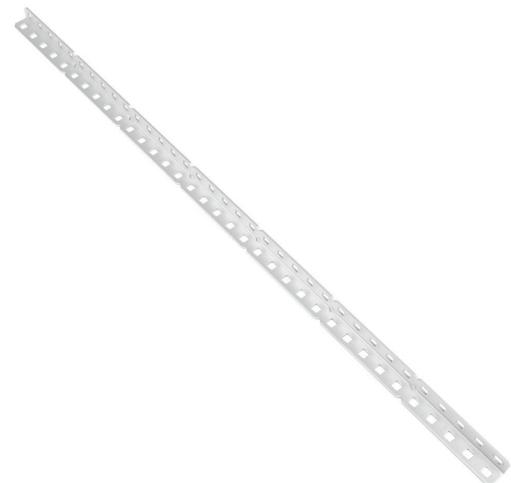


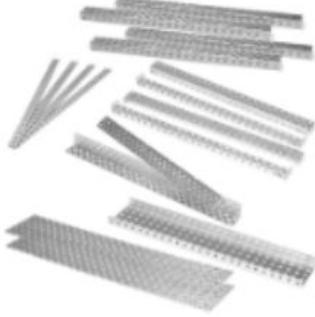
Figure 2: Angle Bar

- ▶ Focus: Compare the pros and cons of different c-channel materials for our robot chassis.

Date: May 10, 2025

Drivetrain Frame & Chassis

There are two main types of metal available for c-channels. We wanted to qualitatively weight the features of each to decide the primary material to use in our chassis.

<u>Drive Type</u>	<u>Pros</u>	<u>Cons</u>
Aluminum  <i>Figure 1: 35-long c-channel (VEX Knowledge Base)</i>	<ol style="list-style-type: none"> 1. Lightweight, lowers motor load and speeds up acceleration 2. Easy to cut, drill, and file with basic tools 3. Great for moving parts like arms and lifts to reduce the weight and stress on motors 4. Matte brushed finish, easy to spray paint 	<ol style="list-style-type: none"> 1. Less stiff than steel, can bend under hits or high torsion/shear stresses 2. Rotating axles can grind down holes if no bearings are used 3. Easier to crush/bend if over-tightened with standoffs or spacers
Steel  <i>Figure 2: Assorted steel parts (Purdue Sigbots)</i>	<ol style="list-style-type: none"> 1. High stiffness and strength for chassis and short spans 2. Holds threads better, fasteners stay tight 3. Less flex when bent and hit by other robots 	<ol style="list-style-type: none"> 1. Heavy, raises motor current draw, slows acceleration for lifts/arms and the chassis 2. Harder to cut and drill cleanly 3. Adds mass up high, raising the robot's centre of gravity

As drivetrain acceleration and the accuracy of our cuts/drilling for our robot is critical to its operation, we've elected to use aluminum as the material of choice for the majority of our robot, except for parts that undergo high-stress.

- Focus: Provide details on legal pneumatics used within VRC.

Date: May 10, 2025

In V5RC, middle and high school teams are allowed **two 100 PSI air tanks** and **unlimited VEX pistons**. With a cap of **88W** across all motors typically used heavily in drivetrains (often 6-7 motors even upwards to 8 motors). Pneumatics offer a great way to **power mechanisms** without sacrificing motor count.

Pistons are fast, strong, and versatile, making them ideal for tasks like actuating claws, lifts, or transmission (more on this later). They come in three stroke lengths: **small (25mm), medium (50mm), and long (75mm)**, each suited for **different applications** based on the required range and force.

Small (25mm)	Medium (50mm)	Long (75mm)
<ul style="list-style-type: none"> → Uses least amount of air → Best used for small range of motions → Good for clamps (such as ones used in High Stakes, Tipping Point, etc) 	<ul style="list-style-type: none"> → Most versatile piston length → Commonly used in transmissions and used when a range of motion between short and long is required 	<ul style="list-style-type: none"> → Longest stroke making idea for hangs as well as large ranges of motions → Commonly used for difficult to mount mechanisms due to its increased stroke length

Figure 1: Image from VEX store of a pneumatic kit containing one air tank, and one of each piston length along with other parts.



Pistons will inevitably be a **vital part** towards this game. While at this current moment, I am unable to find uses, over the course of this season, **designs** from ourselves and other teams will come out, which we may use to **optimize** various aspects within the game.

- ▶ Focus: Overview and explanation of power take-off

Date: May 10, 2025

A **Power Take-Off (PTO)** is a mechanical system that allows a robot to **transfer power** from **one function to another** using the **same motor(s)**. This is helpful when you want to use your limited number of motors for multiple purposes without adding extra ones.

For example, in the **2024–2025 game High Stakes**, many robots needed to lift themselves onto a ladder to score. While this could be done with a **single motor**, it was often too **slow or unreliable**. To solve this, teams, including ours, used a PTO system to temporarily shift power from the drivetrain to a lift mechanism. This allowed us to use **multiple motors** to **climb quickly and consistently**, without **sacrificing drivetrain performance** during the match.

Although **Push Back** doesn't have a **hanging challenge**, a PTO can still be **extremely useful**. One idea is to use a PTO to control a **ball indexing system**. This would allow your robot to **store balls without scoring them immediately**, while still being able to intake new balls, giving you more control over when and how you score.

A PTO typically works by **sliding a gear** from one **gear train** (or gear path) to another. Here's a simple breakdown:

A **powered axle** (connected to motors) is set up with a **gear** that can **move back and forth**. Using a **pneumatic piston**, the gear is pushed to **engage** with a **different gear** on **another mechanism** such as a lift or indexer.

When switched, the **same motor(s)** now power a **different part** of the **robot**, effectively “**taking off**” the power from the original system and “**giving**” it to a new one.

Some PTOs also **change gear ratios** in the process, depending on the needs of the **mechanism being powered**.

This makes PTOs great for saving motors and maximizing functionality in your design.

Drivetrain Design Criteria

Identify Problem

► **Problem:** Identify design targets for our drivetrain to meet.

Date: May 11, 2025

Drivetrain Criteria	
Mobility	<ul style="list-style-type: none">- Enables the robot to move in different directions- Includes lateral/angular motion and strafing
Versatility	<ul style="list-style-type: none">- Ability to maneuver in and out of tight spaces- Different drivetrain setups, such as tank drive, holonomic (omnidirectional) drive, or mecanum drive, offer varying levels of manoeuvrability and are suited to different driver strategies.
Robustness	<ul style="list-style-type: none">- The drivetrain serves as the robot's base, so the weight of everything will be distributed on top of the drivetrain. Proper weight distribution is crucial for maintaining stability and preventing tipping during rapid movements or interacting with game elements.

Goals	Constraints
<ol style="list-style-type: none">1. Withstand and support the weight of other subsystems on the robot2. This drivetrain will be able to maintain an even balance between torque and speed.<ul style="list-style-type: none">- This is so our drivetrain can generate enough power to push other robots, but also maintain enough speed to move around the field at a good pace.	<ol style="list-style-type: none">1. It must fit within the 18-inch cubed limit, though expansion may occur in the game<ul style="list-style-type: none">- In order to fit other subsystems and external components, the drivetrain will have to be constrained further than this2. The total power drawn from the motors in the drivetrain cannot exceed 88W, as per the rules<ul style="list-style-type: none">- Factoring in the intake and elevation mechanism, the DT must have ≤ 6 motors

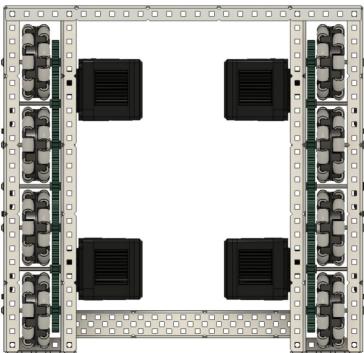
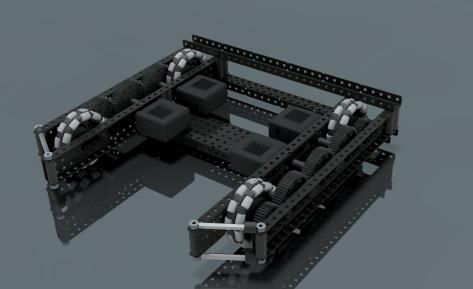
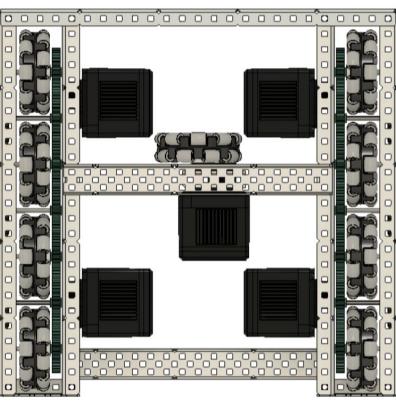
Drivetrain Brainstorming

Brainstorming

- ▶ Focus: Weigh the traits of each popular drivetrain type.

Date: May 11, 2025

A great resource for finding information and pictures of different drivetrains that we used is the Purdue Sigbots wiki:
<https://wiki.purduesigbots.com/hardware/vex-drivetrains/>

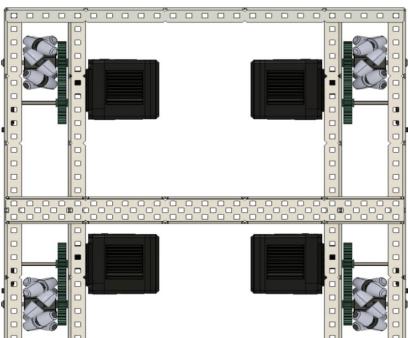
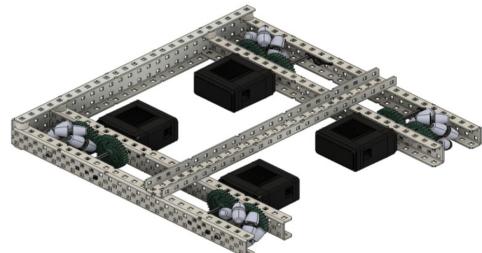
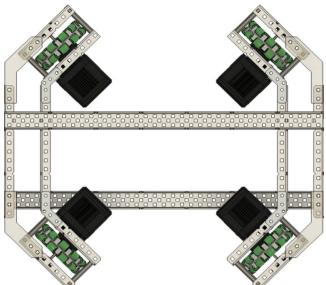
Drive Type	Pros	Cons
Standard Tank Drive  <i>Figure 1: Standard tank drivetrain(Evan, JHAWK)</i>  <i>Figure 2: Standard tank drivetrain (Xenon27, VEX Forums)</i>	<ol style="list-style-type: none">1. Easy to build and saves time for building subsystems2. Simple to program3. Supports lots of different gear ratios that can be added4. Lots of torque in the forward and backward axis, but also can move fast5. The front and back can be open, leaving space for intakes and other mechanisms6. Minimum 4 wheels	<ol style="list-style-type: none">1. Easily pushed from the side unless traction wheels are installed2. Less agile / takes longer to turn3. Limited mobility4. Friction issues are evident and hard to quickly resolve
H-Drive  <i>Figure 3: H-Drive (Evan, JHAWK)</i>	<ol style="list-style-type: none">1. Only slightly harder to build than tank drive2. Accessible space in the front and back3. Can strafe under power horizontally<ul style="list-style-type: none">- Slower than lateral movement	<ol style="list-style-type: none">1. Requires an extra motor for the middle wheel (minimum 5 wheels)2. Not many benefits over tank drive if center wheel isn't motorized3. Middle wheel could stop functioning from overheating due to being powered by a single motor

Drivetrain Brainstorming

Brainstorming

- ▶ Focus: Weigh the traits of each popular drivetrain type.

Date: May 11, 2025

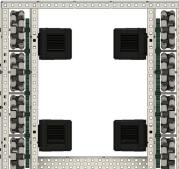
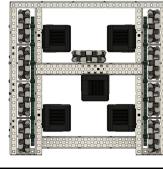
Drive Type	Pros	Cons
Mecanum Drive  <i>Figure 1: Mecanum Drive (image from Evan, JHAWK)</i>  <i>Figure 2: Mecanum Drive (image from Evan, JHAWK)</i>	<ol style="list-style-type: none">1. Same traction as a tank drive2. Difficult to push from the side3. Able to strafe on command4. Able to easily align itself with the field perimeter	<ol style="list-style-type: none">1. Much more complex to program2. Slower than a tank drive3. Strafing is not efficient4. Limited set of rollers may make it hard to build a reliable autonomous routine by pulling motor position values
X-Drive (Holonomic Drive)  <i>Figure 3: X-Drive (images from Evan, JHAWK) H-Drive (Evan, JHAWK)</i>	<ol style="list-style-type: none">1. Agile and can move in multiple axes at a time2. Always travels at the same speed when moving in any direction3. Faster than tank drive because it is holonomic	<ol style="list-style-type: none">1. Harder to attach a gear train to2. More complex to program3. Lacks torque because only 2 wheels are moving it in a given direction at any time4. Lacks space in the front and back for intake/rollers → intake must be placed at a higher angle to compensate

- Focus: Compare every drivetrain type via quantitative analysis.

Date: May 11, 2025

Decision Matrix Ranking Criteria (higher score = better). We decided to rank the following criteria to choose for our drivetrain type.

- **Speed** (1 - 5) - The maximum speed of the drivetrain configuration
- **Turning** (1 - 5) - The ease of turning with the drivetrain configuration.
- **Force** (1 - 5) - This criteria ranks the drivetrain on how much force it can generate
- **Compactness** (1 - 5) - This criteria ranks the drivetrain on how compact it can be
- **Maneuverability** (1 - 5) - The ability of the drivetrain configuration to move in multiple directions
- **Ease of Programming** (1 - 5) - Ranking on how easy the drivetrain configuration is to program
- **Ease of Driving** (1 - 5) - Ranking on how easy the drivetrain is to drive

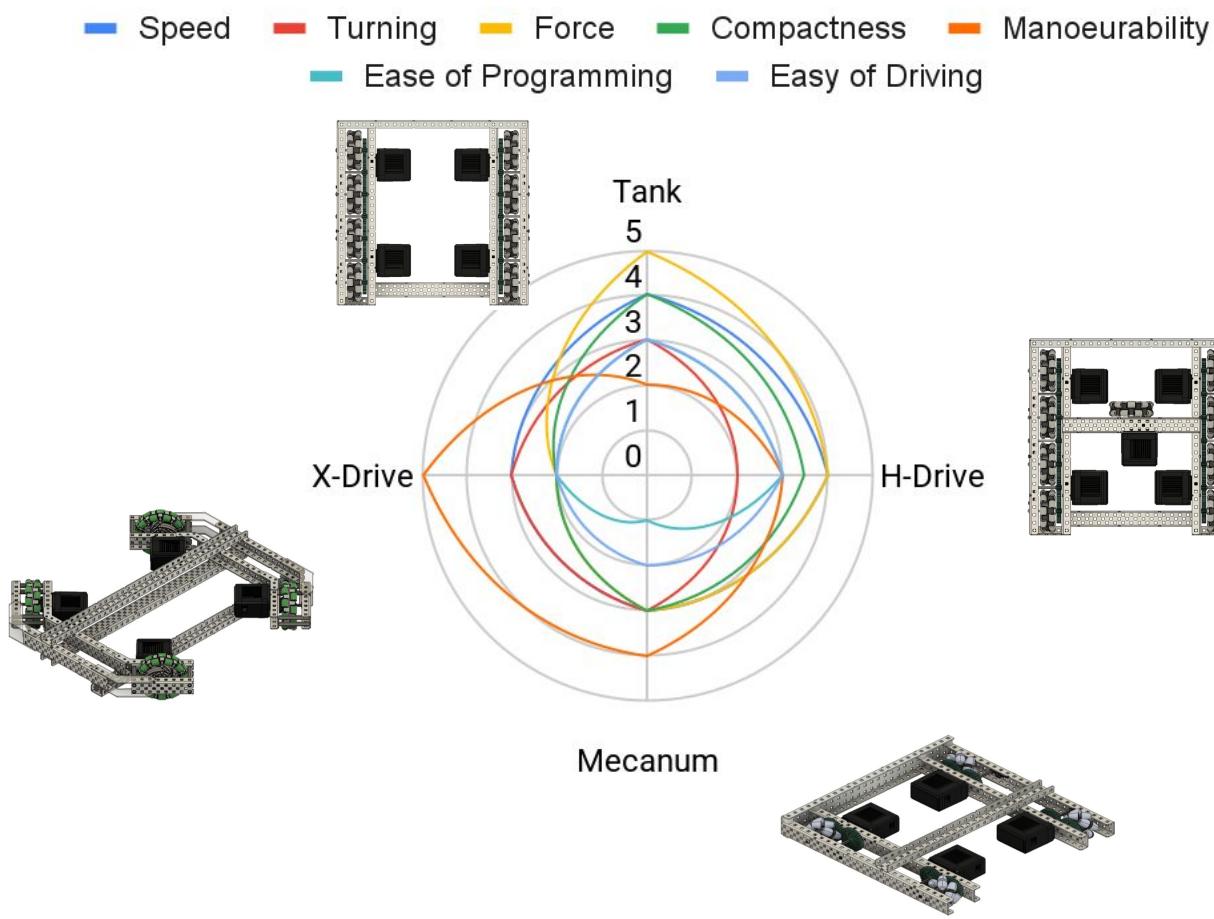
<u>Criteria Options</u>	<i>Speed</i>	<i>Turning</i>	<i>Force</i>	<i>Compactness</i>	<i>Manoeuvrability</i>	<i>Programming</i>	<i>Driving</i>	<i>Total</i>
Tank 	4	3	5	4	2	5	5	28
H-Drive 	4	2	4	3	3	5	5	26
Mecanum 	3	3	3	3	4	1	2	19
X-Drive 	3	3	2	2	5	2	2	19

- Focus: Visualize our drivetrain decision matrix.

Date: May 11, 2025

Below is a **radar chart** showing our design considerations, and how each drivetrain scores in these criteria. All design considerations will be scored in the same way — by evaluating them on a series of different criterion, and picking the design with the highest total, or the **largest area** on the graph. The shape the design makes can also affect our decision; no design should spike heavily toward a single trait, but be more **rounded**.

V1 Drivetrain Design Considerations



Based on all of our research, we've decided to build a **tank drive** going forward because we have previous experience with building one. Additionally, we also see manoeuvrability as less of an asset this season since we think the ability to contest the mobile goals, contests corners, and possibly hang will likely determine the outcome of high-level matches.

- **Problem:** How should our drivetrain respond to this year's challenges?

Date: May 11, 2025

Overview

One of the key problems when creating a drivetrain is coming up with a balance between the necessary torque and speed. In order to do this, we need to consider the multiple gear ratios with different wheel sizes and gear cartridges. All of these variables will lead to unique results.

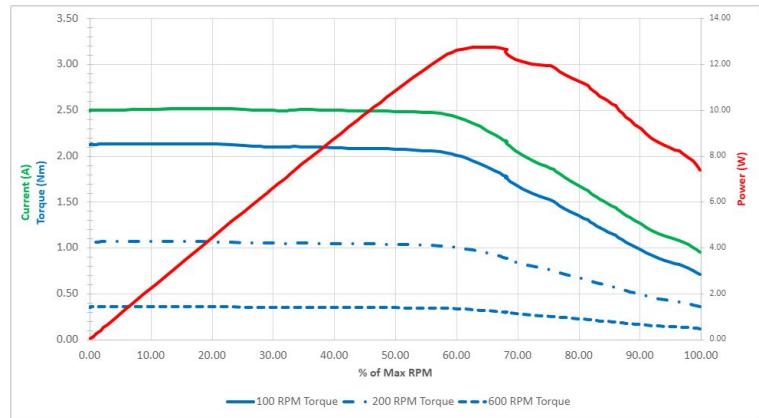


Figure 1: A graph showing the correlation between motor RPM and resulting power and torque. Each colour indicates a cartridge type (Red = 100rpm) (VEX official website)

Torque

Torque at its very base is a rotational force that determines how much power we have in order to push other robots around. There is a correlation between torque and speed defined by equation power = torque x speed. When speed increases, torque decreases.

Speed

How fast our drivetrain is able to move is a key problem when designing other components of our robot and programming autonomous routines.

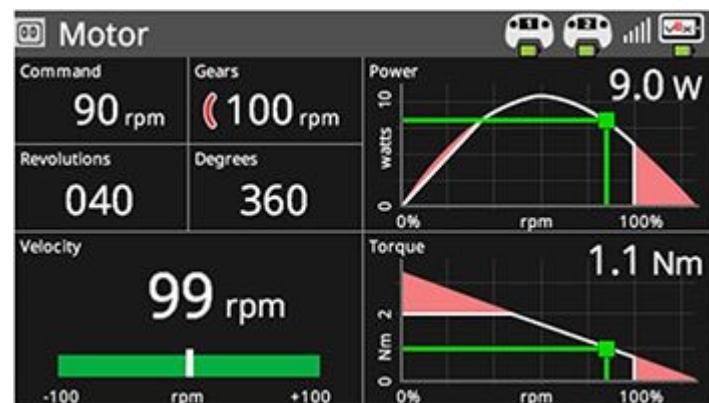


Figure 2: VEX brain display showing motor telemetry stats

- ▶ Focus: Drivetrain gear ratio design and considerations.

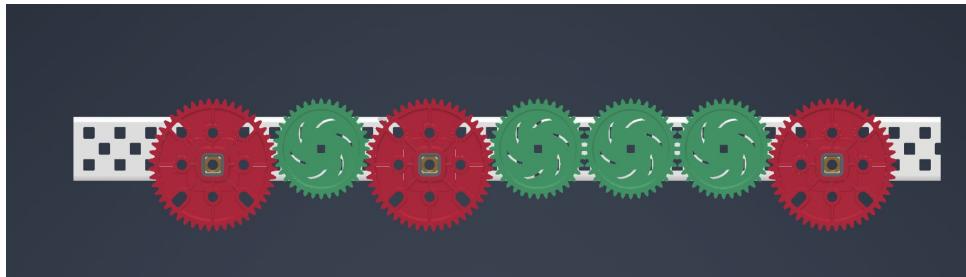
Date: May 12, 2025

Common Drivetrain Ratios

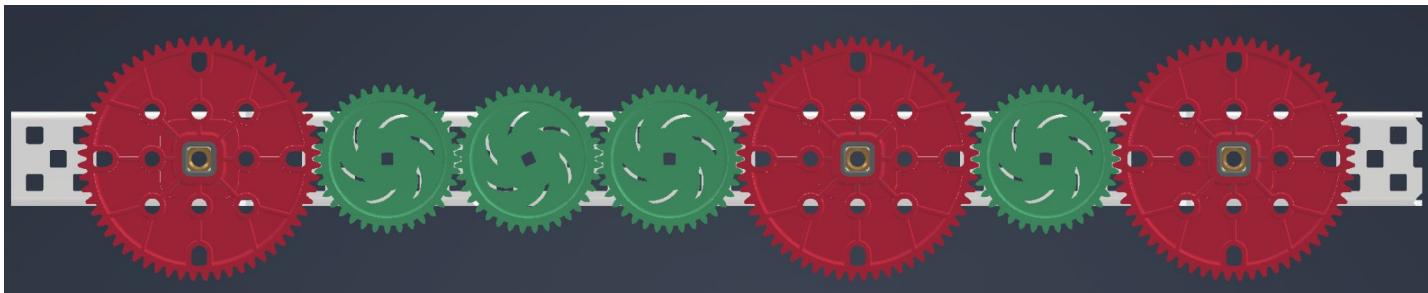
Looking through past games and signature event competitions, there has been multiple common drivetrain ratios that were used at these events.

Some of the common ratios include

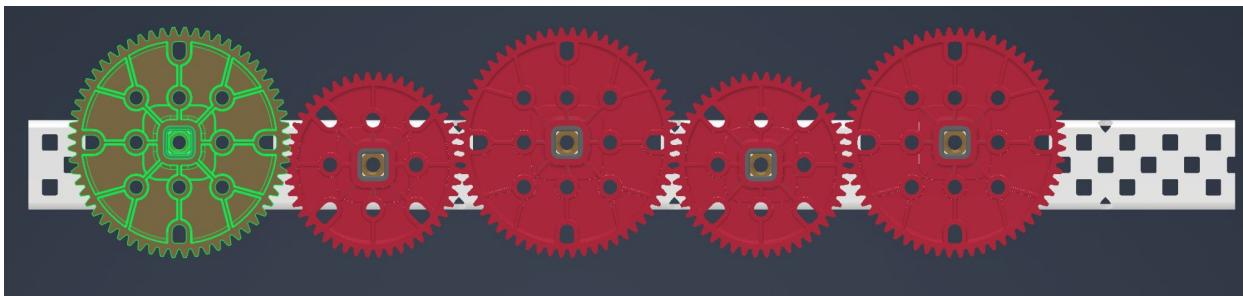
1. 3:4 Ratio, 450 RPM on 3.25" wheels
 - a. 3:4 Ratio, 450 RPM on 2.75" wheels



2. 3:5, 360 RPM on 3.25" wheels



3. 3:2, 400 RPM on 3.25" wheels
4. 4:5, 480 RPM on 3.25" wheels



5. 1:1.75, 343 RPM on 4" wheels

All of these options have proven to be able to compete at the highest level of competition. Each drivetrain will have different benefits and drawbacks that need to be considered before settling on a design.

Drivetrain Gear Ratio

Select Solution

- Focus: Quantitative comparison of gear ratio stats.

Date: May 12, 2025

When selecting a drivetrain speed, **several factors** must be considered, including, but not limited to, build **simplicity**, desired **speed** and **torque**, available **space** for other subsystems and autonomous tracking systems, and driver **preference** or feedback. These elements are key inputs in our **decision matrix** and play a crucial role in guiding our final drivetrain configuration. Each section will be rated on a scale of 1-5 with a higher score being better.

Table of decision matrix for given RPM on different wheel diameters.

<u>Criteria Options</u>	Speed	Torque	Ease of building	Space for subsystems	Driver Preference	Total
343 RPM 4.00" Wheel	4	5	4	3	2	18
360 RPM 3.25" Wheel	2	4	5	3	3	17
450 RPM 2.75" Wheel	3	4	4	4	4	19
450 RPM 3.25" Wheel	5	3	3	4	5	20
480 RPM 2.75" Wheel	4	3	4	2	3	16
480 RPM 3.25" Wheel	5	2	4	2	3	16
600 RPM 2.75" Wheel	5	1	3	2	2	13

While this matrix is being used to judge the **current drivetrain ideal** for us, the game may shift around with each drive having their own benefits outside of this matrix. For example the 343 RPM drivetrain is higher up making it easier to **cross barriers**; however it also makes our center of gravity higher up since it requires a larger drivetrain. With all this information in mind, we will likely start the season off of **450 RPM 3.25" wheels or 480 RPM 2.75" wheels** as we see more benefits towards those drivetrains speed to torque ratios at this current moment in time.

- ▶ Focus: Visualize our gear ratio decision matrix.

Date: May 12, 2025

Following the table on the previous page, we created a radar graph to visualize each drivetrain RPM using some of the criteria.

Speed (1 - 5) - Ranking based on how fast the max speed of the drivetrain is

Torque (1 - 5) - Ranking based on how much torque the drivetrain is able to generate

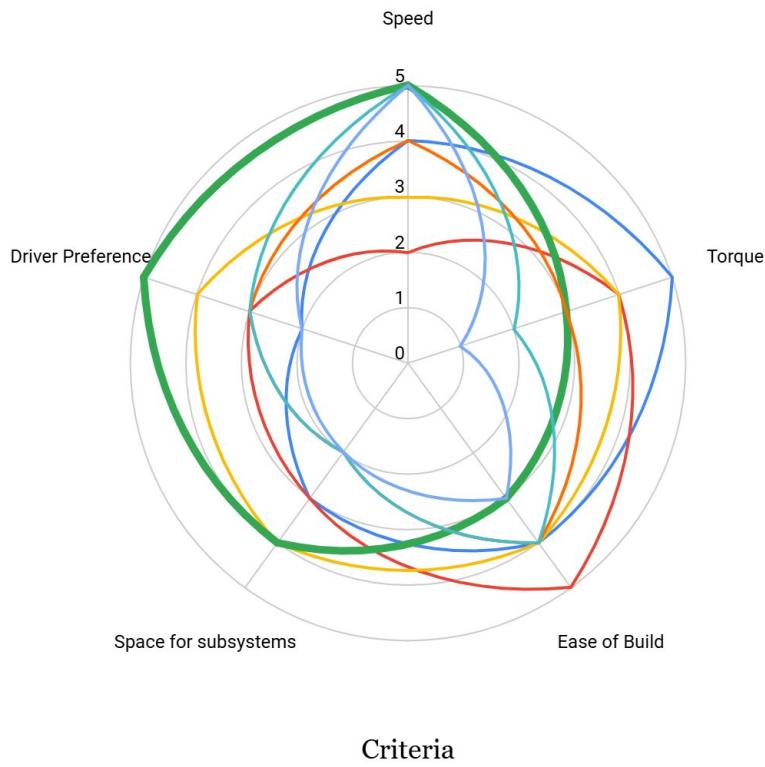
Ease of Build (1 - 5) - Ranking based on how easy the drivetrain is able to be built

Space for Subsystems (1 - 5) - Ranking based on how much space there will be for subsystems later in the robot

Drive Preference (1 - 5) - How confident our driver is able to drive this RPM

V1 Tank Drivetrain Ratio Considerations

- 343 RPM 4.00" Wheel
- 360 RPM 3.25" Wheel
- 450 RPM 2.75" Wheel
- 450 RPM 3.25" Wheel
- 480 RPM 2.75" Wheel
- 480 RPM 3.25" Wheel
- 600 RPM 2.75" Wheel



Based on our rankings, we have decided to create a 450 RPM on 3.25" wheel drivetrain as it has the most points combined from all the categories. Although three of our suggested drive RPM had very high and similar scoring, we still decided to create a 450 RPM as it was the last drivetrain we built in VRC High Stakes, so we are very familiar with building it.

- ▶ Focus: Determine the effectiveness of ball bearing drivetrain wheels.

Date: May 12, 2025

Ball Bearings

Ball bearings are used in robotics due to their **low friction**, making them ideal for smooth, efficient **rotation**. In previous seasons, we implemented custom wheels by **drilling out** our omni and traction wheels to friction-fit ball bearings. This approach allowed us to support **heavier robot builds** without the added weight significantly improving drivetrain friction.

Drawbacks

Despite these benefits, we've found this method to be **excessive**. The custom bearing wheels are both **bulky and heavy**, and the small performance gains in reduced friction do not outweigh the **trade-offs** in the foreseeable future. Specifically, the decrease in acceleration caused by the added weight offsets the smoothness benefits.

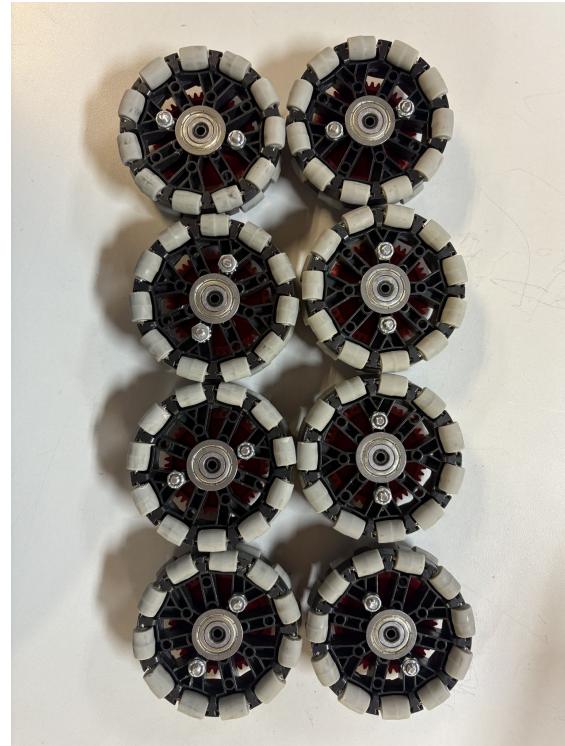


Figure 1: 3.25" ball bearing wheels and gears we created

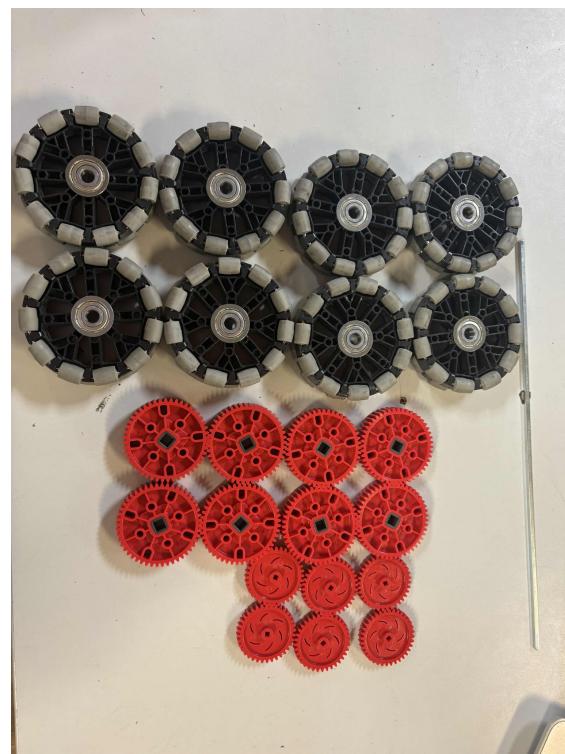


Figure 2: Materials put together for a drivetrain

► Focus: Choosing whether or not to use Ball Bearing Wheels

Date: May 12, 2025

Minimal Benefits

From our experience, building a **lighter, more efficient** robot from the start offers **greater overall performance**. Unless drivetrain friction reaches a point where it's **critically impairing function**, which is rare, we find it more practical to rebuild or adjust the drivetrain rather than rely on ball bearings. This approach also **minimizes future issues**, especially during high-stakes environments like competitions or extended practice sessions.

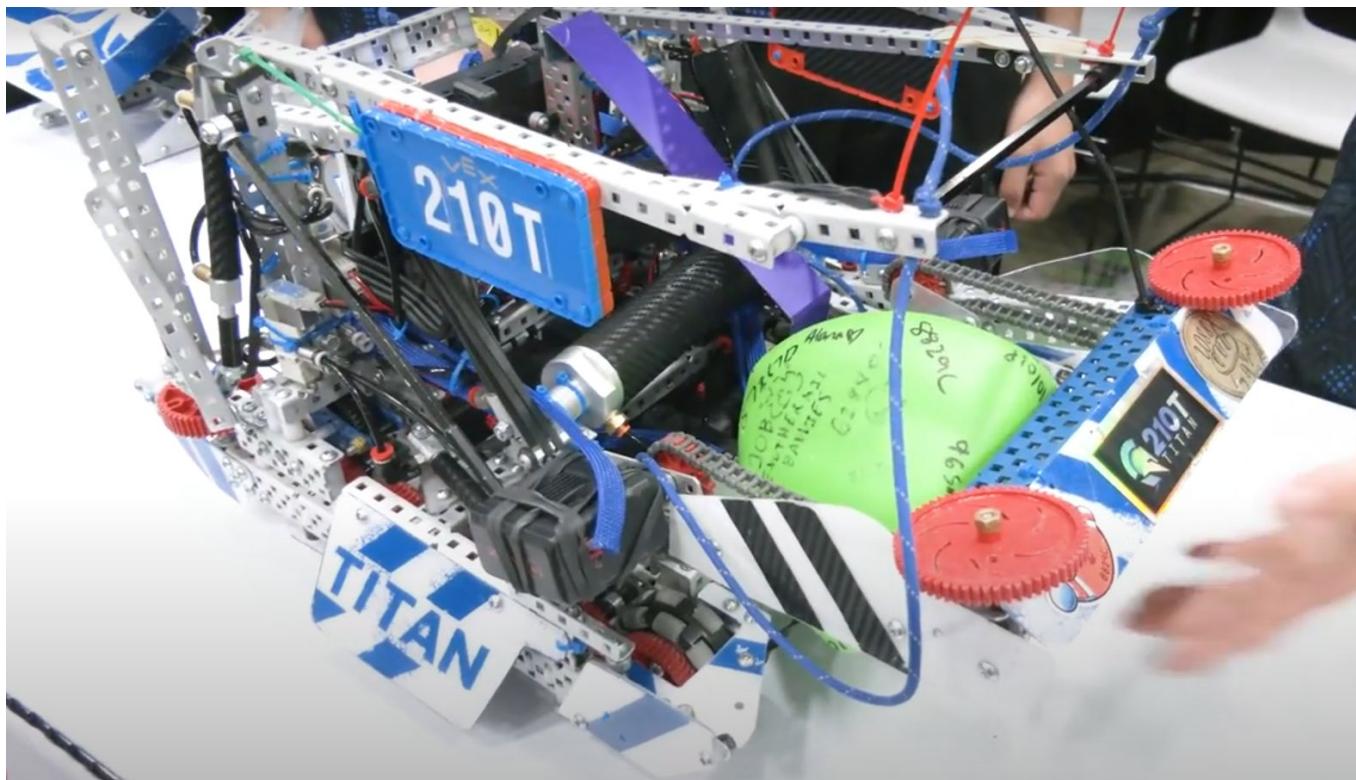


Figure 1: 210T Utilizing a ball bearing drivetrain due to their heavy size during Over Under (FUN Interview)

- ▶ Focus: Planning out our final drivetrain

Date: May 12, 2025

Drivetrain Planning

As a team, we have decided to run a drivetrain that runs on **450 RPM on eight 3.25" wheels**. We have decided to use eight wheels to have an **even distribution of weight**. This drivetrain offers a **good balance** between **speed and torque** for the game. This drivetrain is also something our driver is comfortable with driving. Finally, the last benefit is the **lower center of gravity** we can obtain using this drive. Choosing this design allows us to plan earlier for future mechanisms, such as our intake and other important aspects of our robot.

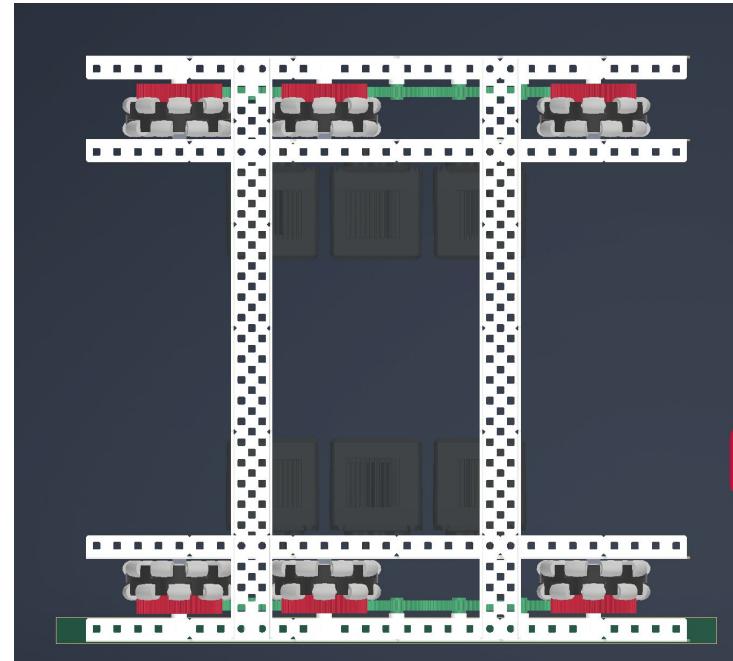


Figure 1: CAD of a 450 RPM on six 3.25" wheels

CAD

In order to visualize our drivetrain design, we have utilized computer aided design to give us a general idea of the size of our drivetrain as well as motor, gear and wheel placements. This allows us to quickly prepare materials needed before we start building.

- ▶ Focus: Prepare materials for building chassis

Date: May 15, 2025

Members Involved

Bryan, Joshua, Daniel

Objective

Today's objective is to follow our CAD to build our 450 RPM on 3.25" drivetrain into real life. In order to have a consistent drivebase, we must make sure that no warping or bending occurs during our building process

Materials for Drivetrain Frame:

- 4, 29 1x2x1 Long C-Channels
- 1, 27 1x2x1 Long C-Channel
- 1, 27 1x3x1 Long C-Channel
- 6 Bearing Flats
- 20 $\frac{1}{2}$ " Spacers
- 4 $\frac{3}{8}$ " Spacers
- 8 $\frac{1}{8}$ " Spacers
- 4 $\frac{7}{8}$ " Spacers
- 18 Nylocks
- 8 0.85" Screws
- 4 1.25" Screws
- 4 1.75" Screws

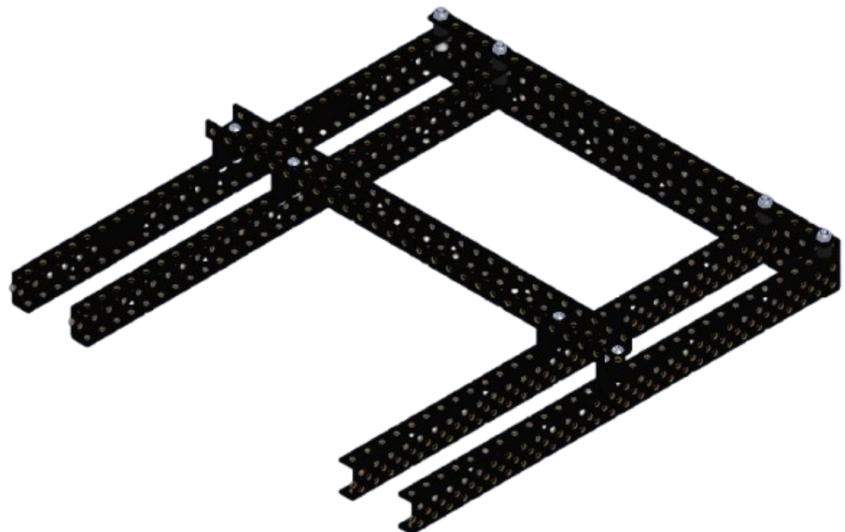


Figure 1: Our completed drivetrain CAD

- Focus: Layout procedure for building chassis

Date: May 15, 2025

Steps to Building Drivetrain Frame:

1



X1 29 Hole 1x2x1 C-Channel



X3 Bearing Flats



2

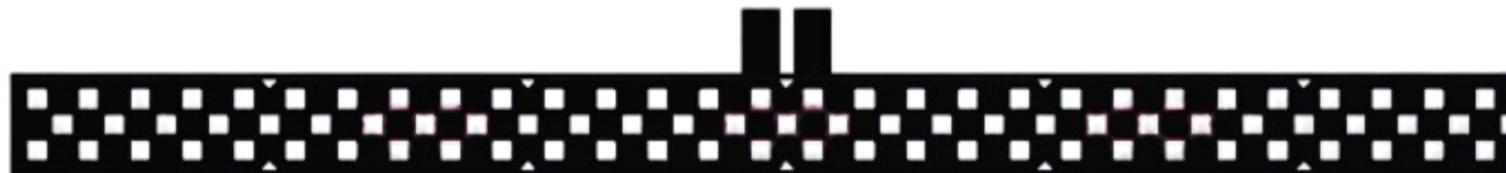
— **X2** $\frac{1}{8}$ " Spacer

█ **X2** $\frac{1}{2}$ " Spacer



3

Mirror Steps 1 & 2 on another C-Channel



*Note Colours of each part do not matter, only purpose is to highlight parts to be more obvious and clear what is being added

- ▶ Focus: Layout procedure for building chassis

Date: May 15, 2025

Steps to Building Drivetrain Frame:

4

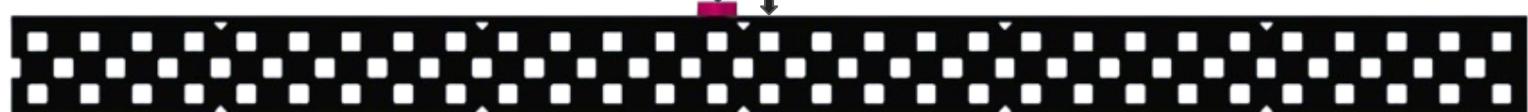


X1 29 Hole 1x2x1 C-Channel

— **X2 $\frac{1}{8}$ " Spacer**

■ **X2 $\frac{1}{2}$ " Spacer**

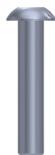
Mirror This
C-Channel
once!



5



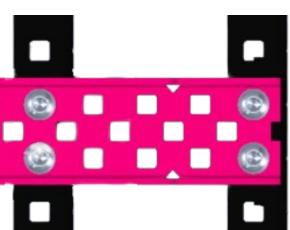
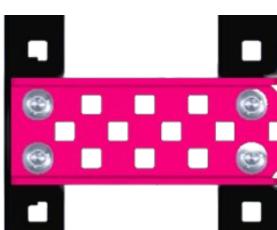
X1 27 Hole 1x2x1 C-Channel



X8 0.85" Screws



X8 Nylocks



► Focus: Layout procedure for building chassis

Date: May 15, 2025

Steps to Building Drivetrain Frame:

6



X1 27 Hole 1x3x1 C-Channel



X4 1.75" Screws



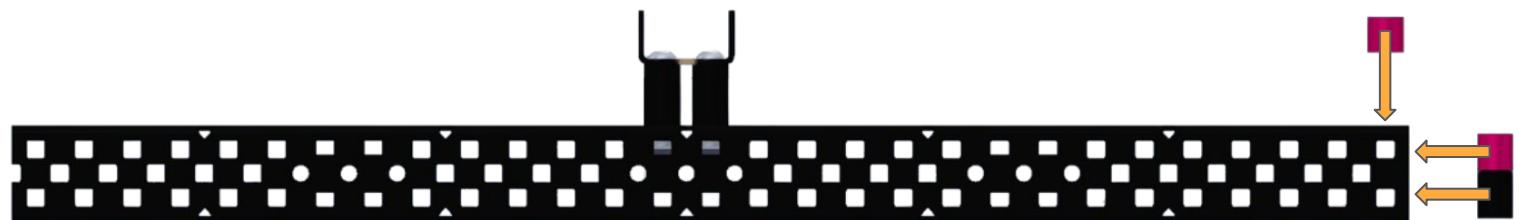
X4 Nylocks



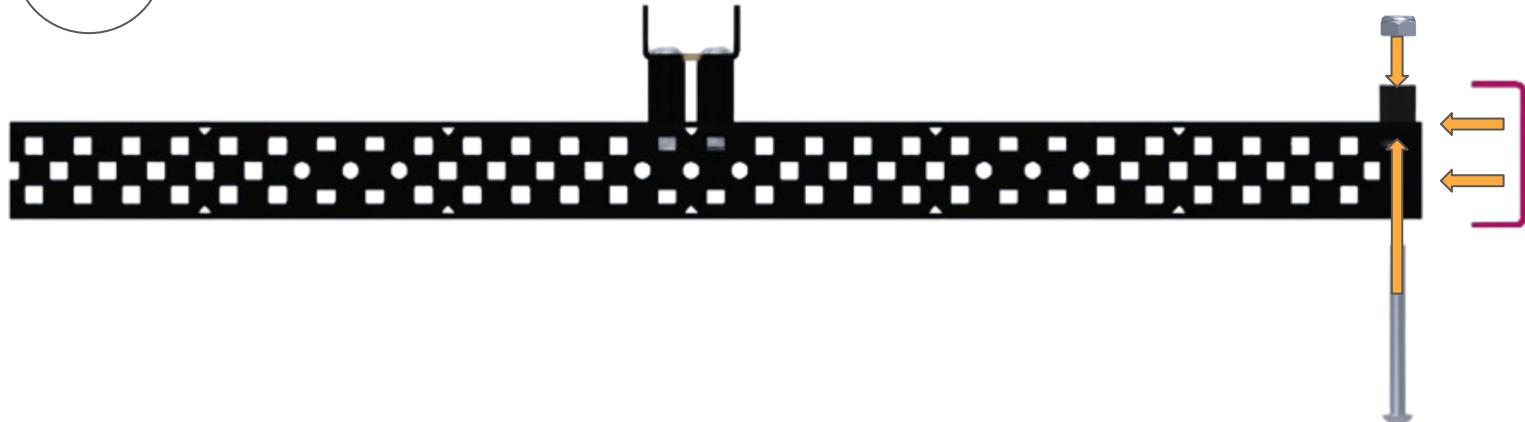
X4 $\frac{1}{2}$ " Spacers



X8 $\frac{3}{8}$ " Spacers

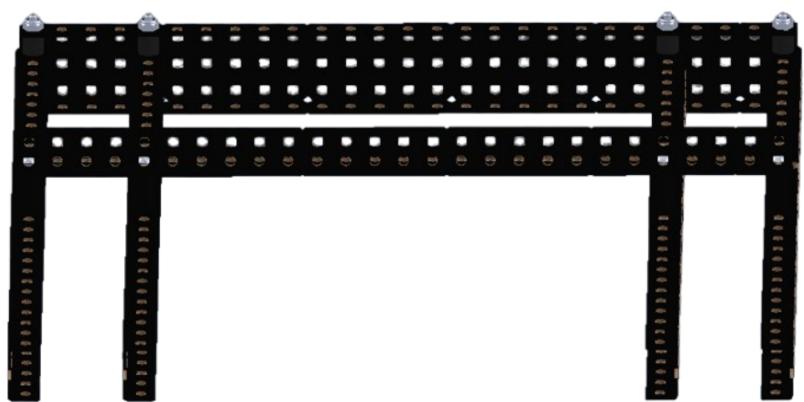


7



8

Final
Drivetrain
Frame is
complete!



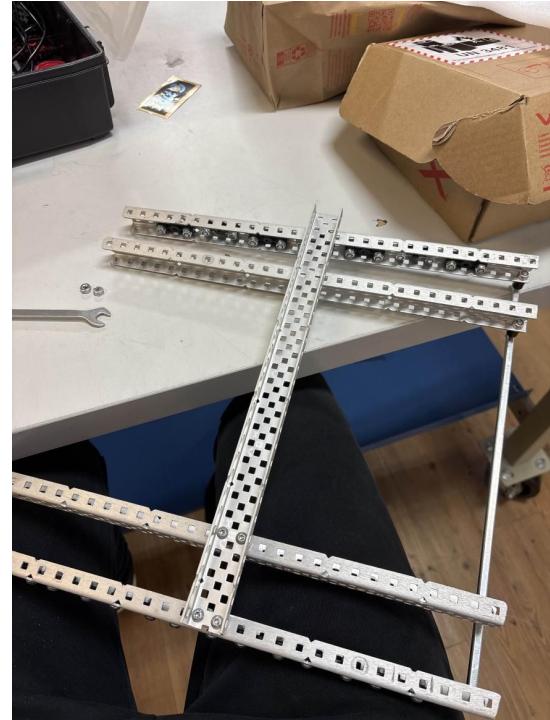
- ▶ Focus: Quality control

Date: May 15, 2025

Problem and solutions:

How do we keep the drivetrain frame straight throughout the building process?

By utilizing temporary bracing as well as shoulder screws to make sure there are no deformations when building.



Next Steps

After completing our drivetrain frame, we will move on to adding wheels, our back 3-wide brace as well as the electronic motors for our drivetrain to be fully completed.

Figure 1: Our semi-completed drivetrain chassis

- ▶ Focus: Completing the drivetrain

Date: May 17, 2025

Members Involved

Daniel, Bryan

Objective

The **main focus** today was first finish the drivetrain by attaching the wheels on both sides. Then we need to ensure the **same build quality** across the entire drivetrain, ensuring **smooth friction on both sides**. We attached a **3 wide c-channel** on the back of the robot as a **defensive wall** to **push people** around with. This acts as a **brace** across the **entire robot to maintain a rigid structure**.

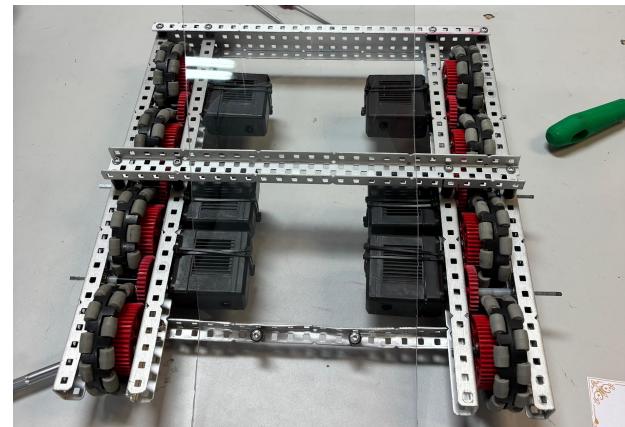


Figure 1: Drivetrain from the front

Materials:

- 8 11W Motors
- 12 0.375" Screws
- 4 0.85" Screws
- 16 1.25" Screws
- 10 2.5" Screws
- 18 Hex Nuts
- 26 Nylocks
- 6 Low-Strength Axle Collars
- Steel Washers (of an amount as needed)
- $\frac{3}{8}$ " Spacers (of an amount as needed)
- $\frac{1}{2}$ " Spacers (of an amount as needed)
- 2, 7 Long 1x2x1 C-Channels
- 2, 5 Long 1x1 Angle Bars
- ~3" Low-Strength Axles
- 8, 36 Tooth Low Strength Gears
- 8, 48 Tooth High Strength Gears
- 8, 3.25" Omni-Directional Wheels
- Drivetrain Frame from Day 1

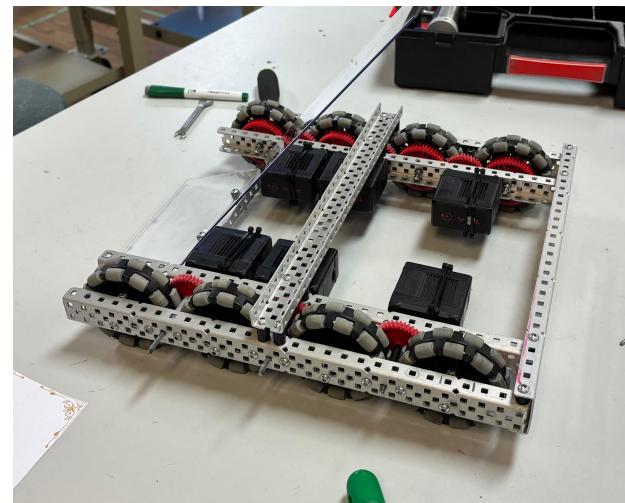


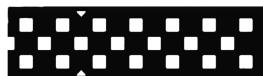
Figure 2: Drivetrain from the side

- ▶ Focus: Layout procedure for building chassis

Date: May 17, 2025

Steps to Finishing Drivetrain:

1



X1 7 Hole 1x2x1 C-Channel



X2 ½" Spacers

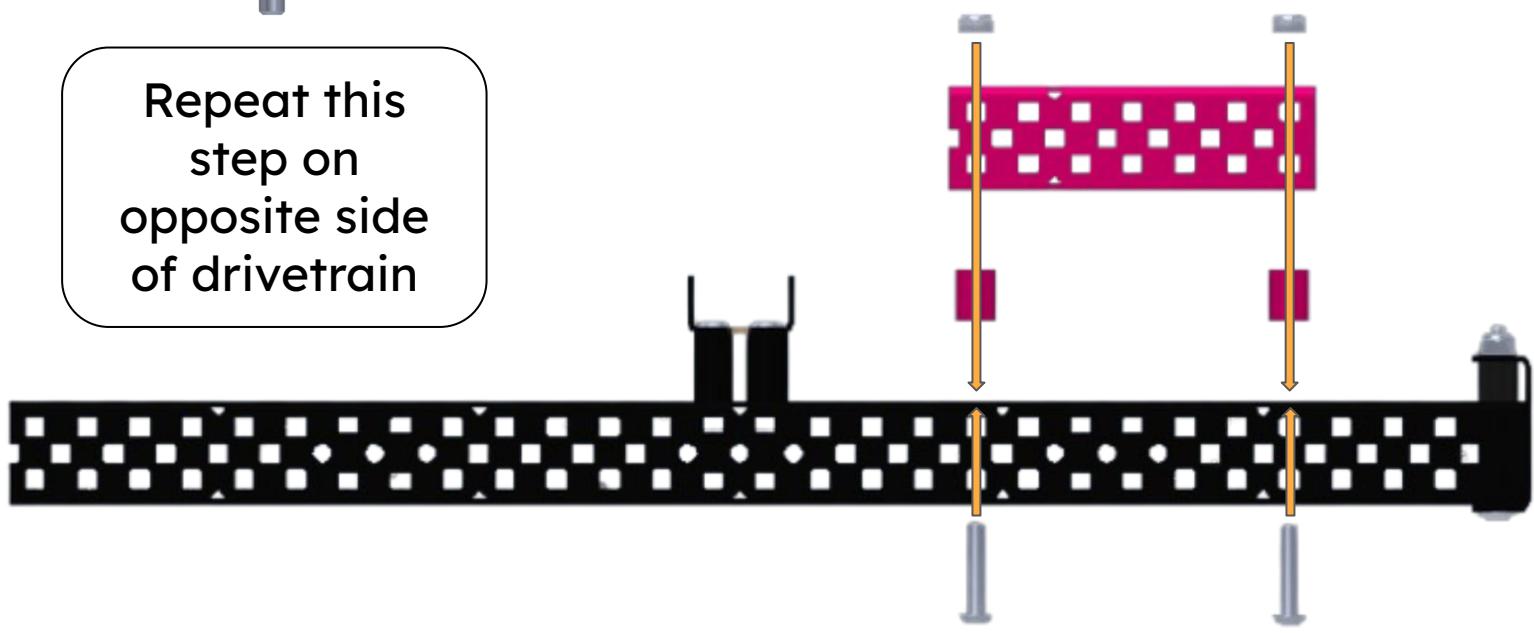


X4 0.85" Screws



X4 Nylocks

**Repeat this
step on
opposite side
of drivetrain**



**Together, it should
look like this**



- ▶ Focus: Layout procedure for building chassis

Date: May 17, 2025

Steps to Finishing Drivetrain:

2



X1 5 Hole 1x1 Angle Bar



X4 $\frac{1}{2}$ " Spacers



X2 1.25" Screws



X1 Nylocks



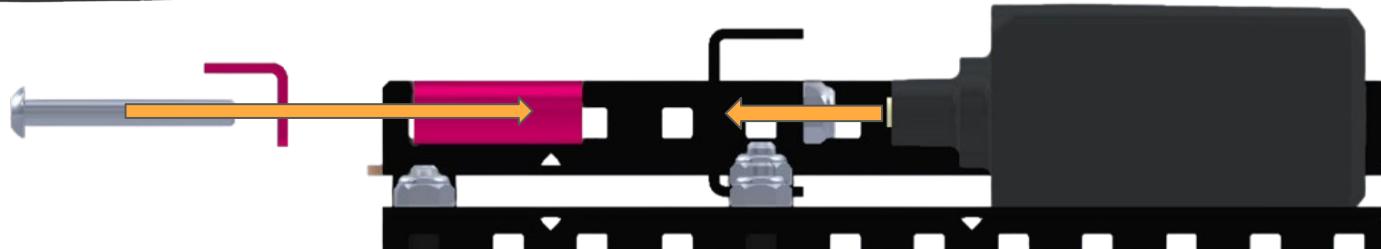
X3 Bearing Flats



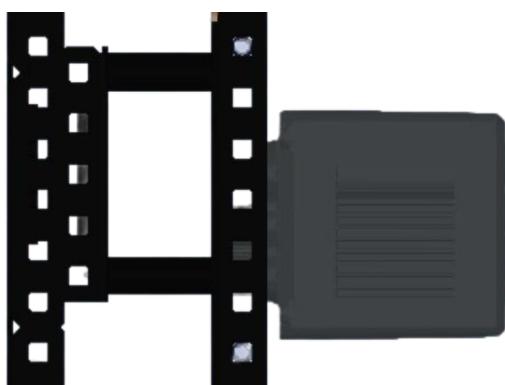
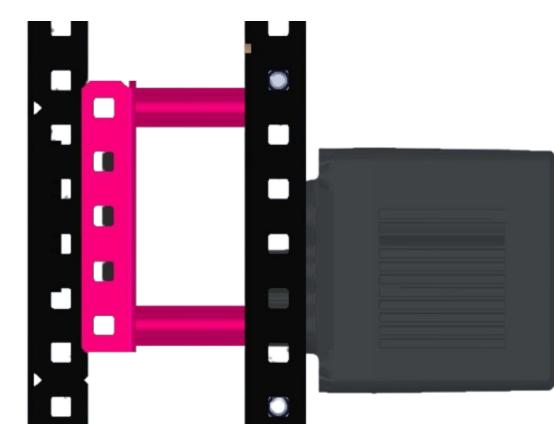
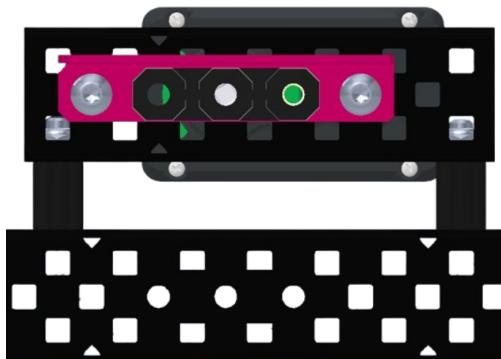
X1 11W Motor



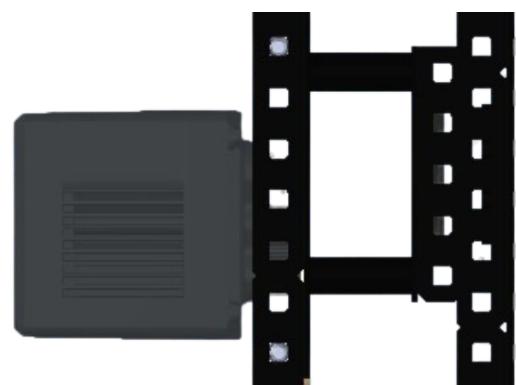
X1 0.375 Screw



Repeat this
step on
opposite side
of drivetrain



Together,
it should
look like
this
 \Leftrightarrow



- Focus: Layout procedure for building chassis

Date: May 17, 2025

Steps to Finishing Drivetrain:

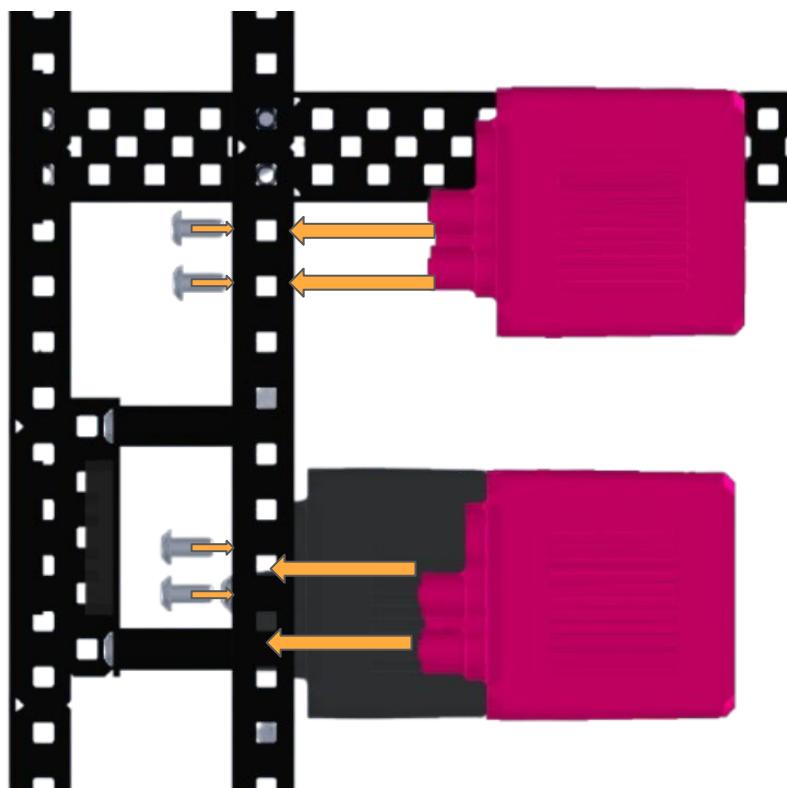
3



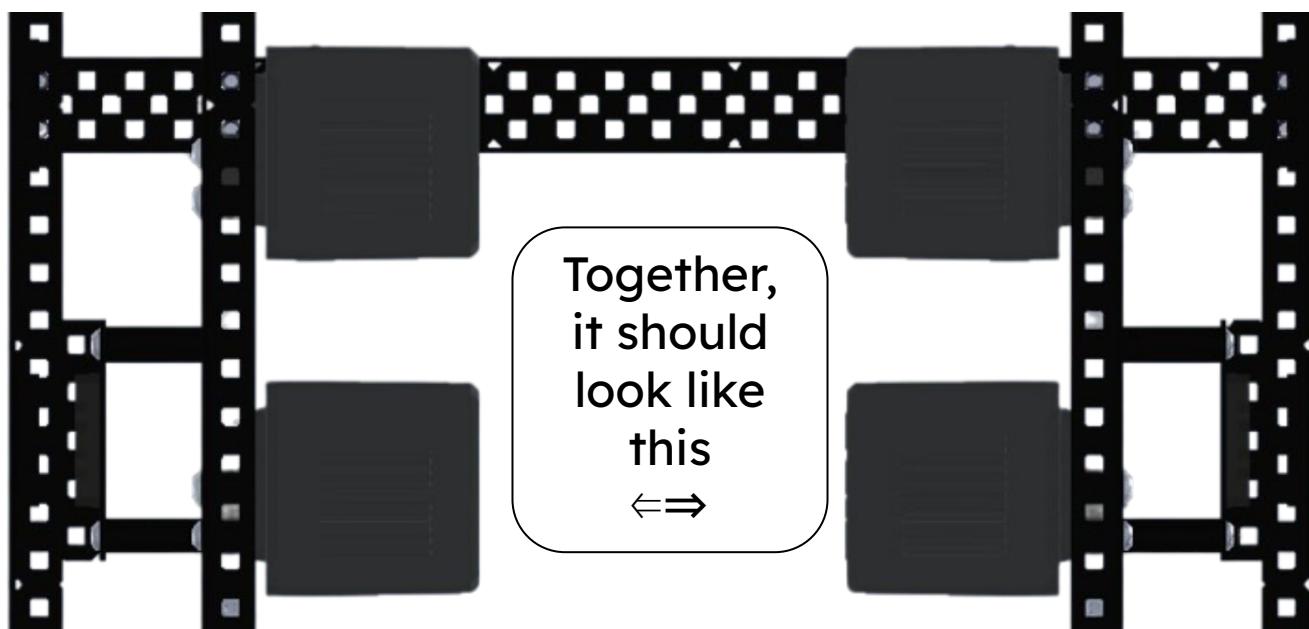
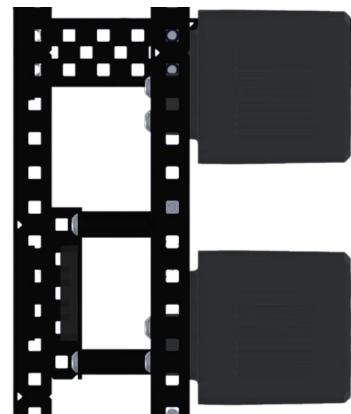
X2 11W Motor



X4 0.375 Screw



Repeat this
step on
opposite side
of drivetrain



- ▶ Focus: Layout procedure for building chassis

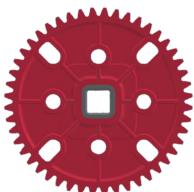
Date: May 17, 2025

Steps to Finishing Drivetrain:

4

Wheel Pods

x8



X1 48 Tooth Gear



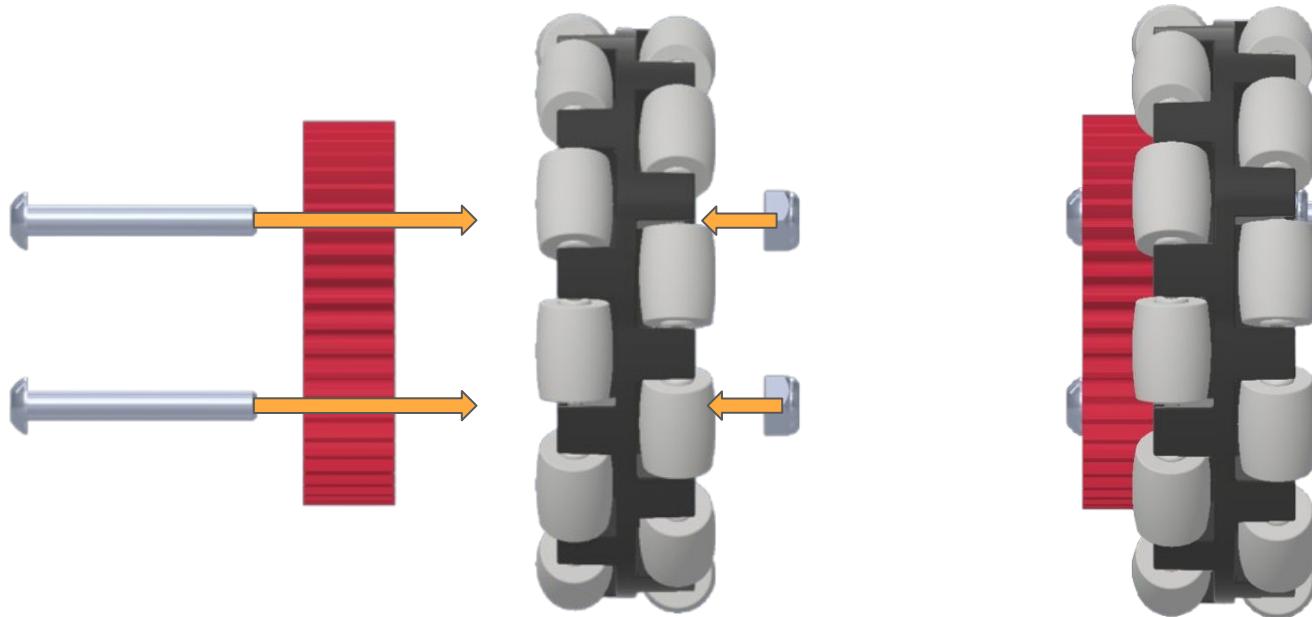
X1 3.25" Omni Wheel



X2 1.25" Screw



X2 Nylocks



- Focus: Layout procedure for building chassis

Date: May 17, 2025

Steps to Finishing Drivetrain:

5

■ **X1 $\frac{1}{8}$ " Spacer**

■ **X1 $\frac{1}{2}$ " Spacer**

■ **X1 $\frac{3}{8}$ " Spacers**

■ **X1 $\frac{3}{8}$ " Axle Collar**

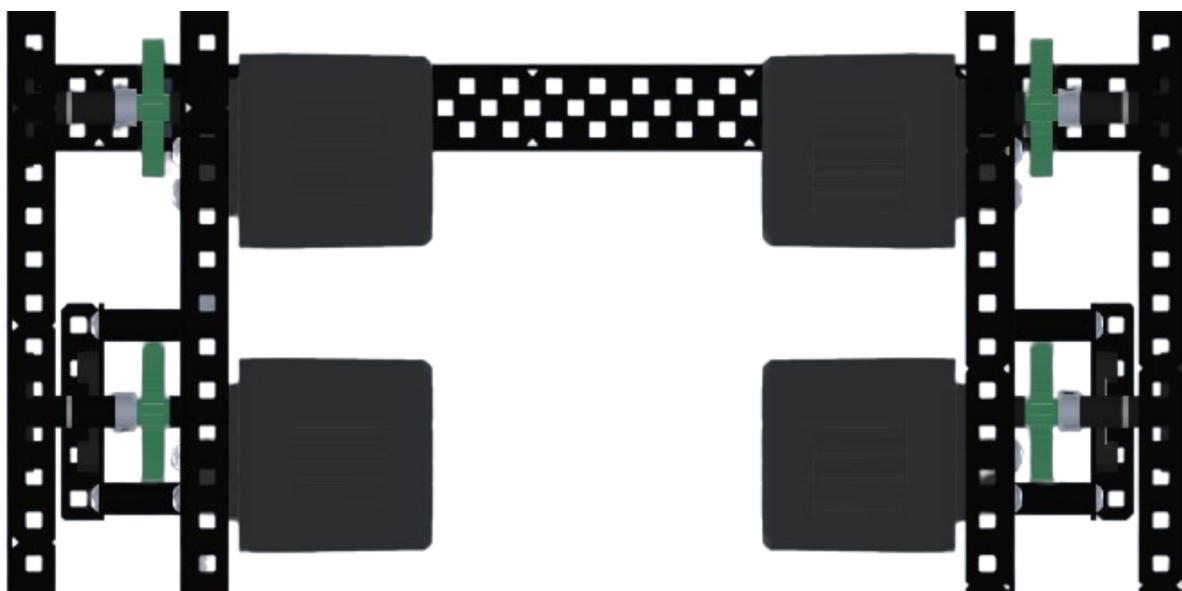
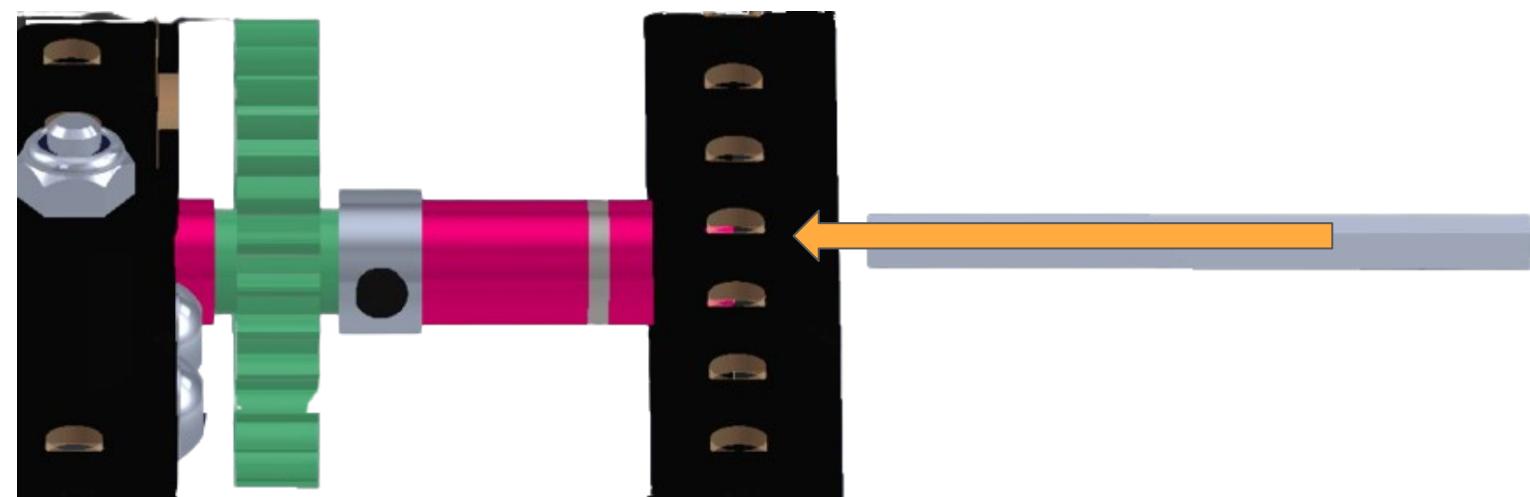


X1 36 Tooth Gear

X2 Washers

**x4
On lower
motors**

**X1 3" Low-Strength
Axe**



- ▶ Focus: Layout procedure for building chassis

Date: May 17, 2025

Steps to Finishing Drivetrain:

6

■ **X2 1/8" Spacer**

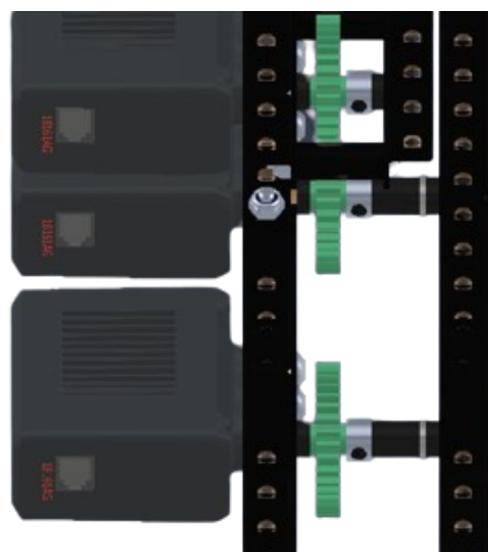
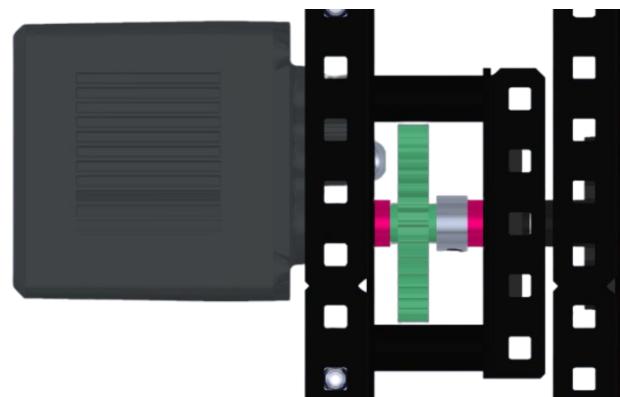
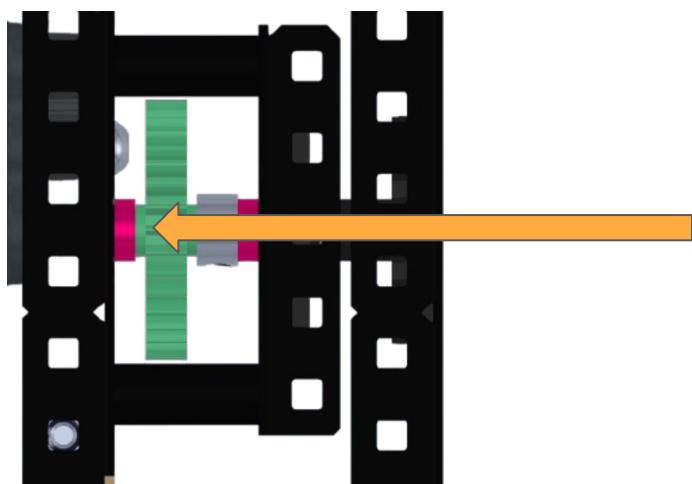
■ **X1 Axle Collar**



X1 36 Tooth Gear

**Repeat x2
On Motor
Stacks**

X1 2" Low-Strength Axle



► Focus: Steps to Finishing Drivetrain

Date: May 17, 2025

Steps to Finishing Drivetrain:

7



X1 2.25" Screw



X2 Hexnuts



X1 Wheel Pod



X1 Nylocks

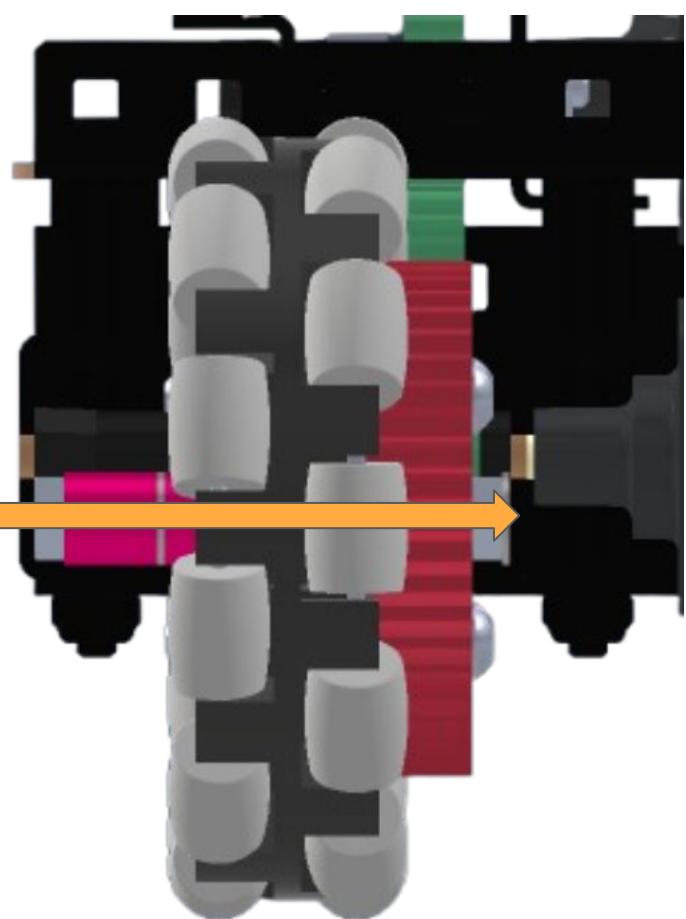


X1 3/8" Spacers



X2 Washers

**Repeat
for 8
Wheels**



► Focus: Steps to Finishing Drivetrain

Date: May 17, 2025

Steps to Finishing Drivetrain:

8



X1 36 Tooth Gear



X1 2.25" Screw

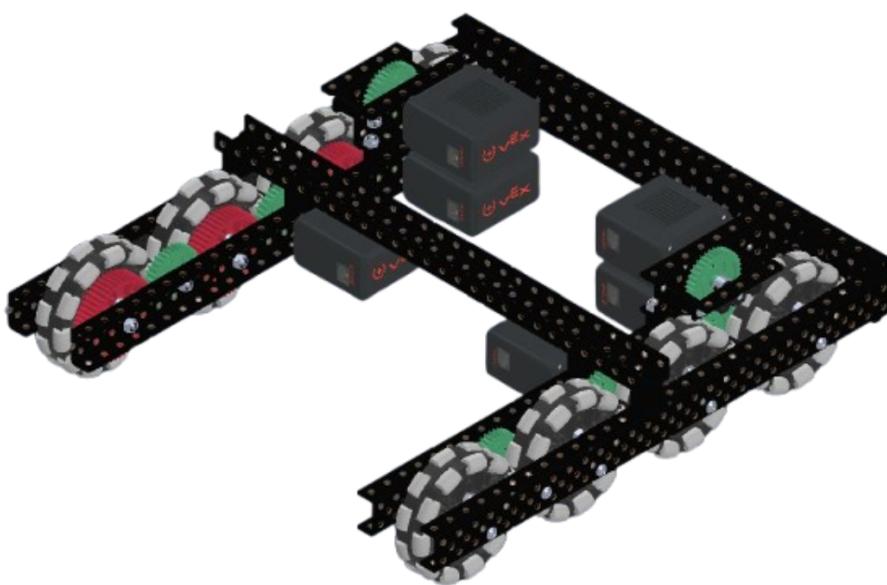
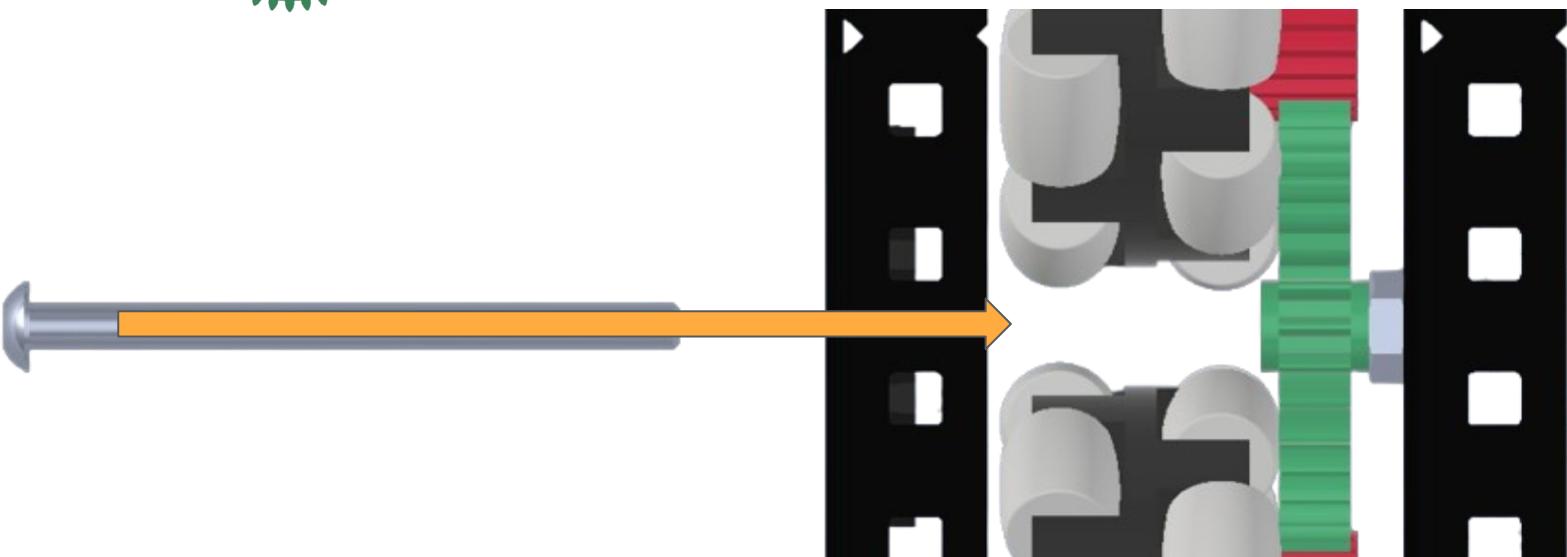


X1 Hexnuts



X1 Nylocks

X2



Here is the
final version
of the 210K
C1.0
Drivetrain

- Focus: Completing the drivetrain

Date: May 17, 2025

Problems and Solutions:

The **positioning of our motors** would be **difficult** to work with when **building other systems** such as our **intake** and **ramps**.

To solve this issue, we switched the position of the **motors** to create a **motor stack**.

How do we ensure there is **balanced friction** on both sides of the drivetrain?

We tuned the friction by adjusting the **tightness** of each **screw joint, spacing**, as well as making sure all parts would not **bend** under pressure.

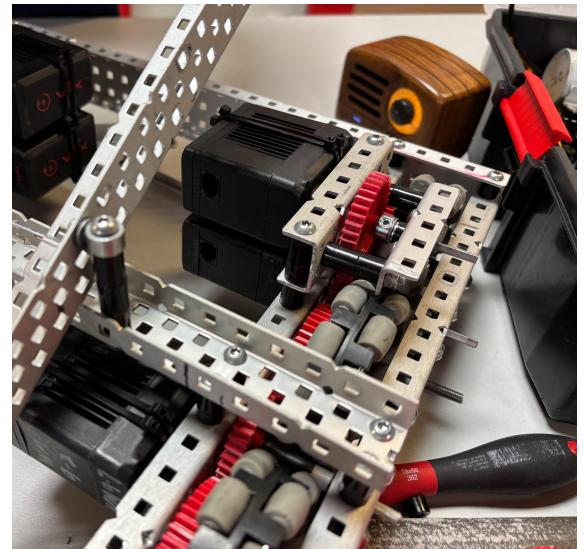


Figure 1: Side view of motor stack

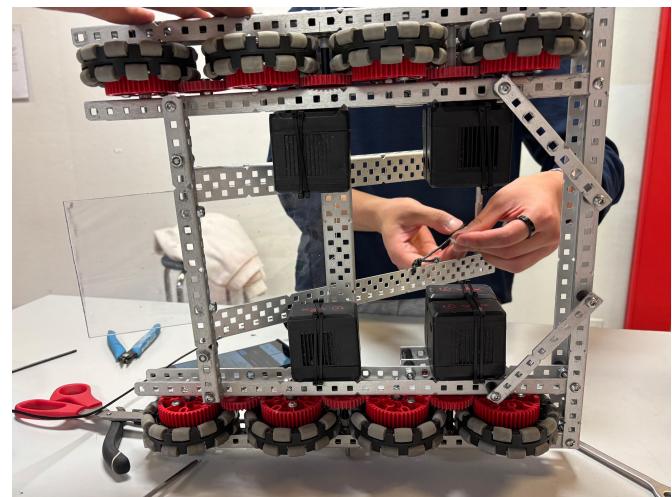


Figure 2: Bottom view of drivetrain

Next Steps

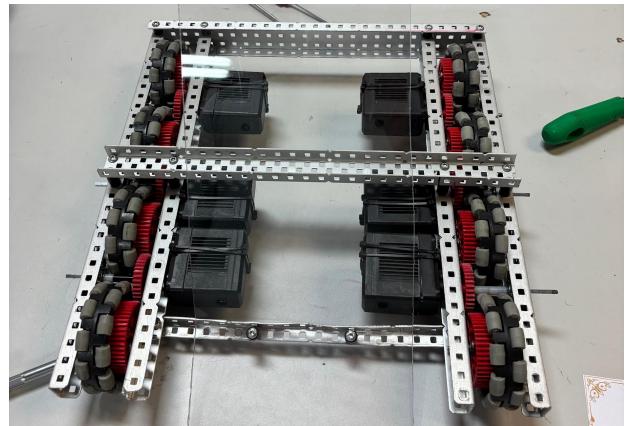
We want to **test** our drivetrain to ensure it is consistent enough to last throughout a match. We will then move onto starting to build the **intake** by positioning a **ramp** at the front of the drivetrain.

- ▶ Problem: Testing the friction of the drivetrain

Date: May 19, 2025

Purpose

Determine any **friction-based** issues in the drivetrain. Friction causes the drivetrain to be less efficient and heats the motors faster, which is **undesirable**. We will aim for **>14s** of freespin.



Procedure

1. Have one team member ready with a **stopwatch**, and another with the **drivetrain**.
2. Count down from three, then have the team member with the drivetrain push it **against the table**, then lift it up, causing the wheels to spin. The team member with the stopwatch should **start** the stopwatch as soon as the drivetrain **loses contact** with the table.
3. **Stop** the stopwatch when one set of wheels on the drivetrain **stop spinning**.
4. **Record** the result under “Freespin time, t (s)” to **three significant digits**. Repeat steps 2-3 until enough trials are produced.

Table 1: Freespin time, t (s), in relation to trial number

Trial Number	Freespin time, t (s)
1	15.3
2	14.2
3	14.7
4	15.1
5	14.8

Analysis & Conclusion

The drivetrain test produced **satisfactory** results. We aimed for over 14 seconds of freespin, and we produced an **average of 14.8s** (14.82s). Since the drivetrain performed to our satisfaction, we can now **move onto another subsystem**: the **intake**.

Version 1.0: Mall of America Signature Event

Intake

- ▶ Problem: How do we pick up a block from the ground?

Date: May 22, 2025

Identify the Problem

In order to use game objects, we need to move them off the ground and into our robot. This is a common trend through VEX games as most game objects will start contacting field tiles. This creates a challenge as the game objects for Push Back have a unique shape.

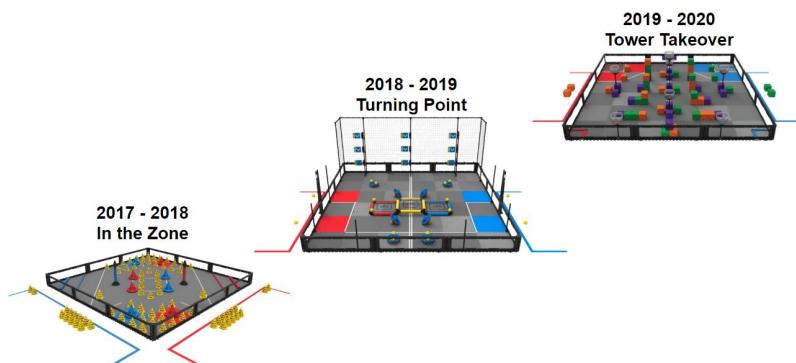


Figure 1: Different Fields throughout the years with different game elements (VEX website)

► Focus: Concepts

Date: May 22, 2025

General Concepts

After analyzing the game, we have **brainstormed** several general ideas and concepts that represent different approaches to this game.

X-Lift

We developed two main concepts, first is a **X-lift system** that incorporates a horizontal **holding chamber** that can be **lifted to different heights** to score on different tubes.

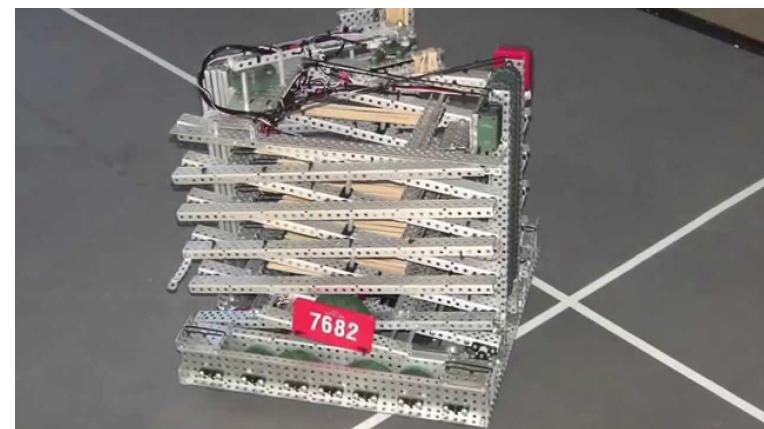


Figure 1: Example of a X-lift (7682, Vex Forums)

Snail Bot

The second design incorporates a **Change Up design** dubbed the “snail” robot. It uses an **curved shape** that was incorporated into change up robots. This design was also implemented into flywheel robots during **Spin Up**. Using this concept we believe it can be modified to function for Push Back.

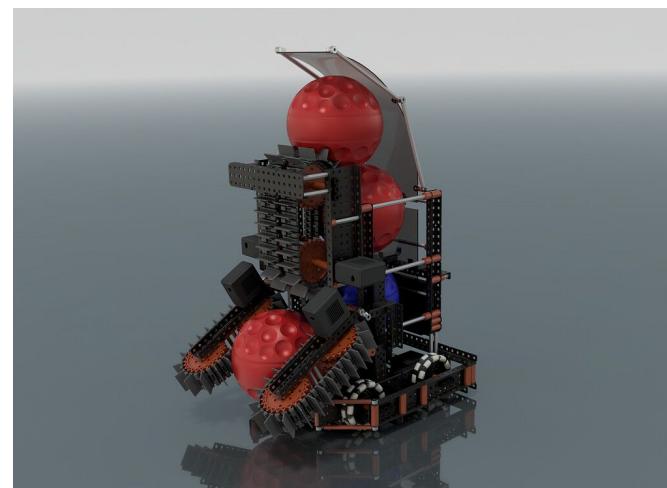


Figure 1: Example of a snail bot during change up (60172S, Vex Forums)

- ▶ Focus: Brainstorming different intake designs

Date: May 22, 2025

Vertical style rollers

When we first looked at the **match loading tubes**, we thought that it resembled the goals from **VRC Change Up**. The change up tube favours **vertical style intakes** just like the match loading tubes.

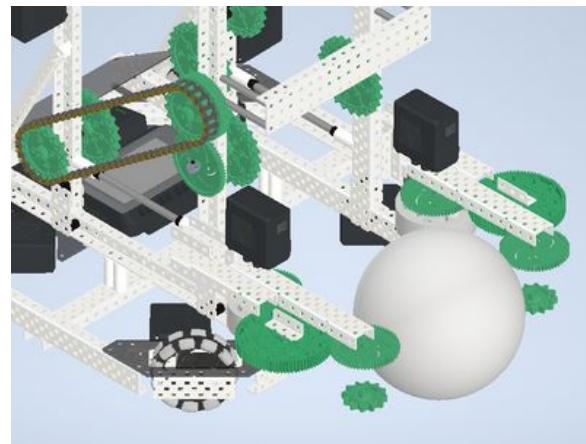


Figure 1: Change up vertical intake (VEX Forums)

Horizontal rollers

Horizontal rollers could be created in multiple ways, looking at past games such as VRC Over Under and Spin Up.

Flex wheels

- Flex wheel rollers use various sizes of flex wheels mounted on a pivoting frame.

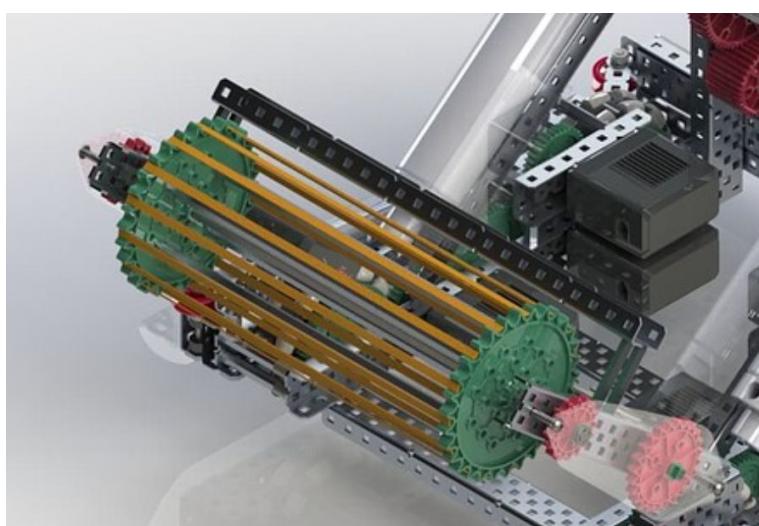


Figure 2: 81988E's rubber band intake in over under (VEX Forums)

Mesh

- Mesh rollers would be very similar to rubber bands, but they would be wrapped in an extra layer of mesh to help with grip

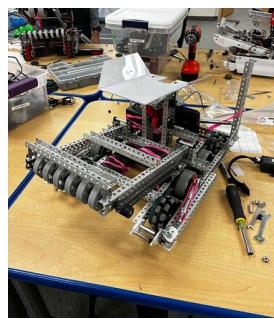


Figure 3: 3142C's vertical flex wheel intake in Over Under (VEX Forums)

Figure 4: 2029C's mesh intake in Over Under (VEX Forums)

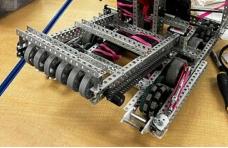
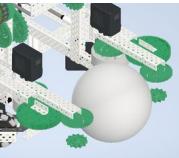
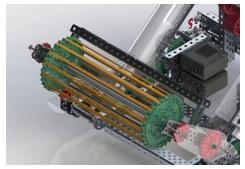


- Focus: Selecting the best type of rollers for our intake

Date: May 23, 2025

After brainstorming, we took into consideration many of the designs we thought about and ranked them based on some of the following criteria.

1. **Ease of Build** (1 - 5) - This criteria is ranked based on how easy the rollers are to build
2. **Flexibility** (1 - 5) - Ranked on how flexible each type of intake is
3. **Risk of Entanglement** (1 - 5) - Ranked how likely the intake is to get entangled during a match (Lower = More risk)
4. **Weight** (1 - 5) - Ranked on how heavy the system is

<u>Criteria Options</u>	Ease of Build	Flexibility	Risk of Entanglement	Weight	Total
Vertical Flex Wheels 	2	2	5	1	10
Horizontal Flex Wheels 	4	3	5	2	14
Rubber Bands 	4	5	2	4	15
Mesh 	3	4	3	4	14

- ▶ Focus: Unique rubber band design

Date: May 23, 2025

Using data from our design matrix table, we visualized our options using a radar graph.

Comparison of V1 Intake Designs

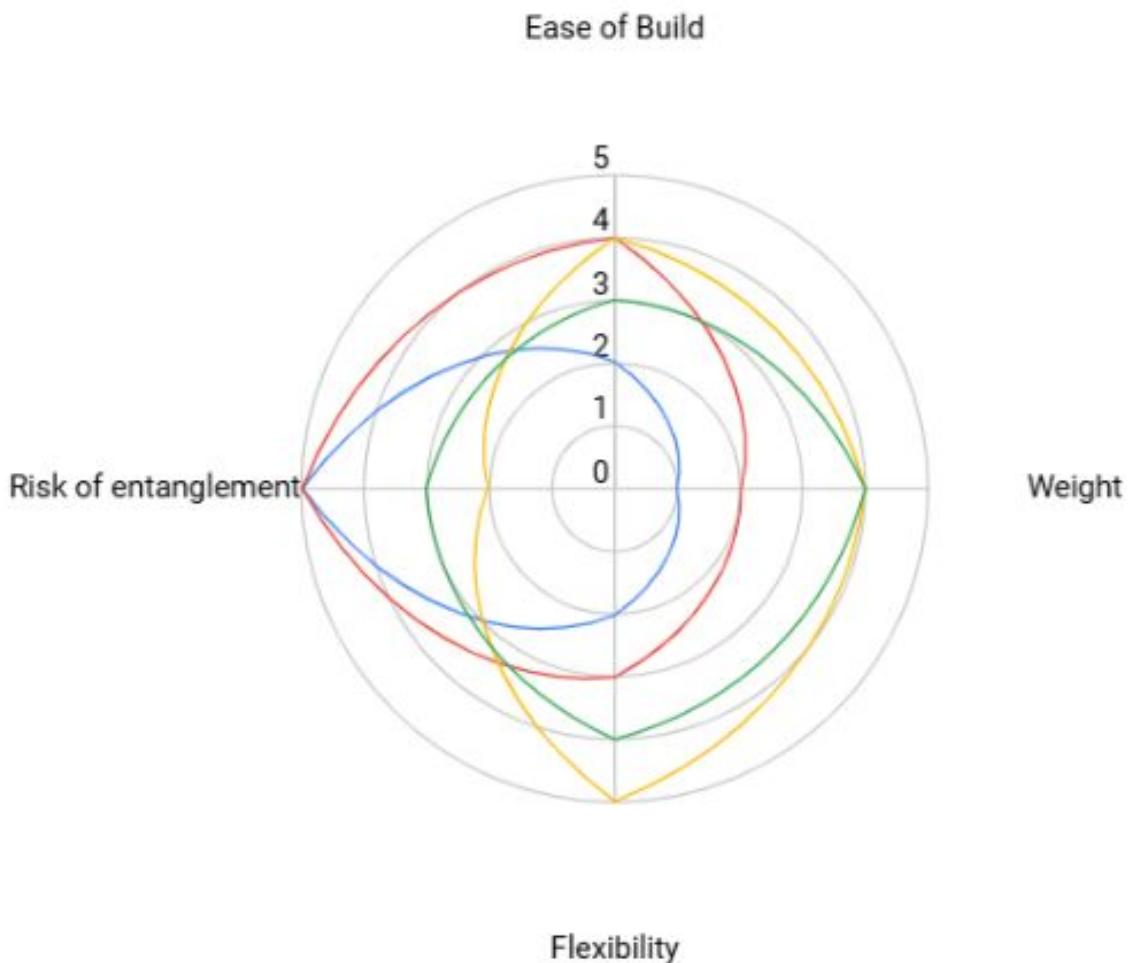


Chart: Comparing Intake Designs

Looking at the different intake variations, we decided to pick something well **rounded and balanced**. Taking into consideration of these different designs, we found the **rubber bands** was the most balanced option to build.

- ▶ Focus: Planning our first stage intake

Date: May 25, 2025

Planning our intake

When first planning our intake, we chose to look at an **older design** from **Over Under**. This design utilizes **rubber bands** wrapped on sprockets so we decided to plan our intake with a similar concept. Our intake would use sprockets with **thicker rubber bands** because they would be less prone to snapping.

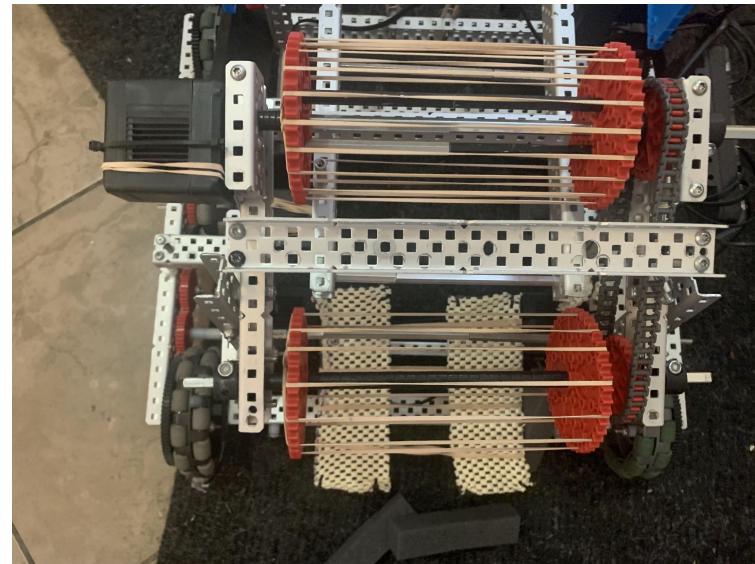


Figure 1: Over Under Rubber Band Intake Built by us

Bracing

We knew from previous games that sprockets will tend to **bend and crack** under the force of the rubber bands, so we planned to prevent this by **bracing our sprockets with standoffs**. This provides a **rigid structure** and prevents the snapping or bending that may happen



Figure 2: Braided goliath intake (Bilibili)

Hourglass Shape

We decided to look into a unique **hourglass** shaped intake using rubber bands known as the "**braided goliath**" intake. This incorporates twisting rubber bands into different position to create a **pattern** like **braiding** hair.

- Focus: Building Intake and Ramp

Date: May 29, 2025

Members Involved:

Daniel, Bryan

Objective:

The **main focus** today was to build a structural base and start our intake.

Structural Ramp

- Our **main structural component** of our ramp uses **L-channels**.
 - This allows us to have a **stable base** to attach rollers for our intake.
- We also utilized a piece of **bent polycarbonate** to help with picking up blocks

Intake Roller

- **We brace the sprockets** to prevent bending from the force the rubber band exerts on it
- We braided the rubber bands to create a unique **hourglass** shape

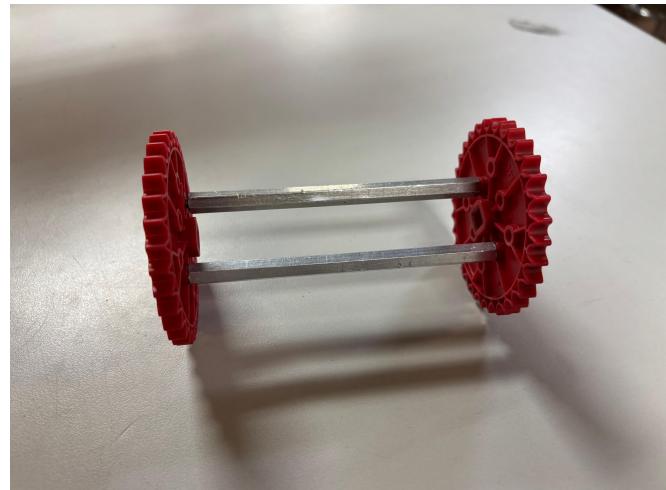


Figure 1: Bracing our sprockets



Figure 2: Braiding rubber bands to create a hourglass shape

► Focus: Building Intake and Ramp

Date: May 29, 2025

Materials:

- 1 drivetrain
- Various high strength spacers
- Various regular spacers
- Screws and nylocks
- Shaft collars
- Bearings
- 2, 25 hole 2x2 L-Channel
- 2, 9 hole, 1x2x1 C-Channel
- 2, 1 hole 2x2x2 U-Channel
- 1, 6 hole 1x2x1 C-Channel
- 2, 14 hole 1x2x1 C-Channel

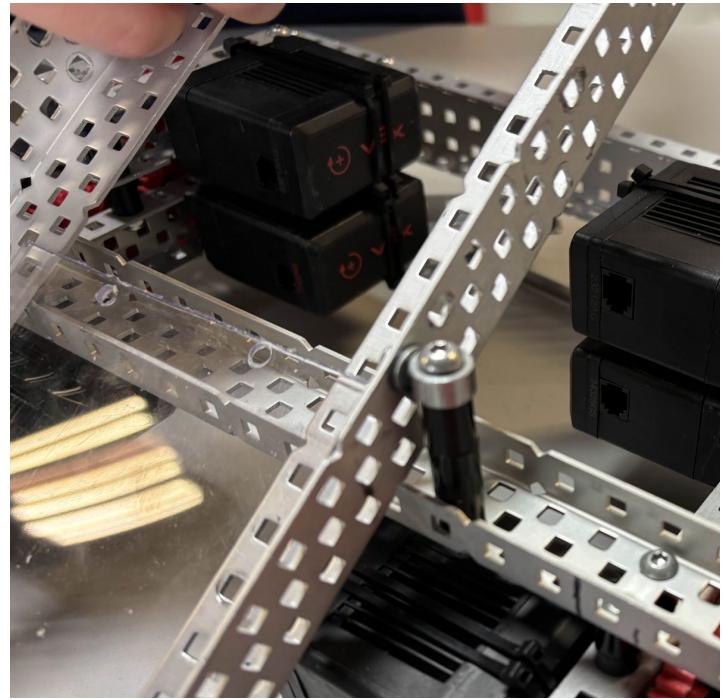


Figure 1: Ramp mount using shaft collars

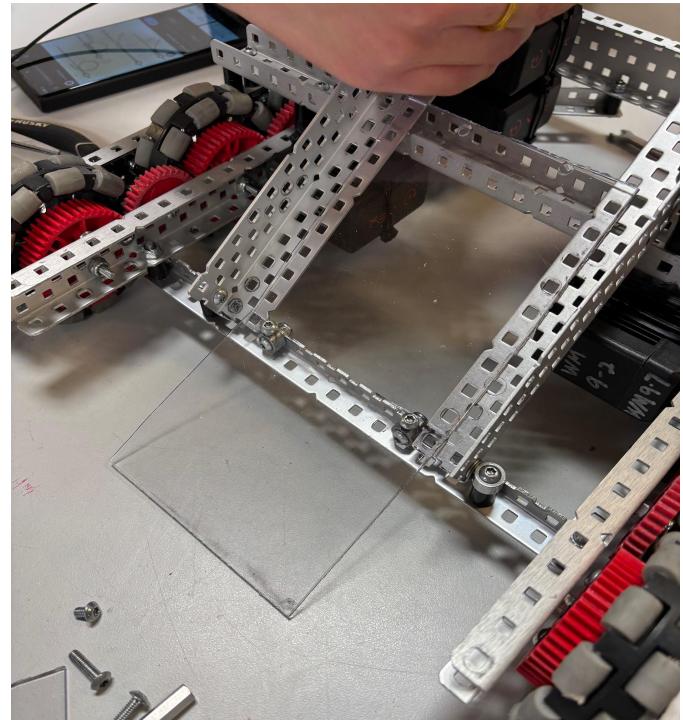


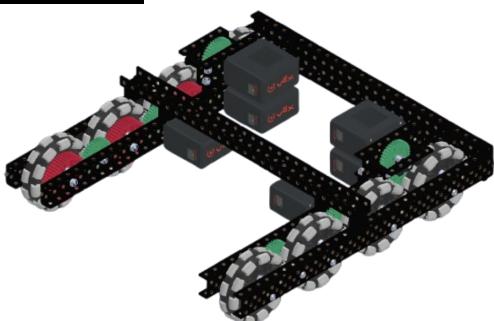
Figure 2: Intake Ramp Structure

- Focus: Building intake

Date: May 29, 2025

Starting Intake

1



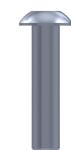
X1 Drivetrain



X2 ½" Spacers



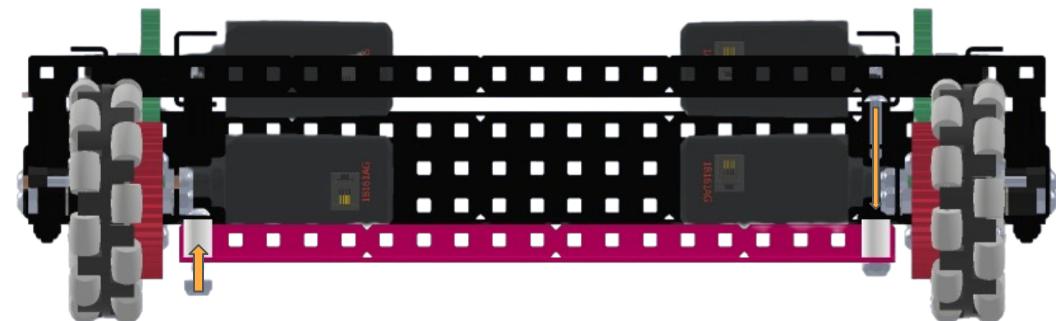
2X Nylocks



X2 0.75" Screws



X1 19 Hole 1x1 Angle Bar



2



X1 ½" Spacers



X1 ⅜" Axle Collar



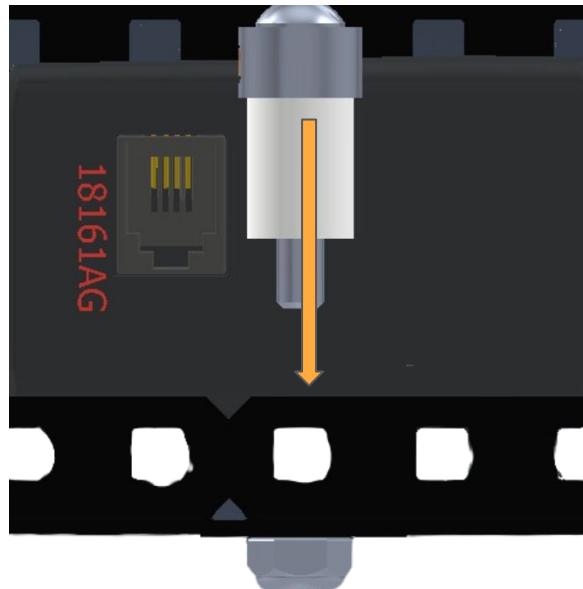
x4

- Focus: Building intake

Date: May 29, 2025

Starting Intake

3



x2



2X Nylocks



X2 1" Screws



Repeat and Mirror on other side

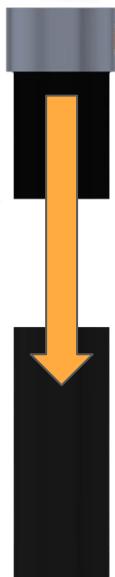
4



x4 ½" Spacers

x2

x2

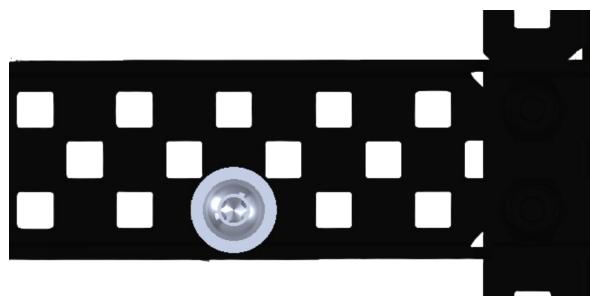


► Focus: Building intake

Date: May 29, 2025

Starting Intake

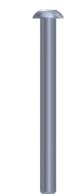
5



X2



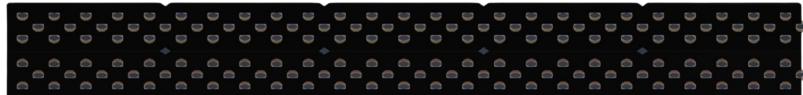
2X Nylocks



X2 2" Screws

Repeat and Mirror on other side

6

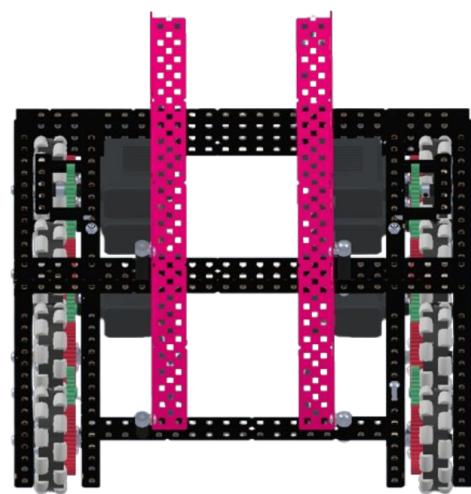
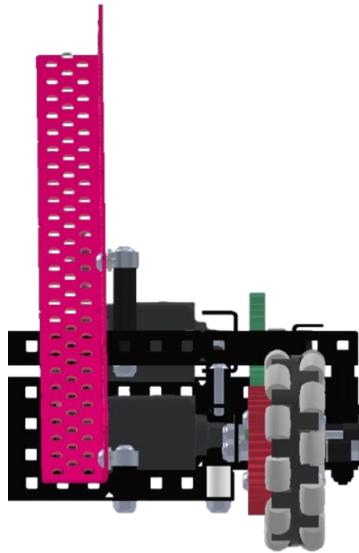
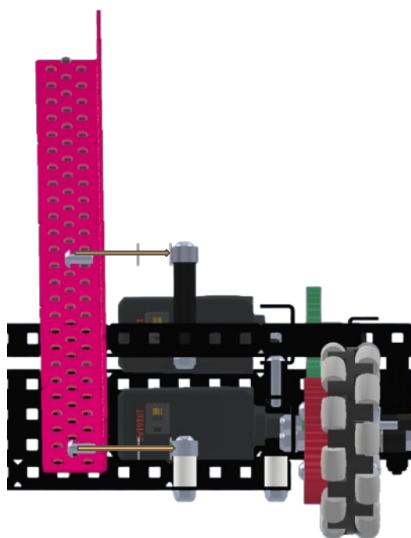


X2 25 Hole 2x2 L-Channel

X2 Washers



X1 0.375 Screw



► Focus: Building intake

Date: May 29, 2025

Starting Intake

7



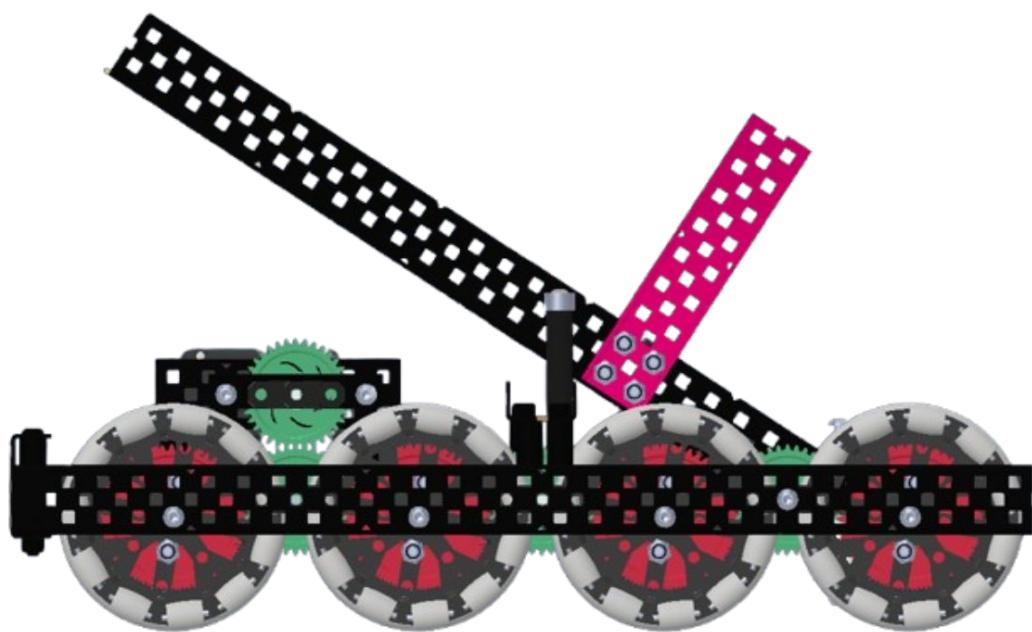
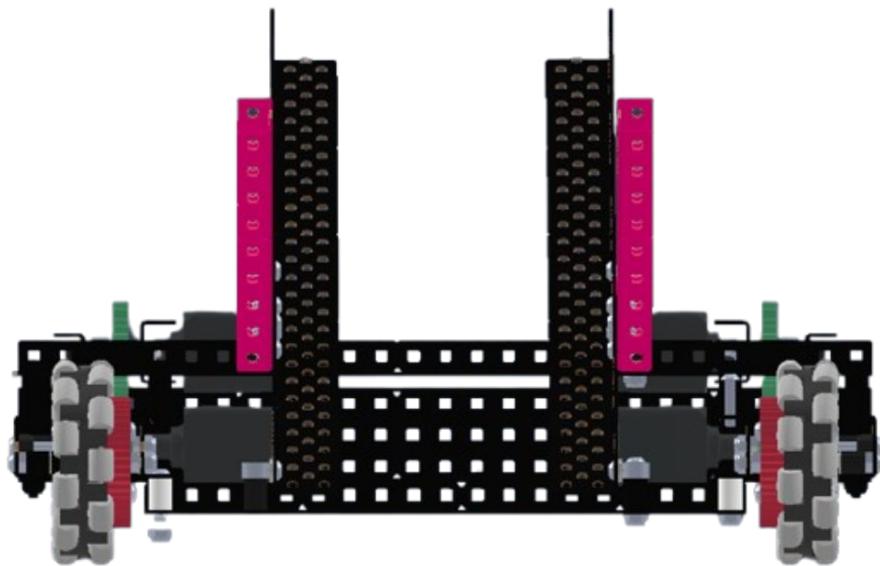
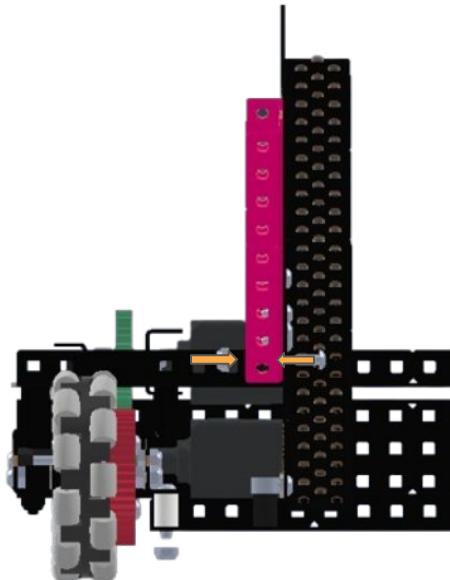
X2 9 Hole 1x2x1 C-Channel



X8 0.375 Screw



X8 Nylocks



► Focus: Building intake

Date: May 29, 2025

Starting Intake

8



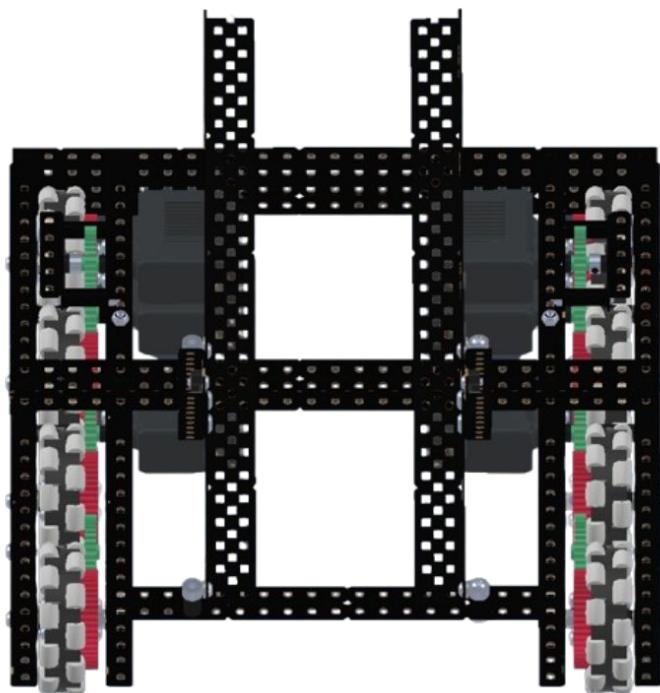
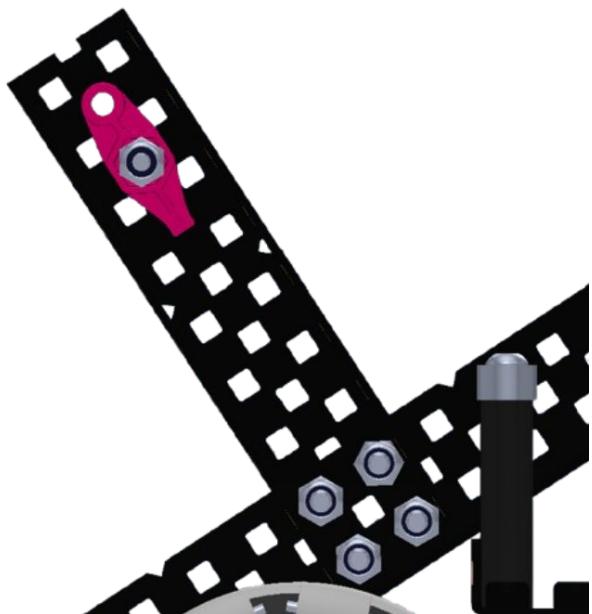
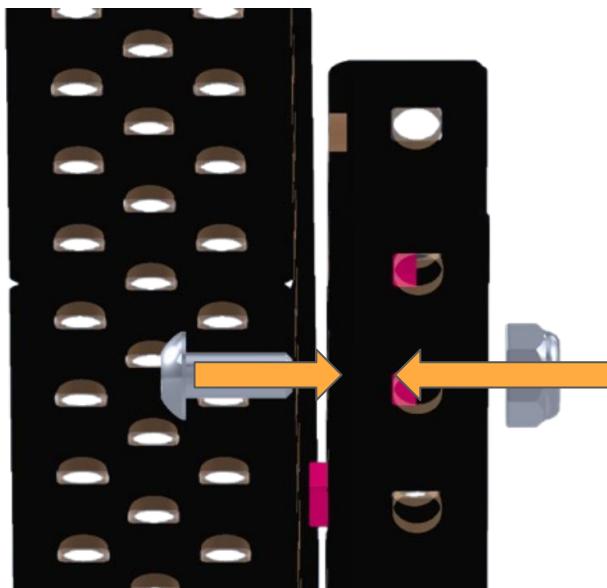
X2 Bearings



X2 Nylocks



X2 0.375 Screw



**Repeat on both
sides of the
C-Channel**

► Focus: Building intake

Date: May 29, 2025

Starting Intake

9



X2 U Channel

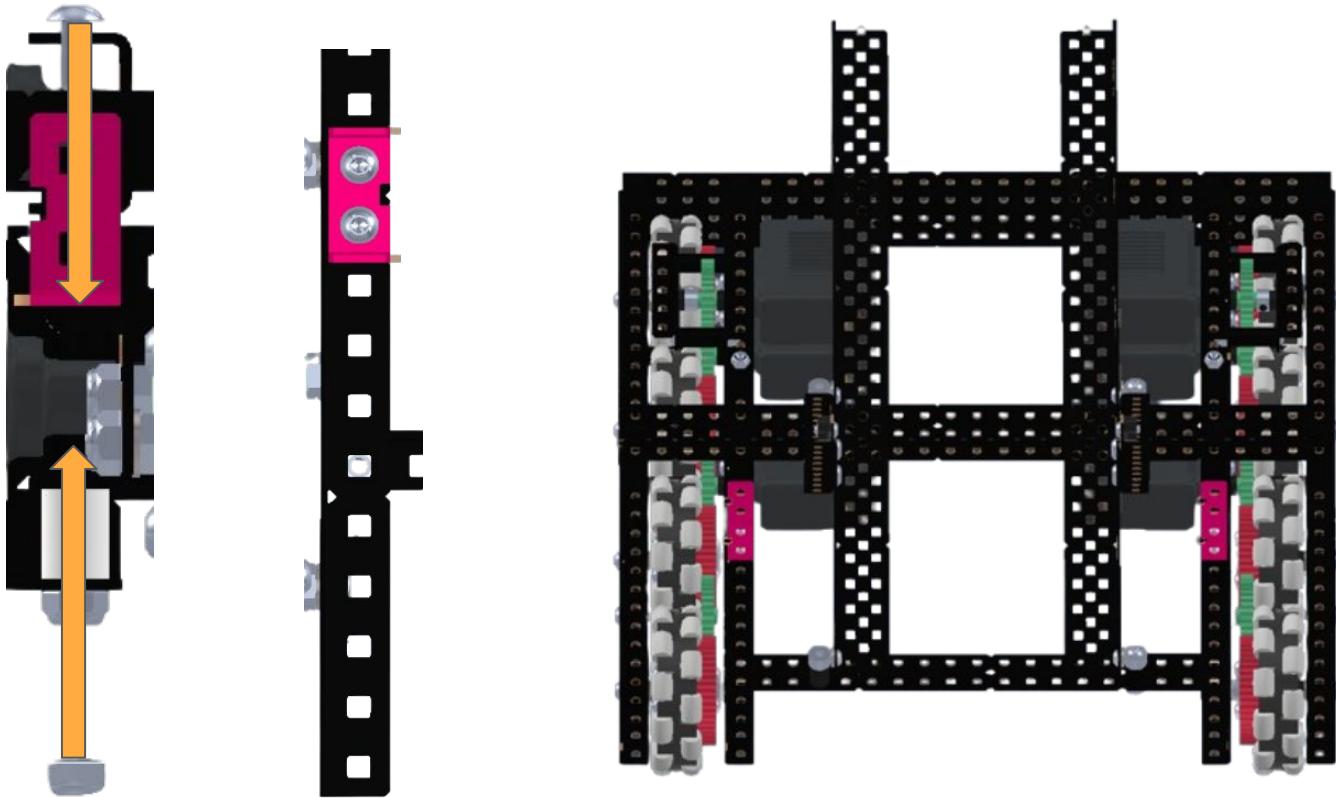


X2 Nylocks



X2 0.375 Screw

**9th Hole on the
C-Channel**



► Focus: Building intake

Date: May 29, 2025

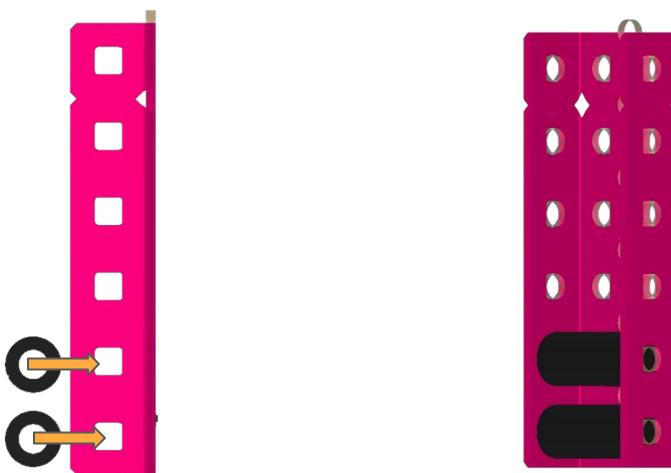
Starting Intake

10



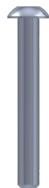
X1 6 Hole 1x2x1 C-Channel

- X2 3/8” Spacers**
- X2 1/2” Spacers**



x2

11

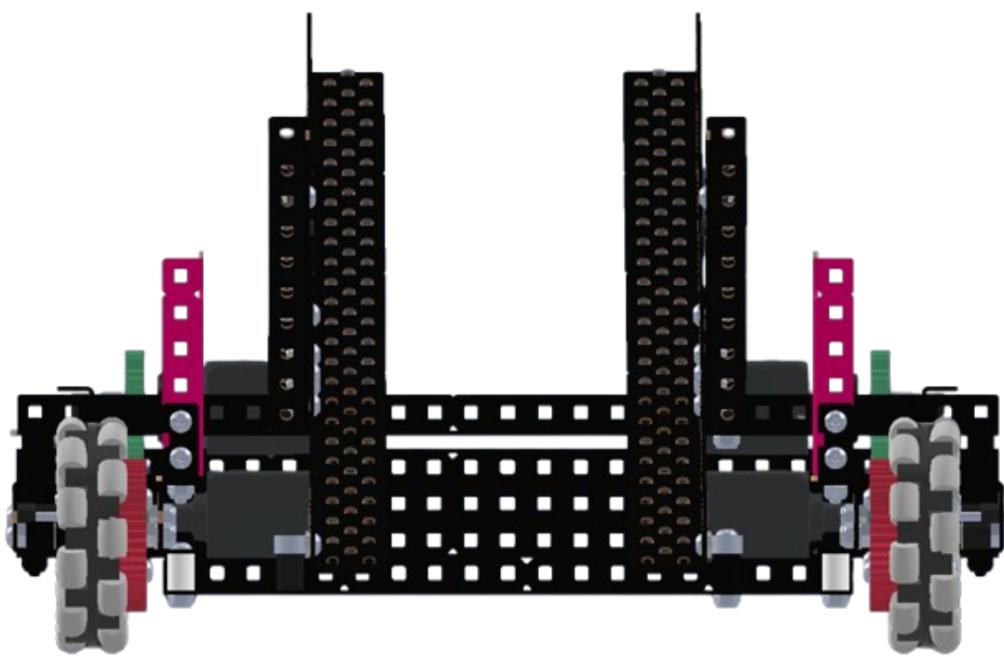


X4 1.25” Screw



X4 Nylocks

Both Sides



► Focus: Building intake

Date: May 29, 2025

Starting Intake

12



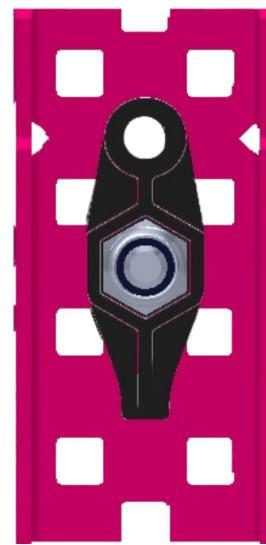
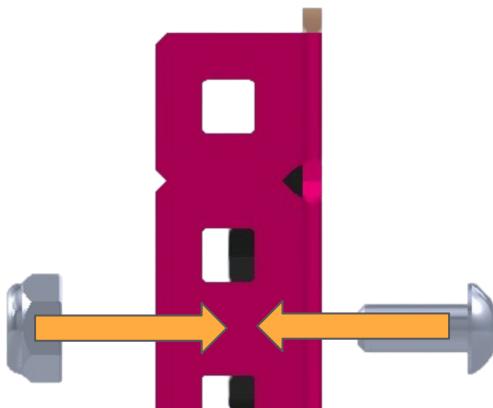
X2 Bearings



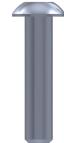
X2 Nylocks



X2 0.375" Screw



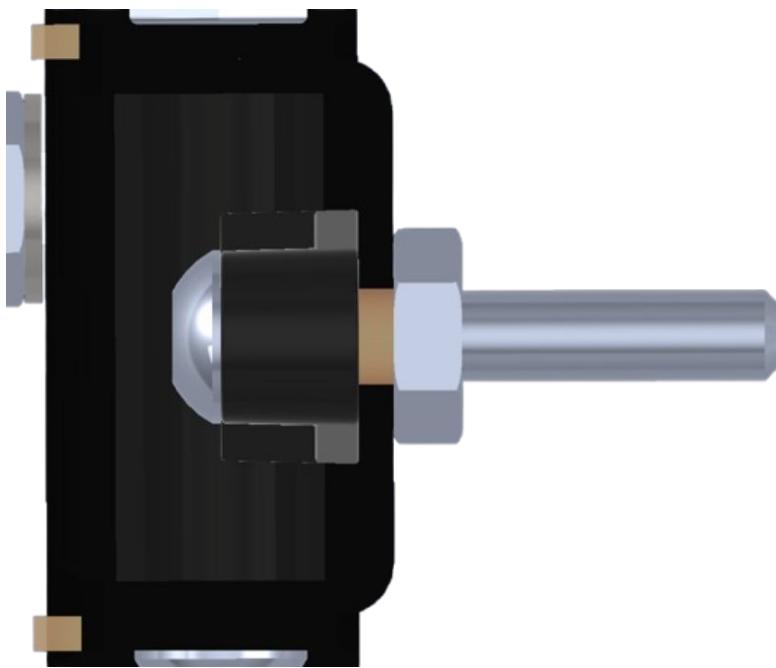
13



X2 1" Screws



X2 Hexnuts



**Repeat
on both
sides**

► Focus: Building intake

Date: May 29, 2025

Starting Intake

14



X2 Bearings



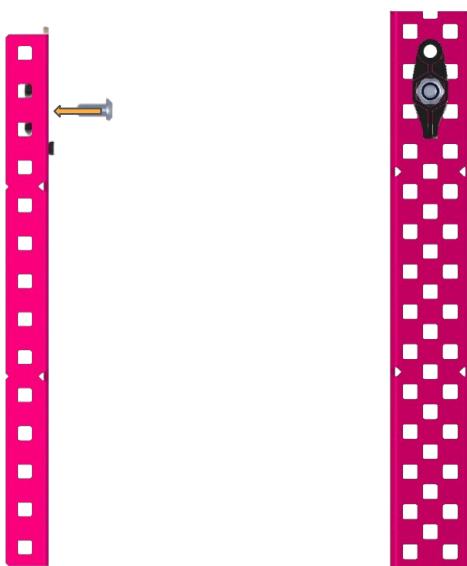
X2 Nylocks



X2 0.375" Screw



X2 14 Hole 1x2x1 C-Channel



15

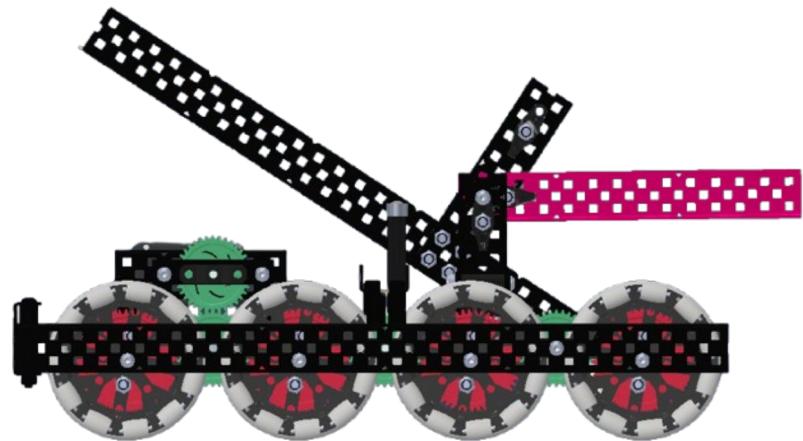
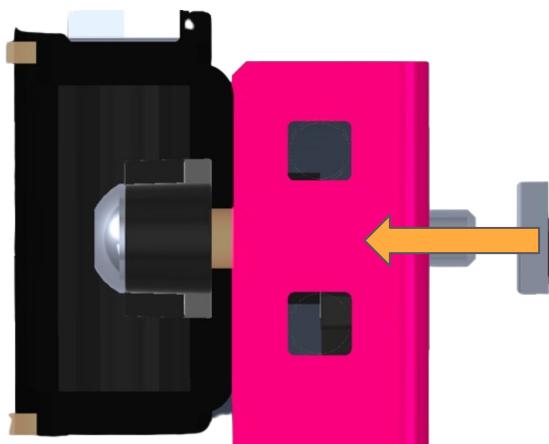


X2 Step 14



X2 Nylocks

X2 1/8" Spacer



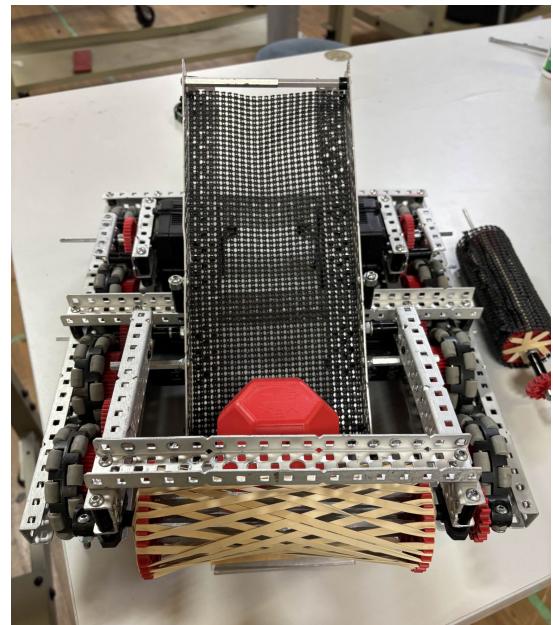
- Focus: Building Intake and Ramp

Date: May 30, 2025

Problems and Solutions

We found that polycarbonate does not have enough grip to help move a block. Blocks would often slip when in contact with plastic

To solve this issue, we added mesh for its grippy properties.



Next Steps

We want to continue building our intake structure and adding rollers.

Figure 1: Intake with mesh ramp

- ▶ **Problem:** How do we move blocks throughout our robot?

Date: June 1, 2025

Manipulating the Objects:

When designing a robot for Push Back, one of the key objectives is to be able to manipulate a block in order to score it. So we need to design mechanisms that allows us to move a block from the ground **throughout our robot** to score it.

This is a common trend throughout VEX games where a robot will transport the game object throughout the robot before it is scored.

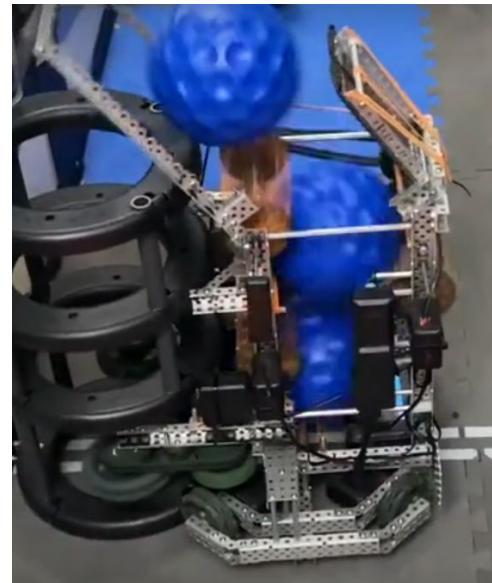


Figure 1: 80001B using rubber band rollers to transport the Change Up game object (YouTube)

- ▶ Focus: Manipulating blocks to move through our robot

Date: June 2, 2025

Transportation Methods

We brainstormed multiple different types of transportation methods in order to move blocks.

Horizontal Rollers

- Going back to when we **first brainstormed for our bottom stage intake**, we came up with various horizontal rollers such as flex wheel and rubber band rollers
 - These are simple to build and simple to implement

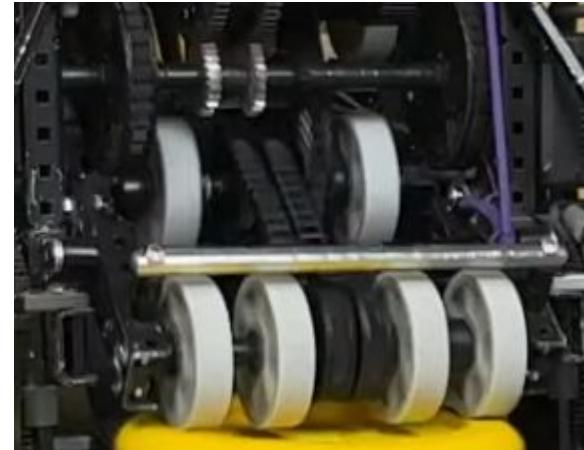


Figure 1: Flex wheel horizontal rollers (606X, YouTube)

Conveyor

- Conveyor belt that uses tank treads and flap pieces to help move the block through the robot



Figure 2: Vertical conveyor used in Tower Takeover (7K, YouTube)

Vertical rollers

- Vertical rollers could be put throughout our robot to help transport the block

- Focus: Selecting the design of our intake

Date: June 2, 2025

After brainstorming, we had three different options: Vertical Intake, Horizontal Rollers, and Conveyor.

To decide which intake to build, we ranked each in terms of the following criteria

- Build Simplicity** (1 - 5) - This criterion helps us make sure our intake is **not overly complicated**, such that we can build and perform maintenance on our intake with **ease**.
- Roller Friction** (1 - 5) - Friction matters because with each new axle or joint, the motors heat up faster, and our intake runs slower. **The less friction, the better**.
- Grip** (1 - 5) - The intake must be able to **grip onto the blocks**; It could slip down the intake, increasing the time it would take the block to go through our intake.
- Jamming** (1 - 5) - We want our intake to be **resistant to jamming**, so that it can keep intaking without stopping.
- Flexibility** (1 - 5) - This criterion is a bit difficult to describe; the block should still be able to move even when it pushes against the intake. The intake should therefore give way to the block.

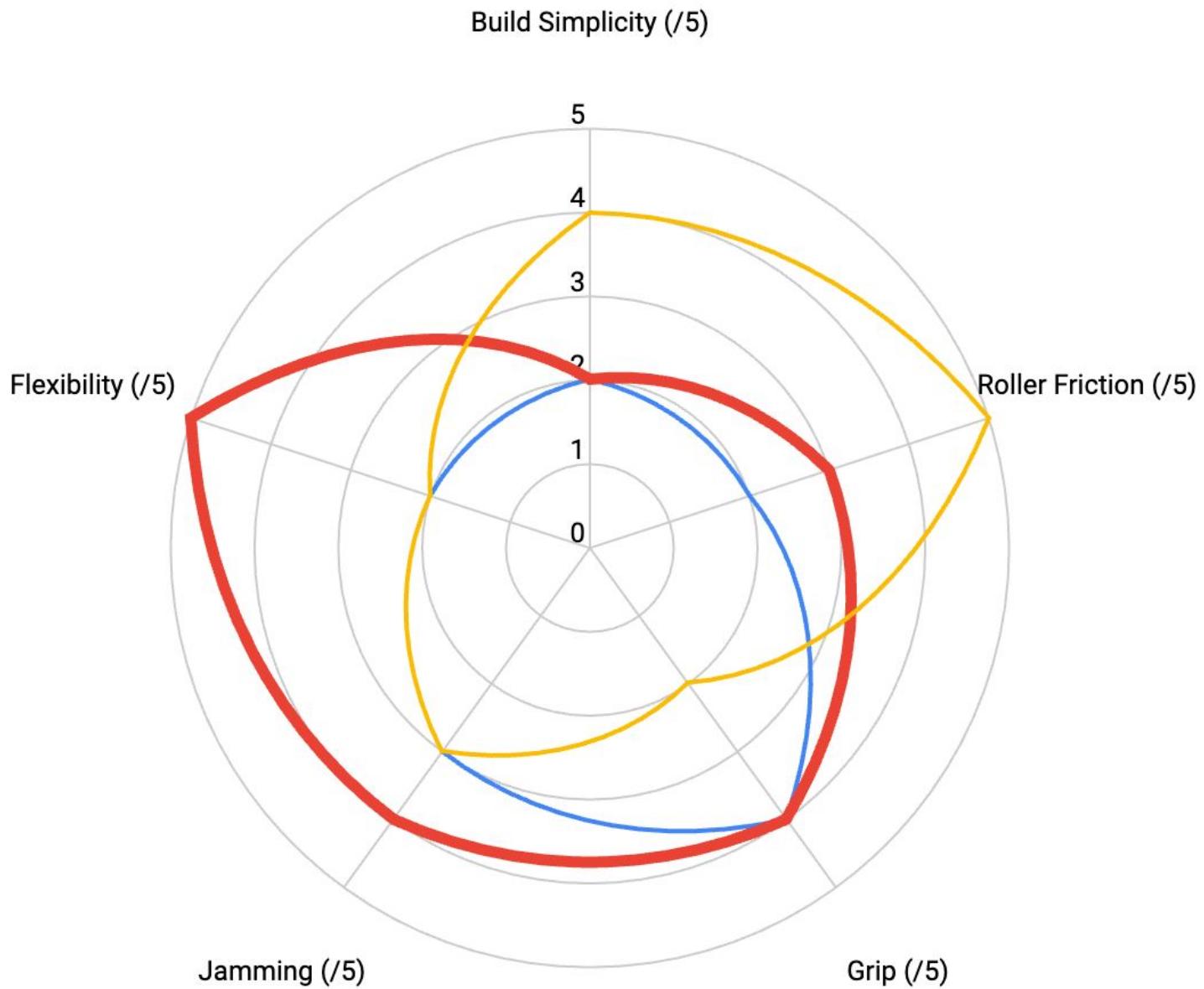
<u>Criteria</u> <u>Options</u>	Build Simplicity	Roller Friction	Grip	Jamming	Flexibility	Total
Vertical Intake	2	2	4	3	2	14
Horizontal Rollers	2	3	4	4	5	18
Conveyor	4	5	2	3	2	16

- Focus: Selecting the design of our intake

Date: June 2, 2025

Graph 1: V1 Intake Solution

— Vertical Intake — Horizontal Rollers — Conveyor



This is a radar chart generated from the table on the previous page. The horizontal roller is our chosen solution since it has the highest total, which corresponds to the largest area on the radar chart. Using a radar chart helps us visualize our decision.

- A reminder to take our ratings with a grain of salt; we are deciding on these criterion by past experience and speculation, not testing.

- ▶ Focus: Planning out our bottom stage intake

Date: June 7, 2025

Rollers

We plan to continue adding rollers up our intake ramp, allowing us to move blocks through our robot. These rollers need to be different sizes with different diameters to ensure no jamming or dead zones occur in our robot

Our rollers will be made of rubber bands wrapped around sprockets and will be powered by one 11W motor with a blue cartridge. This will result in a very fast intake spinning at 600 RPM. We plan to slow down our very first set of rollers to 400 RPM to ensure we can still score on the bottom goals.

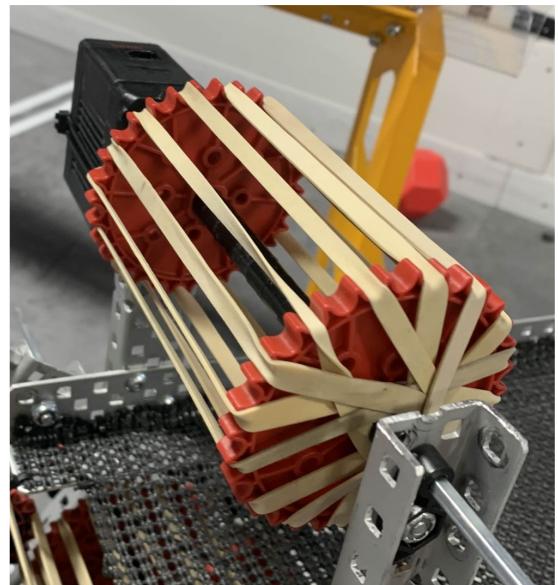


Figure 1: Rubber Band Roller

- ▶ Focus: Finishing our bottom intake system

Date: June 10, 2025

Members Involved

Joshua, Daniel

Objectives

Our goal is to create a system that utilizes different rollers and ramp elements that work together to move blocks through our robot.

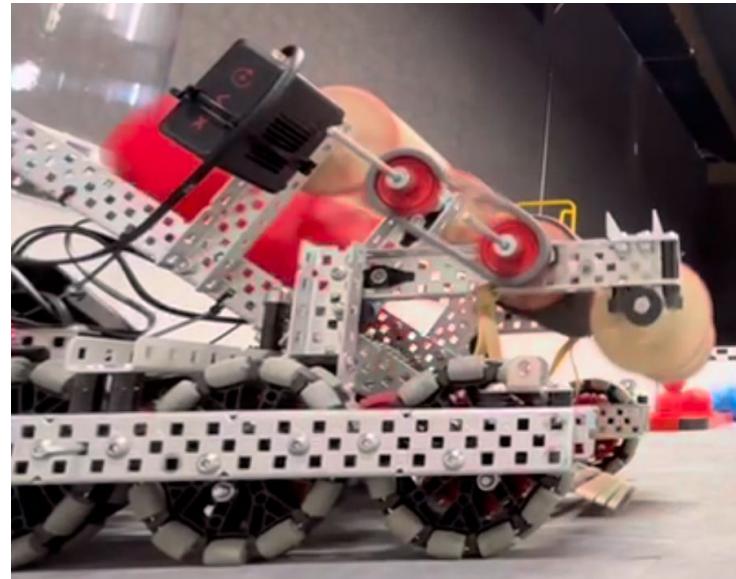


Figure 1: Our Intake System Moving a Ball

Materials

- Bearing flats
- Screws and nylocks
- High strength pillow bearings
- Various regular spacers
- High strength spacers
- Various standoffs
- 6, 32 teeth 6p sprockets
- 2, 24 teeth 6p sprockets
- 1, 17 hole 1x2x1 C-Channel
- 1, 10" long high strength shaft
- 2, 9 hole 1x2x1 C-Channel
- 2, 11 hole 1x2x1 C-Channel
- 1, 11W motor

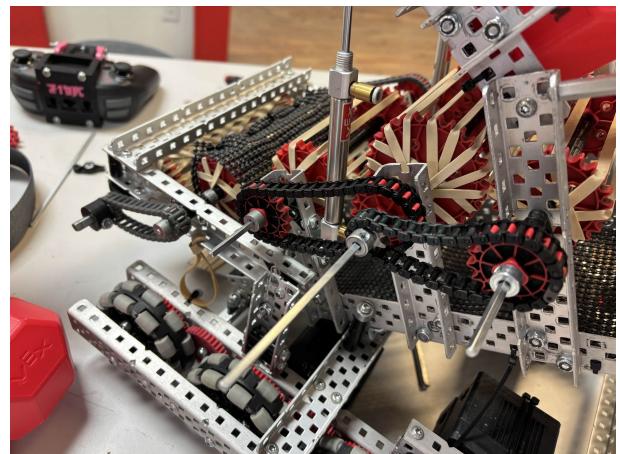


Figure 3: Intake System Chained Together

► Focus: Building intake

Date: June 10, 2025

Finishing Bottom

1



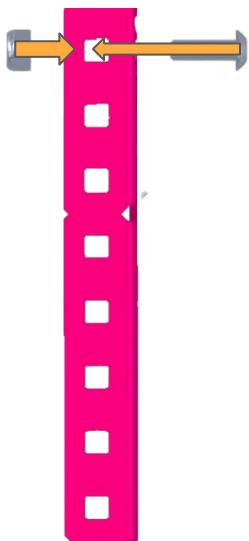
X2 Bearing Flats



X4 Nylocks



X4 0.375 Screw



**Repeat
on both
sides**

2



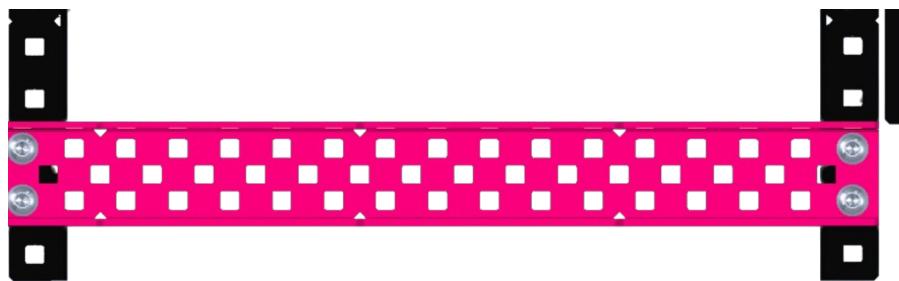
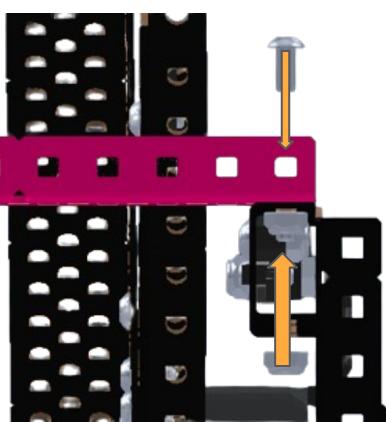
X1 17 Hole 1x2x1 C-Channel



X4 Nylocks



X4 0.375" Screw



► Focus: Building intake

Date: June 10, 2025

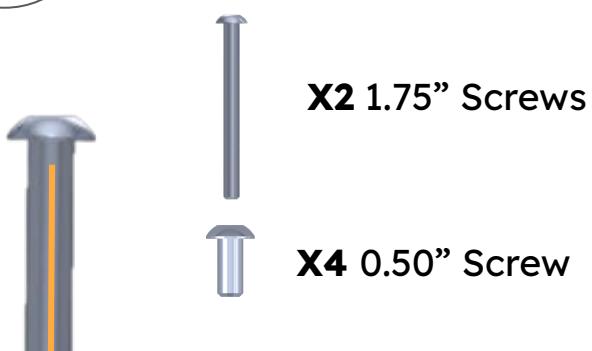
Starting Intake

3



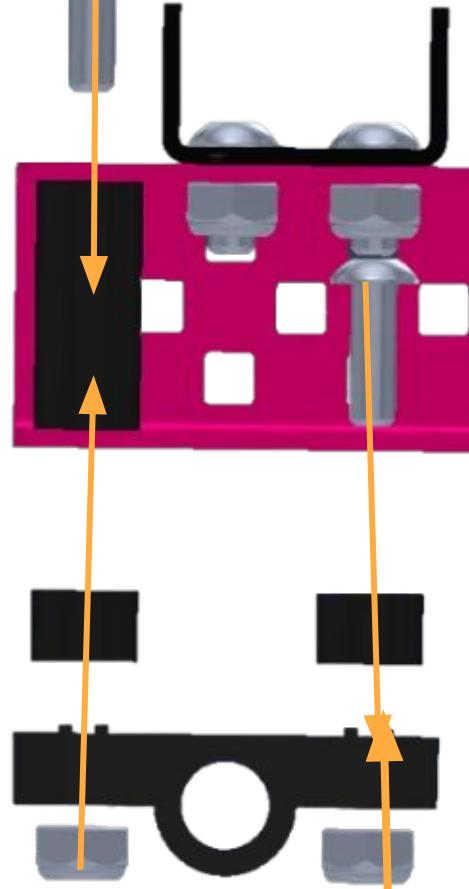
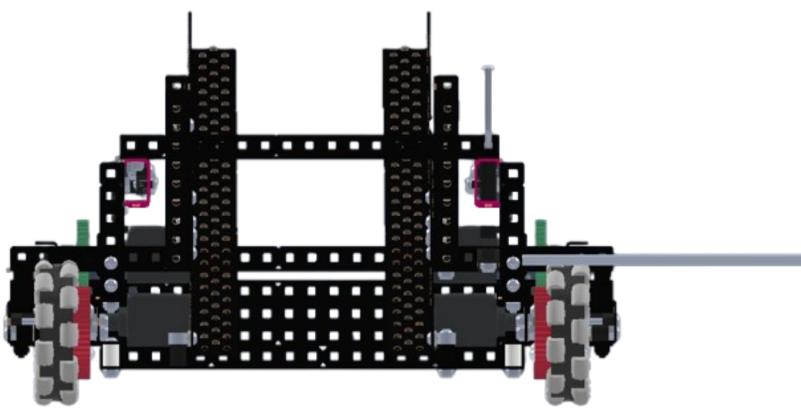
**X2 High Strength
Pillow bearing**

- X2 $\frac{3}{8}$ " Spacers**
- X2 $\frac{1}{2}$ " Spacers**
- X4 $\frac{1}{4}$ " Spacers**



4

X1 10" High Strength Axle



► Focus: Building intake

Date: June 10, 2025

Finishing Bottom

5



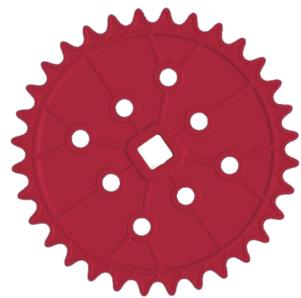
X4 6" Standoffs



X4 1" Standoffs

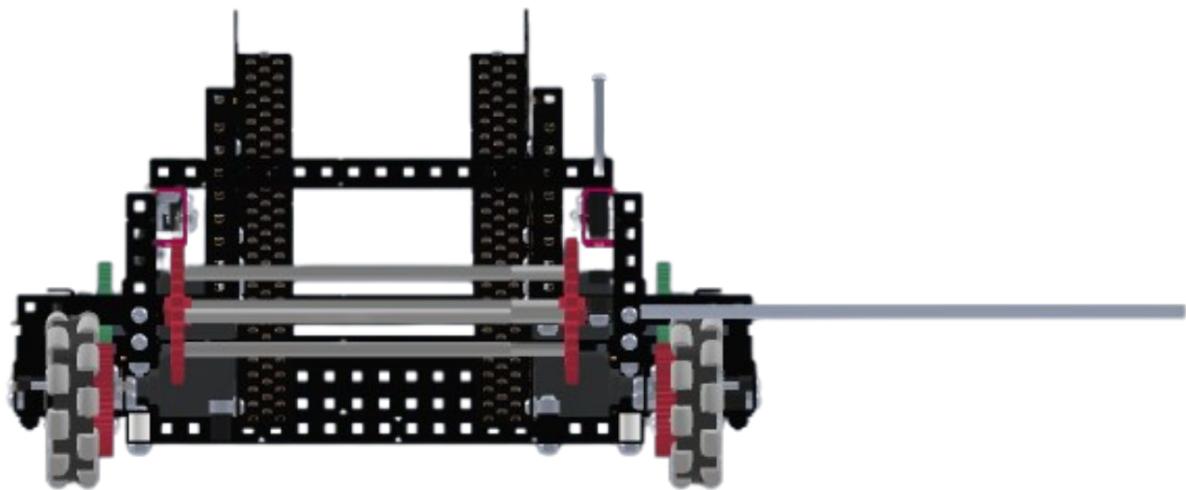
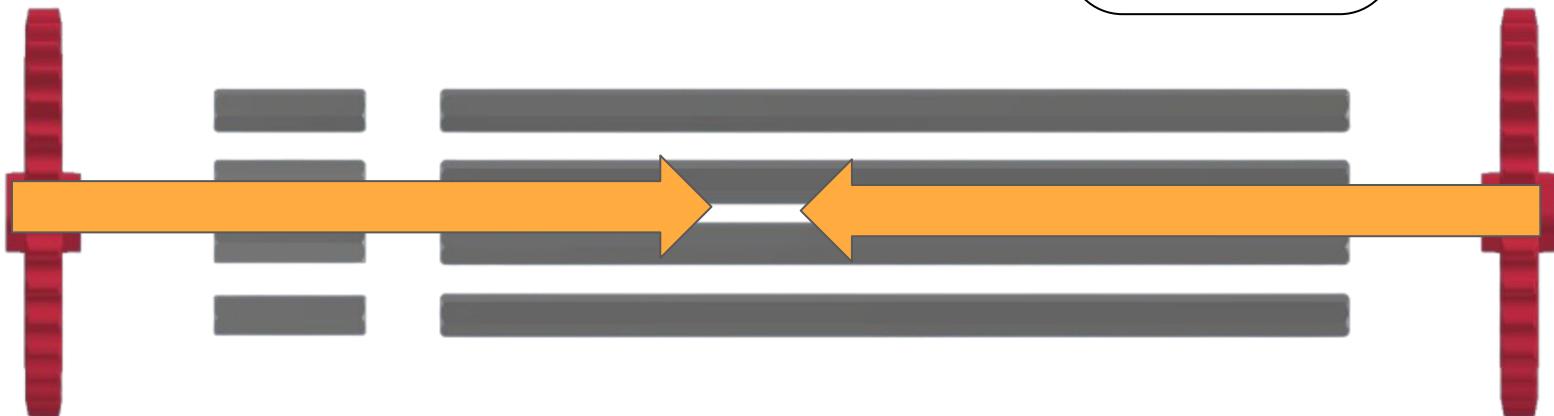


X4 0.375 Screw



X2 32 Teeth 6p Sprockets

x2

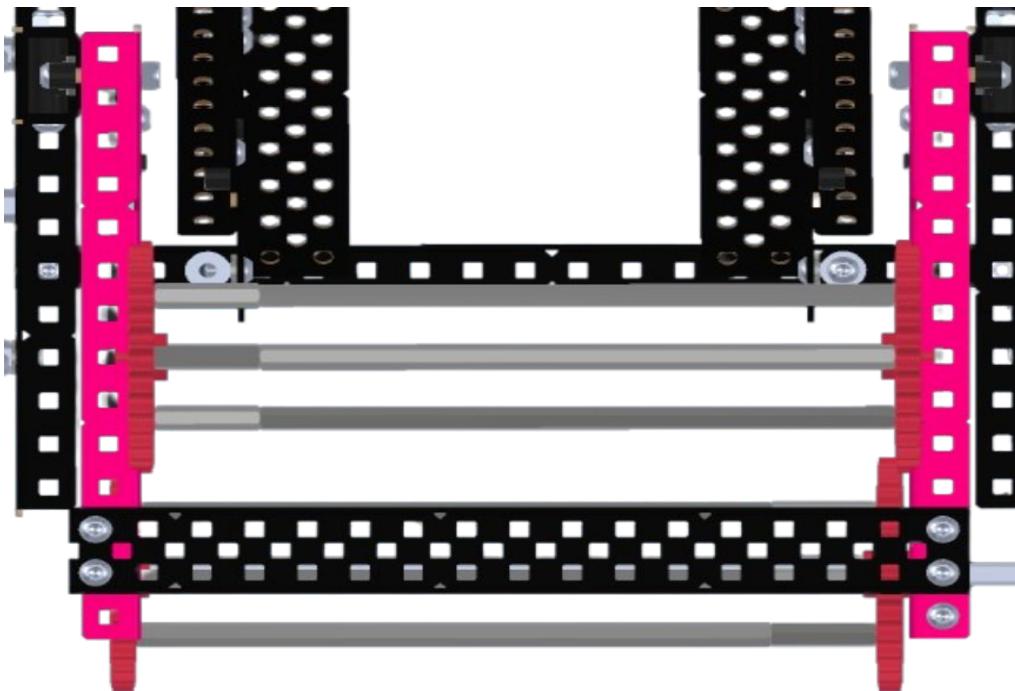


► Focus: Building intake

Date: June 10, 2025

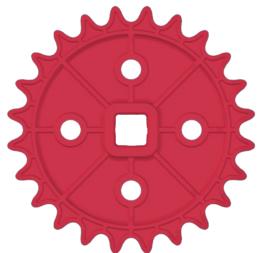
Finishing Bottom

6



Rubber Band the Sprockets

7



x2

x2 24 Tooth 6p Sprockets

x2 4" Standoffs



x4 0.375 Screw

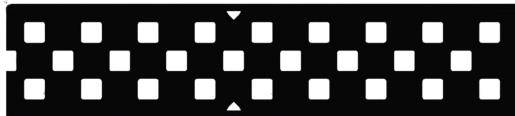


► Focus: Building intake

Date: June 10, 2025

Starting Intake

8

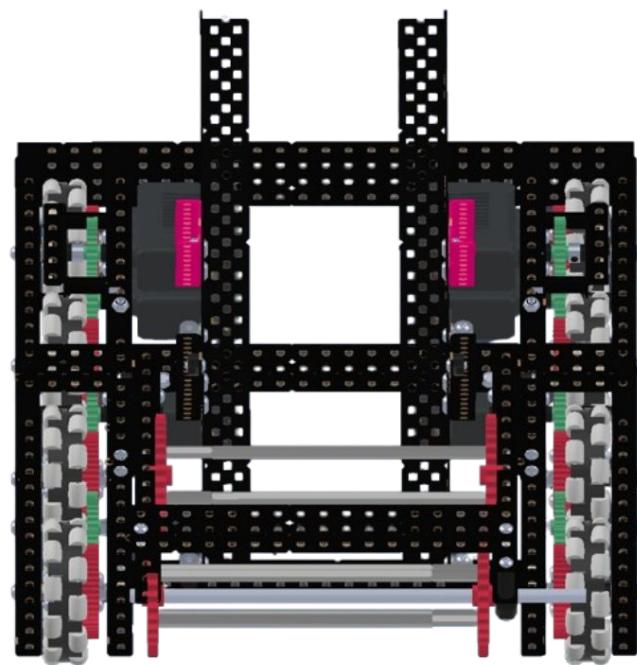
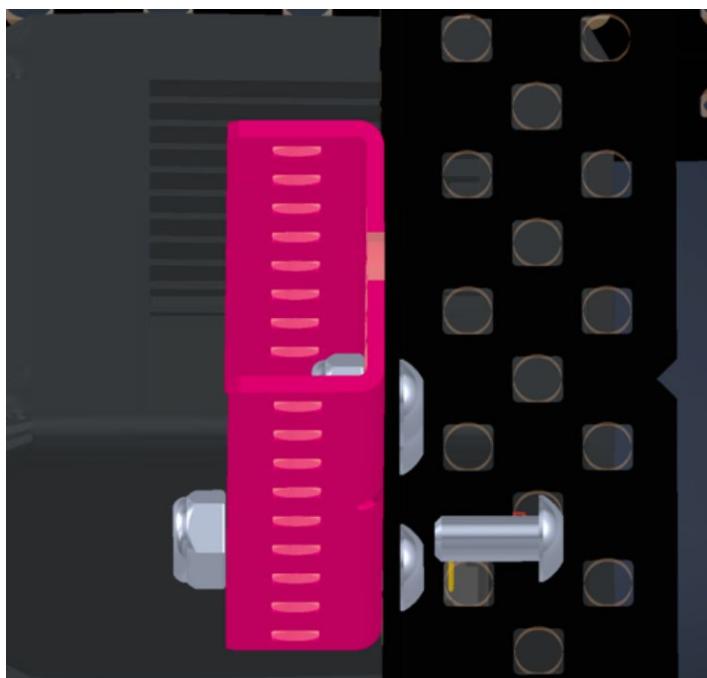


X8 0.375 Screw

X2 9 Hole 1x2x1 C-Channel



X8 Nylocks



9



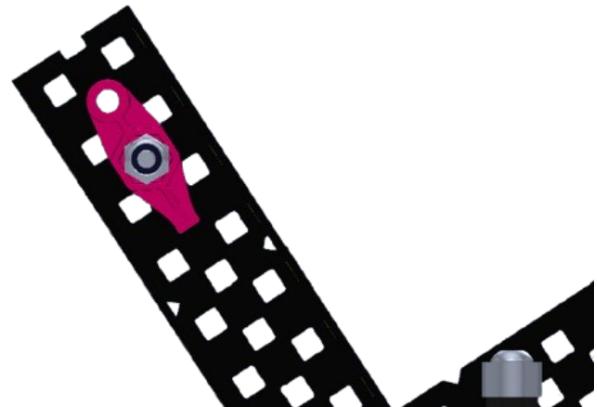
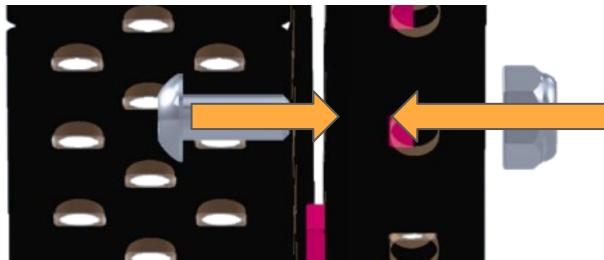
X1 Bearing



X1 Nylocks



X1 0.375 Screw



► Focus: Building intake

Date: June 10, 2025

Starting Intake

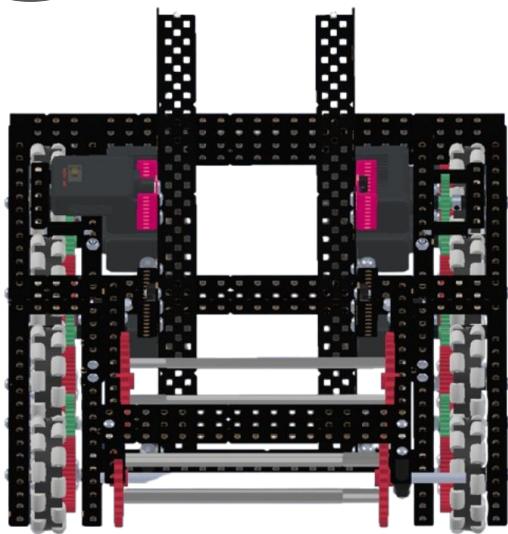
10



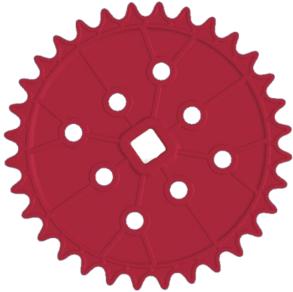
X2 0.375 Screw



X1 11W Motor



11



X2 32 Teeth 6p Sprockets



X4 0.375 Screw



X2 4" Standoffs

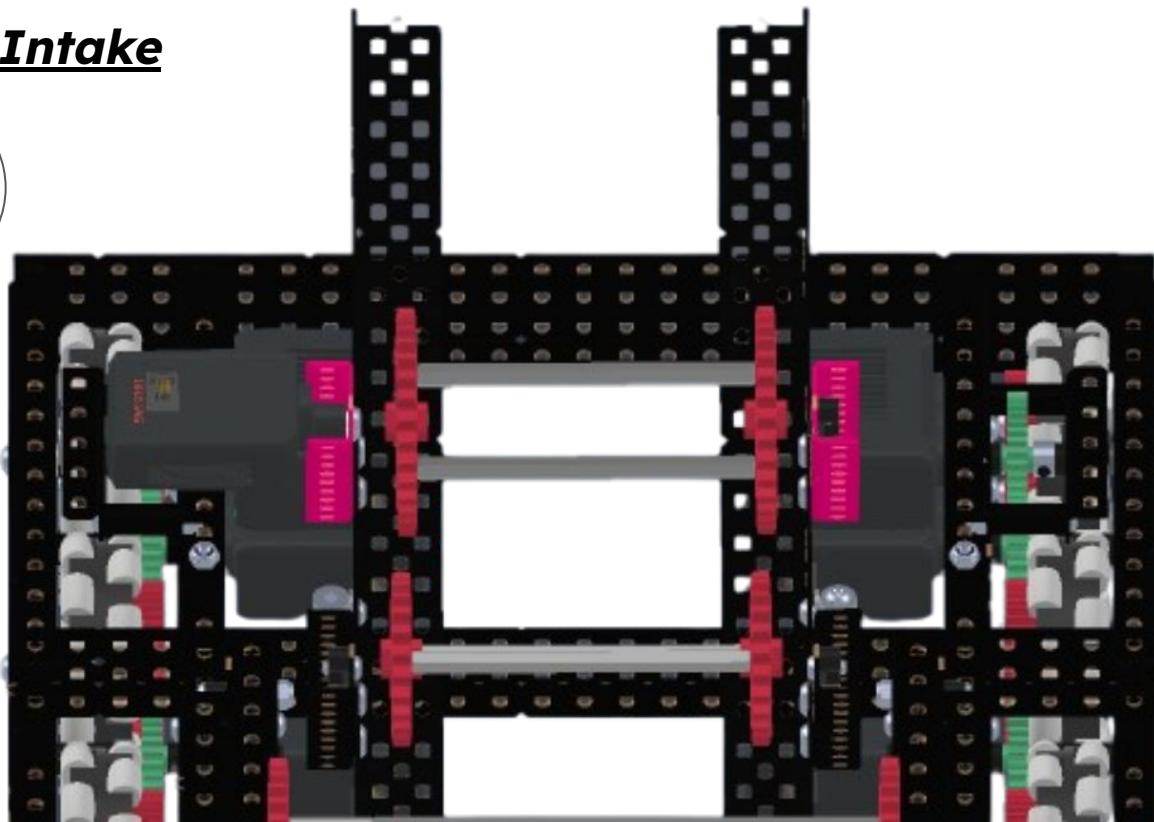


► Focus: Building intake

Date: June 10, 2025

Starting Intake

12



13



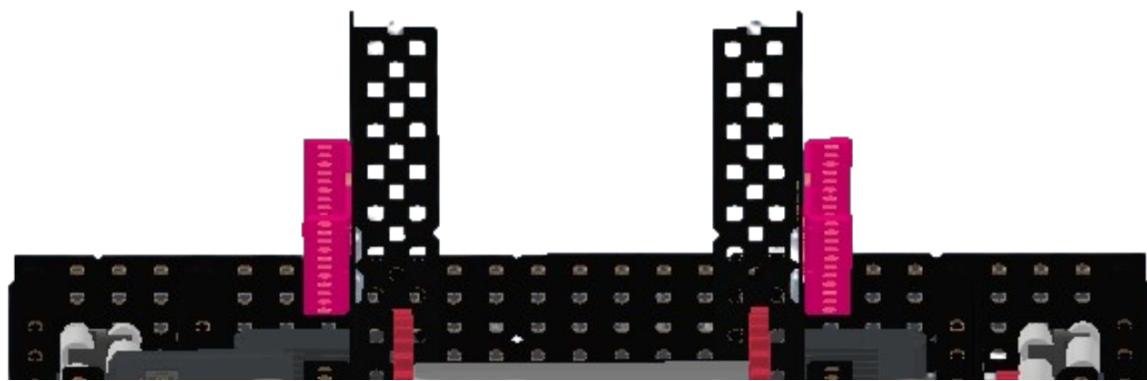
X2 11 Hole 1x2x1 C-Channel;



X8 0.375 Screw



X8 Nylocks



► Focus: Building intake

Date: June 10, 2025

Starting Intake

14



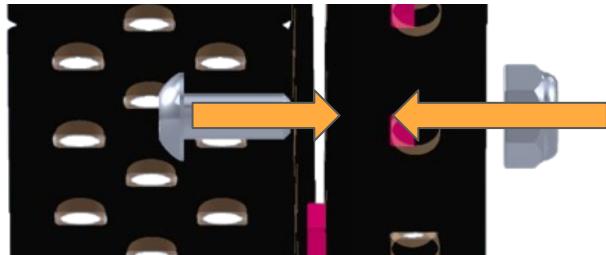
X1 Bearing



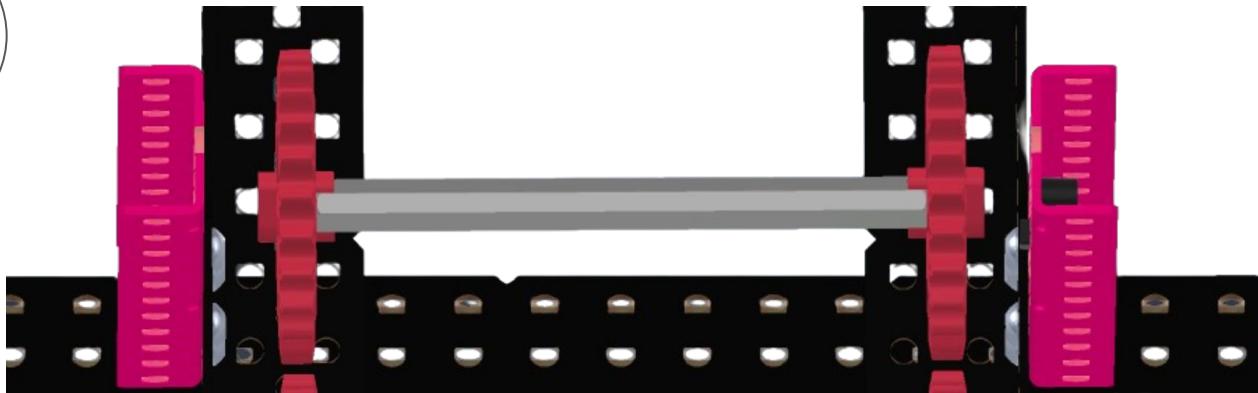
X1 Nylocks



X1 0.375 Screw



15



Bottom Stage of the intake is complete

- Focus: Building Intake

Date: June 10, 2025

Problems and Solutions

We needed to find a way to align our metal in order to prevent it from bending our axles and adding unwanted friction to the system

In order to solve this issue, we utilized temporary bracing with shoulder screws to make sure everything lined up properly.

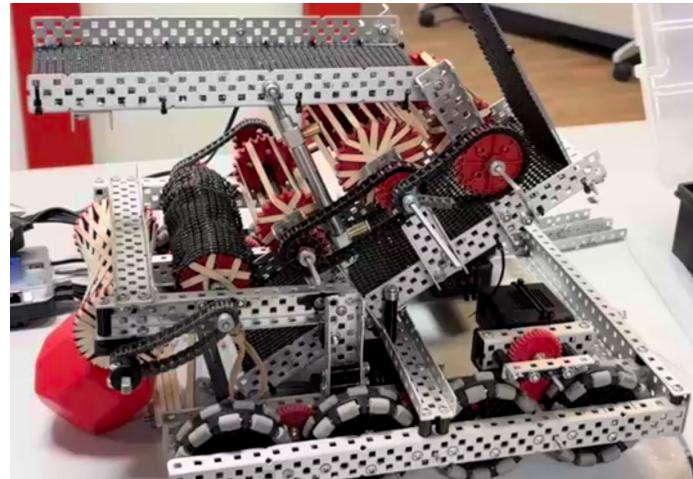


Figure 2: Side View of Our Intake System

Next Steps

We will move onto planning our transition stage which will allow us to move blocks to a different elevation and direction, allowing us to score.

Version 1.0: Mall of America Signature Event

Top Stage/ Scoring

Scoring on Multiple Heights

Identify Problem

- ▶ Problem: How do we score on 3 different goal heights?

Date: June 15, 2025

Scoring on multiple heights

When looking at field specifications, one of the key challenges this year is designing a robot that is able to score on all **three elevations of goals**.

The two long goals are **12.67 inches** off the ground. The upper central goal is **8.01 inches** off the ground. The lower central goal is **0.50 inches** off the ground.

Being able to score on **all levels** of goals opens up more opportunities and potential scoring in a match, as well as **maximizing** our potential skills score.

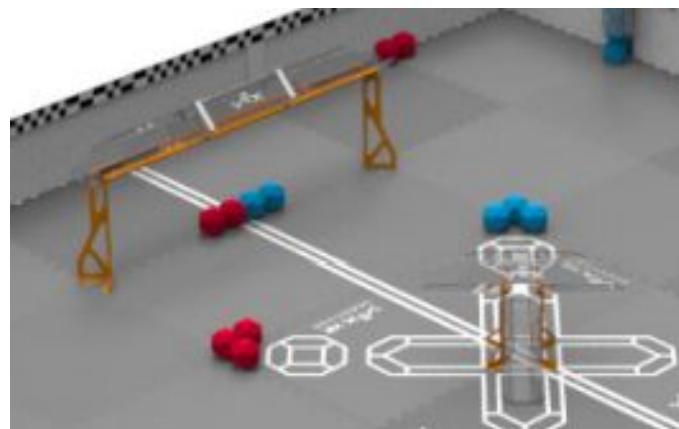


Figure 1: Center Goals (VEX Game Manual)

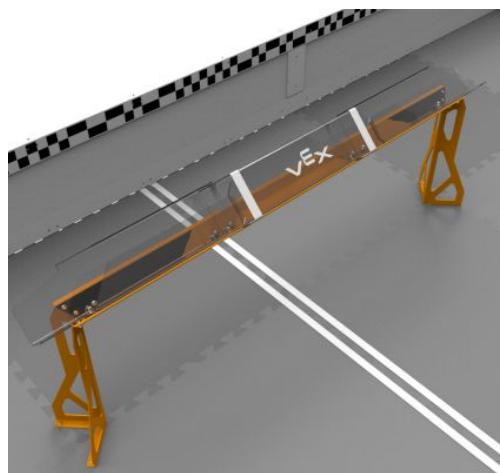


Figure 2: Long Goal (VEX Game Manual)

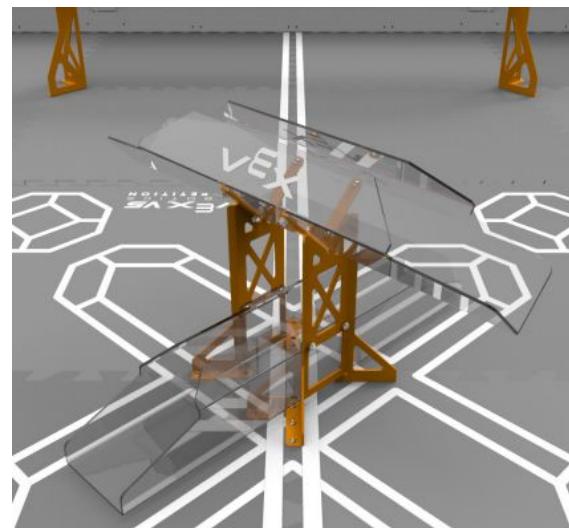


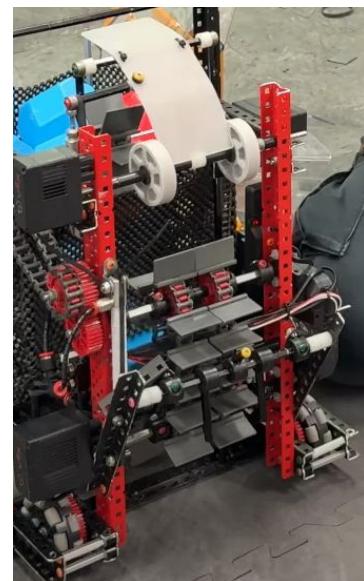
Figure 3: View of all goals (VEX Game Manual)

- ▶ Focus: Looking at multiple designs

Date: June 17, 2025

Research:

We utilize **online resources** to help with **brainstorming** different parts of our robot. Looking through different online resources such as the **Vex Forums** and the **Vex CAD discord server**, we have found various published designs that can score on the different **goal heights**.



Motor System

We first looked at a design created by **4886S** utilizes **different motors spinning in different directions at different times** to be able to score blocks in different goals. They also utilize a **piston trapdoor** at the top of their robot to **prevent** blocks from travelling in that direction.

Figure 1: 4886S's Robot (Vex CAD)

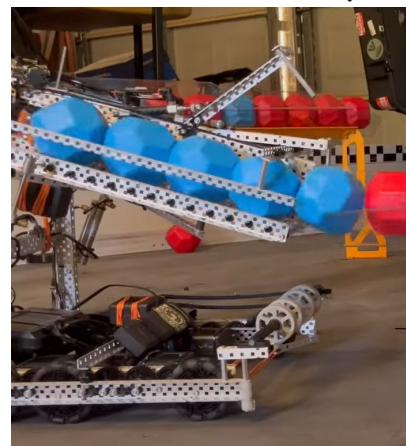


Figure 2: 920B scoring on the center goals

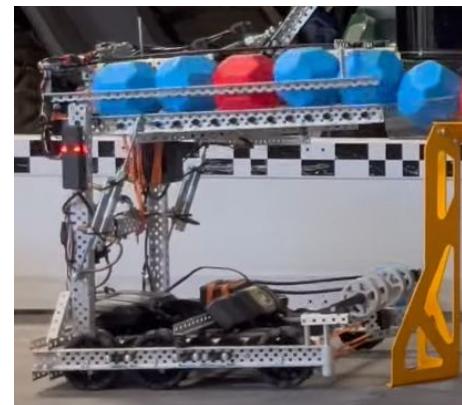


Figure 2: 920B using a lift to score on the long goals

Scoring on Multiple Heights

Select Solution

- ▶ Focus: Selecting a design so we can score on different heights

Date: June 17, 2025

After brainstorming, we came up with three different methods of scoring on multiple heights; piston lift, motor lift and a motor system. We will be ranking these designs using the following criteria:

1. **Weight (1 - 5)** - Ranked on how much weight would this design add on to our robot
2. **Ease of Implementation (1 - 5)** - Ranked on how easily we can build and implement this design onto our current base
3. **Compactness (1 - 5)** - Ranked on how little space the design would take
4. **Speed (1 - 5)** - Ranked on how fast we can switch between the different goal heights

<u>Criteria</u> <u>Options</u>	Weight	Ease of implementation	Compactness	Speed	Total
Piston Lift	4	5	4	5	18
Motor Lift	2	3	3	3	11
Motor System	3	4	2	4	14

After consideration, we decided to choose the piston lift because it is the easiest to implement onto our current robot. The piston lift is also very simple and to build, saving us time in planning out what to build.

- ▶ Focus: Planning our piston lift

Date: June 18, 2025

Planning Piston Lift

We wanted to keep a **simple design** and sticking to **basic building concepts** that we were familiar with. We decided to use a **piston lift design**, using **two pistons** that will **extend and retract** to determine which goal we want to **score on**. Two pistons will also use less air, allowing us to actuate our pneumatics more types during a match.

In our design, one end of the piston is attached to the **top stage** of our intake and the other end is attached to the **lower stage** of the intake. This provides a strong **mechanical advantage** to lift our top stage as well as a fast and reliable way to select which goal we want to score on.

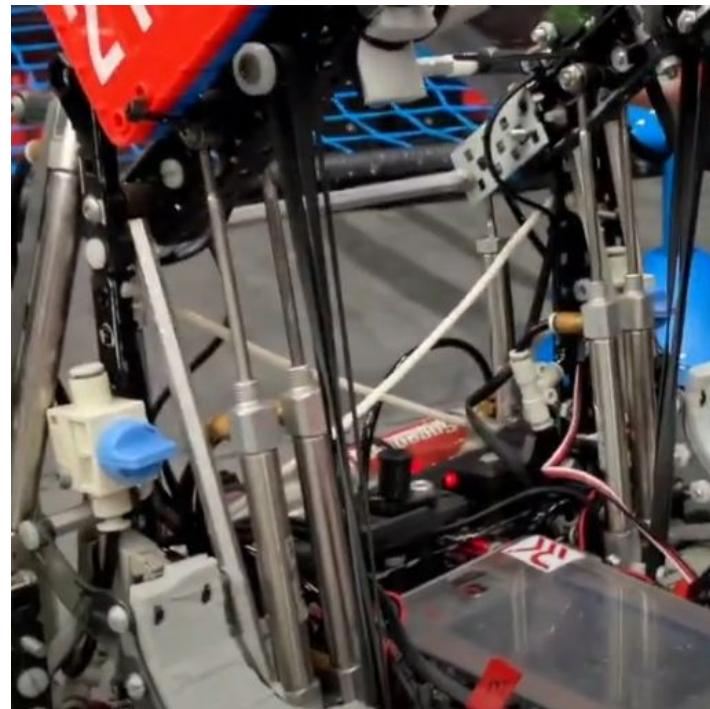


Figure 1: An example of a 4 piston lift on a C-tier hang (Over Under, 21417A, YouTube)

- ▶ Focus: Building a piston lift

Date: June 21, 2025

Members Involved

Joshua, Daniel

Objective

Create a base for our top stage intake as well as attaching our piston lift.

Materials:

- 2, 18 Hole 2x2 L-Channels
- 2 Medium Length Pistons
- 6 Keps Nuts
- 4 bearing flats
- 6 Nylocks
- 2 0.75" Screws
- 4 1" screws
- 2 Axle Collars
- 2 1/8" Spacers
- Mesh
- 2, 1 hole 1x2x1 C-Channel
- 2, 10 Hole 1x1 C-Channel

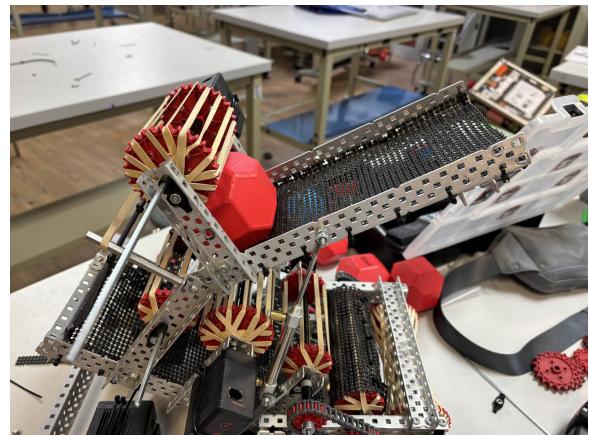


Figure 1: Our piston lift in the extended position



Figure 2: Our piston lift in the retracted position



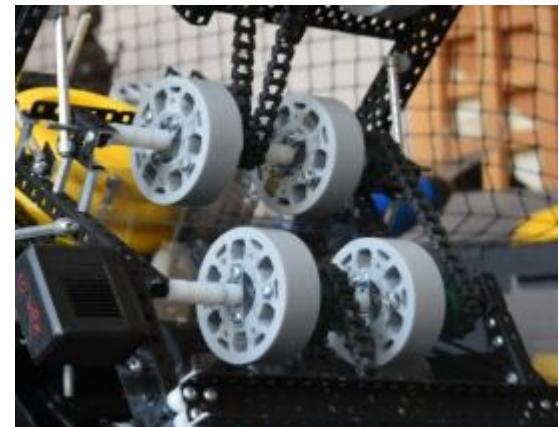
Figure 3: Side view of our piston in the extended position

- ▶ Focus: Planning our top stage rollers so we can score

Date: June 23, 2025

Scoring

In order to continue moving the block through our robot, we plan to **continue our intake** system by building more rollers. These rollers will be created using previous methods of sprockets and rubber bands. Our roller system will **facilitate our scoring** as the block will continue travelling into the scoring tubes.



*Figure 1: Spin Up Rollers
(10012W, YouTube)*

- ▶ Focus: Scoring with rollers

Date: June 25, 2025

Members Involved

Daniel, Bryan

Objective

Continue building rollers until the block can be scored into the tubes

Materials

- 2, 8 1x2x1 hole C-channel
- 4, 9 1x2x1 hole C-channel
- 4, sprockets (HS)
- 2, sprockets (IQ)
- 2, IQ sprockets (6P)
- More sprockets (need sizing)
- 5 bearing flats
- Nylocks
- Shoulder screws of different sizes
- 3 axles
- Rubber bands

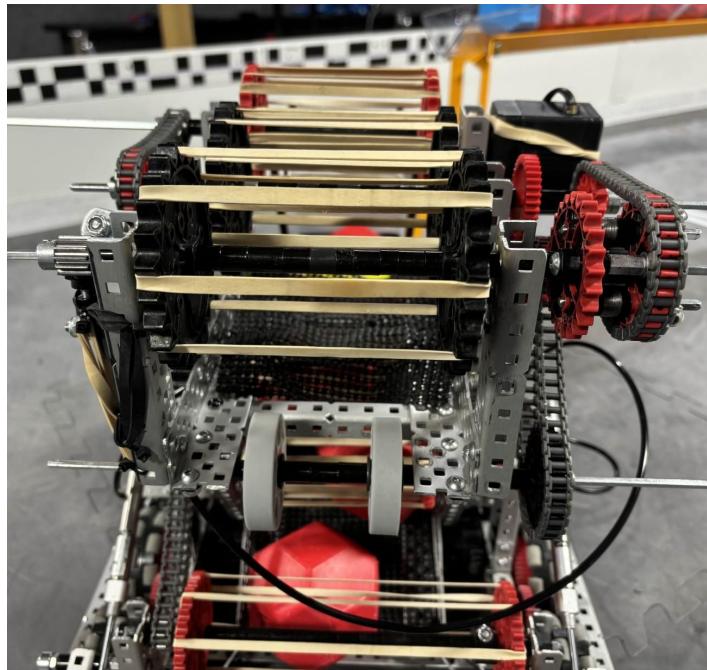


Figure 1: Intake upper stage

► Focus: Building the base for our top stage intake

Date: June 29, 2025

1



X2 18 Hole 2x2 Angle Bar



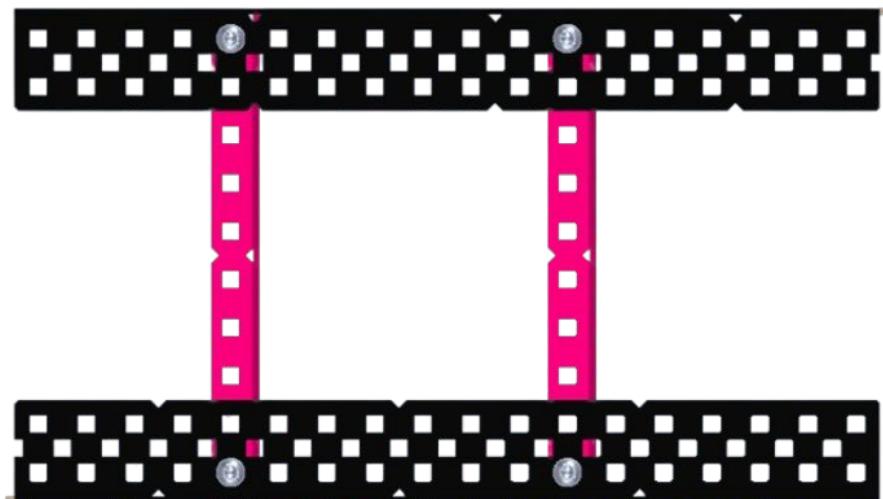
X2 10 Hole 1x1 Angle Bar



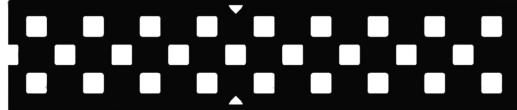
X4 0.375 Screw



X4 Nylocks



2



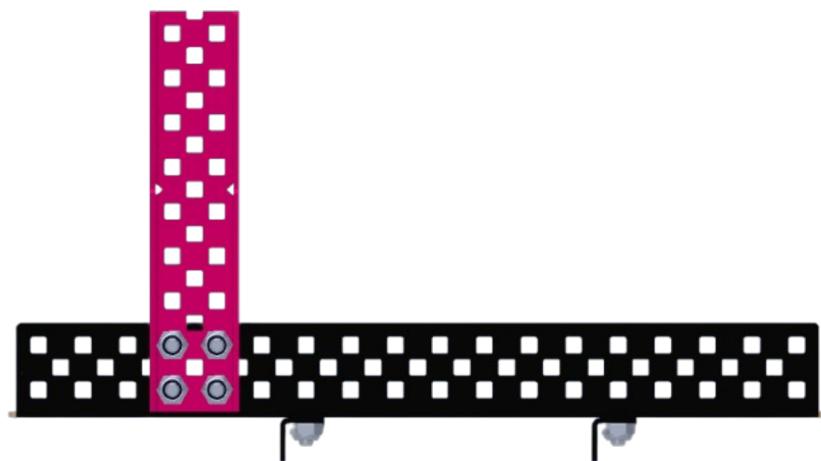
X2 9 Hole 1x2x1 C-Channel



X8 0.375 Screw



X8 Nylocks



Top Stage Intake

Build

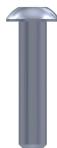
► Focus: Building intake

Date: June 29, 2025

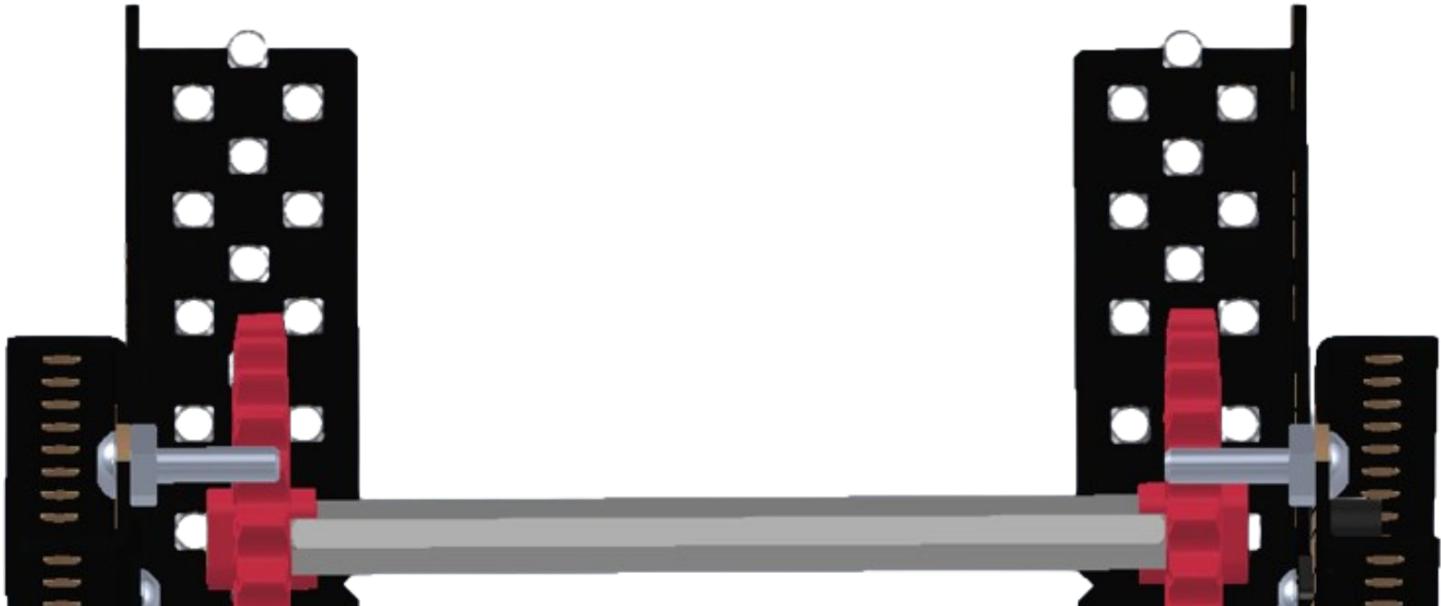
3



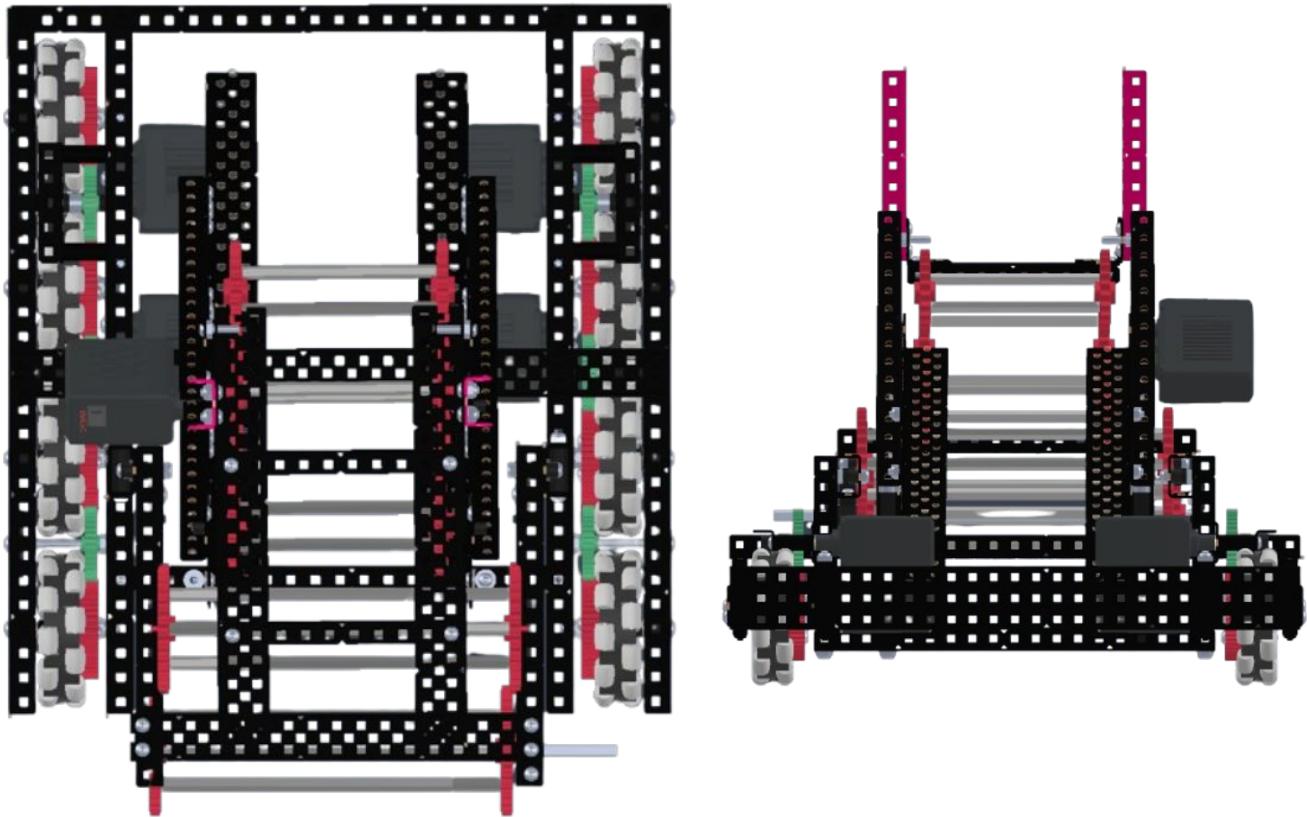
X1 Hexnuts



X2 0.75" Screws



4



► Focus: Building intake

Date: June 30, 2025

Starting Intake

5



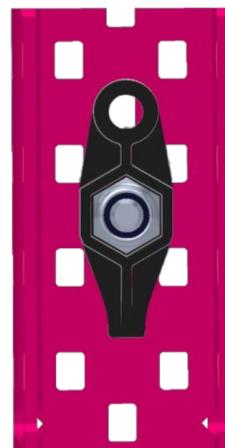
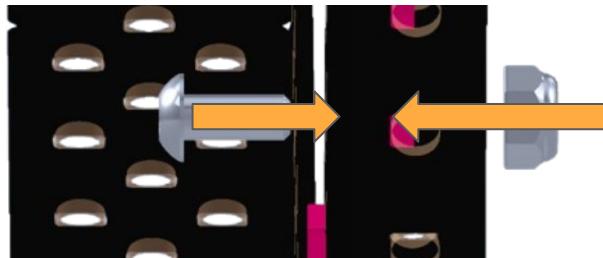
X3 Bearing



X3 Nylocks



X3 0.375 Screw



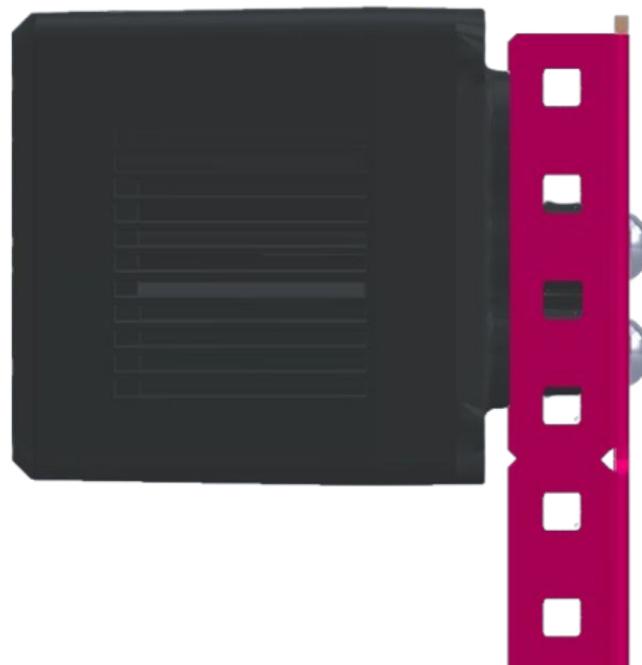
6



X2 0.375 Screw



X1 11W Motor



► Focus: Building intake

Date: June 30, 2025

7



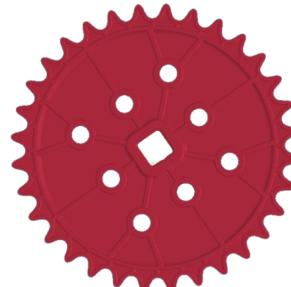
X2 3" Standoff



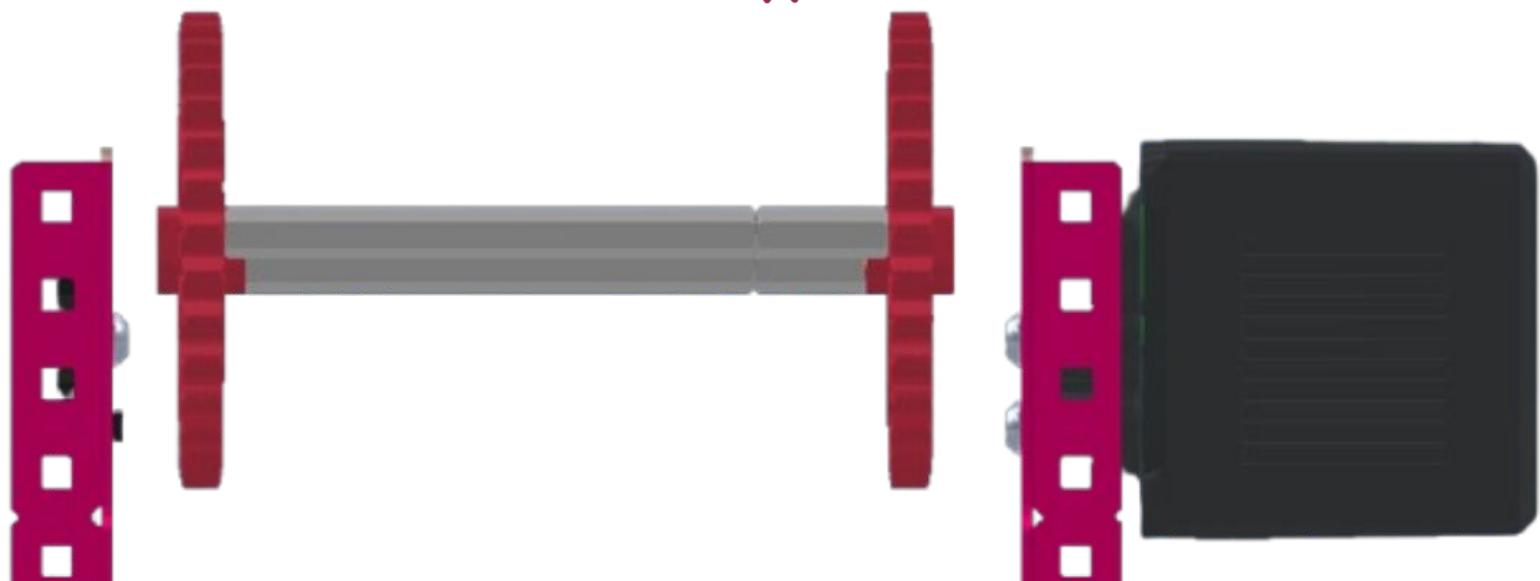
X4 0.375 Screw



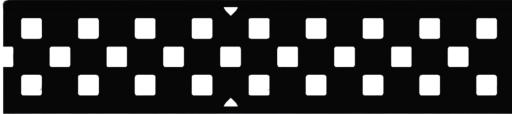
X2 0.75" Standoff



X2 32 Teeth 6p Sprockets



8



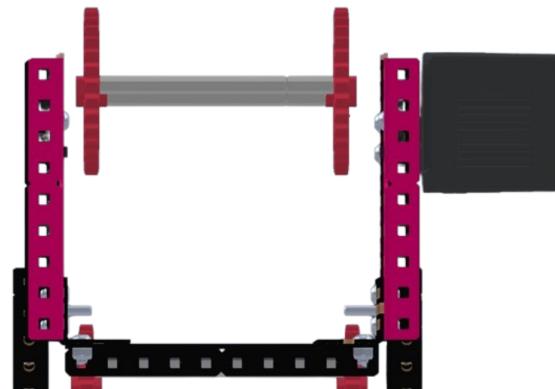
X2 9 Hole 1x2x1 C-Channel



X8 0.375 Screw



X8 Nylocks



► Focus: Building intake

Date: June 30, 2025

9



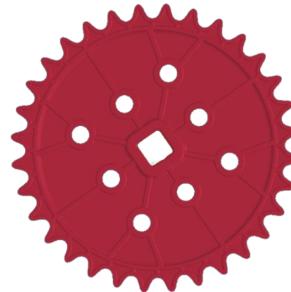
X2 3" Standoff



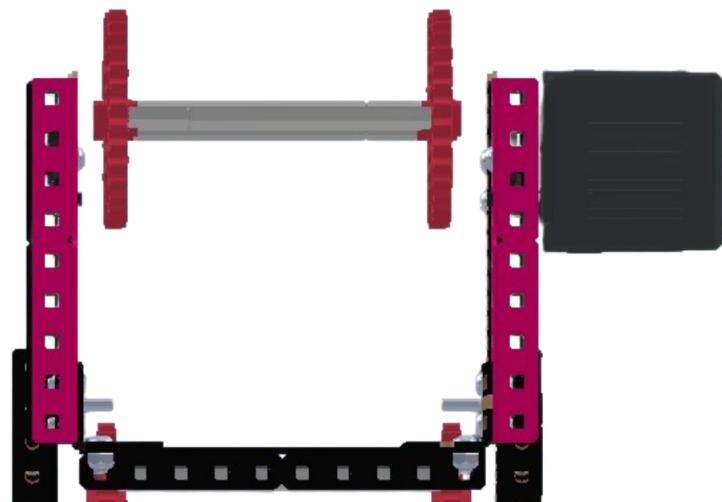
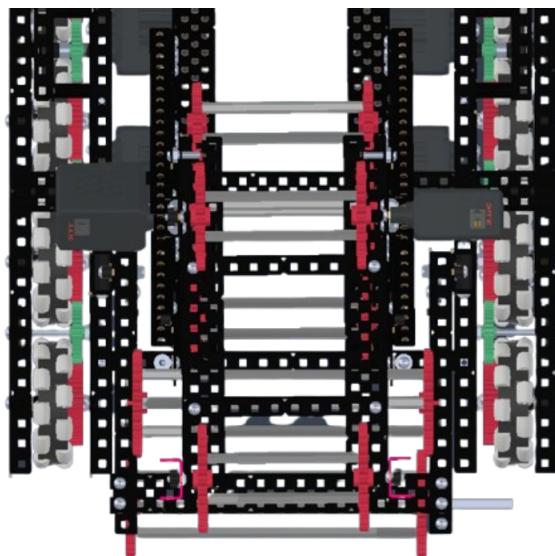
X4 0.375 Screw



X2 0.75" Standoff



X2 32 Teeth 6p Sprockets



10



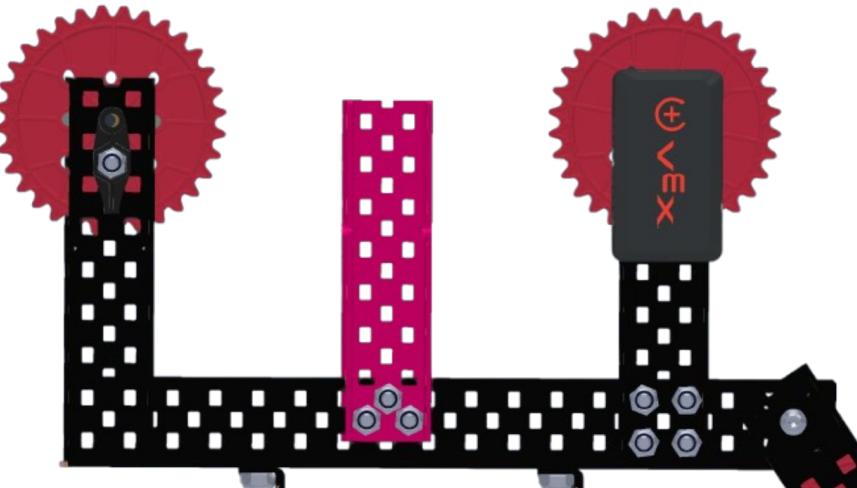
X2 8 Hole 1x2x1 C-Channel



X6 0.375 Screw



X6 Nylocks



► Focus: Building intake

Date: June 30, 2025

Starting Intake

11



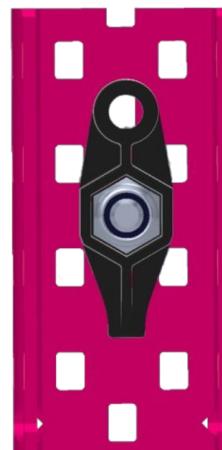
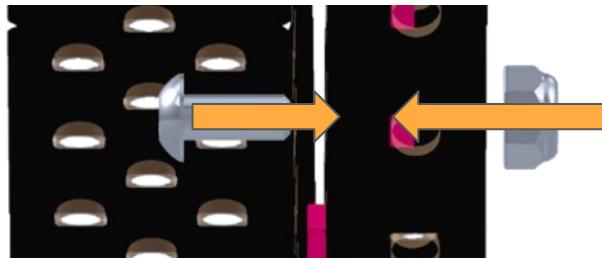
X3 Bearings



X3 Nylocks



X3 0.375 Screw



12



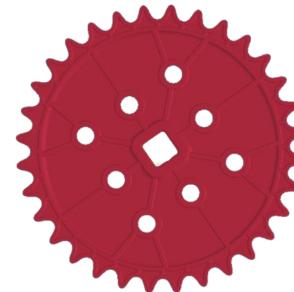
X2 3" Standoff



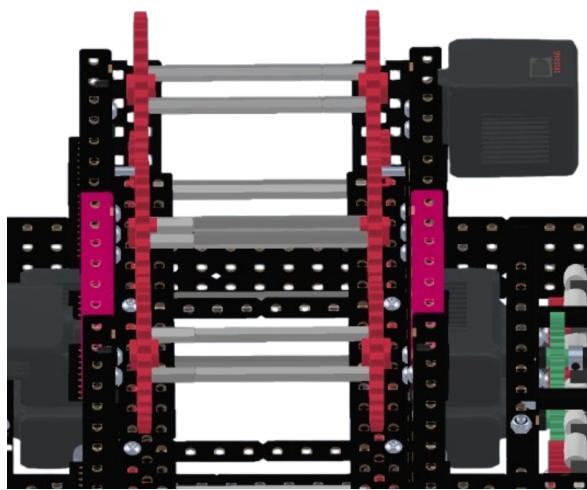
X2 0.75" Standoff



X4 0.375 Screw



X2 32 Teeth 6p Sprockets



**We've finished
the V1 intake
build!**

- Focus: Building Intake

Date: June 30, 2025

Problems and Solutions

When we first tested if our top stage could score, we found that 600 RPM did not have enough power to push blocks out the other end of the tube.

We solved this issue by slowing down our final set of rollers to 400 RPM for more consistent scoring and more torque

We found that C-channels would bend if the chain was too tight

Solve this issue by adding bracing and loosening the chain

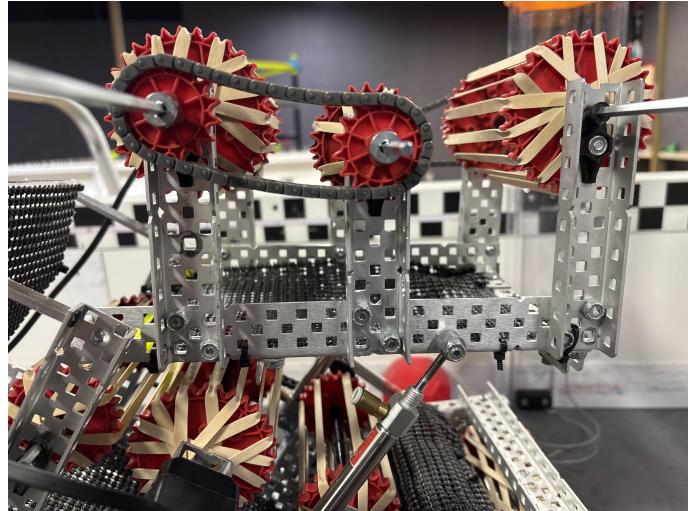


Figure 1: Top Stage Rollers

Next Steps

Creating a transition stage so the blocks can redirect themselves into our top stage intake.

- ▶ Focus: How do we move a block from the bottom to top stage of the intake

Date: June 30, 2025

Looking at past games

When deciding methods to move a block through a directional change, a few ideas came to mind. We looked at previous games such as **VRC Spin Up** and **VRC Change Up** to find inspiration as many robots implemented something to help change the direction of the game object that is moving.

Change Up

Looking at a concept CAD from **41998C**, we can see that they utilized **tank treads** and polycarbonate as a vertical ramp to move the change up game elements up through their robot. We could incorporate rollers into this method of transporting blocks.

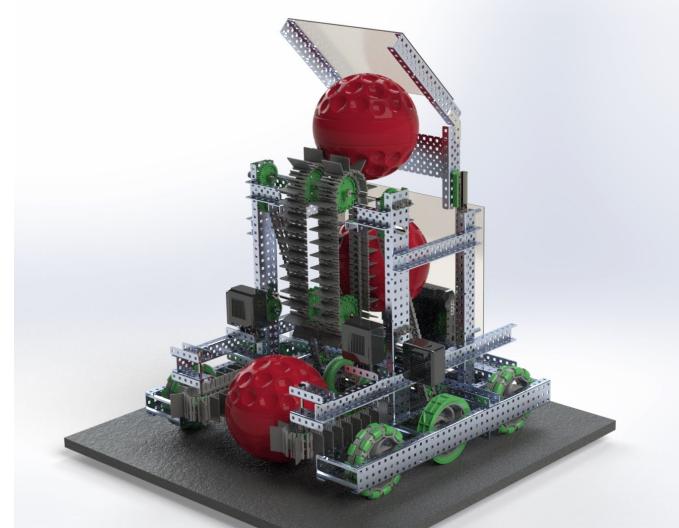


Figure 1: Team 41998C's Change Up Concept Cad (GrabCad)

Spin Up

Looking at **4082B**'s Spin Up states reveal, we saw they used curved polycarbonate pieces and **flex wheel rollers** to help move discs through their robot. Instead of using flex wheels, we could implement **mesh or rubber band rollers** in place of them.

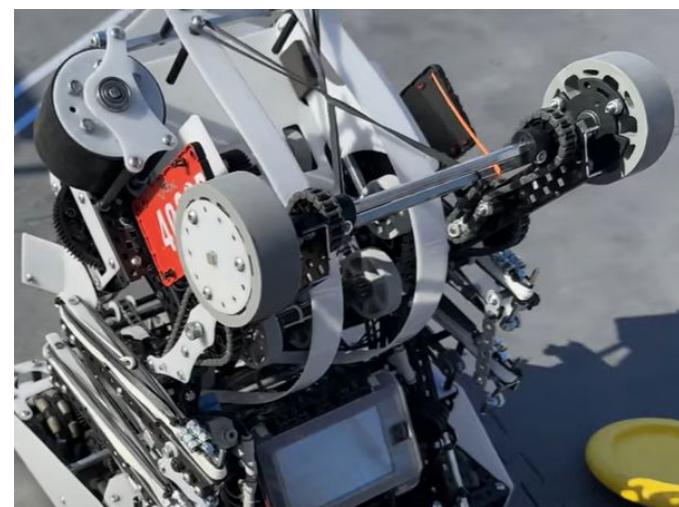


Figure 2: Team 4082B's Spin Up States Robot (YouTube)

- ▶ Focus: Planning a method for transferring blocks

Date: July 1, 2025

Polycarbonate For Block Transfer

After looking at previous games for inspiration, we have come up with our **own version** to help move blocks between our bottom and top stages of intake. We will use a piece of **polycarbonate** that will be spaced to allow for the block to fit between the plastic and a set of rollers. This allows the block to move up through our robot. We plan to use mesh to provide grip and help prevent the block from slipping.



Figure 1: Example of a curved polycarbonate transfer stage (34203X, Change Up, Vex Forums)

- ▶ Focus: Our own version of transferring blocks

Date: July 1, 2025

Members Involved

Daniel, Joshua

Objectives

Create a polycarbonate stage that will help with transferring blocks from our bottom stage to our top stage intake.

Materials

- Polycarbonate
- Mesh
- Screws
- Spacers
- Zip Ties

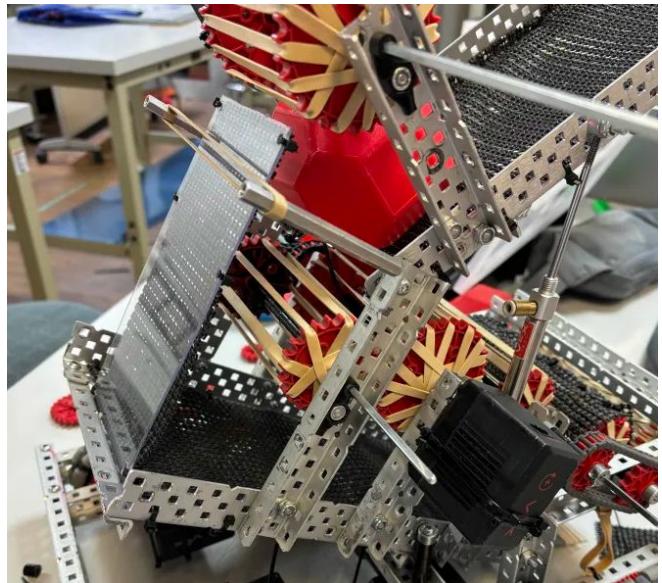


Figure 1: A view of our polycarbonate used for transferring blocks



Figure 2: Side view of our polycarbonate

- Focus: Our own version of transferring blocks

Date: July 1, 2025

Problems and Solutions

We found that the block would get stuck due to a large amount of pressure pushing it against the rollers.

We solved this by spacing out our polycarbonate, allowing there to be more room for the block to move. This also decreased the amount of pressure pushing against the blocks.

The mesh provided too much grip due to its large surface area. This caused the block to get stuck and cause jams due to the blocks slowing down.

Switching to rubber bands as they also provide grip, but they do not have as much surface area

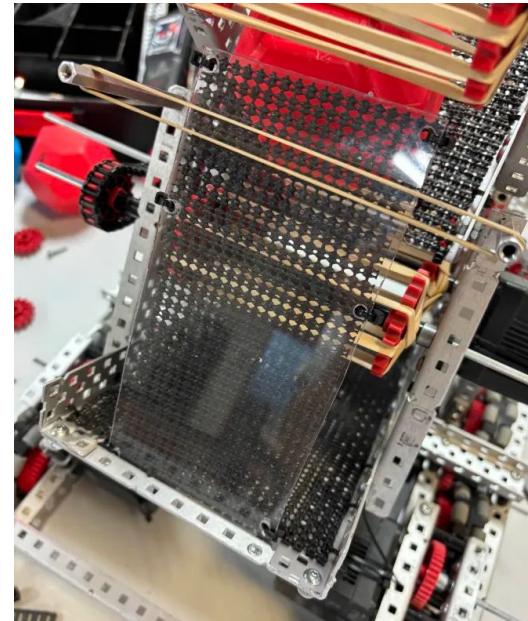


Figure 1: Back view of our polycarbonate with mesh

Next Steps

Testing our robot and then moving on to build a mechanism that allows us to store blocks.

- ▶ **Problem:** Scoring blocks gives us advantages during matches and skills

Date: July 1, 2025

Storing Blocks

One important aspect is to be able to store blocks for future scoring later on in the game. During a match, it would be very inefficient if we needed to go get a block on the field, then come back to score it one by one. Being able to hold multiple blocks and score all of them in succession would provide a large advantage during a match.



Figure 1: Storage robot (Provided by team 3004A)

How do we store in our robot?

Our robot does not have a separate storage mechanism so we need to find a way to be able to store blocks with the constraint of our current robot. The current sections of our robot all of rollers which means blocks will constantly be moving through our robot.

- ▶ Focus: How do we store blocks with our current robot?

Date: July 1, 2025

Trapdoor

The first design we came up with was to create a simple piston activated trapdoor, either vertically or horizontally. This would allow us to open or close the trapdoor to allow for scoring and storage.

PTO Rollers

Another option we thought of was to create a PTO (power takeoff) engaged roller. This would allow us to completely stop running a set of rollers, preventing the block from moving. The PTO can be triggered to determine when we want to score.

Stopping Intake

The last option we came up with was to just completely stop spinning our stop stage intake. Because our top and bottom stages of intake are run by separate motors, we can choose to completely stop the top stage to start storing blocks.

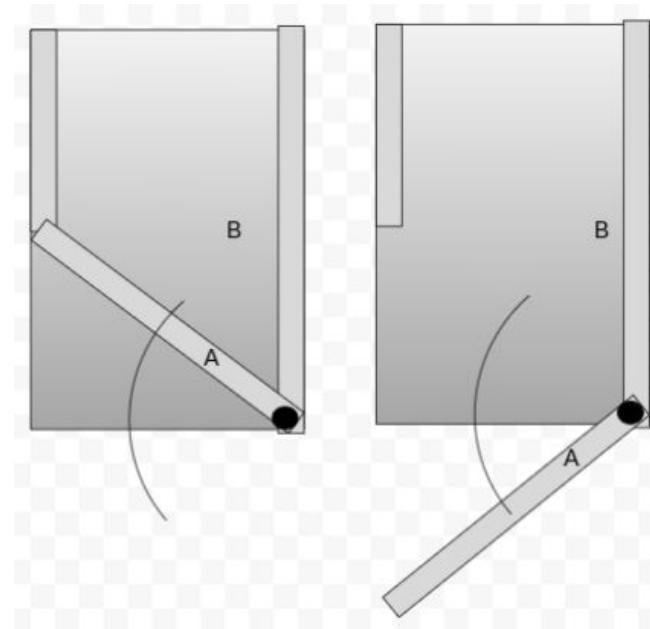


Figure 1: Trapdoor concept schematics (VEX Forums)

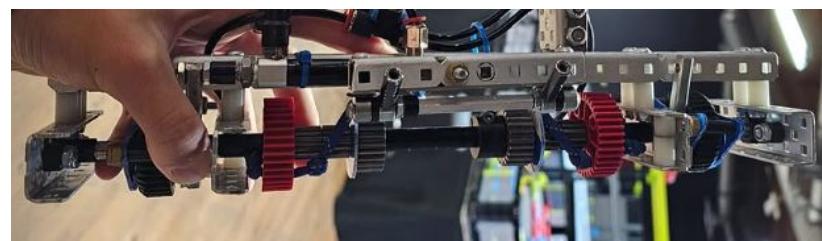


Figure 1: PTO for a high hang created in over under (Team 210T)

- ▶ Focus: Selecting a design to store blocks

Date: July 1, 2025

After analyzing some designs, we decided to consider the trapdoor, PTO and split motors method of holding blocks. We would rank these designs based on the following criteria

- Build Simplicity (1 - 5)** - How easily this design can be built, the design should not be overly complex.
- Ease of implementation (1 - 5)** - How easily we can implement this design onto our current robot
- Compactness (1 - 5)** - How much space would this design take up?
- Weight (1 - 5)** - How light or heavy this design would be

<u>Criteria</u>					
<u>Options</u>	Build Simplicity	Ease of Implementation	Compactness	Weight	Total
PTO	2	1	1	1	5
Split Motors	2	5	3	3	13
Trapdoor	4	5	3	4	16

Looking at our designs and how we ranked them, it was clear that the trapdoor was the best design we have. The trapdoor was something that we were comfortable building which was a factor that lead to us choosing it. A factor that lead us not to choose splitting our motors was the allocation of our motors. We did not want to use 5.5W motors because they historically have performed poorly.

- ▶ Focus: Planning a piston powered trapdoor

Date: July 1, 2025

Piston Powered Trapdoor

We plan to create a piston powered horizontal trapdoor. The concept behind the trapdoor is a class 1 lever. We plan to utilize a hinge piece as our fulcrum to allow it to rotate. The C-Channel will act as the main piece to prevent blocks from travelling out of the robot. Finally, the piston will be attached to one end of the C-channel to allow it to be opened and closed.



Figure 1: VEX hinge (Idesign website)

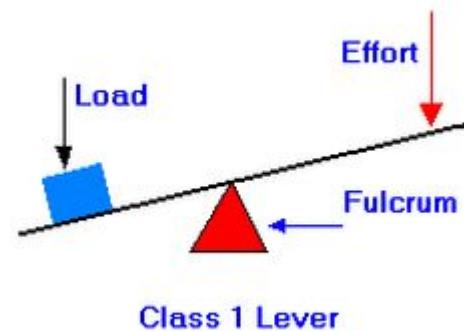


Figure 2: Diagram of a 1st class lever

- Focus: Being able to hold blocks

Date: July 1, 2025

Members Involved

Daniel, Bryan

Objective

Our objective is to create a **1st class lever** which closes when the piston extends and opens when the piston retracts. We will use a **hinge** to act as our **fulcrum**, allowing for our trapdoor to open and close to store blocks in our intake.

Materials

- 1, 1x2x1x10 C-channel
- 1 hinge
- 1 piston
- Spacers
- Nylocks
- Axle collar
- Screws

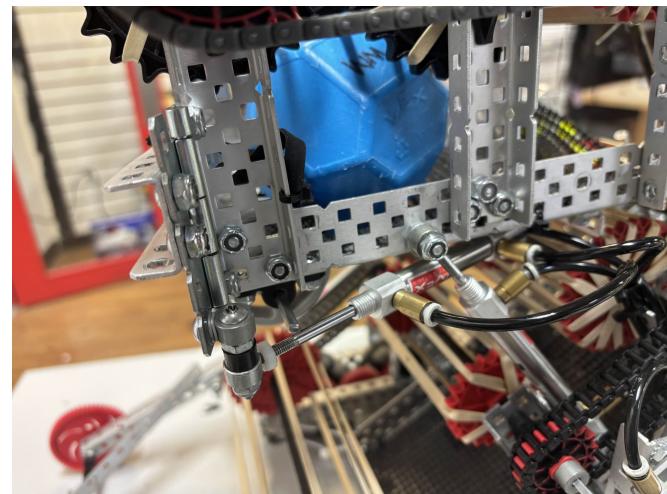


Figure 1: Piston attached to the hinge

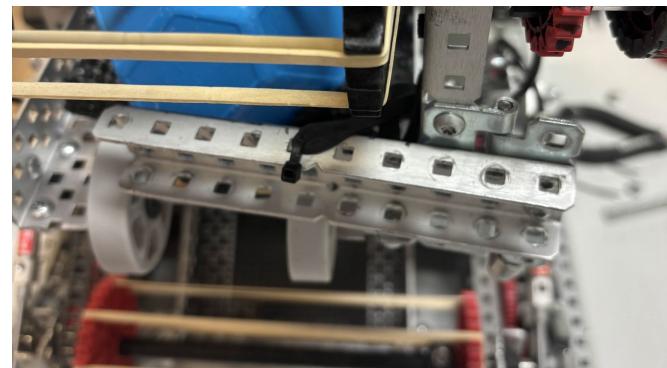


Figure 2: Trapdoor in the closed position

- Focus: Being able to hold blocks

Date: July 1, 2025

Problems and Solutions

The 1st class lever was very weak and was able to be pushed open by a block.

This was solved by changing the piston placement and C-channel length. A rubber band was also added to help maintain force when the trapdoor is closed

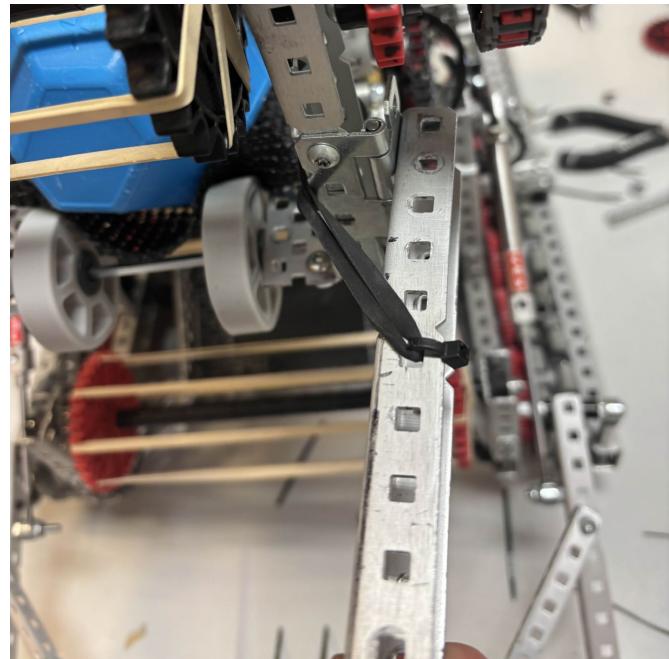


Figure 2: Open trapdoor

Next Steps

Planning and building a matchloading mechanism.

Version 1.0:
Mall of America
Signature Event

Match Loader

- ▶ Problem: How to remove a block from the match loading tube

Date: July 1, 2025

Match Loading Tubes

One of the main challenges with match loading blocks is **taking the blocks out** of the match loading tube. There is a **lip** at the bottom of the tube that is 0.79 inches high, but the block will be lower and touch the field tiles. Having a **vertical roller** is **ideal** for clearing matchload tubes, but with our horizontal intake, we have to find a **different method** of taking blocks out of the match loading tube. A key problem is to create a wedge that is higher than the lip, but still able to **slide underneath the blocks** and make the blocks fall towards our robot.

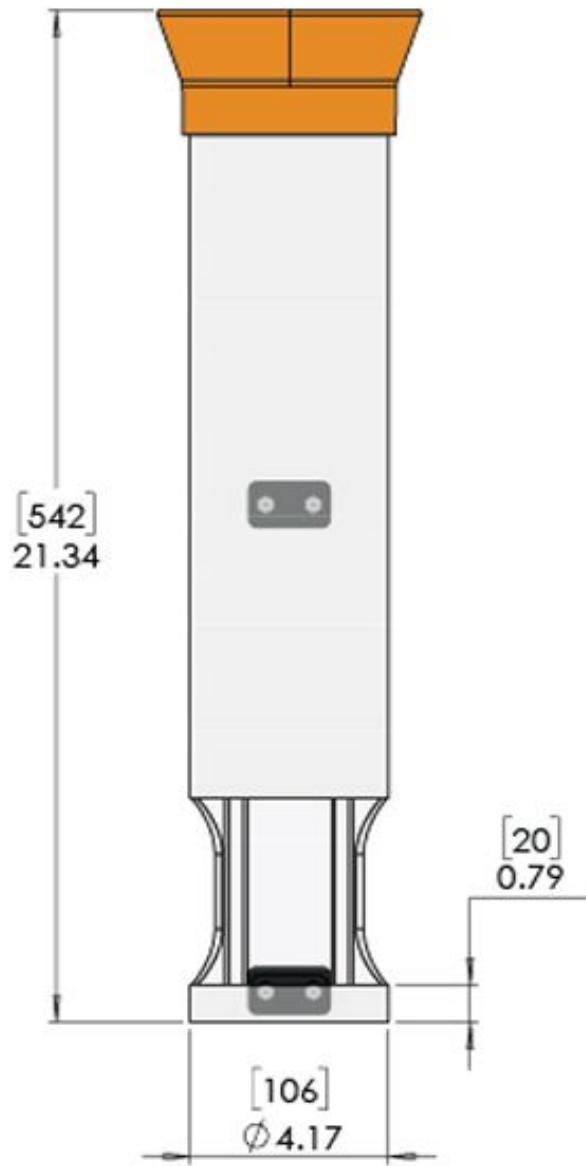


Figure 1: Schematic of a matchloading tube (VEX Forums, Game Manual)

- ▶ Focus: Solution to match loading

Date: July 2, 2025

Importance of Matchloading Mechanisms

A matchloader can be **inserted** into the **matchload tube** to quickly and effectively remove all the blocks out of the tube. This **time save** is going to be especially valuable in the long run for **Solo Autonomous Win Point** as we have 15 seconds. As a result, we believe this one piece of polycarbonate could significantly help us and is worth using towards our limited twelve pieces of polycarbonate.

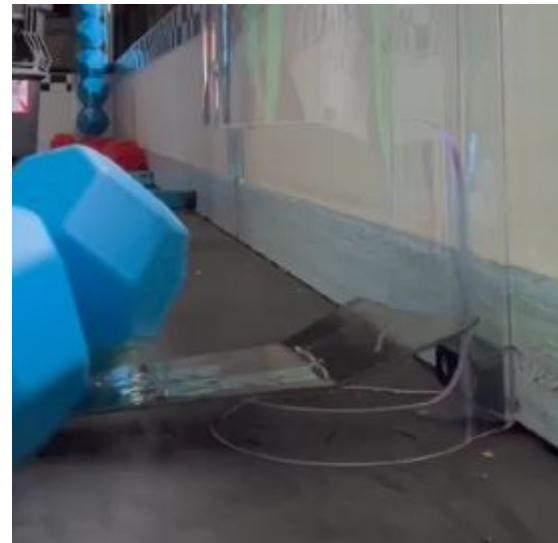


Figure 1: Plastic sheet used as a matchloading mechanism (Vex CAD community, 920B)

Methods of Matchloading

Throughout the past few months, we have seen development on how teams with **horizontal intakes** remove blocks from the matchload tubes. There are **three main methods** that have been created.

1. Counter-rollers
2. Polycarbonate piece
3. Standoffs



Figure 2: A match loading mechanism that does not utilize polycarbonate

- ▶ Focus: Brainstorming methods of match loading

Date: July 2, 2025

Counter-roller

A counter roller utilizes **omni-rollers** taken off from an omni wheel that is **powered by a motor**. The roller is inserted underneath the block, **spinning in a direction** such that the block is moved out of the lip



Figure 1: Counter-rollers (VCAD. 2915V)

Polycarbonate

The polycarbonate method utilizes a **V-shaped piece of polycarbonate**. It initially slides underneath the blocks before **pivoting** where the blocks are naturally **funneled towards** the intake due to its **ramp like structure**.



Figure 2: Polycarbonate piece (VCAD, 9805S)

Standoffs

The standoff method utilizes **standoffs** attached to **high strength shaft collars**. They are positioned like forks, allowing the mechanism to **slide underneath the block**. A second set of standoff forks are used as a **ramp to funnel blocks** towards the robot

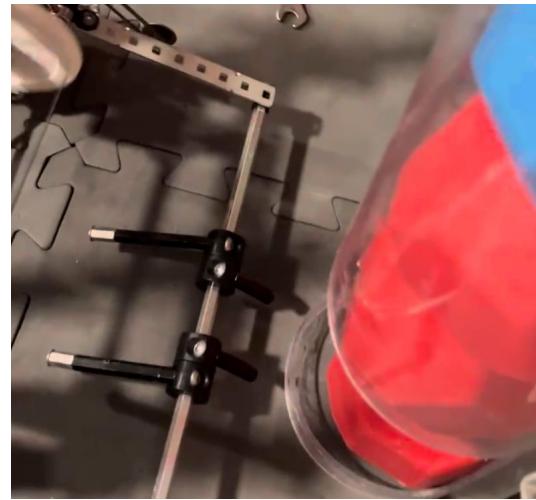


Figure 3: Standoff method (VCAD, 21S)

Match Loading Decision Matrix

Select Solution

- Focus: Deciding on a match loading mechanism

Date: July 2, 2025

When **considering our options** between the **counter-roller**, **polycarbonate** and **standoff** match loading mechanism, we must consider the following criteria:

- Build Feasibility (1 - 5)** - Ranked on how simple the mechanism is to build.
We want to keep our design as simple as possible
- Weight (1 - 5)** - How much weight would this design add onto our robot?
- Speed at removing blocks (from online data) (1 - 5)** - Using online data, we ranked each mechanism based on how fast they could remove blocks from the loader.

<u>Criteria</u> <u>Options</u>	Build Feasibility	Weight	Speed at Removing Blocks	Total
Counter-Roller	2	2	4	8
Polycarbonate	3	3	3	9
Standoffs	4	3	4	11

Looking at the different match loading mechanism designs, we decided to pick something **well-rounded and balanced**. Taking into the different factors, we decided to **experiment between polycarbonate and standoffs** as they are a lot more **simple** to build, and do not need to be powered by motors. Based off of our testing, we will **keep the design that performs better** and continue to **tune** our mechanism for the best results.

► Focus: Deciding on a match loading mechanism

Date: July 2, 2025

Below is a graph of the decision matrix on the previous page.

V1 Matchloading Mechanism Designs

— Counter-roller — Polycarbonate — Standoffs

Build feasibility

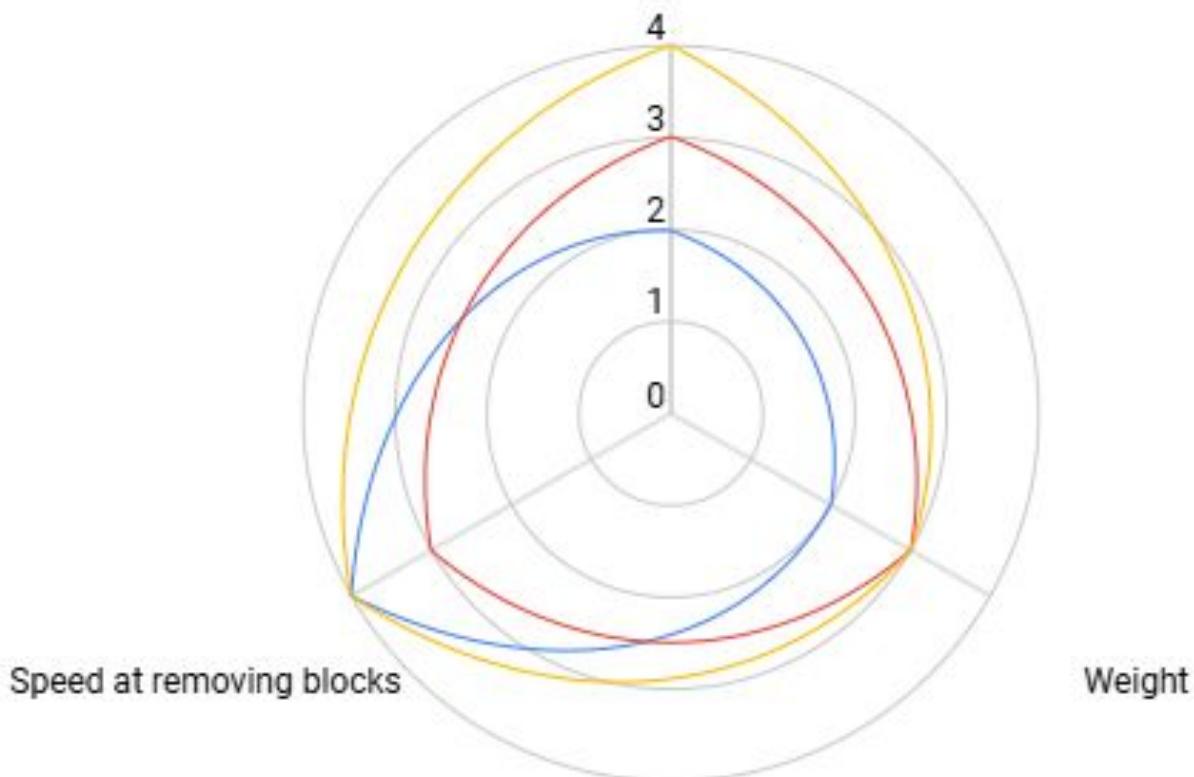


Chart 1: Comparing Match Loading Mechanism Designs

- ▶ Focus: Planning out multiple matchloading mechanisms

Date: July 2, 2025

Matchloading

We plan to try out multiple different variations of our matchloading mechanism.

C1.0

The **first method of match loading mechanism** we wanted to try was the standoff method. We plan to use angle bars with a standoff brace. We will then add strength shaft collars with standoffs attached to them to remove the blocks from the tubes. We will **utilize shaft collars** and angle bars to create a triangle brace for stability.



Figure 1: Standoffs bracing with high strength collars (YouTube, 44252A)

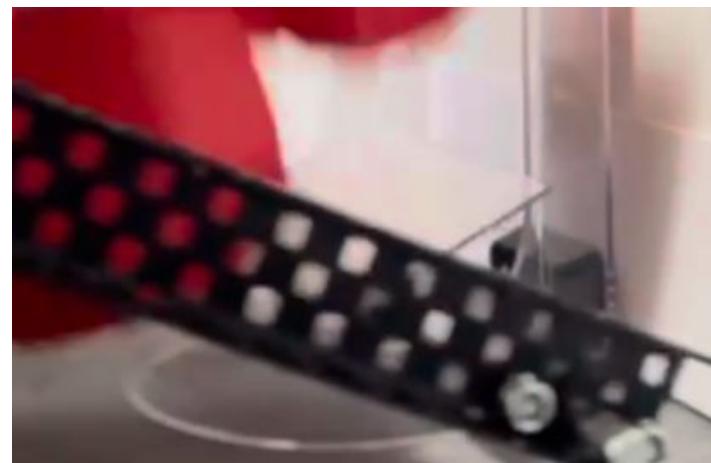


Figure 2: Pivoting polycarbonate piece (VCAD. 1069A)

C1.1

The second version that we want to build is a polycarbonate mechanism that will use a high strength shaft and a piece of polycarbonate to wedge underneath the blocks. The polycarbonate will be made in a way where it can pivot in order to remove blocks from the matchloading zone.

C1.2

The third and final version we will test uses a high strength shaft as a brace. This version will use high strength shaft collars and standoffs to remove blocks from the matchload zone.

► Focus: V1 Standoff Method

Date: July 3, 2025

Members Involved

Joshua, Bryan, Daniel

Objective

Our goal was to follow the planning for matchload mechanism C1.0. We would create a mechanism built using angle bars, standoffs and high strength shaft collars.

Materials

- 2, 1x1x13 angle bar
- 2, 1x1x10 angle bar
- 2 small standoffs
- 1 long standoff
- 4 high strength shaft collars
- 2 big ziptie (optional)
- 2 small ziptie (optional)
- Screws
- Nylock
- 2 sprockets as floor riders (optional)

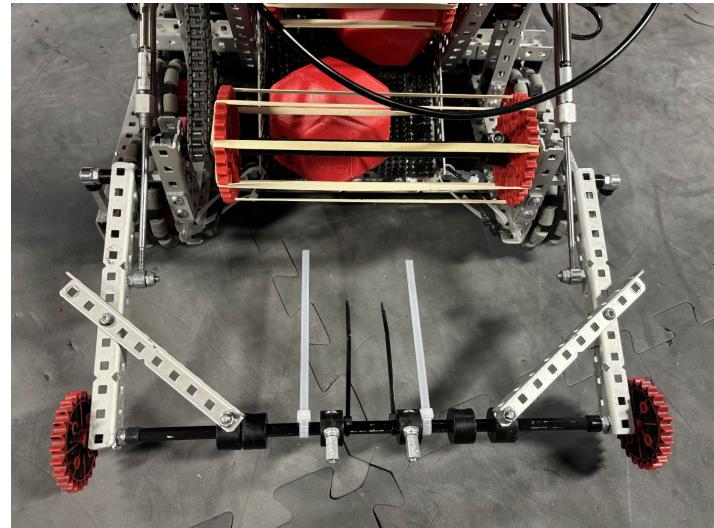
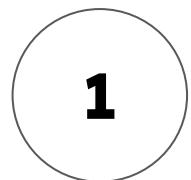


Figure 1: Standoff Matchloading

► Focus: V1 Standoff Method

Date: July 3, 3035

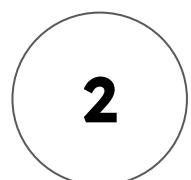
Starting Match Loading System



X1 6" Standoff

X1 5" Standoff

X1 1" Coupler



X6 High Strength Axle Collars



X2 13 Hole 1x1 Angle Bar



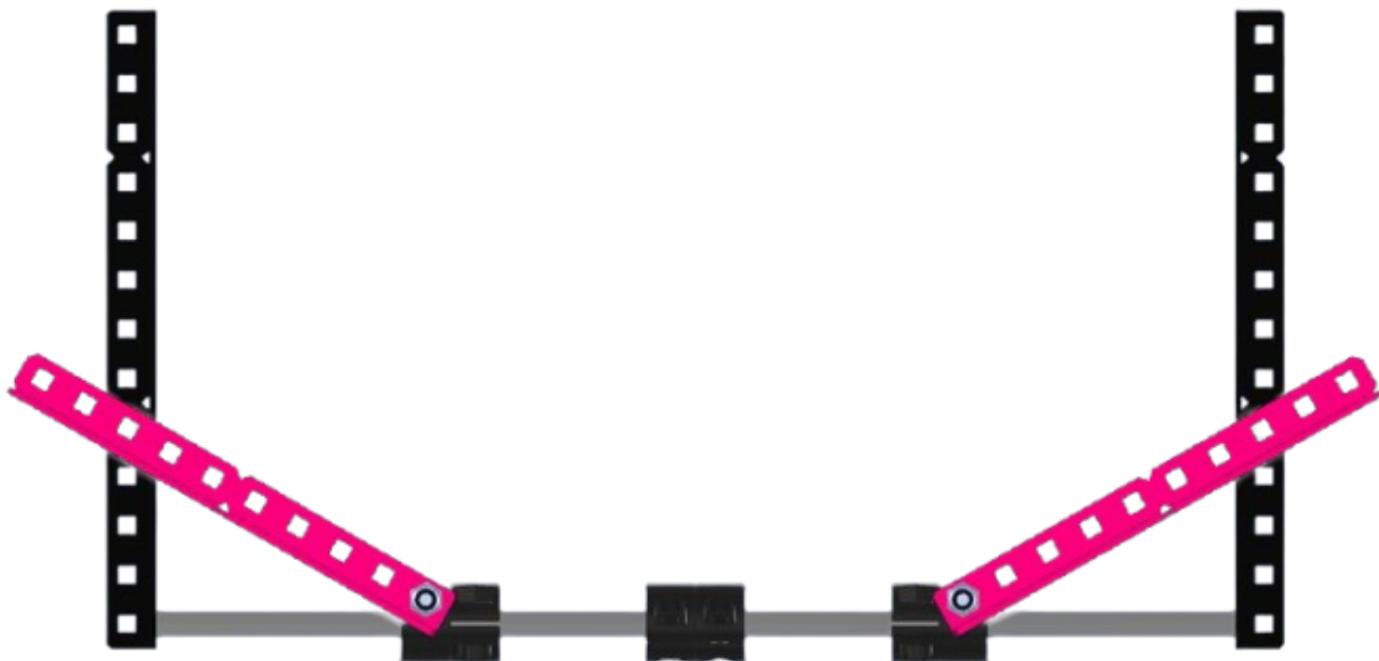
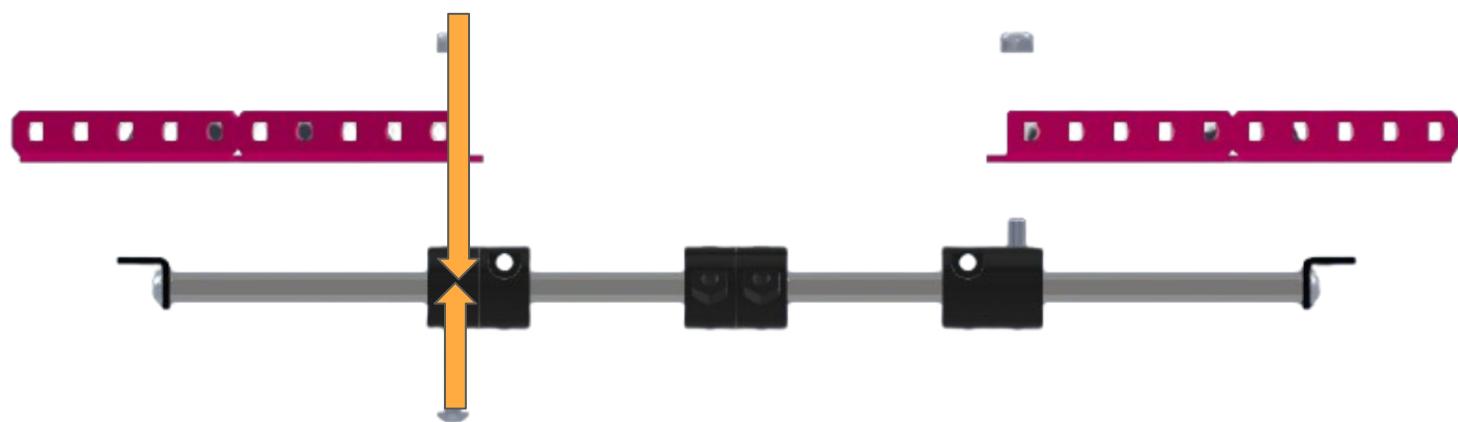
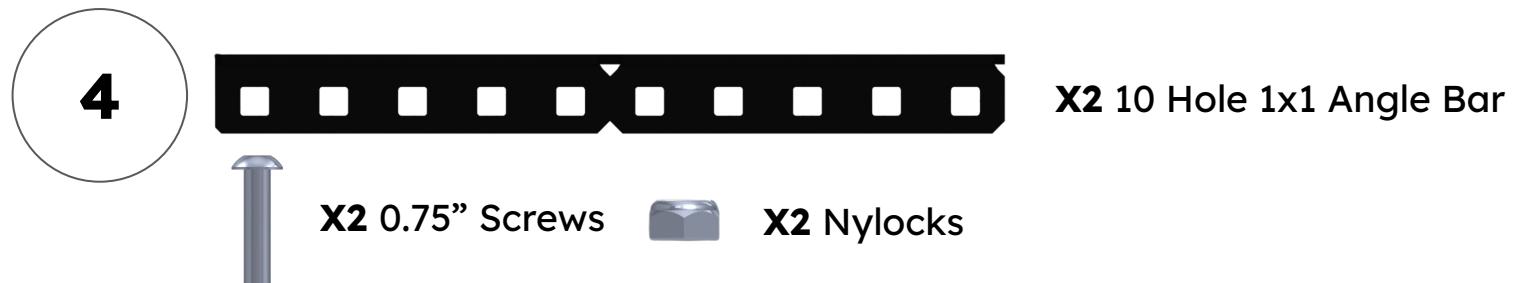
X2 0.375 Screw



► Focus: V1 Standoff Method

Date: July 3, 3035

Starting Match Loading System



► Focus: V1 Standoff Method

Date: July 3, 3035

Starting Match Loading System

5



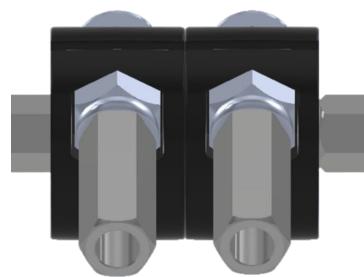
X2 0.75" Screws



X2 Nylocks



X2 0.75" Standoff



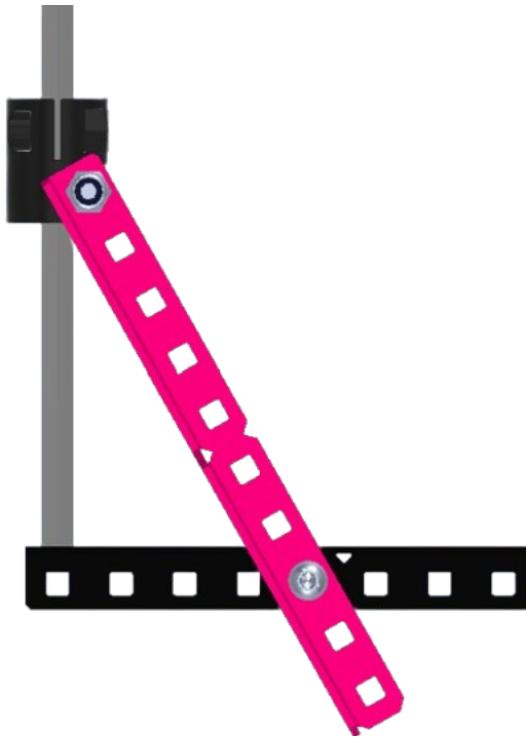
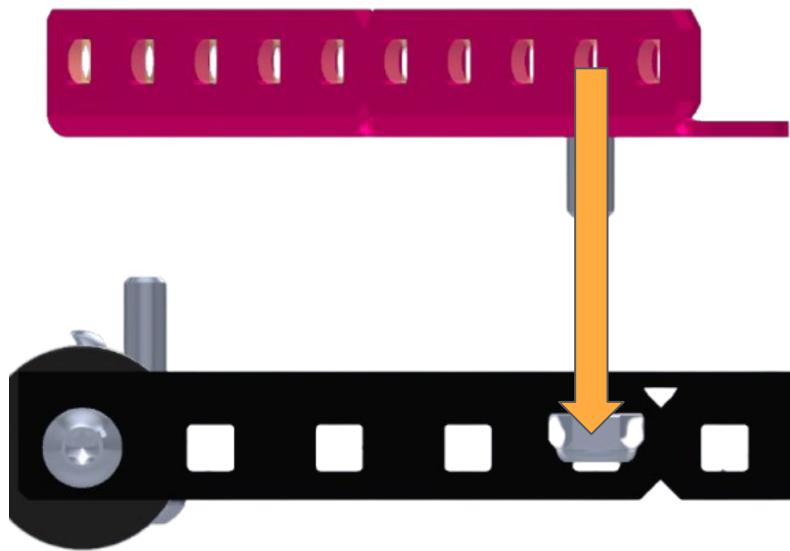
6



X2 0.375 Screw



X2 Nylocks

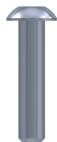


► Focus: V1 Standoff Method

Date: July 3, 3035

Starting Match Loading System

7



X2 0.75" Screws



X2 Nylocks



X1 $\frac{3}{8}$ " Axle Collar



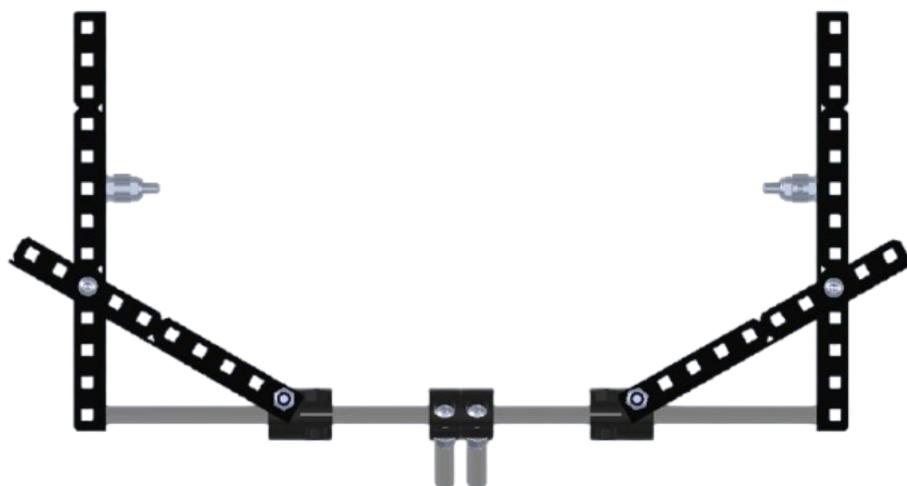
X2 Hexnuts



**Repeat on both sides
and add Medium
piston to the axle
collar**



These are the **basics** to a match loader, with the addition to optional changes such as **rollers** as well as quality of life items such as **zip ties**. Every version of this match loader will follow a similar design to this. It should look something like:



► Focus: V1 Standoff Method

Date: July 3, 2025

Problems and Solution

When initially testing our mechanism, we found that the blocks would not fall towards the intake.	To solve this issue, we added zip ties as ramp funnels to force the blocks to go into our intake.
When our matchloading mechanism was in the down position, we found that it would get stuck in the field tiles.	We added sprockets to the sides of our mechanism to act as floor-riders. This prevents us from getting stuck in the field tiles.



Figure 1: Sprocket floor-rider

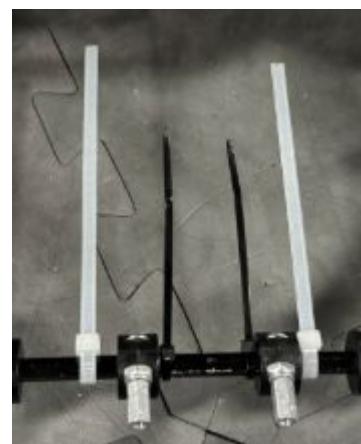


Figure 2: Ziptie Ramps

- ▶ Problem: Testing our match loading mechanism

Date: July 6, 2025

Purpose

The purpose of this test is to find whether or not our matchloading mechanism would be viable during a match, where saving time when performing certain actions is necessary to gain an advantage. We want to aim for 2-4 seconds to matchload 6 blocks.

Procedure

1. Have one team member ready with a stopwatch
2. When ready, drive the robot with the matchloading mechanism into the matchloading tube
 - a. Start the stopwatch at this time
3. See how long it takes for 6 blocks to be removed from the matchload tube and into our intake system
 - a. Stop the stopwatch once the 6th block enters into our intake
4. **Record** the result under “time to matchload, t (s)” to **three significant digits**. Repeat steps **2-3** until enough trials are produced.

Table 1: Time to clear matchload zone, t (s), in relation to trial number

Trial Number	Time to Clear, t (s)
1	23.54
2	21.56
3	19.95
4	20.67
5	17.52
6	18.91
7	30+ (stuck)
8	24.79
9	22.17
10	24.89

- ▶ Problem: Testing our match loading mechanism

Date: July 6, 2025

Analysis and Conclusions

After testing our match loading mechanism. We found it to be extremely **inconsistent and unreliable**. It often **took too long** to be inserted into the tube. Our mechanism proved to be too slow, as it on average, it took **21.55** seconds to remove 6 balls from the tubes. Another key issue is that the **standoff** that runs across the angle bars become **bent** at the end of the testing trials.

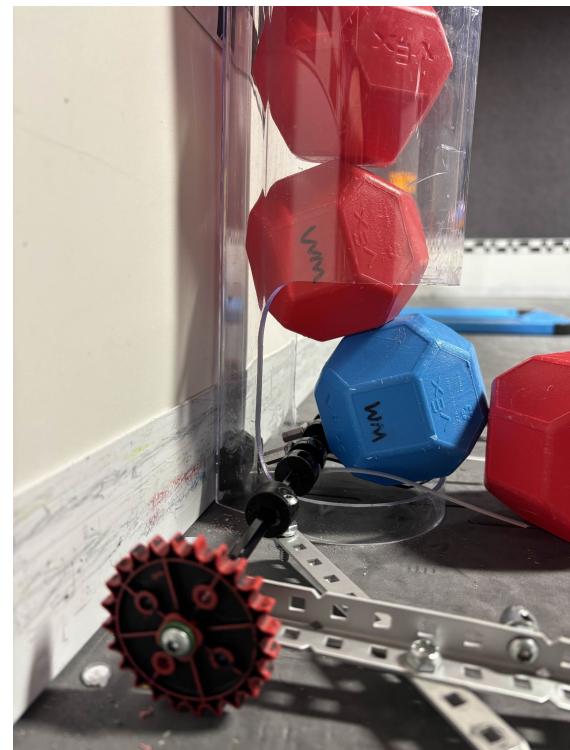


Figure 1: Mechanism inserted into the tube, descoring blocks

- ▶ Focus: Testing a polycarbonate design

Date: July 6, 2025

Members Involved

Daniel, Bryan

Objective

We wanted to build and test our planned C1.1 match loading mechanism that utilized a pivoting V-shaped polycarbonate sheet. We aim to weigh our option through testing results before selecting our final design.

Materials

- 2, 1x1x12 angle bars
- High strength shaft
- Zip Ties
- Rubber bands
- Polycarb piece
- screws/nylocks



Figure 1: Polycarbonate match loading mechanism

- ▶ Problem: Testing our match loading mechanism

Date: July 6, 2025

Purpose

The purpose of this test is to find whether or not the second version of our mechanism would be more viable than the first version during a match.

Procedure

1. Have one team member ready with a stopwatch
2. When ready, drive the robot with the matchloading mechanism into the matchloading tube
 - a. Start the stopwatch at this time
3. See how long it takes for 6 blocks to be removed from the matchload tube and into our intake system
 - a. Stop the stopwatch once the 6th block enters into our intake
4. **Record** the result under “time to matchload, t (s)” to **three significant digits**. Repeat steps 2-3 until enough trials are produced.

Table 1: Time to clear matchload zone, t (s), in relation to trial number

Trial Number	Time to Clear, t (s)
1	6.52
2	5.49
3	5.78
4	6.24
5	5.79
6	5.62
7	6.21
8	6.46
9	5.96
10	6.04

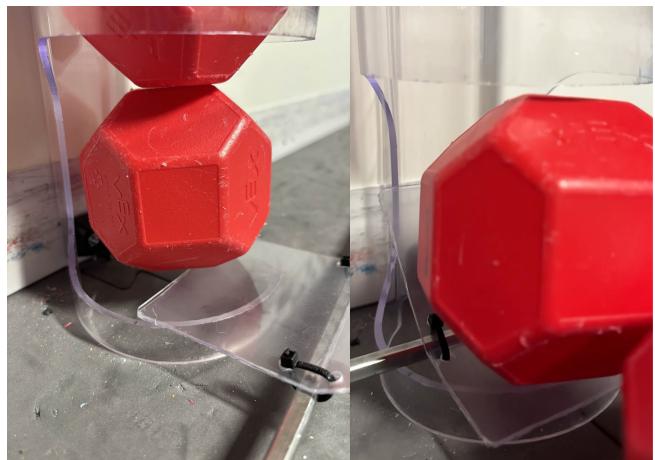


Figure 1: Polycarbonate inserted into the tube then pivoting

- ▶ Problem: Testing our match loading mechanism

Date: July 6, 2025

Analysis and Conclusion

After testing the polycarbonate iteration, we found that the pivoting polycarbonate **removed blocks** at a very **fast speed**.

However, we found that polycarbonate had a different problem. A flat piece of polycarbonate would **struggle to slide underneath** the blocks when the hexagon side is facing the lip.

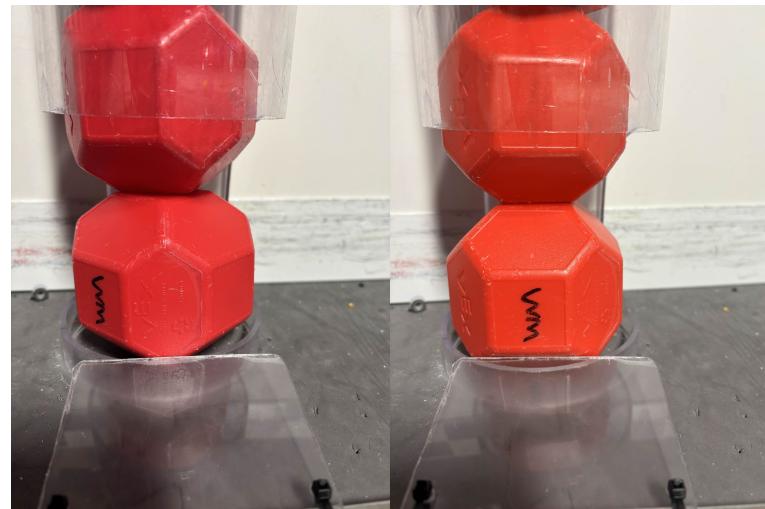


Figure 2: Polycarbonate unable to slide under the block when the hexagon side faces the lip

- Focus: Third iteration of our mechanism

Date: July 6, 2025

Members Involved

Daniel, Bryan

Objective

Build version C1.2 of our match loading mechanism and improving off previous designs. We would maintain stability by bracing our mechanism and using high tolerance materials

Materials

- 2, 1x1x12 angle bars
- 2, 1x1x9 angle bars
- 6 high strength shaft collars
- 2 shaft collars
- Nylocks
- Screws
- High strength shaft
- Spacers
- Pulley (optional however recommended)
- Kepsnuts
- Zipties (optional)



Figure 1: Matchloading mechanism being inserted into the tubes

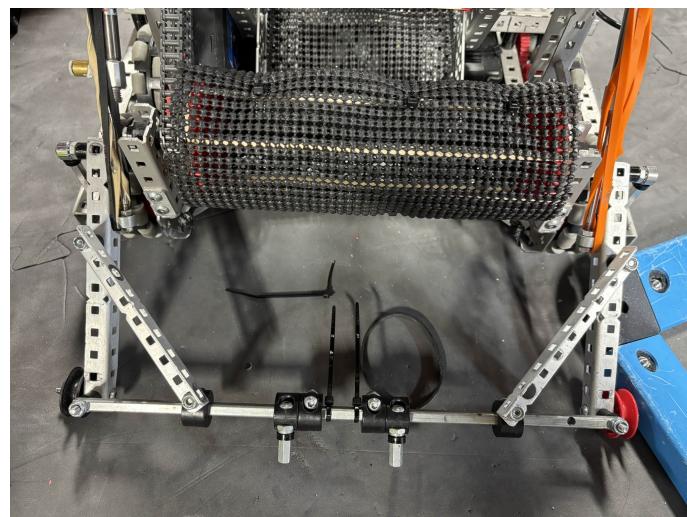


Figure 2: Picture of our matchloading mechanism

- ▶ Problem: Testing our match loading mechanism

Date: July 6, 2025

Purpose

The purpose of this test is to find whether or not the second version of our mechanism would be more viable than the first version during a match.

Procedure

1. Have one team member ready with a stopwatch
2. When ready, drive the robot with the matchloading mechanism into the matchloading tube
 - a. Start the stopwatch at this time
3. See how long it takes for 6 blocks to be removed from the matchload tube and into our intake system
 - a. Stop the stopwatch once the 6th block enters into our intake
4. **Record** the result under “time to matchload, t (s)” to **three significant digits**. Repeat steps 2-3 until enough trials are produced.

Table 1: Time to clear matchload zone, t (s), in relation to trial number

Trial Number	Time to Clear, t (s)
1	1.43
2	1.96
3	3.22
4	2.14
5	1.79
6	2.56
7	1.96
8	2.08
9	1.83
10	1.52

- ▶ Problem: Testing our match loading mechanism

Date: July 6, 2025

Analysis and Conclusion

After our testing trials, it was clear that this iteration was the **best performing** on average taking **2.05** seconds to remove all 6 blocks from the tube. This saves us a lot of time during autonomous and driver control when we want to get matchloads. We **decided to keep this iteration** on our robot.

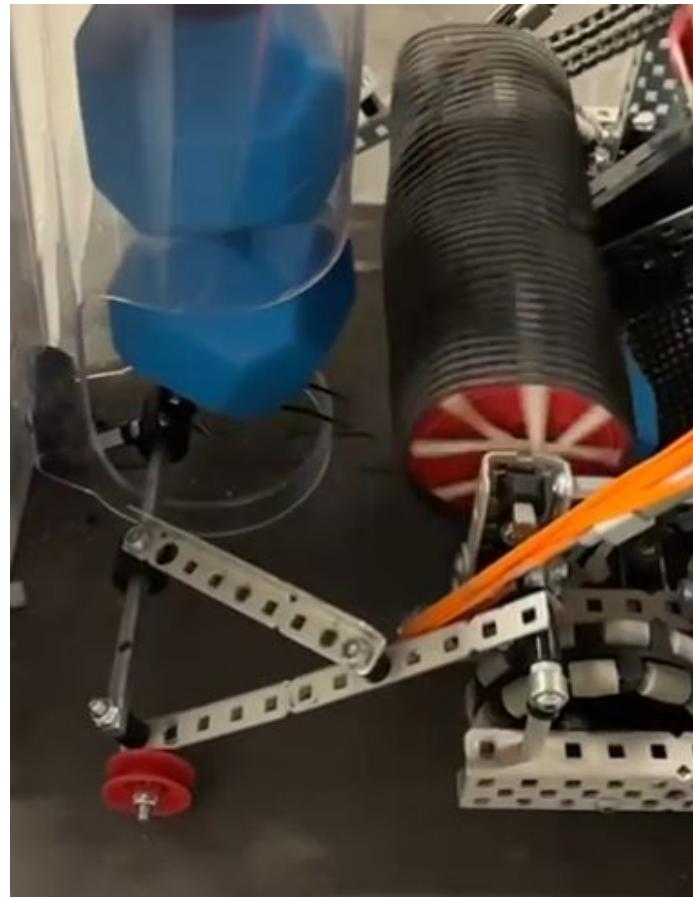


Figure 1: Mechanism removing blocks

- Problem: Testing intake efficacy based on transport speed.

Date: July 7, 2025

Purpose

The purpose is to test if our bottom stage intake can efficiently complete its task of picking up blocks during a match.

Procedure - Test 1

1. Place a block in front of the intake
2. Have team member with a stopwatch prepared
3. When ready, drive the robot forward into the block
4. Time how fast it takes for the block to travel from our initial intake to the second set of rollers.

Procedure - Test 2

1. Place a block to the side of the intake
2. Have a team member with a stopwatch prepared
3. When ready, drive the robot forwards into the block
4. Time how fast it takes for the block to travel into our robot.

Table 1: Intake a single block, t (s), in relation to trial number

Trial Number	Time to Clear, t (s)
1	1.72
2	2.07
3	1.68
4	1.98
5	1.86
6	2.16
7	1.74
8	1.96
9	2.11
10	1.79

- Problem: Testing intake efficacy based on transport speed.

Date: July 7, 2025

Analysis and Conclusion

After the testing trials, we found from our data that our intake performed fine when intaking a block that is directly in front of it. However, it would **struggle to pick up blocks that were off to the side**. Looking at these results, we plan to rebuild our intake.

Table 2: Intake a single block, t (s), in relation to trial number

Trial Number	Time to Clear, t (s)
1	3.24
2	N/A
3	3.51
4	3.66
5	N/A
6	N/A
7	3.43
8	3.27
9	3.18
10	3.37

- ▶ Problem: Identifying issues with our intake

Date: July 7, 2025

Pickup issues

During our testing, we found that our intake was at a very **odd height** and did not perform as well as we hoped. Our intake would often pivot up and down, causing it to **lose contact** that is needed to roller in blocks. The hourglass shape of the rubber bands also caused problems as the block would have inconsistencies with keeping contact with the rubber bands.

Width of intake

Throughout our tests we discovered that our intake was **not wide enough** to successfully intake blocks. These blocks would often get **stuck on our sprockets**, leading to us being **unable to pick them up**.

- ▶ Focus: Redesigning and thinking about new intake ideas

Date: July 7, 2025

Discobots Wide Intake

After testing our intake, we decided to **switch designs** to solve the many issues we encountered. We noticed in **VRC Nothing But Net**, the Discobots organization (2587/1104) had an **extremely wide intake** that funneled towards the center of their robot. We decided to **implement** this design onto our own robot with some modifications.

The discobot intake uses a **fixed point** where the axle is mounted. It is made using sprockets and rubber bands. This intake is **wide enough to intake multiple balls at the same time**. To prevent jamming or balls getting stuck, they utilized **offset funnels** to allow the balls to move towards the center of the robot.



Figure 1: Discobots fixed intake

- ▶ Focus: Planning out our wide intake

Date: July 7, 2025

Rebuilding Intake

We decided to build a **wide intake** because it would solve many of our previously mentioned issues. A wide intake would give use **more surface area** to pick up blocks. It would also solve the issue of blocks getting stuck underneath our sprockets.

We plan to change up how we build our intake. We decided to utilize C-channels as our base structure. This should give us enough structural stability. This version of the intake will **not pivot**. It will utilize sprockets with rubber bands wrapped in mesh to **prevent jamming** but have enough grip to intake the blocks. We do not plan to have offset funnels as we found that they provided minimal benefits as the shape of our robot would naturally funnel the blocks in.



Figure 1: Discobots offset



Figure 2: Mesh intake (YouTube, 2775V)

- ▶ Focus: Discobots Intake Implementation

Date: July 8, 2025

Members Involved

Daniel, Brandon

Objective

Our objective is to rebuild our intake to a very wide fixed intake. This will help us solve our previous intake issues and improve our intake.

Materials

- 2, 1x2x1x5 C-channel
- 2, 2x2x2x1 U-channel
- 1 high strength shaft
- 2, 32 teeth sprocket
- Spacers
- High strength shaft
- Rubber bands
- 1 pinion
- 6p chain (iq chain)
- High strength spacers
- Screws and nylocks



Figure 1: Rebuilt intake

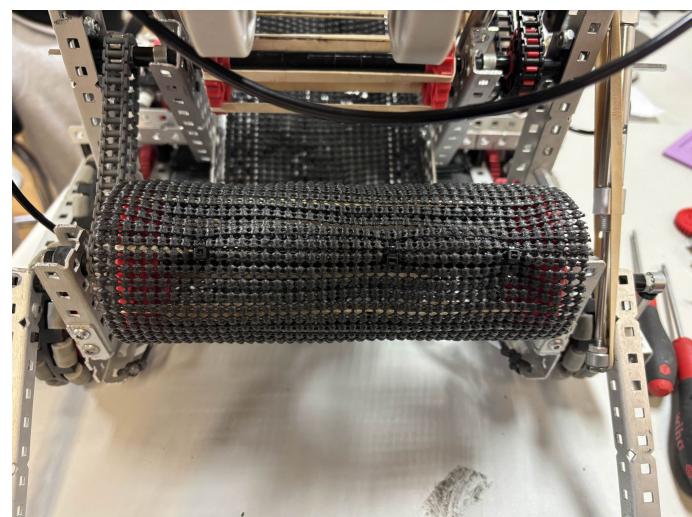


Figure 2: Rebuilt intake wrapped in mesh

► Problem: Issues found when practicing with the robot

Date: July 13, 2025

Scoring Inconsistency

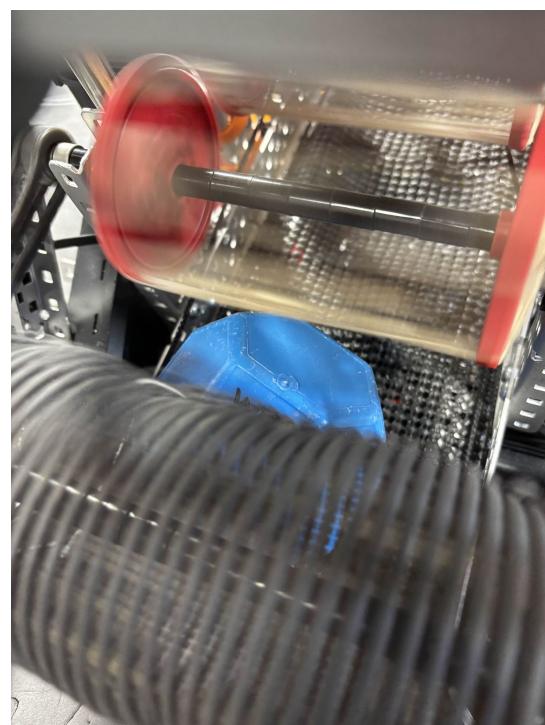
We noticed that many teams had a set of **rollers** made of either flex wheels or mesh **underneath the block** when scoring. This bottom set of rollers would help **elevate the ball slightly**. This helped the block to get scored into the tube.



Jamming

Another issue we encountered is blocks **constantly jamming** when we had 4 or more. This was caused because the blocks would **roll underneath our sprockets**, causing the blocks to roll into each other and eventually get stuck.

Figure 1: Our intake jamming during an autonomous routine



Transition Stage

The polycarbonate sheet that we used for our transition stage had a **gap in between our top and the plastic**, this caused an issue where blocks would be pushed through this gap and **out of the robot**.

Deadzones

Deadzones are caused when there is **no contact** between the game object and our rollers, or the game object **does not have enough momentum** to reach the next set of rollers. We **found a few deadzones** throughout our intake.

Figure 2: A deadzone in our robot

- ▶ Focus: Tuning our top stage

Date: July 14, 2025

Members Involved

Joshua, Daniel

Objective

Add a structural component to ensure consistent scoring. Our set of rollers were mounted on **pillow bearings** and **do not spin**, passively moving the blocks up. The flex wheel rollers being **stationary** helped with our new storage mechanism.

Materials

- Flex wheel
- Inserts
- Pillow bearing
- Spacers
- Axle
- Lock bar

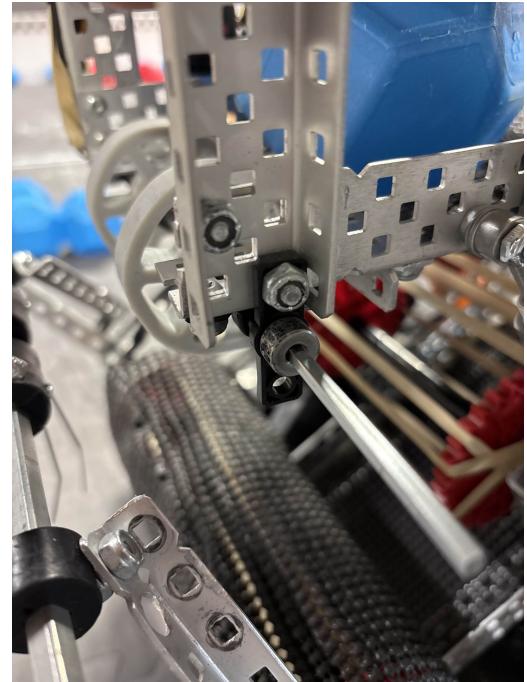


Figure 1: Lock bar preventing our flex wheel rollers from spinning.

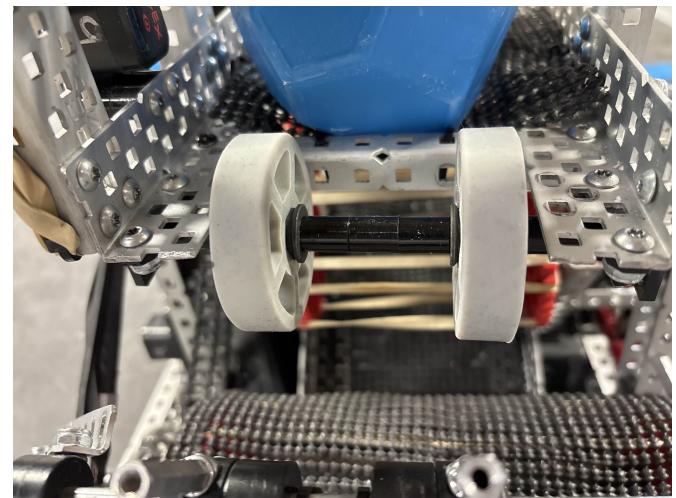


Figure 2: Our flex wheel stage placed underneath the block

- ▶ Focus: Solving deadzone issues, jamming etc.

Date: July 15, 2025

Members Involved

Joshua, Brandon, Bryan, Daniel

Objective

Each team member would take on a different issue we found during testing. We would all try to solve these issues.

Deadzones

The easiest way to solve deadzones was to **increase the sprocket size** of our **rollers**, giving the roller more **surface area** to maintain contact with the blocks. Most notably we changed from a **24 teeth** 6p sprocket to a **32 teeth** 6p sprocket. We also moved the location of our transition stage roller to be mounted on high strength pillow bearings.

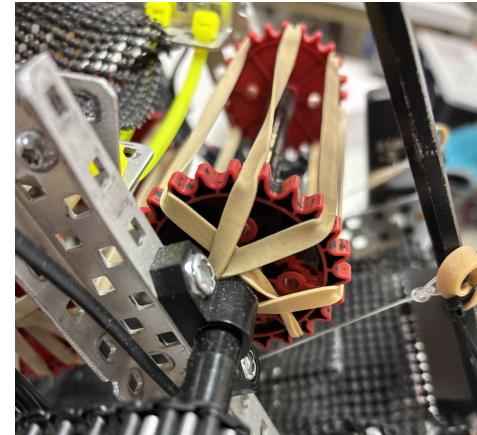


Figure 1: Picture of our moved transition roller mounted on pillow bearings

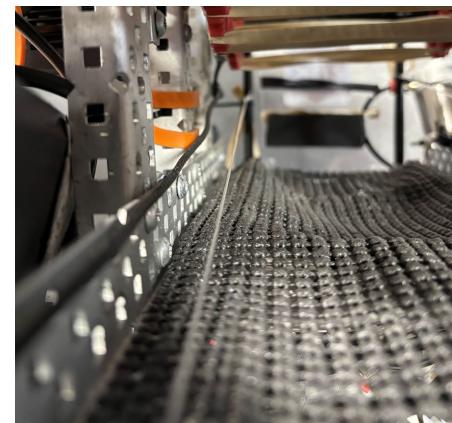


Figure 2: Picture of our fishing line limiter



Figure 3: Horizontal standoff limiter

Storage

In order to fix the issue of blocks falling out of the back of our robot, we added a **horizontal standoff** that is tied to our polycarbonate piece to **stop the vertical movement** of the blocks.

- ▶ Problem: Issues with our trapdoor

Date: July 15, 2025

Trapdoor Issues

When practicing scoring blocks, we noticed that we would often **take a block out** of the tube every time we backed up. This problem occurred because the trapdoor **would not open to a perfect 90 degrees**. It would often sit at an angle and **keep contact** with a block, causing the blocks to be pulled out of the scoring tubes when we **backed away**.

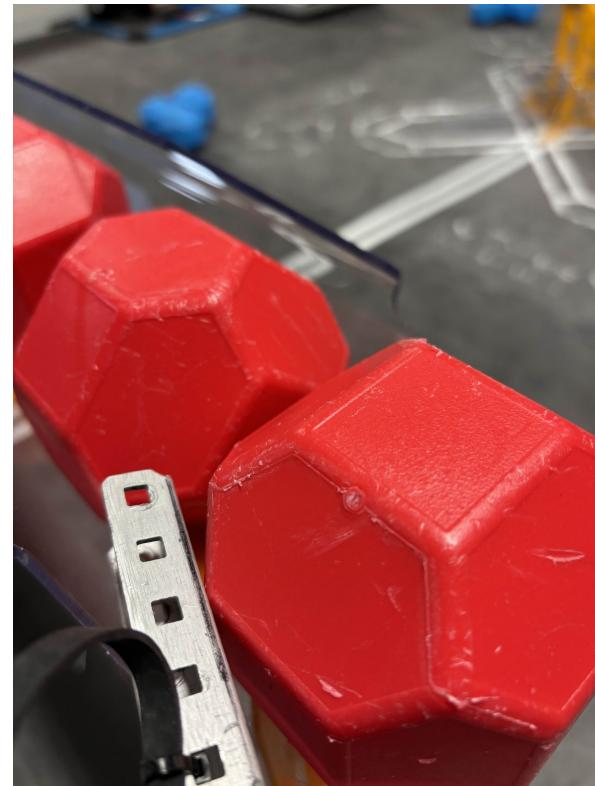


Figure 1: Block getting caught on our trapdoor

- ▶ Focus: Clutch and Ratchet

Date: July 15, 2025

While looking for ideas to address the previous problems, we came across a **clutch design** that allowed for a roller to spin when there is **no force** applied. If there was **too much force** applied, the roller would **stop spinning**. When the roller stops spinning, the rest of the system will not be impacted. For our new storing system,

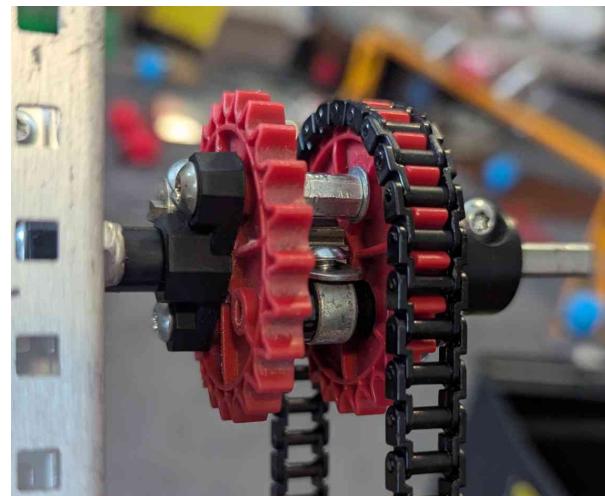


Figure 3: An example of a clutch (VCAD, 21S)

- Focus: Hoarding

Date: July 16, 2025

We decided to implement a piston powered ratchet with the clutch to start **storing blocks**. When the ratchet is engaged, it stops the set of rollers from spinning without stopping the rest of the system. We planned to implement this on our final stage, allowing us to transport blocks up to our final stage and store **9+ blocks**. This would help solve our **jamming issue** as well.



Figure 1: Demonstration of how a clutch works (VCAD, 21S)

Design Function

The design is a **pinion** fixed onto an axle, with two **free-spinning sprockets** around it.

Between the two sprockets are two ratchets made using **loosely-screwed washers**. When a light force is applied, the washers will **lay flat** against the head of the screw they are on, which will hold onto the pinion, since the **threads of the screw** will hold them in place. This is such that the sprockets and the pinion spin together. However, when **heavy force** is applied, the washer will **kick up** into an angle more **tangential to the pinion**, such that the pinion and the sprockets will be able to **spin independently**. The force required to make this effect can be **tuned** with the amount of banding on the ratchet.



Figure 2: Image of a ratchet design (Vex Forums)

- ▶ Focus: Force-engaged clutch to prevent jams

Date: July 17, 2025

Members Involved

Bryan

Objective

- We aimed to keep the clutch compact since it would have to fit on the outside of a roller
 - Thus, we **cut down** the thickness of both sprockets by filing down the axle area (extended to hold the insert)
 - This also gave us space to put bearings on the exterior. We also had to file down two axle collars such that they would not rub against the centre pinion
- After completion, we had to play around with how tightly the ratchet was rubber-banded to the pinion

Materials

- 2 24-tooth sprockets
- High strength pinion gear
- Rectangular square insert
- 2 $\frac{1}{2}$ inch standoffs
- 2 bearing flats
- Axle collars (filed down for smaller diameter)
- 3 axles
- Spacers
- Washers
- Screws
- Rubber band

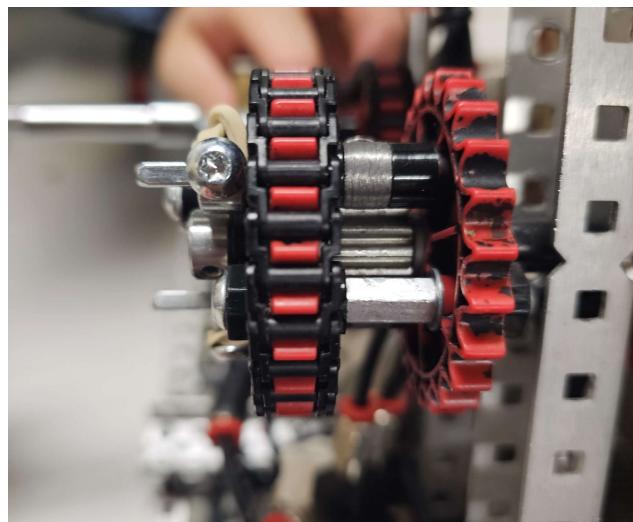


Figure 1: Picture of our clutch from the front of our robot



Figure 2: Side view of our clutch showing the rubber bands

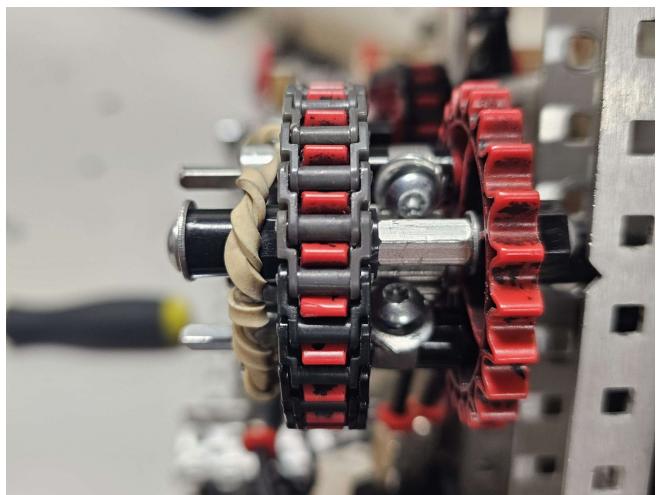


Figure 3: Picture of the internal ratchets

- ▶ Focus: Piston toggled ratchet for storage

Date: July 17, 2025

Members Involved

Bryan

Objective

After implementing the clutch, we were able to swap our trapdoor in favour of a smaller, less invasive design. We first created a **piston-powered ratchet**. This allows us to control when we would stop our last set of rollers to start **hoarding blocks**. Our ratchet would include a piston that would use zip ties to pull back a washer that is mounted to a screw joint. This washer is **banded** to always keep contact with a gear, creating a ratchet. This simple ratchet and clutch solved both our trapdoor and our jamming issues. The **intent for this design** is that when a ball approaches the last roller while the ratchet is active, it would be rolled up the last roller, to **get stuck** between the last and the second last rollers. Then, when the ratchet was deactivated, the ball would be able to pass through as usual.

Materials

- 1 small piston
- 3 large washers
- Spacers
- Screws
- Nylocks
- Shaft collar
- Rubber bands
- Zip ties

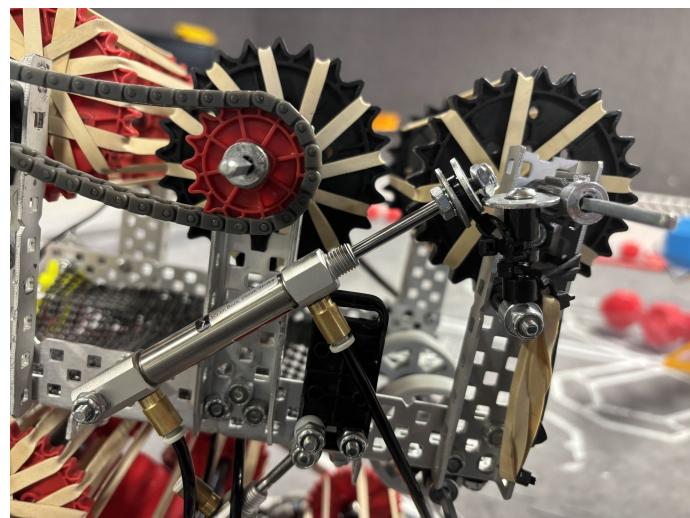


Figure 1: Side view of the piston powered ratchet

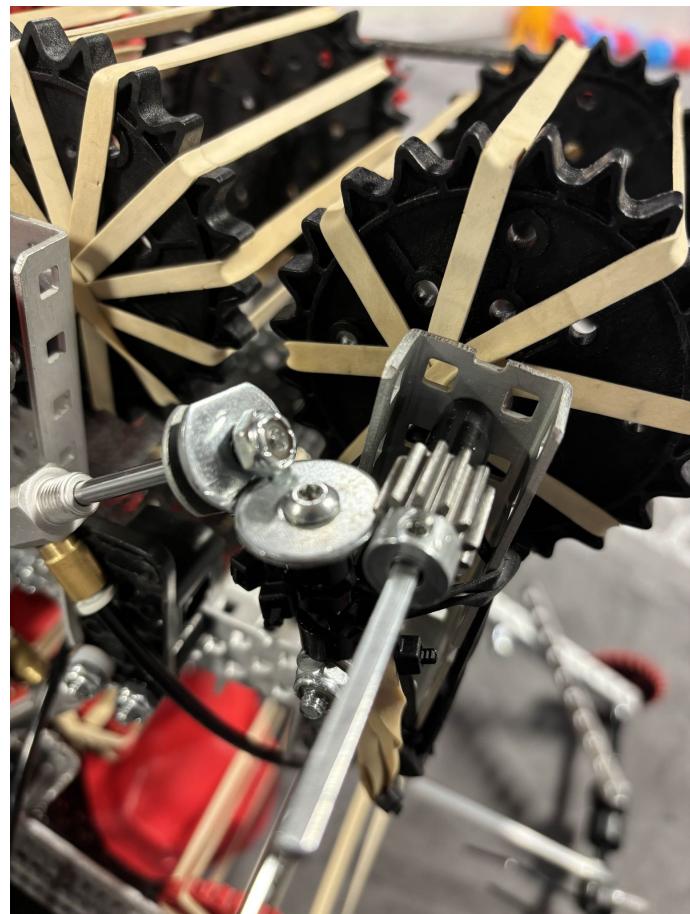
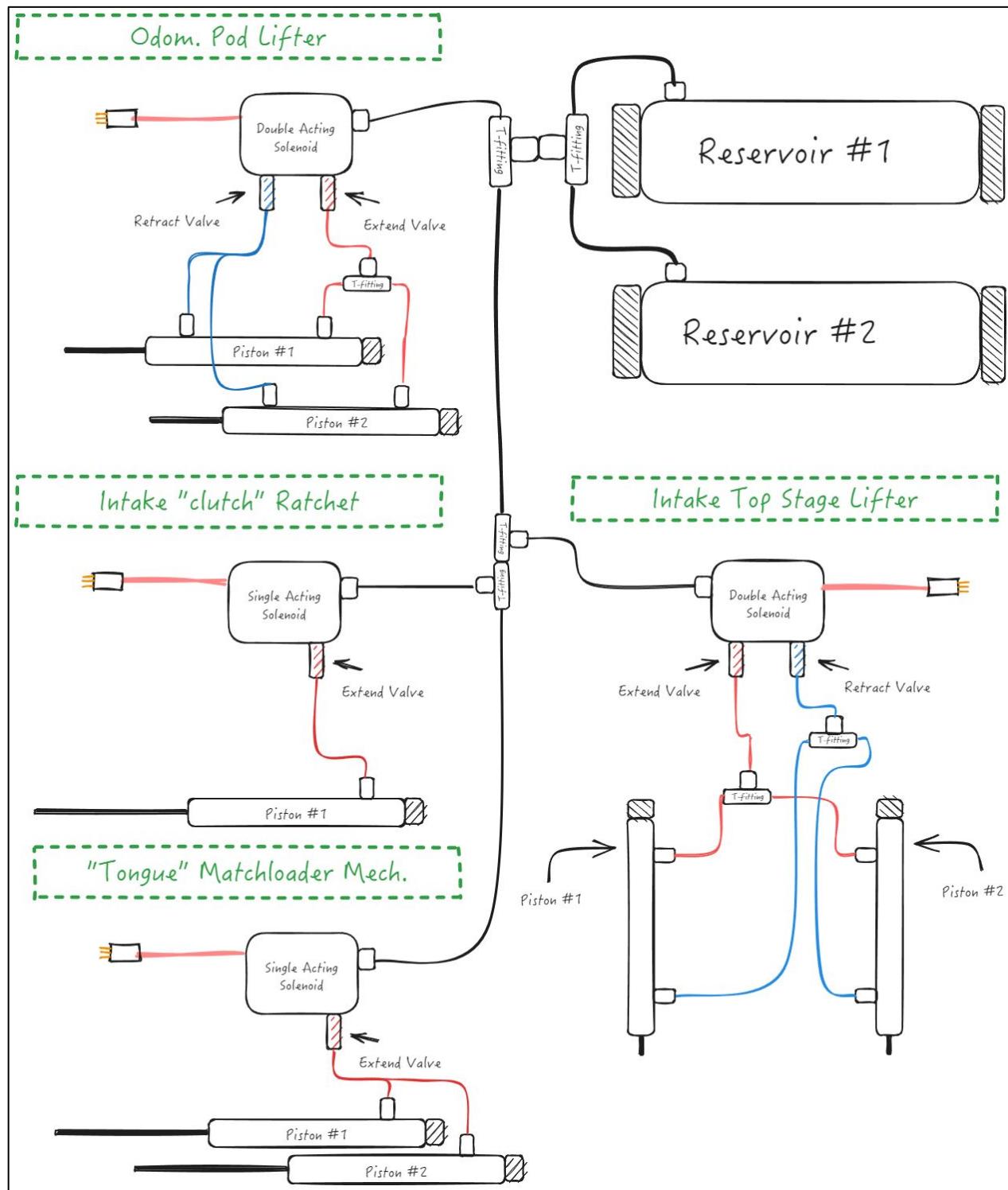


Figure 2: Picture of our ratchet engaged

► Focus: Visualize our pneumatics connections on our robot

Date: July 17, 2025

In the past, we made efforts to plan our pneumatics in a diagram so that we could make our tubing neatly organized. Now that our mechanisms are mostly finished, we wanted to lay out our permanent tubing paths. These diagrams were made with Excalidraw.



- **Problem:** List issues with traditional positioning methods.

Date: July 19, 2025

Our autonomous movement algorithms currently depend on **odometry**, a method of tracking a robot's position in 2D space via input from **external sensors**. Last year, we implemented odometry using direct feedback from the motors directly powering our drivetrain. While simple, this method does lead to **many problems**, which we've experienced firsthand:

- 1) The **motors are not reliably accurate** for reading rotational data, as they are also powering the drivetrain and disparities in temperature or speed may skew its readings.
- 2) If our robot gets stuck on a field element or runs into another robot, it won't move, but the drivetrain wheels will still turn. The robot's position in its memory will update since the odometry code reads from the motors, even if the robot is not physically moving.
- 3) Since tank-style drivetrains do not have any wheels perpendicular to the side of the robot, there is no way to detect direct **horizontal strafing** on the drivetrain.

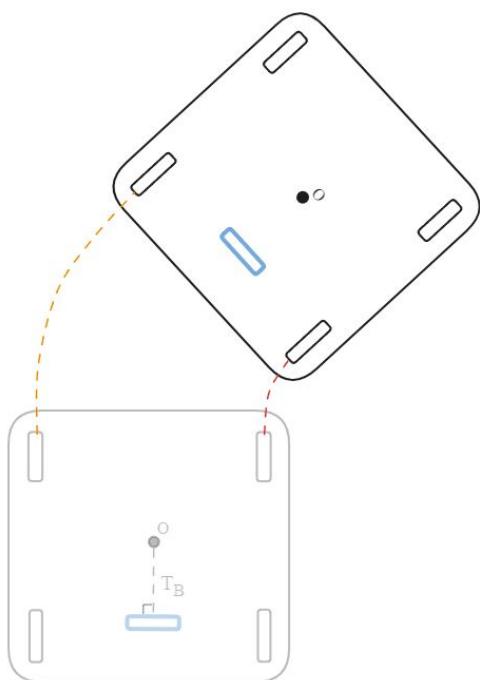


Figure 1: All-omni wheel drivetrains may drive horizontally during a movement. This drift is unaccounted for without horizontal wheels.

- ▶ Focus: Look into currently available tracking wheel designs.

Date: July 19, 2025

Our solution is simple; use **tracking wheels**, which essentially means to use another set of unpowered, free-spinning wheels exclusively for the purposes of doing odometry.

<u>Positives</u>	<u>Negatives</u>
<ol style="list-style-type: none"> 1. Highly accurate since they use rotation sensors 2. Can detect horizontal movement with the use of a perpendicular wheel 3. Only 2 extra tracking wheels are realistically needed 4. This is the only viable alternative to tracking position other than our previous method 	<ol style="list-style-type: none"> 1. Requires us to clear out space in the drivetrain to mount these wheels 2. The wheels have to be pulled towards the ground to ensure continuous contact 3. Needs to be lifted up with pneumatics if our robot were to hang 4. Takes up more Brain smart cable ports

Most of these cons can be addressed in the future, and they are outweighed by the significantly **increased accuracy** of using unpowered tracking wheels. From our research, it appears that tracking wheels are mounted onto a robot by encasing them within an “odometry pod,” containing the wheel along with a rotation sensor.

Here are some sources of **inspiration** we found while planning our pod design:

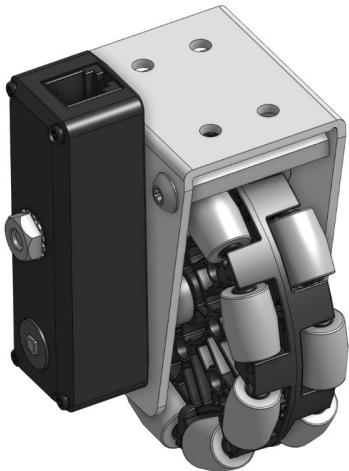


Figure 1: 1961D via Robolytics



Figure 2: 169B via Robolytics

- ▶ Focus: Select the construction materials for our pods.

Date: July 19, 2025

A screw inserted into a channel on the chassis can be used to mount the tracking wheel, allowing it to move up and down to account for elevation differences on the field mat.

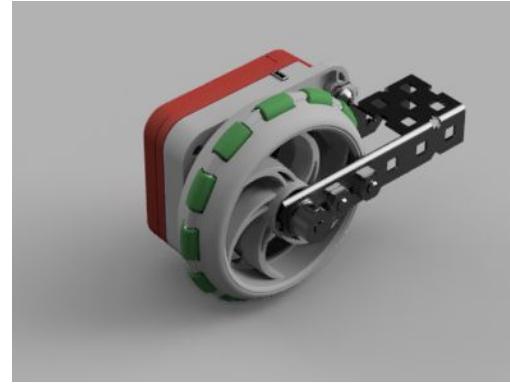
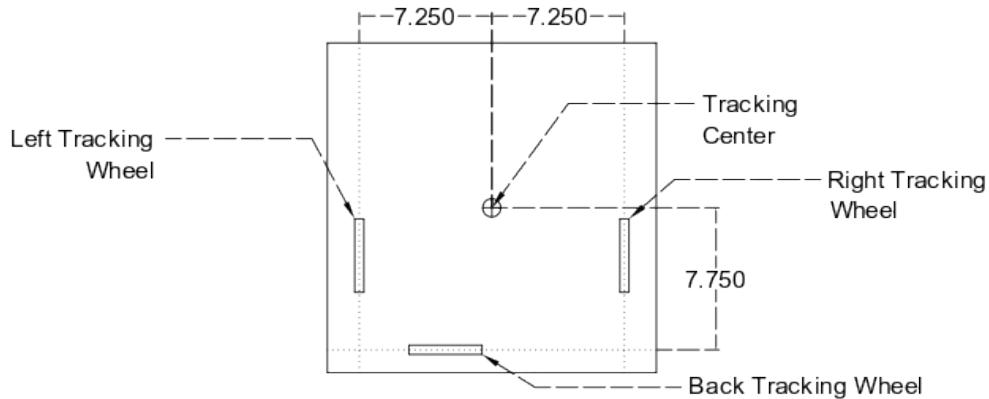


Image from team 24K (VEX Forums)

The structure of an odom pod can be made out of a cut metal channel or Lexan.

Metal odometry pods are stronger than lexan odom pods as they will **torsion and flex less**. Odometry pods can also be made by attaching a rotation sensor to the inside of a c-channel, reducing the need for any materials to “sandwich” the tracking wheel in-place.

We chose to make 2x custom odom trackers with 2" omni wheels, and to test various designs. One pod is to be made out of a cut 5-wide metal channel, while the other is to be made out of a screw-joint.

We wanted to test the c-channel design for:

- 1) Reducing friction between the wheel and sensor
- 2) Testing the strength and flexibility of the wheel axle

We wanted to test the screw-joint design for:

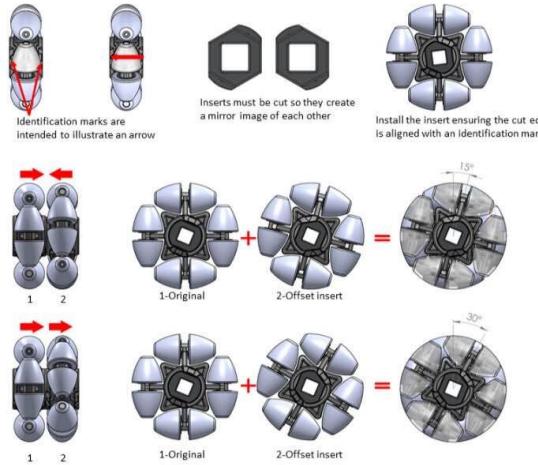
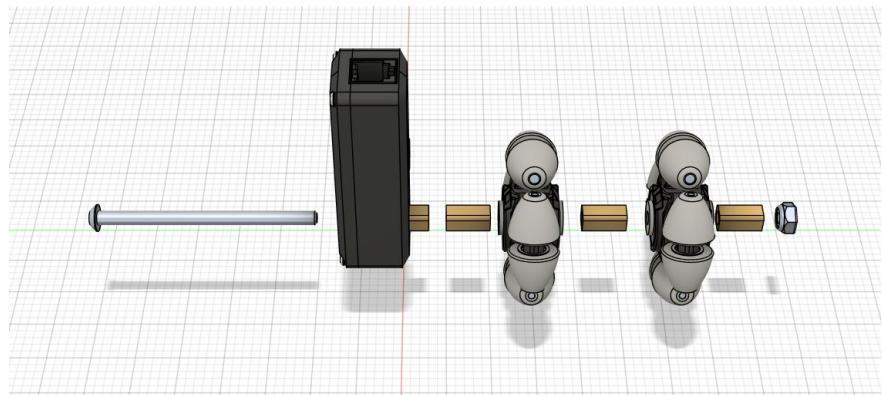
- 1) Ensure there is little slop/leeway in the pod and its mounts
- 2) Improve tracking accuracy by reducing friction thanks to the axle-less design
- 3) Shrink physical footprint to fit in cramped drivetrains

We're choosing to use **2" omni-wheels** for both designs because it won't block lateral movement as a perpendicular pod, and it takes up a small form-factor.

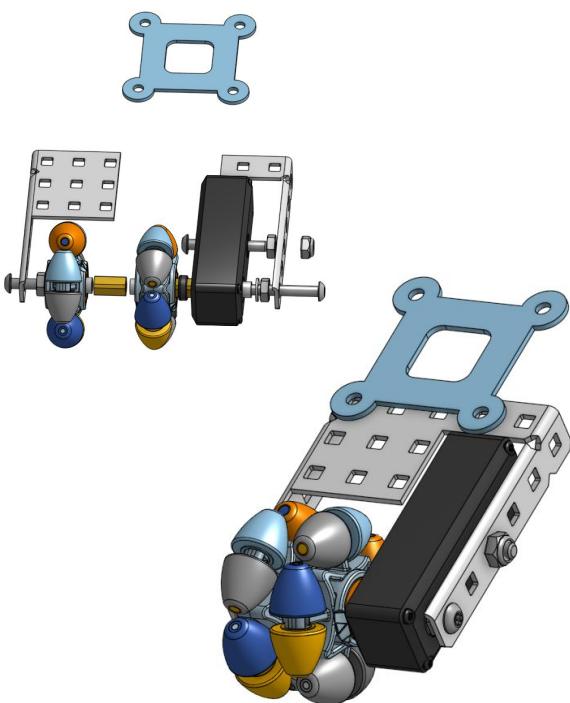
► Focus: Visualize how we made our odom pods.

Date: July 21, 2025

We made a **simple CAD** of the screw-joint odom pod (exploded view) to visualize the placement of parts. The screw and nylock will “clamp” the 2 wheels to the sensor’s rotating slot.



Additionally, we found a graphic that showed how to clock the two wheels’ orientation to make the shaft **rotate smoothly** on the ground (given the omni-rollers are uneven on the ground).

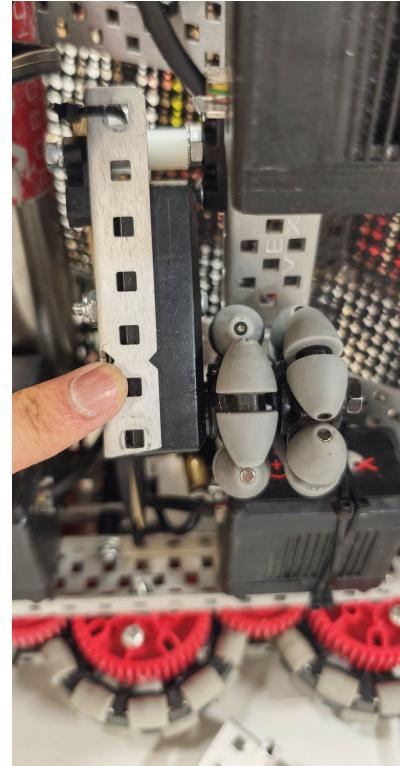


We also designed our c-channel sandwiched odom pods. The blue part represents a flexible, springy **Lexan** material that will ensure the tracking wheel makes constant contact with the field mat. This design is also screw-jointed, however the screw(s) are supported from both sides of the metal channel to reduce friction.

- ▶ Focus: Visualize how we made our odom pods.

Date: July 21, 2025

One of the **assembled pods** is pictured, along with its mount on our chassis as a tracker for horizontal movement. To see how it is used in our autonomous, please visit our **programming logbook**.



Additionally, we also manufactured the “**spring leaf**” metal odometry pod. Here’s what both of them look like on the robot:



Odometry Pod Testing

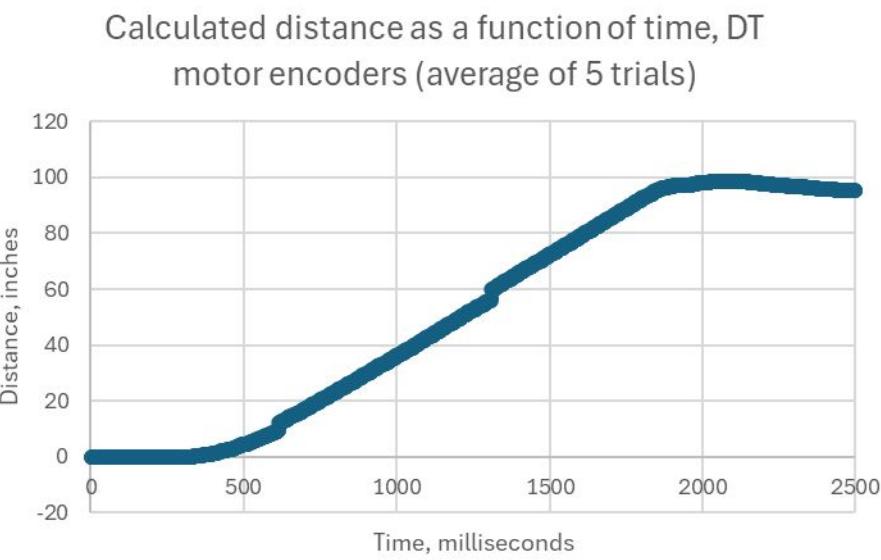
Test

- **Problem:** Verifying the accuracy of our odometry pod compared to motor encoders.

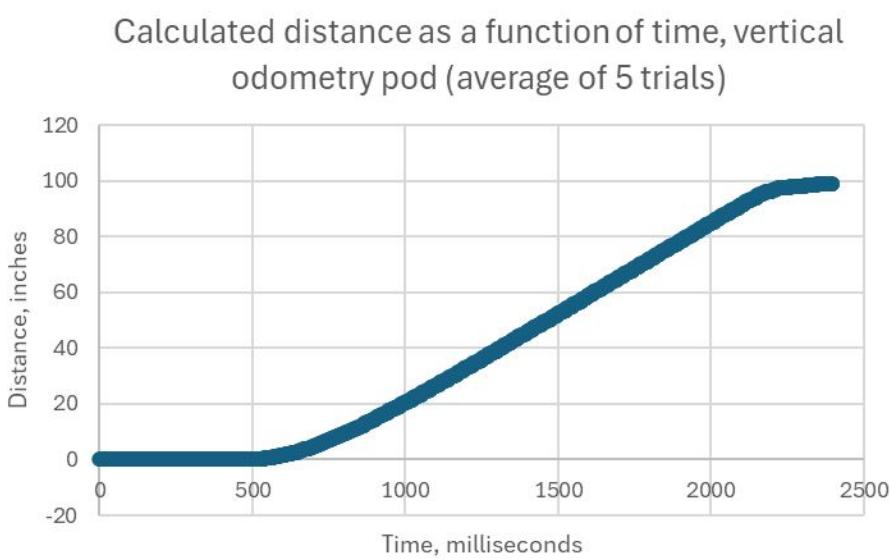
Date: July 22, 2025

To test the **accuracy** of our tracking methods, we had our robot drive forward for 100 inches using a **PID controller**, recording the instantaneous position measured with both motor encoders and our new odometry pods.

*Note: we actually recorded the rotations made by the encoders & rotation sensor respectively, then multiplied them by their wheel circumference to calculate the distance travelled



For this **traditional** tracking method, notice the two **jitters** in the graph (likely wheel slip or motor lockups) and how the position **drifts** even after reaching its target position past 2000ms



The new odometry pods had a much **smoother distance curve** showing **linear velocity** once the robot accelerated to maximum speed. The graph also showed **no indications of rotation drifting** past its target position.

- ▶ Problem: Odom pods getting stuck

Date: July 22, 2025

Odom Pod Issues

When practicing our skills paths and parking, we noticed that our **horizontal odom pod** would clip into the park barrier and would **bend**. This problem would extend to autonomous as our pod would give **inaccurate readings**. Our odom pods bending would also result in our **drivetrain** becoming less smooth as it would deal with the friction caused by dragging the pod along the field tiles.



Figure 1: 2775V's Odom pods in Over Under. These pods needed to be lifted in order to cross the barrier

- ▶ Focus: String system to lift our pistons

Date: July 22, 2025

Piston Lift

In order to solve the issue of our pods getting stuck, we looked at previous games with different elevations, VRC Over Under and Spin Up. In both of these years, teams utilized piston lift that utilized a piston that would help lift their Odom Pods off the ground once they are no longer needed. During driver control and driver skills, we **would not be using our odometry pods** as they are tracking wheels used for autonomous routines. These pods are banded to always be contacting the ground and it **causes issues** when we try to enter the **park zone** as they will hit the park barrier. To solve this issue, we used a **pulley system**, utilizing fishing line and a piston to help **lift up our odometry tracking wheels** off the ground.



Figure 1: Another example of an Odom Pod (YouTube, 21919A)

- ▶ Focus: Lifting our odometry tracking wheels during driver control

Date: July 23, 2025

Members Involved

Daniel

Objective

The objective of today is to create a **fishing line** system that could **pull both odom pods up** using one piston so they could clear the park barrier.

Materials

- Fishing line
- Pulley
- Piston
- 1, 1x1 angle bar
- Nylock nut
- Hex nut

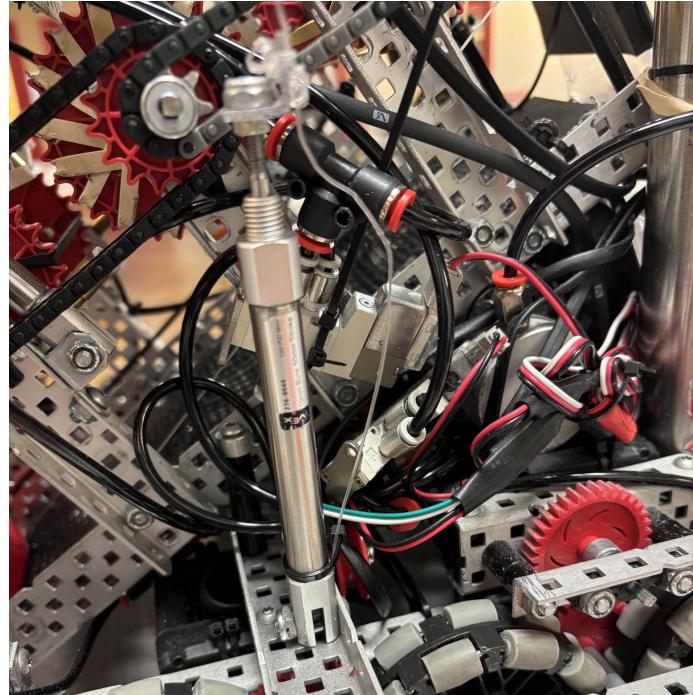


Figure 1: Piston in retracted position

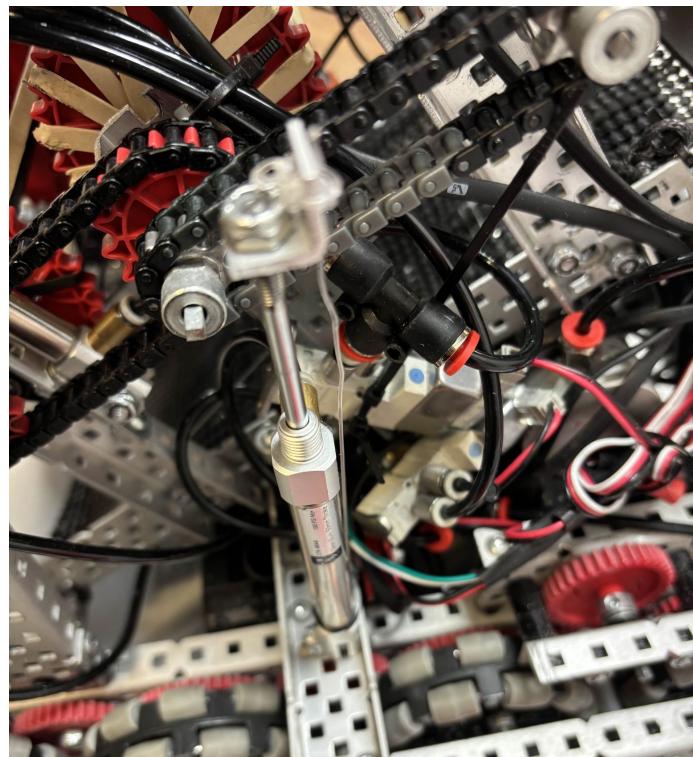


Figure 2: Piston in extended position

- Focus: Lifting our odometry tracking wheels during driver control

Date: July 23, 2025

Problems and Solutions

We noticed that the **fishing line** was falling off the pulley.

To solve this issue, we learned how to tie the **bowline knot** to maintain a good tension and prevent the fishing line from slipping.

The piston would not fully **lift the odom pod** off the ground.

We fixed this by running the fishing line in a way where it maximizes our mechanical advantage.

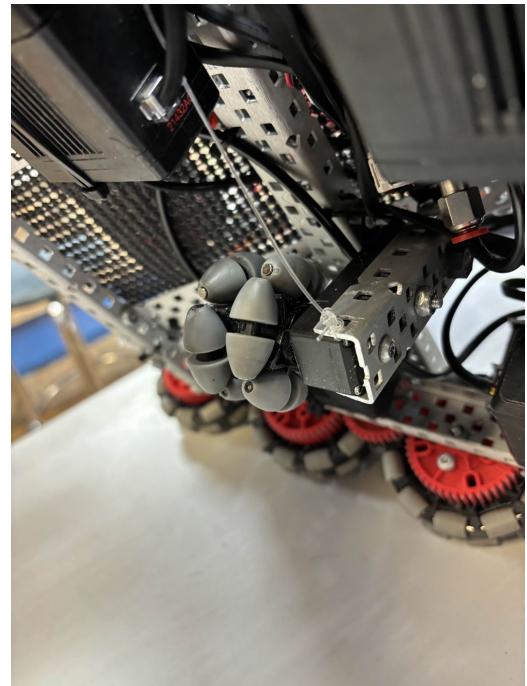


Figure 1: The odometry pod in the down position

Next Steps

Testing various parts of our robot.

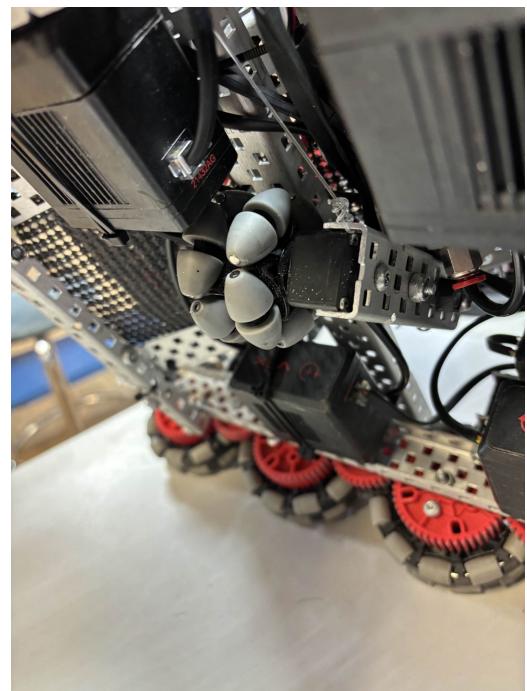


Figure 2: The odometry pod in the lifted position

- Focus: Analyzing skills

Date: July 24, 2025

What is a Skills Challenge?

Skills challenges is a 1v0 match where a team will try to score as many points in a **60 second time period**. Teams are able to score points through **completing different objectives**.

There are two types of skills challenges as stated in the VRC Push Back Game Manual.

1. Driver skills

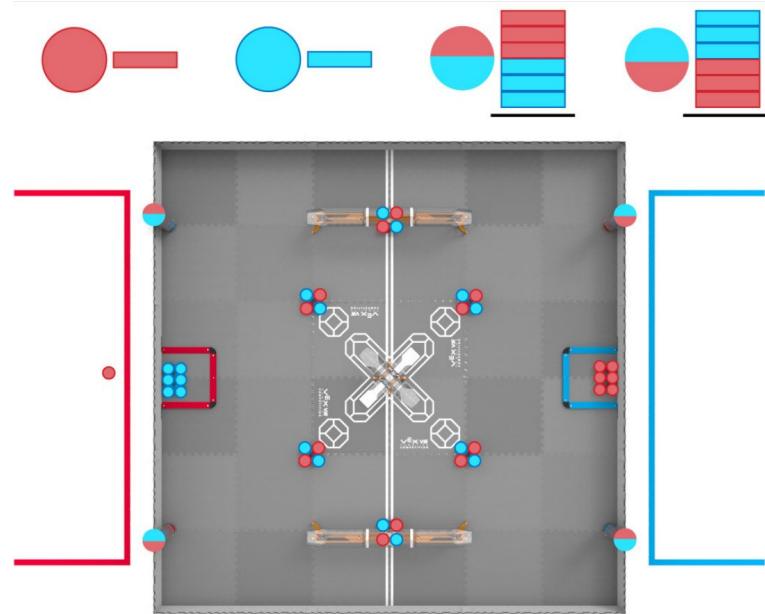
- 60 Seconds of driver control with no autonomous period.

2. Autonomous skills

- 60 seconds of a pre-programmed autonomous routine with no driver control

Skills Setup

As stated in rule **RSC3** as well as being shown in figures **RCS3-1** and **RCS3-2** in the game manual, we are able to see a **unique skills setup** as opposed to a head to head match. There are **36 blocks** in unscored positions **on the field** and **24 blocks** in the **matchload tubes**.



Audience View

Figure 1: Top down view of the skills setup (VEX, Game Manual)



Figure 2: Skills rankings at a previous event we attended in 2024

- Focus: Skills scoring breakdown

Date: July 25, 2025

Skills scoring is defined under **<RCS2>** in the game manual. It gives us a unique set of **scoring rules** with different objectives that can be completed for bonus points.

1. Despite there being both blue and red blocks on the field, **all blocks scored count for points**
2. Control zones will count for bonus points if they are filled with to the maximum with the **same colour**
 - a. **Long goals** will require 3 of the same colour in the control zone
 - b. **Central goals** will require 7 of the same colour to get the point bonus
3. **Clearing a park zone** requires a team to remove all blocks contracting the inside tile of a park zone. The team will gain **bonus points** if the zone is cleared
4. Each **cleared matchloader** zone will gain a team **bonus points** as each matchloading tube will be filled with 6 blocks at the start of every skills match.
5. A team will also gain **bonus points** if a robot is **parked** in the red parking zone at the end of a match

Objective	Points awarded
Each block scored in a goal	1
Each filled control zone in a long goal	5
Each filled control zone in the center goal	10
Each cleared loader	5
Each cleared park zone	5
Parked robot	15

Table 1: Skills scoring breakdown (VEX, Game Manual)

- Focus: Basic skills path

Date: July 26, 2025

After looking at the skills scoring breakdown, there were a few **key objectives** that need to be considered when planning out a path.

1. Matchloading tubes
 - a. The loader tubes will provide the robot with **6 blocks** while giving bonus points when cleared
2. Control zones
 - a. These **bonus points** can be gained while scoring blocks into any tube
3. Parking
 - a. Parking a robot gives a team **15 extra points** for driving into a location which **cannot be given up**

When planning our **first path**, we took these consideration into account. We decided to create a **very basic path** that would score **70 points minimum** but could be raised to 80+ if we chose to score more balls. The long goals can hold a **total of 15 blocks** with the center 3 blocks making up the control zone.

Therefore, there can be **6 blocks on either side** of a long goal control zone. Meaning to **guarantee a control zone**, we need to score a **minimum of 9 blocks**. Our first path is meant to be very **simple and repeatable**, providing a **high but consistent** base skills score during competitions.

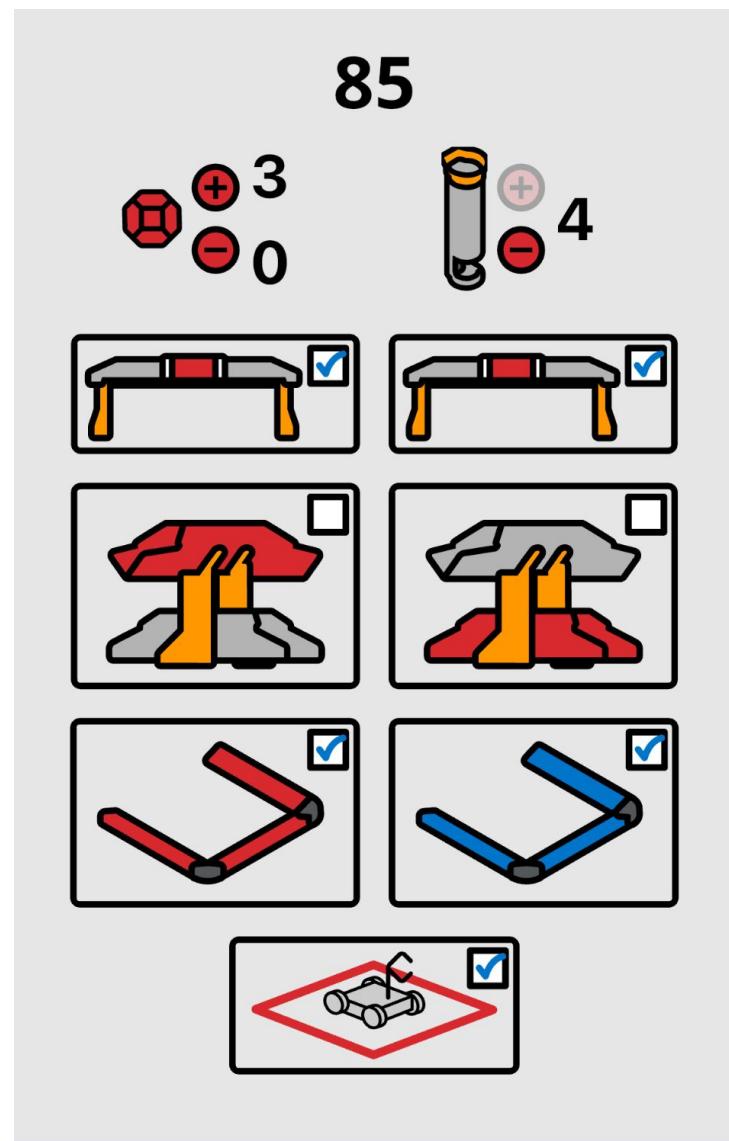


Figure 1: Skills score breakdown shown in a calculator app (V5RC Hub)

► Focus: Basic Skills Path

Date: July 26, 2025

Scoring breakdown:

In this skills path, our **general plan** was to drive a circuit while clearing as many 5 point objectives as we could. This path clears **4 loader tubes**, **2 parking zones** and scores a total of **30 blocks** into 2 long goal control zones. This makes for a total of **85 points**.

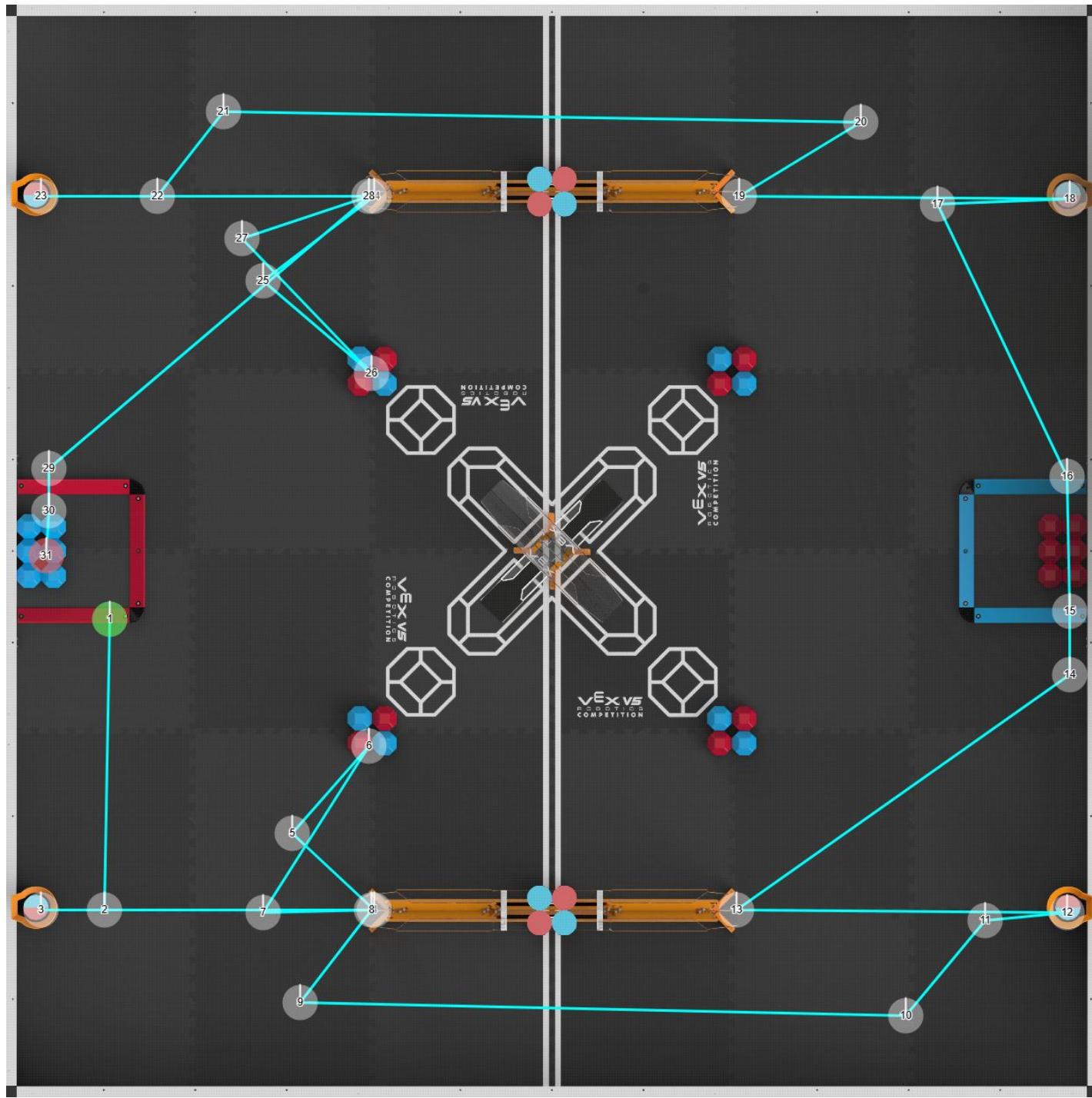


Figure 1: Skills path created using our own path planner

Skills Practice

Skills Runs

► Focus: Practicing the basic route

Date: July 27, 2025

Attempt #	Skills Type	Score
1	Driver	53

Notes

We managed to clear **3 matchloaders and 1 park zone**, as well as **park successfully and score 13 blocks**, filling a long goal control zone. Mostly, we ended up **running out of time**, which led us to missing a matchloader and park zone. However, this can be fixed with **practice**.

Attempt #	Skills Type	Score
2	Driver	64

Notes

This run was more successful because we managed to clear another **matchloader and park zone**. We also scored **14 blocks**, but this number can be improved. We decided to focus more on getting both **control zones** filled.

Attempt #	Skills Type	Score
3	Driver	71

Notes

This was our **personal best**. We cleared **all four matchloaders and both park zones**, as well as **park successfully**, score **21 blocks**, and fill **one long goal control zone**. Moving forward, we should try to fill both long goal control zones.

► Focus: Creating a more complex skills routine

Date: July 28, 2025

After practicing our basic path, we decided to plan out a **riskier** and less consistent path as our interaction with blocks may send them **rolling in different directions**. In theory, this path can score a total of 94 points.

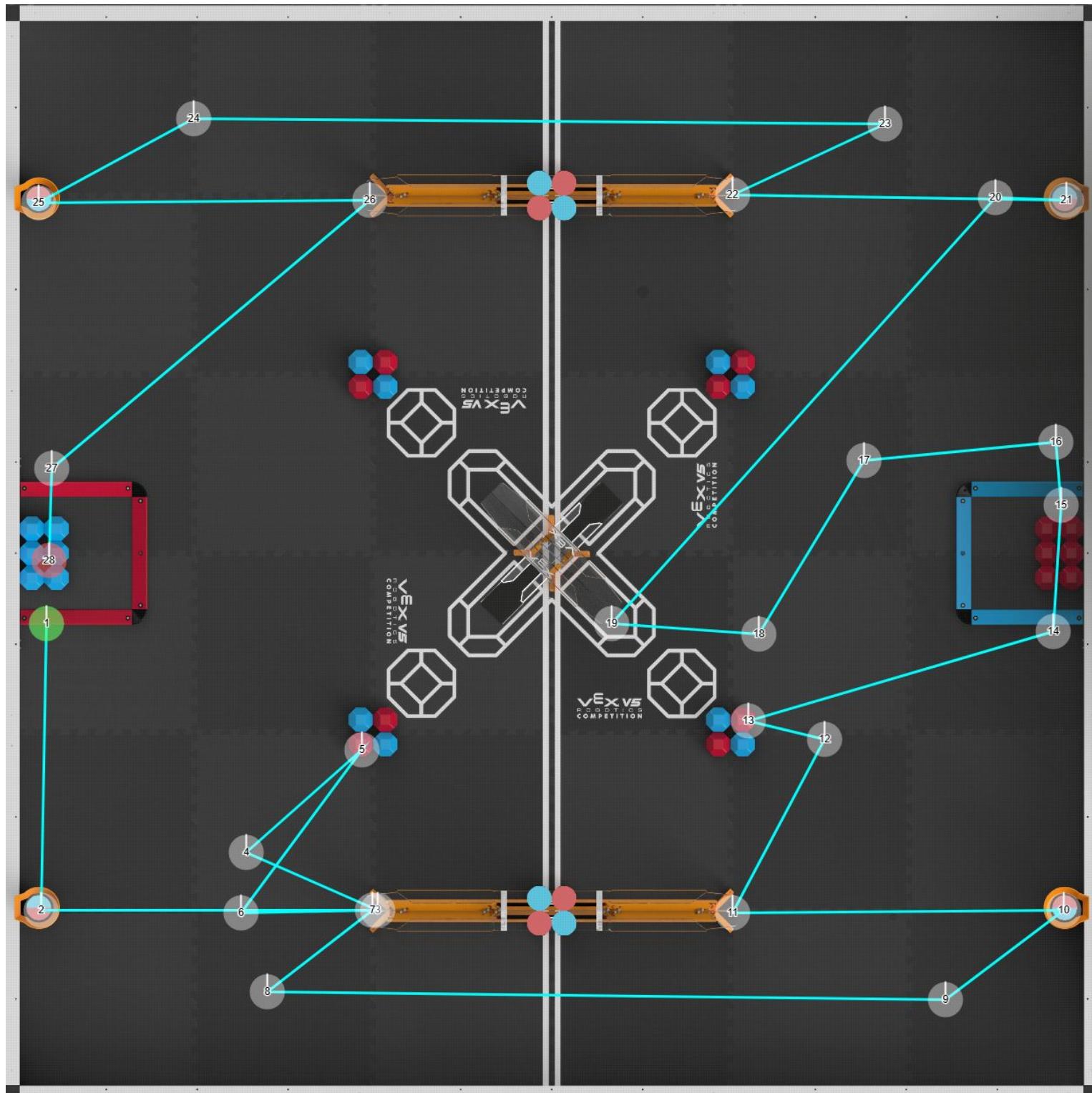


Figure 1: Skills path created using our own path planner

► Focus: General strategies

Date: July 29, 2025

Win Conditions

Every game in previous years has had objectives that can help teams win matches. Every year has had multiple key objectives and the alliance that typically holds more of them will win the match. Below are some of the **crucial scoring objectives** that we have found to be important and will focus on **this year**.

1. Long tube control zones

- a. Each of the control zones are worth **10 points**, which requires **4 extra blocks** scored by the opposing alliance to make up for this point deficit
- b. Maintaining control zones also requires scoring blocks, with a minimum of **8 blocks** needed to be scored from one side of the tube to maintain control of the tube, holding a control zone is at minimum **34 points**

2. Central tube control zones

- a. Although the center goals can be easily **contested** and slightly harder to score on, it is still important to **maintain pressure** by scoring on these goals. If a center goal has 1 block in it and is left uncontested, that is still either a **+8 or +6 point bonus** for an alliance.

3. Double park

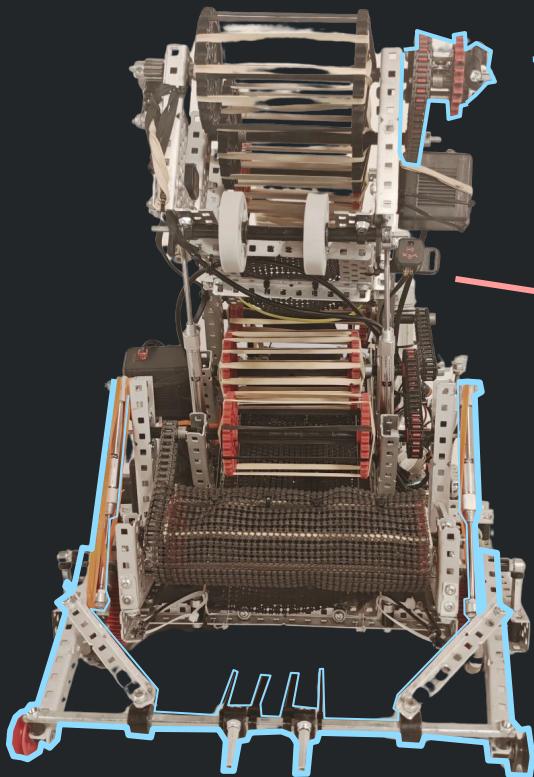
- a. Having both robots park is hard to achieve but still a massive **30 point swing**. If a double park can be achieved consistently, it puts a lot of pressure on the opposing alliance to make up for this point deficit

4. Block control

- a. Having blocks of your colour in the robot **always ready to score** is crucial as you do not need to **waste time to get match loads or intake blocks**. Holding blocks can create **opportunities for reactive plays** and forces your opponents to play defensively or risk getting their blocks pushed out of the tubes.

Design Overview

This is for bot iteration V1.0



“Clutch” indexing mechanism

Prevents jams from propagating and allows us to selectively score blocks using a piston-powered ratchet

276-4852

Forward-facing distance sensor

Used in autonomous position resets and localization

Drop-down “tongue” mech

Quickly descores matchloaders (2 secs) and snatch hard-to-reach blocks during autonomous

228-2500-230

VEX 6P chains

Less prone to snapping and more flexible around sharp bends than regular chain

276-4831

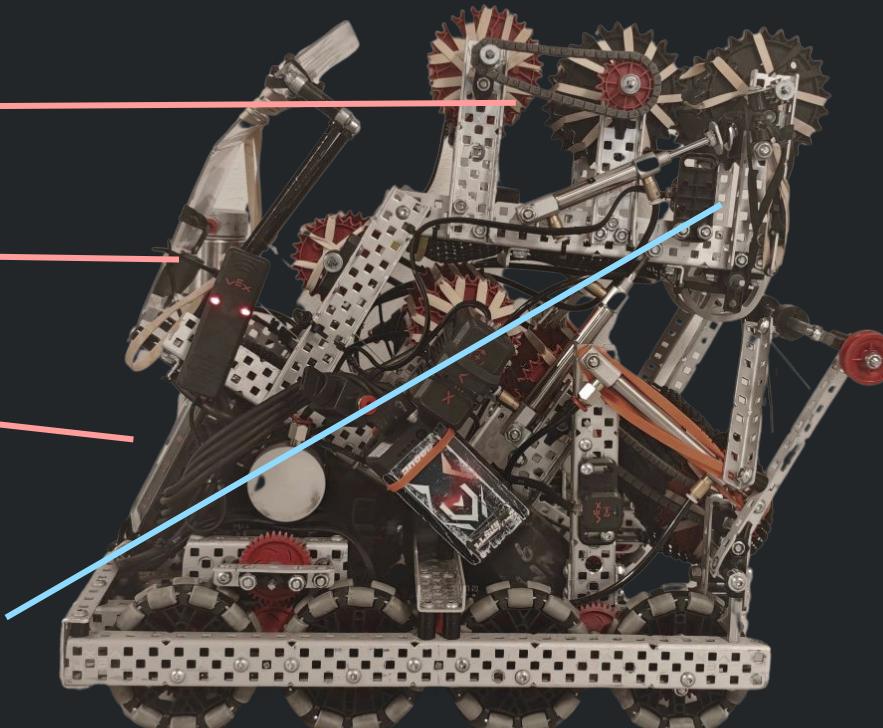
VEX V5 Radio

276-4810

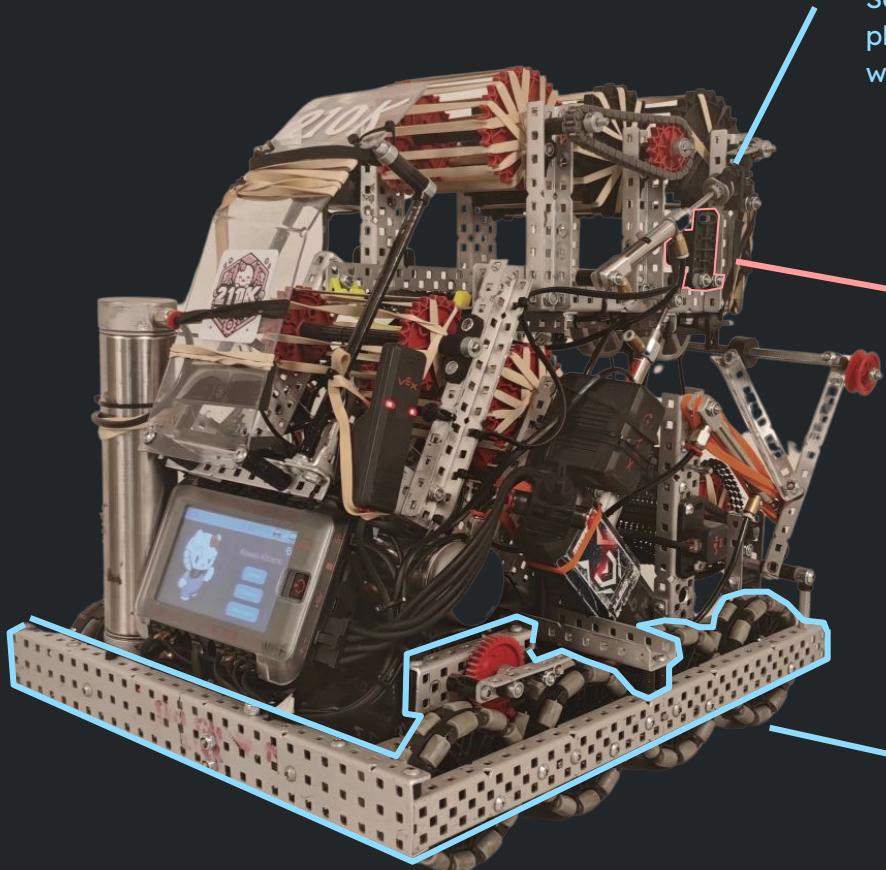
VEX V5 Brain

“Snail” conveyor intake

Chained set of rollers that store blocks and swiftly move them to the upper stage to be scored



Design Overview



“Clutch” ratchet piston

Selectively holds the final intake roller in place, allowing us to score multiple balls with the intake still running

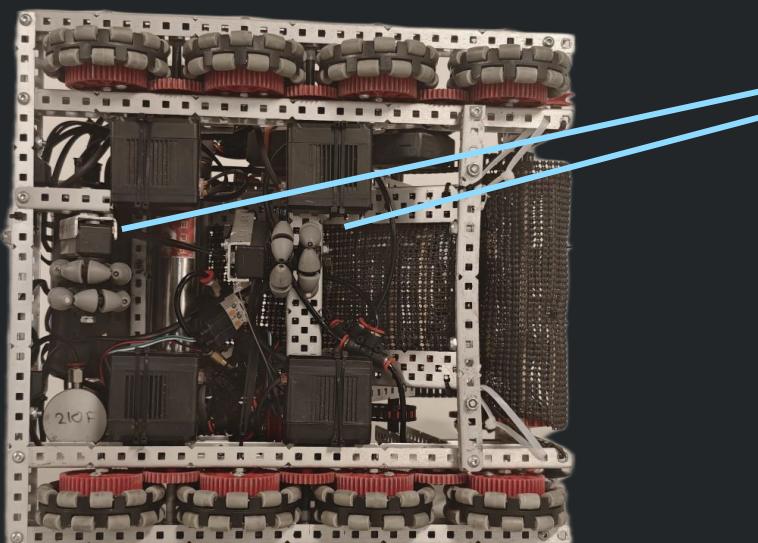
276-7043

Optical Sensor

Verifies the colours of blocks as they pass by the intake to ensure we never score for the opposing alliance

3.25" 450RPM Drivetrain

Speedy enough to outmanoeuvre opponents while still having strong traction force and resist pinning.



Dual-axis 2" Odometry Trackers

Tensioned free-spinning wheels that always touch the ground to measure the robot's travel laterally and horizontally. These pods can be lifted when needed (e.g. driving over the park barrier)

- Focus: Setting goals and planning for the MOA signature

Date: July 29, 2025

The **University of North Dakota Signature Event** (otherwise known as **MOA** for Mall of America), is a **117-team signature event** held in **Bloomington, Minnesota**. It will be held from July 31, 2025, to August 2, 2025. Our flight is on the 31st, where we will arrive in the morning to do **check-in, inspection, and skills** later in the day. The second day and morning of the third day will consist of eight **qualification** matches, then **elimination** matches in the afternoon of the third day. As MOA is the **first signature event** of the season, there will be many different designs and strategies to learn from.

Skills Goals

Driver Attempts	Driver Highscore	Programming Attempts	Programming Highscore	Total Score
3	71	3	45	116

At MOA, we should aim to **use all of our skills attempts**, especially because there are no qualifications on the first day. We will only stop using programming attempts if we have a skills run that hits **perfectly**. Our driver hit a **personal best of 71** in practice, so we will aim to hit the same or a similar score at the tournament. Since we are **focusing more on our autonomous routines for competition**, we will not expect to reach the same as our driver skills score for autonomous. Thus, we think 45 is a reasonable goal.

Overall Goals

Our **overarching goal** is to receive a **worlds qualification** at the event, by winning Tournament Champions or the Excellence Award. However, this is very difficult to achieve, since many **strong teams** will be attending. Below are more attainable goals, of which we hope to achieve (keep in mind that there is only one division at MOA):

- Skills score of **~116**
- Making it to the **quarterfinals** round
- Ranking **20 or higher** in qualifications
- Winning a **judged award**

In conclusion, although we do not expect to receive a worlds qualification, we do hope to reach the goals set above.



	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	Q7	2502X 2145V	17	210K 8823A	100	Yes	No

Notes

- Solo-AWP routine was not consistent

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Loss	44	210K 9144C	23	8110S 334U	124	No	No

Notes

- DC During autonomous routine
- Robot radio disconnected ~20 seconds into driver control
 - Cause was traced to the battery cable, which was replaced

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	80	210K 2550R	64	96671V 99157B	39	Yes	Yes

Notes

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
		2083Z 8110R	6	16689A 210K	98		
	97					Yes	Yes

Notes

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
		5013X 4118D	38	210K 7225T	95		
	123					Yes	Yes

Notes

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
		210K 8110W	70	10085A 94S	60		
	149					Yes	Yes

Notes

- Opponent 10085A came from the Chinese mainland, and is the first non-NA team we have played against as a team

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?
Loss	193	4610Z 88909Y	78	2131N 210K	53
				No	No

Notes

- Autonomous failure

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?
Loss	212	210K 99905C	58	2787V 1532D	73
				No	No

Notes

- Autonomous failure again :(

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?
Win	252	3239V 210K	100	99319E 8823Z	37
				Yes	Yes

Notes

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
	266	9257F 94W	26	210K 77360Z	122	Yes	Yes

Notes

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
	R16	210K 3004A	80	10009A 886S	71	No	N/A

Notes

- Though we lost autonomous, we were able to make a narrow comeback this match through our defence strategies and park advantage

Loss	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
	QF	11101B 9123X	120	210K 3004A	27	Yes	N/A

Notes

- We came into this match with our expectations down, as we were playing against two organizations who had won Worlds in the past and are revered in the VEX community
 - Despite this, we still tried our best and even won the auton bonus, before DCing during driver control

► Focus: Post-MOA assessment

Date: Aug 3, 2025

We ended up hitting two out of four of goals. We:

1. Ranking 13th during qualifications
2. Made it to the quarterfinals round, allied with 3004A QWERTY
3. Scored 2 points behind our driver skills goal
4. Unfortunately had a non-working skills autonomous routine, leading us to a lower score
5. Did not win any awards

Skills

Rank	Driver Attempts	Driver Highscore	Programming Attempts	Programming Highscore	Total Score
15	2	69	2	20	89

We did achieve our goal of a driver skills run around 71, but our autonomous program underperformed. However, our score was still enough to bring us to 15th, which was higher than we anticipated.

Qualifications

Rank	Team	Name	W-L-T	WPs . APs . SPs
13	210K	Kawaii Kittens	7-3-0	19-65-357

Qualifications went very well for us. Our autonomous win point hit 5/10 times, which was consistent enough to bring us to our goal ranking of top 20. Most of our autonomous failures were on the second day of competition, so we should note to test and tune our autonomous routines before each day of competition.

Summary

Overall, the tournament was still successful, as we ranked higher than we expected and still made it to quarterfinals. Our quarterfinals loss was caused by an unfortunate disconnect, and also playing against two of the best teams at MOA.

Next time, we will make sure to prepare for auto skills more thoroughly, use measures to avoid disconnecting, and prepare for our interview beforehand.

V2: Local Event Bot

- ▶ Problem: Park barrier considerations following MoA

Date: Aug 3, 2025

At the Mall of America signature event, we identified that our robot design and match performance were limited primarily by efficiency issues. Across both our skills runs and head-to-head matches, several recurring inefficiencies reduced our competitiveness:

Center of Gravity & Barrier Crossing

Our high center of gravity caused instability when attempting to cross the park barrier. On multiple occasions during matches, we came close to tipping, which forced us to slow down significantly. This cautious driving reduced the speed of our cycles and made barrier crossing an inefficient way to reposition compared to our competitors.



Figure 1: Us at MOA crossing the red park barrier during our driver skills routine.

- ▶ Problem: Park barrier considerations following MoA

Date: Aug 3, 2025

At the Mall of America signature event, we identified that our robot design and match performance were limited primarily by efficiency issues. Across both our skills runs and head-to-head matches, several recurring inefficiencies reduced our competitiveness:

Front-Only Intake and Scoring

Because our robot could only intake and score from the front, we were forced into longer, less direct cycle paths. This resulted in slower cycle times during driver control and inefficient autonomous paths, often making it difficult to keep pace with opposing alliances. From a team standpoint, this design choice consistently reduced the fluidity of our gameplay.

Overall Team Efficiency

Collectively, these factors resulted in an overall lack of efficiency in both skills and matches. As a team, we observed that even when our strategies were sound, the execution was hindered by these design and performance inefficiencies. This meant we were unable to fully capitalize on scoring opportunities, limiting our match outcomes.

► Problem: Overall robot design considerations

Date: Aug 3, 2025

Here is a summary of the issues with the V1 robot at MOA:

Drivetrain

We had no noticeable issues with the design of the drivetrain, so it will stay mostly the same.

Intake

The intake had enough storage for cycling, but we may desire more. Also, a front-to-back intake might be preferred. That way, the robot can directly back up from the matchloader into the goal to score. We needed to use air to lift our intake, using up our limited supply and forcing us to use two air tanks, making our robot heavier. We will look for a design that does not use pistons to switch between the long and middle goals. Also, the indexing method we used with the clutch introduced a lot of friction to the top stage. We will use a different method to index the blocks within the intake.

Parking

We were able to cross into the park zone just fine, but we needed to take up much of the space to double park, meaning we did not park at all at MOA. For the V2, we will find a way to double park using less of the park zone space (eg. that our alliance might be able to use up most of the park zone).

Matchloading

We often had issues matchloading during the tournament, as blocks would not roll out but stay within the matchloader. We will consider a different design for the V2 robot.

► Focus: Selected drivetrain

Date: Aug 14, 2025

For this season, we decided to continue using our **established drivetrain configuration**, as it has consistently proven reliable and effective in past designs. Our team has extensive experience with this drivebase—having implemented it successfully in **five of our previous robots**—which allows our driver to perform at maximum efficiency with minimal adjustment time. Maintaining a familiar drivetrain ensures that the driver's control precision and reaction speed remain consistent throughout the season.

The drivetrain offers an excellent **balance between speed and torque**, which is especially important given this year's field layout and the presence of **barriers and elevated zones**. The gear ratio and wheel configuration provide enough torque to overcome obstacles while still retaining competitive cycle speed for rapid scoring and repositioning.

By reusing and refining this proven design, we can focus our development time on **optimizing other subsystems**, such as the scoring mechanisms and hoarding system, without needing to troubleshoot drivetrain performance. This strategic choice not only saves build time but also enhances reliability, as the drivetrain's performance characteristics are already well-understood by the entire team.

Overall, this drivetrain selection represents a balance of **driver familiarity, mechanical efficiency, and adaptability** to the 2026 game's unique challenges—making it the ideal foundation for our robot this season.

- ▶ Focus: Considering different intake designs

Date: Aug 16, 2025

Alternative Intake Designs

At MOA, we saw many different new robot designs aside from the ones we had already seen. Here are a few we saw that seem promising:

Hoard

The hoard design attempts to take advantage of the fact that there is no possession limit in Push Back. The majority of the robot is made up of a large storage section made of mesh. Rollers in the front can move the blocks in and out of the storage, as well as back out the front, into a center goal height or a long goal height. The main advantage of this design is the lightweight structure and large storage capacity. However, it cannot cycle blocks as quickly as some other designs, which makes it struggle more to be competitive.

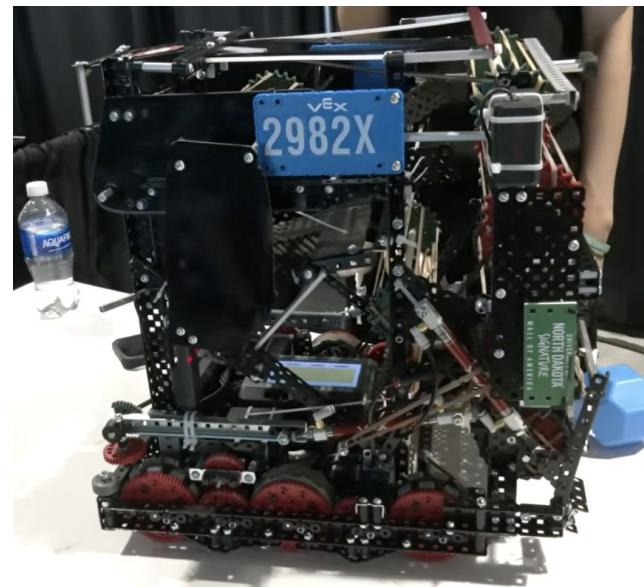


Figure 1: A hoard robot we saw at MOA by 2982X

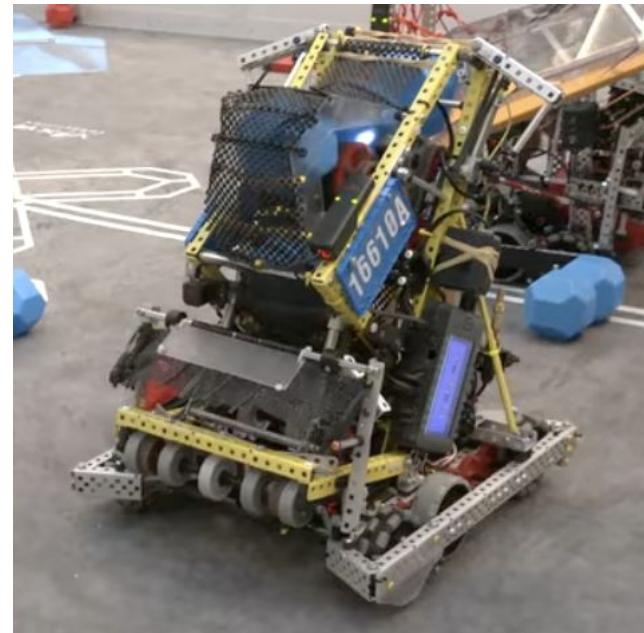


Figure 2: An S shape robot we saw at MOA by 16610A

- ▶ Focus: Considering different intake designs

Date: Aug 16, 2025

AK Bot (Linear)

This is the ultimate cycle time intake. The blocks move straight from the front to the back, such that the blocks have very little distance to travel. A storage of only 6 blocks is needed, enough for the robot to fully empty a matchloader tube. A trapdoor near the back can also pivot downwards to funnel the balls into the top center goal as well.

Notes on Designs

- Some intakes can colour sort more effectively than others. For example, the S shape can easily funnel blocks out through the back, but the hoard might need to store the undesired colour blocks, forcing our driver to empty the intake after scoring.
- Cycle time is strongly affected by the direction that the intake scores in. When the robot scores directly from the front to the back, it can simply back up from the matchloader, rather than having to turn around and score.
- Compactness actually does not matter much, since the most of the other systems are built *around* the intake. It will affect storage space more, which is why we did not include it as a criterion.

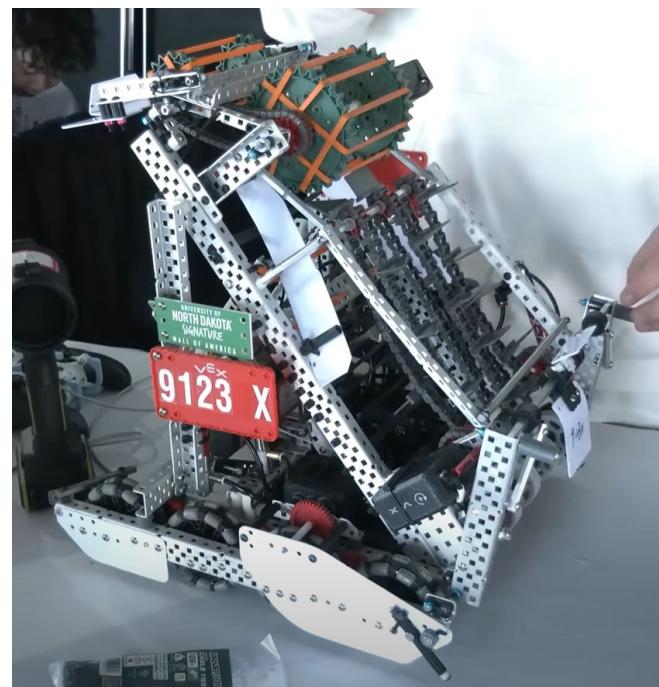


Figure 3: A style of linear bot we saw at MoA (9123X)

- Focus: Selecting a suitable intake design

Date: Aug 16, 2025

Intake Criteria

1. **Cycle Time (1 - 5)** - The theoretical amount of time needed for the intake to take 6 balls and score them
2. **Weight (1 - 5)** - How much weight would the intake system add to our robot? A lower center of gravity is ideal to avoid tipping over.
3. **Storage (1 - 5)** - Ranked on how large of a storage we are able to have. Having a larger storage will help us pick up blocks from the field. This allows more flexibility on when we decide to use match loads.
4. **Simplicity (1 - 5)** - We want to avoid overly complex designs. So choosing to build something simple allows us to streamline the building process and allow for easy maintenance.

<u>Criteria</u> <u>Options</u>	Cycle Time /5	Weight /5	Storage /5	Simplicity /5	Total /20
Hoard	2	4	5	4	15
S Shape	4	3	4	4	15
AK (Linear)	5	4	2	5	16

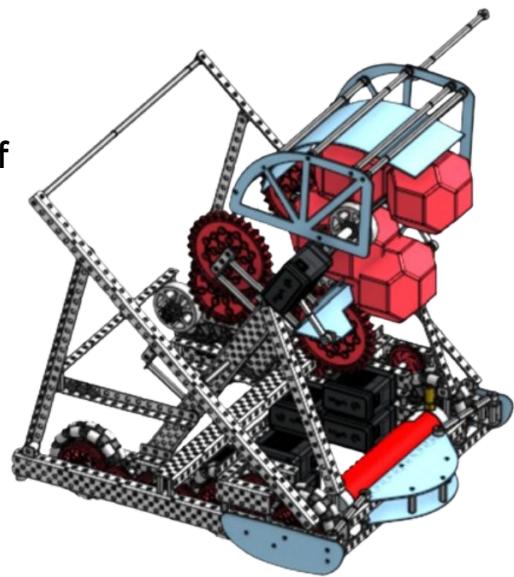
All of the designs are well matched. The linear robot trades storage for cycle time, and the hoard is the opposite. The S shape serves as a middle ground. None of these designs particularly stood out to us, so we decided to design something new by effectively combining the hoard and the linear design (refer to pg. 186).

- Focus: Finished 3D Model of new Robot

Date: August 20, 2025

Today, we completed the CAD design for our robot, which features a **linear-style chassis** integrated with a **hoarding function**. This hybrid design was chosen to give us the advantages of both offensive and control-focused playstyles—allowing efficient ball collection, retention, and deployment while maintaining high mobility and structural simplicity.

Throughout the design process, we prioritized **weight optimization** to improve acceleration and overall maneuverability. To achieve this, we maximized the use of **polycarbonate and other lightweight plastics**, which provided the necessary strength and flexibility while reducing excess aluminum mass. Additionally, we intentionally designed the robot to be **rear-weighted**, balancing the center of gravity toward the back. This configuration was planned to support future **front-mounted park mechanisms**, ensuring they can operate smoothly with less strain on the drivetrain.



The CAD model also emphasizes **low weight distribution**, which significantly enhances stability and minimizes the risk of tipping during rapid acceleration, cornering, or climbing. Every mechanism included in the design serves a functional purpose aligned with this season's game objectives, ensuring that the robot can **complete all scoring tasks efficiently** once assembly begins.

This finalized CAD model represents a critical milestone in our design process. It will serve as the foundation for manufacturing and testing in the upcoming stages, allowing us to verify real-world performance against our theoretical planning and make necessary refinements before competition.

- ▶ Focus: Starting out our frame for the drivetrain

Date: Aug 26, 2025

Members Involved

Daniel, Bryan

Objective

The objective is to create a stable and rigid frame for the very base of our robot. We plan to use straight materials in order to achieve this goal.

Materials

- 1, 35 Hole 1x3x1 C-Channel (Bracing)
- 2, 30 Hole 1x2x1 C-Channel
- 2, 28 Hole 1x2x1 C-Channel
- 1, 27 Hole 1x2x1 C Channel
- 1, 27 Hole 1x1 Angle Bar
- 2, 19 Hole 1x1 Angle Bar
- Spacers
- Screws
- Standoffs
- Nylocks
- Bearings

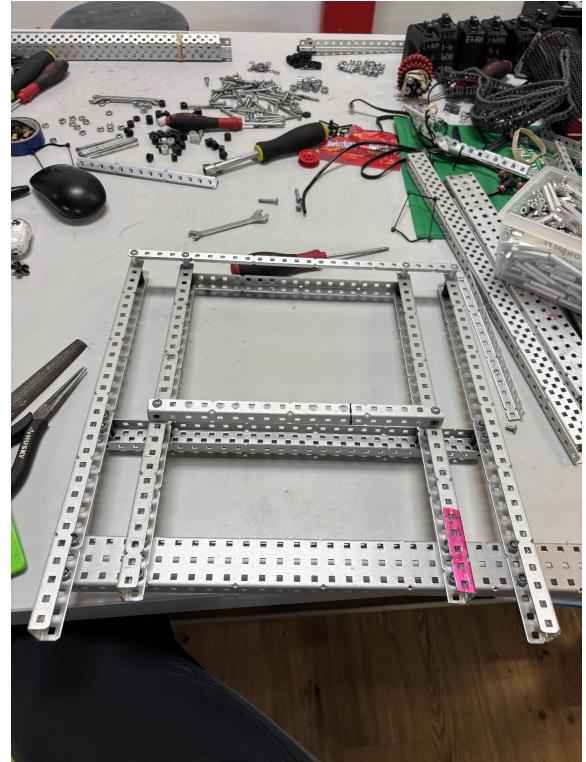


Figure 1: Picture of the bottom of our frame



Figure 2: Picture of the top of our frame

► Focus: Design towards our robot for local events

Date: Aug 26, 2025

1



X2 30 Hole 1x2x1 C-Channel



X2 28 Hole 1x2x1 C-Channel



X1 27 Hole 1x2x1 C-Channel



X16 ½" Spacer



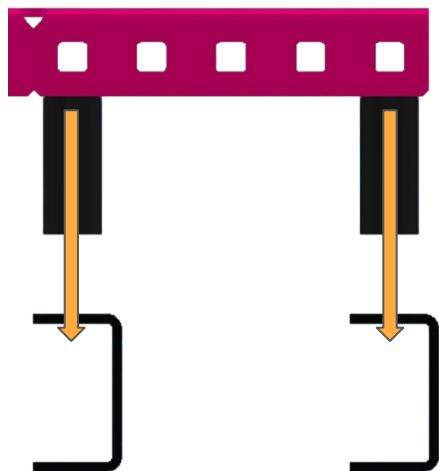
X16 1.25" Screws



X16 Nylocks



X16 ¾" Spacers



Arrows represent where screws and Nylocks go.
From here on lego cad will ONLY be important structure



Repeat on both sides

► Focus: Design towards our robot for local events

Date: Aug 26, 2025

2

 X1 27 Hole 1x1 Angle Bar

 X1 19 Hole 1x1 Angle Bar



X8 1/2" Spacer



X8 5/8" Spacers



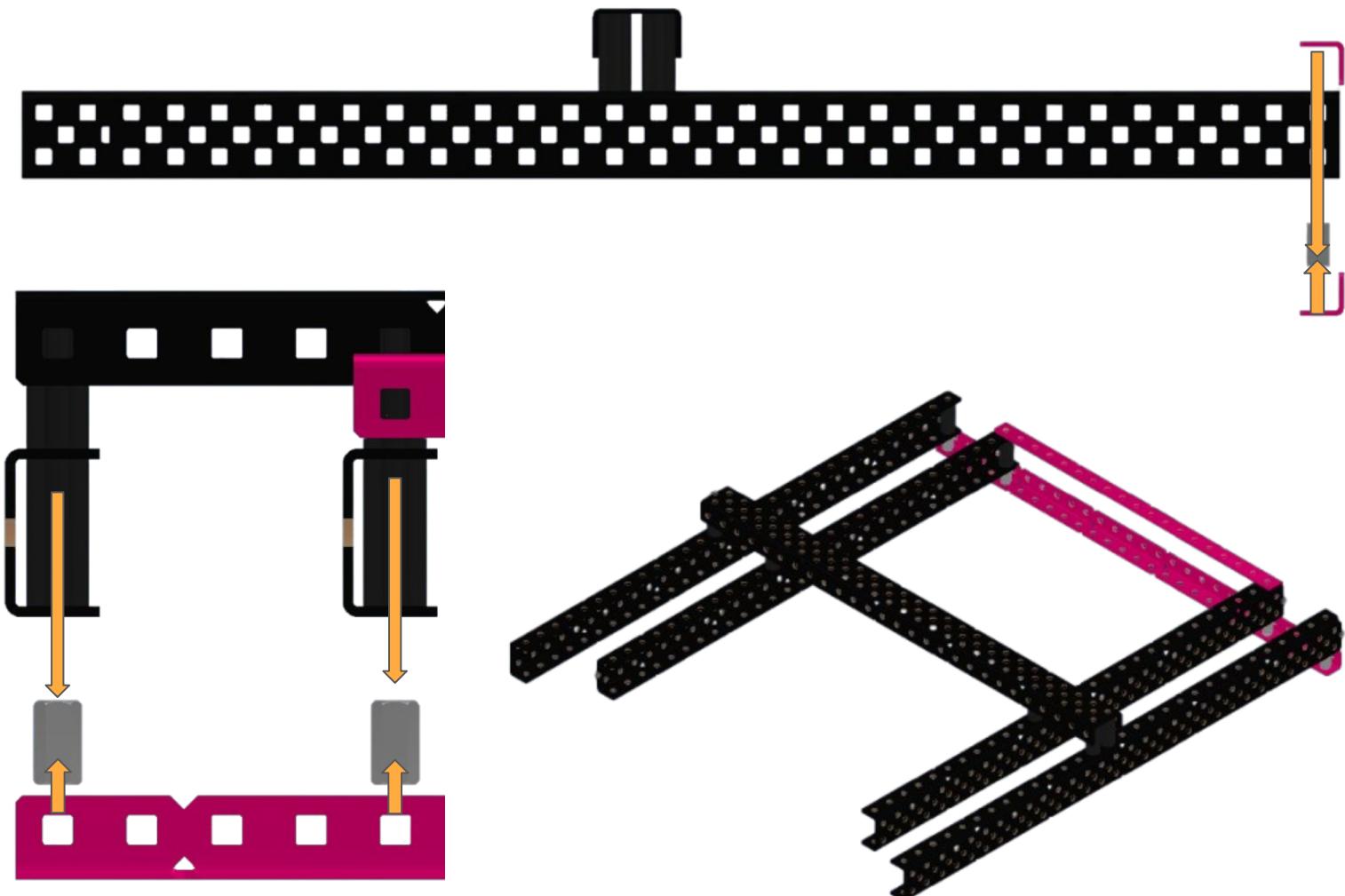
X8 1.25" Screws



X4 0.375 Screw



X4 0.5 Standoff



► Focus: Design towards our robot for local events

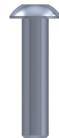
Date: Aug 26, 2025

3

 X1 19 Hole 1x1 Angle Bar



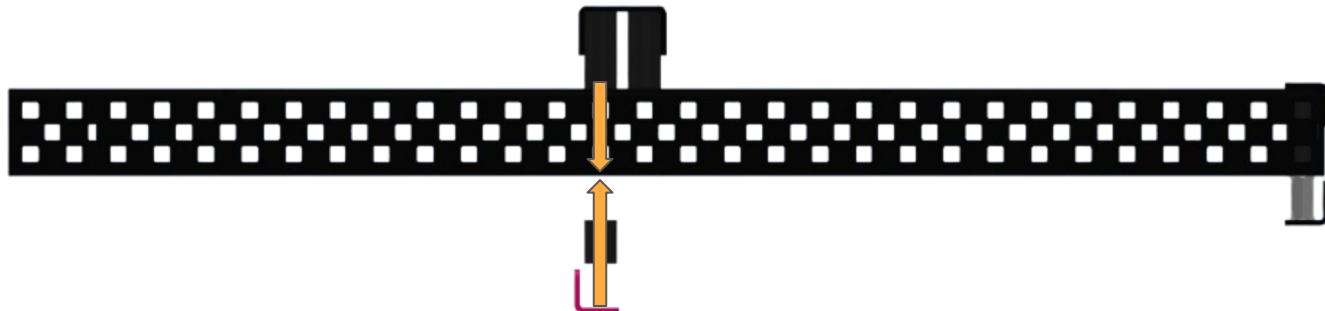
X2 ½" Spacers



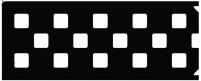
X2 0.75" Screws



X2 Nylocks



4

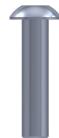
 X4 5 Hole 1x2x1 C-Channel



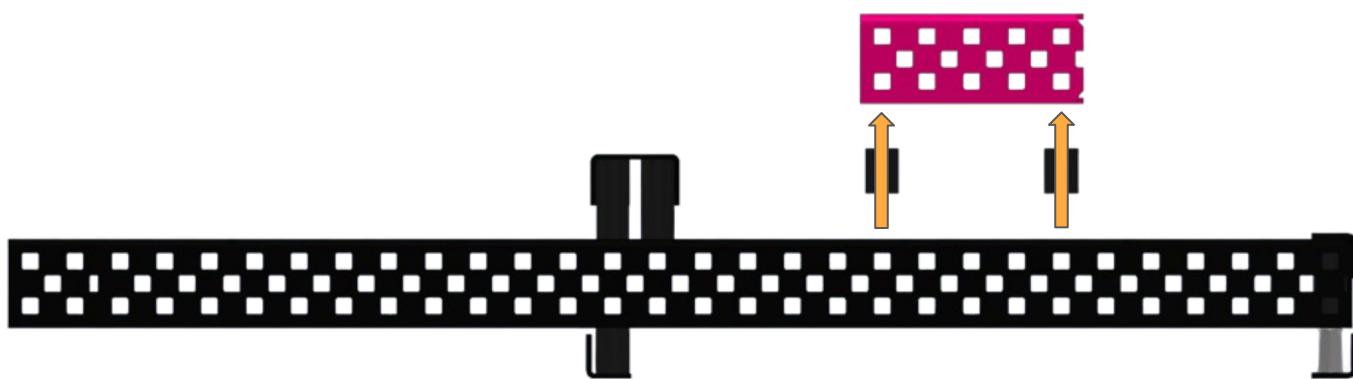
X8 Nylocks



X8 ½" Spacers



X8 0.75" Screws



Repeat on all 28 and 30 Hole C-Channels

► Focus: Design towards our robot for local events

Date: Aug 26, 2025

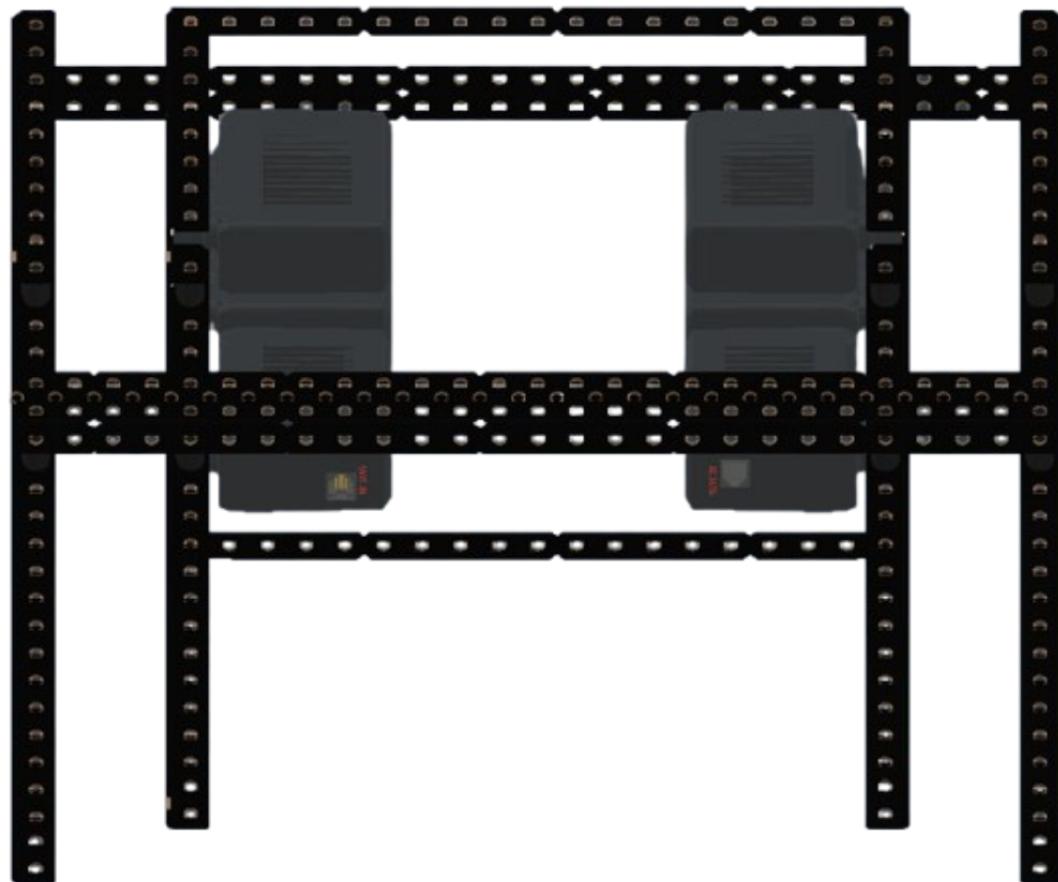
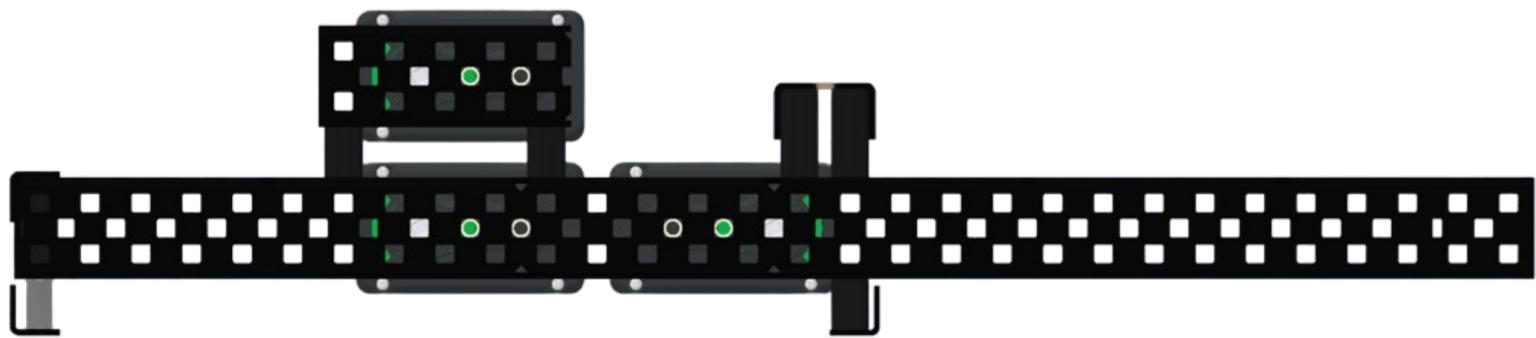
5



X6 11W Motor



X12 0.375 Screw



- ▶ Focus: Starting out our frame for the drivetrain

Date: Aug 26, 2025

Problems and Solutions

A problem we encountered was that we needed to keep everything centered.

To solve this problem, we utilized both temporary bracing and shoulder screws to ensure a perfectly straight drivetrain

Next Steps

We plan on continuing building structure for the rest of our robot.



Figure 1: Picture of our completed drivetrain frame

- ▶ Focus: Creating structure for our rollers

Date: Aug 27, 2025

Members Involved

Daniel, Bryan

Objective

We plan to build framing and structure for our rollers, ramp and basket. This would be done through using angle bars, C-channels and standoffs in order to create stable structure for our robot.

Materials

- 2, 30 Hole 1x2x1 C-Channel
- 2, 35 Hole 1x1 Angle Bars
- 2, 21 Hole 1x1 Angle Bars
- 1, 19 Hole 1x1 Angle Bars
- 2, 17 Hole 1x1 Angle Bars
- 2, 13 Hole 1x1 Angle Bars
- 2, 3 Hole 1x1 Angle Bars
- 4, 1 Hole 1x1 Angle Bars
- Screws
- Spacers
- Standoffs
- Nylocks
- Bearings
- Axle Collars

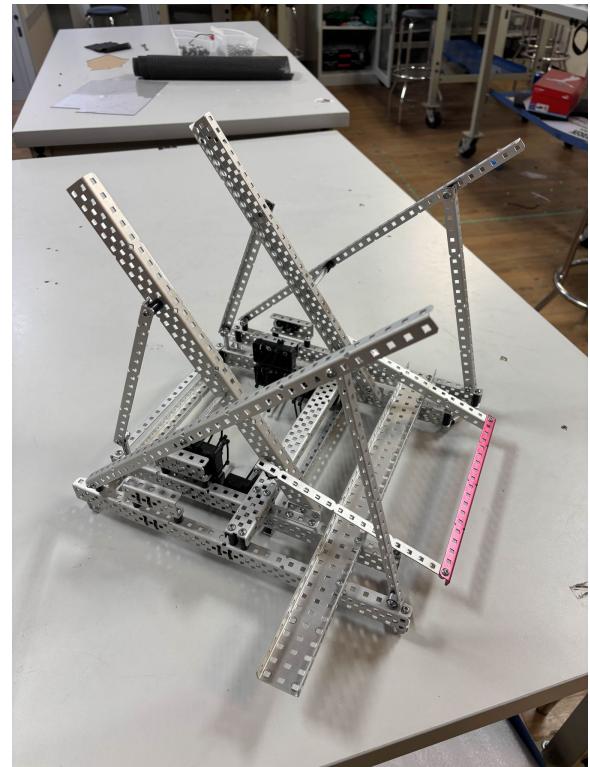


Figure 1: Picture of our intake structure from the side

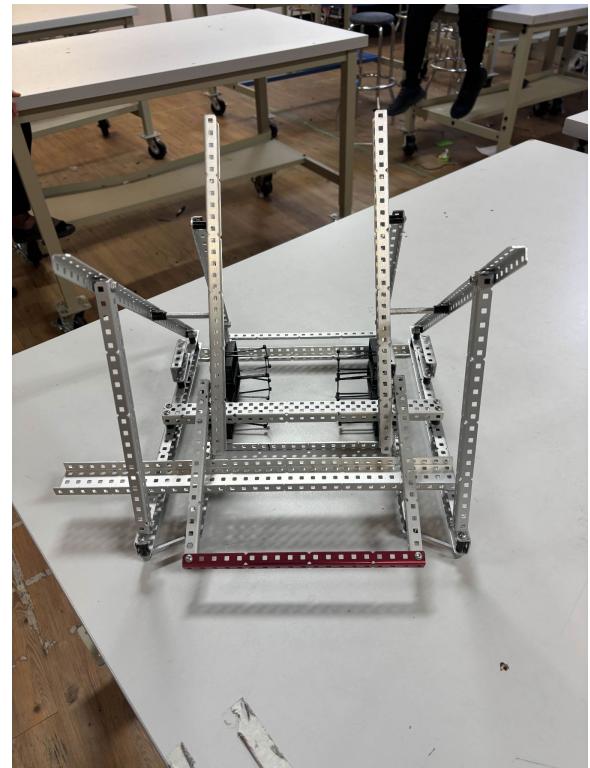


Figure 2: Picture of our structure from the front

► Focus: Design towards our robot for local events

Date: Aug 27, 2025

1

 X2 3 Hole 1x1 Angle Bar



X4 0.375 Screw

 X6 1 Hole 1x1 Angle Bar

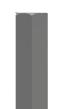


X2 0.625 Screw

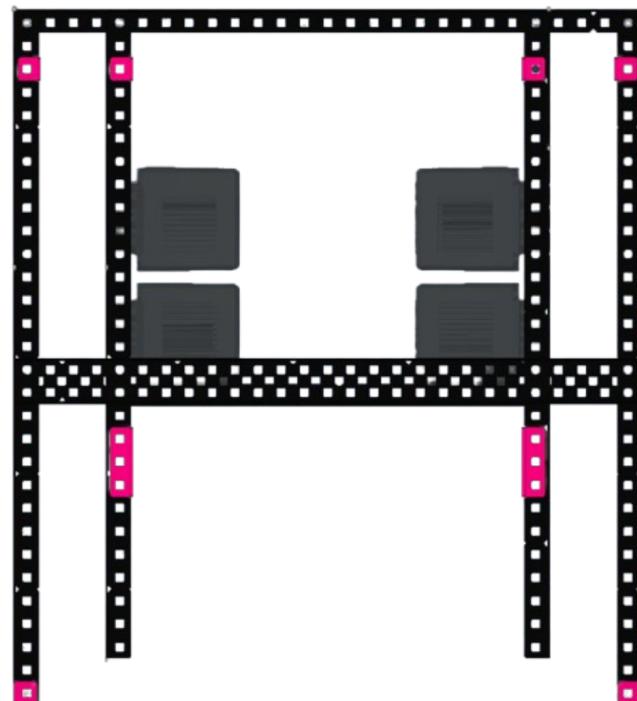
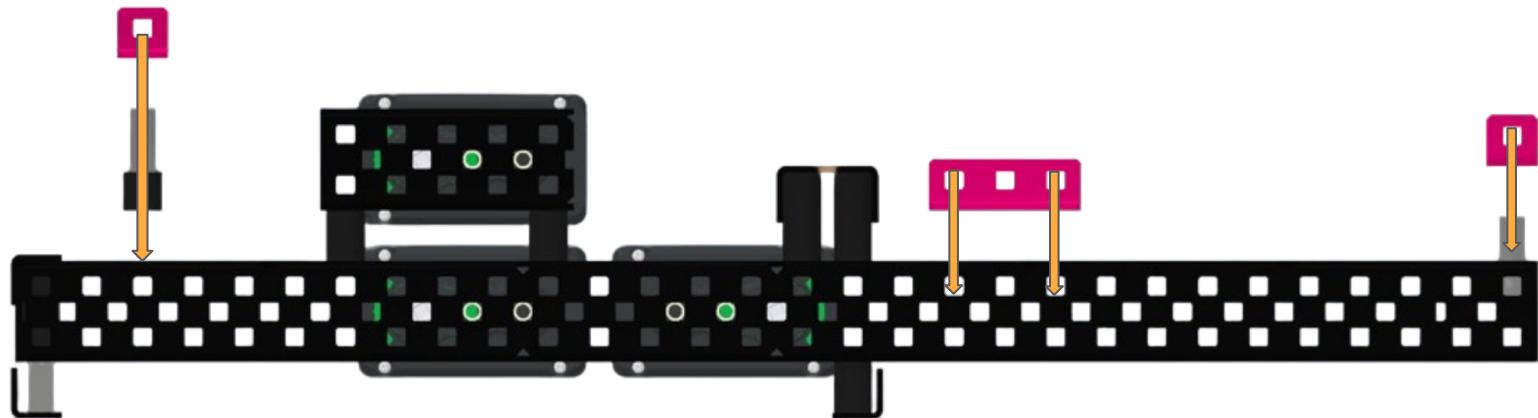
 X2 $\frac{3}{8}$ " Spacers



X2 0.85" Standoff

 X2 1" Standoff

 X6 Nylocks



► Focus: Design towards our robot for local events

Date: Aug 27, 2025

2

 X2 30 Hole 1x2x1 C-Channel

 X2 35 Hole 1x1 Angle Bar

 X2 21 Hole 1x1 Angle Bar

 X2 17 Hole 1x1 Angle Bar



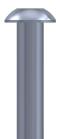
X10 ½" Spacer



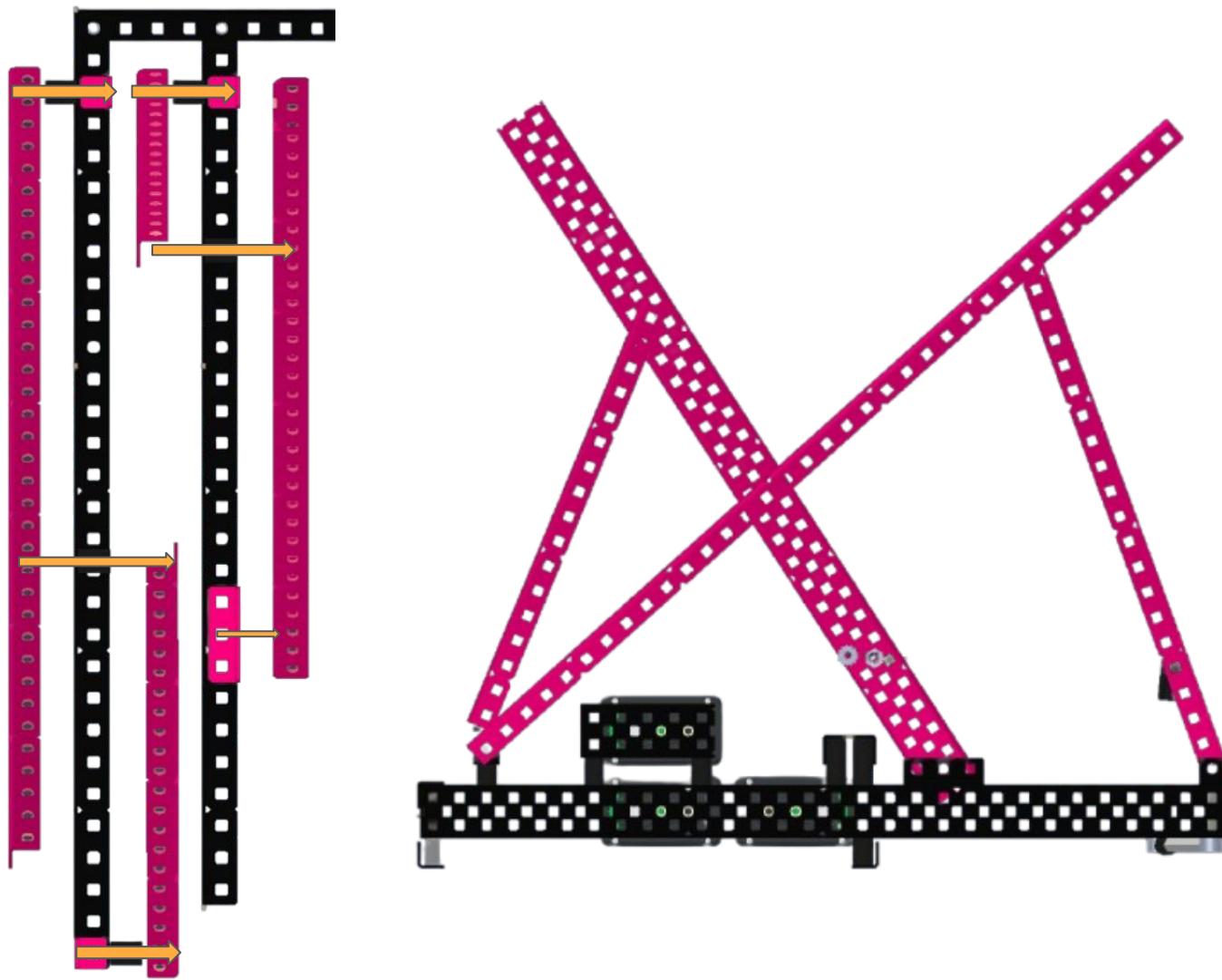
X12 Nylocks



X2 0.375 Screw



X10 0.75" Screws



► Focus: Design towards our robot for local events

Date: Aug 27, 2025

3



X2 1.5" Standoffs



X2 5/8" Spacers



X4 Axle Collar



X4 1" Couplers

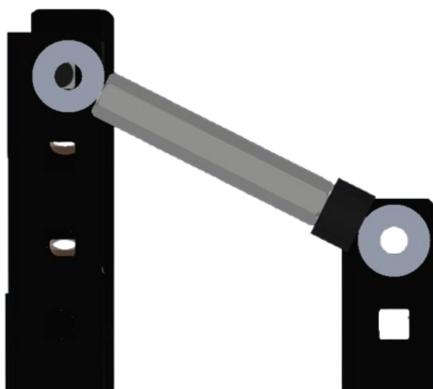
x2



4



X2 0.625 Screw



► Focus: Design towards our robot for local events

Date: Aug 27, 2025

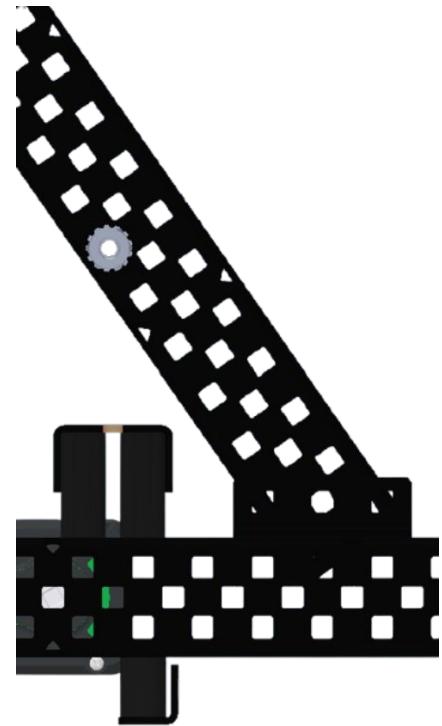
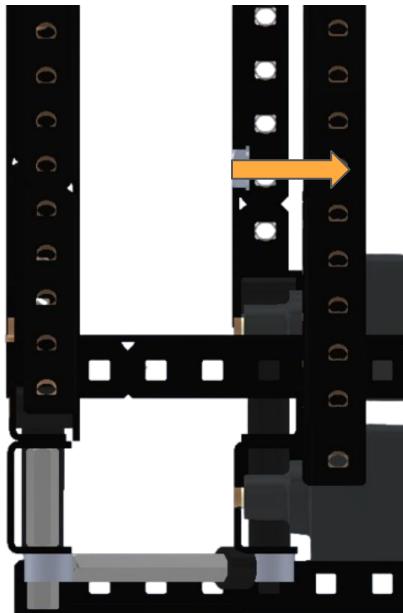
5



X2 0.75" Screws



X2 Kepsnuts



6



X2 13 Hole 1x1 Angle Bar



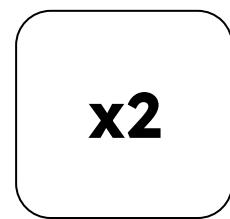
X2 Bearing



X2 Nylocks



X2 0.375 Screw



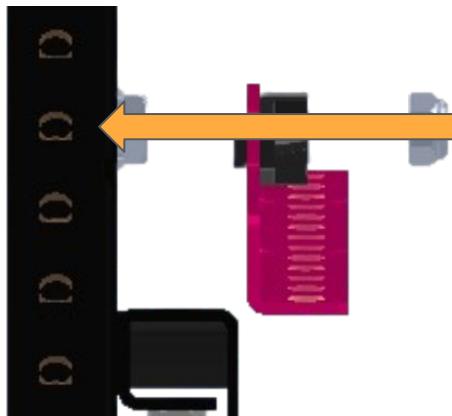
► Focus: Design towards our robot for local events

Date: Aug 27, 2025

7



X2 Nylocks



8



X1 19 Hole 1x1 Angle Bar



X2 1.25" Screws



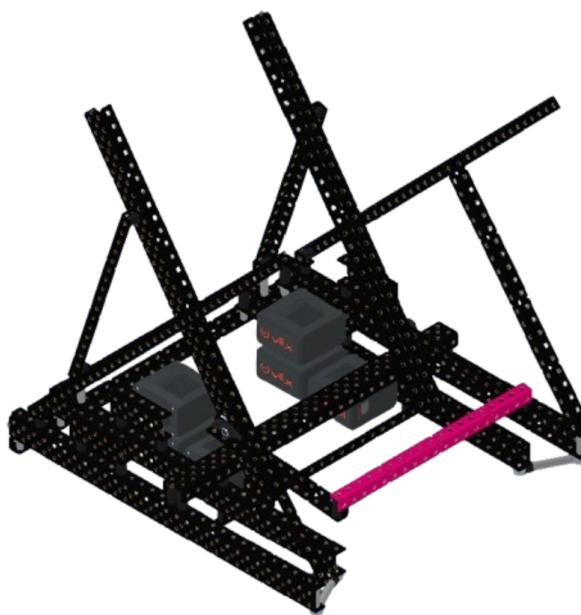
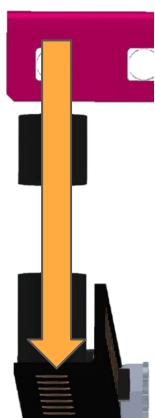
X2 3/8" Spacers



X2 Nylocks



X2 1/2" Spacer



► Focus: Design towards our robot for local events

Date: Aug 27, 2025

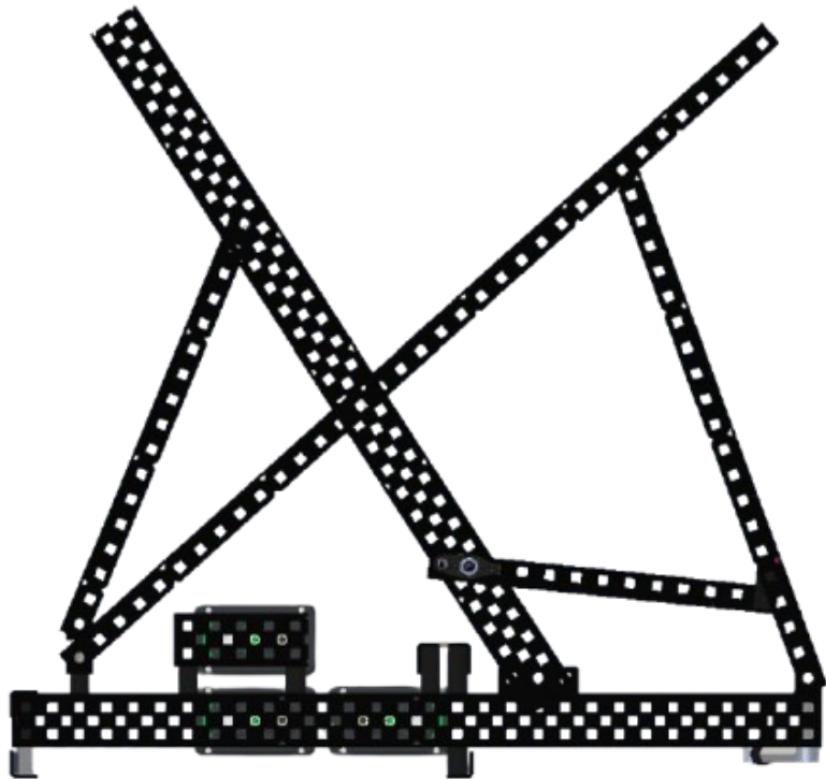
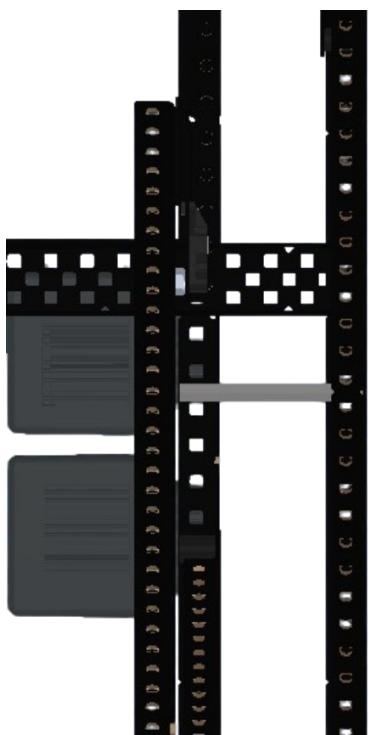
9



X2 2.5" Standoff



X2 0.625 Screw



- Focus: Creating structure for our rollers

Date: Aug 27, 2025

Problems and Solutions

Some of our structure was bending or warped before we could secure it.

To solve this issue, we had multiple people position the materials properly before securing with bracing.

Next Steps

We plan on adding more bracing and we plan to start building our rollers.

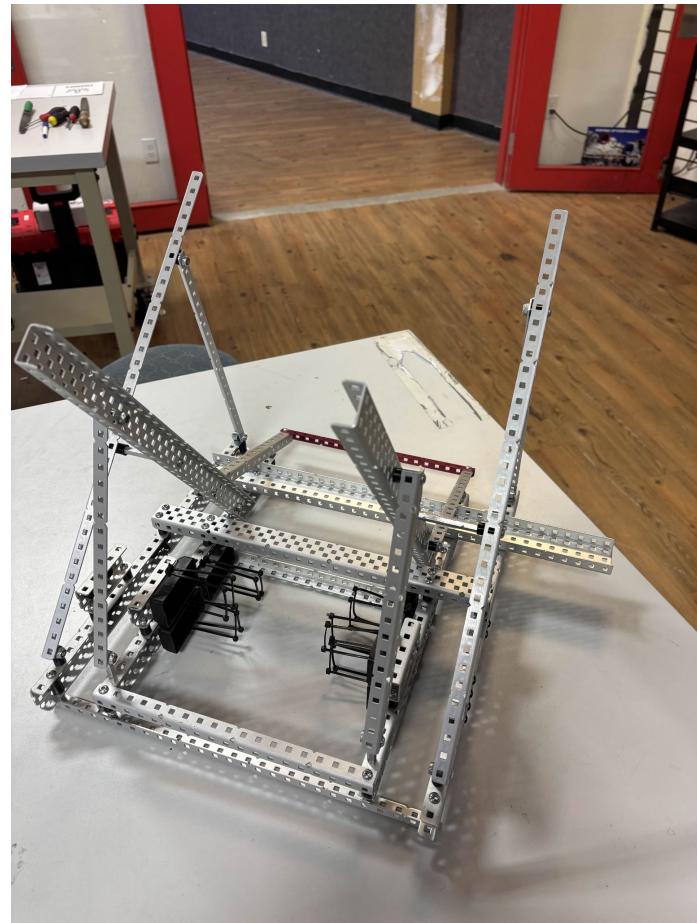


Figure 1: Picture of structural frame from the back

- Focus: Adding bracing to complete the drivetrain structure

Date: Aug 31, 2025

Members Involved

Daniel, Bryan

Objective

Add bracing to our drivetrain in order to make our drivetrain more secure. This would provide extra support for our drivetrain in order to make sure our parts do not bend.

Materials

- Standoffs
- Axle Collar
- Screws
- Nylocks
- Spacers

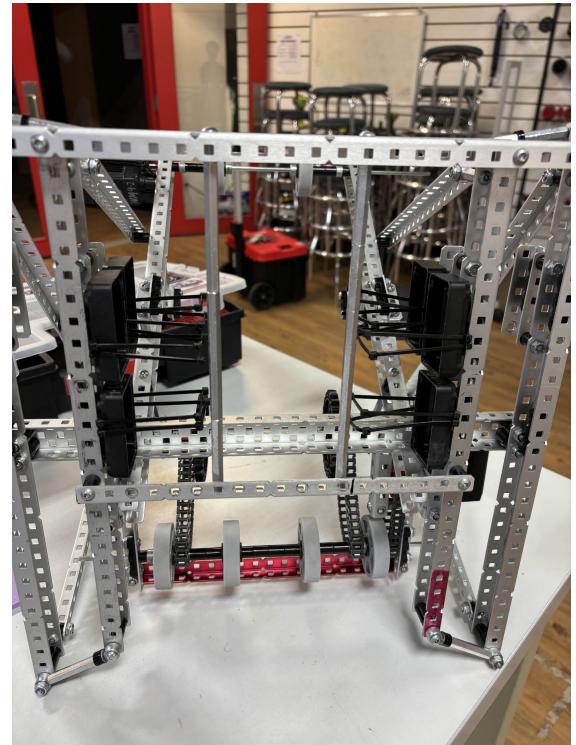


Figure 1: Bottom view of our standoff bracing

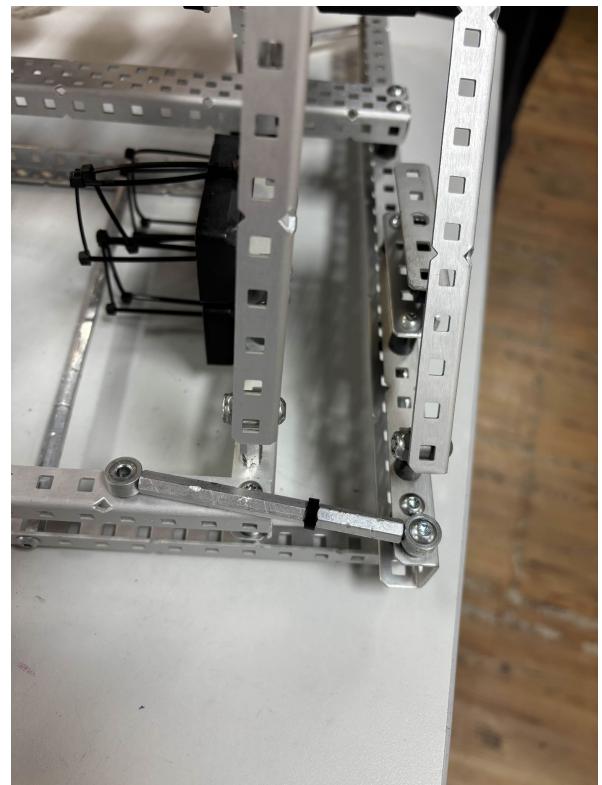


Figure 2: Picture of our back standoff brace

- Focus: Creating the very first set of rollers

Date: Aug 31, 2025

Members Involved

Daniel, Bryan

Objective

The focus of today was to build our very first stage of rollers using flexwheels. This stage uses a 5.5W motor to power two stages of rollers. The first set of rollers would be made out of flexwheels and the second set will be sprockets with rubber bands.

Materials

- 1, 5 Hole 1x1 Angle Bar
- 2, 3 Hole 1x1 Angle Bar
- 2" Flexwheels
- 2, 24 Teeth 9P Sprocket
- 3, 12 Teeth 6P Sprocket
- 3, 6 Teeth 6P Sprocket
- Spacers
- Screws
- Standoffs
- Bearings

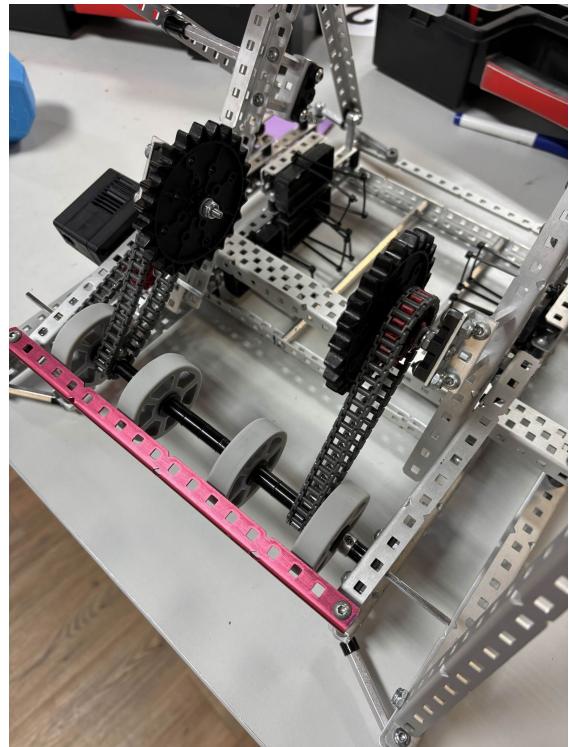


Figure 1: Picture of first stage intake and roller

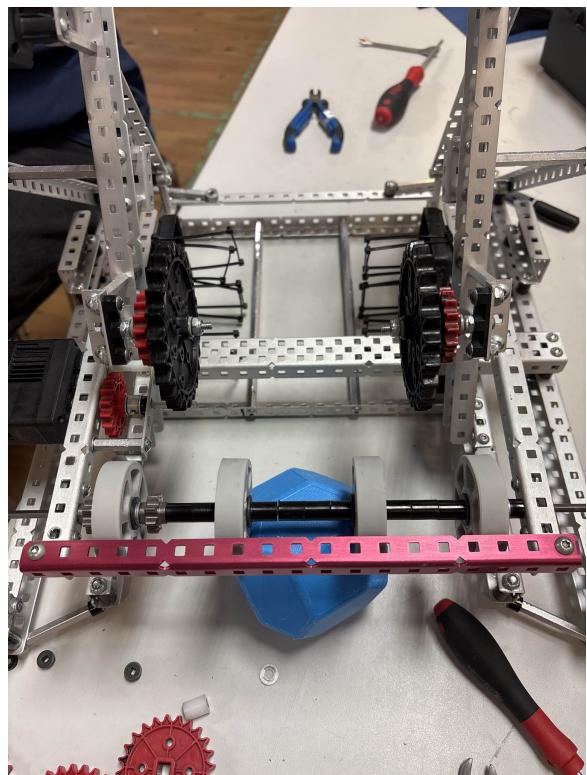


Figure 2: Picture of our first stage intake with a block

► Focus: Design towards our robot for local events

Date: Aug 31, 2025

1



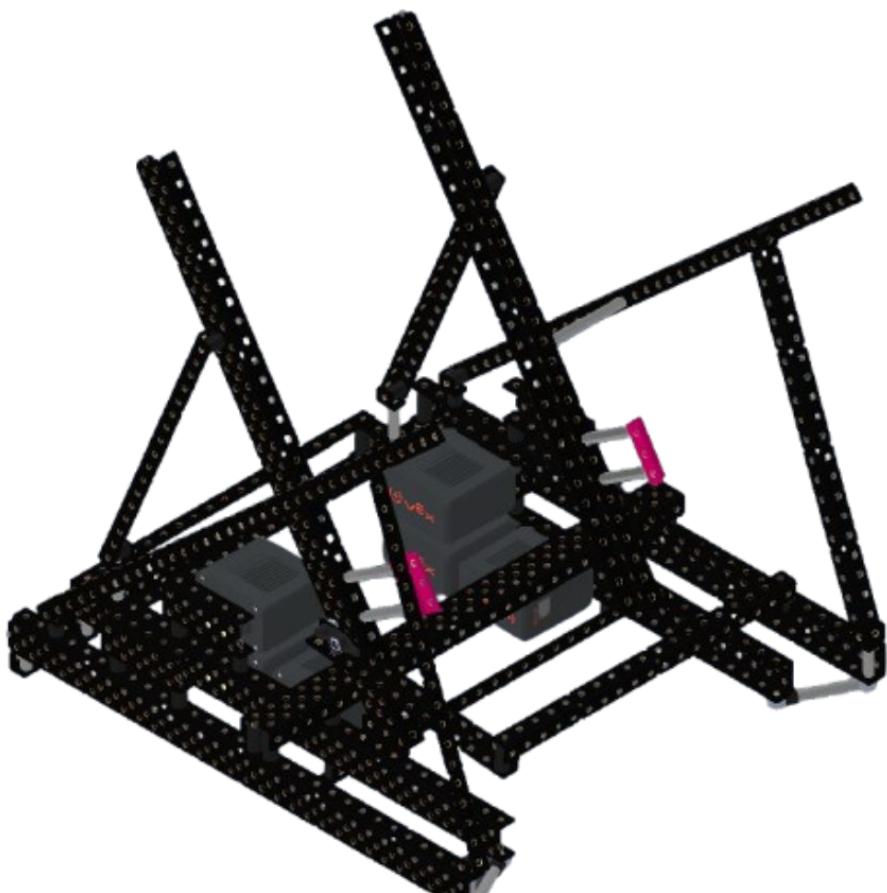
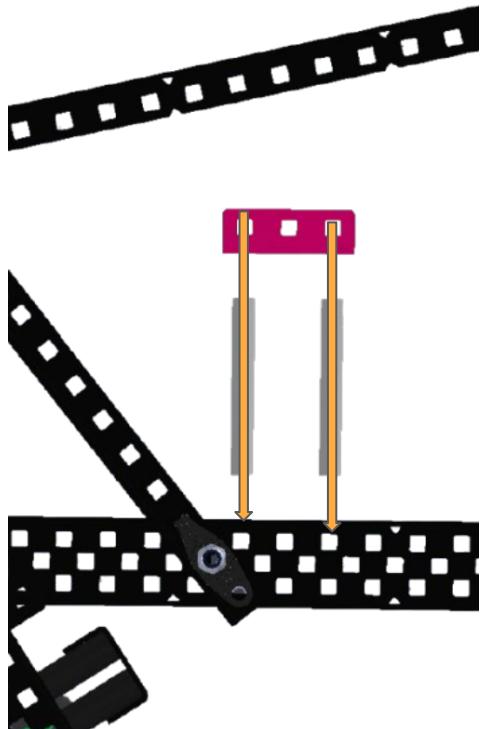
X2 2" Standoff



X4 0.375 Screw

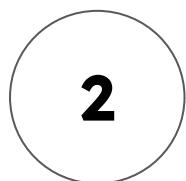


X2 3 Hole 1x1 Angle Bar



► Focus: Design towards our robot for local events

Date: Aug 31, 2025



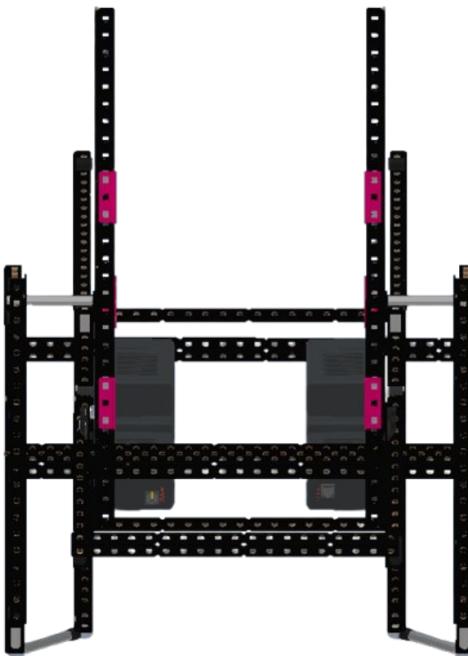
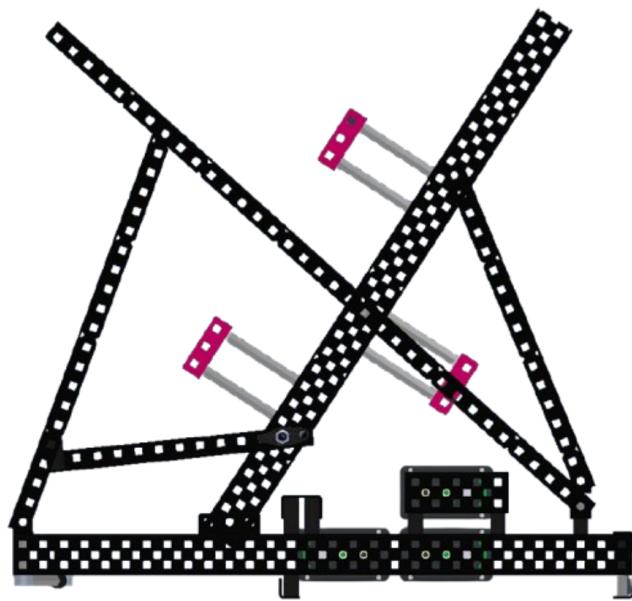
X2 2" Standoff



X8 0.375 Screw



X4 3 Hole 1x1 Angle Bar



- ▶ Focus: Creating the very first set of rollers

Date: Aug 31, 2025

Problems and Solutions

We noticed that the intake is too low to properly intake the blocks.

To solve this issue, we plan to use our double park mechanism as our hardstop

We noticed that since our rubber band rollers were on screw joints, the rubber bands would pull the sprockets towards each other. This would cause a lot of friction.

To solve this issue, we switch to a light strength axle in order to provide structural support.

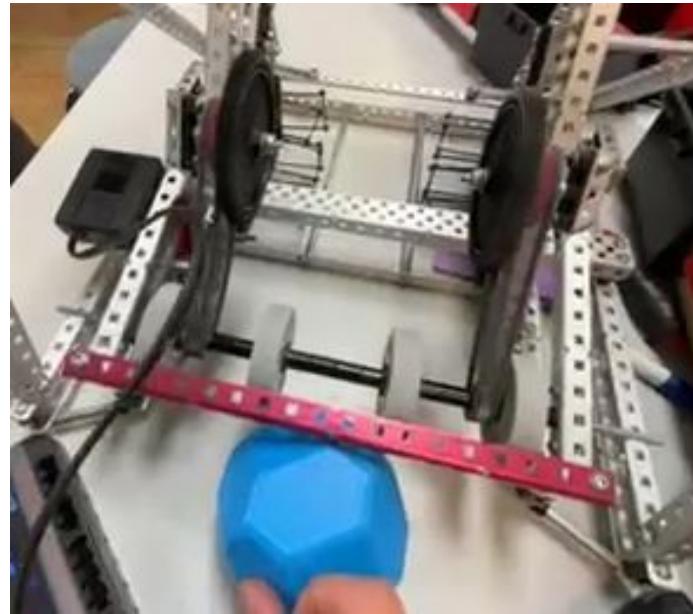


Figure 1: The block is unable to go into the intake because it is too low

Next Steps

We plan to set up our roller structure and motors.

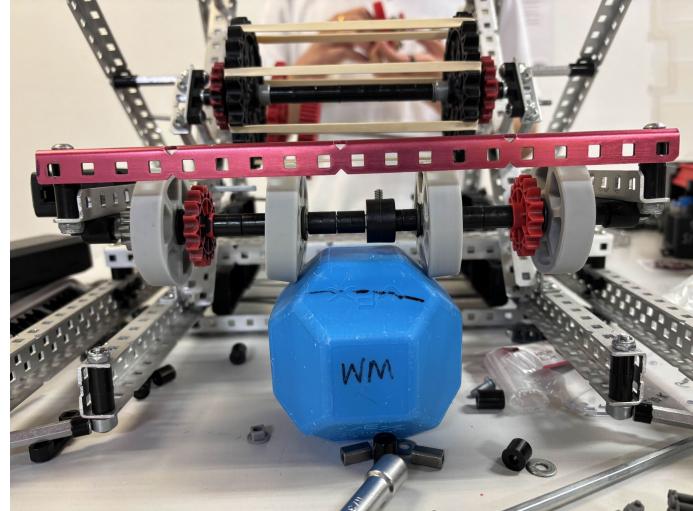


Figure 2: Intake resting on top of a block

- ▶ Focus: Setting up motors and structure for our rollers

Date: Aug 31, 2025

Members Involved

Daniel, Bryan

Objective

The focus is to create setup for our roller stages. This includes mounts for our rollers and our motors that will power our system. This would set up the rest of the intake system.

Materials

- Standoffs
- Screws
- Nylocks
- Angle bars
- 5.5W motors

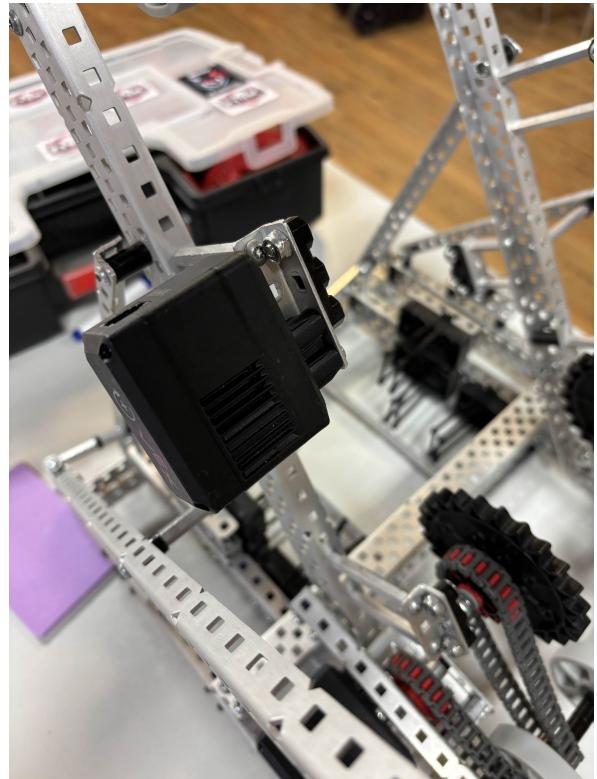


Figure 1: Picture of a 5.5W motor mounted for our roller stages

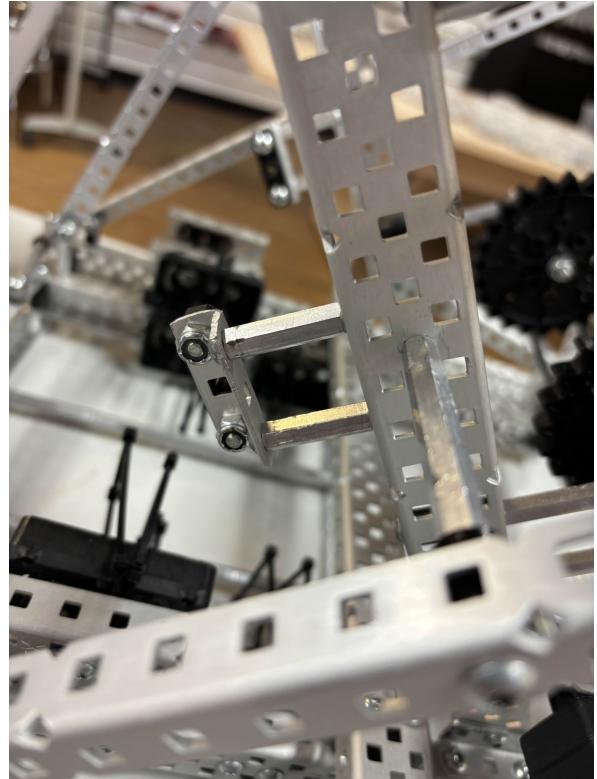


Figure 2: Standoff structure for our rollers

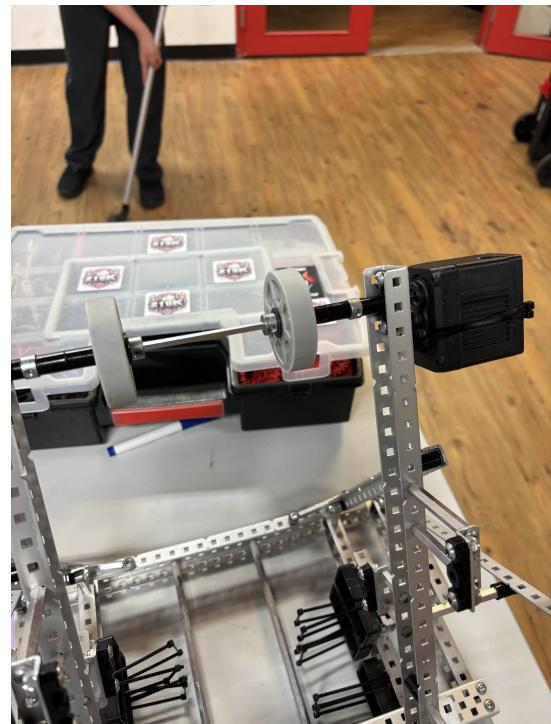
- ▶ Focus: Setting up motors and structure for our rollers

Date: Aug 31, 2025

Problems and Solutions

We noticed that some of our mounts did not line up perfectly..

In order to straighten everything, we adjusted how we mounted the standoffs by adjusting our screw tightness. We also used temporary bracing to ensure everything was straight.



Next Steps

We plan on building our rollers to create our intake system.

Figure 1: 11W directly power the top stage of flexwheels

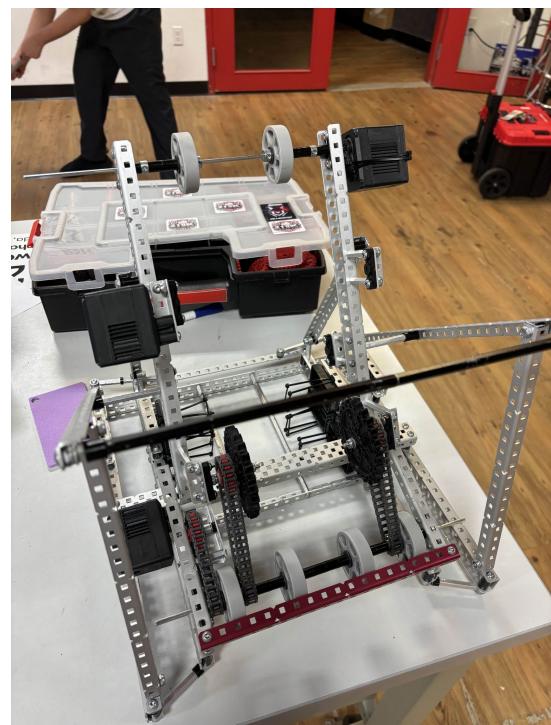


Figure 2: Picture of our robot in its current stage

- Focus: Finishing our rollers

Date: Sept 3, 2025

Members Involved

Daniel, Bryan

Objective

We started to build and attach our rollers onto our robot. We did this by using sprockets attached on axles to create a base. We then used rubber bands to create the rollers.

Materials

- Sprockets
- LS axles
- 11W motors
- Chain
- Rubber bands
- Flexwheels

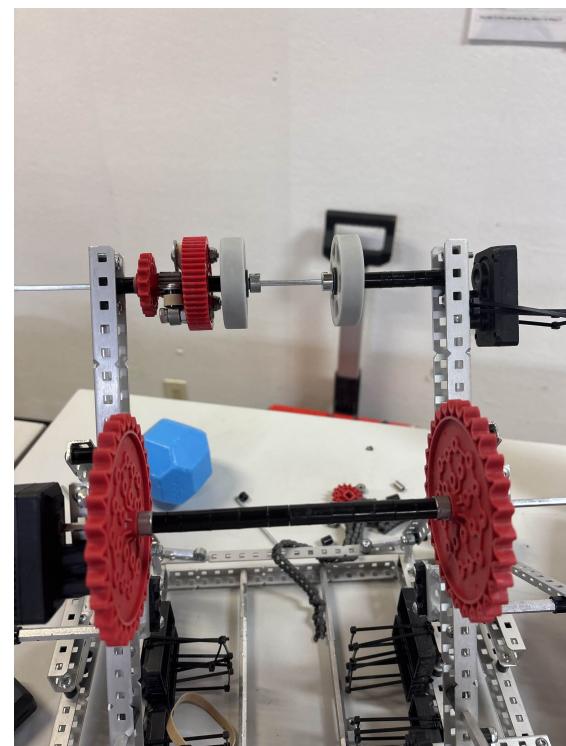


Figure 1: Sprocket roller without the rubber bands

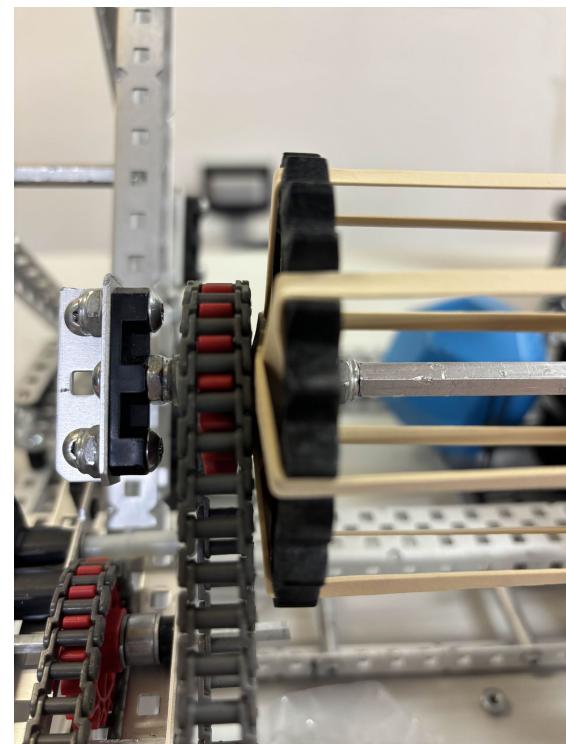


Figure 2: 6P chain powering our rollers

- Focus: Finishing our rollers

Date: Sept 3, 2025

Next Steps

We plan on creating polycarbonate pieces for various functions including alignment, barrier cross and ramps.

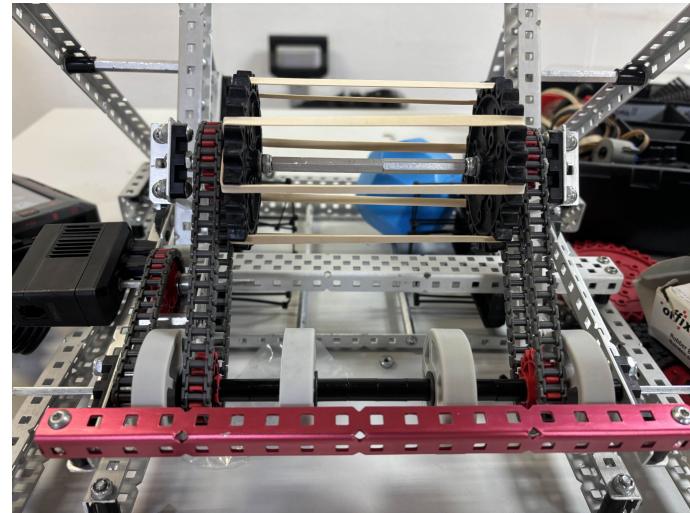
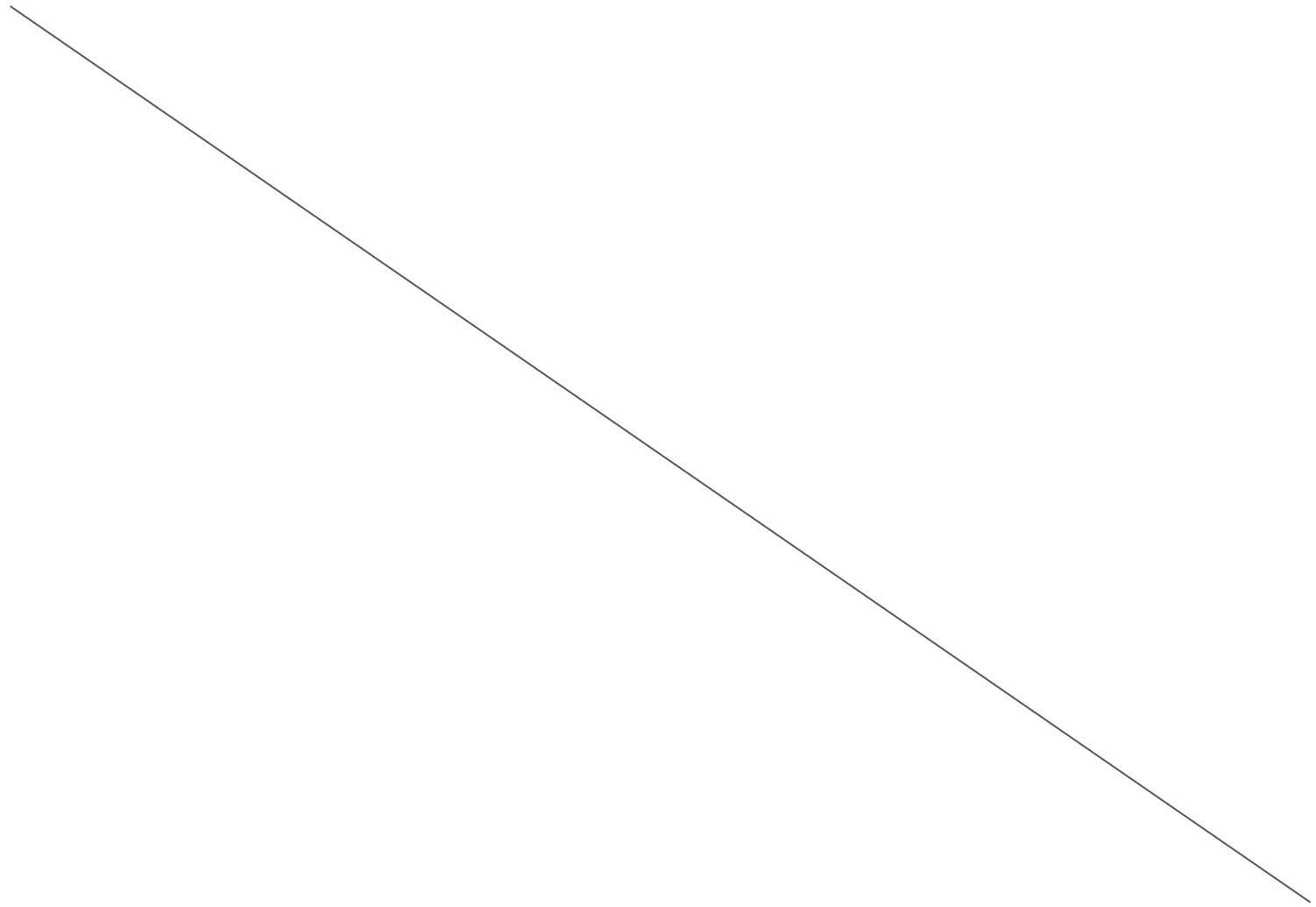


Figure 1: Completed first stage intake and first roller



► Problem: Game Manual V2.0

Date: Sept 4, 2025

This manual came out with a few major changes to how teams can play the game

Pinning Count

One of the biggest changes in this manual update was the pinning count changes. It was changed from where a team could legally pin for 5 seconds to 3 seconds. This will impact how many teams play defense as there is less time where you can pin a robot to prevent it from moving. Pinning is vital for both offensive and defensive play, this changes our approach to how we can utilize pinning to create opportunities within a match.

Park Zone Protection

This game manual update also introduced protection to the park zone. It states that opposing robots may not contact the other teams park barrier in the last 20 seconds of the match. This means blocking teams from parking will inherently have more risks as any contact with the park zone will result in a violation.

- ▶ **Problem:** Powering a one-directional roller

Date: Sept 4, 2025

One of our rollers only needs to move in one direction. Near the middle of the intake, we have a roller that spins under the blocks to move them upwards.

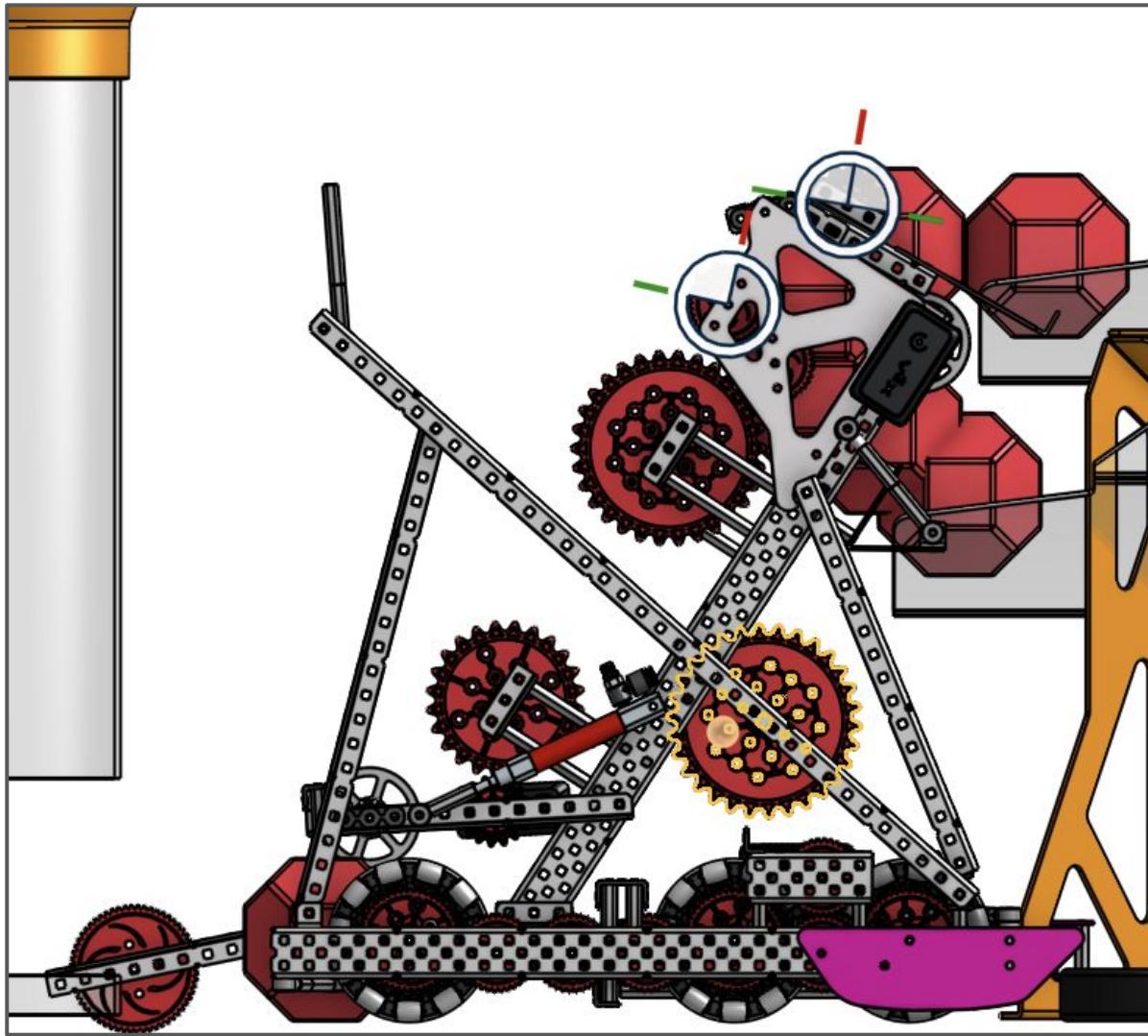


Figure 1: A CAD of our robot, with the roller referenced by this page highlighted.

Looking at Fig. 1, the highlighted roller only needs to spin clockwise. The problem is deciding how to power this roller. It is chained to the flex-wheel roller above, which needs to spin in both directions.

► Focus: Powering a one-directional roller

Date: Sept 4, 2025

A pair of ratchets can be used to make it so that an axle spinning in **either direction** will turn another axle in only **one direction**.

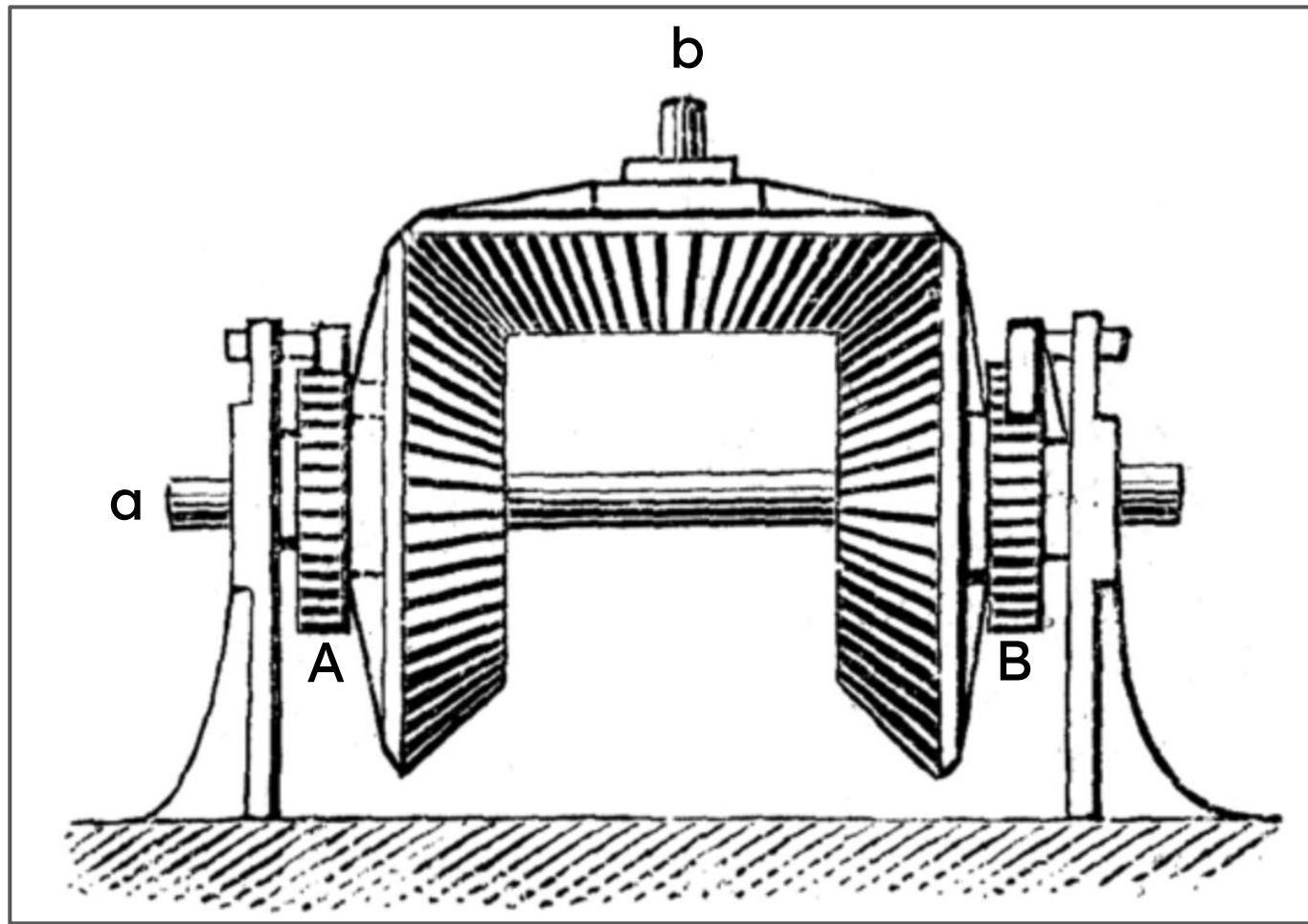


Figure 1: A diagram of the proposed mechanism from 507 Mechanical Movements, labelled by us.

Axle a is the driver, and b the driven. When axle a is spun clockwise, ratchet A starts to skip, meaning only ratchet A is powering axle b. This makes axle b spin clockwise. The opposite happens when axle a is spun counter-clockwise, but axle b still spins clockwise. As a result, axle a can be spun either direction, and the direction of axle b will be unaffected.

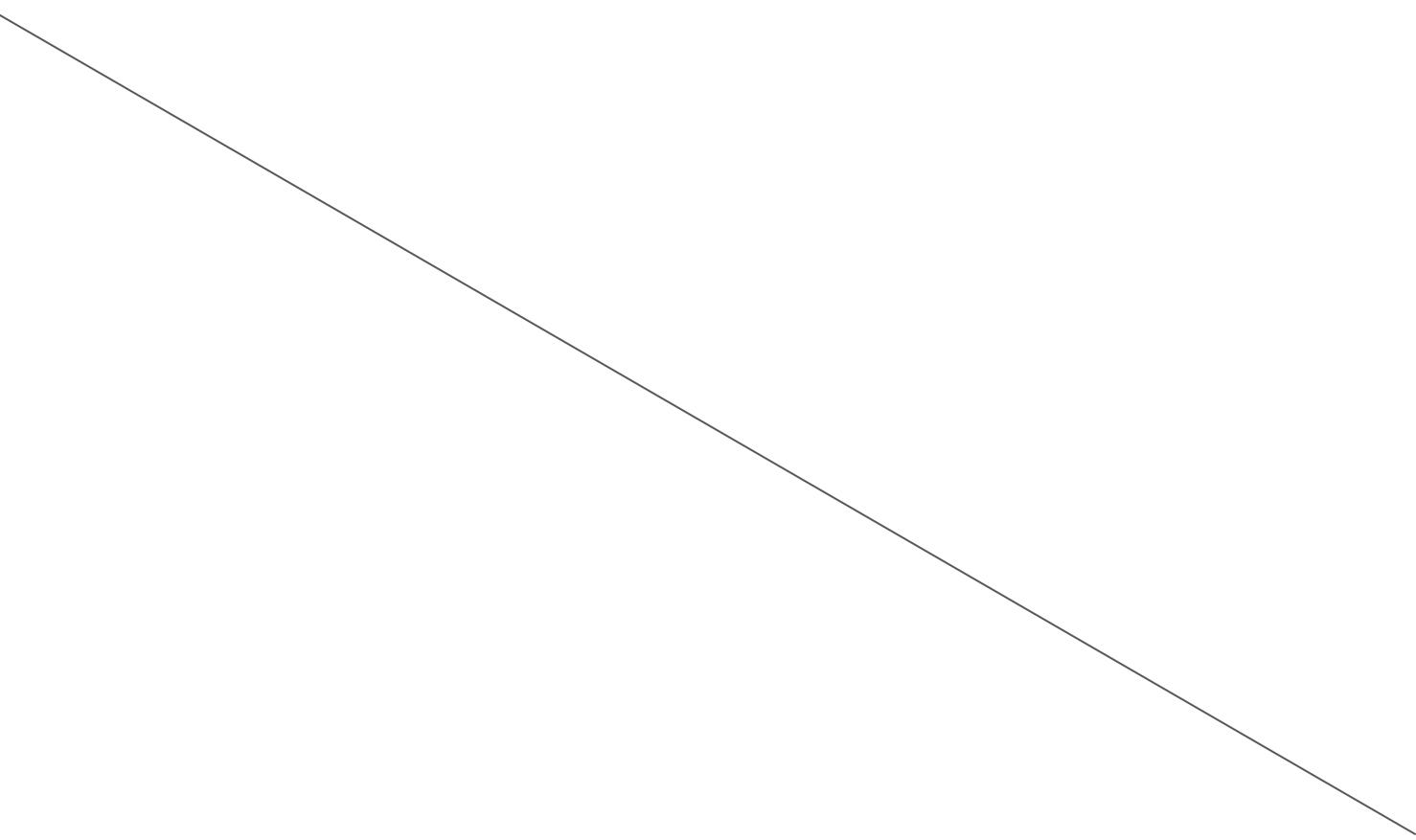
- ▶ Focus: Powering a one-directional roller

Date: Sept 4, 2025

There are two proposed ways to power this roller:

1. Splitting the 11W motor into two 5.5W motors, such that the roller can be controlled independently.
2. The mechanism proposed on the previous page, pg. 214. This will be built differently on our robot. Let us consider two scenarios:
 - a. The driver roller is spun clockwise. One ratchet will be active (and the other will skip, because they are opposite), and it will directly power the driven roller, making it spin clockwise.
 - b. The driver roller is spun counter-clockwise. One ratchet will be active (and the other will skip), which will spin counter-clockwise. The direction will be reversed before spinning the driven roller, making it spin clockwise.
 - c. In both scenarios, the driven roller is spinning clockwise.

Note: there will be **no planning page** for this mechanism, because it is already detailed above.



- Focus: Powering a one-directional roller

Date: Sept 4, 2025

When choosing from the two proposed designs, the most important thing to consider is the specs of the motors. The main difference is the motor split. Here is the issue: 5.5W motors have a higher weight:wattage ratio, they burn out faster, and have more internal friction. Therefore, we should always use 11W motors when possible. We should mainly be considering whether or not the friction from the ratchets will have higher friction, weight, and burn-out time than the two 5.5W motors. Here is a corresponding decision matrix:

<u>Criteria</u> <u>Options</u>	Friction /5	Weight /5	Burn-out Time /5	Total /15
Split 5.5W Motors	3	4	2	9
Ratchet System	4	3	4	11

Figure 1: Decision matrix showing factors influencing our decision between splitting into 5.5W motors, and a ratchet system.

Based on the decision matrix, we will build the ratchet system.

- ▶ Focus: Building Ratchet

Date: Sept 5, 2025

Members Involved

Bryan, Brandon

Objective

Powering the bottom roller from the top roller. The bottom should only spin one direction, outwards. One ratchet was placed on the driver axle, one near the bottom, powering a gear on the driven axle (refer to Fig. 1). We made each ratchet by having a pinion attached to a sprocket so that they spin together, then attaching rubber-banded screws which dig into the pinion (refer to Fig. 2)

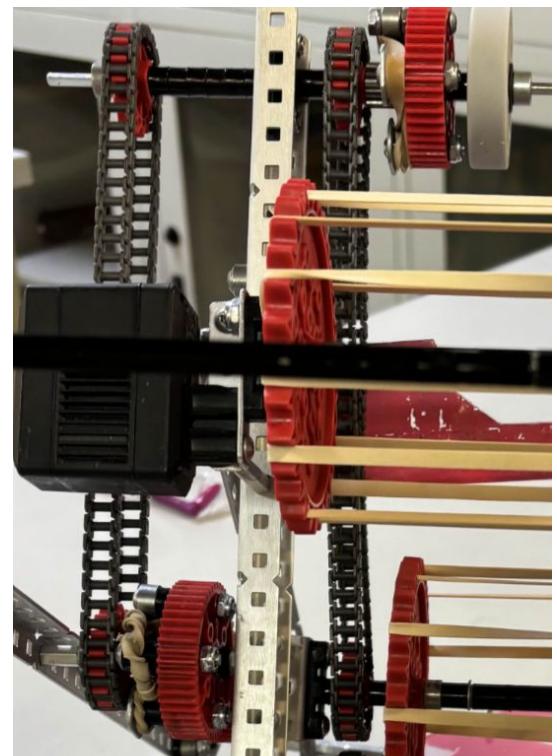


Figure 1: Both ratchets placed on the robot

Materials

- Sprockets
- Axles
- Gears (pinions and a larger size)
- Washers
- Chain
- Rubber Bands
- Spacers
- Screws
- Nylocks

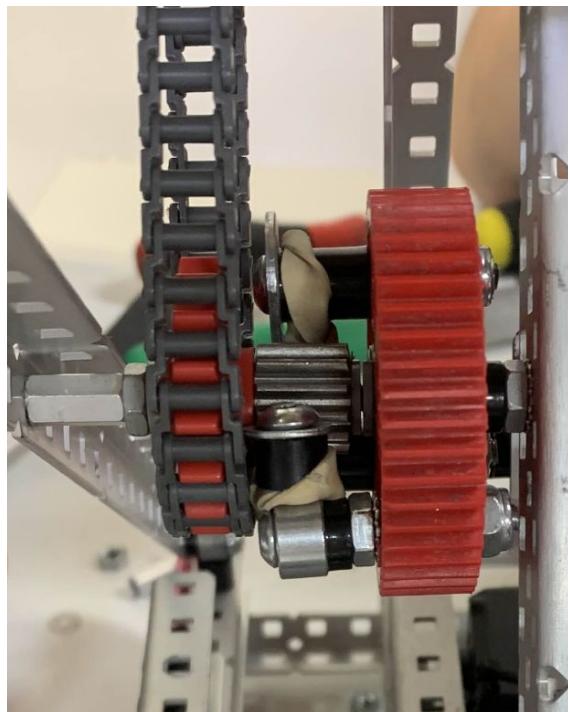


Figure 2: Picture of how we chain our ratchet

- Focus: Building Ratchet

Date: Sept 5, 2025

Problems and Solutions

Many spots on the ratchets were cantilevered.

We supported the sprocket near the bottom on both sides and placed the gear on the inside to avoid pressure bending the axle.

We had difficulties balancing center of gravity issues, and space.

One ratchet was placed at the top, and chained near the roller to avoid making the roller too short. The other ratchet was redirected with a gear, placed near the bottom to preserve a low center of gravity.

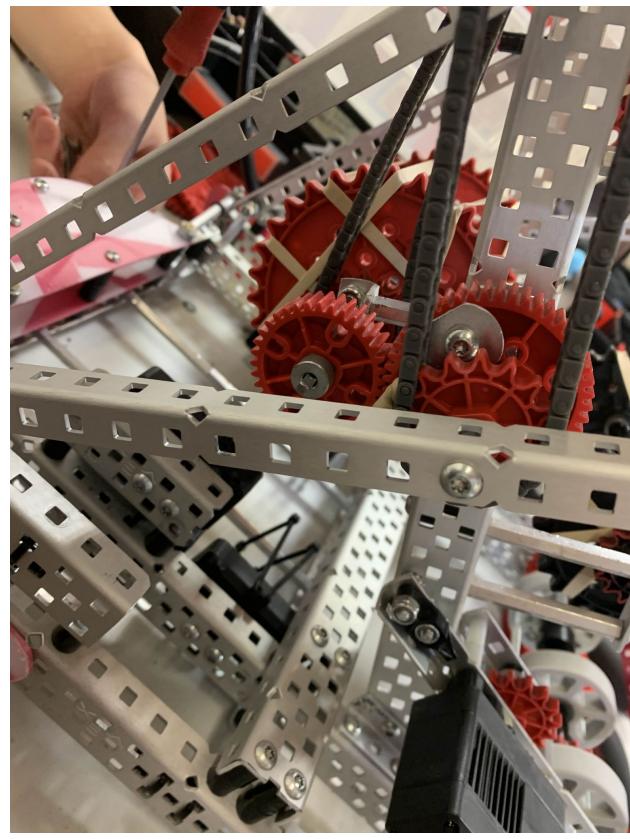


Figure 1: Side view of how we fit the ratchet within our robot

Next Steps

Continue building the intake and test it once completely built.

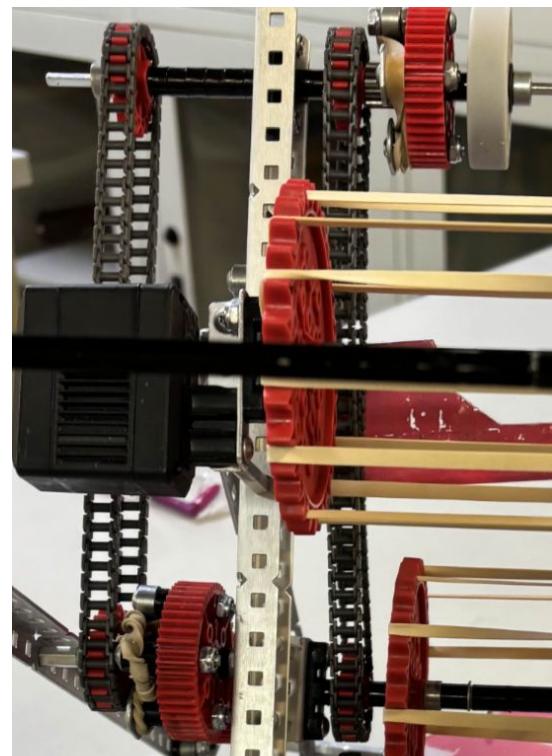


Figure 2: Same as Fig. 1, Pg. 213

- ▶ **Problem:** How do we align ourselves with long and middle goals? How do we go over the park barrier?

Date: Sept 8, 2025

Observations from MOA

After our previous experience at MOA, we noticed that the most successful teams had either a metal or polycarbonate alignment piece that would help them line up with long and middle goals. This would help in both autonomous as the aligner would help the robot line up. Meaning our autonomous could be more lenient and still work. The aligner would also help us line up quicker in drive control, leading to quicker cycling times between the loaders and long goals as well as quicker scoring in general.

Park Barrier

Due to our drivetrain orientation and the wheels being on the top hole, we noticed that we would not be able to cross into the park zone without any help. In order to solve this issue, we looked at both Spin Up and Over Under as they had field elements that required teams to go over. Most teams in the previous seasons had polycarbonate sleds that would assist in helping the robot go over these field elements.

CADing polycarbonate

Plan

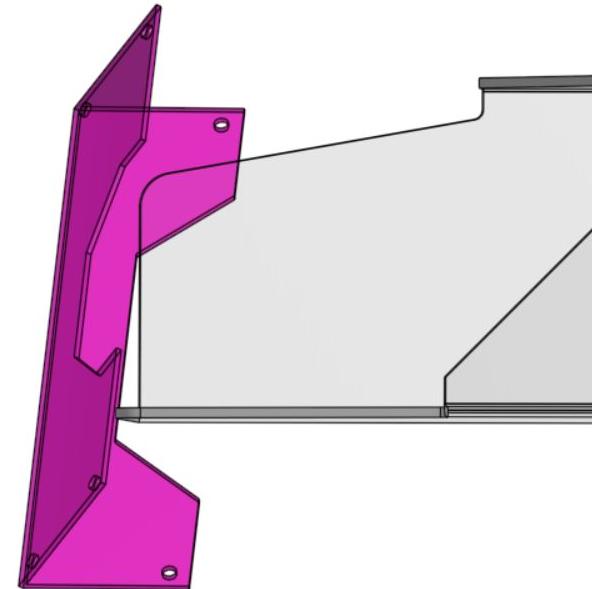
- Focus: Using CAD to create our polycarbonate pieces

Date: Sept 8, 2025

In order to create our polycarbonate aligner and sleds. We utilized CAD to plan out a perfect sized polycarbonate piece that would do this job. We used CAD to create three pieces that we would later CNC.

1. Long goal aligner

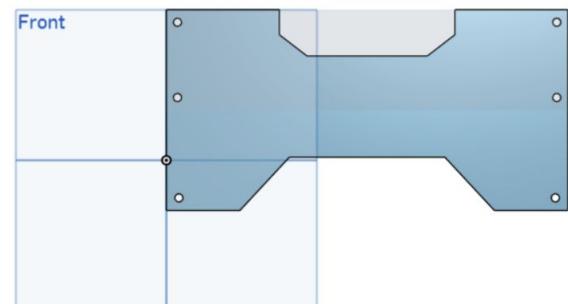
The long goal aligner would be a semicircle that could help us align with the long goal. We chose to use a semi circle as the round edge could automatically help us align.



2. Mid goal aligner

Our mid goal aligner would both act as a ramp and aligner. This piece would have a cutout that would align with the bottom of the mid goal. It also features a cutout that would allow blocks through in order to score on the mid goal.

Figure 1: Middle goal aligner in CAD to line up with the upper middle goal



3. Sleds

Our sleds are a semi circle piece of polycarbonate that helps us ride up the park barrier and into the park zone.

Figure 2: Middle goal in CAD so we can use a CNC to cut out a flat shape

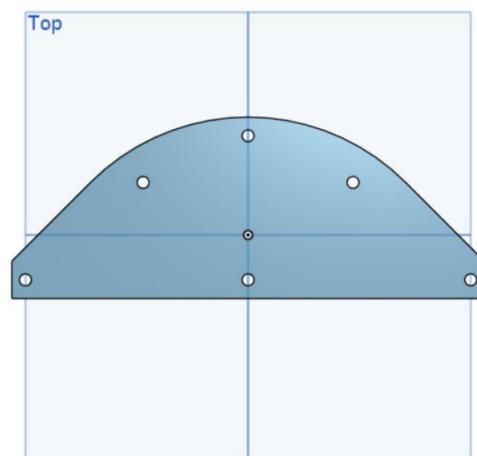


Figure 3: CAD of our long goal aligner

- ▶ Focus: Creating polycarbonate pieces using the CNC

Date: Sept 9, 2025

Members Involved

Daniel, Joshua, Brandon

Objective

First we CNC the polycarbonate pieces before attaching them to our robot. There are 3 main functions of our polycarbonate.

1. Semi-circle aligner
2. Park barrier sleds
3. Middle goal aligner/ramp

These pieces have holes drilled so we can screw them in for attaching.

Materials

- Polycarbonate
- Spray paint
- Screws
- Spacers
- Nylocks

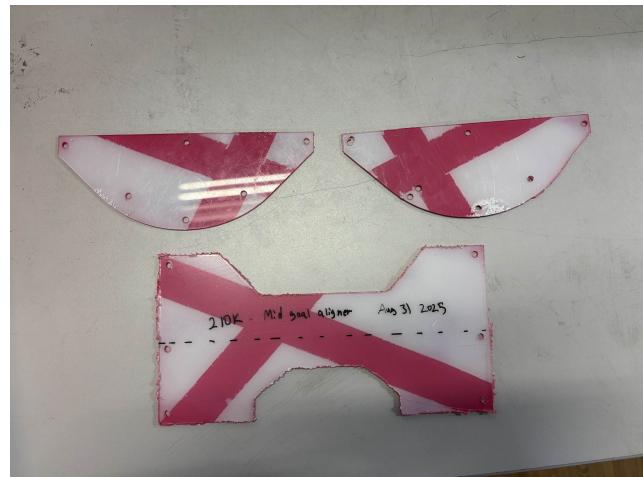


Figure 1: Painted aligner pieces that were cut using a CNC

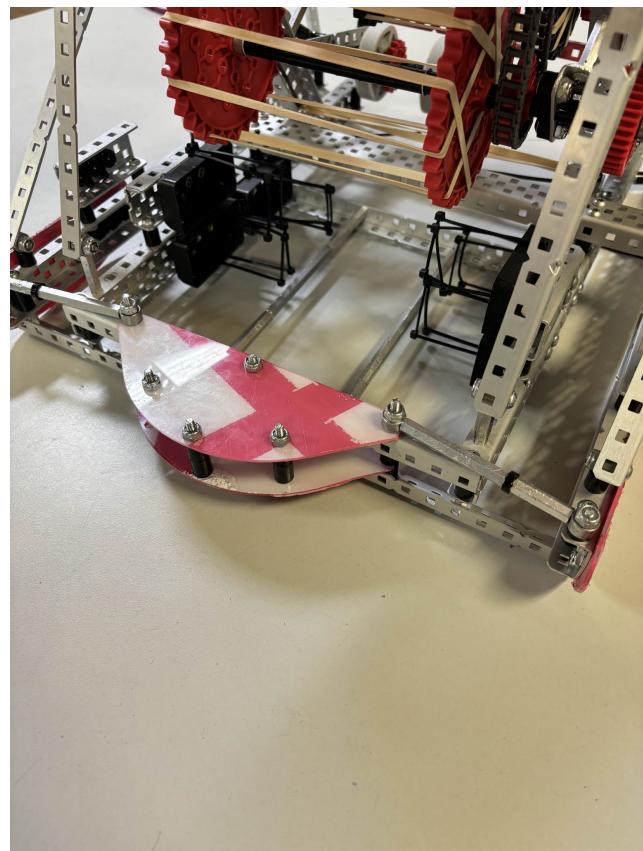


Figure 2: Long goal aligner attached to our robot.

- Focus: Creating polycarbonate pieces using the CNC

Date: Sept 9, 2025

Problems and Solutions

Our middle goal aligner needed to be bent in order to line up properly with the goal.

In order to bend the middle goal aligner. We used a vice to very slowly shape the bend until it was perfect. We then used clamps so the polycarbonate would hold its shape.

We noticed that our polycarbonate sleds were not perfectly lined with the drivetrain C-channel. Due to our zip tied bearings. This caused the hole containing the axle to bend.

We decided to cut our axles so they would not be touching the polycarbonate. This help smoothen potential drivetrain friction.

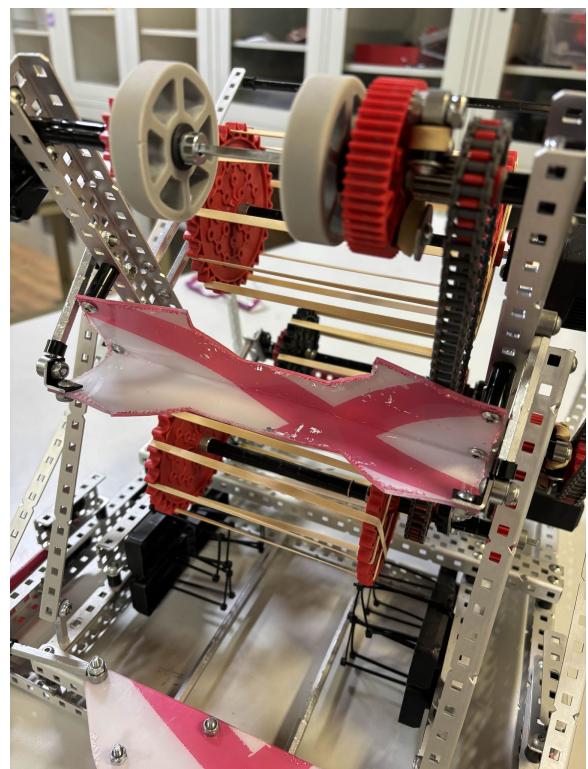


Figure 1: Middle goal aligner

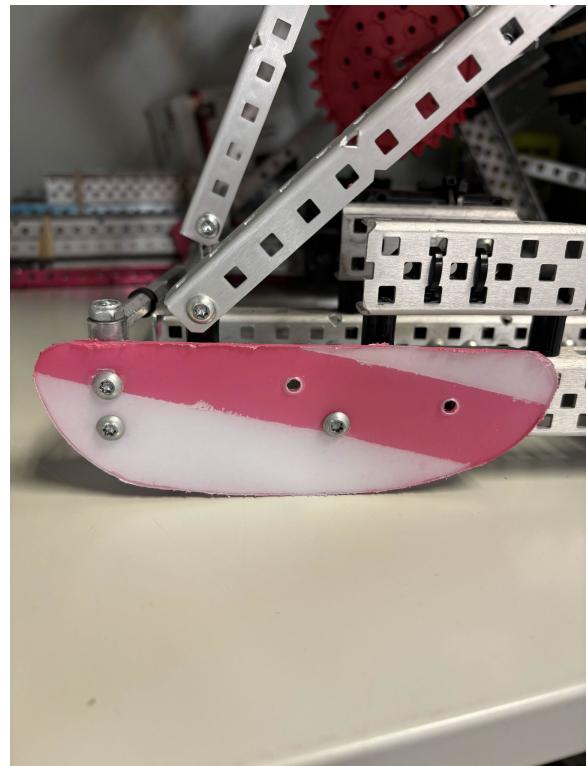


Figure 2: Picture of our sleds

- ▶ Focus: Adding wheels to our drivetrain frame

Date: Sept 9, 2025

Members Involved

Daniel. Bryan

Objective

We aim to finish building the wheels and so we can test the robot. This would allow us to drive around to test our intake system. We also aim to have minimal friction in our drivetrain.

Materials

- Gears
- Omni-wheels
- Screws
- Keps-nuts
- Nylocks
- Spacers
- Washers
- Inserts

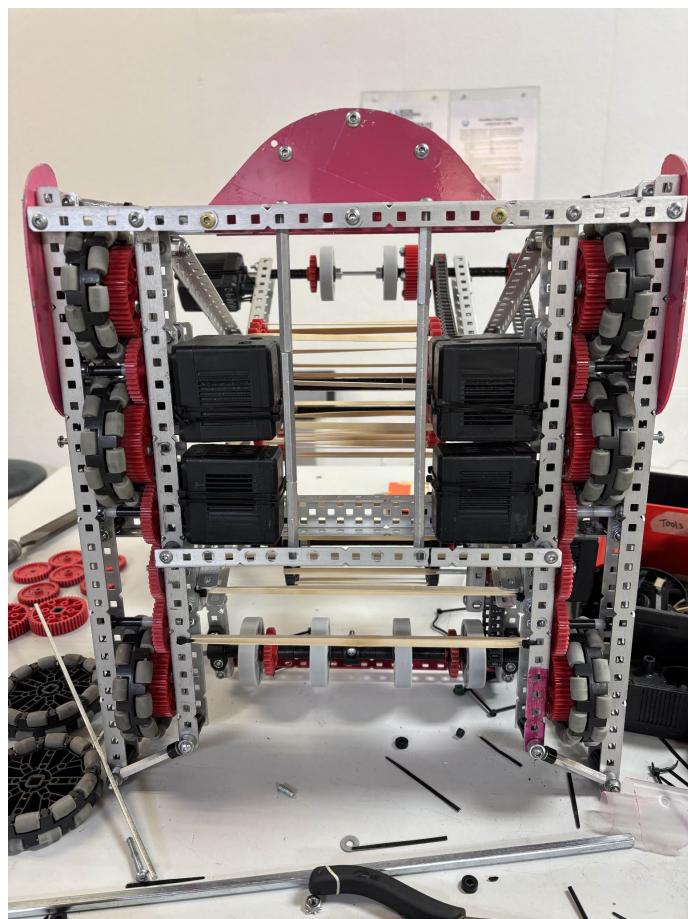


Figure 1: View of the standoff bracing across the bottom of the drivetrain.

Problems and Solutions

The friction was unusually high on the left side of our drivetrain.

We minimized the friction by isolating each wheel to determine which object was causing friction. We determined it to be a wheel being too tight and solved the issue.

- ▶ **Problem:** What resulted in teams performing well at MOA?

Date: Sept 9, 2025

Descore

Looking back at our experience at the Mall of America Signature event we found that both alliances in finals had the ability to descore large amounts of blocks at a time if they were left undefended. The ability to descore blocks opened up many opportunities for point swings. We wish to implement a version of a descore mechanism so we would also be able to descore blocks.

Blocking

We also noticed that teams could prevent blocks being descored by blocking. Notable examples of this include 11101B utilizing a metal piece of their robot to prevent blocks from coming out of the goal during semifinals and finals at MOA. Teams like 9123C and X were also able to use a polycarbonate piece that extended into the goal in order to prevent blocks from being pushed out. Blocking a descore allows you to protect and maintain control over scoring objectives, putting you ahead in a match.



Figure 1: 11101B using a C-channel on the back of their robot to block the long goal (MOA Finals)

- ▶ Focus: Finding methods to both block and descore blocks

Date: Sept 9, 2025

Looking through robots MOA, we found many possible options for descoring and blocking. We also researched Chinese teams to look at unique blocking options.

Ruiguan Hood

This design was very prominent at MOA as both 9123X and 9123C, teams that were in the finals, had this design. The Ruiguan hood is a piece of polycarbonate that is mounted near the top of their intake system. This allows them to descore and block the long goals.

C-Channel Blocker

A C-channel blocker is very simple to build as it is just a rigid piece of metal that sits in front of the long goals, preventing blocks from getting descored. This design could be made using a variety of metal parts, so it is very easy to modify.

New Start Hood

Another design we saw from 10222R was a hood that could rotate in order to block the goal. This would be a piston powered mechanism that can toggle in order to put the hood down to block the goal.



Figure 1: 9123C hood piece extending into the goal to prevent scoring



Figure 2: 9123X using their hood piece to descore blocks from the long goal



Figure 3: 10222R's hood that is able to rotate to prevent blocks from getting descored

Selecting a hood option

Select Solution

- Focus: Choosing an option for a hood

Date: Sept 9, 2025

After researching possible options, we decided to rank some of our designs using the following criteria.

- Compactness (1 - 5)** - How much space does this design take?
- Build feasibility (1 - 5)** - Ranked on whether or not the design is easy to build. We also take into consideration if there are any parts that are needed such as pistons and custom made polycarbonate.
- Tuning (1 - 5)** - Is it easy to make adjustments to this design?
- Descore capability (1 - 5)** - Can this design open up options for descore blocks?
- Blocking ability (1 - 5)** - Ranked on how well the design is able to block the long goals

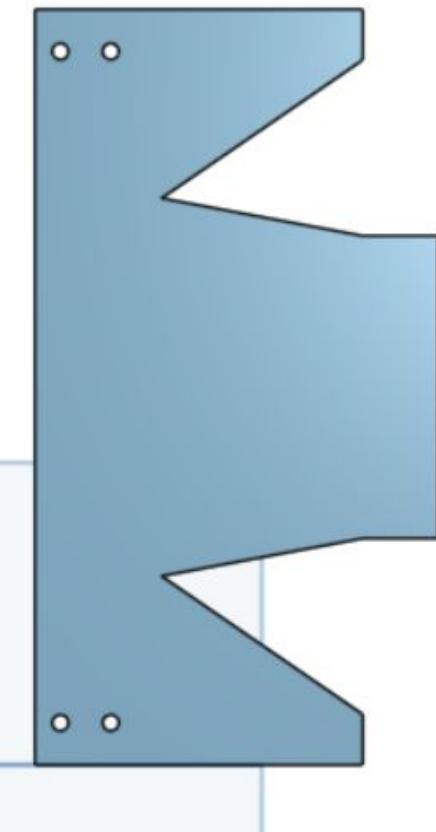
<u>Criteria</u> <u>Options</u>	Compactness	Build feasibility	Tuning	Descore capability	Blocking ability	Total /20
Ruiguan Hood	3	4	5	5	4	21
New Start Hood	2	3	3	4	5	17
C-Channel blocker	2	5	3	1	5	16
No hood (just rollers)	4	4	4	2	3	17

- ▶ Focus: Using CAD to design our hood pieces

Date: Sept 10, 2025

Hood Piece

After considering all of our options. We planned to build the Ruiguan hood as it provides the most versatility in both offense and defence. In order to build the Ruiguan hood, we first used CAD to find dimensions for our polycarbonate piece. This piece should have a section that is able to enter into the long tube. This allows for both blocking and descoring.



Polycarbonate Mount

We also plan on using CAD to create a polycarbonate mount. Using polycarbonate as a mount allows us to create custom geometry that is not bound by the limitation of VEX parts.

Attaching

We plan to attach this hood at the very top of our intake system, once all parts have been designed and cut out using a CNC machine.

Figure 1: Polycarbonate hood piece in CAD

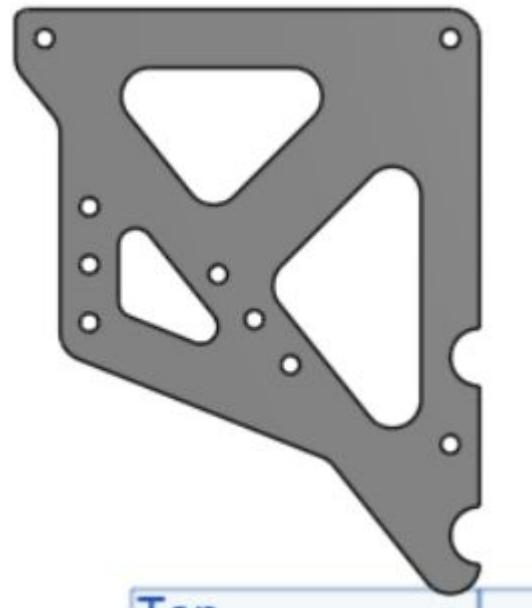


Figure 2: Polycarbonate mount piece

- ▶ Focus: Creating our hood mount and hood piece

Date: Sept 10, 2025

Members Involved

Daniel, Bryan

Objective

We aim to complete our hood by attaching our hood piece to a custom polycarbonate mount. This would allow us to defend long goals by blocking and allow us to descore by knocking the blocks out of the goal. We also created our rotating hood piece, this would allow us to score, descore and block at the long goal.

Materials

- Screws
- Nylocks
- Polycarbonate
- Gussets
- Angle bars

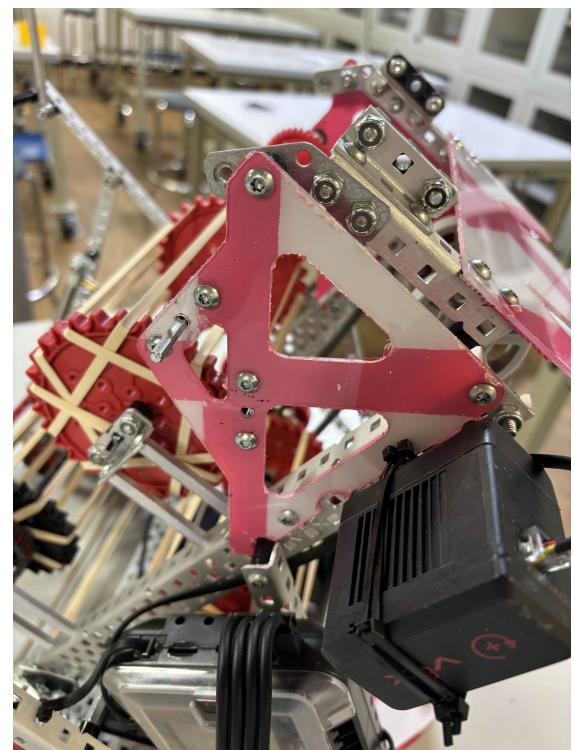


Figure 1: Polycarbonate mount used to attach our hood piece.

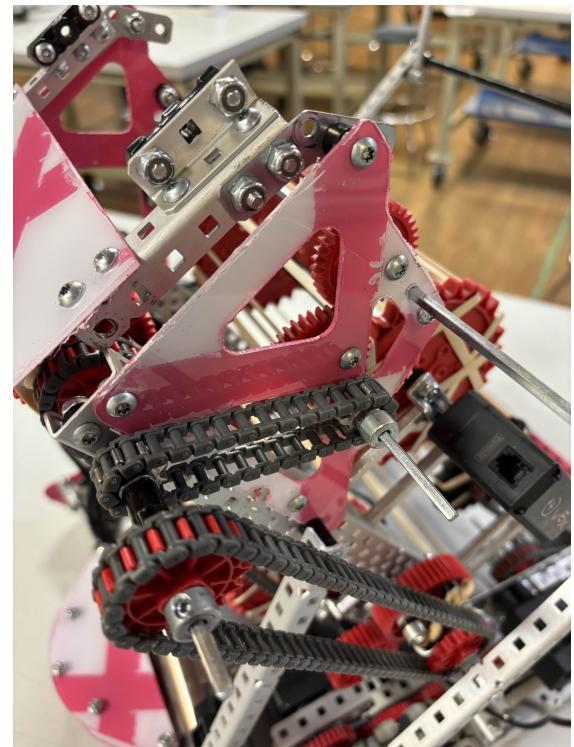


Figure 2: Side view of how we powered a small roller stage and bottom roller

- Focus: Creating our hood mount and hood piece

Date: Sept 10, 2025

Problems and Solutions

We found that in the natural resting position, the hood would be too low and be unable to block the long goal.

We solve this, we created a hardstop to ensure the hood to be at a perfect height for descoring and blocking

Next Steps

Creating top rollers for smoother scoring.

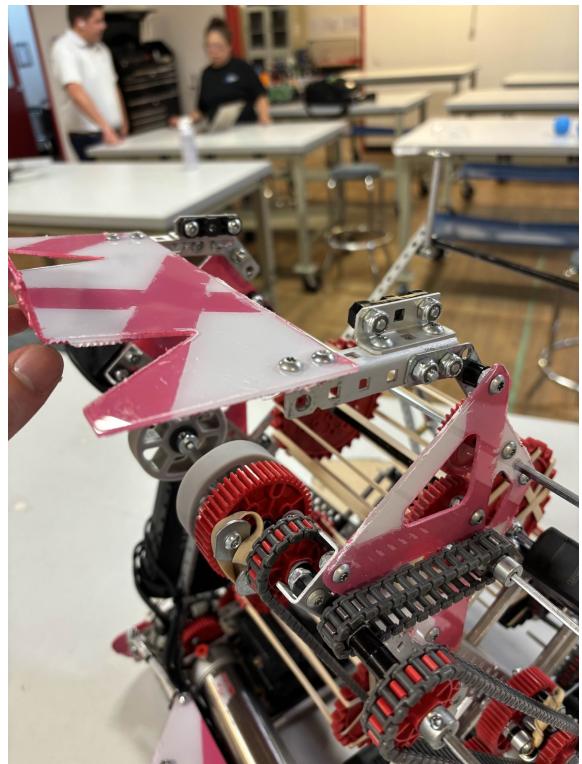


Figure 1: Side view of our hood piece in an up position

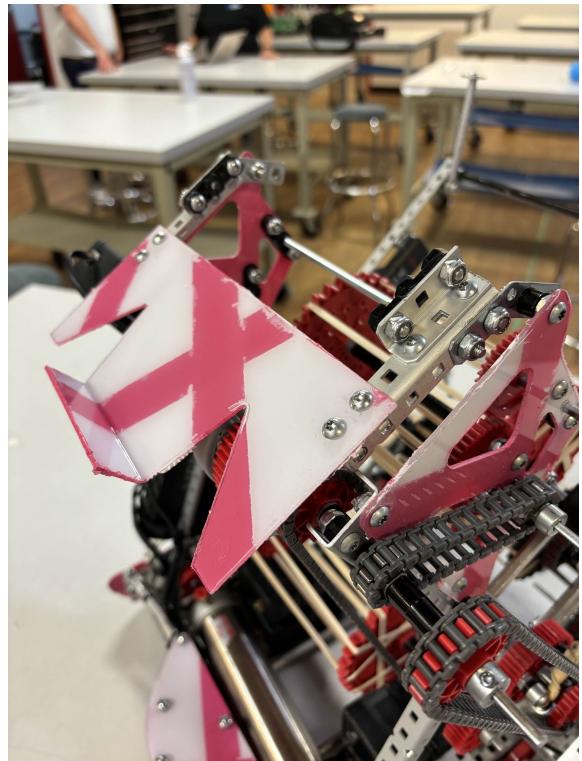


Figure 2: Hood piece in a natural resting position

- ▶ Focus: Creating and powering a small roller stage

Date: Sept 11, 2025

Members Involved

Daniel, Bryan

Objective

Create a small top stage roller to help smoothen our scoring. This small roller should also prevent deadzones and jamming within our intake system.

Materials

- Gears
- Axles
- Flexwheels
- Axle collars
- Sprockets
- Chain
- Spacers

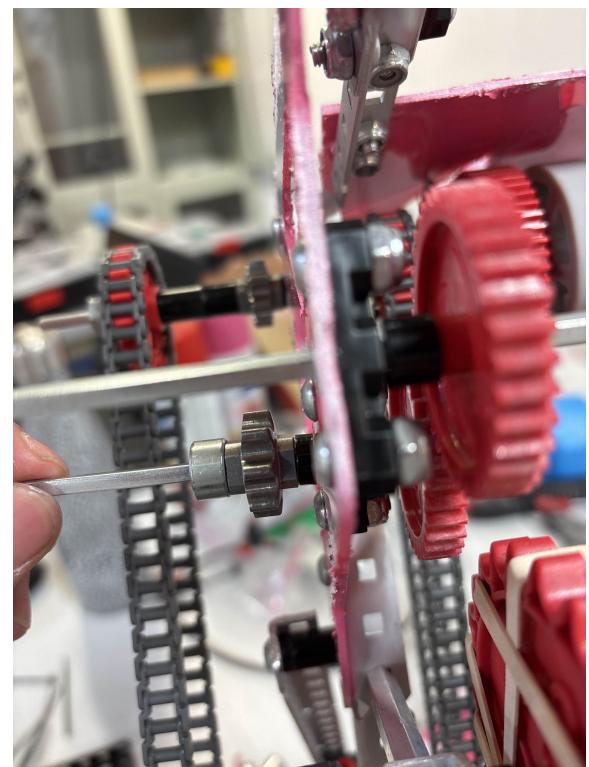


Figure 1: Cantilever gear to power our top stage intake

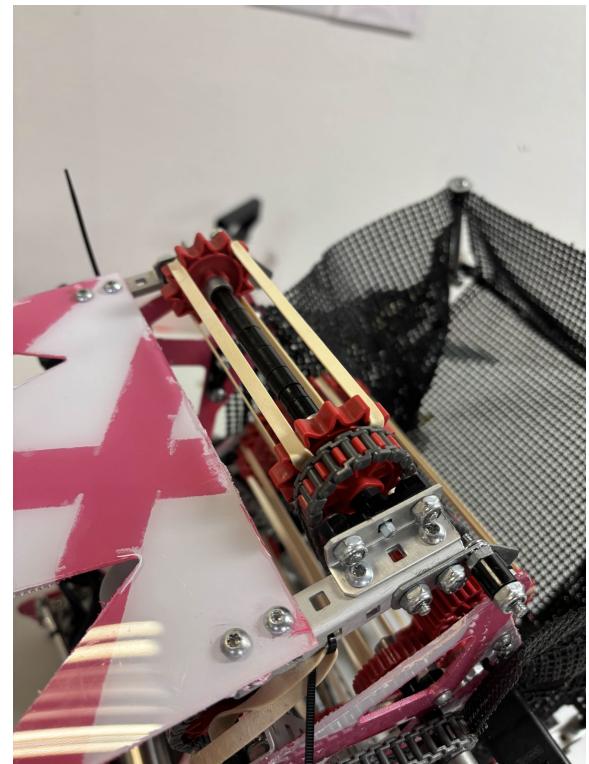


Figure 2: Top stage roller

- Focus: Creating and powering a small roller stage

Date: Sept 11, 2025

Problems and Solutions

The cantilevered gear was very unstable and would bend.

We minimized the bending by equalizing the weight on both sides and making it as close to the polycarbonate as possible

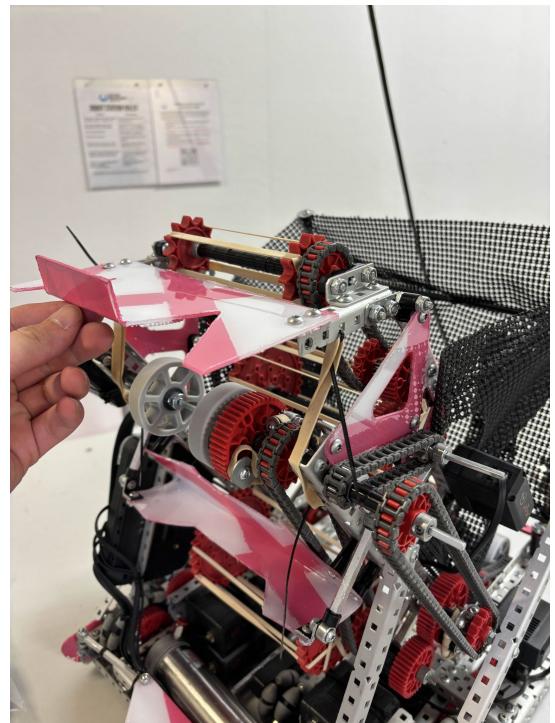


Figure 1: Hood with the top stage roller

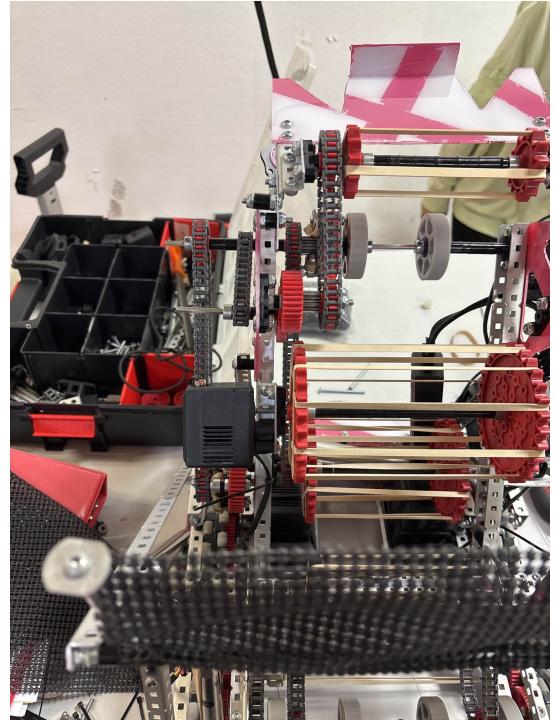


Figure 2: View of our chain and gear system for powering our top roller

- ▶ Focus: Creating a basket

Date: Sept 11, 2025

Mesh Basket Planning

Following our general design and plan of our robot. We planned out the specifics of our basket. Our basket would be made of mesh because the mesh can be easily warped into a custom shape. We plan to make the mesh in a way where blocks would funnel towards the middle so the blocks can enter back into the intake system.

Our basket would be made of three pieces of mesh. Two pieces would be the wall, creating the sides of our basket. One piece would be a large angled piece of mesh to help funnel the blocks towards our intake.



Figure 1: Example of a mesh basket (VCAD, 40000A)

- ▶ Focus: Creating a mesh basket for hoarding blocks

Date: Sept 11, 2025

Members Involved

Daniel

Objective

We created a mesh basket in order to hold blocks. This would be the first version and we expect to need to tune it in the future.

Materials

- Mesh
-



Figure 1: Top view of our mesh basket

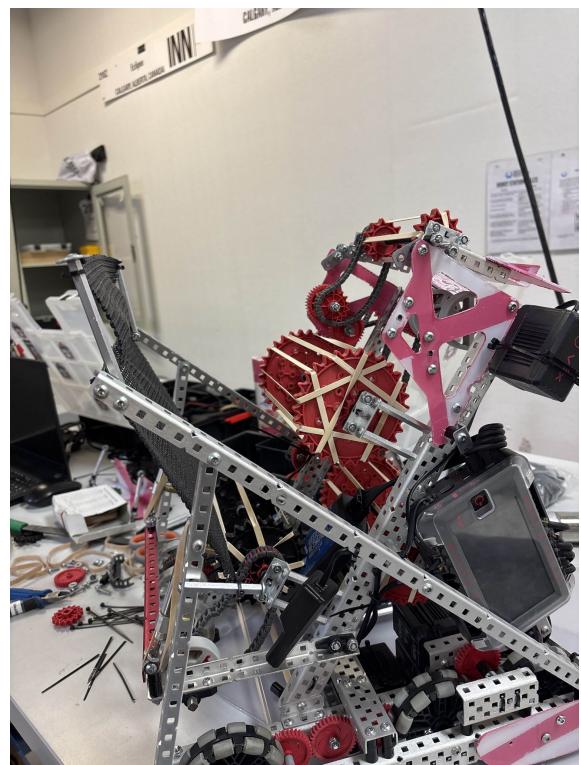


Figure 2: Side view of robot before the basket is created

- Focus: Creating a mesh basket for hoarding blocks

Date: Sept 11, 2025

Problems and Solutions

We realized the mesh was too loose and would expand out when there were blocks pressing against it.

We tightened the mesh by using zip ties to tie the mesh to itself, tightening it so it would hold its shape.

Next Steps

We plan to tune our intake and creating our hardstop/double park mechanism.

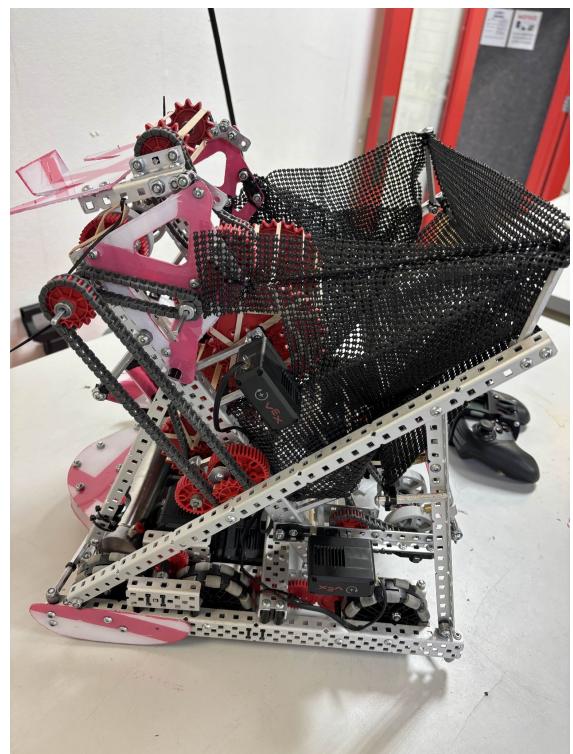


Figure 1: Side view of our mesh basket

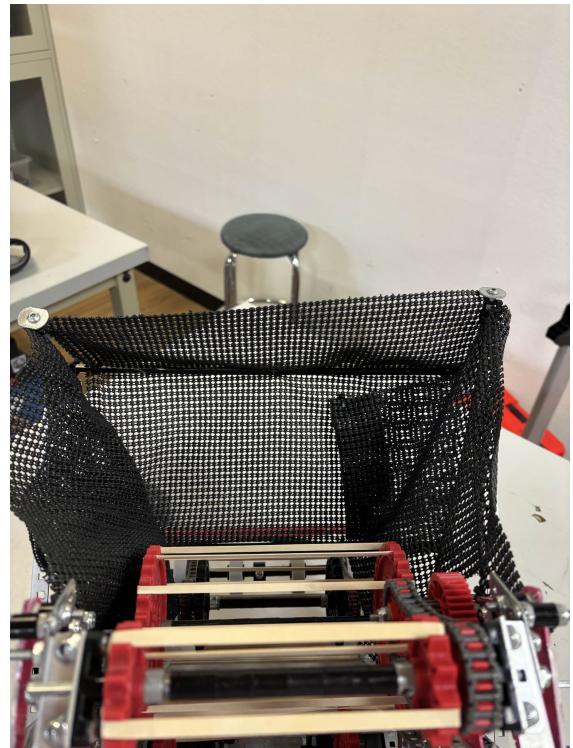


Figure 2: Top view of our mesh basket

- ▶ Focus: Double park mechanism

Date: Sept 11, 2025

Members Involved

Brandon

Objective

We want to mount our pistons to the intake to act as a hardstop. These pistons are also able to push down and act as our double park mechanism. In order for a little bit of flexibility in our intake, we cut a flexwheel into a triangle shape to use as a spring, meaning the intake would not be perfectly still. The movement in the intake should allow for better pickup and bottom goal scoring.

Materials

- Pistons
- Screws
- Nylocks
- Keps-nuts
- Flexwheel
- Standoffs
- Spacers

Next Steps

We plan on testing various parts of our robot, mainly the basket. This will allow us to get an idea on what needs to be improved.



Figure 1: Piston attached to our intake



Figure 2: Cut flexwheel wrapped around our piston. This acts as the spring

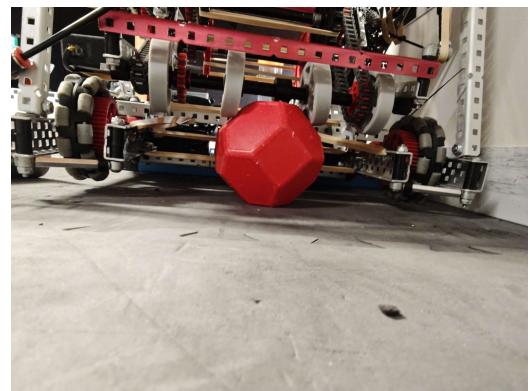


Figure 3: Double park mechanism in action, pushing a block to lift us off the ground.

Highlander Signature Event Analysis

Strategy

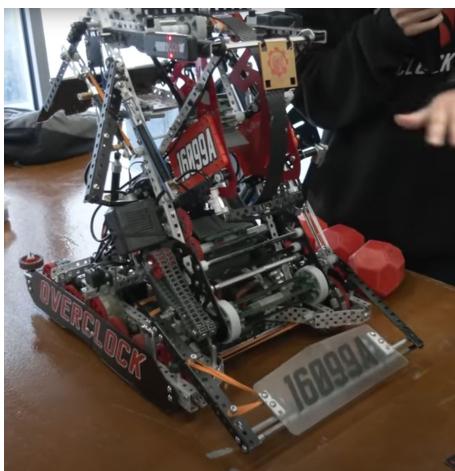
- Focus: To analyze other teams' matches to inform our strategy.

Date: Sep 14, 2025

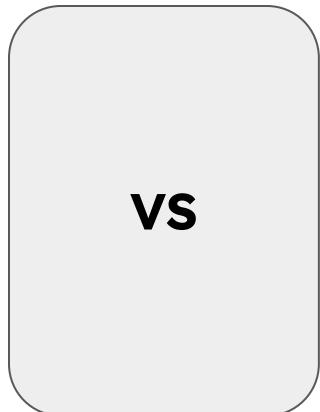
This past weekend, the Highlander Summit sig. event happened in New Jersey. We collectively sat down to watch the eliminations matches, and took notes on our own analysis of teams' strategies in the next slide.



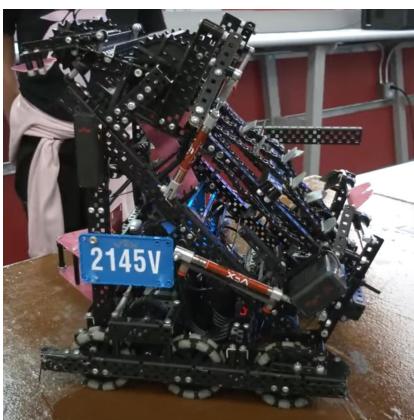
The grand finals featured these teams:



Red, 16099A, via FUN
Pits & Parts



Blue, 1168A, via FUN
Pits & Parts



Red, 2145V, via FUN
Pits & Parts



Blue, 2054V, via FUN
Pits & Parts

- Focus: Elimination Match Semi-Final 1-1

Date: Sept 14, 2025

Red: 16099A + 2145V **Blue:** 2787V + 9364V

Autonomous

- 2787V runs early control zone auto; 16099A runs 9-ball auto (misses).
- 9364V use MoA auto; 2145V run 4+3 mid-goal auto for mid goal control zone.
- Blue wins auto (control zone secured).

Driver Control

- 16099A and 2145V focus on matchloads; 2787V attempts a wing descore.
- 16099A swaps to top goal, later returns to bottom goal to score successfully.
- 2787V pushes two blocks back into control zone but are overusing wings.
- 2145V pushes 9364V off the top long goal; 9364V respond with a wing descore.
- Middle goal remains uncontested for most of match.

Results

- Red wins due to mid-goal advantage and control bonus.

Reflections & Next Steps

- Avoid overusing wings
- Secure uncontested mid-goals, they provide major point swings because of their control zone bonus.
- Chose different autos depending on strategy. Most autos should prioritize fast access to the mathloaders
- Try to secure priority over a long goal control zone at all times

- Focus: Elimination Match Semi-Final 2-1

Date: Sept 14, 2025

Red: 2054V + 1168A **Blue:** 3150V + 8189X

Autonomous

- 3150V and 1168A run standard autos; Infinity uses same.
- 2054V runs unique mid + long goal auto.
- Red wins auto due to mid-goal bonus.

Driver Control

- 1168A scores early and secures long goal control zone.
- 2054V focuses on match loads; 8198X plays wings for descore.
- Blue alliance struggles with coordination their robots block each other from moving.
- Red plays consistently: securing long goal control zones. They coordinate switching between goals very nicely
- Blue loses top long goal priority due to poor communication and lack of blocks for scoring

Results

- Red maintains long goal control and wins the match; Blue fails to capitalize off of open opportunities

Reflections & Next Steps

- Maintain active communication between alliances and avoid friendly collisions.
- If descoring, ensure bots are loaded with blocks and ready to score.
- If down auto, immediately fill long goals to recover points.

- ▶ Focus: Elimination Match Finals 1

Date: Sept 14, 2025

Red: 16099A + 2145V **Blue:** 2054V + 1168A

Autonomous

- 16099A and Victory miss.
- Pika cancels mid goal.
- Blue gains early control—16099A pre-fills matchload tube first.

Driver Control

- Pika leaves bottom; 16099A takes and scores.
- Victory wins long-goal duel vs PSU.
- Pika clears opponent park zone (ineffective).
- PSU vs Victory stalemate at long goal.
- 16099A defends mid and descored efficiently.
- Victory parks early—PSU retakes control.

Result:

- Red wins via superior coordination and mid-goal control.
- Blue loses tempo once Pika leaves bottom goal.

Reflections & Next Steps

- Don't clear opponent park zone.
- Maintain pressure during matchloads.
- Communicate 2v1 mid-goal timing.
- If long goal lost, refocus mid or double-park.

► Focus: Elimination Match Finals 2

Date: Sept 14, 2025

Red: 16099A + 2145V **Blue:** 2054V + 1168A

Autonomous

- Both **red** bots miss auto.
- **Blue** secures double control zones.

Driver Control

- **OCA** left undefended; fills long goal completely.
- **Victory** descored own color.
- **Unicorns** perform damage-control descoreing for **red**.
- **Red** holds bottom + mid; **blue** focuses only on long.
- **Pika** left 1 v 2 mid fight; **Victory** unprepared (no balls).
- **Blue** loses tempo—**OCA** uncontested too long.

Result

- **Red wins** through better mid-goal balance and awareness.
- **Blue** strong start but coordination fades.

Reflections & Next Steps

- If undefended, fill long goal before contesting mid.
- Decide early: mid fight vs goal fill.
- Stay pre-loaded before every engagement.
- Clear park zone before double park.

- Focus: Creating a tensioner for our chains

Date: Sept 16, 2025

Members Involved

Daniel

Objective

Create a tensioner to tighten the chains. This will ensure the intake will run smoothly without the chains skipping. The chains being able to move also means our double park mechanism works alongside our intake system.

Materials

- Rubber bands
- Standoffs
- Axle collars
- Screws
- Keps-nuts
- Chain
- Nylocks

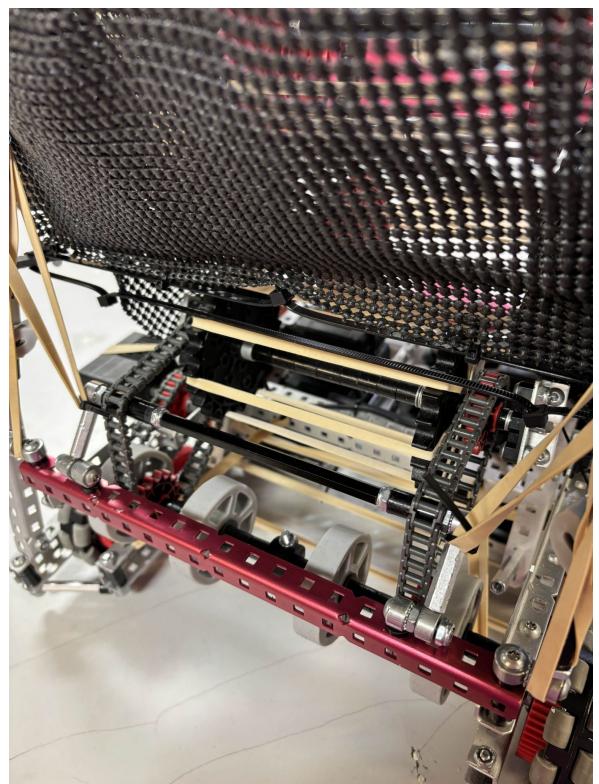


Figure 1: Picture of our tensioner



Figure 2: Close up of the tensioner

- ▶ Problem: Testing our intake system

Date: Sept 22, 2025

Purpose

The purpose of this test was to find any issues in our intake system. Possible issues include deadzones, jamming points and overall friction.

Procedure

1. Have one team member ready with a stopwatch
2. When ready, start running the intake and have another member push 6 blocks into the first stage intake
3. Time how long it takes for all 6 blocks to go through the intake system and out the other side

Analysis and Conclusion

After the trials, we noticed that some jamming occurred near the top of our intake system.

Table 1: Intaking 6 blocks through our system, t (s), in relation to trial number

Trial Number	Time to Clear 6 Blocks, t (s)
1	4.47
2	2.12
3	2.56
4	4.18
5	N/A
6	3.79
7	2.46
8	N/A
9	4.37
10	3.26

- ▶ **Problem:** Various issues we found with our intake system

Date: Sept 22, 2025

Major Intake Issues

After testing our intake system. We noticed four main issues

1. There is a gap in our rollers near the top, so when there are multiple blocks in the system, some of the blocks would go through the gap and leave our intake system.
2. The top roller stage and hood that was mounted on the polycarbonate could flex and move from side to side
 - a. This caused a lot of friction and strain on the 11W motor because the chains could bend
 - b. The flexing polycarbonate mount also meant that our hood would move around when driving. So we attempted to descore, the hood would not line up properly and miss
3. We also noticed that the top stage roller was too low, this would result in the rubber bands pressing down, causing the blocks to get stuck
 - a. This defeated the original purpose of the roller, which was to help the blocks score into the tubes.
4. We also noticed that the gear attached to our top flex wheel roller was too rigid. When a block contacted the gear, it would lead to the block getting stuck. This would impact our scoring as there is always a possibility of the block getting stuck.

- ▶ Focus: Planning out hood and intake system changes

Date: Sept 22, 2025

Planning Hood Changes

We planned out a few adjustments we wanted to make for our hood and intake

1. The biggest change was to switch from a flexible polycarbonate hood mount to a rigid standoff and angle bar mount.
 - a. This new mount would be the same structure that we used for our rollers, this should help solve the flexing issues
2. Another change we wanted to make is to remove the very top roller that was causing the jamming issues
 - a. Instead, we plan to add some compression using mesh and rubber bands to help with scoring
3. Thirdly, to patch up the hole in our intake system, we plan to add standoffs and rubber bands to prevent blocks from travelling through that hole
4. We plan to remove the ratchets entirely and split up our rollers so we can power them separately.

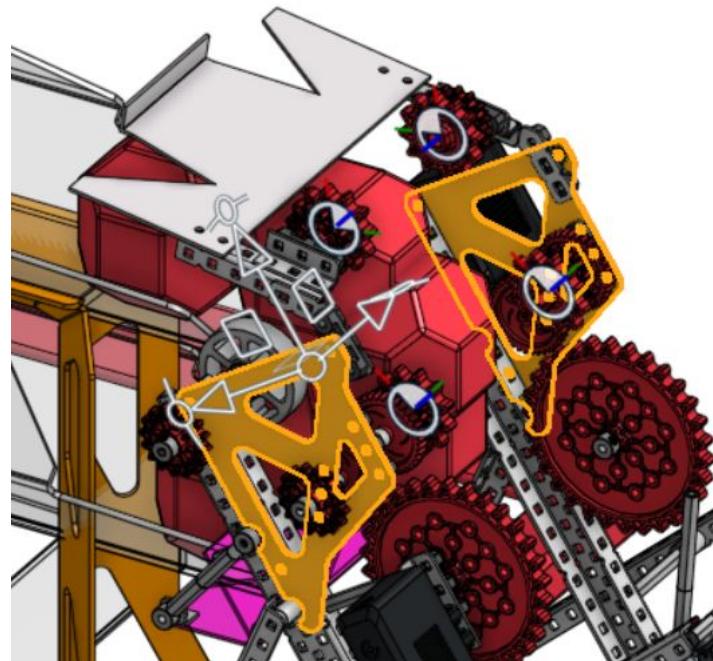


Figure 1: Highlighted polycarbonate pieces that we plan to remove

- ▶ Focus: Adjustments to the hood

Date: Sept 24, 2025

Members Involved

Daniel, Bryan

Objective

Follow the plan we laid out and make adjustments to our hood. This includes building a more stable and rigid mount, creating a compression stage for blocks and adding standoffs to close the gap.

Materials

- Polycarbonate (3.5" x 8.0")

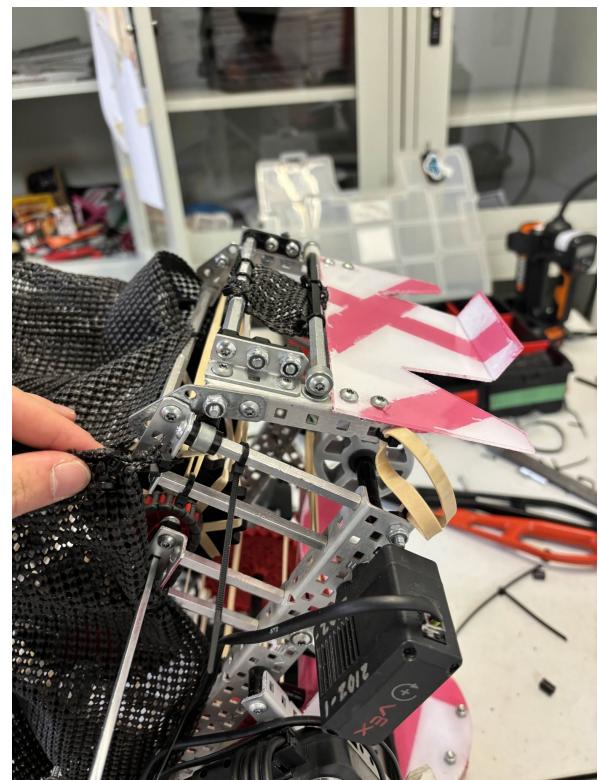


Figure 1: Side view of the adjusted hood



Figure 2: Picture of our compression

- ▶ Focus: Adjustments to the hood

Date: Sept 24, 2025

Problems and Solutions

We noticed that our original hood is a perfect fit between the standoffs. This means we cannot create a screw joint for the hood.

We used a short screw and screwed in our hood. The nylock was not tight enough to prevent movement, but not loose enough that the hood could move around horizontally.

Switching to 5.5W motors proved to be a slight issue as our bottom roller structure was not meant for a roller to be attached to it.

We made the necessary adjustments, including switching angle bars and changing spacers to accommodate the 5.5W motor.

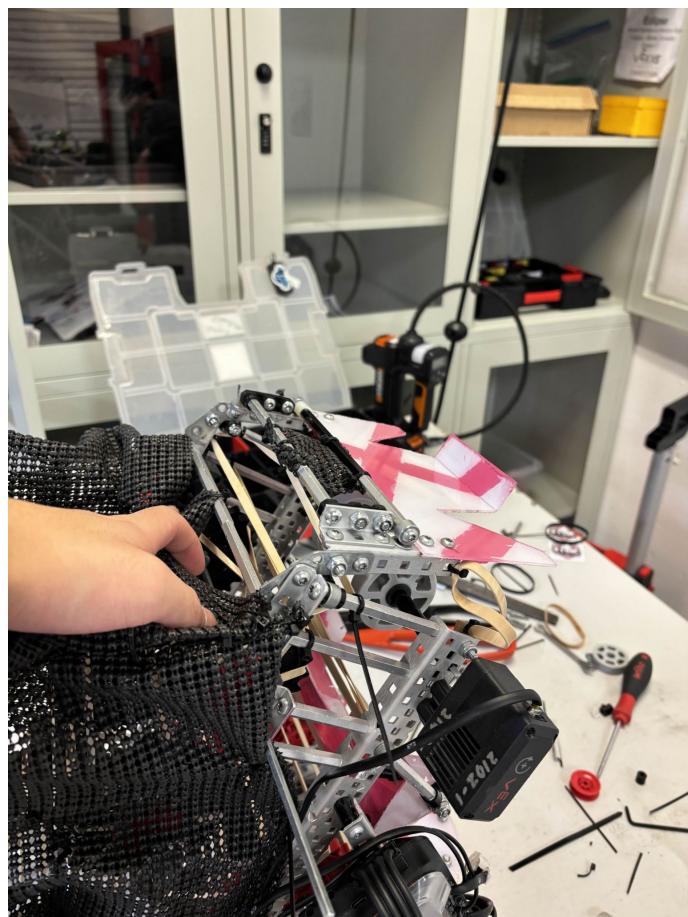


Figure 1: Side view of our adjusted intake and hood.

Next Steps

We plan on making small adjustments to our basket.

- ▶ Focus: Tuning our basket

Date: Sept 25, 2025

Members Involved

Daniel, Brandon

Objective

There were three main objectives. Firstly, we needed to create a mesh top so blocks would not fly out the top of our basket. Secondly, we created mesh funnels by shaping and tightening our side walls so the blocks would funnel towards the middle into the rollers. Thirdly, we wanted to change the front of the hoard as we noticed that the mesh was too tight to allow blocks to travel back into the intake.

We changed the front section of mesh to rubber bands because rubber bands would flex, allowing for blocks to travel in and out of the basket.

Materials

- Zip ties
- Mesh
- Rubber bands

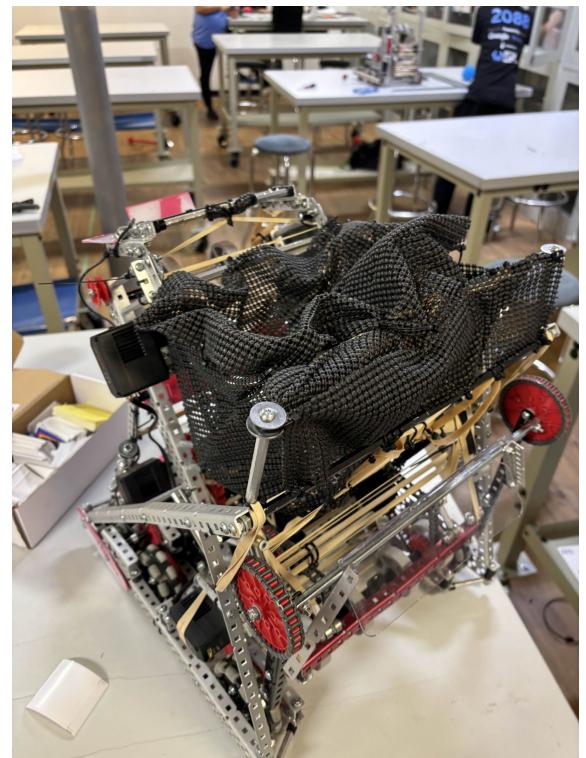


Figure 1: Front view of our mesh cover



Figure 2: Mesh cover with blocks inside

- Focus: Tuning our basket

Date: Sept 25, 2025

Problems and Solutions

After creating the basket cover, we realized that if there were too many blocks in the basket, the mesh cover would press the blocks down into our rollers. This would slow down our intake system.

To solve this issue, we used very loose zip ties to create tolerance in our top cover. This would allow the top cover to move up when there are many blocks in the basket.

There were some funneling issues that we noticed.

We used a unique rubber band pattern to help funnel the blocks into the center of our rollers

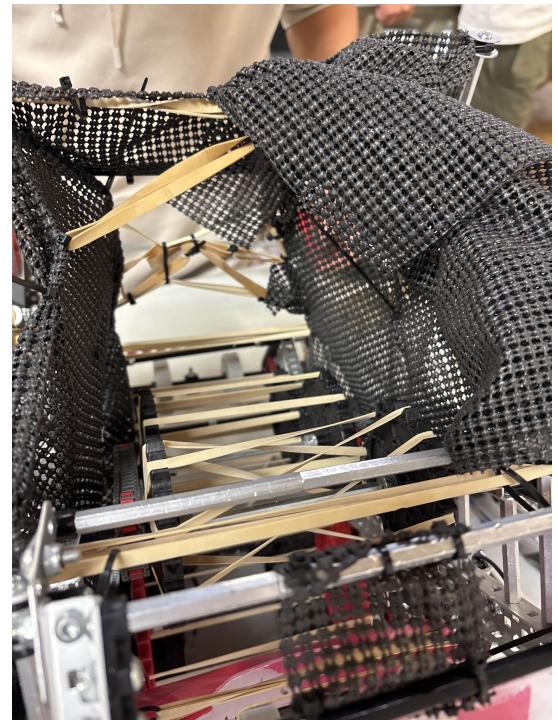


Figure 1: A look into our basket with our new rubber banded front.

Next Steps

Testing our hood by setting up various trials.

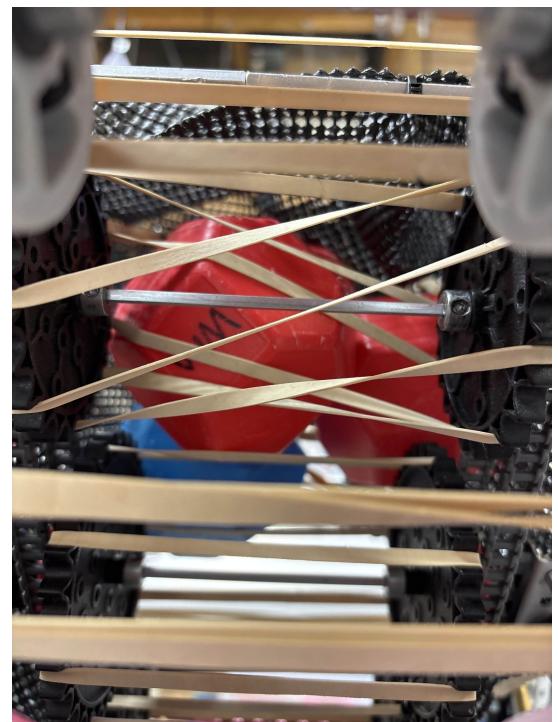


Figure 1: Unique rubber band pattern

- ▶ Problem: Testing our basket and intake system

Date: Sept 26, 2025

Purpose

The purpose of this test is to find issues in our basket and intake system.

Procedure

1. First intake 7 blocks into our basket
2. Have a team member prepared with a stopwatch
3. When ready, position the robot at one end of the long goal and score from the basket
 - a. Time how long it takes for all 7 blocks to be scored

Analysis and Conclusion

After the trials, we found that our basket had many jamming issues.

Table 1: Scoring 7 blocks from the basket, t (s), in relation to trial number

Trial Number	Time to Score 7 Blocks, t (s)
1	11.04
2	5.09
3	N/A
4	13.59
5	8.77
6	12.62
7	N/A
8	4.20
9	N/A
10	N/A

- ▶ Problem: Issues with our basket

Date: Sept 26, 2025

Jamming Issues

One of the main issues we noticed after our testing was that our basket is too wide. This means our blocks can get stuck on our sprockets, leading to the blocks losing contact with the rubber band rollers. We also noticed that the mesh is at a perfect space where when there are two blocks placed next to each other in the basket, the mesh would press the two blocks together, preventing them from moving up or down.



Figure 1: Two blocks stuck on sprockets. Causing them to get stuck

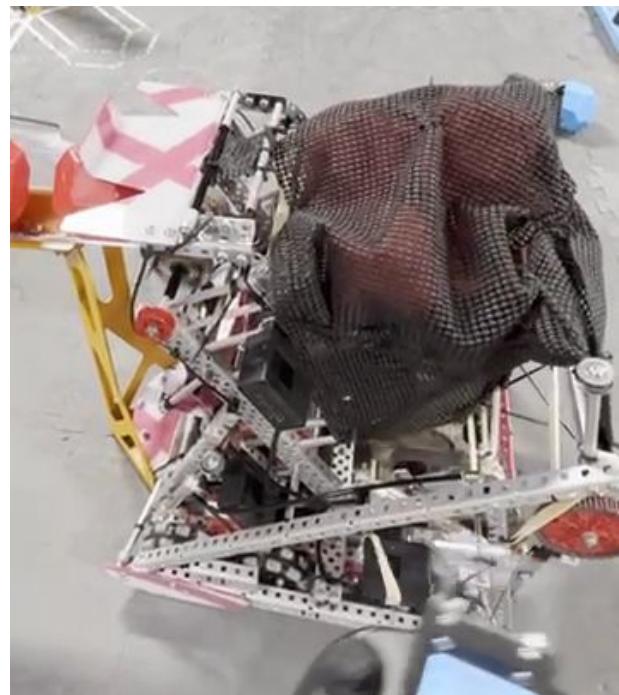


Figure 2: Blocks jamming when we attempt to score

► Focus: Brainstorming possible additions

Date: Sept 26, 2025

Agitators

The first addition we thought of creating is agitators. Agitators are an addition to our basket that can help “agitate” the blocks into moving. There are a variety options that we could create

Zip Ties

One of the first ideas we had was to utilize zip ties that can help knock the blocks in our baskets towards our. These zip ties can be attached our sprockets and rubber band rollers

Flex Wheels

We thought of adding either an addition flex wheel stage in our basket or adding a flex wheel onto our existing rubber band roller stage. This would allow more contact with the blocks in our basket, causing the blocks to move back into our intake system.

Flaps

Another idea we had was to wrap flaps around our existing sprockets. These flaps would be spread out along the circumference of the sprocket and would help knock out any stuck blocks in the basket.

Funnels

Another idea we thought of implementing was to add funnels inside of the basket. This would help the blocks go towards the center of our robot. Funnels could be shaped in a variety of ways, some methods we thought of are

1. Shaping mesh to create funnels
2. Using polycarbonate to create funnels
3. Using standoffs to create rigid funnels

- Focus: Selecting a design for agitators

Date: Sept 26, 2025

After brainstorming possible agitator options, we ranked the designs we thought of using the following criteria.

1. **Ease of build (1 - 5)** - How easy can this be implemented
2. **Space needed (1 - 5)** - Does this design take up a lot of space? Will it interfere with other systems?
3. **Weight (1 - 5)** - How heavy or light would this addition be?
4. **Ease of tuning (1 - 5)** - Can adjustments be implemented easily when tuning?

<u>Criteria</u> <u>Options</u>	Ease of Build	Space Needed	Weight	Ease of Tuning	Total /20
Zip Ties	4	3	5	1	13
Flex Wheels	5	5	3	3	16
Sprocket	4	2	3	3	12

- ▶ Focus: Planning out our agitator

Date: Sept 26, 2025

Flex Wheel Agitator

Following our decision matrix we plan to implement a unique agitator. We plan to 3D print a jig that allows us to smoothly cut a 3 inch flex wheel in half. Using half of the flex wheel, we are able to attach it to our existing roller. We plan to add this flex wheel to one side of the roller, allowing for biased funneling towards the center of our intake system.

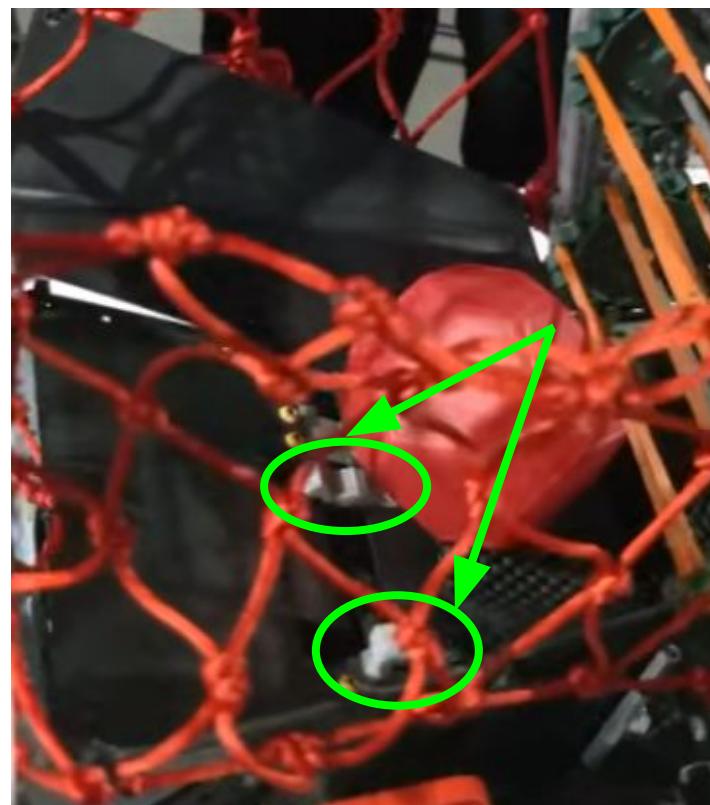


Figure 1: Flex wheel agitators used by 11101B (YouTube, FUN Interview)

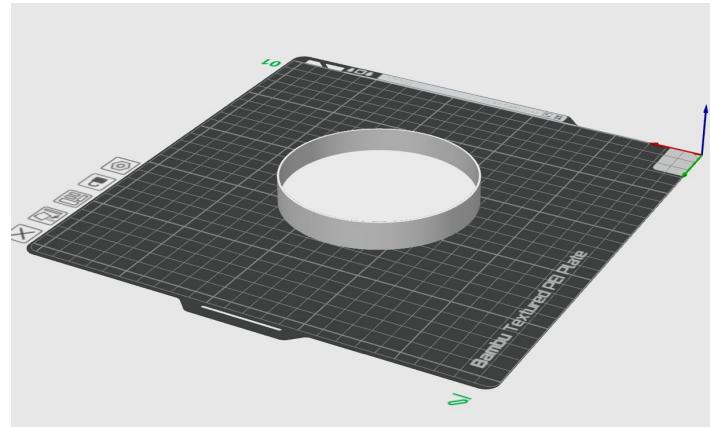


Figure 2: 3D print design of the jig

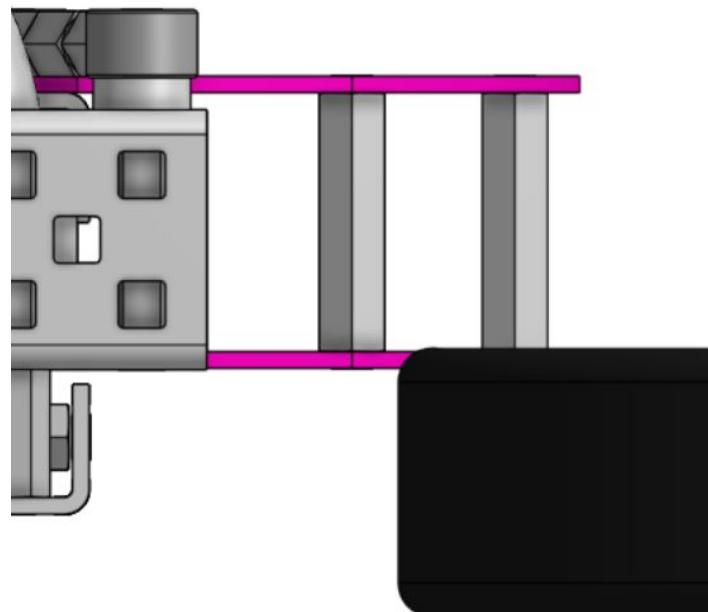
- ▶ **Problem:** We are unable to go into the park zone

Date: Sept 26, 2025

When testing our parking abilities. We noticed a two main issues

Issue 1 - Aligner

1. When attempting to park, we noticed that a section of our aligner was mounted too low. Meaning when we drive into the park barrier, it would ram into the barrier and deflect us away. This would prevent us from driving into the park zone at all.



Issue 2 - Sleds

2. Another problem we found was with the polycarbonate sleds themselves. The angle created by the sleds was too steep. Because of this issue, we would struggle to ride into the park zone when driving into it.

Figure 1: A diagram showing how our aligner hits the park barrier

- ▶ Focus: Using CAD to redesign our sleds

Date: Sept 26, 2025

CAD Redesigning

We used CAD to redesign and simulate tests on our sleds. For our new sleds, we designed a shallower angle allowing for the drivetrain to ride up the park zone. We designed a curve in the bottom sled, allowing for smoother driving when going across the park barrier. We plan on cutting them out and switching the old sleds out for our new design.

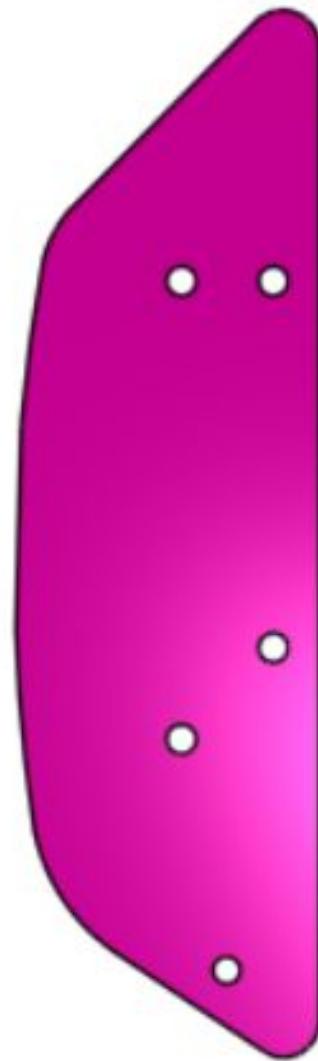


Figure 1: CAD of our new sled design

- ▶ Focus: Replacing our old sleds for the new sleds

Date: Sept 27, 2025

Members Involved

Daniel

Objective

The plan was to remove the old sleds that do not work and replace them with our newly designed pieces.

Materials

- Polycarbonate pieces
- Screws
- Nylocks

Problems and Solutions

In order to replace our old sleds, we needed to completely remove a wheel because one of the screw holes lines up with our screw jointed wheel. When we re-attached the wheel, we found there to be an increase in friction.

In order to solve these friction issues, we first isolated our system and found the issue to be a kepsnut tightened the wrong way.

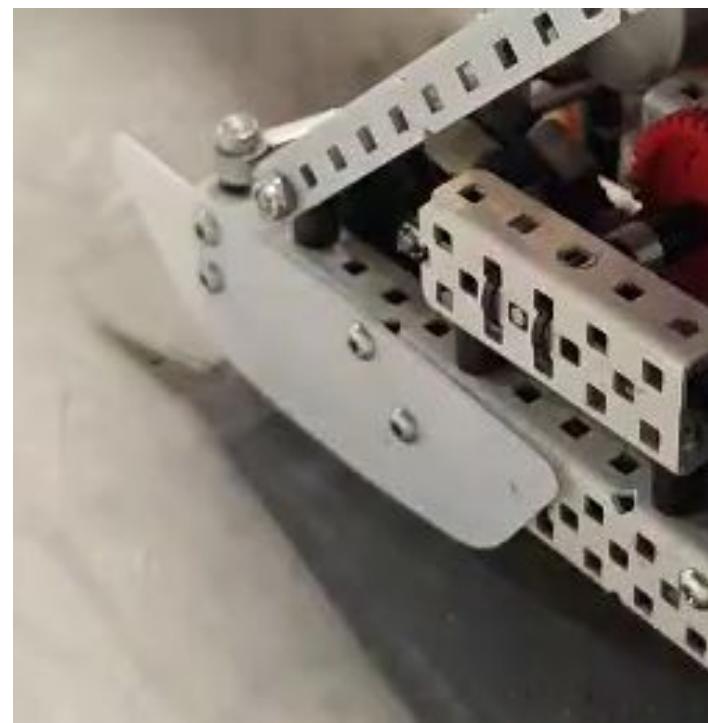


Figure 1: Picture of our new sled attached to the drivetrain

Next Steps

Plan out our matchloading mechanism

- **Problem:** How do we improve our matchloading mechanism?

Date: Sept 28, 2025

Previously in our notebook, we have established a design challenge when playing VRC Push Back and that is the match loading tubes. These issues were stated on page 117 in our first design cycle. When designing a matchloading mechanism, there are two main problems to consider.

1. How do we remove blocks from the loaders and into our robot?
2. Using previous knowledge and designs that we built, how do we improve our design?

Match Loader Inefficiency

We noticed that during MOA, our matchloader system was not fully optimized. This slowed down our ability to intake and score quickly. During matches, this added downtime directly impacting our scoring capabilities especially in high-pressure situations where quick decision making is crucial for coming out on top. For this design iteration, we need to identify parts of the matchloading mechanism that we can improve on through tuning and small adjustments.



Figure 1: Despite the matchloading mechanism in the tube, we were unable to remove blocks from the loader



Figure 2: The same issue occurring during our skills run at MOA

- ▶ Focus: Different options for matchloading

Date: Sept 28, 2025

We looked at some designs that were able to make signature event finals as these designs have proven to work consistently.

Bended Poly

The bended poly design uses either some form of heat or pure force in order to bend the polycarbonate to a specific shape. One notable example of this is 11101B. This piece of polycarbonate is slightly curved down so it is able to enter into the loader. It is followed by a section of polycarbonate that is bended down to act as a ramp so blocks can move towards the intake.



Figure 1: 11101B's matchloading mechanism (YouTube, FUN)

Overclock A (OCA)

Another design we looked at is a mechanism created by 16099A. This team triple crowned the Highlanders Summit Signature Event as well as reaching skills world record using this mechanism for matchloading. The design utilizes a piece of polycarbonate that is mounted on a screw joint. It is angled in a way where it is able to slide underneath the blocks in the loaders. Once the polycarbonate is in the tube, it uses gravity in order to push down on the poly piece, allowing the entire flat piece to become a ramp for blocks to move towards the intake.

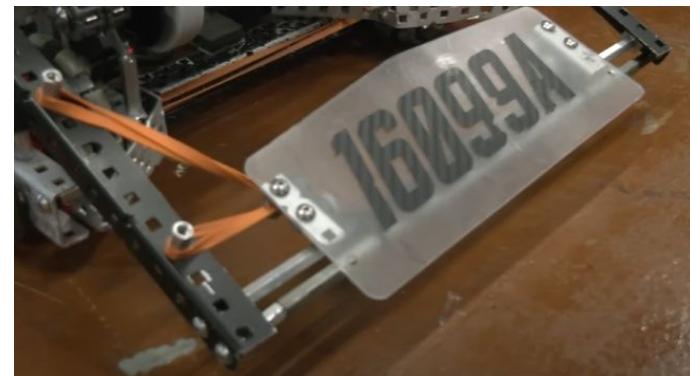


Figure 1: 16099A's matchloading mechanism (YouTube, FUN)

- ▶ Focus: Different options for matchloading

Date: Sept 28, 2025

Flat Poly

Another design that we found that was successful for other teams is a design used by both worlds skills #2 and #3 teams. This design is similar to the design by OCA as it also uses gravity in order for blocks to leave the loaders. This design is a flat piece of polycarbonate that is able to slide underneath the blocks in the loader. The blocks are then able to fall into the robot due to gravity and the piece of plastic creating a ramp shape to facilitate this. One downside of this design compared to OCA is that it expands outwards a lot more.



Figure 1: 81988E's matchloading mechanism (VCAD)

Ruiguan Poly

The final polycarbonate design we looked at was from MOA finalists 9123X. This design utilizes two pieces of polycarbonate mounted on high strength shaft collars. One piece is mounted at a very steep angle allowing for the mechanism to slide underneath the blocks in the loader. The second piece of poly is bent using heat to create a curve shape. This acts as a ramp to allow blocks to fall towards the intake.



Figure 1: 9123X's matchloading mechanism (YouTube, FUN)

► Focus: Selecting a matchloading design

Date: Sept 28, 2025

Following the research we have done on possible methods of matchloading, we will rank these designs using the following criteria.

1. **Build Simplicity (1 - 5)** - This ranks the design on how easy it is to build
2. **Weight (1 - 5)** - Weight added to the robot
3. **Ease of Use (1 - 5)** - The design should minimize errors that can happen during autonomous and drive period.
4. **Space Needed (1 - 5)** - Ranked based on how much space the mechanism needs for both mounting and how much it expands out. Ideally our mechanism should take minimal space.

<u>Criteria</u> <u>Options</u>	Build Simplicity	Weight	Ease of Use	Space Needed	Total /20
Bended Poly	3	3	3	4	13
Overclock	3	3	4	3	13
Flat Poly	3	3	2	2	10
Poly + Standoffs (Ruiguan)	1	3	4	4	12

- ▶ Focus: Our Plan for the match loader

Date: Sept 28, 2025

During this stage, we identified a **recurring issue** with our bot: it **required extremely precise alignment to successfully load balls**, often resulting in missed loads during competition. To address this, we planned to redesign the match loader with a **focus on increasing tolerance and ease of use**.

Our proposed solution was to construct a new match loader out of **polycarbonate**, chosen for its **balance of strength, flexibility, and ease of shaping**. By widening the loader's opening, we aimed to provide a **larger margin for error, reducing the need for near-perfect positioning** during loading. This would allow for **smoother and faster match cycles** under the **time constraints of competition**.

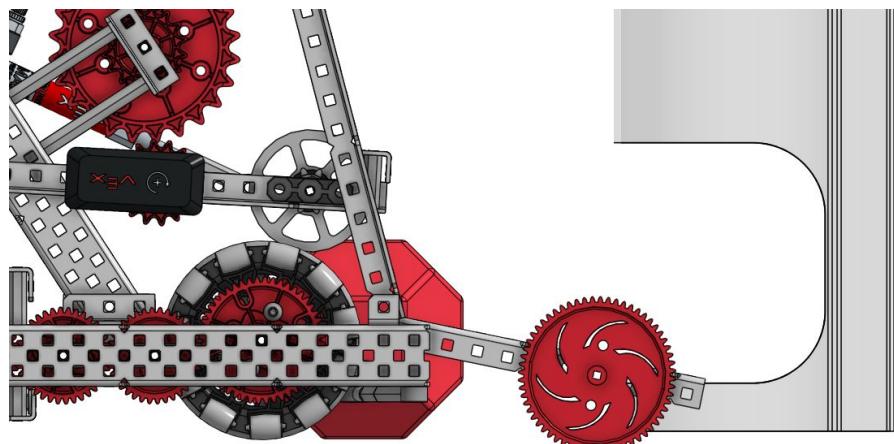


Figure 1: 2D CAD profile of our matchloader structure.

We also decided to integrate **bent polycarbonate slants** in place of the previous standoff design. The slanted surfaces were intended to **first guide the balls upward**, then allow them to **slide down smoothly** into the intake path. This change would not only **improve consistency** but also **minimize the chance of jamming or misalignment during loading**.

Overall, this **redesign aimed to create a more forgiving, reliable, and efficient match loading mechanism**, improving the robot's **performance under real match conditions and reducing operator error** during **high-pressure moments**.

► Focus: Building Our Matchloading Mechanism

Date: Sept 29. 2025

Members Involved

Brandon

Objective

We first created the structure for our matchloading mechanism that consists of angle bars and a high strength shaft. We then shaped the plastic by bending the piece into a shape that could both act as a ramp and slide underneath blocks in the loading tubes. Once the plastic was bent, we screwed it in to attach it.

Materials

- 2, 14 hole 1x1 angle bar
- 2, 8 hole 1x1 angle bar
- 1 high strength shaft
- 2, 32 teeth 6P sprockets
- Polycarbonate
- Screws
- Nylocks

Problems and Solutions

Bending the polycarbonate proved to be an issue because we do not have a heat gun to reshape the plastic

We used a combination of a vice and multiple clamps in order bend the polycarbonate. We then let the plastic stay for about 20 minutes so the shape would hold



Figure 1: A picture of our matchloading mechanism

► Focus: Design towards our robot for local events

Date: Sept 9, 2025

1

 X2 14 Hole 1x1 Angle Bar

 X2 8 Hole 1x1 Angle Bar

 X4 ¼" Spacers



X2 0.625 Screw

 X6 Nylocks

 X1 13" HS Axle



- ▶ Problem: Testing our matchloading mechanism

Date: Sept 30. 2025

Purpose

Testing our matchloading mechanism to see if it would be viable within matches and skills.

Procedure

1. Have one team member ready with a stopwatch
2. When ready, drive the robot with the matchloading mechanism into the matchloading tube
 - a. Start the stopwatch at this time
3. See how long it takes for 6 blocks to be removed from the matchload tube and into our intake system
 - a. Stop the stopwatch once the 6th block enters into our intake
4. Record the result under “time to matchload, t (s)” to **three significant digits**. Repeat steps **2-3** until enough trials are produced.

Table 1: Removing 6 blocks from the matchloader, t (s), in relation to trial number

Trial Number	Time remove 6 blocks, t (s)
1	N/A
2	5.82
3	N/A
4	5.09
5	4.38
6	5.71
7	N/A
8	5.48
9	N/A
10	5.97

Conclusion

Not including the trials where the blocks jammed, the **average time** to remove 6 balls from the matchloader was **5.41 (5.40833) seconds**. This is a **subpar** result, because it is still outperformed by the mechanism (C1.2) on our **V1** robot, which had an average of **2.05 seconds**. Given this time, we need to switch to a design that can remove 6 balls faster than it, otherwise we will keep using our old design.

Matchloader C2.0 Problems

Identify Problem

► Problem: Matchloading Mechanism Issues

After finishing testing our matchloading mechanism, we found a key issue which is that the plastic being inserted into the loader could prop the blocks up. This causes them to get stuck in between the top lip and the plastic piece. This means our mechanism is not performing to the best of its abilities and would be a liability during drive control and autonomous due to its inconsistencies.

To solve these issues we decided to switch to our other high ranking design which was the OCA design.

Date: Sept 30, 2025

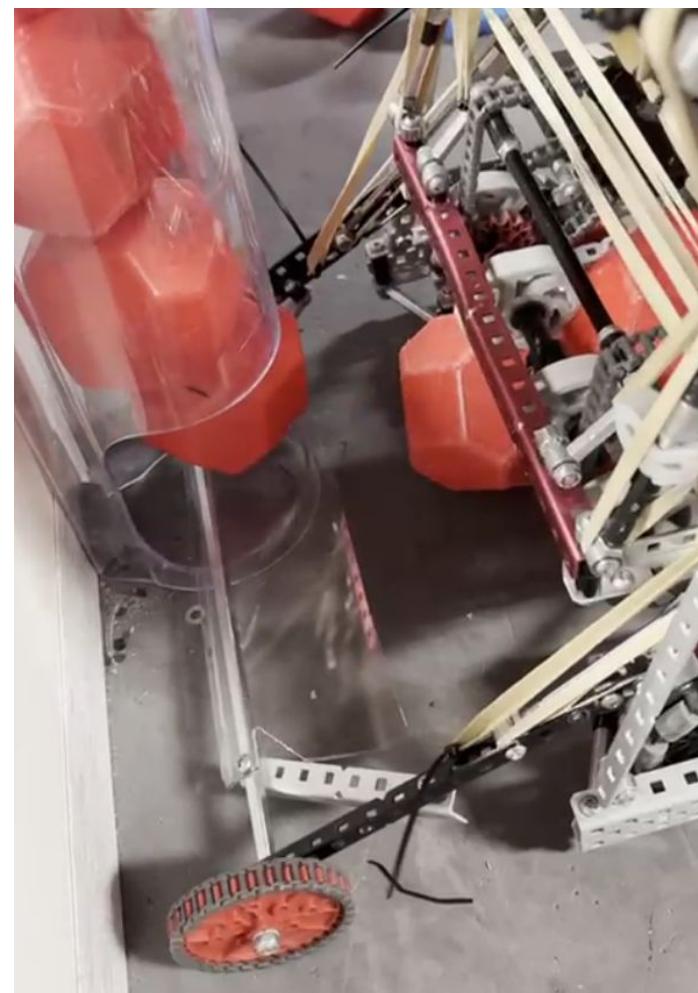


Figure 1: A block getting stuck between the lip and our plastic

- ▶ Focus: Rebuilding our matchloading mechanism

Date: Oct 3, 2025

Members Involved

Brandon

Objective

Build the Overclock matchloading mechanism by creating a plastic piece that can rotate on a screw joint.

Materials

- 2, 3 Hole 1x1 angle bars
- Standoffs
- Nylocks
- Screws
- Spacers
- Polycarbonate piece

Problems and Solutions

When initially testing our mechanism, we found that when there were less than 2 blocks in the loader tube, the force of gravity was not enough to overcome our banding.

To solve this, we changed the angle of banding and adjusted how tight the rubber bands were pulling.

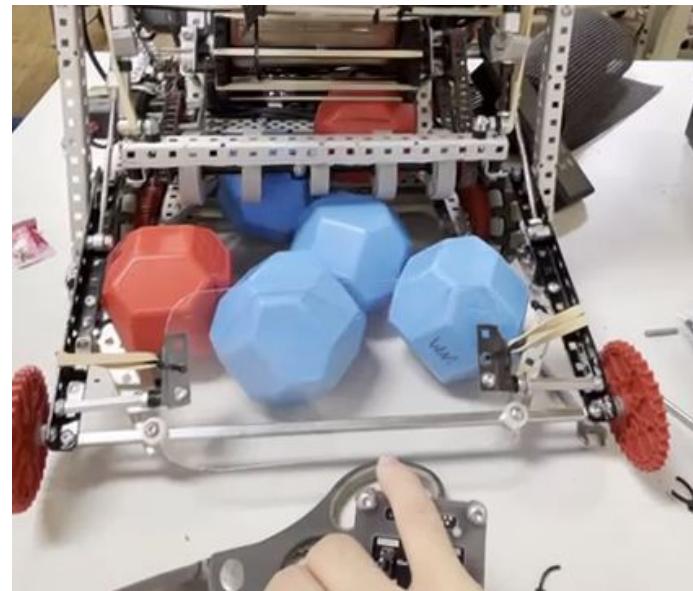


Figure 1: Front view of our new matchloading mechanism



Figure 1: A different angle showing our new matchloading mechanism

Next Steps

Test the new mechanism.

- ▶ Problem: Testing our matchloading mechanism

Date: Oct 3, 2025

Purpose

Testing our matchloading mechanism to see if it would be viable within matches and skills.

Procedure

1. Have one team member ready with a stopwatch
2. When ready, drive the robot with the matchloading mechanism into the matchloading tube
 - a. Start the stopwatch at this time
3. See how long it takes for 6 blocks to be removed from the matchload tube and into our intake system
 - a. Stop the stopwatch once the 6th block enters into our intake
4. Record the result under “time to matchload, t (s)” to **three significant digits**. Repeat steps **2-3** until enough trials are produced.

Table 1: Removing 6 blocks from the matchloader, t (s), in relation to trial number

Trial Number	Time remove 6 blocks, t (s)
1	0.88
2	1.47
3	2.31
4	0.95
5	1.80
6	0.72
7	2.04
8	0.84
9	1.19
10	2.41

Conclusion

The **average time** to remove 6 balls from the matchloader was **1.46 (1.461) seconds**. Given this time, this version of the matchloader (C2.1) is **very successful** because it not only outperforms our last matchloader (C2.0), but it also outperforms the best matchloader we tested for V1 (C1.2). Therefore, we will keep this design on our robot.

- ▶ **Problem:** Various materials used in our first stage intake have bent

Date: Oct 4, 2025

Cross brace

The cross brace attaching across our intake was bent when we checked on it. This brace was made using an angle bar and there is only one point of contact on each side.

We plan to swap out this brace with a halfcut 3-wide channel as it would provide more stability and be less likely to bend again.

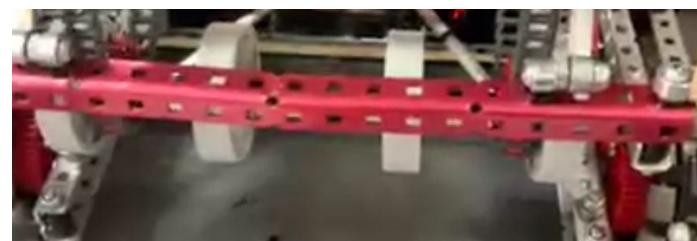


Figure 1: Bent front brace on our intake

Axle

When running our intake, we found that our light strength axle that contained all our rollers have bent as well after pressing down on blocks when double parking.

We plan to swap to a high strength shaft as it would be less likely to bend. This would provide us with the necessary structure in order to both intake and double park.

- ▶ Focus: Fixing our intake

Date: Oct 4, 2025

Members Involved

Brandon, Daniel, Bryan

Objective

We aimed to make our intake more stable through a variety of changes.

1. Changing our bent angle bar for a halfcut 3 wide channel for more stability
2. Replacing our light strength axle to a high strength axle to avoid bending.

Materials

- 1, 20 Hole halfcut 1x3 channel
- High strength shaft
- Pillow bearings
- Screws
- Nylocks

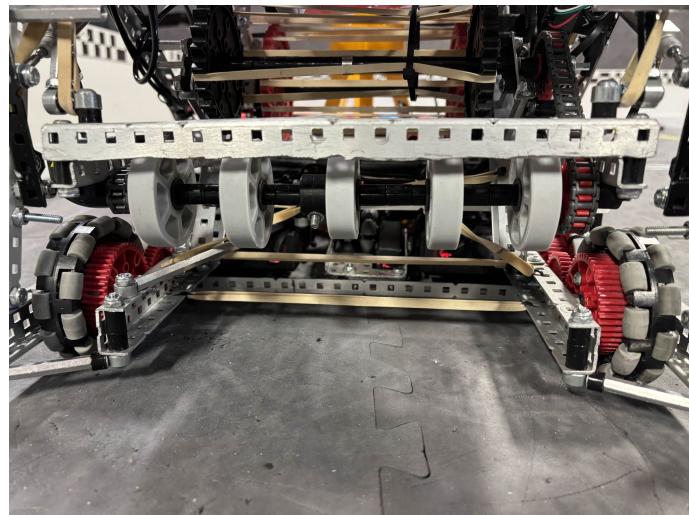


Figure 1: Rebuilt intake with high strength shaft and new brace.

- ▶ Focus: Documenting our minor changes to tune to tune funnelling.

Date: Oct 5, 2025

Following the plan we made on Sept 26, pg 254* (likely to change when we edit) pages

Members Involved

Daniel

Objective

Adjusting the basket in a few ways

1. Making the funnels very steep in order for blocks to funnel properly
2. Adding an flex wheel to act as an agitator
3. We also switched the front of our basket to a polycarbonate piece as it has less grip and allows blocks to slide down with the help of gravity

Materials

- Plastic Piece
- Cut 3 Inch Flexwheel
- Mesh
- Zipties
- Rubber bands

Next Steps

Testing our intake to ensure it will not encounter any issues during driver control and autonomous routines.



Figure 1: Implemented hood adjustments

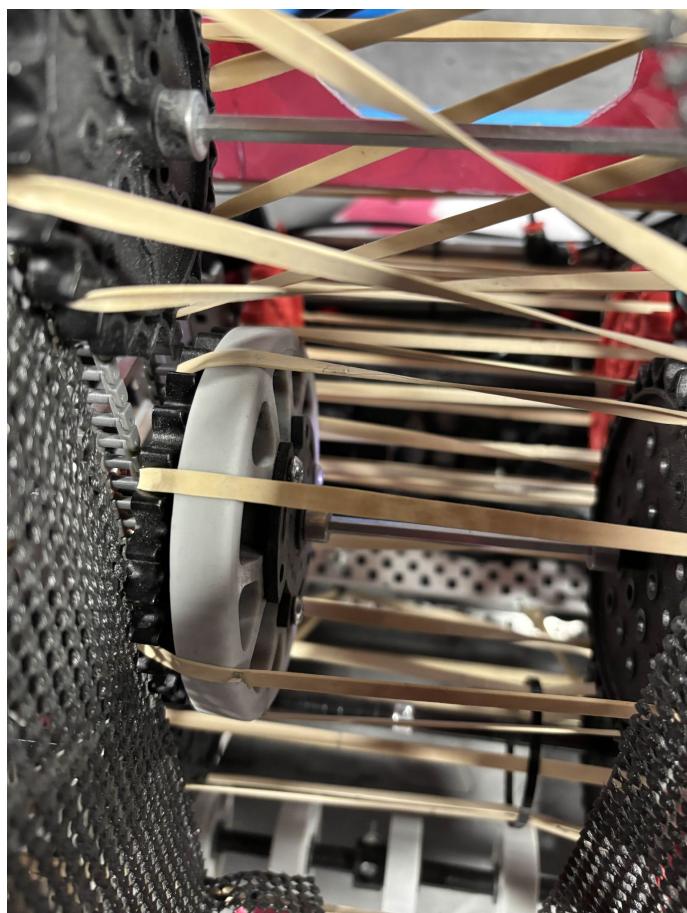


Figure 2: Picture of our agitator

- ▶ Problem: Testing our intake system

Date: Oct 5, 2025

Purpose

The purpose of the following tests is to find any issues with our intake system.

Procedure

1. Have a team member ready with a stopwatch
2. When ready, drive the robot with the intake running into two blocks that are positioned next to each other
3. When both blocks have made it past the third set of rollers, stop the timer and take the time down in our table
4. Repeat steps 1-3 for until there is sufficient data

Table 1: Intaking two blocks placed next to each other, t (s), in relation to trial number

Trial Number	Time to intake, t (s)
1	N/A
2	N/A
3	1.65
4	N/A
5	1.89
6	2.73
7	N/A
8	2.41
9	N/A
10	2.66

- ▶ **Problem:** Our intake gets stuck when intaking two at the same time

Date: Oct 5, 2025

After our testing, we noticed that our intake had a 50% chance of jamming and getting stuck when intaking two blocks that were placed next to each other. This would cause many issues during autonomous routes as it would disable our entire intake system once our intake gets jammed. This is a major issue for the robot in general as it would impact all competition matches and skills.

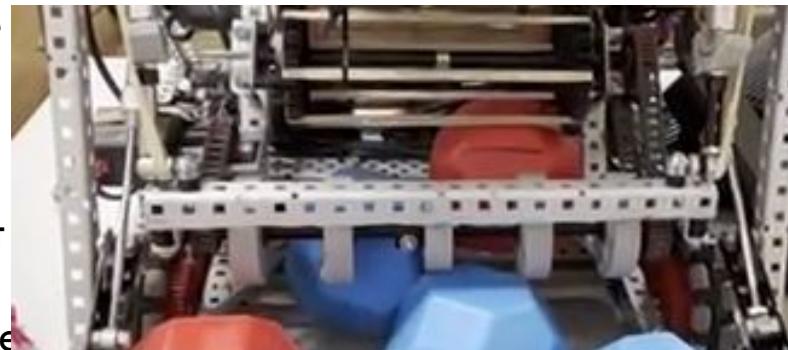


Figure 1: Blocks getting stuck in the intake.

We found a few main causes of this issue

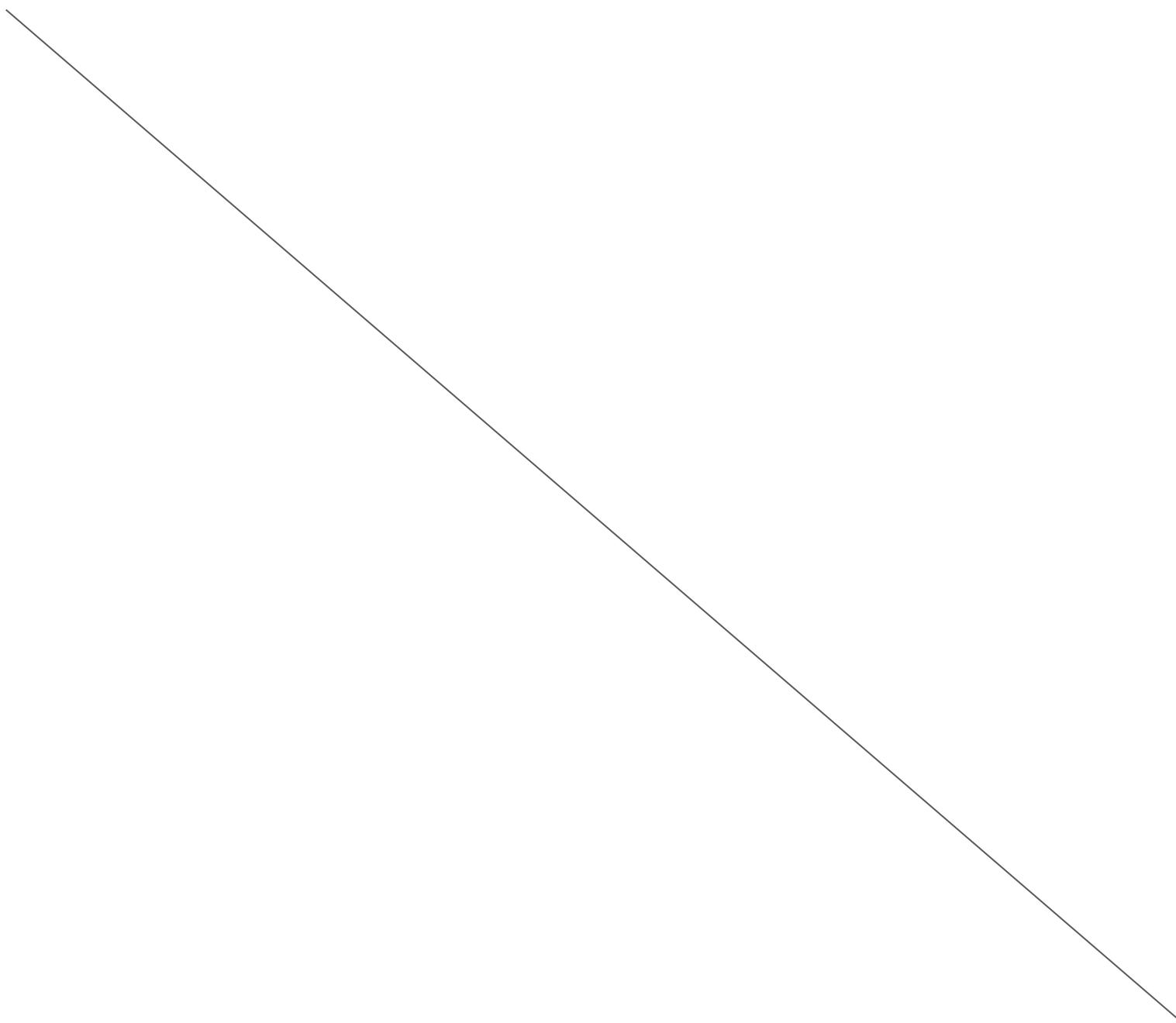
1. Our sprockets are too narrow compared to our flex wheel rollers, so when we intake blocks, the blocks near the edge of the intake would get stuck underneath the sprockets.
2. Our intake system lacked funnels to help direct the blocks towards a certain point. Once a block entered our intake, it would roll around and there was nothing to help funnel the block towards the center of our intake

- ▶ Focus: Attempt to fix the jamming issue

Date: Oct 5, 2025

To fix the jamming issue, we will bias our funnels, such that one side will move upwards faster than the other side. This can both be done with standoffs and zipties, or rubber bands.

The funnels can also be biased by making one side steeper than the other, so that the blocks funnel more single-file instead of side-by-side. We will attempt to use both of these strategies to eliminate jamming.



- ▶ Focus: Building funnels to avoid jamming

Date: Oct 5, 2025

Members Involved

Bryan

Objective

Create funnels that help direct the block towards the center of the intake. We create funnels using rubber bands. We had a couple of iterations of these funnels, but they were mostly the same.



Materials

- Rubber bands
- Zip-ties
- Standoffs
- Axle collars
- Screws
- Nylocks

Figure 1: Our funnels from a front-facing view

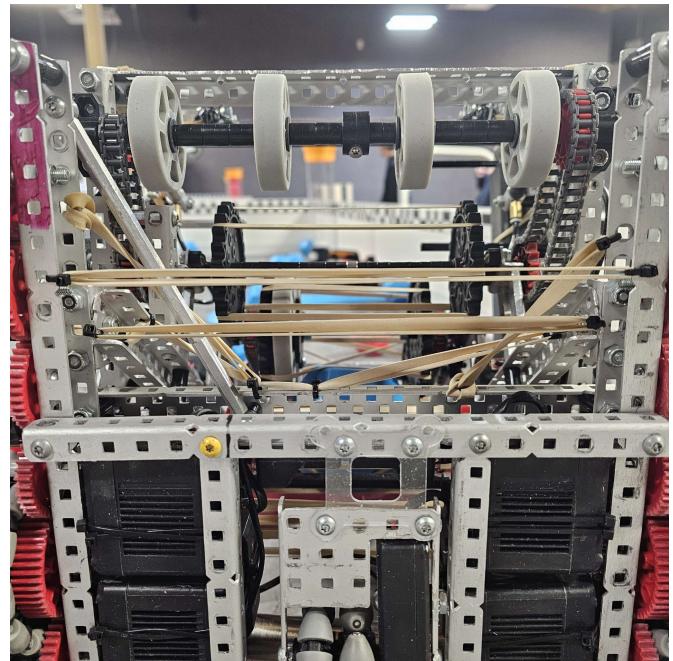


Figure 2: Our funnels from a bottom-facing view

October 9 Game Manual Update

Identify Problem

► **Problem:** Revise the implications to our hood design

Date: Oct 10, 2025

The definition of rule <GG9> was changed to disallow reaching into the volume of a goal with no intention to move balls.

<GG9> **Don't clamp your Robot to the Field.** Robots may not intentionally grasp, grapple, hook, attach to or otherwise Entangle with any Field Elements. Strategies with mechanisms that react against multiple sides of a Field Element in an effort to latch or clamp onto said Field Element are prohibited. The intent of this rule is to prevent Teams from unintentionally damaging the Field and/or from anchoring to or otherwise Entangling themselves with the Field.

Whenever possible, Head Referees should alert Teams to potential Violations before they happen to prevent actual Violations. If a Robot takes immediate action to avoid or resolve the issue, and if the Head Referee determines that the issue had no effect on the Match, no Violations should be recorded.

Violation Notes:

- Major Violations of this rule should be rare, as Robots should never be designed to intentionally violate it.
- Reaching into any open portion of a Goal to move Blocks isn't considered anchoring or a Violation of this rule. During the Driver Controlled Period of any Head-to-Head Match, keeping your mechanism there while the Robot isn't actively moving Blocks inside the Goals is considered anchoring, and is a Violation of this rule. This guidance does not apply during Autonomous Periods or Robot Skills Matches.

This means that “defending” a long goal by preventing balls from being scored through having a part of the robot inside the long goal is illegal. Our interpretation is reinforced by Q&A 2837 and Q&A 2826:

If part of a Robot is inside of any part of a Goal, it is considered to be reaching into that Goal.

If it is considered reaching into the goal, would this violate SG10 and Q&A 2791 if the robot is trying to outtake a block into the goal, but is not successful, such that the block is rolling in the robot? In other words, if the robot is moving a block but the block is not in the goal? What if the robot has stopped moving and is not trying to score a block?

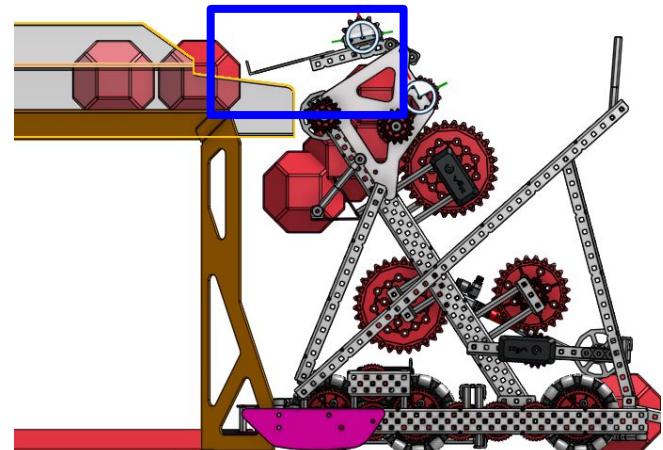
My question is, what is considered “actively moving”? Is an attempt at moving Blocks enough to be considered actively moving Blocks, or should there be visible movement of the Blocks?

For the purposes of this rule, actively moving Blocks inside the Goal in an effort to change the score of the Match (i.e., adding or removing Blocks from a Goal or a Control Zone) should qualify as movement. Holding the position of Blocks that are already in a Goal or Control Zone should not.

The lexan hood on our current robot (see highlighted) reaches into the long goal both to descore balls out the other end and to prevent opponents from scoring on the other end. With the rule clarification, our hood has to be redesigned to fit the new constraints.

Sources:

<https://www.robotevents.com/V5RC/2025-2026/QA/2837>
<https://www.robotevents.com/V5RC/2025-2026/QA/2826>
https://www.robotevents.com/storage/game_manual/VEX_V5_Robotics_Competition_2025-2026_Push_Back/rules/GG9.html



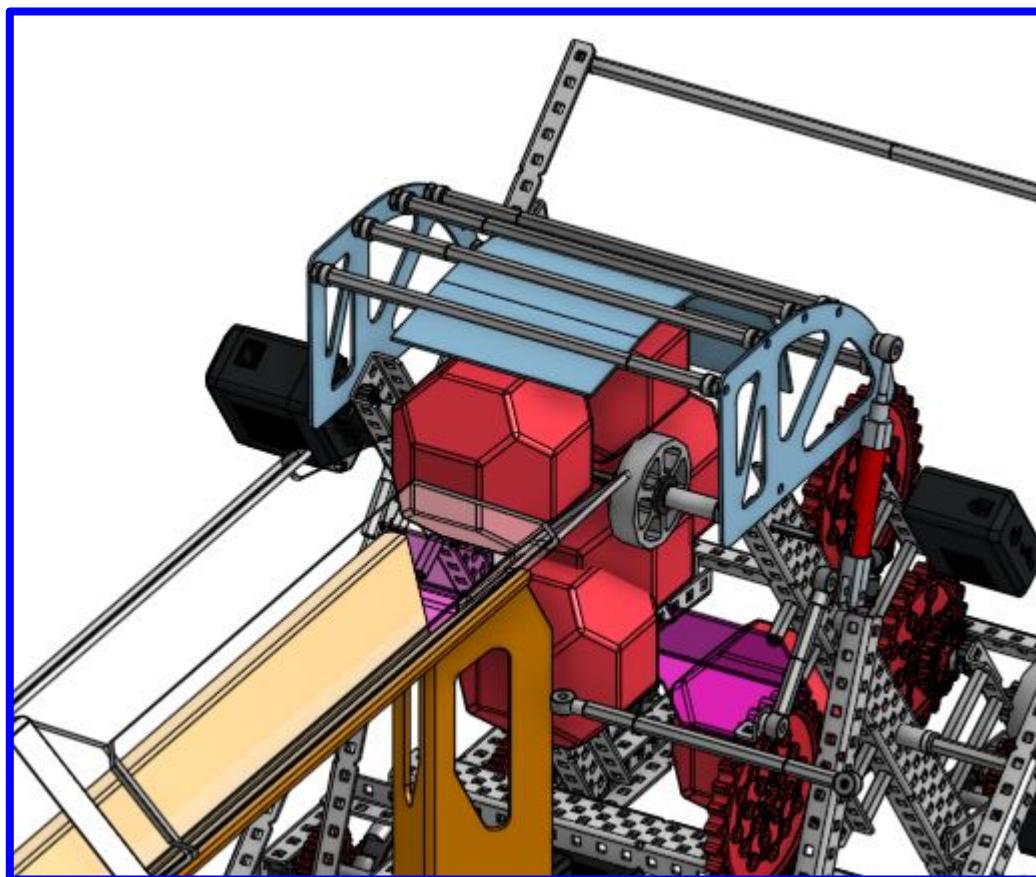
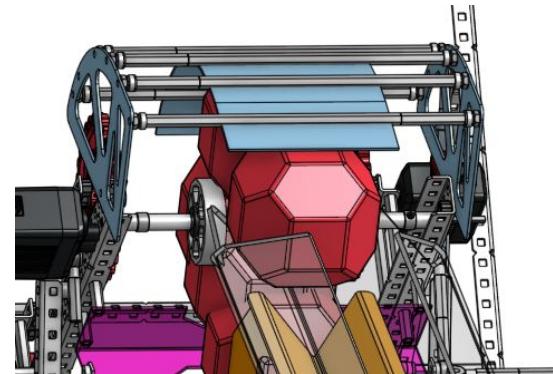
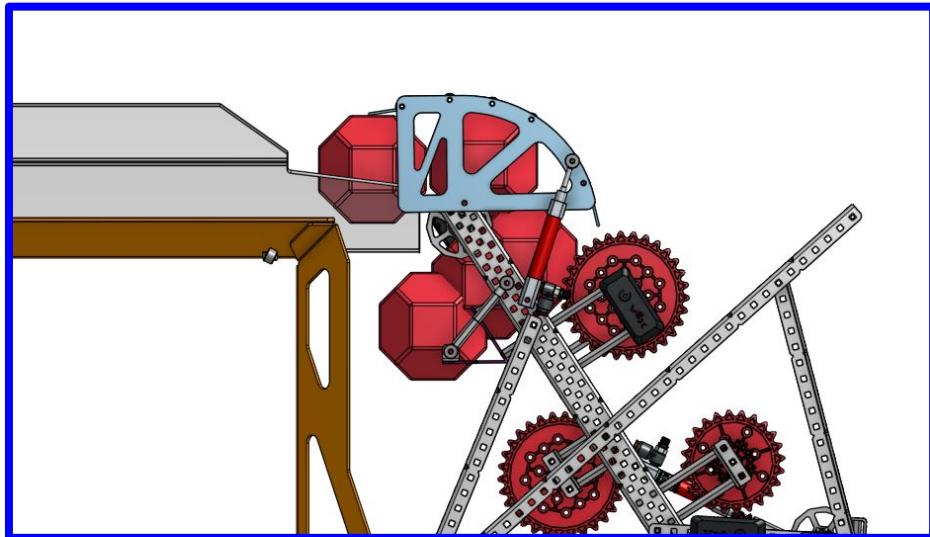
Hood Polycarbonate Brainstorming

Brainstorming

- Focus: Revise the implications to our hood design

Date: Oct 10, 2025

To address the new robot build ruling, we would like to consider switching our hood mechanism to an alternate design in the future, after the completion of our scoring mech C3.



- ▶ **Problem:** Driving through the park zone.

Date: Oct 11, 2025

When trying to drive through the entire park zone. We noticed that we would get stuck. There were a few reasons for this

1. Because our wheels were offset on the top hole of the C-channel, we would have very little grip with the ground
2. Due to our wheel orientation, there was a large section of gears that could get stuck on the park barrier
3. Our low bracing would cause us to ride on top of the metal, stopping the wheels from contacting the ground.

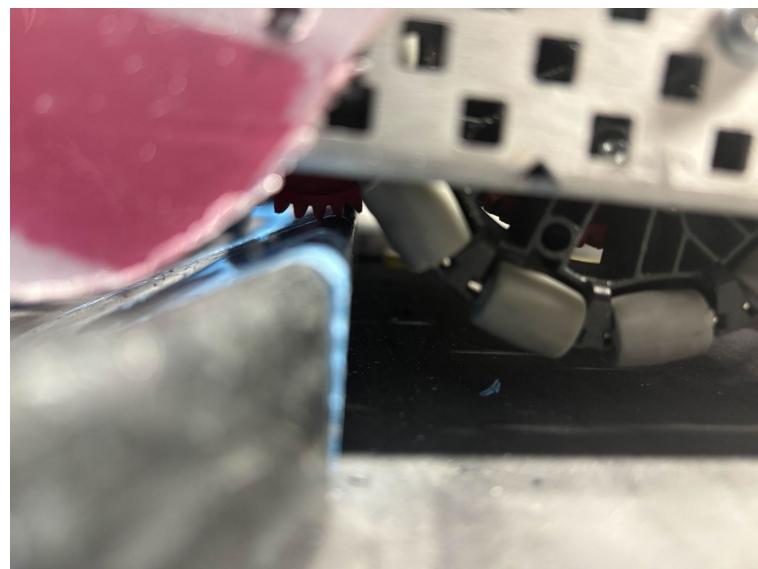


Figure 1: Our drivetrain gears contacting the park barrier

- ▶ Focus: Adding a small support wheel for barrier crossing

Date: Oct 11, 2025

Members Involved

Daniel

Objective

Add a smaller wheel that can assist in crossing over the park barrier. To do this, we needed to raise our brace up by $\frac{1}{4}$ of an inch to accommodate for space needed for our wheel.

Materials

- 2, 48 teeth high strength gears
- Flexwheel cutout
- Zip ties
- Screws
- Spacers
- Kepsnuts
- Nylocks

Next Steps

Making finishing touches on our robot while we start to tune autonomous routines.

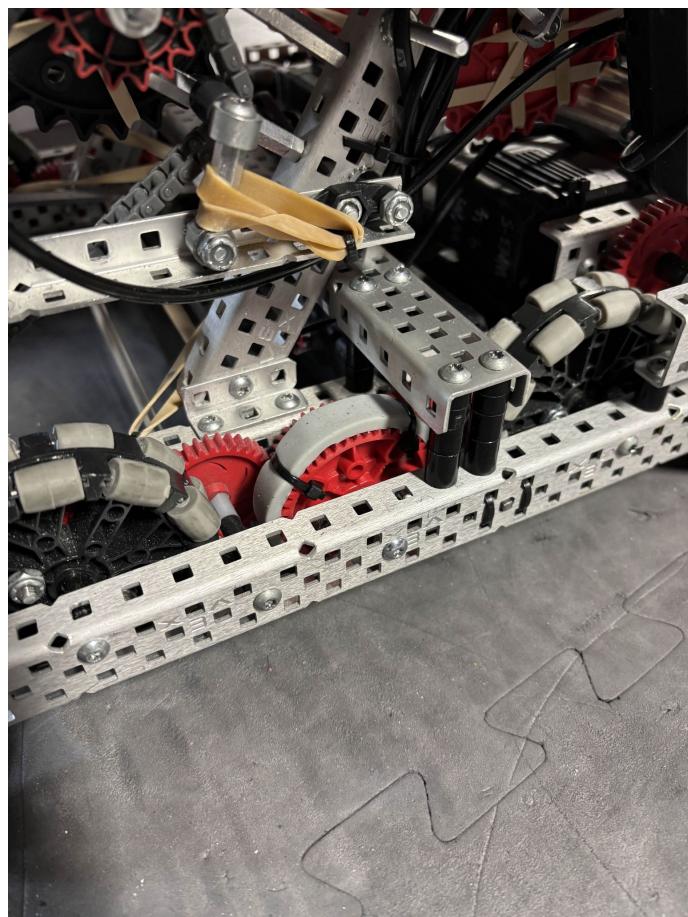


Figure 1: Gear wrapped with flexwheel to assist in our barrier cross

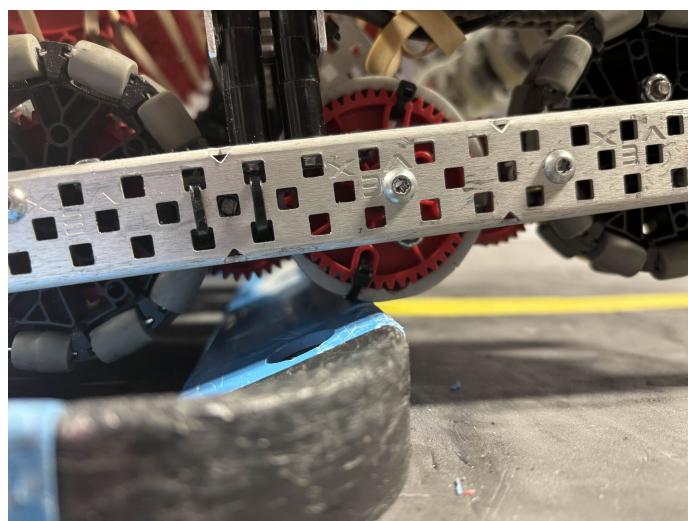


Figure 2: The flexwheel helping us with crossing the park barrier

- ▶ Focus: General tuning of the robot

Date: Oct 14, 2025

Members Involved

Daniel, Bryan, Brandon

Objective

We made small adjustments to our basket and funnels. We tightened the mesh on our basket so the funneling would be improved. We added various bands and standoffs to prevent blocks from getting stuck underneath our intake sprockets.

To ensure our mesh does not stretch out again, we used flat metal to create a rigid shape that prevents the blocks from pushing against the mesh.

Materials

- Rubber bands
- Standoffs
- Zipties
- Flat metal
- Screws

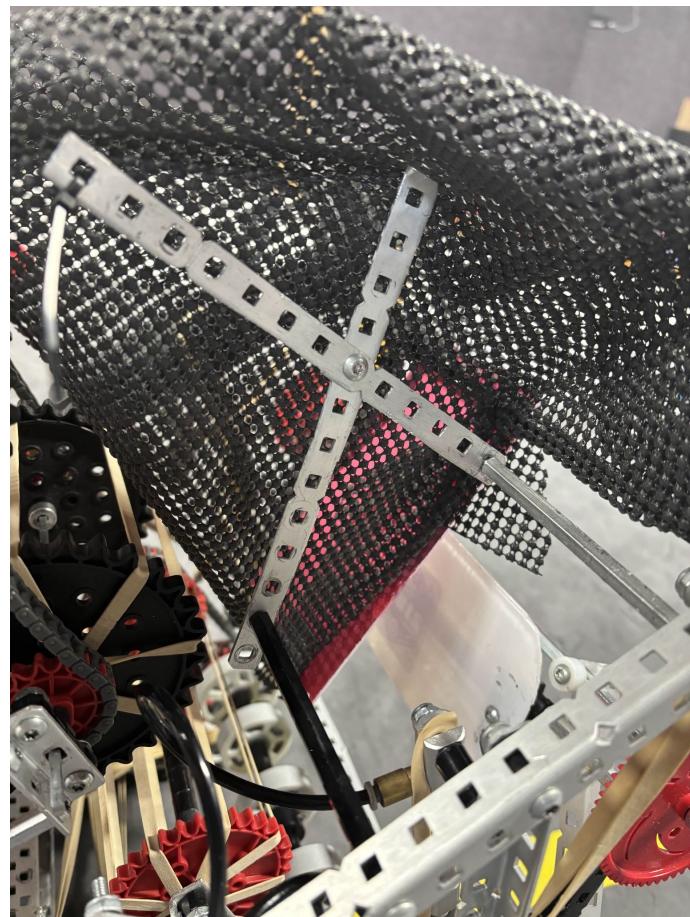


Figure 1: Flat metal support to prevent the mesh from expanding out

- ▶ Focus: Solo Auto Winpoint Struggles

Date: Oct 15, 2025

Autonomous Win Points (AWPs) play a critical role in overall tournament performance, especially during qualification matches where securing early ranking points can significantly improve standings. Achieving many AWPs requires the robot to meet a specific set of autonomous objectives consistently and efficiently.

Today, our goal was to develop and tune a **Solo AWP (SAWP)** routine, which demands handling a higher number of game elements compared to the standard AWP sequence used in local competitions. This version involves more precise movements, increased cycle speed, and tighter synchronization between intake, indexing, and shooting mechanisms.

- Today, we worked extensively on fine-tuning the pathing and timing of this routine
- However, despite the significant effort invested, the results were **inconsistent and unreliable**
- Often missed critical alignment points or failed to complete the routine within the allotted time. These inconsistencies made the SAWP path unsuitable for use in an actual match at this stage.

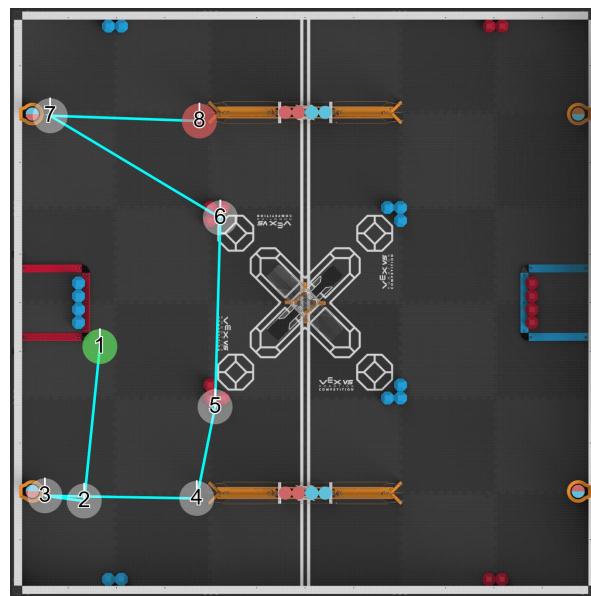


Figure 1: Target SAWP path for a 13 ball SAWP

Given our **time constraints** leading up to the next competition, we concluded that it would be more effective to **pivot toward a simplified autonomous path**. This adjusted version would prioritize reliability and repeatability over complexity, allowing us to secure at least a standard AWP consistently.

- Focus: Outline using a diagram.

Date: Oct 16, 2025

Given that there are only **48 hours remaining before the upcoming competition**, our focus will now shift toward developing and refining **Skills autonomous routines and elimination autonomous paths**. Time management will be critical, and we will need to allocate testing hours efficiently to ensure all autonomous programs function smoothly. Despite the tight schedule, our team remains confident in our progress and prepared to push through this **final time crunch** before the event.

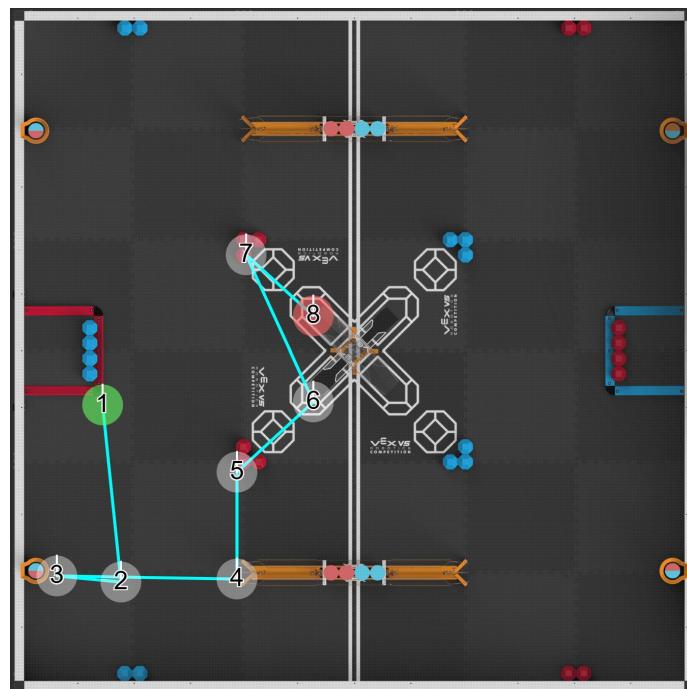


Figure 1: Local Event 11 Ball Eliminations path

► Focus: Layout our performance goals and potential issues.

Date: Oct 17, 2025

This is the first tournament where we are using our V2 robot's basket "hoard" design. A significant change in strategy is required, as we have increased storage capacity blocks compared to our V1 robot. We also expect that issues with our intake may arise due to the complex nature of our roller layouts to facilitate the storage compartment.

Skills Goals

Driver Attempts	Driver Highscore	Programming Attempts	Programming Highscore	Total Score
3	80	3	40	120

At the MOA Signature Event, we scored 69 pts in driver skills and 20 pts in auton skills. We think that a 20-point increase in both categories is possible given:

- 1) We've had more experience with playing the game in scrimmages
- 2) Our robot is designed to score well in skills due to its larger storage capacity
- 3) A improved matchloader compared to our V1 matchloader, which was identified as a pain point at MOA.

Overall Goals

- 1) Rank in the top 5 teams during qualifications
 - a) We believe that our "hoard" design can outscore other meta designs due to our larger storage capacity
 - b) Our aim is to have a consistently tuned local solo-AWP routine that will help us rank higher by WPs
- 2) Make it to finals in the tournament during eliminations
 - a) We also aim to develop eliminations-exclusive autonomous routines that will help us take the control zone of a single long goal during autonomous, which will increase our chances of winning the match
- 3) Rank in the top 5 teams for overall skills
 - a) We believe our skills goal is achievable given the current state of our robot
- 4) Finish the tournament with a trophy

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	5	2088S <u>210K</u>	109	1554P 3388A	44	Yes	No

Notes

- We focused on scoring while 2088S focused on defense, mainly on 1554P
- A major strategy that helped us to control the long goals was making 2088S push our opponent off of the goal, letting us (de)score freely.
- Middle goal control, autonomous and our double park made us win this match

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	8	210C 210Z	29	2088A <u>210K</u>	85	Yes	No

Notes

- Slower scoring match due to heavy defense from both alliances
- 2088A tipped at 0:58, forcing us to play more defensively and removing our ability to double park
- We won because of middle goal and long goal control, as well as autonomous

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	17	<u>210K</u> 45519X	88	210R 210E	35	Yes	No

Notes

- We played offensively while 45519X played defensively
- Unfortunately no livestream available for this match

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?
Win	19	2088K 2088H	25 210K 2885B	48	Yes No

Notes

- Our alliance had no intake, forcing us to play offensively while they played defense
- We played for long goal control
- Our opponents neglected the middle goals, which left them open for us

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?
Win	24	3388C 3388S	24 210K 3388H	80	No No

Notes

- We had a fatal jam in our intake during autonomous, causing us to completely miss scoring on the mid goal(s) after scoring 3 blocks in the long goal
- During the match, both opponent teams focused on defending us instead of scoring, letting our teammate score undefended to win the match after our autonomous deficit

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?
Win	33	210K 45519E	40 2088U 3388A	28	Yes No

Notes

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	QF 1	<u>210K</u> <u>210E</u>	89	2088H 2088K	23	Yes	N/A

Notes

- Our autonomous pathing was flawless, however our hoard setup caused us to only score 5/9 blocks in our inventory
- We focused heavily on using our hood to push red blocks into the long goal control zones and score opponent blocks from the other end

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	SF 1	<u>210K</u> <u>210E</u>	119	3388A 3388S	21	Yes	N/A

Notes

- We scored 6/9 blocks in autonomous as one was stuck in our hoard
- We defended our opponent robot while calling our alliance robot from their long goal defense to descore our long goal

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	F1	<u>210K</u> <u>210E</u>	99	210Z 1554P	33	Yes	N/A

Notes

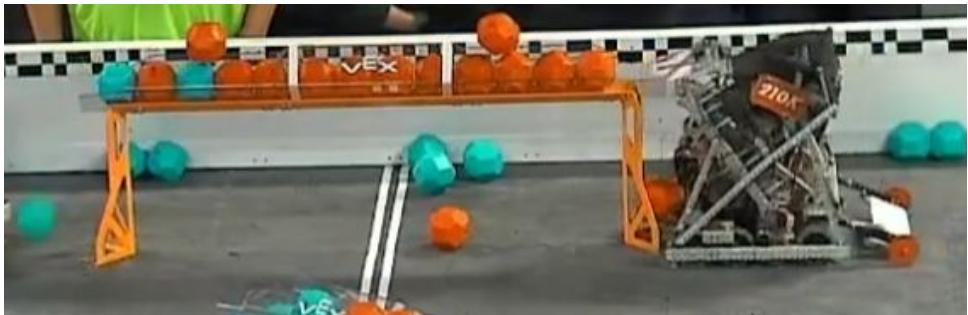
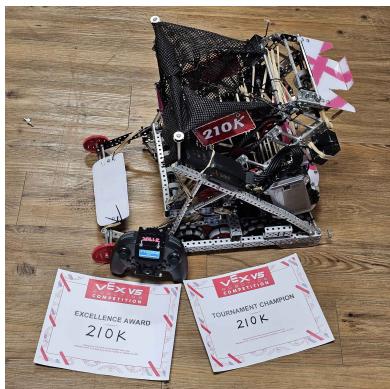
- Both teams failed double park,

Analyzing WM Season Opener

Tournament Summary

► Focus: Looking at statistics from our tournament.

Date: Oct 18, 2025



Skills

Rank	Driver Attempts	Driver Highscore	Programming Attempts	Programming Highscore	Total Score
4	1	54	3	18	72

- Lack of practice for driver runs
- Auto was not consistent or tuned

Qualifications

Rank	Team	Name	W-L-T	WPs . APs . SPs
1	210K	Kawaii Kittens	6-0-0	12, 40, 185

Though we won every qualifications match, we failed to score any AWPs due to our inconsistent intake jamming issues during autonomous.

Summary

Overall, the tournament was very successful; we did lose any matches, and also won Excellence. We did end up achieving all of our goals from pg. 277. However, we did not achieve any AWPs, nor a satisfactory skills score. Moving on, we will focus on leaving more time to tune the robot, as many of our lost AWPs stemmed from build issues. Strategy-wise:

- Many teams left the middle goals open, which gave us the chance to secure control
- Double park is important because it is a 30 point swing, and will become almost mandatory at signature events; we predict that many teams will be able to double park

V2.1: UC Berkeley Signature Event

Issues with our robot

Identify Problem

- ▶ **Problem:** Reflect on multiple issues affecting our scoring.

Date: Oct 19, 2025

There were some prevalent issues we noticed during our season opener tournament

1. Intake jamming

- a. Our intake jamming was a key issue during autonomous routines. This issue caused us to not score a single SAWP during qualification. The intake jamming also meant our eliminations autonomous did not perform to our expectations.

2. Intake is too slow

- a. Another issue that we found was our scoring was very slow. The top roller was powered by a single 5.5W 200RPM motor. This meant that we had to wait 1-2 seconds before balls would start entering the tubes. This delay caused us to get pushed off the goals before we could start scoring

3. Middle Goal Scoring

- a. We noticed that when we wanted to score on the upper middle goal. We often had to outtake blocks first before we could start scoring. This caused delays when we tried to score middle goals.



Figure 1: Our robot getting pushed off from the long goal. Unable to score due to the slow intake



Figure 2: Robot struggling to score them middle goal

- ▶ **Focus:** Establish plans to rebuild for the One World signature event.

Date: Oct 19, 2025

Heading into the **One World @ UC Berkeley** competition, our team decided to shift our focus from skill-based performance to a **more competition-oriented** approach. To align with this goal, we made the decision to replace our **hoarding mechanism** with a **double intake system**. This new design significantly **increases** our **robot's storage capacity** while **minimizing** the need for **major structural changes**. As a result, we can collect and **cycle game elements** more **efficiently**, giving us a stronger competitive edge during matches.

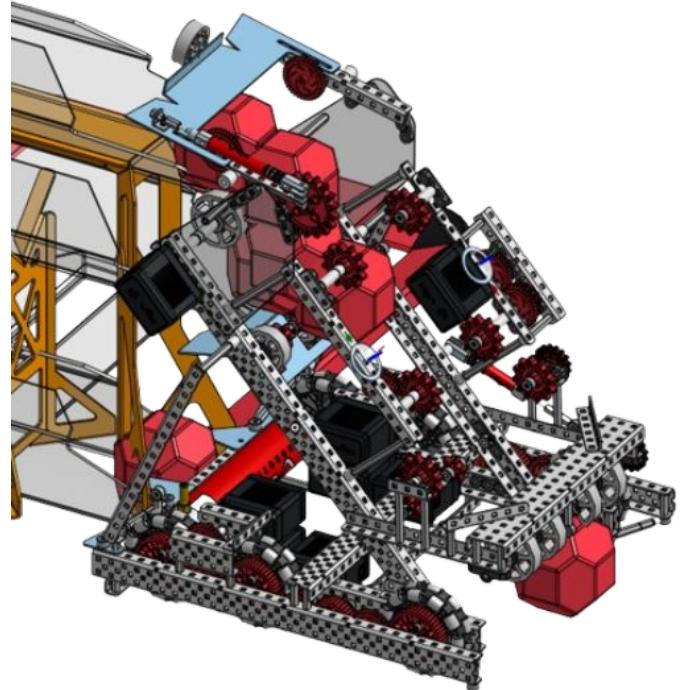


Figure 1: General CAD plan for our new robot

- Focus: Switching our intake type to a conveyor.

Date: Oct 20, 2025

Members Involved

Brandon

Objective

Our objective was to start creating structure for our conveyor stage of the intake. After creating necessary structure, we would then add axles for various rollers and conveyor stages.

Materials

- 2, 15 Hole 1x1 Angle Bars
- 4, 3 Hole 1x1 Angle Bars
- Bearing Flats
- Standoffs
- 2, 36T gears
- 16T 6P Sprocket
- 8T 6P Sprocket
- 6P Chain
- Screws
- Nylock Nuts

Next Steps

Continue building our intake system

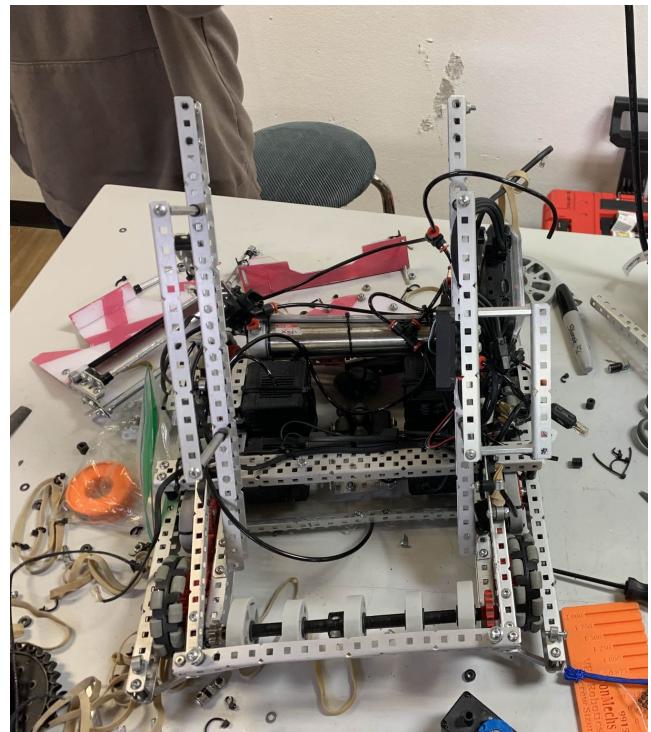


Figure 1: Structure for the conveyor

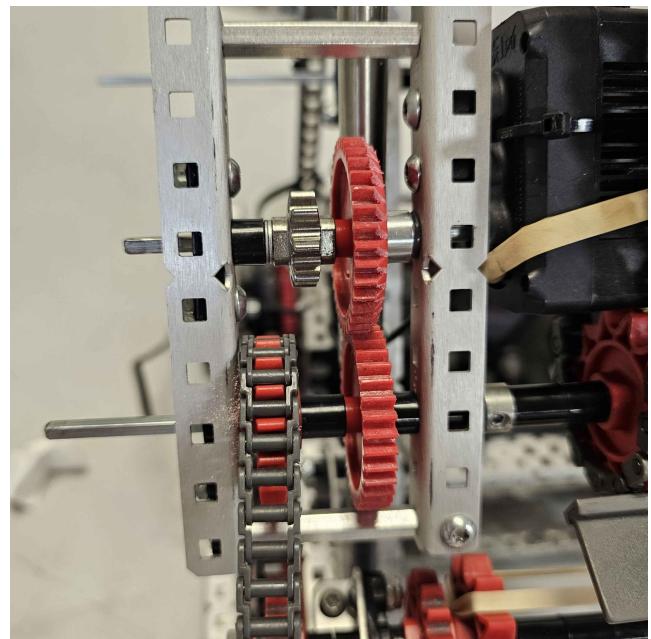


Figure 2: Gearbox to power different rollers

- ▶ Focus: Switching our intake type to conveyer

Date: Oct 21, 2025

Members Involved

Daniel, Bryan

Objective

We continued building our intake system by adding rollers and conveyor. This included a small stage of rubber bands and a flex wheel counter roller to help with middle goal scoring. We also added a middle goal aligner and descore polycarbonate piece. This would help us align with the upper middle goal and the point would allow us to easily descore the goal.

Materials

- 8, 12T 9P Sprockets
- 2, 16T 6P Sprockets
- 9P Chain Links with flaps
- Spacers
- Shaft Collars

Problems and Solutions

We noticed that the rubber band roller was touching our flaps when they spun around. This moved the bands off the sprocket, making the roller useless

To solve this, we plan to replace the rubber bands with 1.65" flex wheels. They are the same diameter as the 12T sprockets and would serve the same purpose without contact

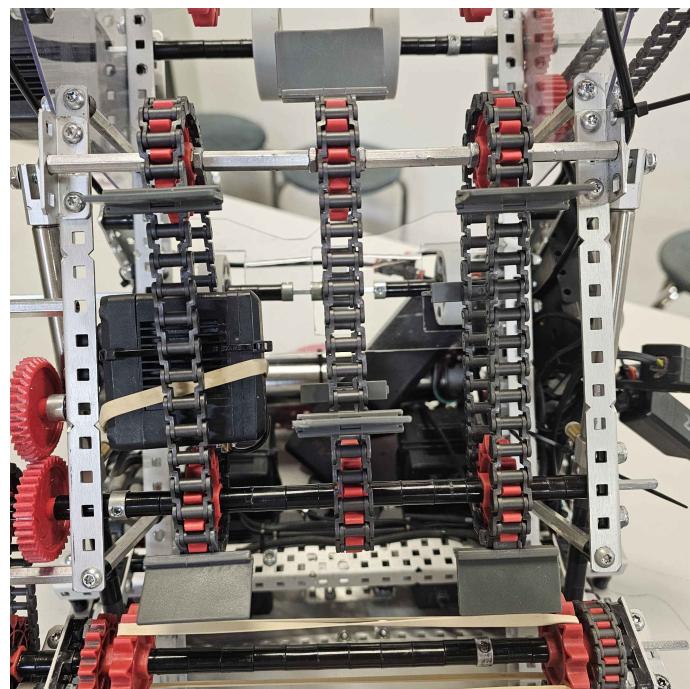


Figure 1: Finished Conveyer Stage

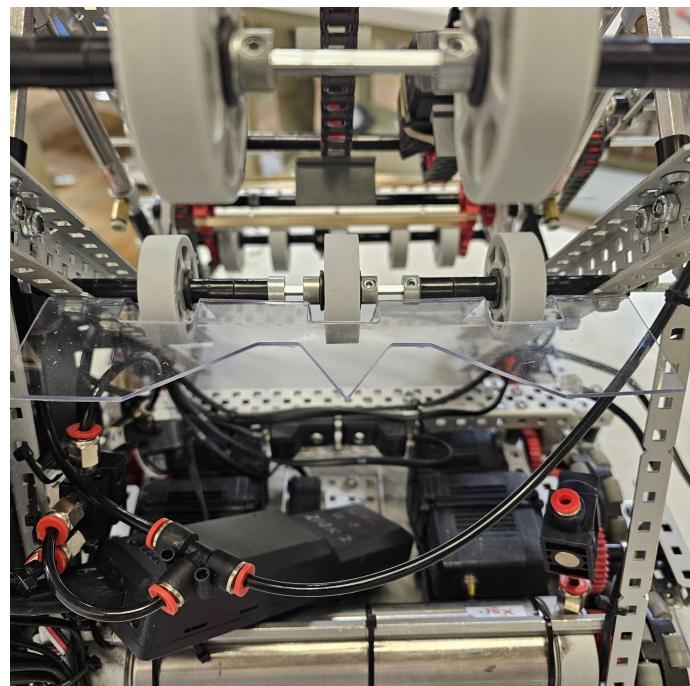


Figure 2: Middle Goal Counter Roller

- Focus: Tensioning our chains and creating double park

Date: Oct 21, 2025

Members Involved

Bryan

Objective

We attached pistons onto our intake to create a double park mechanism. We used long pistons nestled closely to the structure our robot to allow the intake to press down onto a block. We also used banded standoffs to create tensioners. This will prevent our chains from skipping.

Materials

- Standoffs
- Shaft Collars
- Rubber bands
- Screws
- Nylocks

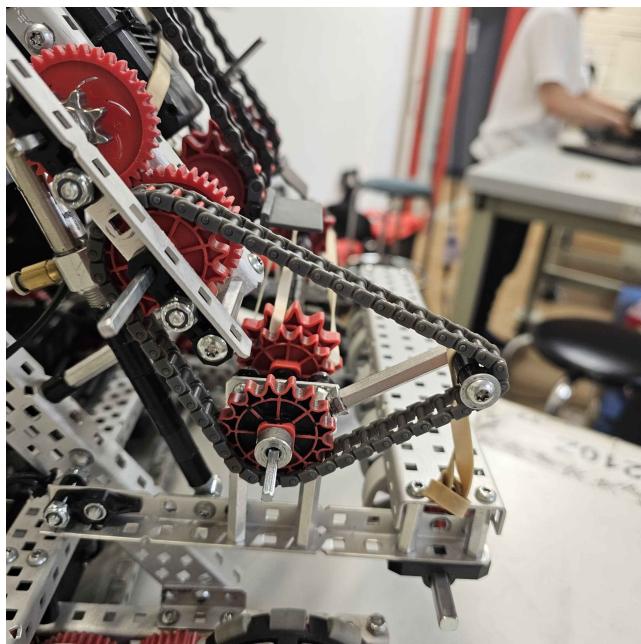


Figure 2: Tensioner from our conveyer to the rubber band stage

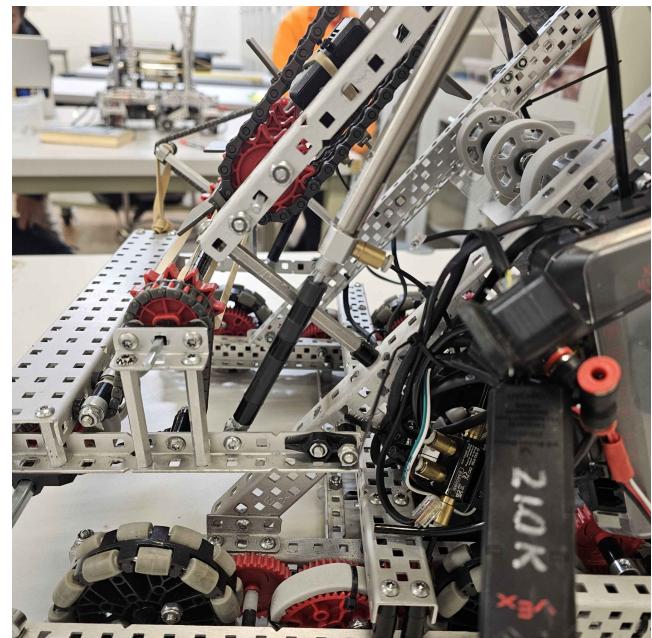


Figure 1: Attached Piston for Double Park

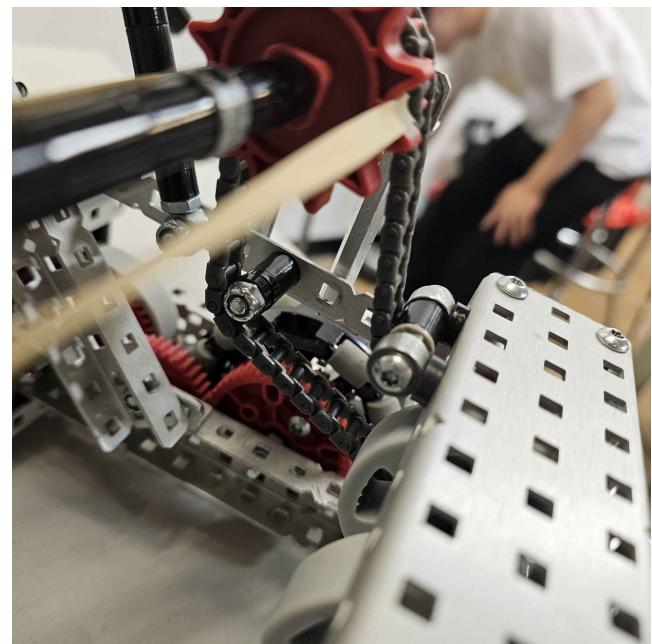


Figure 3: Tensioner from rubber band to first stage

- Focus: Tuning our intake system

Date: Oct 22, 2025

Members Involved

Daniel, Bryan

Objective

Our goal for today was to finish our robot so it could be in a working condition for testing. This includes creating a ramp and chaining all our rollers together so they can be powered

Materials

- 6P Chain
- Rubber Bands
- Zip ties

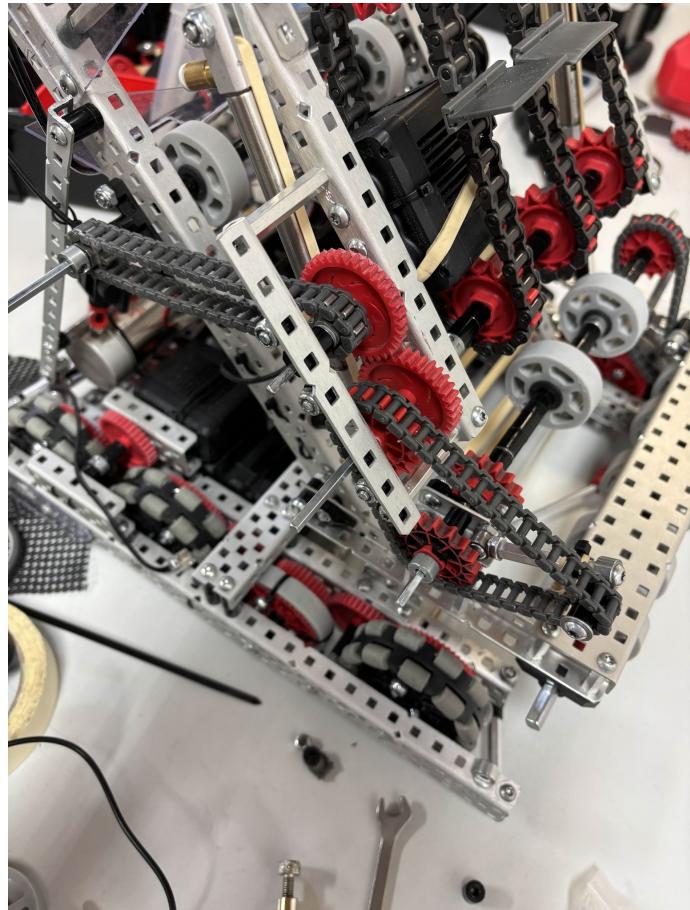


Figure 1: Chains to power our rollers

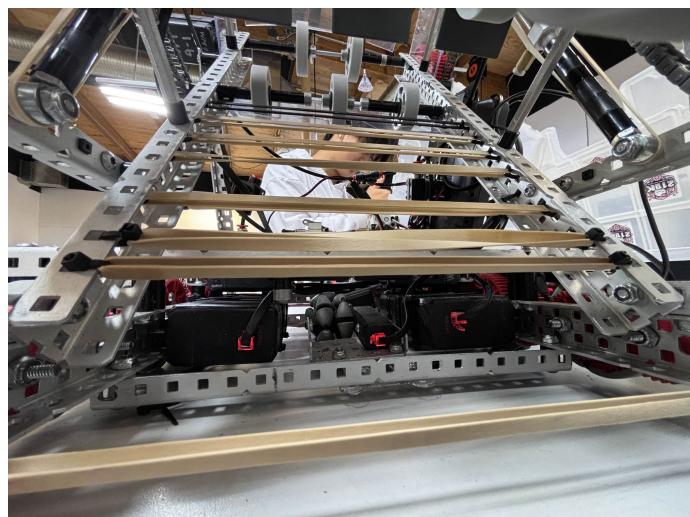


Figure 2: Rubber Band Ramp

- ▶ Problem: SG10 Challenges

Date: Oct 23, 2025

SG10 Ruling

In the October 9th game manual update. A paragraph was added to rule SG10 to clarify what anchoring could count as. Along with Q&A 2864, the GDC clearly does not want robots simply sitting in front of the goal with a mechanism inside the tube to prevent teams from scoring. With this new ruling, our previously built hood would break the new SG10 as part of the hood would be inserted into the open section of the long goal.

A Robot can only reach into any open portion of a Goal to move Blocks (e.g., into or out of the Control Zone or the entire Goal). If a Head Referee sees a Robot that has reached into a Goal and stayed there while the Robot isn't actively moving Blocks inside the Goal, that Robot should be verbally warned away and should receive a <GG9> Violation if it remains. See the <GG9> Violation Notes for more details, including how rules <GG15> and <GG16> may apply.

Figure 1: New paragraph under SG10 (Game Manual)

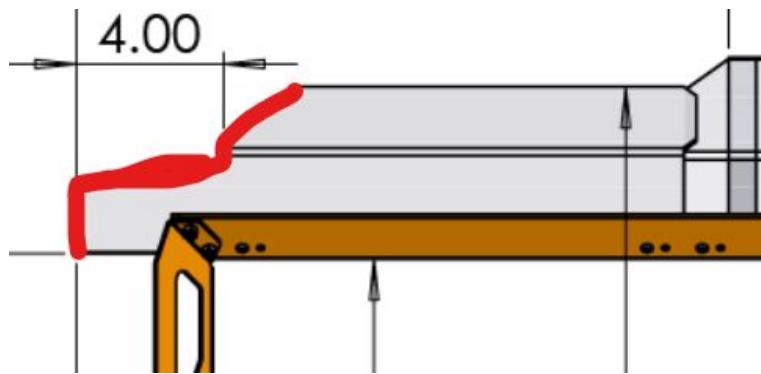


Figure 2: Anything below the red line is considered “in the open portion of the goal”

- ▶ Focus: Thinking of concepts for a hood design

Date: Oct 23, 2025

In order to comply with the SG10 ruling. We need our hood mechanism meet a few criteria in order to meet our needs.

1. It must rest above the open section of the goal. This is so we do not violate SG10
2. It must be able to prevent other teams from scoring. Our hood needs to be able to prevent scoring, this will improve our defensive capabilities
3. The hood must still be able to descore in some way.
 - a. If we ram into a goal with blocks in it, blocks should fall out the other end

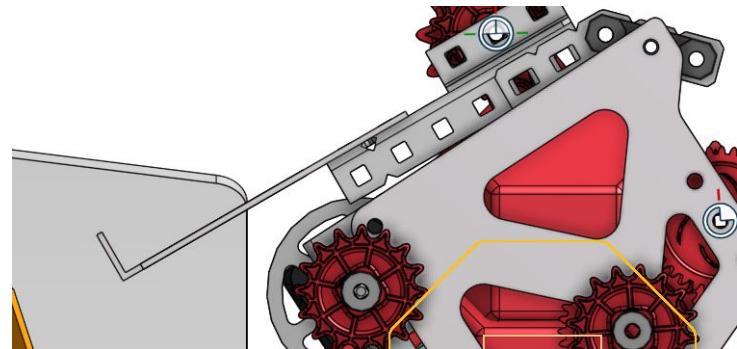


Figure 1: Our previous hood design entering the open section of the goal



Figure 2: A single piston pushing out 7 blocks from the middle goal



Figure 3: TNTN linear slide

Linear Slides

VURC World Champions TNTN had a unique piston linear slide mechanism for their 15 inch robot. This linear slide allowed for their wall stake mechanism to extend forward using a piston. This extension was limited and can be implemented onto our hood

Piston Descore

At MOA we saw that teams had early designs for piston descoring. They often had a flex wheel mounted on a piston that would stick into the long tube for pushing blocks out.

New Hood Design

Plan

► Focus: New Hood Design

Date: Oct 23, 2025

We planned out a new hood design to work around the issues caused by rule <SG9>.

This new hood design would be a Ruiguan hood that is attached to a piston on a linear slide. This means we are able to expand and retract our piston for linear movement in our hood. This allows our hood to sit outside of the goal when retracted and still allows us to descore when expanded.

We designed our hood to be attached on a machined linear slide incorporated into the plastic. When our pistons are retracted, it allows us to sit in front of the long goals. When the pistons are extended, the hood is able to push blocks out the other end. The extended pistons allow the our hood to function like a normal Ruiguan hood, allowing for us to run into the long goal for descoring blocks.

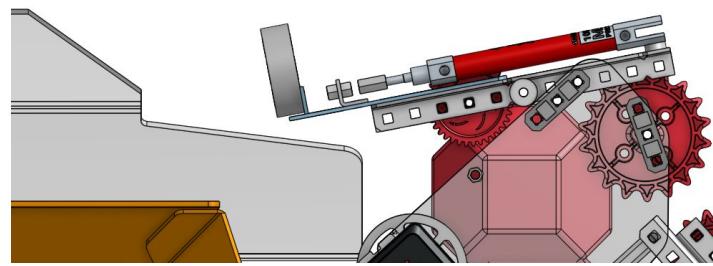


Figure 1: Retracted Hood

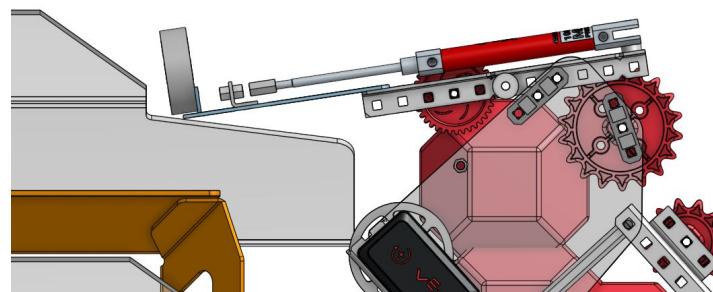


Figure 2: Extended Hood

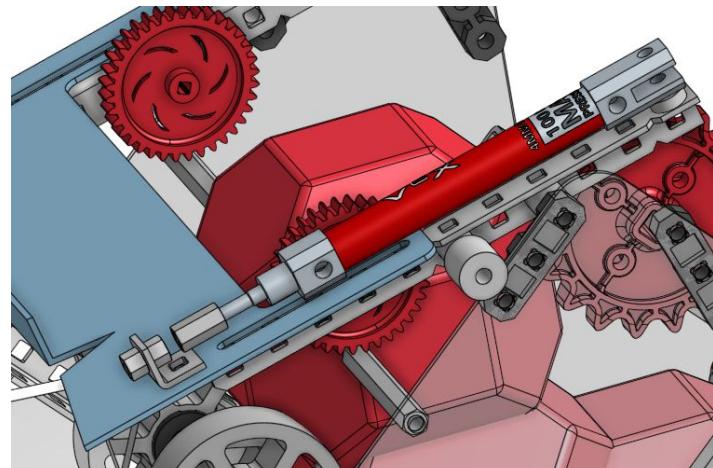


Figure 3: CAD of piston and sliding hood

New Hood Design

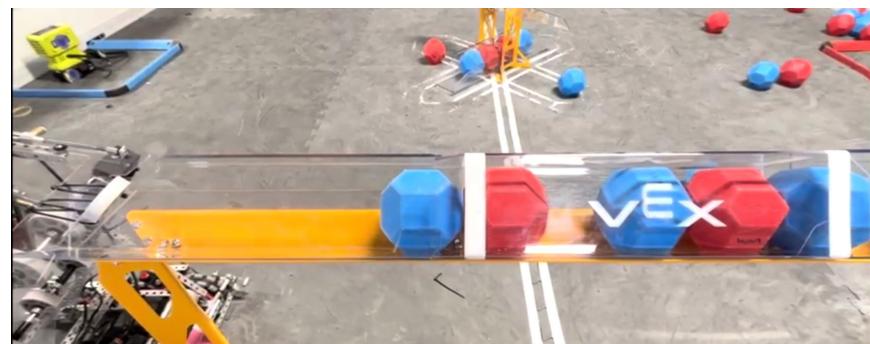
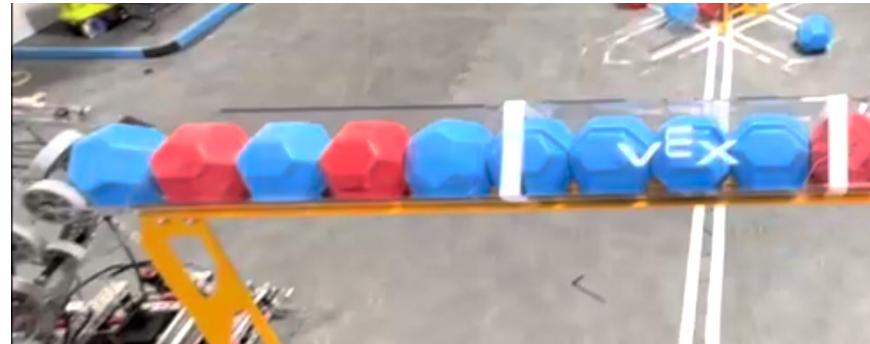
Plan

- Focus: Describing our pistonized hood design.

Date: Nov 5, 2025

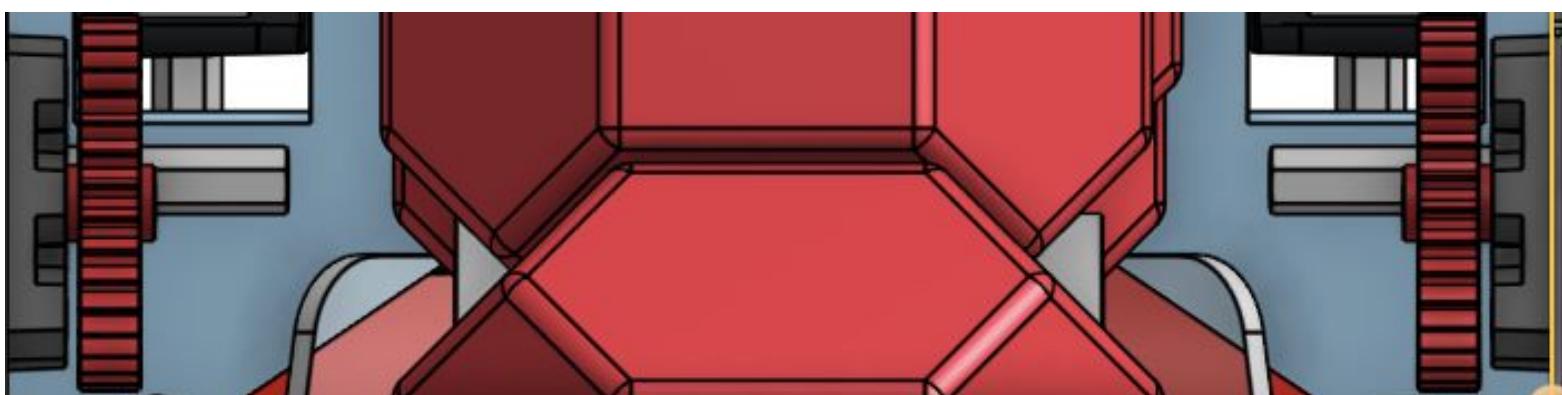
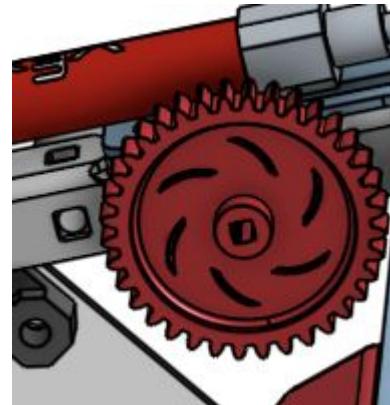
We designed the new hood design is made to block game pieces in the long tube. By being able to sit in front of long goals throughout the duration of a match, we are able to ensure we do not get our blocks descored as well as more effectively hold the control zone.

Our pistonized hood can further offer protection, being able to instantly push more blocks into the control zone.



Our design also incorporates a compression roller on the hood that helps with smoother transitions and a source of power transmission a lot more forward on the intake.

This includes rubber bands on 36T gears.



- Focus: Building our new hood design

Date: Oct 24, 2025

Members Involved

Michael, Daniel, Bryan

Objective

First, we used a CNC machine to cut out a base polycarbonate piece. This allows us to drill out holes for attaching our hood and rollers to. We also used the CNC to cut out our polycarbonate hood. We bent the very end of it to allow for contact with the block. This plastic base also serves as structure for another set of rubber band rollers.

After attaching the polycarbonate base. We attached angle bars onto our hood piece to create a linear slide. We then used two pistons and attached them to the plastic hood to allow for a sliding motion. When powered with air, the plastic piece is able to push outwards and descore blocks from the goal.

Materials

- 2, 11 Hole 1x1 Angle Bars
- 2, 1 Hole 1x1 Angle Bars
- 2, 50MM Pistons
- Sprockets
- 6P Chains
- Bearings
- Screws
- Spacers
- Nylocks

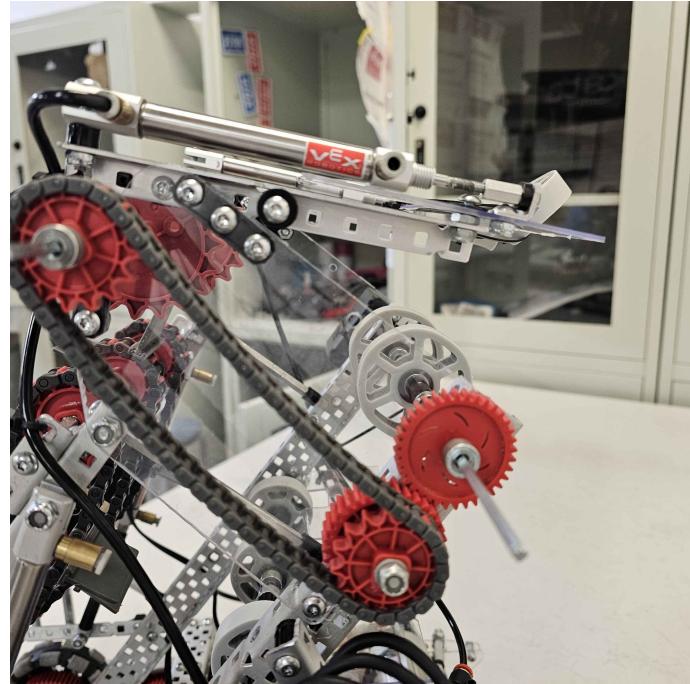


Figure 1: Side view of our hood

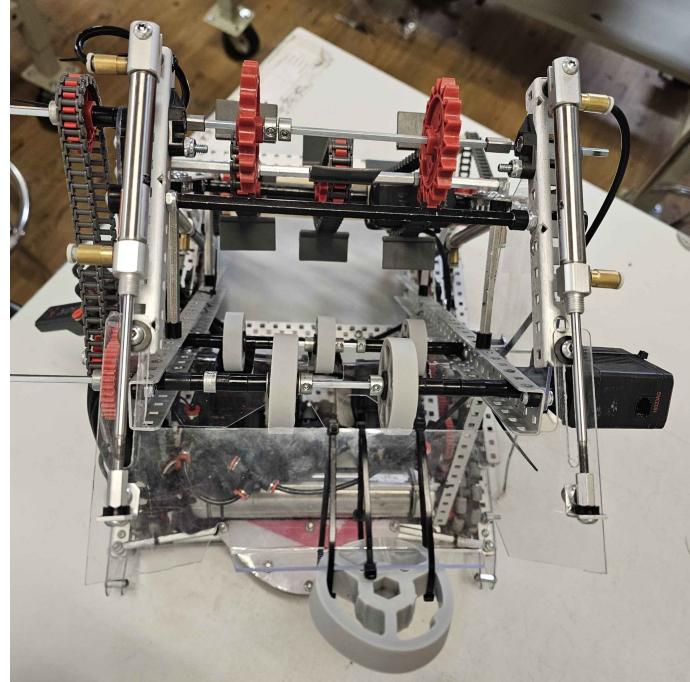


Figure 2: Top view of our hood

- ▶ Problem: Testing our new intake system

Date: Oct 24, 2025

Purpose

The purpose of the following tests is to find any issues with our intake system.

Procedure

1. Have a team member ready with a stopwatch
2. When ready, drive the robot with the intake running into two blocks that are positioned next to each other
3. When both blocks have made through the entire intake system and out of the robot, stop the timer and take the time down in our table
4. Repeat steps 1-3 for until there is sufficient data

Table 1: Intaking two blocks placed next to each other, t (s), in relation to trial number

Trial Number	Time to intake, t (s)
1	1.24
2	N/A
3	1.90
4	2.11
5	1.57
6	2.95
7	1.36
8	2.42
9	N/A
10	2.03



Figure 1: Testing Intake

Intake Jamming Issues

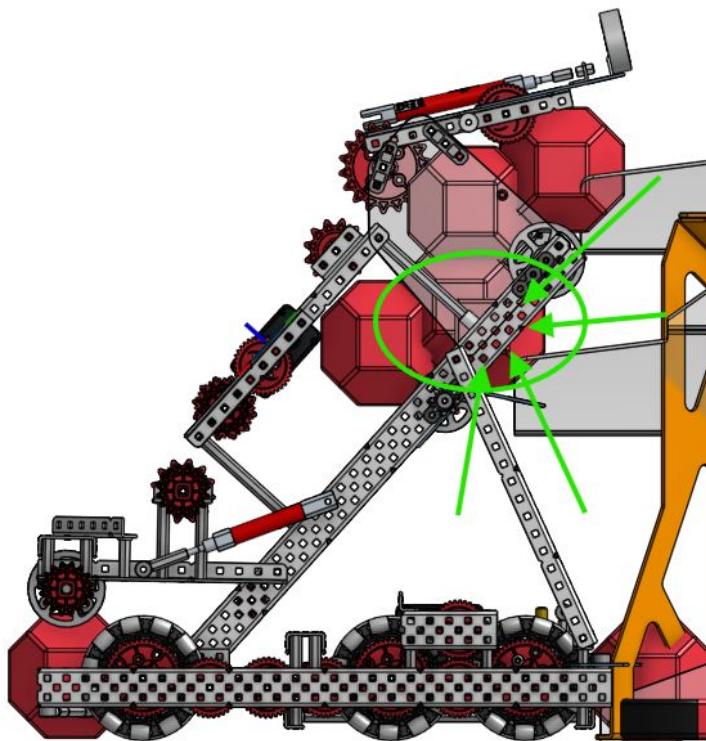
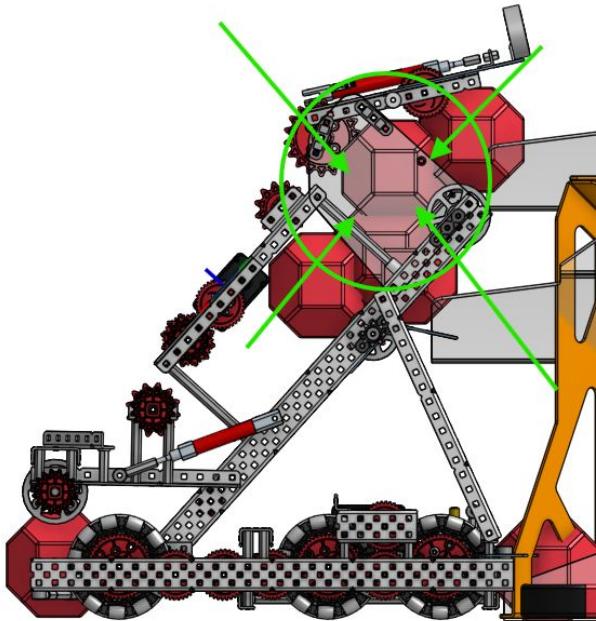
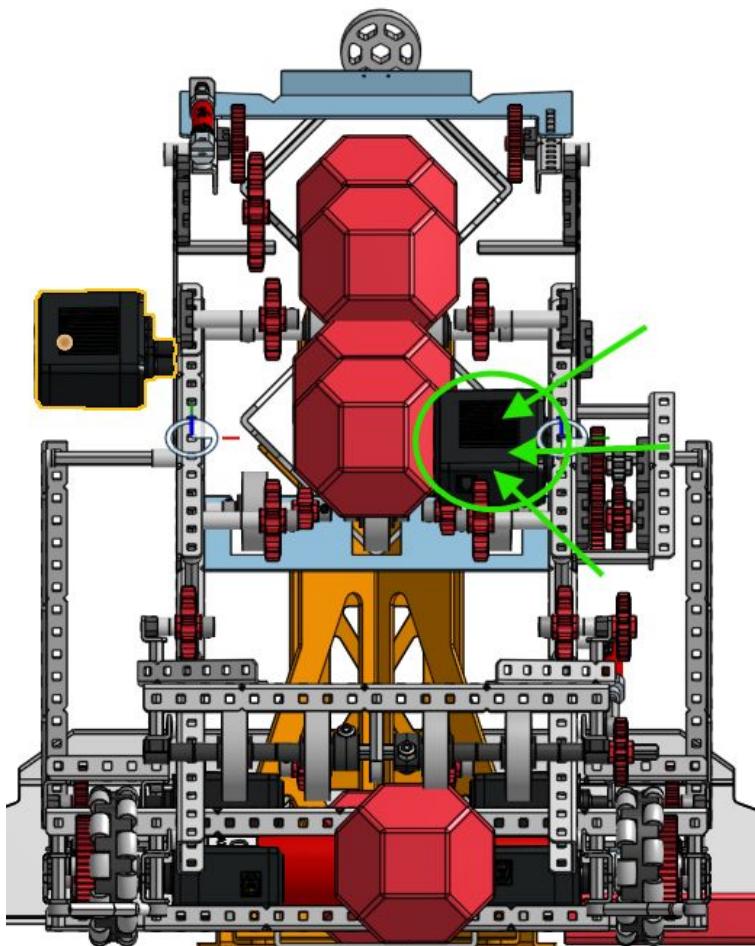
Identify Problem

- ▶ Problem: Why we are struggling on the intake

Due to the motor's positioning, our intake system frequently got stuck on it. The motor was mounted too low, causing the intake to catch and lock up the entire mechanism. Additionally, the motor's thickness exceeded that of the sprockets used on the ramp, which further contributed to clearance and interference issues within the system.

Additionally, the mid-goal featured a gap that was too large, allowing balls to slip into the mid-goal area and jam the bottom counter-roller. Combined with the intake interference issue, this effectively limited our robot's maximum capacity to just one ball at a time. This was further worsened by the large deadzone we found near the top of the intake.

Date: Oct 24, 2025



- ▶ Focus: Finding fixes to the jam issues

Date: Oct 24, 2025

There are several potential solutions to address the **mid-goal issue** and improve the overall **reliability** of our **intake system**:

1. Relocate the motor to the outside of the C-channel:
2. Remove the bottom counter-roller and replacing with a rubber band roller
3. Create a piston activated trap door that opens when we want to score on the middle goals
4. Implement a banding technique seen in 66556Z and multiple Californian teams robots.
 - a. This banding technique utilizes a piece of string attached to the rubber band ramp. When the string is punished against, the rubber band ramp is pulled and opens up allowing for blocks to travel through it.

Implementing these solutions would **significantly reduce jamming** and **clearance issues**, **increasing** both our **storage capacity** and the **overall efficiency** of the intake system.

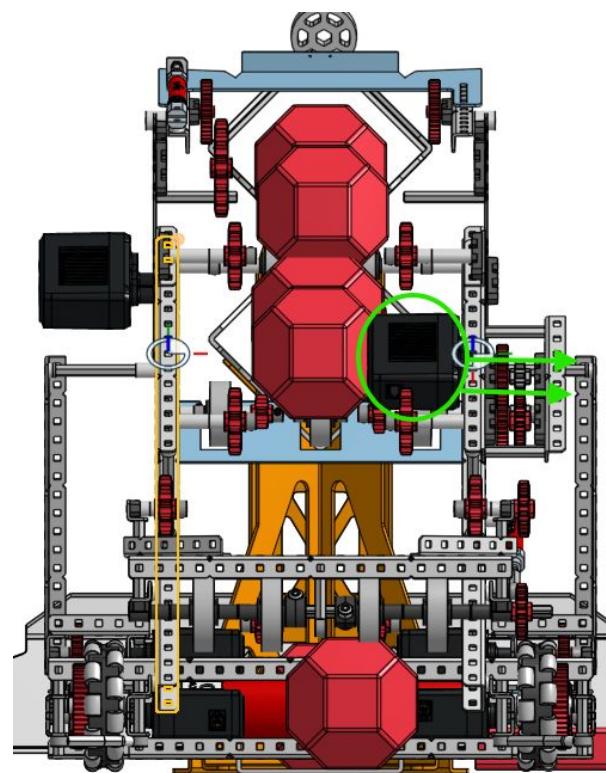


Figure 1: We could move the motor to where the arrows point

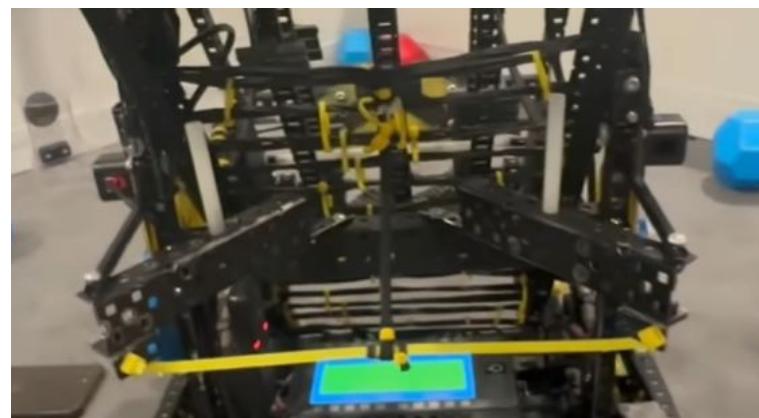


Figure 2: 66556Z's Middle Goal

Solving Jamming Issues

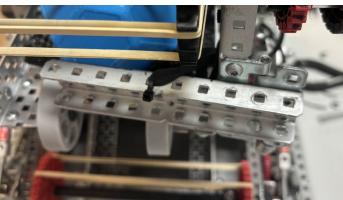
Select Solution

- ▶ Focus: Selecting a solution for our jamming issues

Date: Oct 24, 2025

After researching possible solutions, we decided to rank the following designs using the following criteria

1. **Build Simplicity (1 - 5)** - How simple the design is able to be implemented
2. **Weight Added (1 - 5)** - Ranked on how much weight this designs adds on
3. **Space Needed (1 - 5)** - Ranked on the amount of space needed to implement the design

<u>Criteria</u> <u>Options</u>	Build Simplicity	Weight	Space Needed	Total
Rubber Band Roller 	3	2	2	7
Piston Trapdoor 	3	2	4	9
String Banding 	4	5	5	14

Based on our decision matrix, we plan to implement either the string banding or a variation of this design onto our current robot.

- ▶ Focus: Solutions to fixing jam issues on the intake

Date: Oct 25, 2025

To address the first issue of the blocks getting stuck on the motor, we plan to relocate the intake motor to the outside of the intake structure. This adjustment will allow the motor to continue powering the intake system effectively while eliminating interference with the moving components. By doing so, we expect to prevent jamming and ensure smoother, more reliable operation during intakes and cycles.

Our second solution involves implementing a variation of the string and banding technique. Using this approach, we aim to create a precisely spaced gap in the mid-goal that securely holds the blocks without letting them jam or fall through. Through iterative testing and adjustment, we can fine-tune the spacing to achieve an optimal balance between grip and release, ensuring consistent and efficient block transfer.



Figure 1: NorCal teams mid goal design as shown for second solution

- Focus: Adjusting our intake system to prevent jamming

Date: Oct 26, 2025

Members Involved

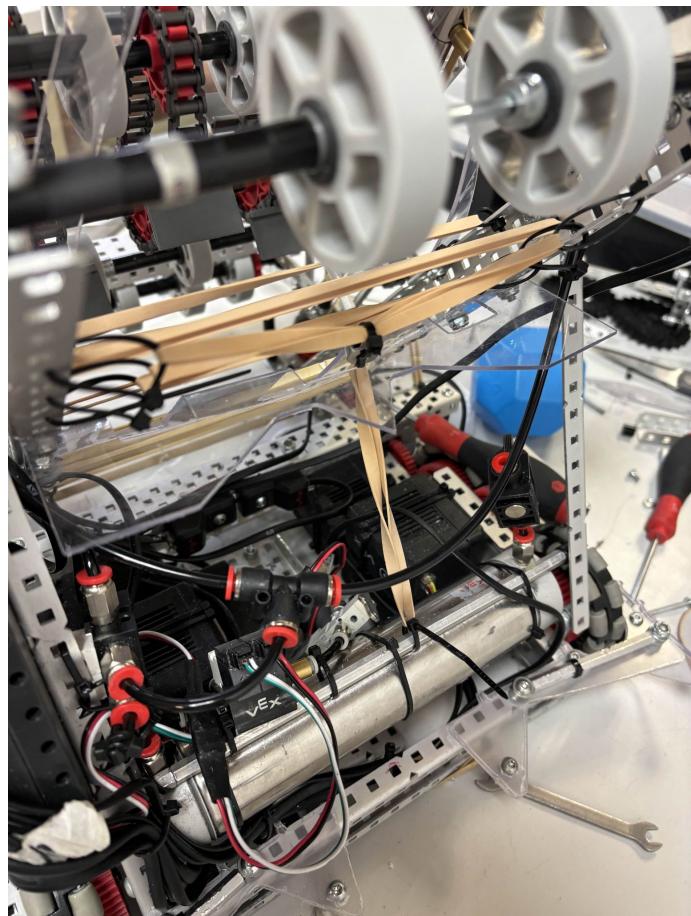
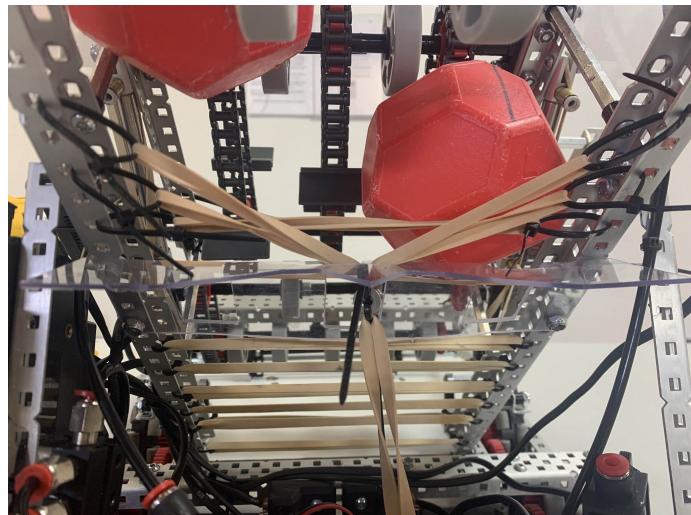
Bryan, Brandon, Daniel

Objective

Stop jamming while also keeping mid goal scoring quick and efficient.

Materials

- Zipties
- Rubber bands



- ▶ Focus: Small adjustments to improve our robots performance

Date: Oct 27, 2025

Members Involved

Daniel, Bryan, Joshua

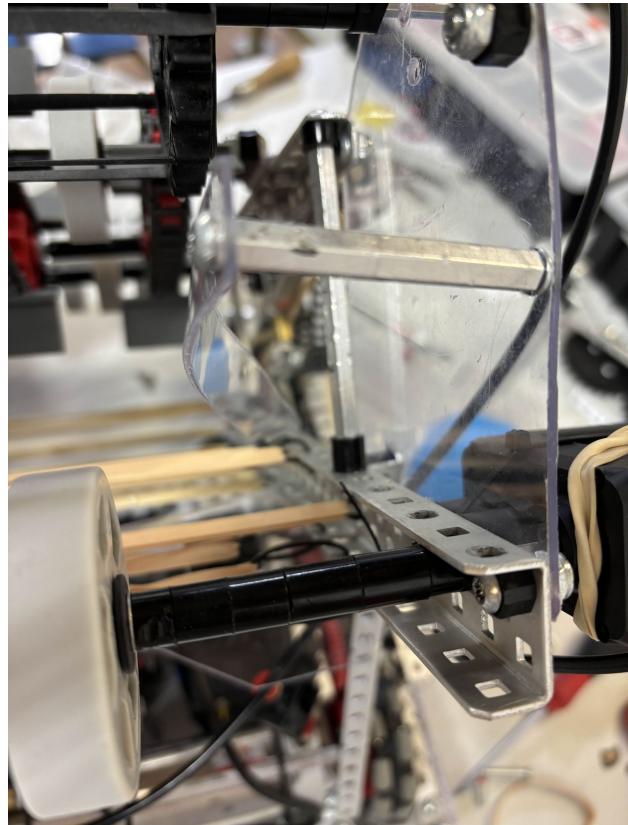
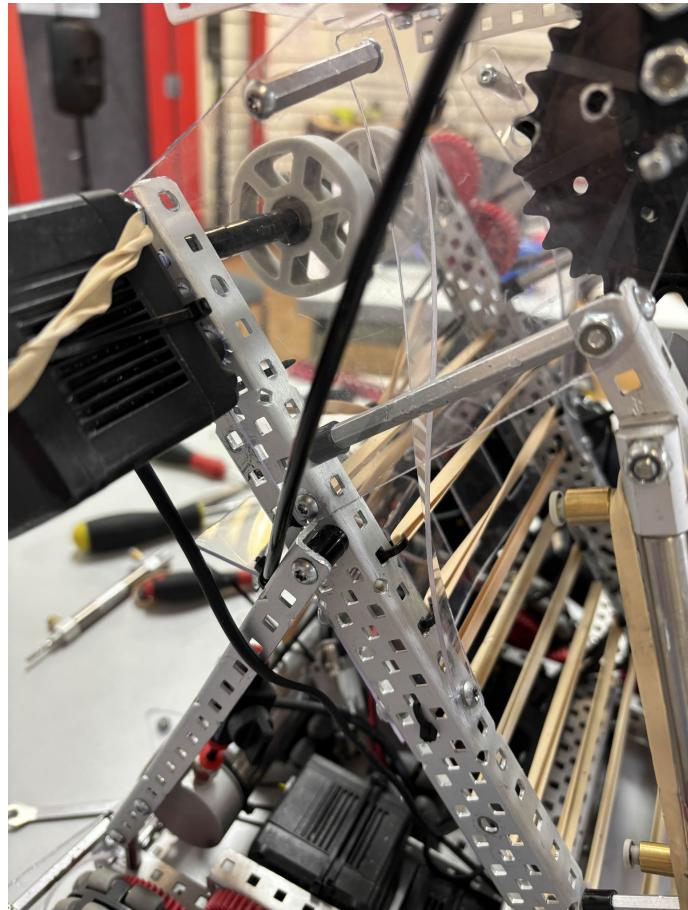
Objective

The first step to improving our robot was to minimize any friction sources. To do this, we isolated the system down to one roller and fixed anything causing friction. We continued this for all rollers and found that the main cause of friction was misaligned bearings. To solve this, we drilled the bearings out so the axle would not bend when attached.

Another addition we made were plastic funnels. We used strips of polycarbonate that are thin and easy to bend. This is so we are able to create a unique twisting shape for our funneling. These funnels allow us transition the intake from two wide to one wide in order to score into the goals.

Materials

- Polycarbonate strips
- Standoffs
- Screws
- Nylocks



Previous Loader Issues

Identify Problem

- ▶ **Problem:** Identifying issues with our previous matchloader

Date: Oct 29, 2025

The loader failed to operate consistently due to several design issues. In the first version, the system did not properly account for the loading angle, leaving virtually no tolerance for misalignment or variation during operation. As a result, even slight deviations caused jams or misfeeds. The second issue arose from the pivoting match loader design — at certain angles, the hexagonal face of the block sat below the lip of the loader. This misalignment prevented the block from engaging correctly with the mechanism, leading to inconsistent loading and occasional stalls.



Figure 1: MOA Livestream, qualification 7, showing our matchloader failing

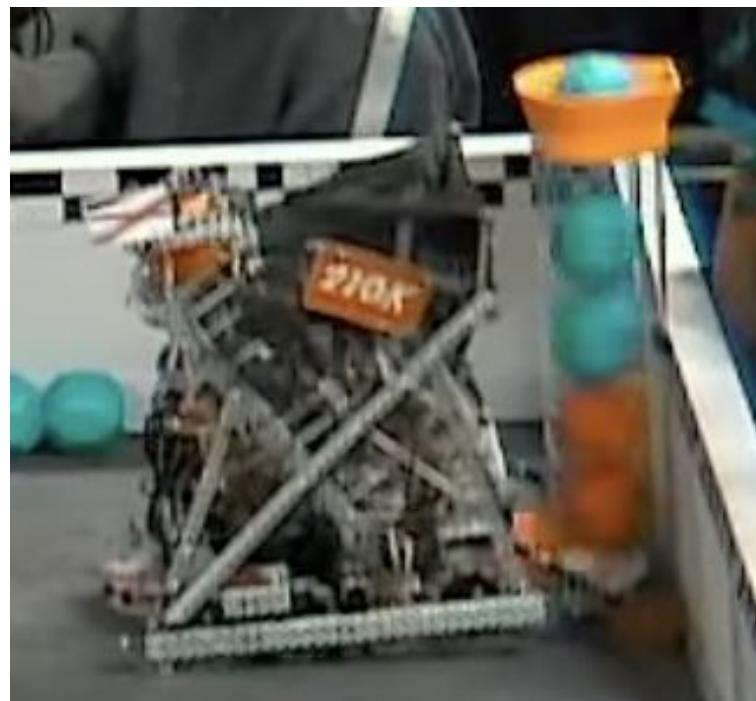
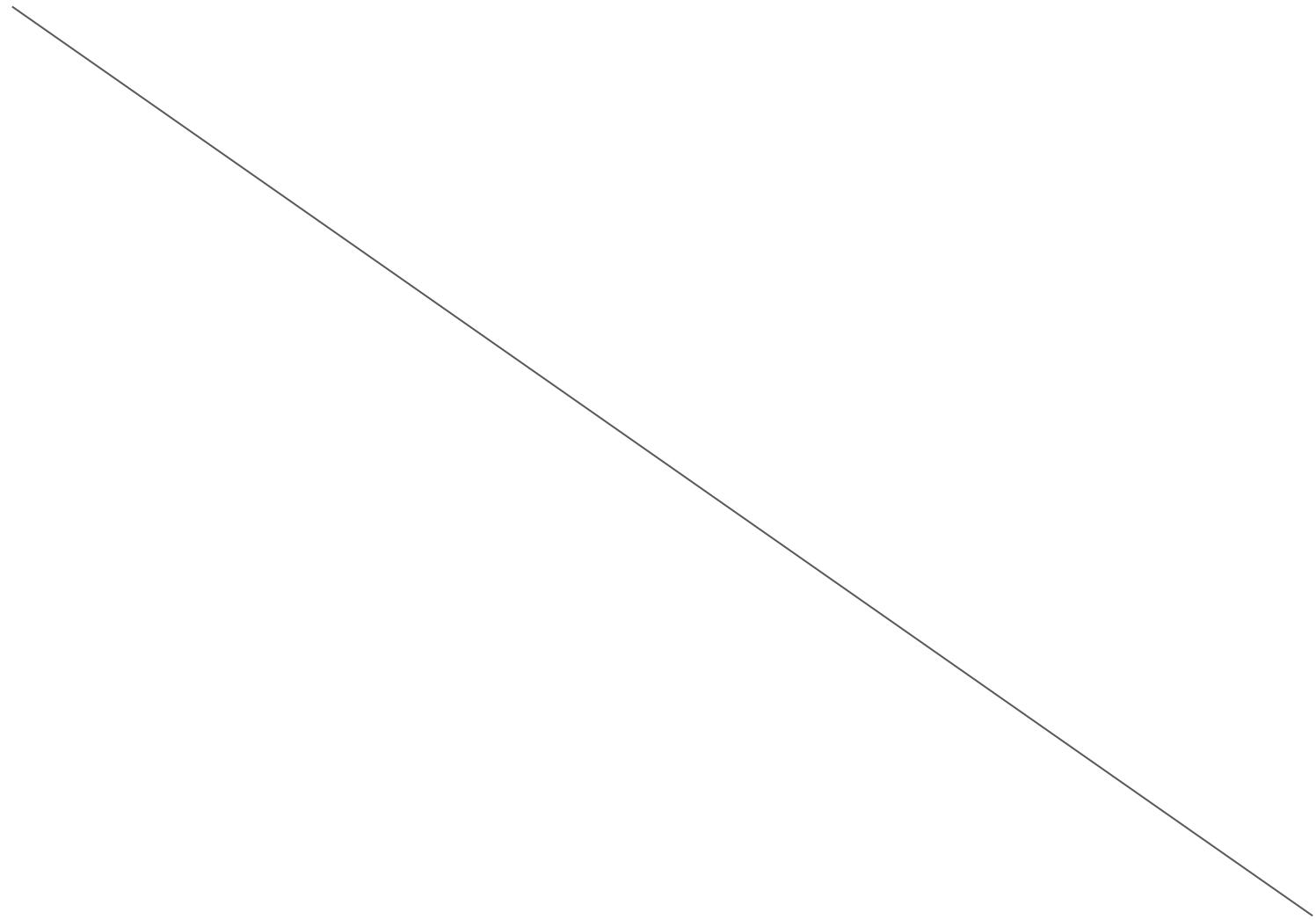


Figure 2: WM Opener Livestream, qualification 33, showing our matchloader working

- ▶ Focus: How to make the match loader

Date: Oct 29, 2025

The match loader has been designed to pivot in order to maximize loading efficiency. To further improve its performance, we plan to include a small V-shaped cutout. This cutout will prevent the lower segment of the ball from catching on the match loader, reducing the risk of jams or backward rebounds during loading. By ensuring smoother ball entry, this modification should enhance both reliability and cycle speed during matches.



- Focus: Materials for Matchlaoder

Date: Oct 31, 2025

Members Involved

Brandon, Daniel, Bryan

Objective

Building the matchloader

Materials

- Polycarbonate
- Screws
- Nylocks
- Angle bars
- High strength axles
- Bearings
- Pistons
- Standoffs
- Spacers
- Rubber bands
- Zipties

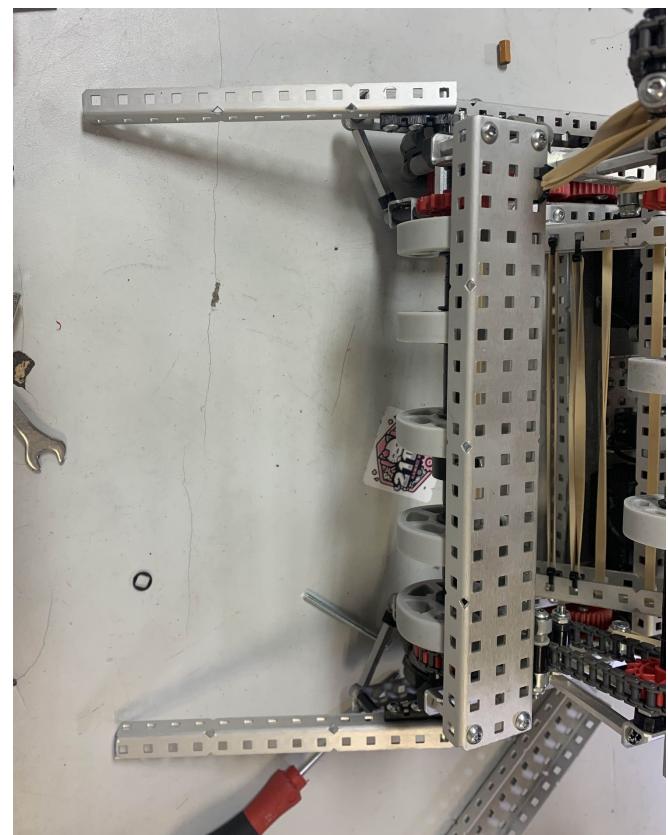


Figure 1: Mounted angle bars for the matchloader to be built on

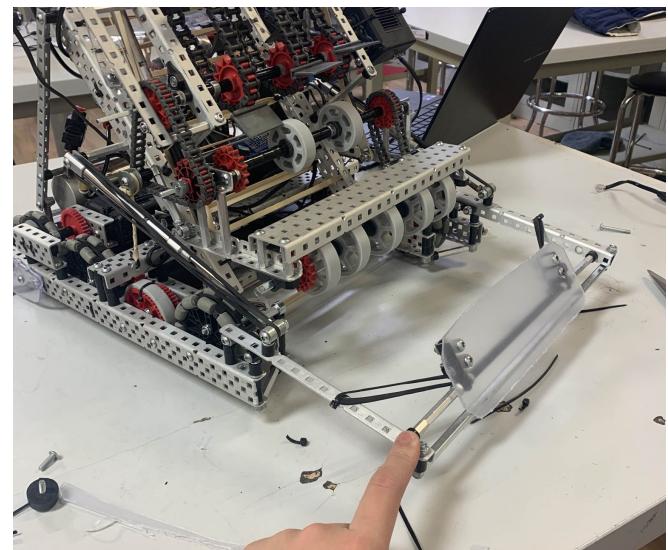


Figure 2: Finished matchloader

Testing New Matchloader

Test

- ▶ Problem: Testing for efficiency

Date: Nov 1, 2025

Objective:

Verify that with the loader held in a fixed orientation (no pivoting) the system reliably feeds blocks across the range of angles we previously investigated.

Method:

Simulated test runs used pivot-angle-equivalent orientations from 0.80 to 2.30 radians (two-decimal precision) to represent different approach geometries; the loader itself was held fixed for every run. Each row below is one simulated cycle under controlled conditions.

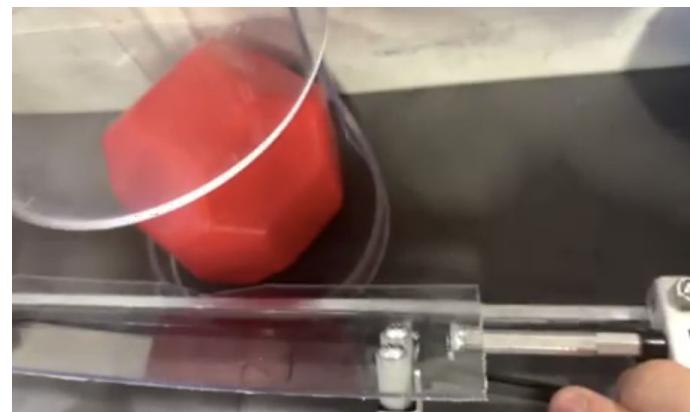


Figure 1: Aligning our matchloader with the tube

Trial	Feed Result	Time (s)
1	Pass	1.32
2	Pass	1.15
3	Pass	2.13
4	Pass	0.94
5	Pass	1.68
6	Pass	1.74
7	Pass	1.32
8	Pass	2.13
9	Pass	1.46

Analysis:

Most runs were satisfactory, however the extraneous 2sec+ runs should be investigated.

- The rubber band strength on the pivoting lexan may need further tuning to ensure the plane is intersecting the tube at an optimal angle.

- ▶ Focus: Planning out new plastic pieces according to our old issues

Date: Nov 2, 2025

Old Plastic Issues

During the competition we noticed two major issues with our sleds and aligners.

1. Our aligner pieces were mounted low and would not work if there was a block in front of the long goal. Because our plastic aligner was in the center of our robot, if there was a block in the V shaped post, we would not be able to align properly.
2. Our single sheet sleds bent very easily when being used. In order to solve this on our old robot, we cut a small piece of plastic to create support by bracing the sled. However, on our current robot we are unable to spend extra plastic on our sleds

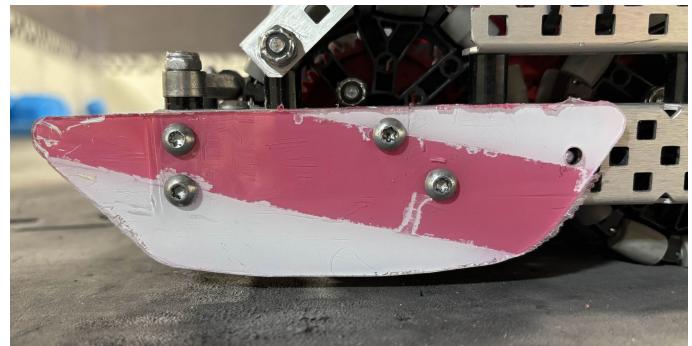


Figure 1: Crease in our sleds after being bent

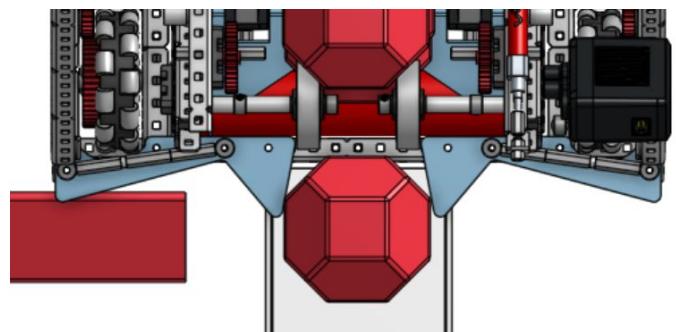


Figure 2: New aligner design

New Design

To solve this problem, we planned out a new aligner. This aligner still uses two pieces of plastic however they are an **inverse** design where the aligner is cut out to align with the edge of the posts. This allows us to align even if there is a block in the way. Our design is double stacked where we are able to bend our plastic in half and make our aligner two layers. This strengthens the plastic and makes sure it does not bend.

We applied the same concept of double stacking on our sleds to ensure they do not bend.

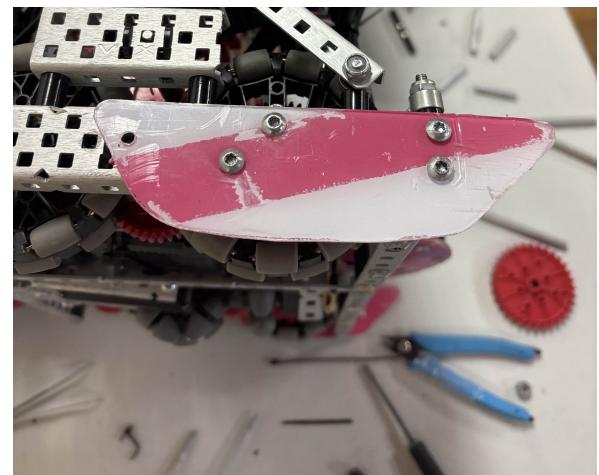


Figure 3: Side angle of our old sleds

- ▶ Focus: CAD and implementation of improved sleds and aligner

Date: Nov 5th, 2025

Members Involved

Brandon, Daniel, Bryan

Objective

CAD and CNC new plastic pieces, and replace old pieces with them.

Materials

- Lexan (Polycarbonate)
- Screws
- Nylocks

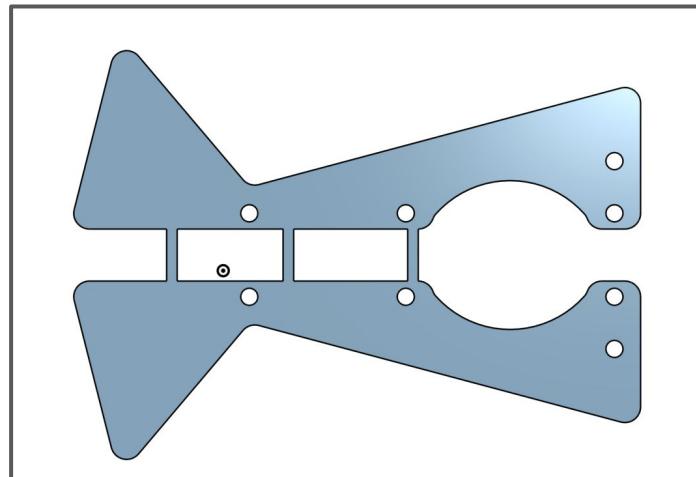


Figure 1: CAD of our new aligners

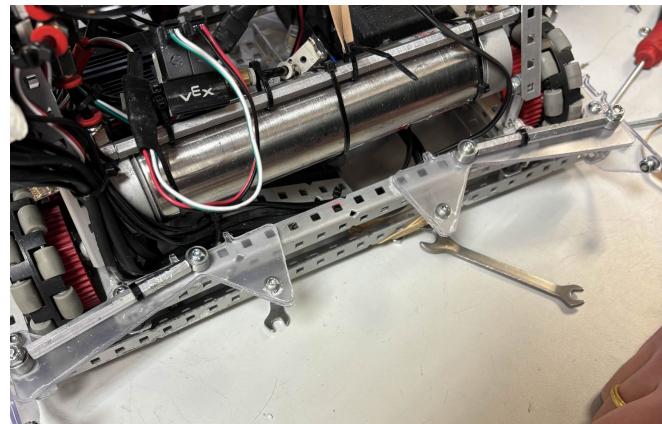


Figure 2: Folded and mounted aligners

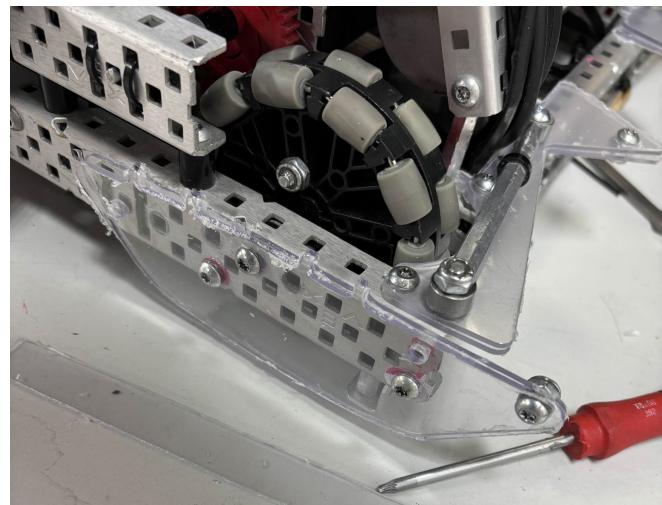


Figure 3: Folded and mounted sleds

► Problem: Determining how well our aligners work

Date: Nov 5th, 2025

The aligners need to:

- Be able to align regardless of whether or not there is a block at the base of the long goal
- Be able to align at many different angles to the long goal

Objective

Determine the angle of approach at which the aligner can still align to the long goal.

Method

- Drive the robot into the long goal such that θ is 5° .
- After contact with the long goal, continue driving backwards into the goal.
- Record success if the robot is aligned to the long goal, such that θ is now 0° , and failure if it is not.
- Upon success, repeat steps 1-3 such that $10^\circ + \theta_{\text{old}} = \theta_{\text{new}}$. Upon failure, repeat steps 1-3 with the same value of θ , and end testing if failure is recorded 3 times.

Trial	Status	Angle, θ , ($^\circ$)
1	Pass	5
2	Pass	15
3	Pass	25
4	Pass	35

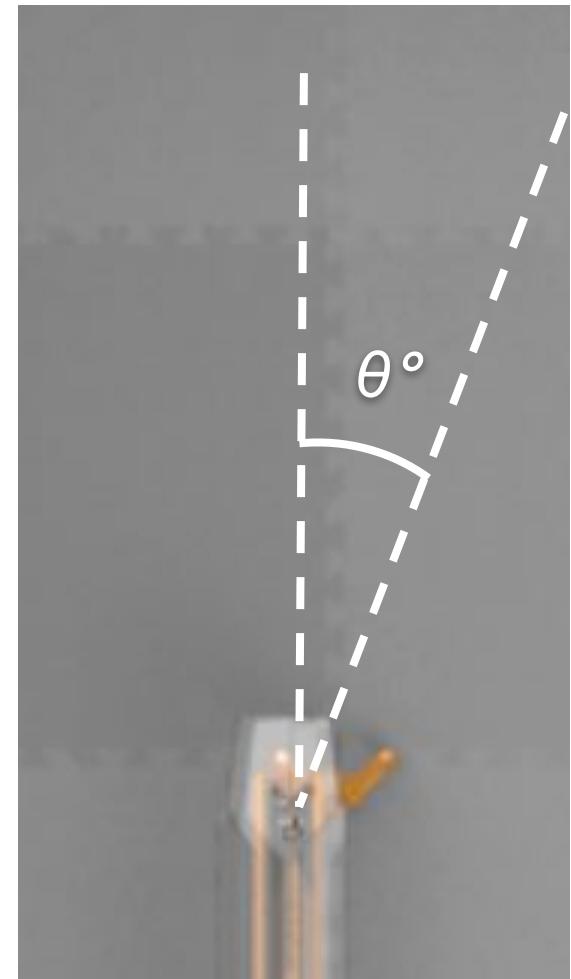


Figure 1: The angle theta (θ) showing the angle of approach

5	Pass	45
6	Pass	55
7	Pass	65
8	Fail	65
9	Fail	65

Figure 2: Status as a function of angle, θ , ($^\circ$) for the 210K robot driving backwards into the base of a long goal

Why our intake is struggling

Identify Problem

- ▶ Problem: Why the intake is struggling

Our intake is currently underperforming due to several mechanical issues, including but not limited to:

- Large deadzones

These deadzones significantly reduce efficiency, leading to inconsistent ball control. In many cases, balls behave almost like “excited electrons,” bouncing unpredictably when we attempt to score rather than feeding smoothly through the system.

- Suboptimal tensioner geometry

Because our robot must accommodate the double park, a tensioner is necessary. However, the current placement and angle of the tensioner are far from ideal. While the intake itself performs adequately, the geometry results in a noticeably weak outtake, limiting our cycle speed and throughout.

The tensioner is needed due to our double park being on the intake. So to solve this issue, we first tried to rebuild our double park mechanism.

Date: Nov 8, 2025

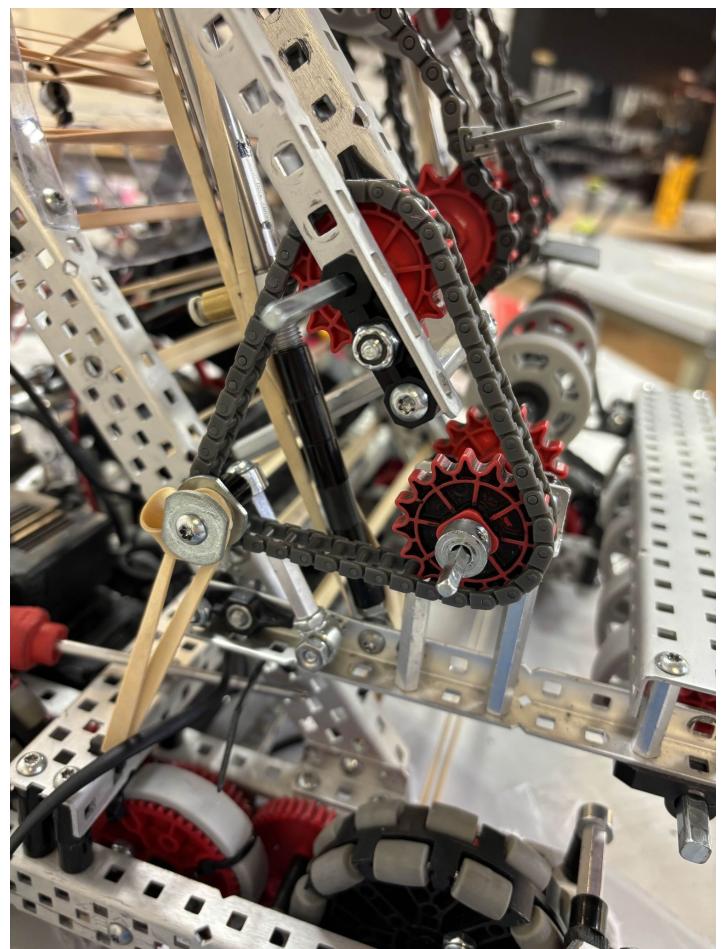


Figure 1: Intake tensioner

► Problem: Testing our newly rebuilt mechanisms

Date: Nov 13, 2025

Objective

Determine the average time taken for a single block to fully cycle from our first intake stage to being scored on the long goal hood.

Method

1. Ensure motors are at room temperature
2. Run all intake rollers at maximum speed
3. Record a video of the setup
4. Slowly push a block into the first rollers, wait until it comes out the long goal scoring end
5. Calculate time taken between the block making first contact with the roller and the last frame where it breaks contact from the hood
6. Record success if $t < 2.0$ seconds

Trial	Status	Time, t (sec)
1	Pass	1.6
2	Pass	1.5
3	Pass	1.8
4	Pass	1.5

5	Pass	1.6
6	Pass	1.6
7	Pass	1.9
8	Fail	2.1
9	Fail	2.0

- We think the last two fails are partially due to motor temperatures overheating above 45°C, causing internal software power draw limiting
- Additionally, we noticed the deadzones are better, but balls spend most of their time in the storage stage

- ▶ Focus: Rebuilding our double park

Date: Nov 12, 2025

Members involved

Bryan, Daniel

Objective

We rebuilt our double park mechanism by using a separate piece of metal in order to push the block down. This is so we would no longer need a tensioner on our intake and would lead to a performance increase in our intake.

We created a separate mount for the pistons that were then braced and attached to an angle bar. This angle bar would push down in between our first and second set of flex wheels. We then added a layer of mesh in order to maintain grip on the block when pushing down.

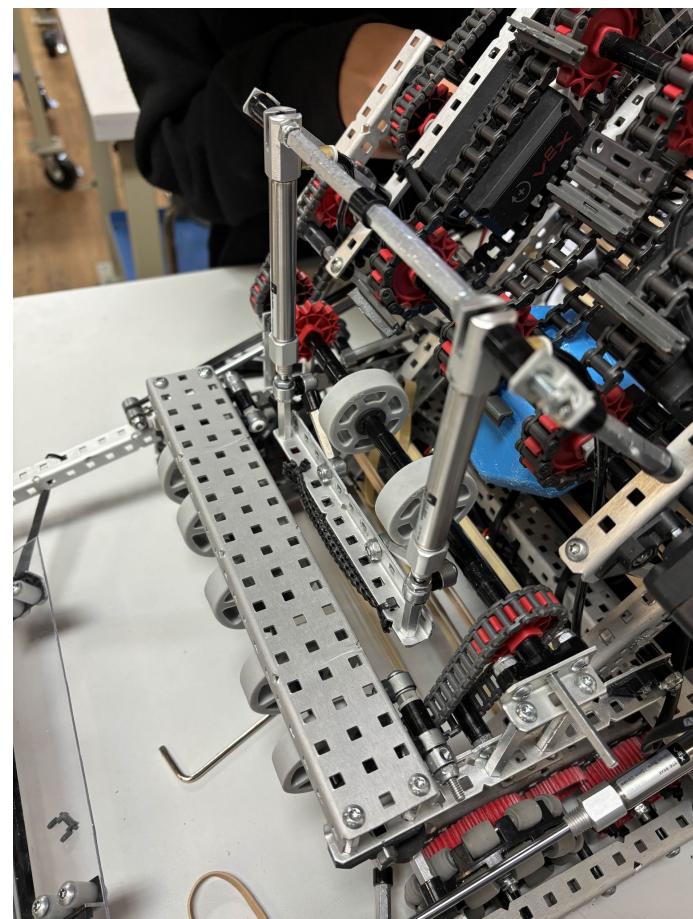


Figure 1: New double park mechanism

Materials

- 2, 50mm pistons
- 1, 11 hole 1x1 angle bar
- Standoffs
- Couplers
- Screws
- Axle Collars
- Nylocks
- Zipties
- Mesh

► Problem: Testing our newly rebuilt mechanisms

Date: Nov 13, 2025

Objective

How long will it take for our double park macro to detect a ball and trigger the pistons?

Method

1. Set the back of the robot up on the park barrier
2. Intake a block into storage and let it naturally come to rest between the storage tank treads and the 2nd roller stage
3. Trigger the macro and record the process to determine the time later
4. Record success if the entire robot is “parked” or elevated off the ground

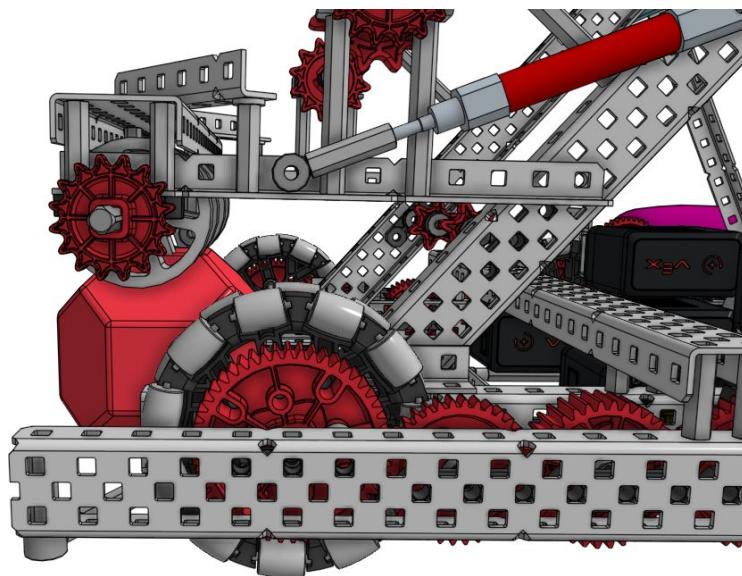


Figure 1: 2D perspective CAD with the park pistons and a block under the 1st stage rollers

Trial	Status	Time, t (sec)
1	Pass	2.2
2	Pass	1.9
3	Fail	X
4	Pass	2.1

5	Pass	2.3
6	Pass	1.8

- In trial 3, the macro did not trigger at all. We likely need to increase the refresh rate of the sensor in the macro code

► Focus: Strategies of removing double park

Date: Nov 15, 2025

After careful consideration, we have decided to completely remove our double park system. There are three main reasons justifying this:

1. Extra air usages

We want to minimize weight and because we have so many pistons, if we want to be able to double park, we needed a second tank, thus more weight.

2. Its slow

At signature events, we commonly see that wings have dominated the meta, in a new low score, control zone dominated game. By the time you pull off the park at 10 seconds, your entire goal will be descored and your opponent will take the control zone using their wing.

3. Score differential

As mentioned earlier, having a full wing descore means you lose 15 balls + the long goal control zone. That is 45 points + the 10 point control zone, for a total of 55 points. The double park is worth 30 thus granting a net -25 point differential.

4. Interference with our intake

Due to the fact we have a double park mechanism. This causes us to require a tensioner in order for the intake system to work. The removal of the double park would mean we would not need a tensioner which is causing our intake to perform suboptimally.

These are some reasoning why we have decided to remove our double park system.

- ▶ Focus: How to build the wings

Date: Nov 15, 2025

Wings are game-changers this season-literally. A well-timed wing hit can swing the entire match in seconds, which is why so many teams are choosing wings over double parks. To build a double-wing system that actually wins matches, three traits matter most:

1. Long Reach

Your wings need serious length. The more space they control, the bigger the impact—both in points and in how much pressure they put on the opposing alliance.

2. Fast, Clean Alignment

A good wing doesn't waste time. It deploys instantly and lines up without the driver having to fight it. Quick, reliable activation means you can use it aggressively, even in chaotic endgame moments.

3. Strong Momentum Transfer

Power matters. The wing should hit with enough rigidity and control to move what it needs to move—no wobble, no loss of force, just clean, effective momentum transfer.

These three elements are what separate a “nice idea” from a match-deciding double-wing system.

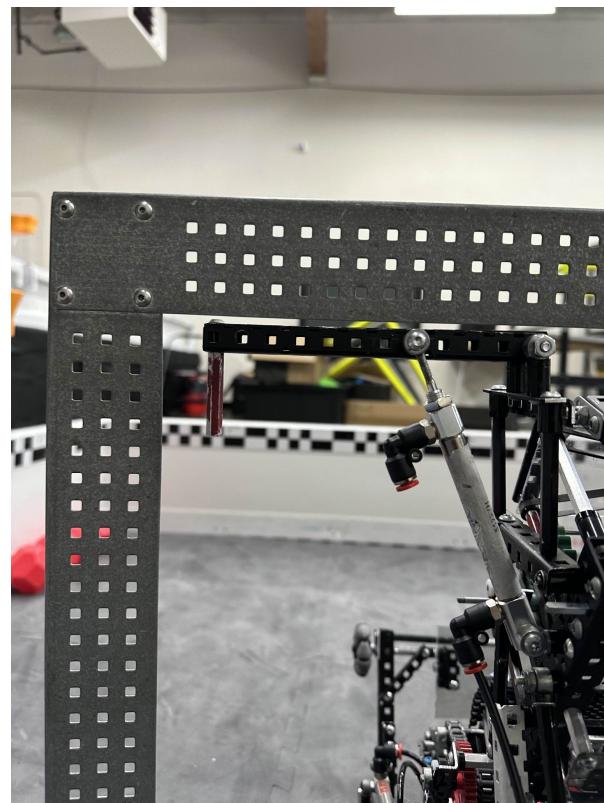


Figure 1: Picture of 917X's wing

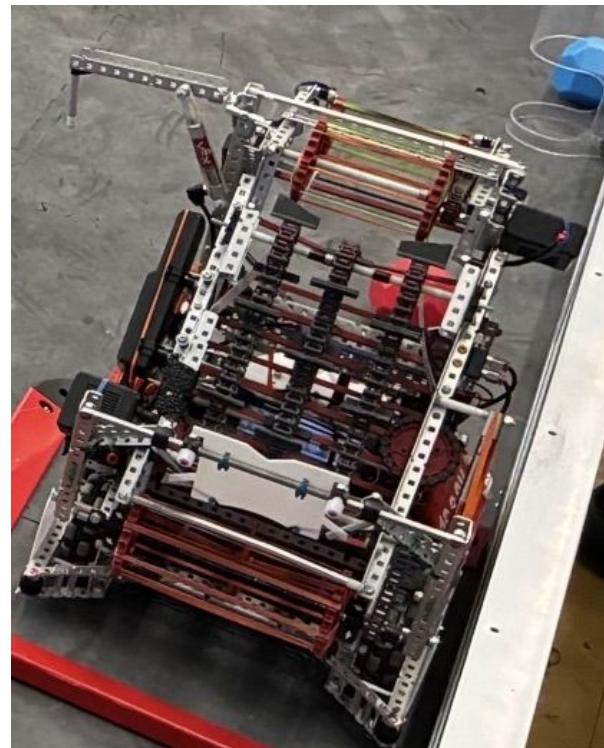


Figure 2: Picture of 1698V's wing

- ▶ Focus: Figuring out spacing and length

Date: Nov 15, 2025

Members Involved

Daniel, Bryan

Objective

The objective was to figure out the size our wing would need to be in order to

- a. Stay in the 18x18x18 and the 22x22x22 size limit
- b. Be able to extend beyond the drivetrain and stick into the long goals
- c. Be rigid enough that our wing would not bend

We first started with figuring out the size of our wing. In order to do this, we found the maximum length our wing could be, 12 holes long, and attached it to an axle collar so it could rotate. We then attached a standoff to the other end of the angle bar so we would be able to push blocks with it



Figure 1: First test of building our wing. A 15 hole angle bar is used to find out how long we need in order to insert it into the long goals.

Materials

- 1, 25mm piston
- 1, 12 hole 1x1 angle bar
- Axle collars
- Spacers
- Screws
- Standoff

- Focus: Refining our wing

Date: Nov 16, 2025

Members Involved

Daniel, Bryan

Objective

Once we figured out the spacing needed on our wings, we started with creating a stable mount where we could attach our wing to. This would be a 19 hole halfcut 3 wide. We then created our wing using a 12 long angle bar that was boxed to make it stronger.

Materials

- 1, 19 hole 1x1 halfcut
- 2, 12 hole 1x1 angle bar
- Screws
- Spacers
- Nylocks
- Standoffs
- 1, 25mm piston

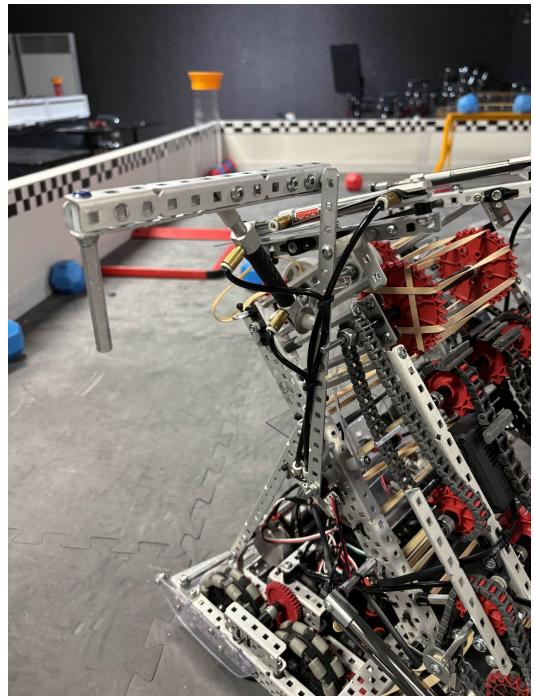


Figure 1: Our finished and refined wing.



Figure 2: A different angle of our wing

► Focus: Our Various Paths drawn out

Date: Nov 20, 2025

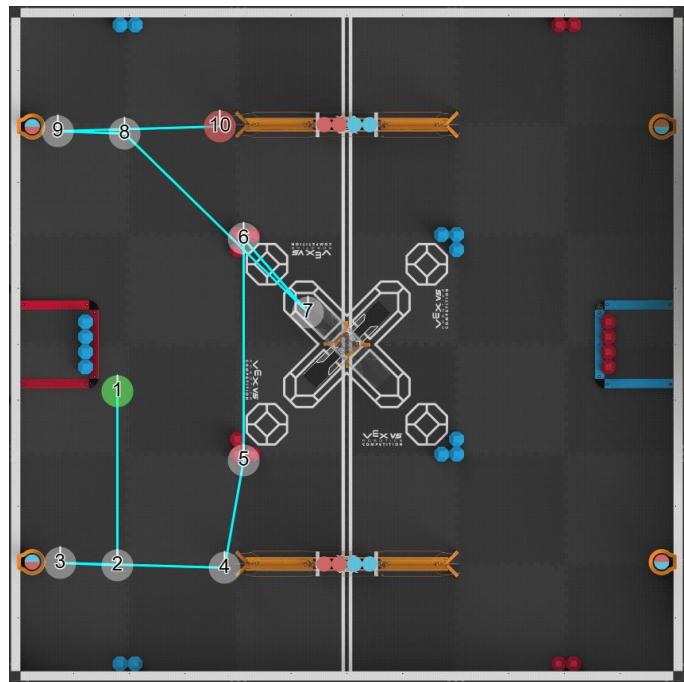


Figure 1: Solo AWP

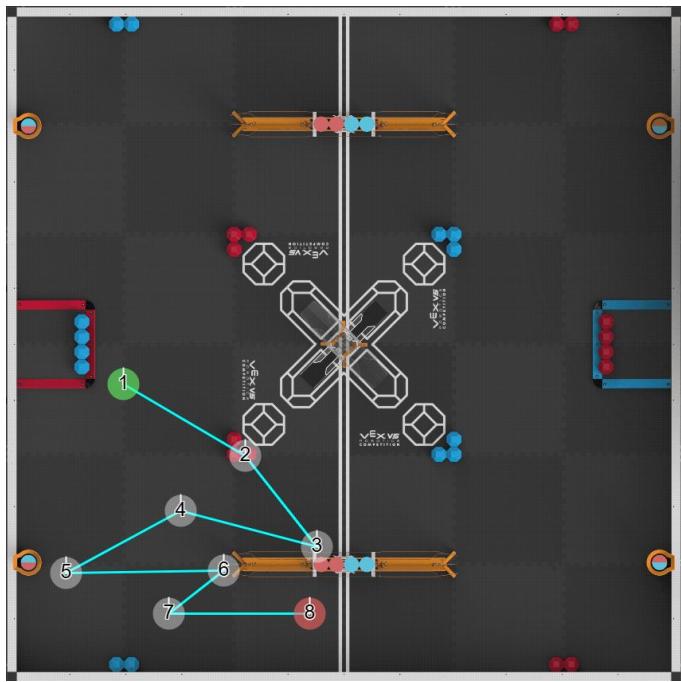


Figure 2: 9 Ball Wing

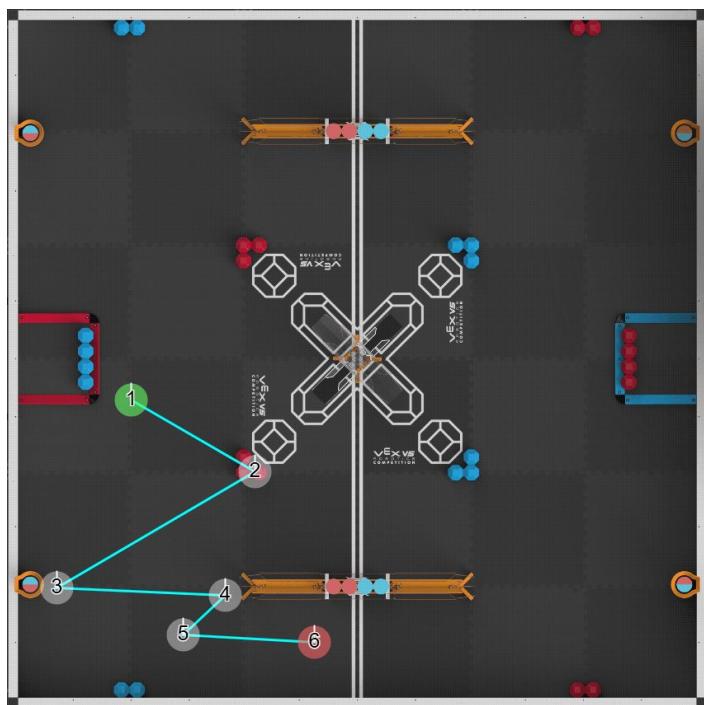


Figure 3: 7 Ball Wing

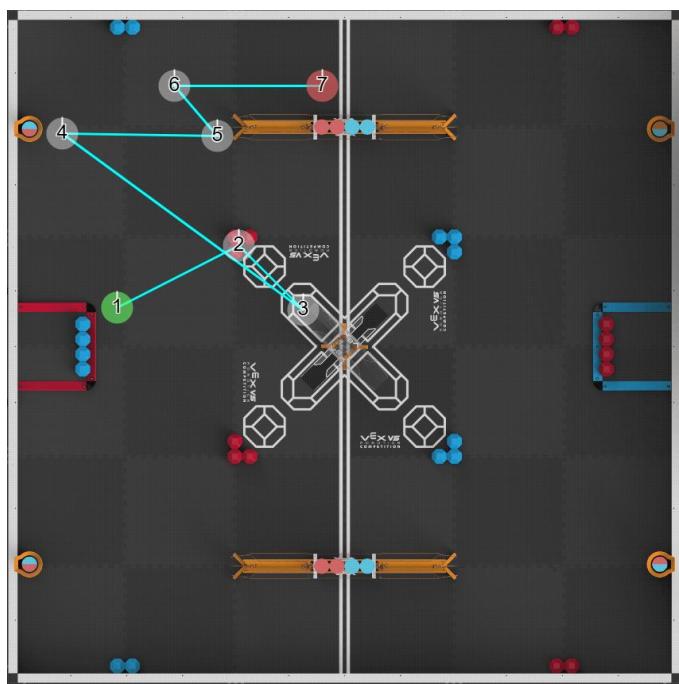


Figure 2: 3 + 4 Wing

Path Planner used: <https://210k.westernmech.ca/path/>

- ▶ Focus: Explain our goals heading into UC Berkeley

Date: Nov 21, 2025

With our robot now fully completed and our signature event fast approaching, we're excited to set clear goals for ourselves and push toward our strongest performance yet. We have real confidence in this new V2 bot—its improvements, reliability, and overall capabilities give us every reason to believe we can meet (and possibly exceed) the standards we've set. With focused preparation and a motivated team, we're ready to take on this event and make these goals a reality.

Skills Goals

Driver Attempts	Driver Highscore	Programming Attempts	Programming Highscore	Total Score
X	75	X	75	150

In theory, our skills should be able to hit 150, as well as hopefully our driver will get practice during the competition to hit a semi high driver run.

Overall Goals

Our goals for One World @ UC Berkeley are ambitious, but absolutely within reach. Specifically, we're aiming to:

- Win an award
- Qualify within the top 16 teams
- Break the 100+ skills barrier

We're confident in these targets because of the strength of our current robot, the consistency we've shown in skills, and the momentum we've built throughout the season. Our interview on November 5 went exceptionally well—what we perceive to be our strongest interview to-date—which gives us even more confidence heading into this signature event. With our preparation, teamwork, and continued improvement, we believe we're positioned not just to compete, but to truly stand out at One World.

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Loss	Q5	89445A 210K	31	91231X 7700S	57	No	No

Notes

Our alliance partners could not score and ended up being a bottom 3 ranked team in the competition. We did not have many opportunities to swing the game in our favour.

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	Q27	5776K 2397A	25	210K 5776T	81	Yes	No

Notes

A well played match. Our colour sort triggered and caused us to miss SAWP.

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	Q38	89445B 89445G	9	2550X 210K	130	Yes	No

Notes

Our opponents were not able to effectively score, so we did not utilize our wings that much and mainly focused on scoring.

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	Q50	9568X 5327R	14	2428W 210K	107	Yes	No

Notes

Our SAWP missed on the last goal. We were able to score on the middle goal in the last second to secure it.

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	Q71	210K 95071Y	60	10C 99125Z	23	Yes	No

Notes

This match was a very competitive match against a Ruiguan team and Ex Machina a 4 time dome. We primarily scored on the bottom goal and the right side long goal. We were able to secure our long goal using our wing.

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	Q80	9621V 16689G	22	7899B 210K	74	No	No

Notes

Our auton crossed the line, resulting in an auto loss. We played relatively well, holding onto long goal control zones and the upper middle goal.

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	109	98040C 210K	63	5327V 10009A	15	Yes	No

Notes

Another relatively competitive match but we were able to hold onto long goal control zones and the upper middle goal.

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	122	95071V 210K	56	10081K 1862A	50	Yes	No

Notes

This match was very close, we were able to hold onto middle goal control zones which swung the match in our favour.

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Loss	R16 2-1	210K 95071Y	43	5776A 11101K	48	Yes	N/A

Notes

Despite winning the autonomous bonus, we lost both middle goal zones and a long goal which caused us to lose this match.

- **Focus:** Explain what we learned and our redesign plans.

Date: Nov 24, 2025



Skills

Rank	Driver Attempts	Driver Highscore	Programming Attempts	Programming Highscore	Total Score
32	1	43	3	23	66

- We prioritized match-plays leading up to the tournament, hence why our programming routine was undeveloped compared to our match autonomous routines

Qualifications

Rank	Team	Name	W-L-T	WPs . APs . SPs
11	210K	Kawaii Kittens	7-1-0	14WP 60AP 189SP

- We lost Q5 partially due to a unlucky alliance pairing, along an autonomous loss
- We won Q71 allied with team 95071Y against the 2nd seed team, which is why we picked them for eliminations.

Summary

- Our autonomous skills routine had issues due to the field walls flexing beyond expected. We plan to adopt distance-based position resets in place of physical resets.
- Our mid-goal scoring geometry needs to be redesigned to improve its efficiency and prevent blocks from falling out.

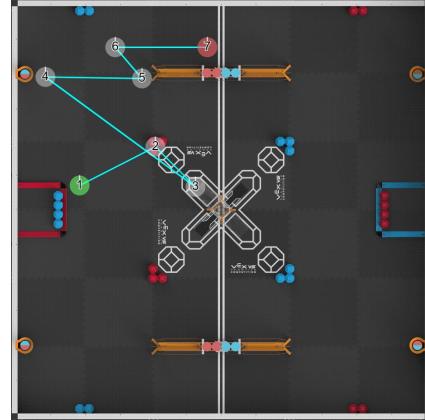
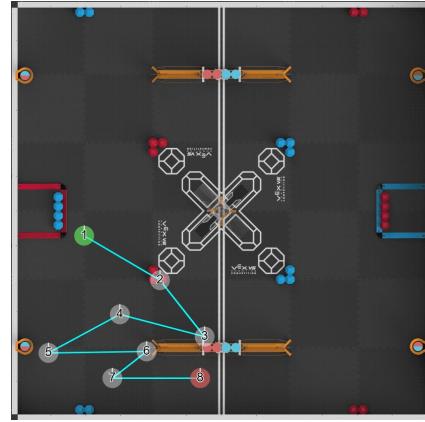
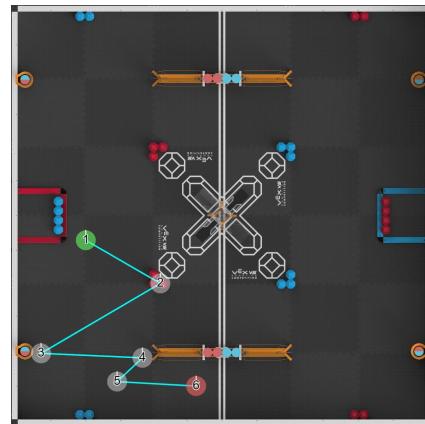
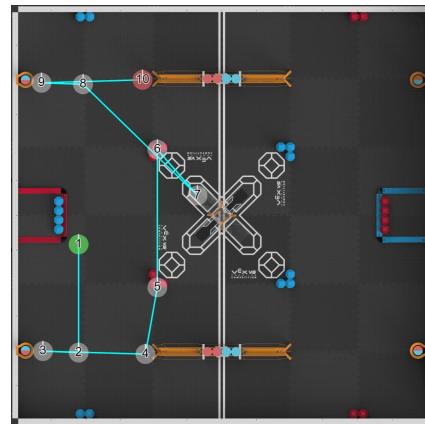
- ▶ Focus: Weighing effectiveness of different auton. routines.

Date: Nov 24, 2025

After UC Berkeley we refined our understanding of auto strategy. In qualifications, the only reliable option is a **Solo AWP**, since making it 100% consistent removes alliance variability and increases our chances of ranking high.

For eliminations, four autos are viable: **4+3**, **9-ball**, **7-ball wing**, and **4-ball wing**. Each secures the control zone and scores quickly, but they interact differently:

- **4+3** has the best alliance synergy. If our partner has a fast 7-ball (finishing with ~5 seconds left), it's better for us to start driver control by scoring mid rather than contesting the far control zone. This counters most 3-ball wing autos but depends more on alliance strength.
- **7-ball wing** is the fastest high-scoring option and guarantees control. Because it's quicker than 9-ball, it reliably counters it.
- **4-ball wing** is the absolute fastest but loses on scored-ball tiebreakers, so we rarely run it. Its main value is that it counters the 7-ball.



- ▶ Focus: Aligner for wings

Date: Nov 26, 2025

Wings

After attending our second signature event, we found that many teams had an aligner for their wings. This was often made with a trapezoid shaped plastic piece that is bent so they could glide along the long goal.

This new design was covered in 11101B's FUN interview at the Speedway signature event.

Another design was created by 2145 PSU. This design used a bent standoff and high strength axles. We found this design troublesome to make and too heavy to implement.



Figure 1: 11101B's wing aligner (FUN RECF, YouTube)



Figure 1: 2570R's wing aligner (VCad)

- Focus: Quick alignment when using wings

Date: Nov 27, 2025

Members Involved

Daniel

Objective

We created a wing aligner by bending a piece of polycarbonate using a vice into a trapezoid shape. This shape allows us to glide along the goal and align perfectly when using our wings. We attached the plastic to a C-channel for stability so the plastic does not warp. This was then attached to standoffs that mounted the aligner off our halfcut 3-wide.

Materials

- 1, 5 hole 1x2x1 C-channel
- Standoffs
- Plastic
- Screws

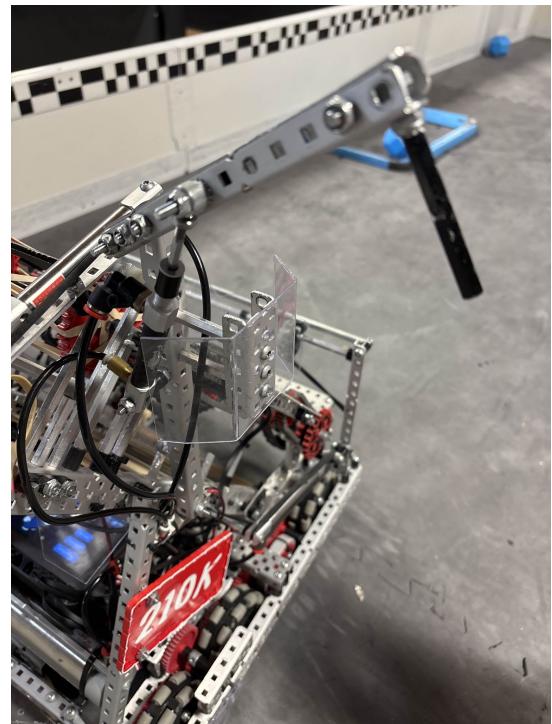


Figure 1: Our wing aligner mounted on our angle bar

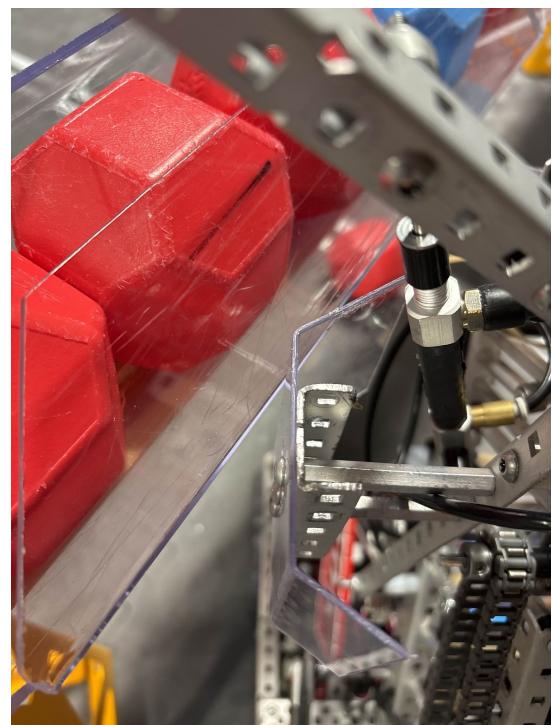


Figure 2: Wing aligner against the long goal

- **Focus:** Goals for a province-wide event with teams outside our city.

Date: Dec 12, 2025

In previous years, the STEM Innovation Academy Competition has been one of the larger and more competitive events within our region. This gives us a good idea of who will be competitive going into the 2026 split of the year. We plan to use this competition as a scouting opportunity and see what improvements we can apply to our robot after the tournament.

Skills Goals

Driver Attempts	Driver Highscore	Programming Attempts	Programming Highscore	Total Score
X	97	X	60+	157+

Based on our driver practice attempts, we've been able to score 97 points in practice. Our skills autonomous still has some variance in the precision of its movements, but we believe a minimum of 60 points is achievable based on recent performance.

Overall Goals

As this will be the last tournament that we compete in with this V2 robot design, we're mainly hoping to further develop our programming and refine our wing-based strategy. In matches and based on preliminary scouting, we believe that our wing will provide a significant advantage over the majority of other competitors, so we plan to utilize wings as much as possible. Some of our goals include

- Combined skills score of 157+
- Winning the tournament
- Ranking top 3 in qualifications
- Winning a judged award

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
		210R 14353S	3	45519J 210K	94		
	3					Yes	None

Notes

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
		210K 10053B	85	210E 3388X	6		
	16					Yes	None

Notes

- Solo Win Point hit, however alliance did not drive off the line

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
		45519X 45519B	3	5760A 210K	77		
	35					Yes	None

Notes

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
		45519H 210K	110	99197S 3300X	9		
	53					Yes	Yes

Notes

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
		45519G 221E	3	210K 99197V	88		
	66					Yes	Yes

Notes

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
		210K 3300D	3	3300J 3300G	77		
	76					Yes	None

Notes

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	R16	210Z 210K	136	36287A 45519A	11	Yes	None

Notes

The round of 16 match allowed us to fine tune our strategy after auto ends. Our main goal in this game was to score and prevent any chance of descoring.

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	QF	210Z 210K	97	45519C 45519D	11	Yes	None

Notes

We adapted our game strategy in this match as one of the teams had a rubber band release mechanism that had the capability to descore an entire tube.

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	SF	210Z 210K	75	1554P 45519X	12	Yes	None

Notes

The both red teams lost long goal control zones early into the match due to driver error, but through apply pressure onto the center goals we were able to win back the long goals and descore most blue blocks.

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?		
Win	F1-1	210Z 210K	80	210E 210C	12	Yes	None

Notes

After gaining a winning position after auton, we adapted our game strategy to prevent the opponents from scoring and holding onto our long goal control zones.

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?

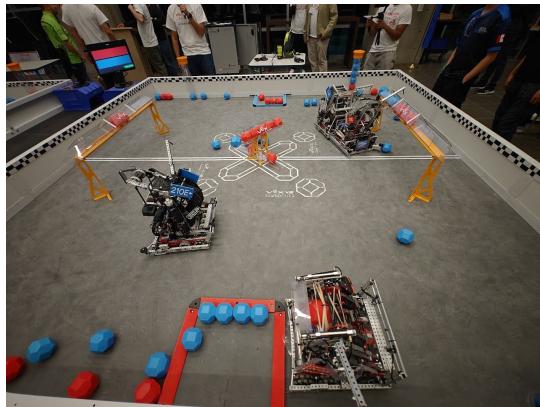
Notes

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?

Notes

► Focus: Results from STEM Innovation Academy

Date: Dec 13, 2025



Skills

Rank	Driver Attempts	Driver Highscore	Programming Attempts	Programming Highscore	Total Score
1	3	93	3	68	161

- One long goal control zone was missed during our driver skills run, bringing us from a potential 98 to 93pts
- We missed one control zone during our autonomous skills run, bringing us from a potential 73pts run to 68pts

Qualifications

Rank	Team	Name	W-L-T	WPs . APs . SPs
2	210K	Kawaii Kittens	6-0-0	14-60-35

Although we received two AWP's, we could have achieved more. AWP's were missed due to teammates being unable to move off the line and path inconsistencies from our own robot.

Summary

Overall, this tournament was very successful as we did not lose any matches. We achieved a skills score that placed us at top 38 in worldwide skills. We are very happy with this result as we secured three awards as a team and seven awards as an organization.

Moving onto our tournament next week, we are not looking to change anything with the current bot. Instead, we plan on starting planning and build on our January bot.

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
		210K 2088K	83	3300A 16688K	11		
	P1					Yes	None

Notes

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
		2088U 3388N	19	210K 3388A	65		
	4					No	None

Notes

- Auto missed two of the three goals

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
		45519X 210K	101	3388K 2088H	3		
	6					Yes	None

Notes

- Leaving the long goal early caused a last second descore by 2088H, this should be a reminder not to leave the long goal if there are many blocks scored in it.

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
		3388C 210R	16	2088C 210K	64		
	12					No	None

Notes

- We used the wrong autonomous routine, causing us to lose autonomous

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
		45519X 2088S	14	3388H 210K	73		
	16					Yes	None

Notes

- We had a losing position with only 40 seconds left. By successfully moving to less contested areas, we were able to descore opponent blocks

Win	Q#	Red Alliance		Blue alliance		Auton Win?	Auton WP?
		210Z 210K	61	210E 16688K	13		
	20					Yes	Yes

Notes

- One of the most competitive qualification matches of the tournament feature many strong teams. Although we lost a wing fight to 210E, our middle goal scoring and control made up for the deficit.

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?
Win	26	3300A 210K	64	3388H 2088K	29
				Tie	None

Notes

Auton tie, we were able to stay in a deadlock with 3388H to prevent a descore attempt to secure our long goal control zone.

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?
Win	QF	210Z 210K	124	2088C 2088S	8
				Yes	None

Notes

- 210Z tipped 2088S

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?
Win	SF	210Z 210K	94	2885K 2088U	3
				Yes	None

Notes

- Intake jammed during autonomous
- Used our wings to prevent opponents from scoring

Win	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?	
	F1	210K 210Z	91	210E 2088K	0	Yes

Notes

- A well played match and wing fights from both teams
- Applied constant pressure by descoring any threats

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?

Notes

	Q#	Red Alliance	Blue alliance	Auton Win?	Auton WP?

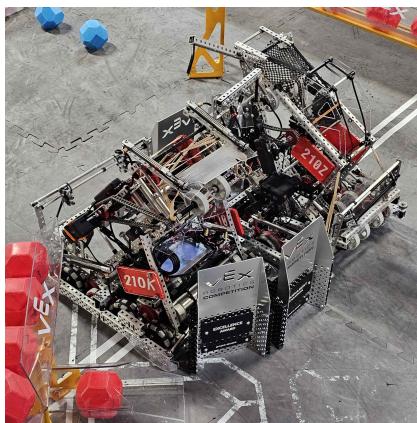
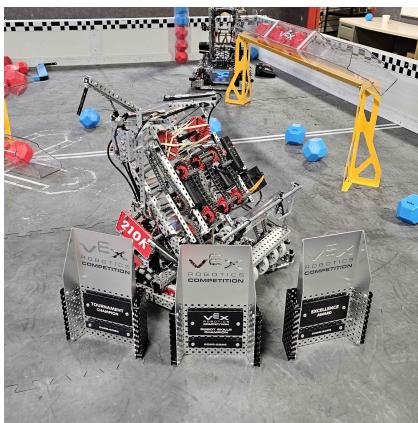
Notes

X'Mas Holiday Competition

Tournament Summary

- Focus: More relaxed tournament

Date: Dec 20, 2025



Skills

Rank	Driver Attempts	Driver Highscore	Programming Attempts	Programming Highscore	Total Score
1	3	86	2	64	150

- Due to a lack of practice, we were able to achieve a score of 86 with a 4 second stop time. The score could have been higher if long goal control zones were secured.
- The programming score was expected to be similar to our previous tournament and it ended hitting a 64pt score.

Qualifications

Rank	Team	Name	W-L-T	WPs . APs . SPs
2	210K	Kawaii Kittens	6-0-0	11-35-94

Our SAWP became very inconsistent even though no changes were applied to it. We aim to solve these issues moving into January.

Summary

We were very unhappy with our autonomous as it became very inconsistent. Resulting in only 35 AP and 0 SAWP hit. Our single AWP was given by 210Z's SAWP routine.

V3: Ignite the Northwest

- ▶ Problem: Key area to improve upon

Date: Dec 15, 2025

After attending three events with our robot. We have noted down a few of the key issues that could be improved on.

1. Funneling

One issue that we noticed primarily in skills was our funneling. When we would intake blocks in a specific order, due to our biasing and two wide intake. The blocks would not come out in the same order. This would sometimes cause control zones to be missed in skills.



Figure 1: Blocks scored in a random order due to funneling

2. Scoring

Another weak point of our robot is our scoring speed. The current scoring speed is at 400 RPM on bottom rollers, meaning the block being scored has no roller contacting the top of it, resulting in weaker scoring. The 400 RPM also limits our scoring speed, causing us to spend more time scoring during autonomous routines, decreasing the amount of time we have for the rest of the path.



Figure 2: Our robot is forced to cross the barrier from the back

3. Park Zone Crossing

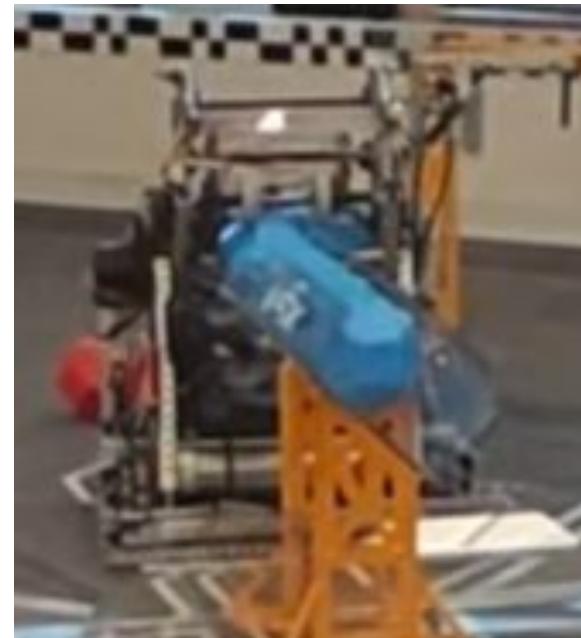
A lack of front sleds caused issues as we were unable to enter into the park zone with the front of our robot. This impact our skills routing as we could not intake the 6 blocks placed within the park zone. Only being able to enter the park zone in one direction limited both our skills and flexibility when parking as we would have to always enter the zone with the back of our robot.

- ▶ Problem: Key area to improve upon

Date: Dec 15, 2025

4. Middle Goal in Skills

Another issue we encountered in our skills runs was scoring 7 blocks into the upper center goal. Although it was achievable with our robot, it took too much time and was hard to align due to having a bottom roller. This bottom roller would give blocks an extra push that we did not need.



5. Loader

Although our loader was relatively successful, we found that a good match loader mechanism required a good intake working together for it to be effective. We also found that our rubber bands would wear out, causing the loader to deteriorate over time.

Figure 1: Struggling to score 7 blocks in the upper center goal

6. Jamming

Another minor issue we found was our intake system jamming. This occurred when we had 9 or more blocks in the intake. The main causes of this were our funnels and the 1.65" flex wheels on our intake.



Figure 2: Jamming occurring during semi-finals autonomous

- ▶ Focus: Different funnel options for a Ruiguan robot

Date: Dec 15, 2025

Looking at the VEX communities development throughout the year, we have found a few different methods for funneling on a two wide intake robot.

Standoff Funnels

The standoff funnels would use two standoff triangles that are on a rigid mount. This would in theory allow the blocks to funnel towards the center of the intake for scoring due to the angle of the triangles.

Fixed Plastic Funnels

The fixed plastic funnels uses a similar concept as the standoff funnels where the plastic acts as a funnel towards the center. These plastic pieces would be very stiff and screwed into place.

Flexible Plastic Funnels

The flexible plastic funnels would also use plastic, but would not use a screw mount. Instead, they can be ziptied into place to allow side to side flexing for the block to be pushed towards the center.



*Figure 1: Flexible plastic funnels
(1010G, YouTube)*

► Focus: Selecting the best funneling option

Date: Dec 15, 2025

After researching different funneling options on a two wide intake, we ranked all the different option using the following criteria

1. **Build Simplicity (1 - 5)** - How simple the design is able to be implemented
2. **Space Needed (1 - 5)** - Ranked of how much space there is needed to add the funnels
3. **Funneling Ability (1 - 5)** - Ranked on how well the funnels are able to direct blocks into the center

<u>Criteria</u> <u>Options</u>	Build Simplicity	Space Needed	Funneling Ability	Total
Standoffs	2	3	1	6
Fixed Plastic	3	3	2	8
Flexible Plastic	3	2	4	9

After considering all options available to us, we decided to implement the flexible plastic design as it would help with our bottom goal scoring and improve our funneling for the long goal.

- ▶ Focus: Methods for scoring on the upper midgoal

Date: Dec 15, 2025

Development throughout our own testing and from other teams. Many different middle goal scoring options were created.

Piston Pull-down

This design features a piston that pulls down a section of the rubber band ramp, creating an opening for blocks to fall through into the middle goal.



Figure 1: 1698V's piston pull down opening their ramp

Double Roller

This design uses both a top and bottom rubber band roller to score blocks into the upper middle goal.



Figure 2: 1474K's trap door mechanism with a flex wheel roller

Trapdoor Design

One different design that we discovered by 14749K is a trapdoor roller hybrid design. The trapdoor can be opened with a piston and allows the blocks to be scored into the upper center goals with the roller.



Single Counter Roller

This is the design we currently have, consisting of a single flex wheel roller that acts as a counter roller for long goal scoring and a regular roller for the middle goal.

Figure 3: 9189X's double roller

- ▶ Focus: Selecting the best option for middle goal scoring

Date: Dec 16, 2025

Based off of the four designs we have researched, we used a design matrix to rank each based on the following criteria.

- Build Simplicity (1 - 5)** - Ranked on how easy the design is able to be implemented
- Scoring Speed (1 - 5)** - A general ranking on how fast each design is able to score blocks
- Viability in Skills (1 - 5)** - Can the design score 7 blocks in the upper center goal?
- Viability in Matches (1 - 5)** - Ranked based on whether or not the design has fast scoring and potential to descore opponent blocks.

<u>Criteria</u> <u>Options</u>	Build Simplicity	Scoring Speed	Viability in skills	Viability in matches	Total
Piston Pulldown	5	3	3	2	13
Double Roller	2	4	3	3	12
Single Counter Roller	4	4	1	3	12
Trapdoor	2	4	4	4	14

Based on our rankings, we found that the trapdoor roller design would best fit our needs for both for regular matches and skills matches. It will have a balanced scoring speed that can be adjusted depending on our needs.

- ▶ Focus: Researching methods to descore the middle goals

Date: Dec 16, 2025

There have been three main methods of middle goal descore developed in the past months.

C-Channel Stick

The simplest design where a C-channel is mounted on the side of the robot which is used to ram into the middle goal for descoring.



Figure 1: 1698V's piston activated middle goal descore.

Front Descore

A design developed by 886Y where a hood is mounted on a C-channel on the front of their robot

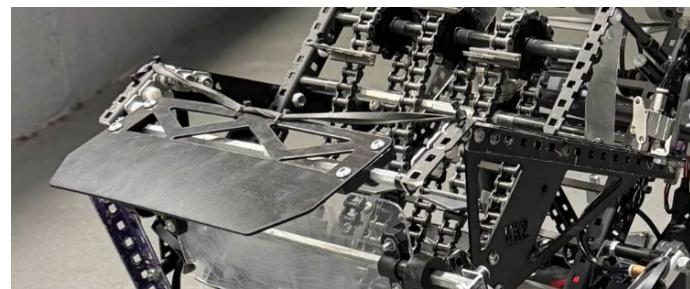


Figure 2: 886Y's descore mech mounted on the front

Piston Activated Descore

A design found in many Northern California robots. This design is piston activated and uses a C-channel or standoff to descore the upper center goals.



Figure 3: 2145Z's C-channel for middle goal descore

- ▶ Focus: Selecting the best method for descoring midgoals

Date: Dec 16, 2025

- Build Simplicity (1 - 5)** - How simple the design can be implemented
- Weight (1 - 5)** - How much weight the design adds
- Functionality (1 - 5)** - How well the design can descore the middle goals
- Driver Preference (1 - 5)** - How comfortable we are with using each design

<u>Criteria</u> Options	Build Simplicity	Weight	Functionality	Driver Preference	Total
C-Channel Stick	5	3	3	1	10
Piston Activated	2	2	4	2	10
Front Descore	4	4	4	4	16
Standoff Sticks	5	5	1	1	12

After considering all designs, we decided to implement the front descore. We also wanted to implement a descore method from the back so we plan to design a descore method that is incorporated with our scoring mechanism.

- ▶ Focus: Highlighting our design prior to building

Date: Dec 16, 2025

After extensive research and consideration, we have developed a new ideal robot design for this competition moving forward. This design aims to **address the issues** identified in our **previous robot** while keeping the **core system** simple and familiar. Rather than reinventing the entire mechanism, we focused on correcting inherent flaws within the existing design.

We have opted for a two-motor intake system that runs consistently throughout the robot, improving **reliability** and **control**. In addition, we chose a hybrid mid-goal scoring mechanism with the descorer integrated directly into it for **increased convenience** and mechanical simplicity. For object guidance, we decided to use flexible plastic funnels alongside two motor intake throughout all rollers, which allow for **smoother ball** flow while maximizing plastic usage to reduce weight and complexity.

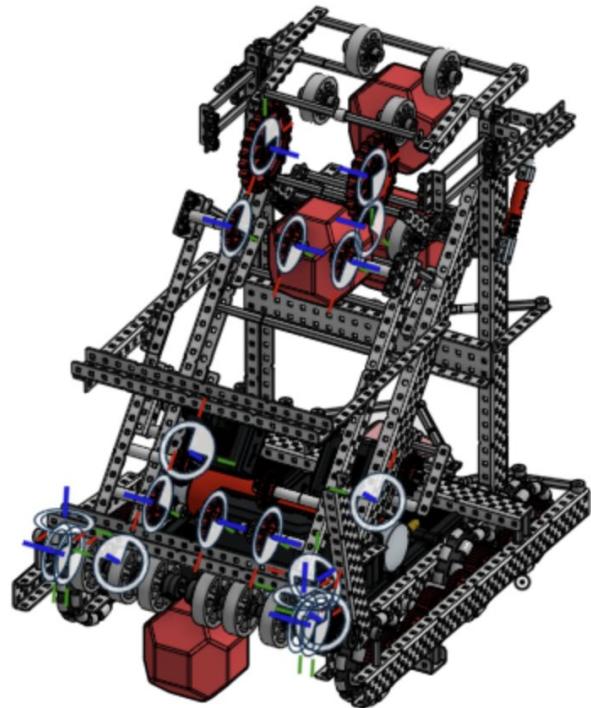


Figure 1: Front view of our CAD

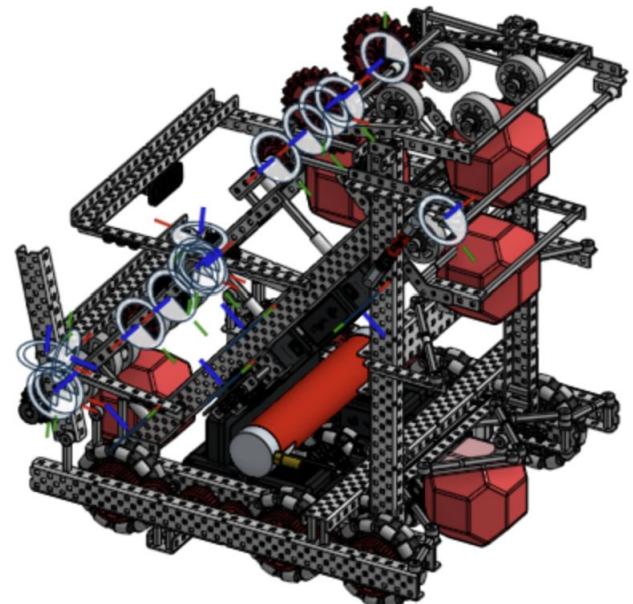


Figure 2: Back View of our CAD

- ▶ Focus: Building our drivetrain frame

Date: Dec 17, 2025

Members Involved

Brandon

Objective

The objective was to start building the CAD for the new robot, completing as much as possible. We also attempted to ensure all the c-channels and everything maintained as straight as possible.

Materials

- 4, 27 Hole 1x2x1 C-Channels
- 4, 30 Hole 1x2x1 C-Channels
- 2, 1 Hole 2x2x2 C-Channels
- 1, 35 Hole 1x3x1 C-Channel (Temporary)
- Screws
- Spacers
- Boxing Standoffs (0.875")
- Bearings
- Zipties

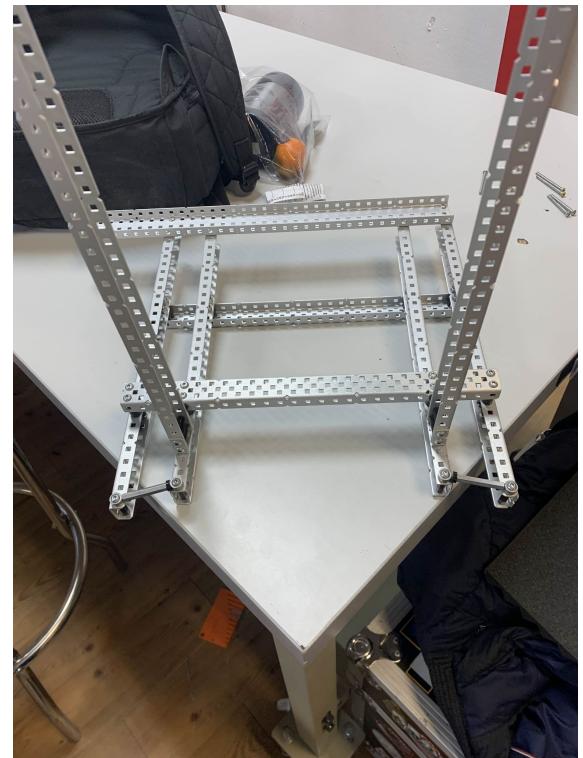


Figure 1: Frame from the back

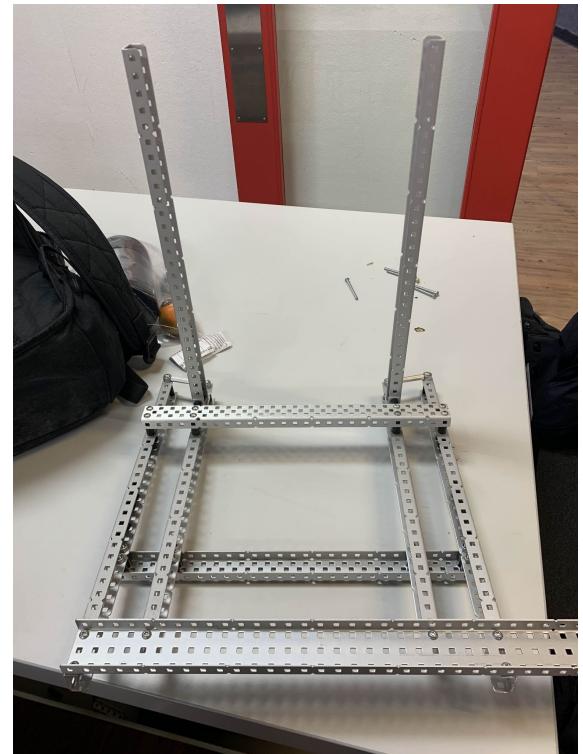


Figure 2: Frame from the front

- ▶ Focus: Building our drivetrain frame

Date: Dec 17, 2025

Problems and Solutions

One issue we encountered was planning out our funnels.

We planned to use a cut U-channel instead of standoffs because they would provide more structural support

Next Steps

We plan to build our intake structure next.



Figure 1: Side view of our squared fram



Figure 2: Side view of our drivetrain

- ▶ Focus: Building our intake structure

Date: Dec 18, 2025

Members Involved

Brandon

Objective

Our objective is to create structure and the frame for our intake system. This structure must be rigid and braced as it will make up the majority of our robot.

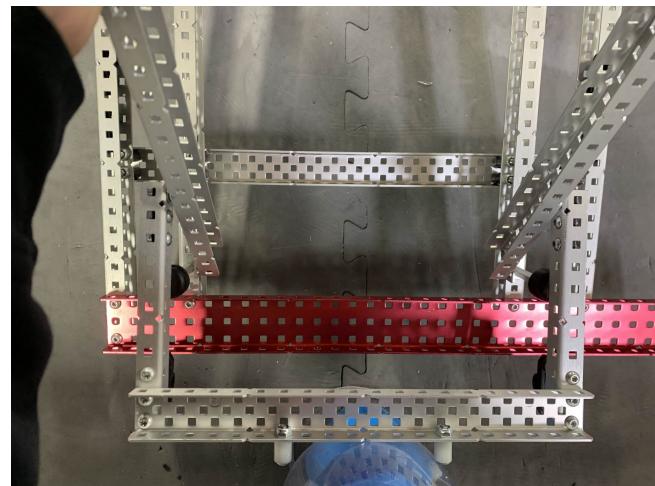


Figure 1: Front view of the bottom stage

Materials

- 4, 27 Hole 1x2x1 C-Channels
- 4, 30 Hole 1x2x1 C-Channels
- 2, 1 Hole 2x2x2 C-Channels
- 1, 35 Hole 1x3x1 C-Channel (Temporary)
- Screws
- Spacers
- Boxing Standoffs (0.875")
- Bearings
- Zip ties

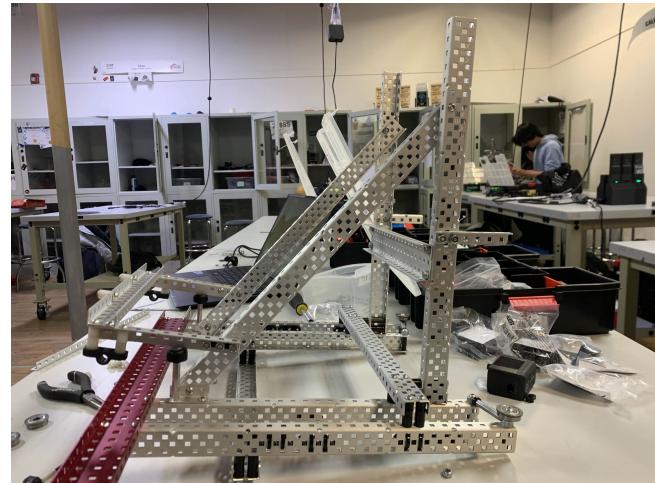


Figure 2: Side view of intake structure

- ▶ Focus: Building our intake structure

Date: Dec 18, 2025

Problems and Solutions

Ensuring everything is straight and rigid is one of our main priorities.

To make this happen, we added the main braces first before continuing with anything else. The 3 wide is added as an indicator where our middle goal scoring will occur and serves as a brace for our vertical C-channels.

We were worried that our vertical C-channel was going to bend.

To ensure our C-channel structure is parallel, we used shoulder screws on our structure to ensure no bends happen.



Figure 1: Top view of our intake structure

Next Steps

Finish the structure by adding our angle bars where we will mount of rollers. We then want to polish the drivetrain by adding minor mechanisms such as funnels and wall riders. Then move onto creating rollers.

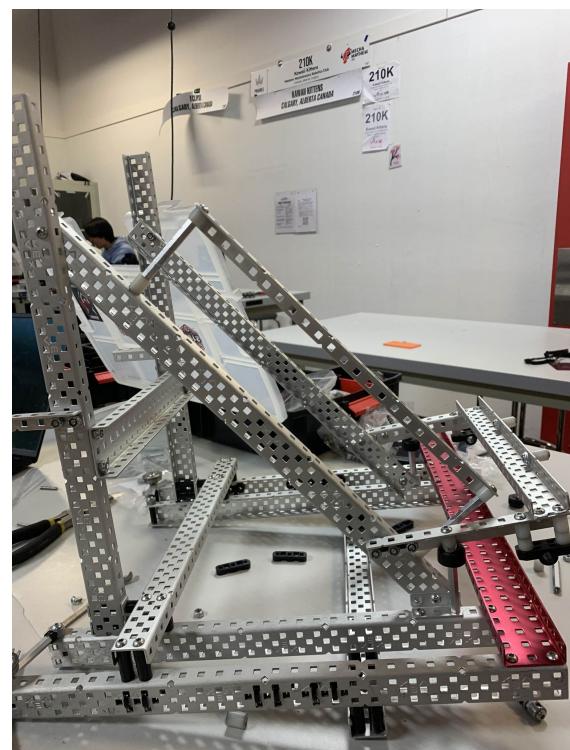


Figure 2: Another side view of our intake structure

- ▶ Focus: Continuing our intake structure by building the roller mounts

Date: Dec 19, 2025

Members Involved

Brandon

Objective

The objective is to finish the intake structure and prepare to mount our rollers and converter belts for the actual intake system.

Materials

- 2, 21 hole 1x1 Angle Bars
- 4, 3 hole 1x1 Angle Bars
- 1, 19 hole 1x3x1 C-Channel
- 2, 6 hole 1x1 Angle Bars
- 2, 9P, 24T Sprockets
- 2, 6P, 16T Sprockets
- Low Strength Axle
- Bearings
- Standoffs
- Spacers
- Axle Collars
- Screws
- Nylock Nuts

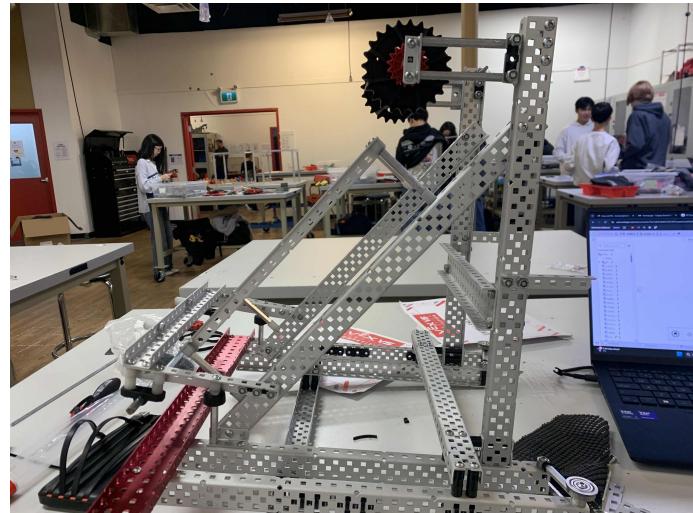


Figure 1: Side view of the mounted angle bars

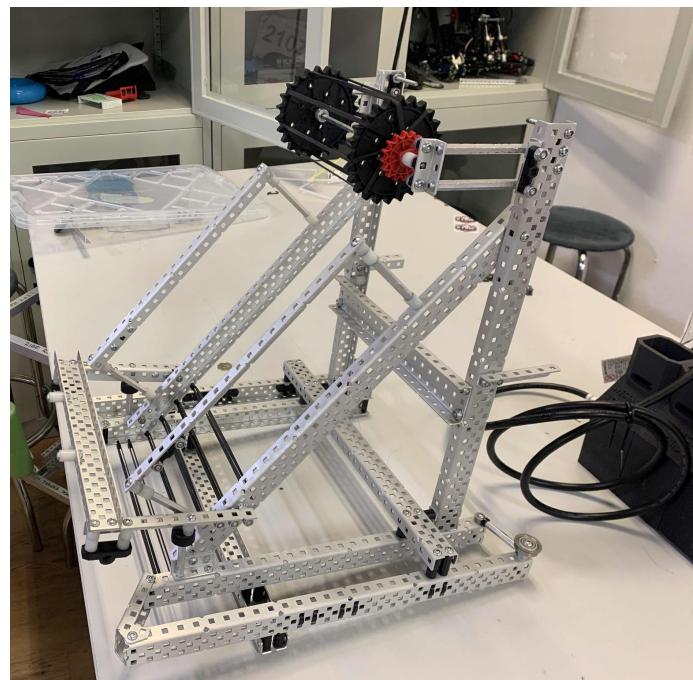


Figure 2: View of our roller and structure

- Focus: Continuing our intake structure by building the roller mounts

Date: Dec 19, 2025

Problems and Solutions

Making sure that our roller mount was straight was a challenge when building it.

We used an axle to test free spin and attached a temporary angle bar to make sure everything lined up

When testing with our first roller, we noticed that it would get caught and stop spinning

This was found to be caused by over tightening our shaft collars. This was solved by giving the roller move horizontal movement.

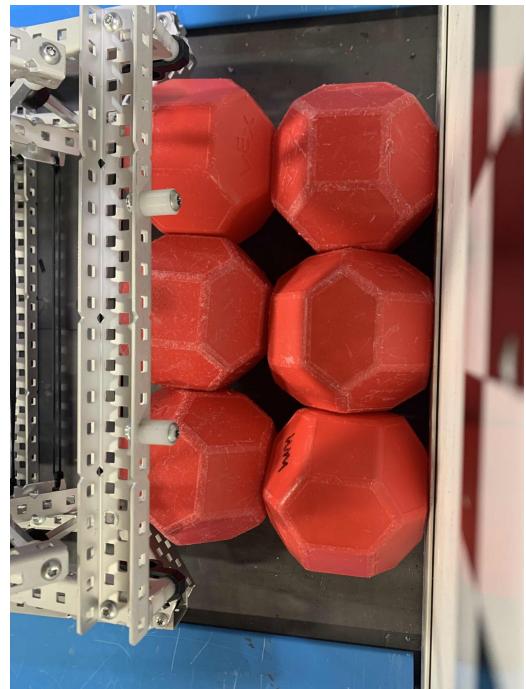


Figure 1: Intake setup for skills

Next Steps

Finish small adjustments including funnels and wall riders before moving onto powering our intake system.

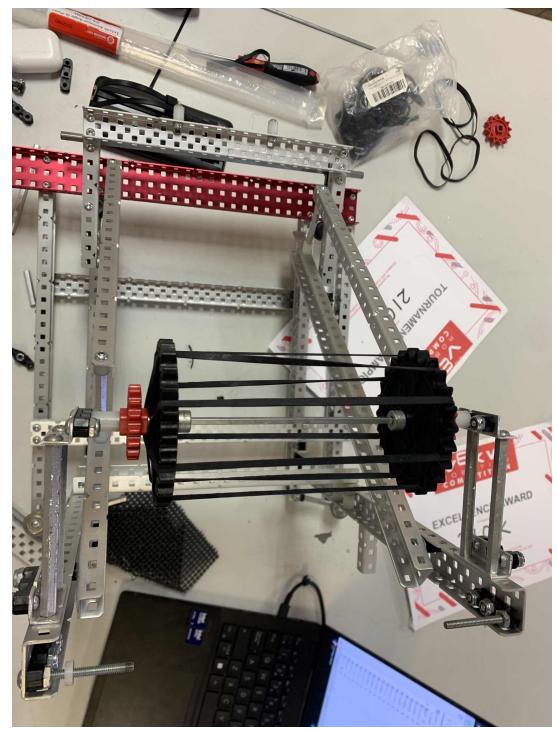


Figure 2: Top view of our rubber band roller

- ▶ Focus: Funnels and wall riders

Date: Dec 20, 2025

Members Involved

Brandon

Objective

Polish the drivetrain by adding small quality of life attachments.

Materials

- 2, 5 hole 2x2x2 U-Channel, custom cut
- 2 ball bearings
- Screws
- Nylocks
- Spacers

Next Steps

Start creating our hood structure and frame for scoring flex wheel rollers.

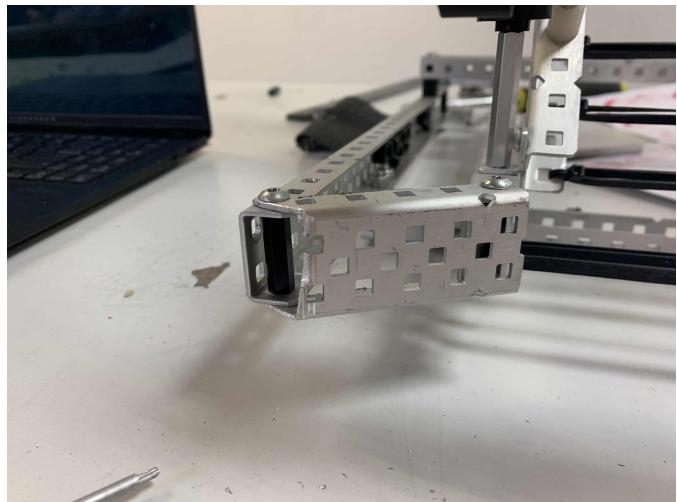


Figure 1: U-Channel Funnel

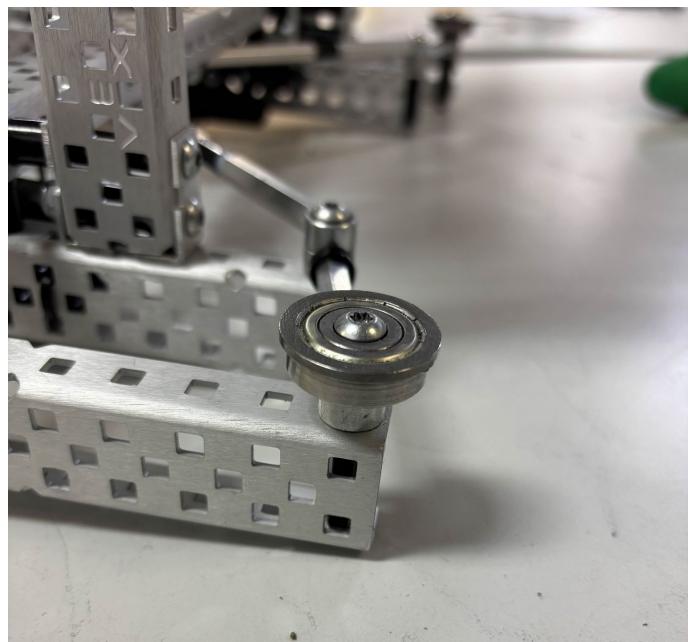


Figure 2: Ball bearing wall rider

- ▶ Focus: Starting to build the hood structure

Date: Dec 20, 2025

Members Involved

Brandon

Objective

Create a “double hood” setup where our scoring flex wheels will be attached. This will help us maintain contact with the block for more time when scoring, allowing for more power.

Materials

- 4, 6 hole 1x1 angle bars
- Spacers
- Standoffs
- Bearings
- Screws
- Keps Nuts
- Nylocks

Next Steps

We plan to attach chains for the intake system before completing the hood with flex wheels.



Figure 1: Screw joint mount fo the hoods



Figure 2: Back view of the hood setup

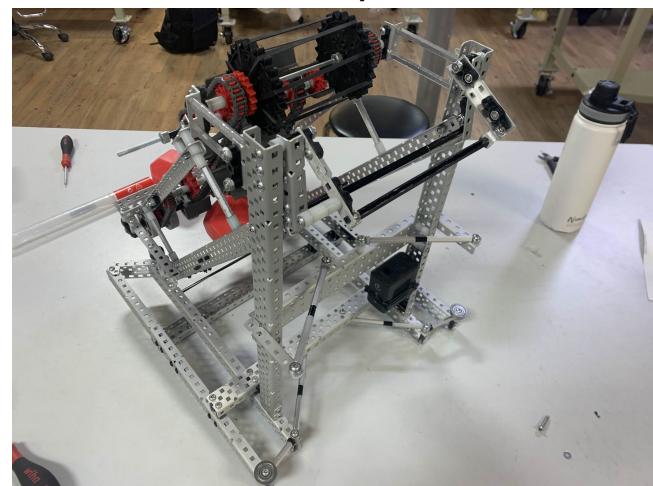


Figure 3: Another view of the hoods

- Focus: Building the first stage and chaining the system

Date: Dec 20, 2025

Members Involved

Brandon

Objective

Building the first stage and attaching chains to test intake system friction.

Materials

- 6, 2 inch flex wheels
- High strength Axle
- 2, Low Strength Axles
- 8, 6P, 16T Sprockets
- 6, 9P, 12T Sprockets
- 9P chain
- 6P chain



Figure 1: Front view of our chained intake

Problems and Solutions

We noticed that some of the chain was not lined up.

To solve this, we adjusted the spacing around each sprocket to ensure everything was straight.

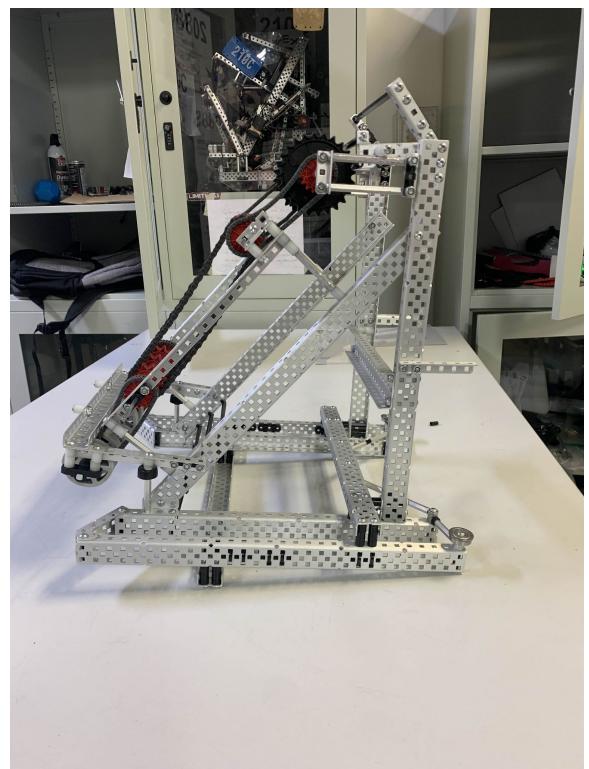


Figure 2: Side view of our intake system

- Focus: Powering the system and finishing the hood

Date: Dec 20, 2025

Members Involved

Brandon

Objective

We plan to complete our intake system by attaching motors to power the system. We also plan to complete our intake system by attaching the flex wheels.

Materials

- 2 Motors
- 2, 5 Hole, 1x1 Angle Bars
- 4, 6P, 16T Sprockets
- 4, 6P, 8T Sprockets
- 6P Chain
- Low Strength Axles of Various Length
- Bearings
- Screws
- Nylocks
- Axle Collars

Problems and Solutions

With our current motor setup, we use a long length of chain which is prone to snapping.

We plan to move our motor mount elsewhere to shorten the chain and prevent any snapping.

Next Steps

Move our motor placement and start building our middle goal contraption.



Figure 1: Side view of our motor powering the intake

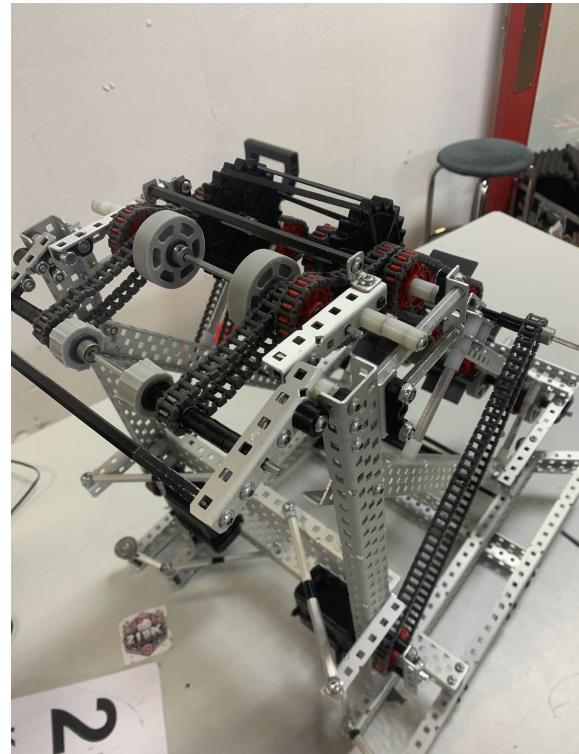


Figure 2: Top view of our completed hood

210K



Kawaii Kittens

Calgary, Alberta, Canada

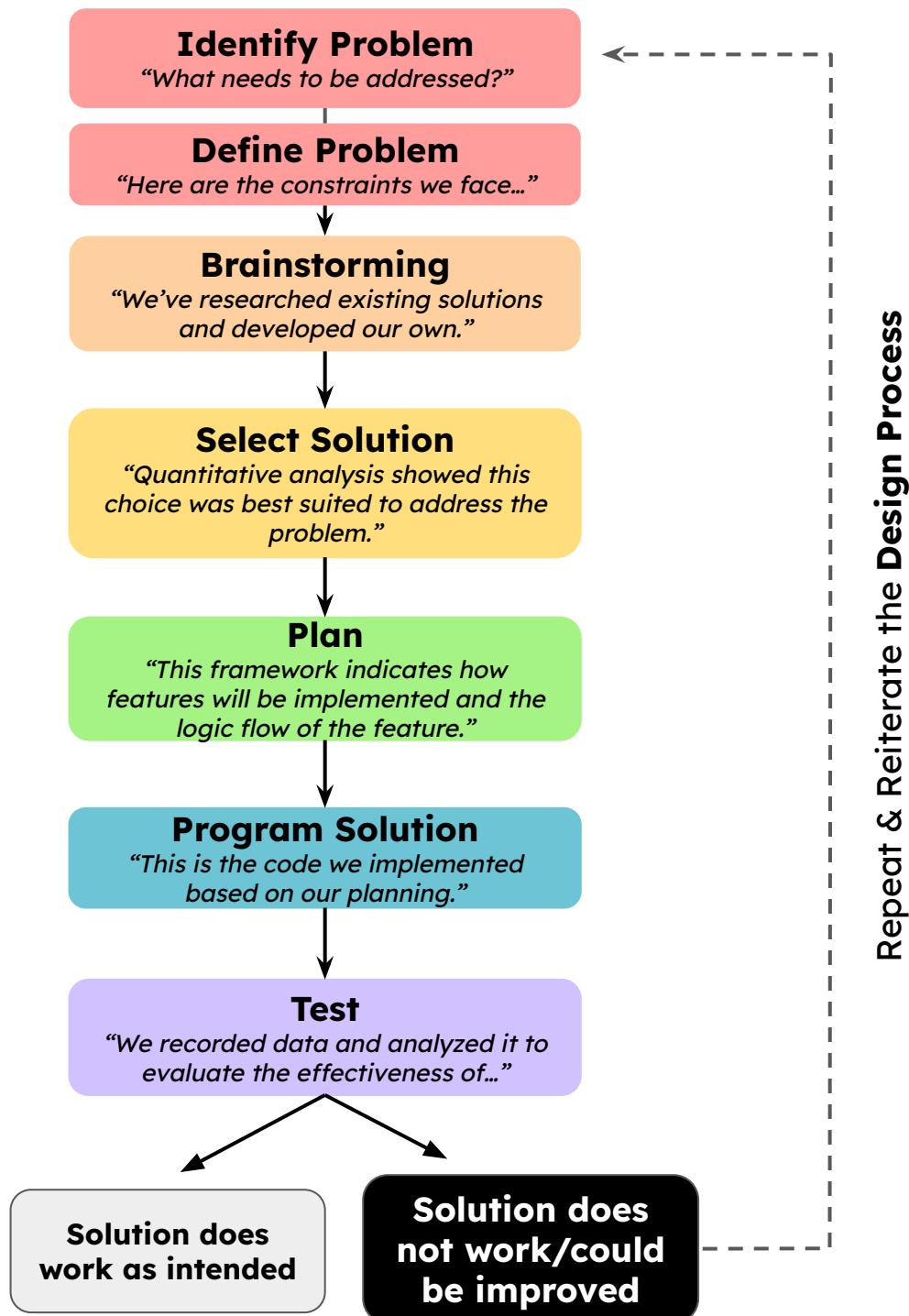
Digital Programming Notebook



The following is a diagram of the **Programming Design Process** that we will be using throughout the year, visualized with the colour coded categories for reasons discussed on the page beforehand.

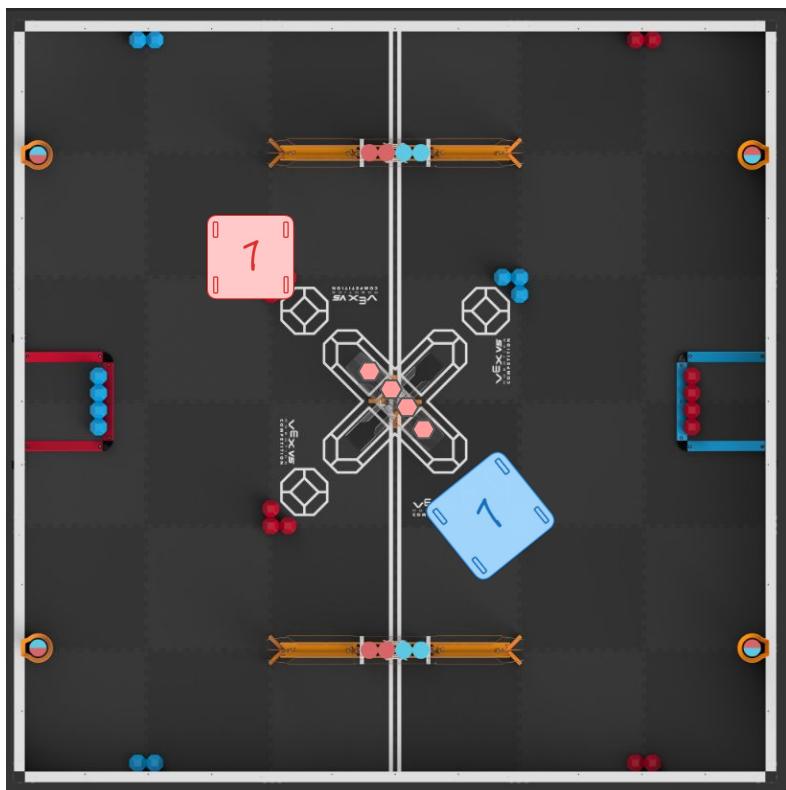
While the system mostly goes the route of the arrows traversing linearly, at any time we will have to revert stages or skip them depending on issues we face with our design implementation.

Programming Design Process Diagram



We have already analyzed the mechanical and strategical considerations for this year's game, Push Back, in pages 16-23 of our engineering logbook. We also wanted to identify constraints related to the game setup that would affect how we develop and test our autonomous programming processes **for the whole season**.

- 1) During autonomous, opposing robots can legally interact with each other through scoring



- Presume that **red robot 1** filled up the mid-high goal with red blocks during autonomous
- **Blue robot 1** could then fill up that goal with blue blocks, completely **negating** red's scoring
- If **red's auton** was setup to gain an AWP, **blue's** autonomous would have **blocked red's AWP**
- Takeaway: even a consistent autonomous is not guaranteed to score AWP

- 2) Distance sensors could be used for detecting the position of blocks in an intake, and for localizing the position of the robot on the field.



Figure 1: Distance Sensor via VEX KB

The distance sensor is a “time-of-flight” recorder that shoots out a low-energy laser beam from its window. By calculating the time it takes for the beam to bounce back to the sensor, it can determine an object’s distance.

- This feature can also be used as a “tripwire”

3) Optical sensors could be used to sort blocks based on colour, as we did with last season's game object:

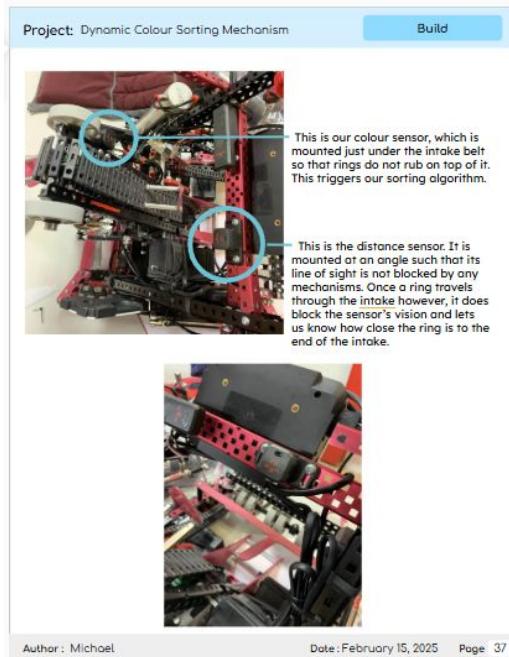


Figure 1: Our previous season's logbook featuring colour sorting

- The optical sensor quantifies the colour of different objects, allowing us to differentiate red blocks and blue blocks

- Based on the colour detected, the code can be set to trigger certain logic loops and perform hardware actions (e.g. stop intake, etc)



Figure 2: Optical Sensor via VEX KB

4) The **rotation sensor** intuitively tracks the instantaneous rotation of an axle/shaft with +/- 0.5° precision

- Stores the last recorded position of rotation even after powering off the brain
- Use cases include lifts and other rotating mechanisms such as wheels
- Rotation sensors can be attached to unpowered wheels that track the lateral movement of the chassis



Figure 3, 4: Rotation sensor applications via VEX KB

There are numerous “software development environments” available for programming in VEX. Each SDK is maintained by a different developer, offering unique features with their own caveats. The following is our own comparison of the most popular options:

Library	Pros	Cons
VEXCode (made by VEX)  VEXcode V5	<ul style="list-style-type: none"> Officially maintained by VEX; fastest to receive updates for new hardware Lots of documentation and support Guaranteed to be bug-free 	<ul style="list-style-type: none"> Closed-source and lack of access for low-level commands Lacks two-way communication, robot cannot send data to computer Documentation is thorough but undescriptive
PROS (Purdue Sigbots) 	<ul style="list-style-type: none"> Most popular within the VRC community Supports all current electronics parts Allows for serial console debugging Supports advanced graphics via libvgl 	<ul style="list-style-type: none"> Some complex features use industry-level implementations, which have a steep learning curve
vexide 	<ul style="list-style-type: none"> Supports near-instantaneous uploading Actively supported Resistant to memory leaking Planned support for simulating code 	<ul style="list-style-type: none"> Steeper learning curve to learn the syntax of the Rust language Youngest of the available choices Lacks support for many hardware components
vex-rt (QUEENS) 	<ul style="list-style-type: none"> Based on the PROS kernel, Resistant to memory leaking 	<ul style="list-style-type: none"> Last updated in 2023; “abandoned” by creators

Based on the pros and cons described previously, we'd also like to share our **qualitative reasoning** to explaining the rankings:

Rank	Library	Reasoning
1st	PROS (Purdue Sigbots) 	<ul style="list-style-type: none"> ✓ Best support for sensor-based programming & accessing low-level sensor controls ✓ We already know how to program in C++ ✓ Most popular within the VRC community; most programmers use PROS ✓ Better debugging support for troubleshooting via serial terminal ✓ Support for external graphics libraries allows us to create data visualization interfaces
2nd	vexide 	<ul style="list-style-type: none"> ✓ vexide is written in Rust, a memory-safe language that prevents memory-related crashes which C++ does not ✗ However, we'd have to learn a new language as Rust has a different syntax from C++ ✗ Extremely young, does not have widespread adoption and we cannot trust it is competition-ready
3rd	VEXCode (made by VEX)  VEXcode V5	<ul style="list-style-type: none"> ✓ We already know how to program in C++ ✓ Extremely intuitive syntax ✗ Documentation is thorough but does not provide examples of code usage ✗ Difficult to access low-level threading controls or memory access
4th	vex-rs (QUEENS) 	<ul style="list-style-type: none"> ✗ Does not support new hardware, such as 5.5W motors or the AI vision sensor ✓ vex-rs is written in Rust, a memory-safe language that prevents memory-related crashes which C++ does not ✗ However, we'd have to learn a new language as Rust has a different syntax from C++

Thus, we have decided to continue with **PROS** as our library of choice.

We use GitHub as part of our software development process. [GitHub](#) is a super helpful code sharing/hosting platform, because it allows us to collaborate between people to implement **different pieces of code simultaneously**. It's also helpful when we can write the code on our computers at home to **speed up the development process**, rather than being restricted to only writing code when we are working on the bot in-person.

Being hosted on GitHub, our code is also **openly accessible** to other VEX teams. This makes it easy for other teams to take inspiration from our framework and find success in programming for their own bots.

Thanks to this workflow, our code is available publicly at
<https://github.com/NoozAbooz/210K-PushBack-2026> and in PDF format at
<https://210k.westernmech.ca/docs/>.

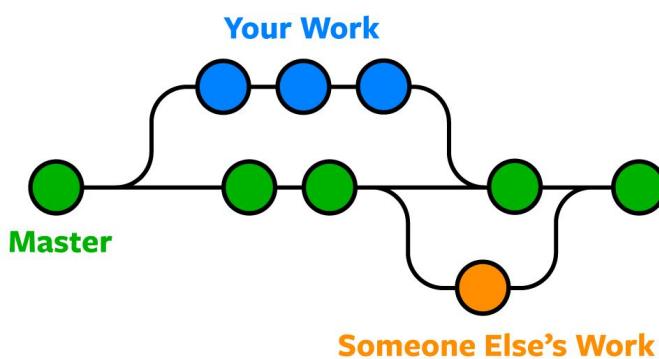
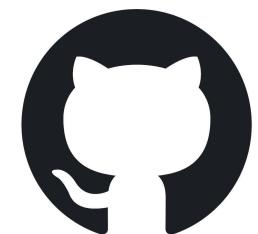


Figure 1:
[https://anaalahrech/why-using-git-and-how-14bd34386752](https://anaelahrech/why-using-git-and-how-14bd34386752)



GitHub

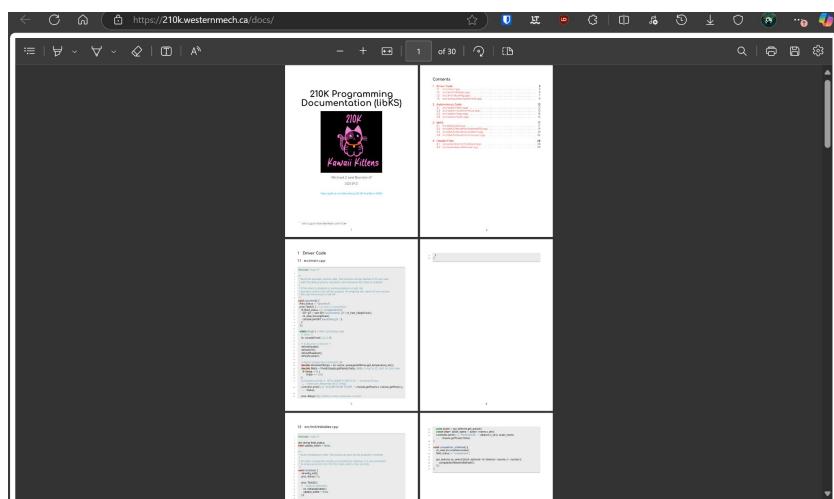


Figure 3: Screenshot of our code documentation

GitHub is an extremely complex version control software that serves its function best. when used effectively. The core of git revolves around its “branching” feature. Imagine the history of your codebase as a linear tree. Branches may “grow” off a point in the main branch’s history and continue to coexist with its own changes.

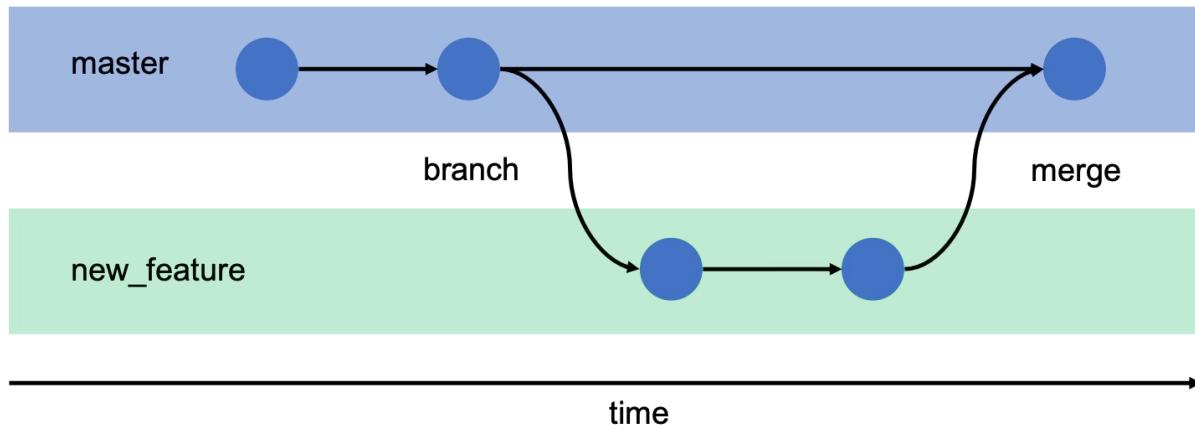


Figure 1: A sample git project history via Atlassian.

Use Cases

Branching can be used for:

- Separating codebases into major revisions (v1, v2, etc)

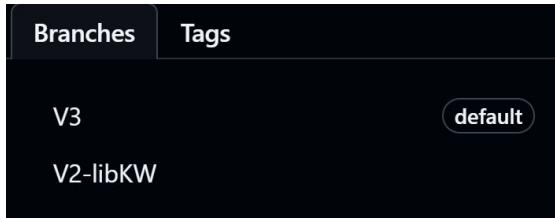


Figure 2: Our branch structure from last season

- Working on new features while preserving old code (branching **v2-new-movement** off the primary **v2** branch)

Commands

- Git is used by instructing it with various commands
- The following are some important commands we plan on using often:
 - 1) > `git add -A`
Adds every file to a list of those that will be tracked by git
 - 2) > `git commit -m "msg"`
Labels changes to multiple packages into one package that represent a feature
 - 3) > `git push`
Uploads commits to the cloud server
 - 4) > `git pull`
Downloads remote commits

PROS is a third party SDK developed by Purdue University (VEXU team BLRS) in Indiana. Having conducted deeper research on the toolkit, we've identified that PROS also features a **faster refresh rate**, and just overall better efficiency and **community support**. While code in Vexcode V5 updates around the 20-50ms range, PROS updates at approximately 10ms. Although this change may seem small, it can provide extremely drastic results while in game, as we have many things running simultaneously such as Driver Control, Odometry, etc. PROS also offers unprecedented **direct hardware access** and **multitasking** scheduling.



PROS: Documentation Home

Welcome to the PROS Documentation!

If this is your first time using PROS, it is recommended that you check out one of the **Getting Started** tutorials:

Getting Started

For topical tutorials on everything from the **ADI (3 Wire Ports)** to **Wireless Upload and Hot/Cold Linking**, check out the **Tutorial** section:

Tutorials

And for documentation on using the PROS **API**, see the **API** section:

API Home

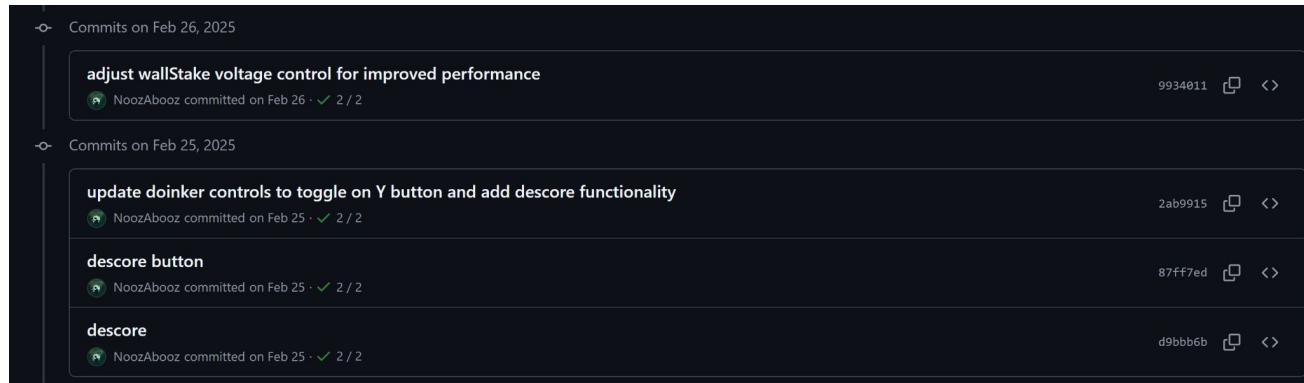
We're proud to support **OkapiLib**, a library designed to make it easier to incorporate complex functionality in your PROS project. For documentation on the latest OkapiLib version, go here:

Latest OkapiLib documentation

Additional features of FreeRTOS that are intended for **advanced users** can be found in the **Extended API**. Tutorials on these features can be found in the **Extended** section:

Figure 1: PROS documentation via <https://pros.cs.purdue.edu/v5/>

PROS integrates heavily into the robot's code, because Vexcode was too basic for our needs and PROS allows us to use development programs we already have experience with, like VS Code. Tools like git/GitHub also store **change histories** and backs up the code to the cloud, which can be helpful if Brandon and I experience technical difficulties, or need to **roll back** our code to a previous version in case we break something. (this happens very often in coding, especially robotics).



The screenshot shows a GitHub repository's commit history for the 2024-2025 season. It is organized into two main sections: 'Commits on Feb 26, 2025' and 'Commits on Feb 25, 2025'.
Commits on Feb 26, 2025:

- adjust wallStake voltage control for improved performance** (commit ID: 9934011) - NoozAbooz committed on Feb 26 · 2 / 2

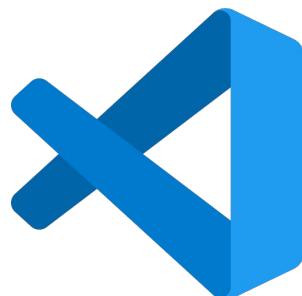
Commits on Feb 25, 2025:

- update doinker controls to toggle on Y button and add descore functionality** (commit ID: 2ab9915) - NoozAbooz committed on Feb 25 · 2 / 2
- descore button** (commit ID: 87fff7ed) - NoozAbooz committed on Feb 25 · 2 / 2
- descore** (commit ID: d9bbbb6b) - NoozAbooz committed on Feb 25 · 2 / 2

Figure 1: A screenshot of our code history from the 2024-2025 season.

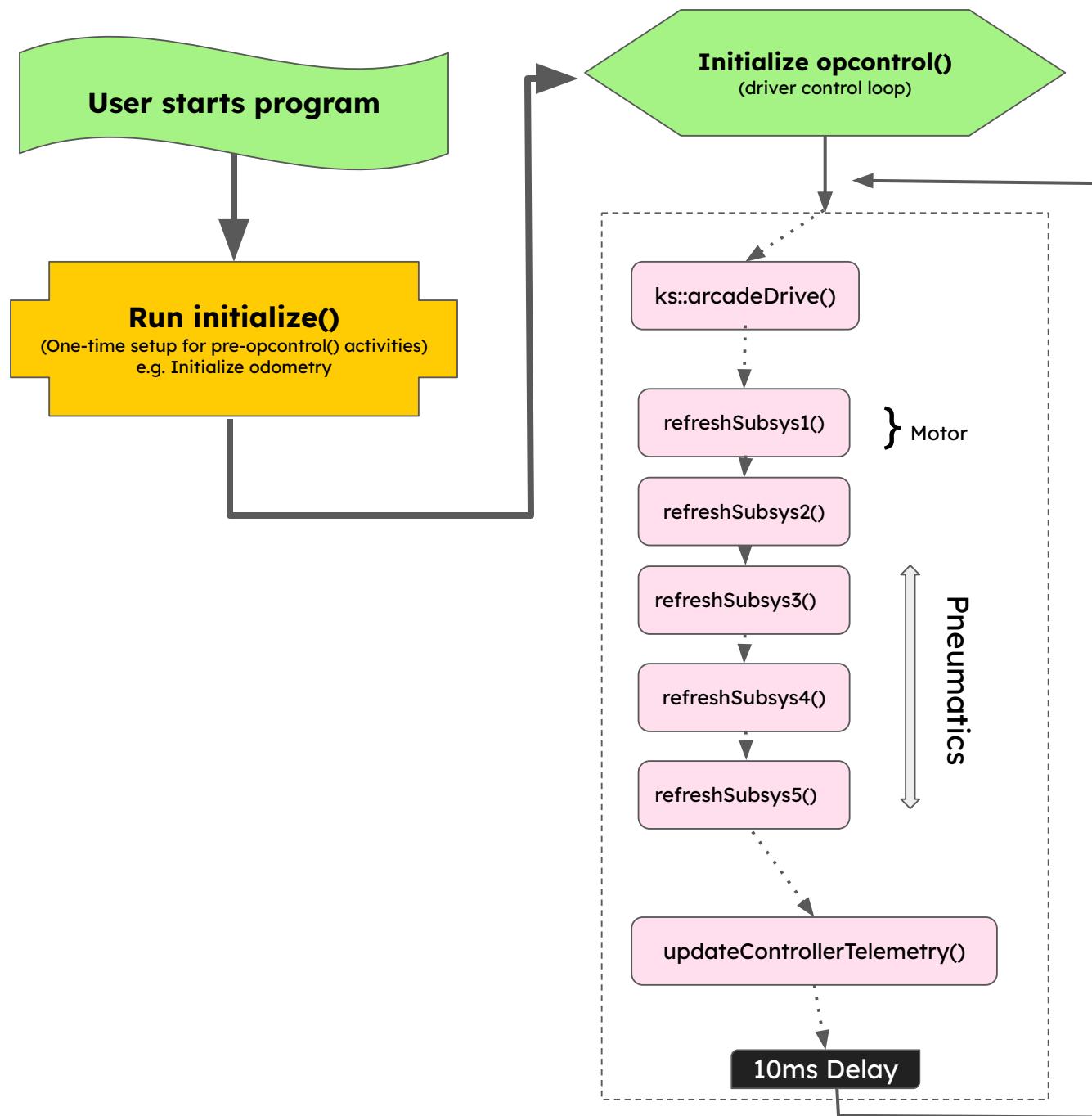
In addition to migrating from VEXCode many years ago, we also switched to using voltage in our motors instead of velocity. This is because velocity has a default PID provided by VEX. However, when we run our own custom PID, it results in the two PIDs overlapping, which can make certain movements inaccurate. True raw voltage **allows us to be more accurate with more range of customization**. ~~More details on our usage of voltage and our reasoning will be addressed in later entries when we discuss our driver control code.~~

RETROACTIVE EDIT: Raw Voltage Control is detailed in pg. 26



Based on our experience in previous seasons, we think it's important to plan out the control flow of logic within our main program.

In PROS, all of the driver-controlled code operates within the opcontrol() function in main.cpp—inside that, we intend to set up a host of run <subsystem>() functions that encapsulate all driver functionality for each component within said function. These are run sequentially in a infinite loop until the program is exited. We started by diagramming this with a control-flow flowchart, as shown below:

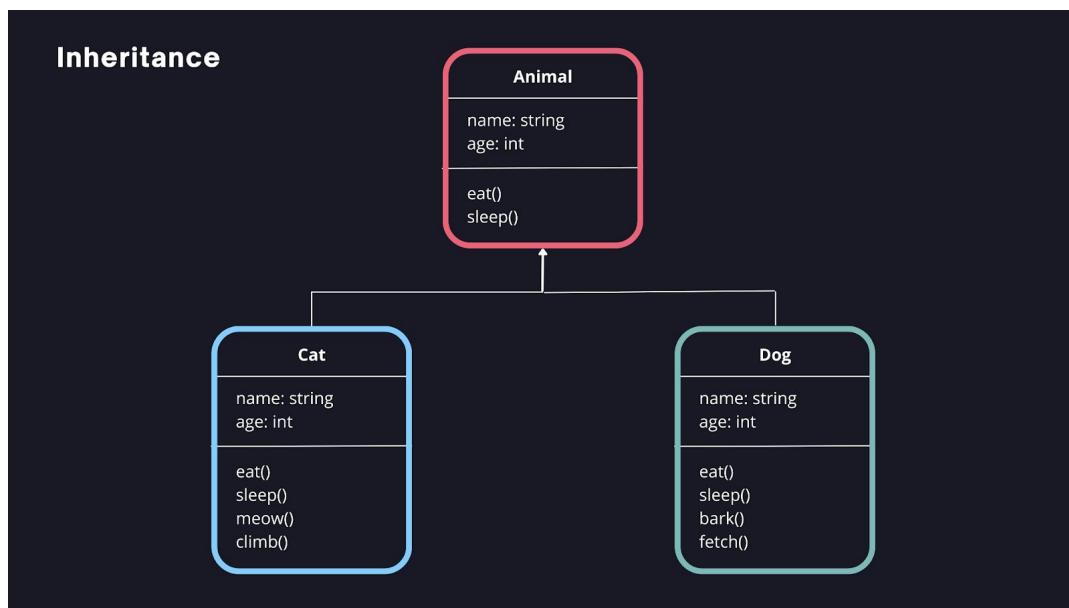


Previously, our codebase had been fully contained in one file, which spanned over 500 lines long. This made it extremely difficult to find sections of code to edit, and difficult to scroll through. With this codebase, we're looking to:

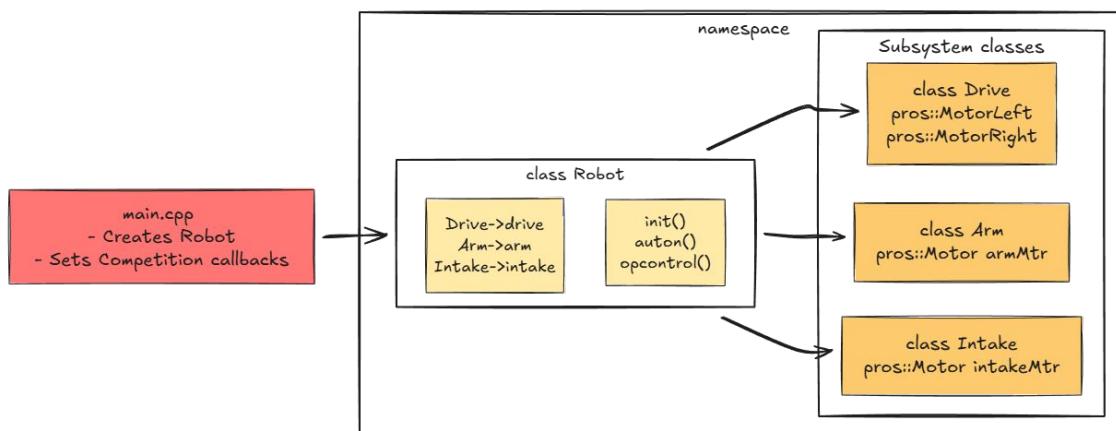
- ❖ Organize similar files together
- ❖ Make it easier to find, edit, and remove snippets of code or replace entire sections

Solution 1 - OOP

We considered using an object-based structure, where each system receives its own “object” created from the same blueprint, based on its type (motor, lift, pneumatics, sensor)



*Figure 1: Diagram from
<https://www.masaischool.com/blog/introduction-to-object-oriented-programming/>*



Solution 2 - Single-file

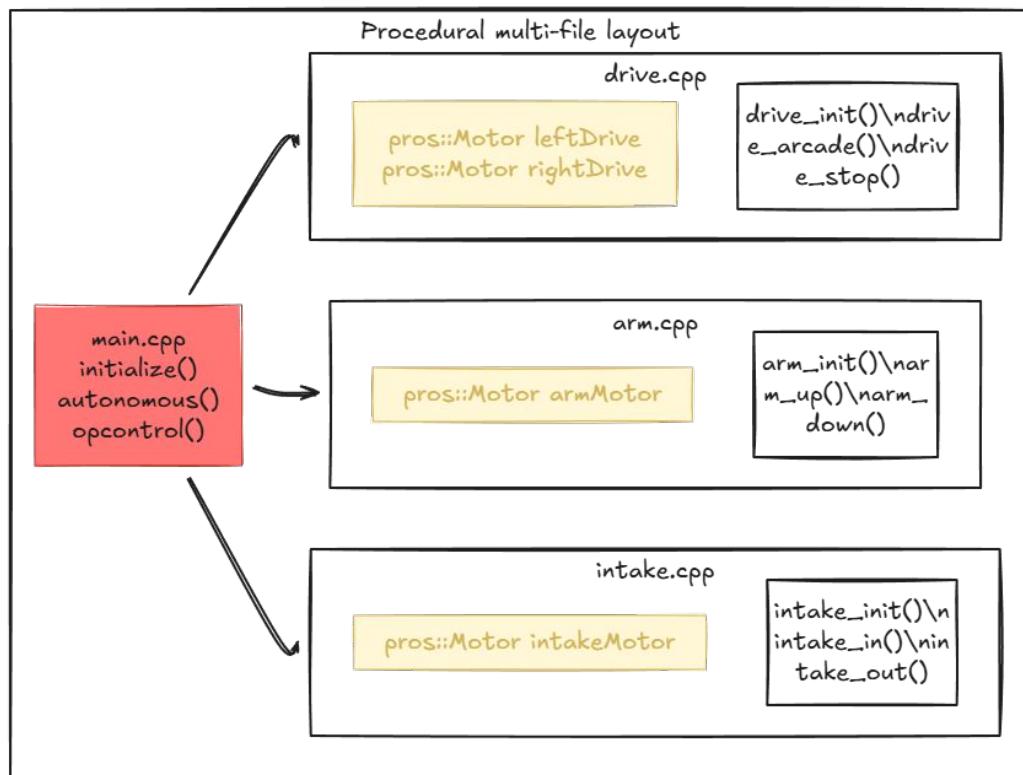
This solution would be identical to our previous solution, involving a single file to handle all of our code.

- ❖ A singular file removes the need for global headers, as variables are shared locally across all function in a single file

Solution 3 - Multi-file

This solution would be identical to our previous solution, involving a single file to handle all of our code.

- ❖ A singular file removes the need for global headers, as variables are shared locally across all function in a single file

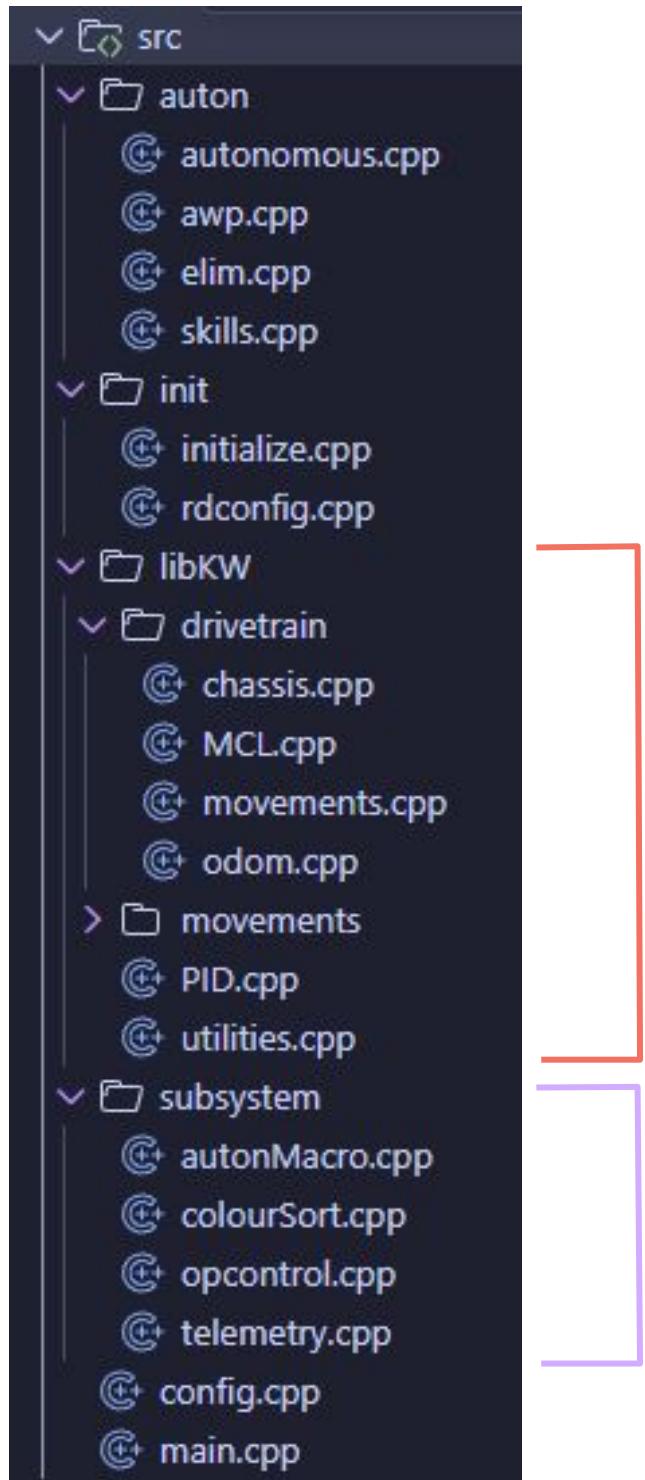


Solution	Pros	Cons
OOP	<ul style="list-style-type: none"> Functionality is split into clear components Object creation makes intent easier to read Code is grouped by file, so locating and updating logic is simpler Less reliance on global state 	<ul style="list-style-type: none"> Large and detailed header files Adding big features can feel tedious Longer compile times Still lacks clear separation between parts like PID logic vs subsystems
Single-file	<ul style="list-style-type: none"> Minimal header files Fewer files may shorten compile times 	<ul style="list-style-type: none"> Heavy dependence on global state Requires many comments to explain intent Very long files, which increases risk when editing Functionality is not clearly separated
Simple multi-file	<ul style="list-style-type: none"> Cross-file communication stays straightforward Headers stay light 	<ul style="list-style-type: none"> Some files become overloaded and hard to search through File names don't clearly reflect responsibilities Functionality isn't separated into clear areas Often relies on global state

We ended up deciding on a combination of **OOP** and **simple multi-file**. This way, we can take commonly used functions (like PID controllers) and make multiple instances of them for the drivetrain and subsystem(s). Yet, the multi-file structure also allows us to split functions up into multiple files and solve our organization issue.

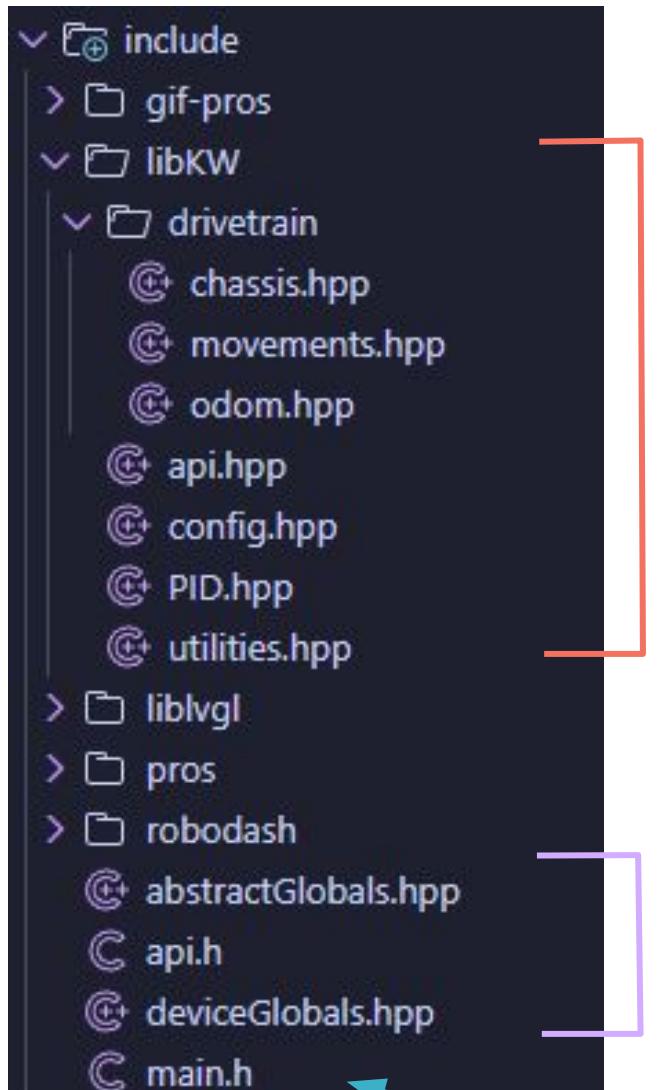
Based on our planning, we've created the following file structure:

- For all “custom” code related to a specific season (intake, autonomous routines, we put them in the “subsystem” folder.
- libKW is a **custom library**, where we put generalized autonomous-specific code for movements and localization, which are transferable between robots and across multiple seasons



These files make up our **libKW** library, which provide functionality during the autonomous period

These files contain robot-specific functionality for our intake, driver control, macros (e.g. sorting blocks by colour) and for brain <-> controller two-way data relay (telemetry.cpp)



Each file in the src/ folder (last page) has a corresponding headers entry in include/

This allows for the functions declared in those files to be made “global”, accessible in any other file within the codebase.

These files contain season-specific declarations in src/ which are outside the scope of libKW

main.h nests the inclusion of every other file within the includes/ folder. Thus, each new <x>.cpp file only needs to include main.h rather than every relevant header.

- Using this structure, we are able to develop **libKW** as an independent library alongside our season-specific codebase. This helps us publish it online as its own standalone package
- Another advantage is that different **versions** of libKW can be swapped into our codebase, allowing for simultaneous testing between our two programmers

(p.s. libKW stands for “lib-Kawaii”, based on our team name.)

- Examining our structure further, let's look into a snippet of **opcontrol.cpp**, the function that allows our driver to control the robot remotely
- Specifically, we will look at the execution of our robot's movement and turning

210K-PushBack-2026 - main.cpp

```

11 void opcontrol() {
12
13     while (true) { // Main continuous Loop
14         /* Drive */
15         kw::drive_arena(0, 0, 0.7); ←
16
17         /* Subsystem Listeners */
18         refreshIntake();
19         refreshLoader();
20         refreshKnocker();
21         refreshWing();
22
23         pros::delay(10); // Delay to save resources on brain
24     }
25 }
```

This is our main function for handling driving in **libKW**. Every 10ms, this loop will refresh and call **drive_arena()**

210K-PushBack-2026 - chassis.hpp

```

1 #pragma once
2 #include "main.h"
3
4 namespace kw {
5     extern double drive_curve(double input, double curve);
6     extern void drive_arena(int LinCurve = 0, int rotCurve = 0, double turnScale = 1); ←
```

This function is declared global in **chassis.hpp**, allowing the code in **opcontrol()** to access the drive function

210K-PushBack-2026 - chassis.cpp

```

8 void kw::drive_arena(int LinCurve, int rotCurve, double turnScale) {
9     double power = controller.get_analog(pros::E_CONTROLLER_ANALOG_LEFT_Y);
10    double rawTurn = controller.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_X);
```

Finally, the global declaration will resolve to the function stored in **chassis.cpp** which performs the movement action

- Examining our structure further, let's look into a snippet of **opcontrol.cpp**, the function that allows our driver to control the robot remotely
- Specifically, we will look at the execution of our robot's movement and turning

210K-PushBack-2026 - main.cpp

```

11 void opcontrol() {
12
13     while (true) { // Main continuous Loop
14         /* Drive */
15         kw::drive_arena(0, 0, 0.7); ←
16
17         /* Subsystem Listeners */
18         refreshIntake();
19         refreshLoader();
20         refreshKnocker();
21         refreshWing();
22
23         pros::delay(10); // Delay to save resources on brain
24     }
25 }
```

This is our main function for handling driving in **libKW**. Every 10ms, this loop will refresh and call **drive_arena()**

210K-PushBack-2026 - chassis.hpp

```

1 #pragma once
2 #include "main.h"
3
4 namespace kw {
5     extern double drive_curve(double input, double curve);
6     extern void drive_arena(int LinCurve = 0, int rotCurve = 0, double turnScale = 1); ←
```

This function is declared global in **chassis.hpp**, allowing the code in **opcontrol()** to access the drive function

210K-PushBack-2026 - chassis.cpp

```

8 void kw::drive_arena(int LinCurve, int rotCurve, double turnScale) {
9     double power = controller.get_analog(pros::E_CONTROLLER_ANALOG_LEFT_Y);
10    double rawTurn = controller.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_X);
```

Finally, the global declaration will resolve to the function stored in **chassis.cpp** which performs the movement action

- Here's an example of our OOP-based PID implementation, as it provides many utilities that we believe will be used repeatedly in the codebase:

210K-PushBack-2026 - movements.cpp

```
67 PID pid_distance = PID(distance_kp, distance_ki, distance_kd);
```

A PID object is made using the PID object constructor

210K-PushBack-2026 - PID.hpp

```
5 class PID {
6     public:
7         PID(double new_kp, double new_ki, double new_kd);
```

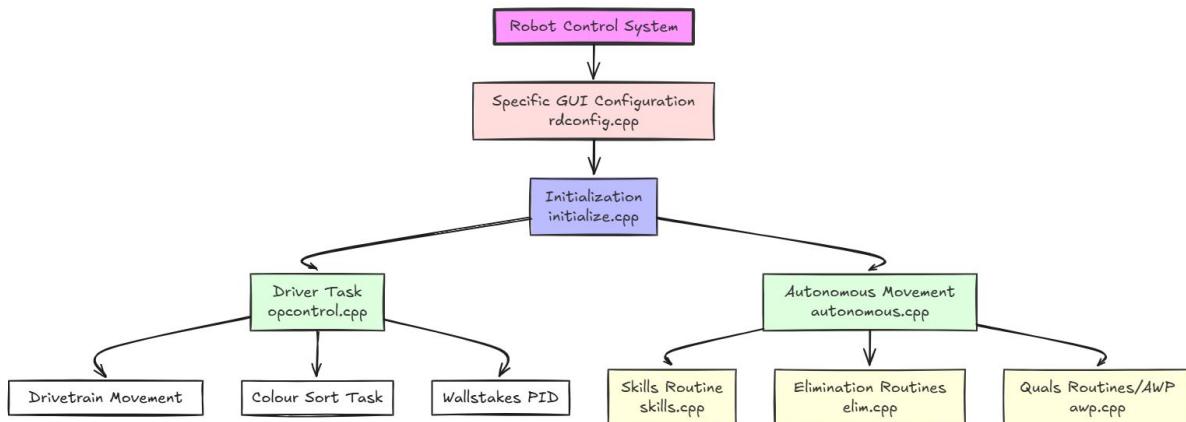
This declaration will resolve to the global definition, which will be included in main.h and imported into every .cpp file

210K-PushBack-2026 - PID.cpp

```
4 // Constructor
5 kw::PID::PID(double new_kp, double new_ki, double new_kd)
6 : arrived(false),
7  arrive(true),
8  small_error_tolerance(1),
9  big_error_tolerance(3),
10 small_error_duration(100),
11 big_error_duration(500),
12 small_check_time(0),
13 big_check_time(0),
14 first_time(true),
15 integral_range(0),
16 integral_max(500) {
17 // Set up the coefficient.
18 kp = new_kp;
19 ki = new_ki;
20 kd = new_kd;
21 // Not arrived initially.
22 arrived = false;
23 }
```

The function itself is declared in **PID.cpp**, where elements of the object will be specified and some variables will have defaults set

- Below is an overview of an example overview of the connection between files:



Version 1.0
Codebase C1

Mall of America
Signature Event

The standard method of driving a robot is called the “**arcade scheme**”, where the left joystick maps to lateral movement and the right joystick maps to turning.

- Most teams use this scheme by mapping the joystick values to the motor speeds directly; this is a **linear relationship**, where motor speed = joystick position
 - ◆ When a joystick is pushed as far as it can go, the motors corresponding to it would turn at maximum velocity. If it is pushed halfway, the motor turns at half of its max speed.



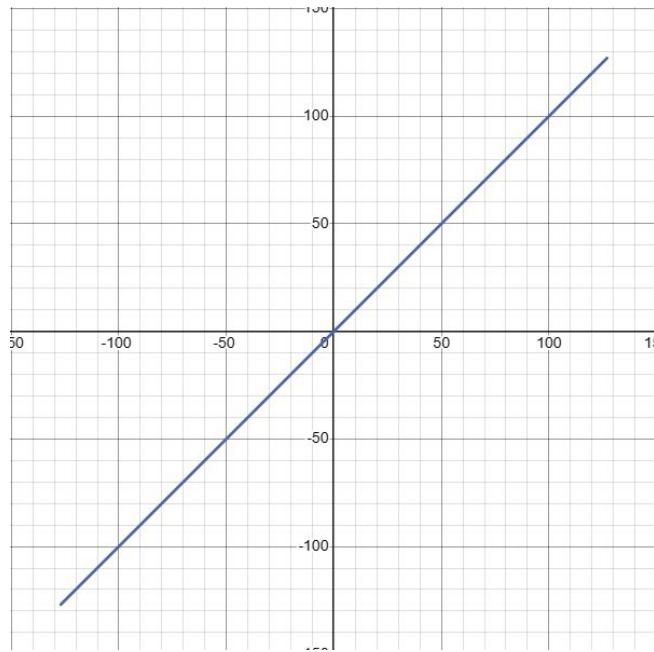
However, this default mapping is not ideal for our drivetrain because it becomes hard for the driver to make fine adjustments.

- Just moving the joystick by 1/4ths of an inch can affect the speed of the motors by up to 25%
- By using an **exponential curve** instead of a linear one, we shift that control to when the robot is moving at slower speeds.
 - **Slight changes to the joystick will have more impact and precision when driving slowly**, but when the joystick is pushed to max throttle, the drivetrain will still go to full speed and become less responsive to small shifts in input

We implement a formula initially derived from Team 5225A to implement the drive curve.

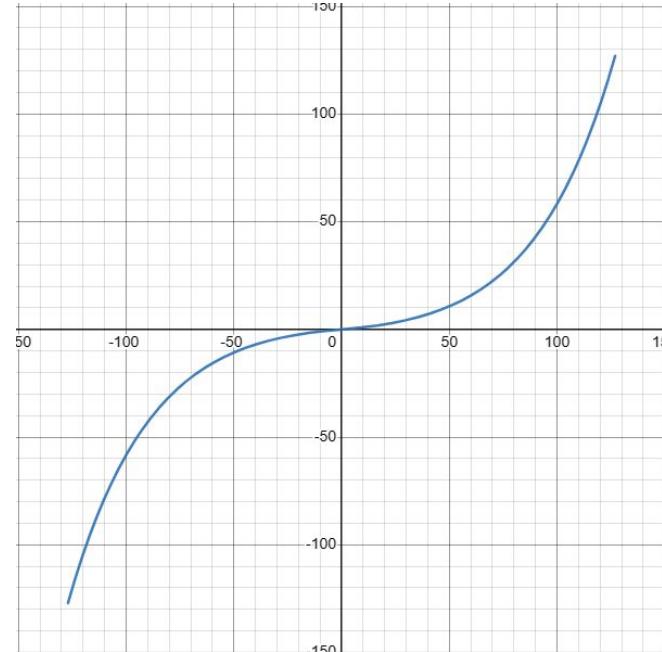
Though there are many ways to curve controller input such as with exponential equations (x^2 , x^3 , etc.). Our method is advantageous in our opinion because it is much **simpler to tune**.

The basic gist of this system is that the drive curve allows us to **modify the sensitivity of the controls**, allowing us to adapt the robot to our driver, not the other way around. Since most drivers (like ours) often use controllers the most by playing video games, and video games allow players to change the sensitivity, the same should be possible in robotics, so that our driver can be as proficient with the controls as possible based on their muscle memory.



Standard Drive Mapping

v.s.



Our Curve

This equation relies on Euler's number, which is commonly used in other applications involving exponential growth. The exponential function e^x always grows at a rate (or derivative) of e^x . This can be changed with the t value, which determines how "aggressive" the curve is.

x = joystick analog input, between 0 and 127 X

t = curve constant, set as variable "drive_curve_constant" in the example code. this can be changed to your driver's liking X

$$y = \left(e^{-\left(\frac{t}{10}\right)} + e^{\left(\frac{(\text{abs}(x)-127)}{10}\right)} \cdot \left(1 - e^{-\left(\frac{t}{10}\right)}\right) \right)_x$$
X

$$(\text{powf}(2.718, -(\text{drive_curve_scale} / 10)) + \text{powf}(2.718, (\text{fabs}(\text{joy_stick_position}) - 127) / 10) * (1 - \text{powf}(2.718, -(\text{drive_curve_scale} / 10)))) * \text{joy_stick_position}$$
X

The first term calculates exponential decay, which is needed to shift the beginning of the curve upwards by increasing the rate of change based on the curve constant t .

$$e^{-(t / 10)}$$

The first coefficient of the second term calculates the exponential curve value itself, by using absolute input values to ensure the output will be the same for when the input is positive or negative.

$$e^{(\text{abs}(x) - 127) / 10}$$

The second factor scales up the curve to ensure when the input is “127”, the output is also “127” (maximum). All of this is multiplied by the input value (x) to ensure that the resulting curve is **proportional** to the input values. Otherwise, the curve is useless.

$$1 - e^{-(t / 10)}$$

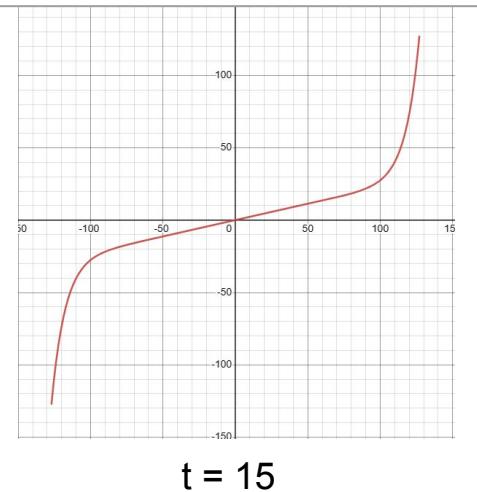
Since we use the absolute value of x earlier, we need to multiply the result by x to account for when the input is negative.

x

The “t” value used in the equation is a constant we can modify to change the slope of the curve, or the “sensitivity” of our controls.

Here's an example of our implementation if, t was set to 15:

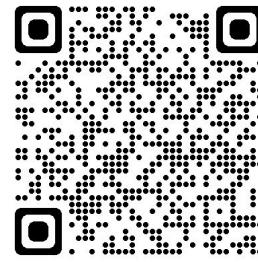
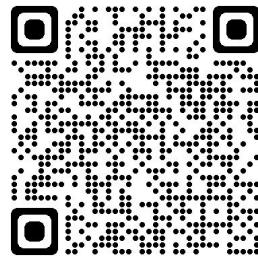
- The bot accelerates very slowly until we push the joystick $\frac{3}{4}$ of the maximum position, which helps us make ultra-precise adjustments while still driving at max speed both forwards and backwards. This drive curve will only be enabled for lateral movements on our drivetrain.



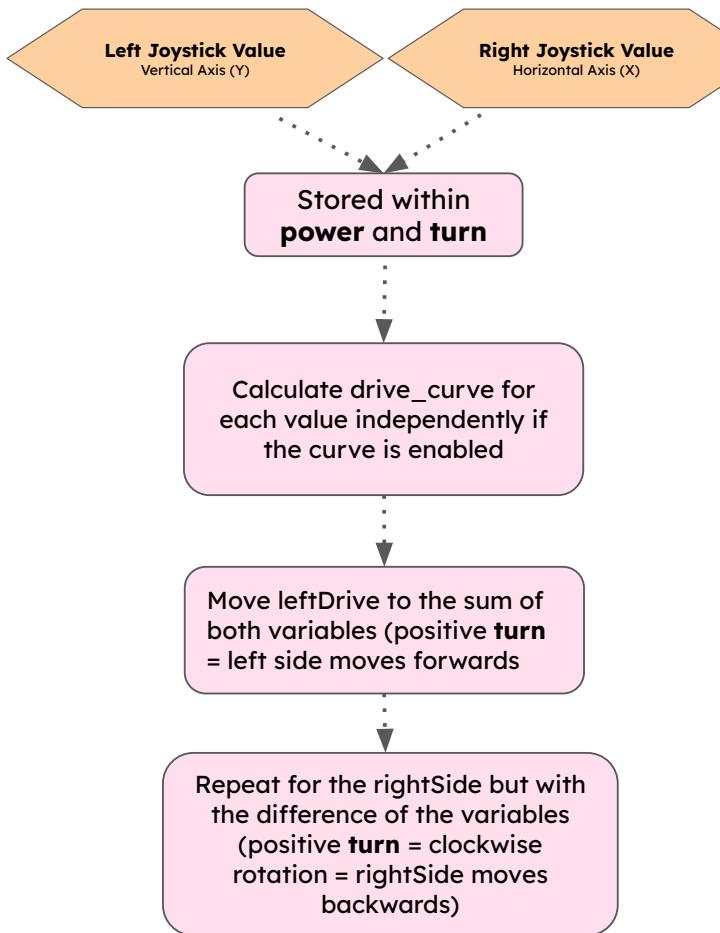
In our codebase, the equation is represented as a function which is continuously fed the joystick inputs. An input of 64 (~50% throttle) with a t constant of 15 would give an output of 14, which would only turn the motors at 11% of their max velocity. This process of polling the joystick position and running the calculations is run 100 times per second.

Forum post detailing the original concept from 5225V (VEXCode)
<https://www.vexforum.com/t/vexcode-joystick-curving/76987>

Interactive graph of our own implementation for demonstration
<https://www.desmos.com/calculator/xis0uxez1>



As stated previously, we use the **arcade control scheme** for our robot. To program it, we just take the input from each joystick and add/subtract them to get the linear motor power for each side. To add the drive curve, we just call its function and replace the value of the measured joystick position with the curved output.



```

210K-PushBack-2026 - chassis.cpp

8 void kw::drive_arduino(int LinCurve, int rotCurve, double turnScale) {
9     double power = controller.get_analog(pros::E_CONTROLLER_ANALOG_LEFT_Y);
10    double rawTurn = controller.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_X);
11
12    if (LinCurve != 0) {
13        // poll joystick input and convert to mv, then run through drive_curve function
14        power = kw::drive_curve(power, LinCurve);
15    }
16    if (rotCurve != 0) {
17        // poll joystick input and convert to mv, then run through drive_curve function
18        rawTurn = kw::drive_curve(rawTurn, rotCurve);
19    }
20
21    // move motors based on direction (eg move Left more when turn is positive)
22    leftDrive.move_voltage((power + rawTurn * turnScale) * (12000.0 / 127));
23    rightDrive.move_voltage((power - rawTurn * turnScale) * (12000.0 / 127));
24 }
  
```

As mentioned beforehand, one major reason we use PROS is because of its direct control over motors by changing the voltages supplied to them.

From reading the VEX documentation, we learned that the motors run a PID control algorithm internally (PID is a way of ensuring precise and fine-tuned control in many different technical applications, like in a robot drivetrain or a smart thermostat). According to VEX themselves in a support article, any motor instructed to run at full throttle will operate at **less than 100% of the max speed** it is capable of, unless it is directly controlled with voltage commands. Additionally, the internal PID controller would lead to oscillation for the motor voltages, which would not be beneficial for situations like when we want to drive forward at max speed and want to push as much power as possible through the motors.

Motor specifications published by VEX:

The motor's internal circuit board has a full H-Bridge and its own Cortex M0 microcontroller to measure position, speed, direction, voltage, current, and temperature. The microcontroller runs its own PID (proportional-integral derivative) with velocity control, position control, torque control, feedforward gain, and motion planning similar to industrial robots. PID is internally calculated at a 10 millisecond rate. The motor's PID values are pre-tuned by VEX for excellent performance across all operating conditions.

Advanced users can bypass the internal PID and take direct control with raw, unaltered PWM (pulse-width modulation) control. Raw control still has the same rpm limits, current limits, and voltage maximum that keep the motor's performance identical.

Additional control of the V5 Smart motor is achieved by internal encoders. These measure the amount of rotation of the shaft socket. The rotation is divided into a number of steps or "ticks" which provides feedback as to the amount a shaft has turned. The resolution of the encoder is determined by the internal gear cartridge of the motor. The encoder values are provided in the chart below per each gear cartridge.

Sourced from VEX 11W Motor Documentation:

<https://kb.vex.com/hc/en-us/articles/360044325872-Understanding-V5-Smart-Motor-11W-Performance>

Since we want fast acceleration speeds on our drivetrain and our driver should have **full control** over the robot's movements, we found a way to bypass this control system using move_voltage in PROS, which directly instructs the brain to send a certain voltage to motors, expressed in millivolts. Polling joystick values from the controller will still return a simpler arbitrary value between -127 and 127. We solve this problem by doing simple unit analysis to convert the values. Since we know the VEX motors operate at a maximum of 12,000 millivolts and the fastest "speed unit" it runs at is 127, we derive the following equation:

$$\text{joystickValueInMillivolts} = \text{originalValue} \times \frac{12000\text{mV}}{127}$$

This data is then passed on into other functions to be used for driver control.

Using the concepts we've previously discussed, it is now possible to set up basic driver control on our robot, using a split arcade control scheme between the two joysticks. Here's the code we use to make that possible:



210K-HighStakes-2025 - chassis.hpp

```
19 inline void arcadeDrive(int LinCurve, int rotCurve, double turnScale) {  
20     // poll joystick input and convert to mv, then run through drivecurve function  
21     int power = ks::driveCurve(controller.get_analog(pros::E_CONTROLLER_ANALOG_LEFT_Y), LinCurve);  
22     int turn = ks::driveCurve(controller.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_X), rotCurve);  
23  
24     // move motors based on direction (eg move left more when turn is positive)  
25     leftDrive.move_voltage((power + turn) * (12000.0 / 127));  
26     rightDrive.move_voltage((power - turn) * (12000.0 / 127));  
27 }
```

- 1) The function will take in constants we've defined from main.cpp. It will then poll the current state of both joysticks, and pass along these numbers to the driveCurve function we defined earlier.
- 2) The return values from the curve will be stored as **power** and **turn**, representing the forwards control and turning controls respectively.
- 3) For simplicity's sake, we both convert the values to voltages and calculate motor speeds in one step.
 - a) **turn** is both added to and subtracted from motors on different sides of the drivetrain, as **turn** is a vector quantity, representing magnitude and direction since it can also be negative.
 - i) This will slow down or speed up sides of the drivetrain relative to each other, which creates the turning effect
 - ii) If there is no turning applied, then the motors will both move with the speed defined in **power**, thus moving forward
 - b) The calculated motor speeds are converted to a voltage using simple unit analysis as described prior
- 4) The final result is then given to the **move_voltage** command that moves all the motors on both sides of the drivetrain independently with voltage control.

This season, we are using some intermediate-level movement algorithms to move our robot during the autonomous period. We'd still like to show the evolution of our robot's autonomous code from past seasons that use simpler logic to convey past problems and how we addressed them.

Many seasons ago (2022-2023), we used lines of code like the one shown. We had to manually specify the amount of millivolts being supplied to our drivetrain motors and how long to spin them for. We quickly abandoned this method because having time delays on motor runtime was **difficult to tune**.

Instead, we could find the circumference of our wheel and divide our target distance by it to find how many wheel rotations the wheels had to spin.

Applying the same logic, we determined how much to turn our wheels by thinking of the turn as an arc made by the robot's wheels, with the radius being the horizontal distance from the centre of the robot to the wheels. This arc **subtends our target angle** based on the formula $a = \theta \times r$.

```

210K-HighStakes-2025 - chassis.cpp

3 void ks::moveRaw(int voltage, int time) {
4     leftDrive.move_voltage(voltage);
5     rightDrive.move_voltage(voltage);
6
7     pros::delay(time);
8
9     leftDrive.move_voltage(0);
10    rightDrive.move_voltage(0);
11 }

```

```

210K-HighStakes-2025 - chassis.cpp

28 void ks::moveDistance(int target, int velocity) {
29     int wheel_diameter = 4;
30     int wheel_circumference = wheel_diameter * M_PI;
31     int rotations_count = target / wheel_circumference;
32
33     leftDrive.move_relative(rotations_count * 360, velocity);
34     rightDrive.move_relative(rotations_count * 360, velocity);
35 }
36
37 void ks::turnAngle(int target_theta, int velocity) {
38     int wheel_diameter = 4;
39     int wheels_offset = 8;
40
41     int wheel_circumference = wheel_diameter * M_PI;
42     int arc_length = target_theta * wheels_offset;
43     int rotations_count = to_radian(arc_length / wheel_circumference);
44
45     leftDrive.move_relative(rotations_count * 360, velocity);
46     rightDrive.move_relative(-rotations_count * 360, velocity);
47 }

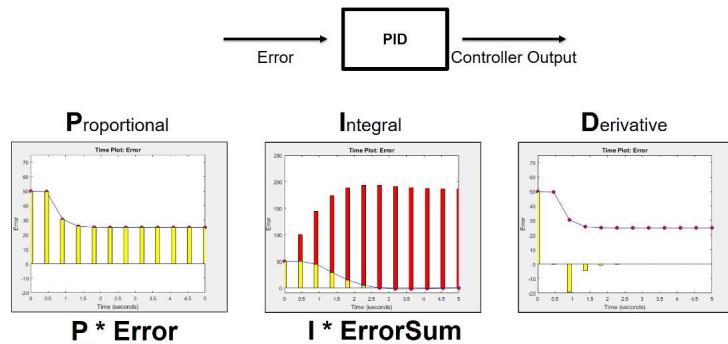
```

Still, this code made our robot's autonomous movements inaccurate and hard to map out, given we still had to guess or hand-measure the distance the robot needed to travel. We realized the biggest problem the robot faced was overshooting its target point because it was **attempting to brake instantaneously** once it reached its target point.

At first glance, programming autonomous movement in VEX may seem simple enough. One may simply believe that by inputting specific lateral commands, such as “move 3 inches left” or “move forwards 5 feet”, the robot will move to these targets with accuracy. While this may be apparent theoretically or virtually, this is extremely unlikely to be satisfactory in the real world because of several external factors that alter the physics of robot movement. This may include friction, latency, smart cable issues, and even the humidity and wear on a field at a competition.

A simple, yet effective way to rectify this issue is to change our movement based on sensor feedback, done through the usage of a proportional, integral, and derivative motion controller, which we've been investigating for multiple seasons. Much of what we developed from last year is still implemented for our current autonomous movements.

What is PID?



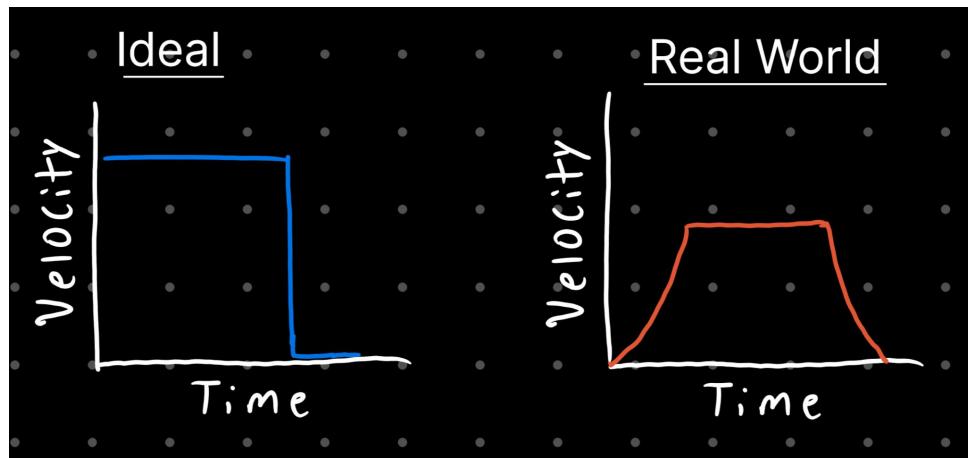
*Mobile Robots, Part 2:
Using PID Controllers
from MATLAB*
<https://www.youtube.com/watch?v=uxb6wl9JWBU>

This concept was pioneered by Nicolas Minorsky, a US navy engineer during the 1920s. “He noted the helmsman steered ships based not only on the current course error but also on past error, as well as the current rate of change” (sourced from https://robotics.caltech.edu/wiki/images/1/15/Minorsky_Paper.pdf).

The formula for a PID Controller is as follows:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t)$$

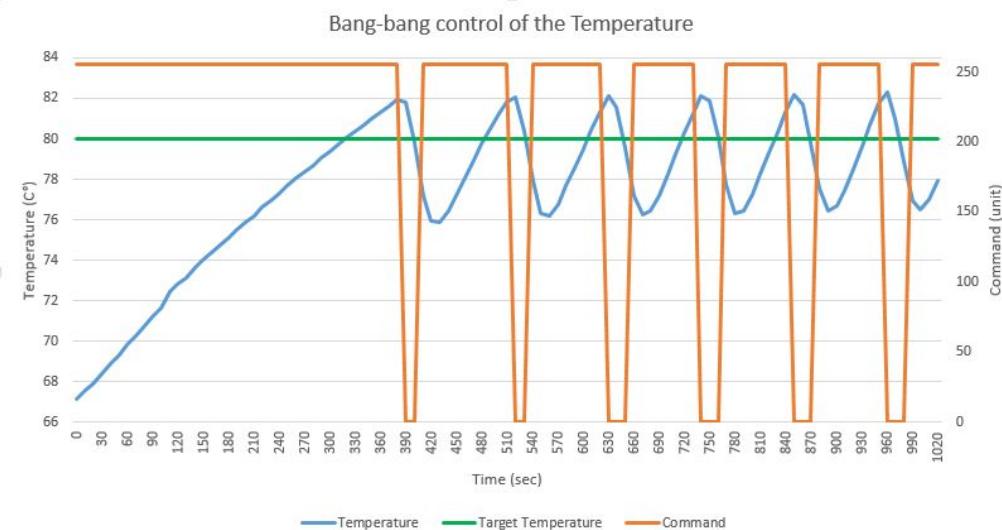
Each portion of the equation is responsible for handling a specific part of correcting robot movement as it attempts to reach a target position.



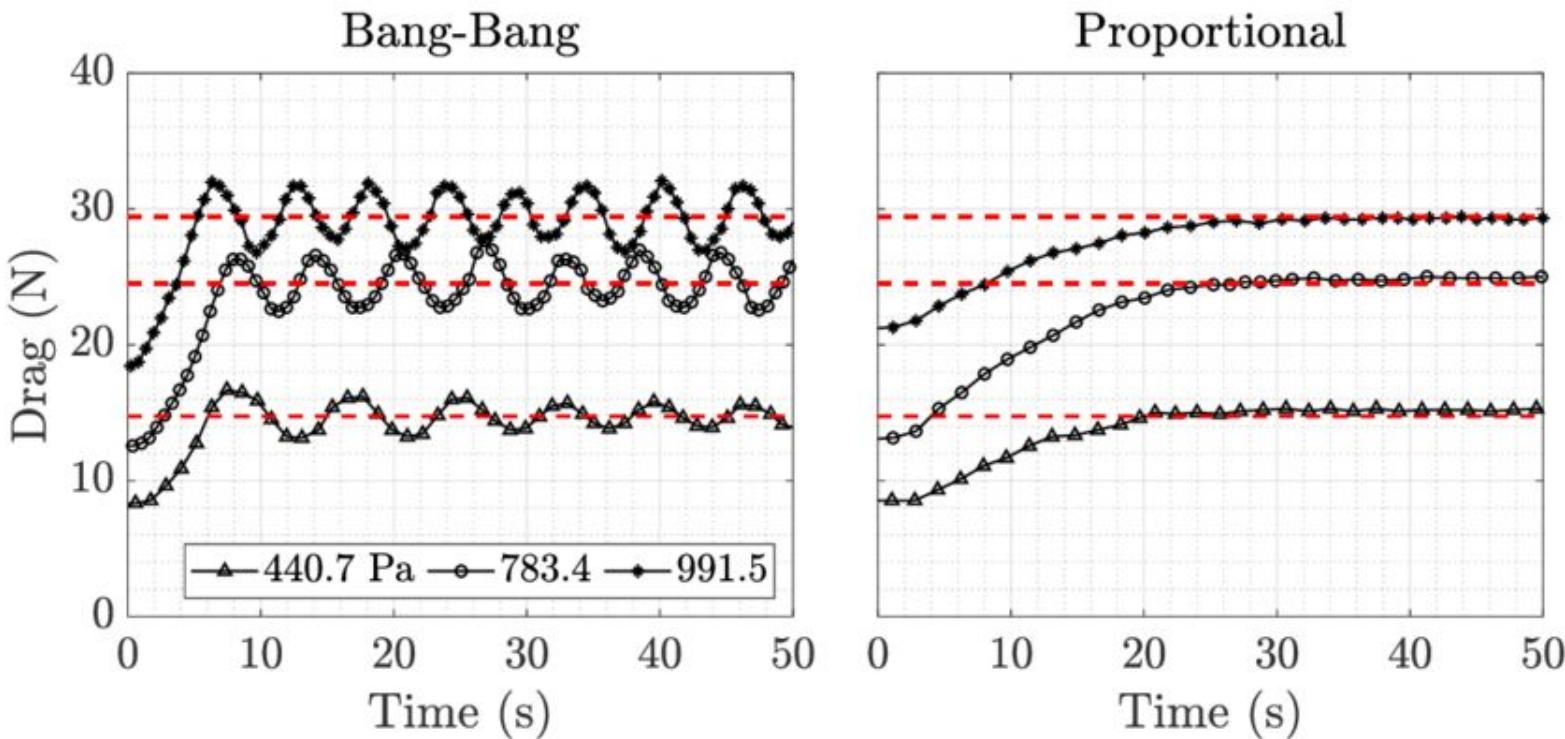
In an ideal world, the velocity at which the robot travels will always be constant. However as we all know, it is impossible to jump to a specific speed immediately. There will always need to be some sort of acceleration to get a robot moving and again to slow it down.

The control algorithms we attempted to implement in the past were made with the assumption of the “ideal” model, otherwise known as a **“bang-bang controller.”** It assumes two states, either the wheels are turning at some set speed or they aren’t. Once the robot reaches what it assumes is its target position, it instructs the motors to brake. In reality, driving a robot is similar to driving a car; stomping on the gas or brake pedals will cause the robot to not react fast enough because it has **inertia**.

Below is a graph of a thermostat controlled by a bang-bang controller. Notice how to maintain the same temperature (blue), there is both **overshooting** and **undershooting**? Sourced from <https://chi.camp/blog/2016/05/26/talking-control/>.



With a proportional controller, we solve this by setting the robot's speed proportional to its **distance from a target position** (known as “error”). This **error** will always be decreasing as it moves toward its target. By multiplying this value with a set constant K_p to adjust the controller’s output for the acceleration and deceleration profile of the chassis, we get an **output** that decreases over time depending on how far the robot is from its desired target, which is **set as the robot’s speed**.



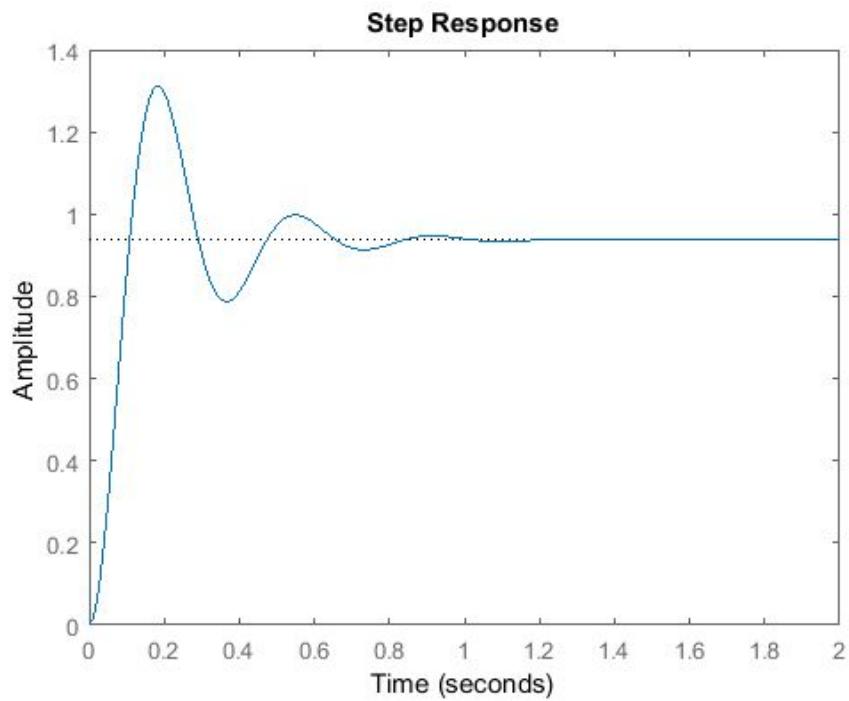
Graph sourced from
https://www.researchgate.net/figure/Bang-bang-and-proportional-control-law-responses-for-the-Dino-at-dynamic-pressure-of-fig3_335737380

Using the proportional results in **slower movement** as the robot **approaches** the target. This prevents overshooting the target position, which results in **greater consistency** when performing linear movements. Note that because of the way we calculate velocity based on position, initial acceleration may seem extremely fast and results in slight jerks at the beginning of the movement. For us, this was remedied by utilizing a **max speed threshold**, and clamping the robot velocity between this threshold.

Below is a table of values to illustrate our point. Note that the proportional controller can also account for overshooting its target position with a negative output to reverse its movement.

Target	Current Position	Error (target - current_pos)
100	0	100
100	40	60
100	80	20
100	120	-20
100	100	0

A common issue with the proportional controller is that it can **never settle**. No matter how well it is tuned and its kP value is adjusted, the robot will still inevitably **oscillate about its target** because it will never have a position sensor reading matching its target position.

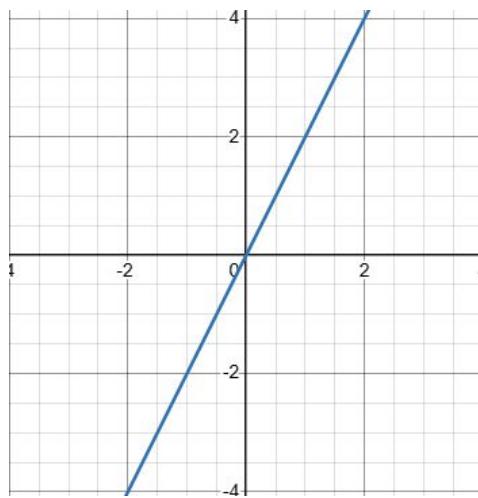


Graph from TMU:

<https://pressbooks.library.torontomu.ca/controlsystems/chapter/8-4the-effect-of-an-additional-pole-on-the-2nd-order-system-response/>

Based on the problems with the proportional controller mentioned previously, we need something to dampen or “normalize” its output. The **derivative controller** does just this.

The derivative is a mathematical concept that helps us understand how a quantity changes in relation to another quantity. It measures the rate at which one thing is changing with respect to another. For example, consider a quadratic equation $y=2x$:

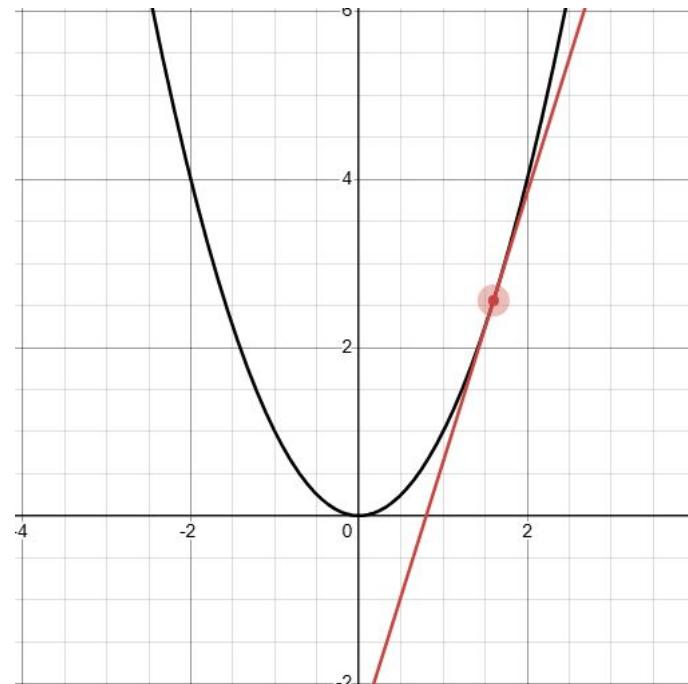


The value 2 represents m , the slope of the equation. If the y-axis stood for distance and the x-axis stood for time, we could say the constant rate of change in this graph is 2 inches / 1 second.

Since we have the proportional controller, our robot's movement looks more like a curve. Because our code runs in cycles spaced milliseconds apart, we can take the **difference of our positional error** between two time periods to find the instantaneous derivative:

$$\text{Derivative} = \text{error} - \text{previous error}$$

We multiply the derivative by a constant kD too, so that we can tune its output relative to what is expected of the robot's movement.



Graphs made by us in Desmos

The integral controller takes the accumulated error over a period of time and accounts for it.

Why would this be important? Well in the case of just using “P” and “D” in the PID controller, they are very accurate and consistent when creating and tuning to reasonable distance and accuracy. Though there will always be a slight offset to the target goal by only using this method, as the accumulated error, especially over longer runs such as skills tends to lead to more deviation between location the robot is at and the location the robot thinks its at. Below are two (exaggerated) graphs to visualize the uses for the I portion of the PID controller.

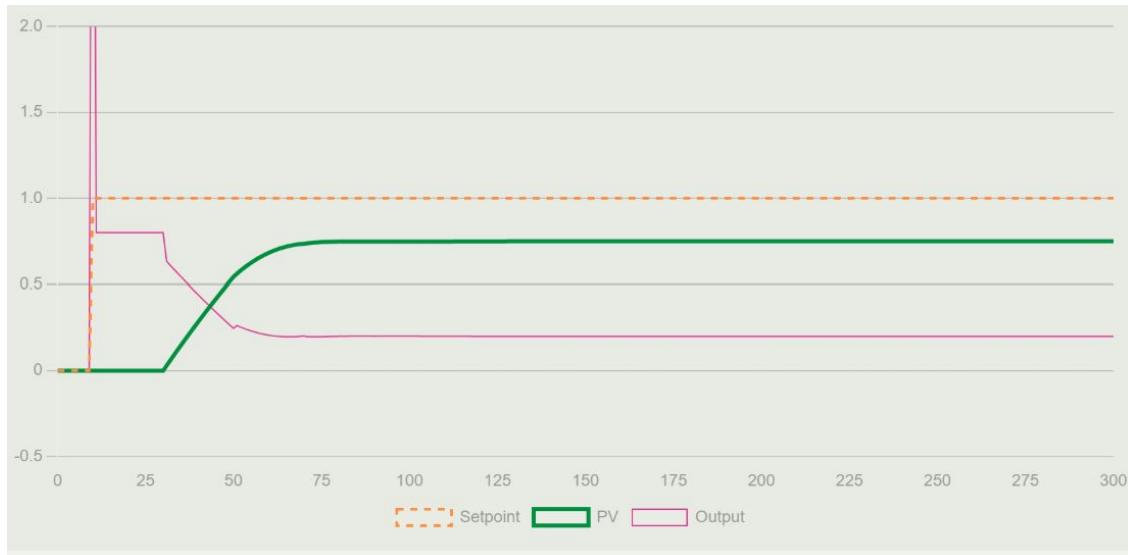


Figure 1: Only P and D portions of the PID controller

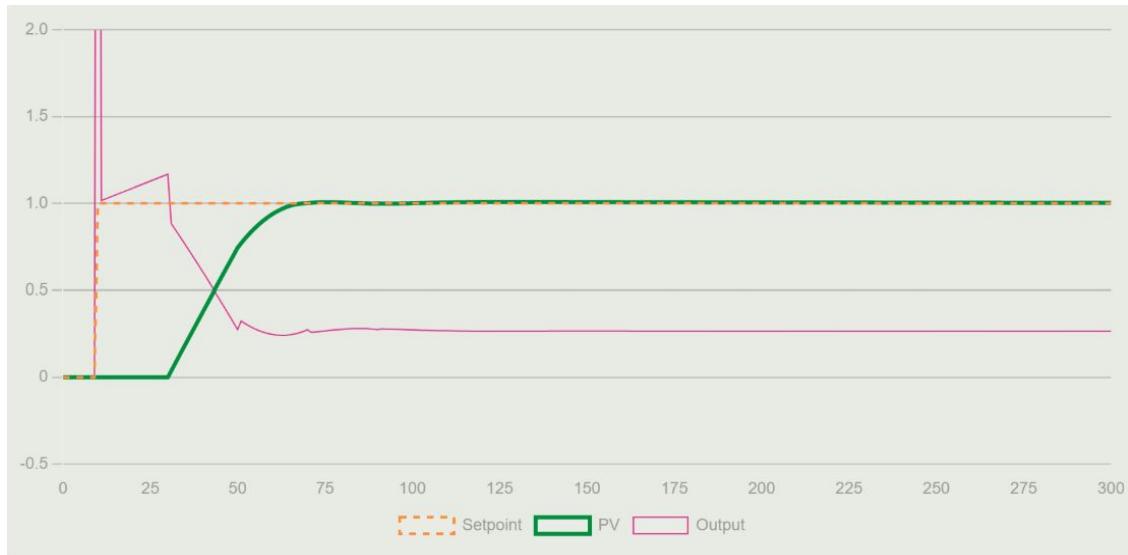


Figure 2: Fully tuned PID function using P, I, and D.

Mathematically, we can solve and prove for the PID controller. As stated previously, the PID controller is defined by this equation:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t)$$

Now let's assume these variables: $K_p = 2$, $K_I = 0.01$, and $K_D = 15$. Let's assume we are at the point (20, 0) and our goal is to reach (100, 0) (all in meters), making our error ($e(t)$) equal to 100m. Lets also assume we just started the program so we are at 0 seconds, thus the accumulated error is only the beginning error. Also assume the values of the time base function move in integer value times. Finally let's assume we have been moving for say 2 seconds.

Proportional:

This part is the easy part. Since our error is 80m and our K_p value is 2, just like the equation we multiply the two together, getting a value of 160m/s

Integral:

The integral is an interesting part having entered the world of calculus based mathematics. In the equation we multiply our K_I value (0.01) by the integral between 0 and our time (2s) and our accumulated error (all the error added up. For simplicity sake, we are going to assume this is 100m). The integral of 100 with respects to time. This ends up being $[100t]$ between the values of 0 and 2. This however just simplifies to $2(100)$ which is just 200. Now we multiply this by our K_I value which is 0.01. This results in our overall accumulated error to require our robot or function to increase by 2m/s.

Derivative:

Finally derivative, the rate at which something (in our case, the slope at which the robot is accelerating compared to its last error). This value looks like the (current error - error say one second ago) all over the time taken multiplied by K_D which we said to be. Let's assume that the error one second ago was 100 and the current error say is now 15. We take all this $(80-100)/2$ and we get -10. We now multiply by the K_D which gives us -150.

All Together

Now adding this all together, for our robot to move 100m accurately in two seconds, it must **initially move at 12m/s** (if it were at the point 20m) according to these calculations. This is what would be going through the logic of the PID controller every few milliseconds during our autonomous period.

With these improvements to our autonomous controller and extensive testing, we have tuned our lateral (forward/backward) control error (how far the real-life performance is from the predicted and expected performance) to within less than 0.5 inches. Additionally, our angular (turning) movements are constricted to within ± 1 degree. We measure the lateral PID by using both field tiles and a ruler as constants, and modify our PID gains until the robot is able to perfectly move to our tuning distances (24 inches and 10 inches).

For tuning the angular PID, we firstly eyeball the robot's movements until it is able to turn close to 90 degrees. We then strap a phone to the robot, and use its inbuilt compass to measure the turning more precisely by aligning it with the North Pole. Our angular control system still needs work through, as when tested with 3 turning commands chained together, the robot will accumulate roughly 10 degrees of error through its movements.

```

210K-PushBack-2026 - deviceGlobals.hpp

49 // Lateral motion controller
50 ControllerSettings lateralController(7.5, // proportional gain (kP)
51                                     0, // integral gain (kI)
52                                     35, // derivative gain (kD)
53                                     0, // anti windup
54                                     1, // small error range, in inches
55                                     100, // small error range timeout, in milliseconds
56                                     3, // Large error range, in inches
57                                     500, // Large error range timeout, in milliseconds
58                                     7 // maximum acceleration (slew)
59 );
60 // angular motion controller
61 ControllerSettings angularController(4.5, // proportional gain (kP)
62                                     0, // integral gain (kI)
63                                     45, // derivative gain (kD)
64                                     0, // anti windup
65                                     1, // small error range, in inches
66                                     100, // small error range timeout, in milliseconds
67                                     3, // Large error range, in inches
68                                     500, // Large error range timeout, in milliseconds
69                                     0 // maximum acceleration (slew)
70 );

```

Figure 1: Our current PID constants

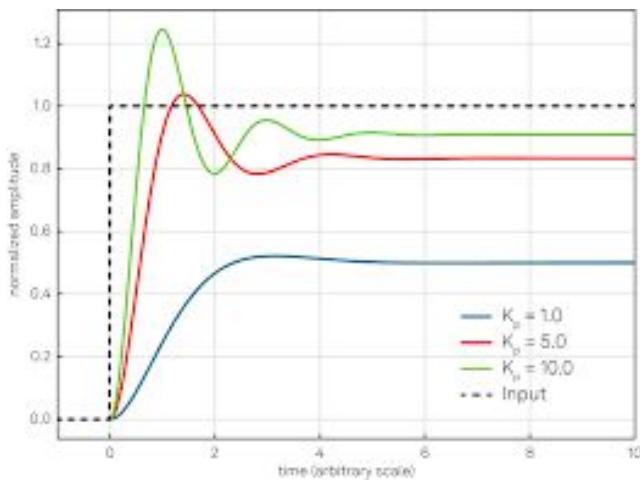


Figure 2: Graph from Zurich Instruments

As a rule of thumb, we tune both controllers by resetting all PID constants to 0, thus disabling the feedback loop. We then start to slowly increase the **P constant** until the robot moves/turns to its target, but slightly oscillates around the target point due to overshooting. This is intended, because our next step is to tune the **derivative**. Since our robot is now overshooting its target, we start increasing the derivative gain to “dampen” the overall output of the PID controller until the robot is landing on target.

```
210K-PushBack-2026 - lateralPID.cpp

17 void ks::LateralPID::move_Lateral_pid(double target, double maxSpeed, double minSpeed) {
18     double prevError = 0;
19     double integral = 0;
20     double derivative = 0;
21     double power = 0;
22     double currentTime = 0;
23
24     double local_timer = 0;
25
26     while (true) {
27         double distance_travelled = ((verticalEncoder.get_position()) * 2 * M_PI / 36000);
28         //printf("%f\n", distance_travelled);
29         double error = target - distance_travelled;
30         integral = (integral + error);
31         derivative = (error - prevError);
32
33         power = (global_kp * error) + (global_ki * integral) + (global_kd * derivative);
34         if (power > maxSpeed) {
35             power = maxSpeed;
36         } else if (power < -maxSpeed) {
37             power = -maxSpeed;
38         }
39
40         if (power < minSpeed) {
41             power = minSpeed;
42         } else if (power > -minSpeed) {
43             power = -minSpeed;
44         }
45
46         // printf("%f\n", power);
47         leftDrive.move_voltage(to_milivolt(power));
48         rightDrive.move_voltage(to_milivolt(power));
49
50         if (local_timer > (global_timeOut * 100)) {
51             leftDrive.move_voltage(0);
52             rightDrive.move_voltage(0);
53             break;
54         }
55
56         if (fabs(error) < 0.2) break;
57
58         local_timer++;
59         prevError = error;
60         pros::delay(10);
61     }
62 }
```

While the integral value can be tuned, we've found no need to use it because the robot rarely undershoots its target since the it moves at max speed most of the time.

Figure 1: Our lateral PID implementation

Using PID, we can tell our robot to move to a target, but it has no way of sensing if it actually arrived at its target or if it ran into something. We recognized that it needs some way of localizing its position relative to some starting point. We found two ways of doing this: **1) the VEX GPS** and **2) odometry**. Both the GPS and Odometry assume the VEX field to be a 2D grid plane viewed from above the field, returning a set of X and Y coordinate values.

- 1) The VEX GPS is the easiest way to find our robot's position. It is a singular sensor that can scan a barcode on the field walls and return the robot's position. However, there are two glaring issues we see with it:

- a) Latency

Based on VEX documentation, the GPS refreshes 25 times a second, or **every 40 ms** ($1/25\text{Hz} * 1000\text{ms}/1\text{sec}$). Odometry can refresh upwards of every 5ms if needed. The VEX GPS also takes pictures of the GPS strip to find the robot's position, which is prone to **motion blurring** and inaccurate readings when used with a fast drivetrain.

- b) Deadzones

Within the yellow shaded area shown in the diagram, the GPS cannot see enough of the barcode strip to accurately determine the robot's position and switches to a inertial measurement unit with **limited accuracy**. This game season, there are also concerns of field elements obscuring the sensor's view of the barcode. Given our robot's autonomous movements **primarily operate within this deadzone** to collect game objects, we find this con to be the biggest concern with GPS positioning.

- 2) Odometry is much more difficult to develop and tune than using a GPS, but it can be so much more accurate and suffers from few of the same flaws. It is prone to drifting values due to shifting heading readouts from the VEX IMU that accumulate over time but we have some ideas on how to fix this in the future.

For now, odometry is the localization method we have opted to implement and develop.

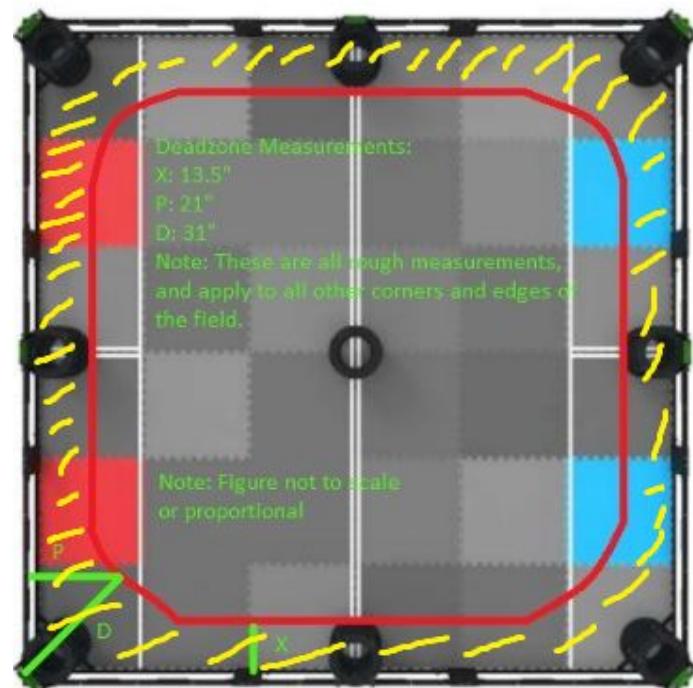


Image attribution: BLRS Purdue Sigbots

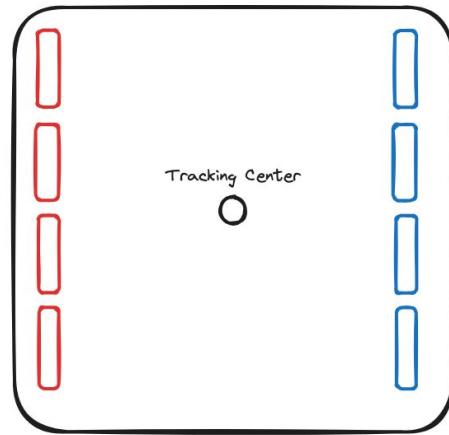
While PID movement for lateral and angular movement works effectively, it is very **time-consuming** to use for autonomous development as we have to tediously tune the movement values through guessing and testing.

The PID controller assumes the robot is moving in a 1D plane where the robot can only move laterally or turn, but not both at the same time. Especially with this year's game, robots will definitely come into contact with each other and field elements often, bumping the robot off course from its predesignated path with no way to know its new position and correcting for it, since it does not take in feedback from external sensors.

These issues can be remedied with the implementation of an Absolute Positioning System, commonly referred to as odometry. The APS is used in the robotics industry by organizations as big as NASA for their Mars rovers, which helps with accurate and precise movement. Our implementation is inspired from the conceptualizations published by team 5225A, one of the first VEX teams to utilize odometry. Unless expressly specified, all resources below belong to us.

Our robot drivetrain is built in the standard tank drive configuration, meaning that there are two sets of parallel wheels on the drivetrain. For a basic odometry implementation, the robot's current heading is tracked using readings from our inertial measurement sensors. Based on our current direction relative to our starting position, we track the amount of drivetrain wheel rotations from the motors.

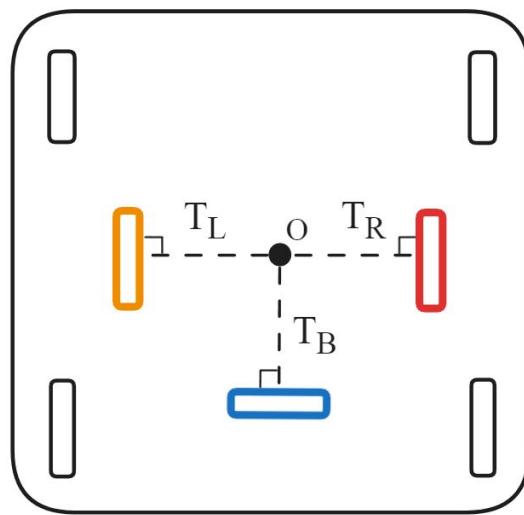
Multiplying this result by the circumference ($\pi * d$) gives us the total lateral distance covered by the robot. We store the x and y position of the robot as we drive in a vector.



In the APS, the robot is represented in 2D space using position and theta vectors. The theta represents the current angle of the robot, relative to its starting position. The position vector is represented by X and Y, similar to the way points are plotted on a graph.

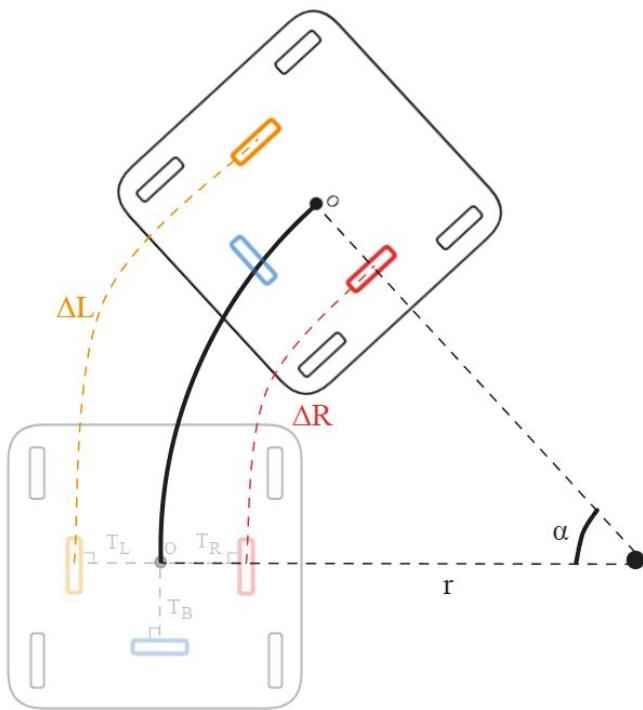
For explaining the math behind odom., we will assume we are tracking the robot's movement using free spinning wheels mounted to rotation sensors, so that the rotation of these free-spinning wheels could be used to track how much distance the chassis has moved across a flat surface. To accomplish this task, I chose to use rotation sensors as a way of measuring movement. My robot has two powered wheels on either side of its chassis, like that of a car. Extra unpowered wheels to track movement can be mounted onto the chassis parallel to its sides and an equal distance from its centre.

Figure 1: A bird's eye view of a robot chassis mounted with three unpowered wheels.



In **Figure 1**, these coloured "tracking wheels" are mounted perpendicular to the robot and are evenly spaced around the "tracking centre," labelled as "O." This equal distance between the wheels and "O" is represented by T_x , where x stands for the side of the robot that the wheels are mounted to. Thus, $TL=TR=TB$.

The easiest way to track the location of the robot at any given time is to map its position onto a 2D Cartesian plane, where its tracking centre can be defined as a X-coordinate and a Y-coordinate. Given that the robot may also turn while moving, we also need to calculate its heading relative to its starting position. **Figure 2** describes a situation where the chassis defined in **Figure 1** moves along the circumference of a circle. Some of the math below may be screenshots taken from external software for typesetting purposes.

Figure 2: A robot chassis moving along an arc subtending angle α .

Given that the robot's tracking centre is perpendicular to the centre of the circle defined by its movement, the change in its orientation ($\Delta\theta$) would be equivalent to the angle of the arc it forms (α). Each tracking wheel's change in measured distance is defined as ΔL and ΔR . Using this information and each tracking wheel's distance from its tracking centre, we can solve for $\Delta\theta$ using the arc formula $l = r\theta$:

$$\Delta L = r_L \Delta\theta$$

Since the arc formed by each tracking wheel has a different radius, we can represent r_L and r_R as the sum and difference of r with respect to T_L and T_R .

$$\Delta L = (r + T_L) \Delta\theta \frac{\Delta L}{\Delta\theta} = r + T_L r = \frac{\Delta L}{\Delta\theta} - T_L$$

Doing the same for ΔR ,

$$\Delta R = r_R \Delta\theta \Delta R = (r - T_R) \Delta\theta \frac{\Delta R}{\Delta\theta} = r - T_R r = \frac{\Delta R}{\Delta\theta} + T_R$$

Combining them together:

$$\frac{\Delta L}{\Delta \theta} - T_L = \frac{\Delta R}{\Delta \theta} + T_R \Delta L - T_L \Delta \theta = \Delta R + T_R \Delta \theta \Delta \theta (T_L + T_R) = \Delta L - \Delta R$$

$$\Delta \theta = \frac{\Delta L - \Delta R}{T_L + T_R} \text{ rad.}$$

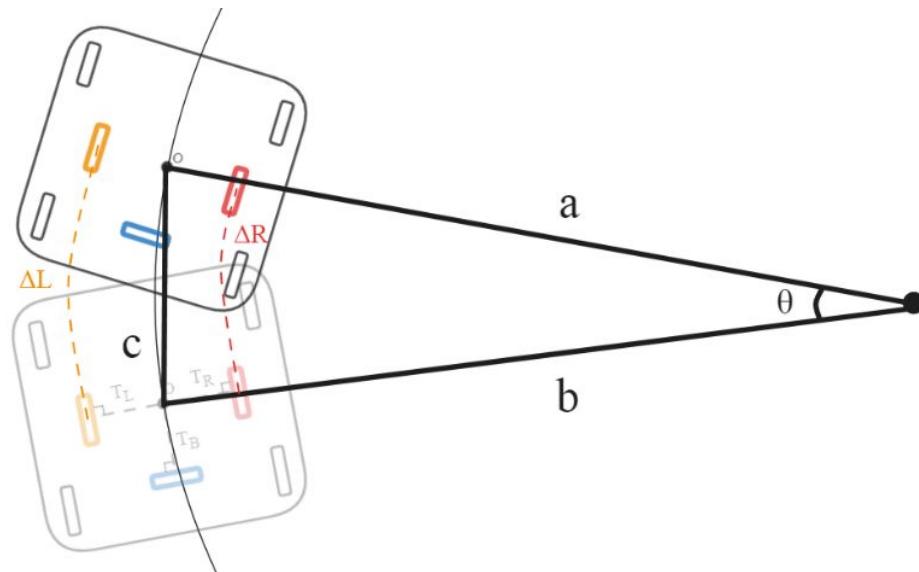
Since the movement of these tracking wheels are measured using rotation sensors, we also need to substitute ΔL and ΔR with the appropriate equations to calculate the distance travelled, with β_L and β_R representing the number of radians that the wheels have rotated respectively along with d_L and d_R representing the wheel diameters:

$$\text{Distance} = \text{rotations} \times \text{circumference}$$

$$\Delta L = \frac{\beta_L}{2\pi} \times \pi d_L \Delta R = \frac{\beta_R}{2\pi} \times \pi d_R \Delta \theta = \frac{\frac{\beta_L d_L \pi}{2\pi} - \frac{\beta_R d_R \pi}{2\pi}}{T_L + T_R} \Delta \theta = \frac{\frac{1}{2}(\beta_L d_L - \beta_R d_R)}{T_L + T_R} = \frac{\beta_L d_L - \beta_R d_R}{2(T_L + T_R)}$$

Beyond just moving along a circular arc, this formula will be applicable for any type of movement to calculate the heading of the robot, even if it moves forward without turning. Using this angle, we can derive the X and Y coordinates of the robot too, solely based off rotation feedback from the tracking wheels and trigonometry as demonstrated in figure 3:

Figure 3: A robot chassis moving along an arc subtending angle θ .



$$\cos 2\theta = 1 - 2 \sin^2 \theta$$

$$\sin^2 \theta = \frac{1 - \cos 2\theta}{2}$$

$$\sin^2 \frac{\theta}{2} = \frac{1 - \cos \theta}{2}$$

$$\sin \frac{\theta}{2} = \pm \sqrt{\frac{1 - \cos \theta}{2}}$$

Then, we can arrange our equation to match this identity:

$$(\Delta y)^2 = 4r^2 \left(\frac{1 - \cos \theta}{2} \right)$$

$$\Delta y = 2r \sqrt{\frac{1 - \cos \theta}{2}}$$

$$\Delta y = 2r \sin \left(\frac{\theta}{2} \right) = 2 \left(\frac{\Delta R}{\Delta \theta} + T_R \right) \left[\sin \left(\frac{\theta}{2} \right) \right]$$

$$\Delta y = 2 \left(\frac{\beta_B d_B}{\Delta \theta} + T_B \right) \left[\sin \left(\frac{\theta}{2} \right) \right]$$

This process can be reapplied to the X-axis to determine a set of complete cartesian coordinates along with heading, given the constants β_B , d_B , and T_B .

$$\Delta x = 2 \left(\frac{\beta_B d_B}{\Delta \theta} + T_B \right) \left[\sin \left(\frac{\Delta \theta}{2} \right) \right]$$

We mentioned previously that one of the reasons we chose to use PROS was because of its superb multitasking capability. But what does this mean?

We came across a problem when programming at a tournament many seasons ago. It was that we couldn't run two functions at once; we needed a loop to handle driver joystick input and another to reload a catapult mechanism. This same problem is apparent this season too, as we want our odometry algorithm to share resources with code that controls physical subsystems.

We did some research and came across multithreading, which is basically splitting up your code to run two or more tasks in **parallel**. It works by initializing a task, and repeating this process to each run a function in its own isolated loop. This is how parallel computing is performed on modern computer processors by using “threads.” VEX Brains don’t have the resource overhead to directly multitask, so it emulates this behaviour by executing the code in each thread for a short amount of time, pausing it, and moving onto the next thread (but it switches them out so fast that there are no visible delays or slowdowns in code execution).

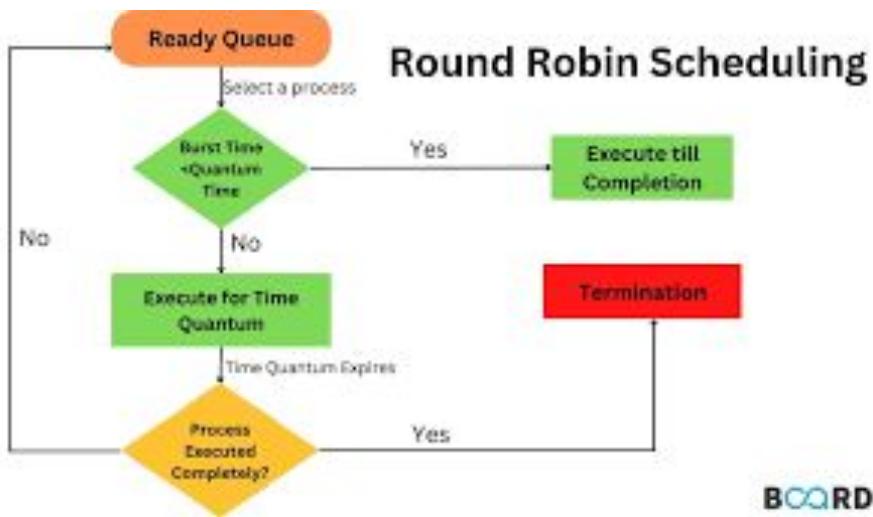
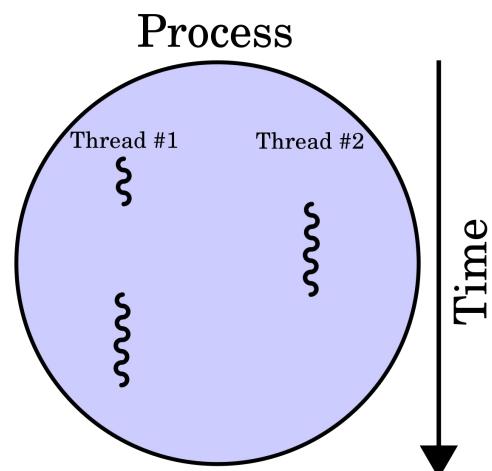


Figure 1: Round-robin scheduling diagram from <https://www.boardinfinity.com/blog/introduction-to-round-robin-scheduling/>

Figure 2: Multithreading diagram from Wikipedia



There's an issue that arises when tasks are reading and writing to the same variable or code at the same time. This could result in corrupted code because if you try to read a variable while it's being written you'll get a unexpected value. Luckily for us there is a concept called **mutexes**, which is the programming equivalent of the "talking stick" that children use. Basically whichever task has the "mutex" has the authority to write and read the code, and no other task can. This ensures no garbage values are created.

It's helpful to imagine multithreading as branches on a timeline, and the mutexes as connections in between those branches. Setting up multithreading was also very easy with PROS, as they have extensive documentation dedicated to multithreading and good support for debugging errors in threads. VEXCode has meager support for multithreading at best because it is considered a "advanced" feature.

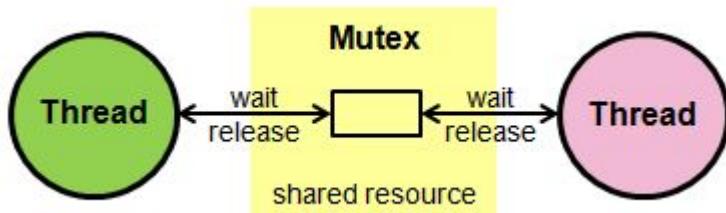
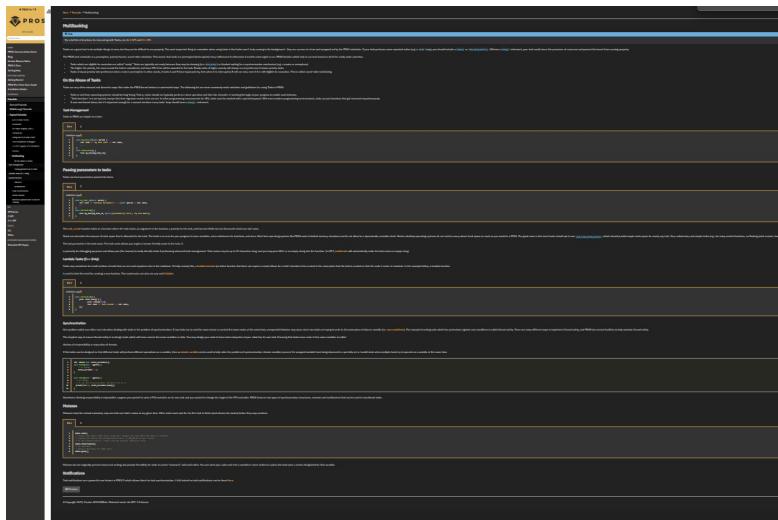


Figure 1: Mutexes diagram from <https://www.linkedin.com/pulse/mutexes-its-implementation-aditya-soni/>



Thread

Initializing the Thread Class

The `Thread` constructor creates a `Thread` object. This constructor uses two parameters:

- `callback`: A previously defined function that will be executed as the entry point when the thread is invoked.
- `arg`: Optional. A tuple that is used to pass arguments to the thread entry function.

```

1 # Define the Function "thread_print".
2 def thread_print():
3     brain.screen.print("Printing")
4 # Construct a Thread "thread" with the
5 # Thread class.
6 thread = Thread(thread_print)
  
```

This `thread` object will be used in all subsequent examples throughout this API documentation when referring to Thread class methods.

Class Methods

stop()

The `stop()` method stops the thread.

Returns: None

sleep_for()

The `sleep_for(duration, units)` method makes the thread sleep for a specified duration.

Parameters	Description
------------	-------------

Figures 2-3: Comparing the feature set of PROS vs VEXCode threading.

2-<https://pros.cs.purdue.edu/v5/tutorials/topical/multitasking.html>, 3-<https://api.vex.com/iq2/home/python/Thread.html>

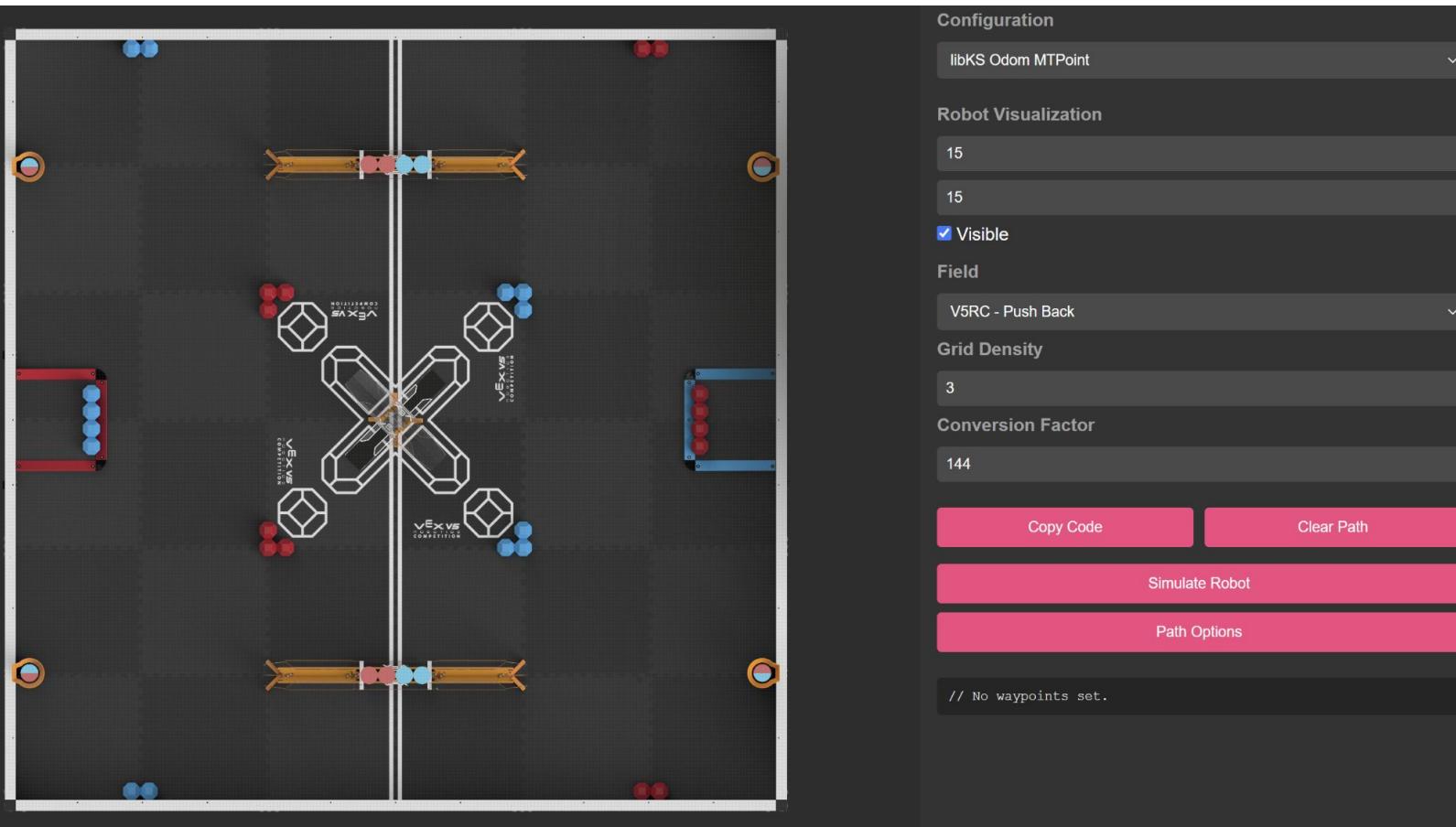
This season, we can use multitasking to run all of our odometry code in a loop and keep it separated from other subsystems, which makes errors easy to debug and reduces the possibility of our PID code trying to read odometry values that are actively being written.

Pairing with our odometry code, we need a way of generating a list of waypoints for our robot to navigate to

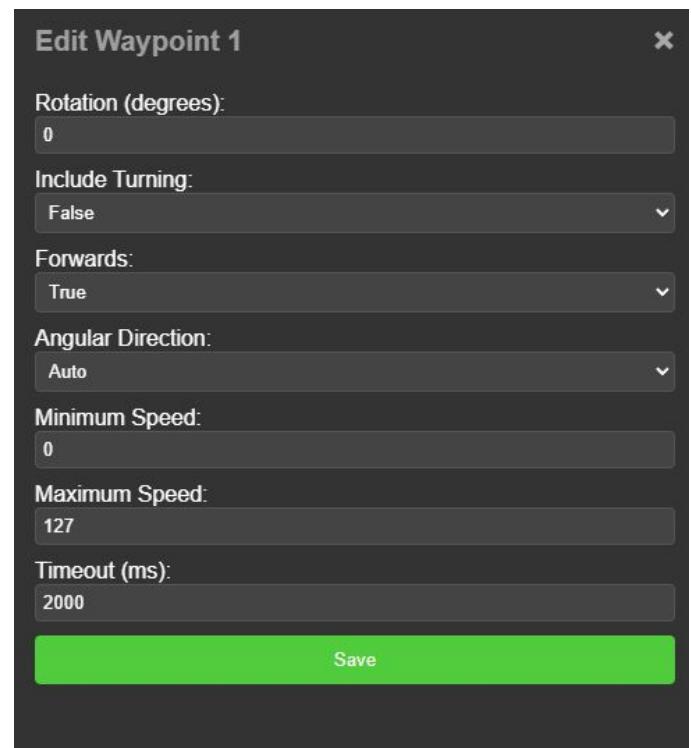
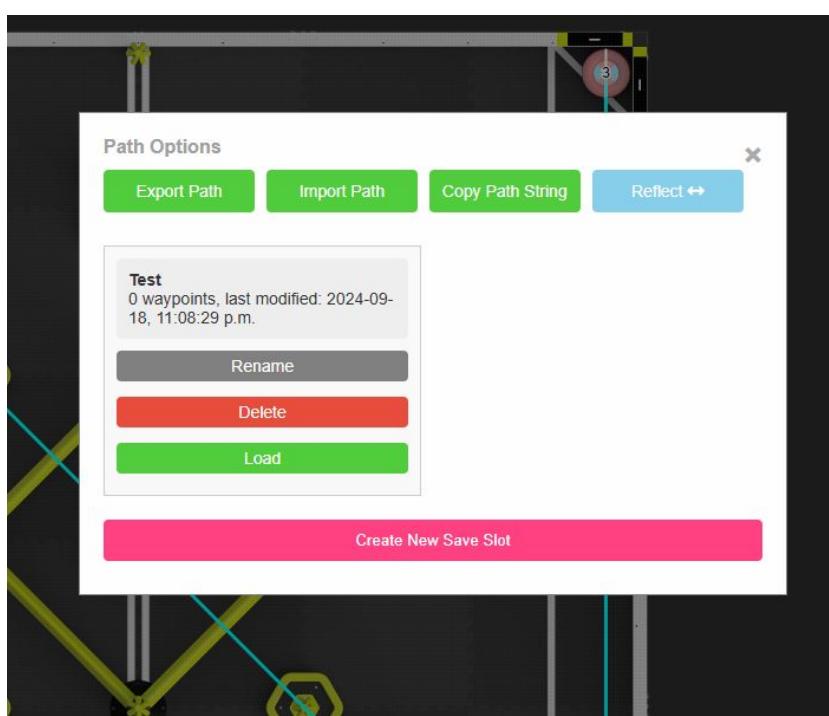
- This task *could* be accomplished by looking at the field and guessing the distances the robot needs to travel
 - ◆ However, such a method is tedious and time-consuming whilst being imprecise
 - ◆ By using an external tool to visualize the field and draw waypoints, we can speed up our autonomous routine development process and visualize the path of our entire auton.

We've decided to make a website in HTML/JS for this purpose, as it is easily accessible on multiple platforms regardless of operating system.

- Uses the p5.js library to handle graphics by rendering the field top-down
- User can click anywhere on the field to add a waypoint, and a chain of waypoints is interpreted as a path
- The coordinates of the waypoints on the computer screen are **converted to life-scale inch measurements for our robot**, which is then outputted by the website as a **string of code commands** to be pasted into our codebase



- We can set the properties of each waypoint (max speed, rotation, exit timeout)
- Since the path generator is a website, its working memory can be cached into the user's browser and restored for later use
- We also added the functionality to export paths to files, which can be shared to other computers
 - ◆ At many tournaments, we often do not have access to the internet which would limit our access to the path planner website. We used service workers so that it could be used regardless of network connectivity



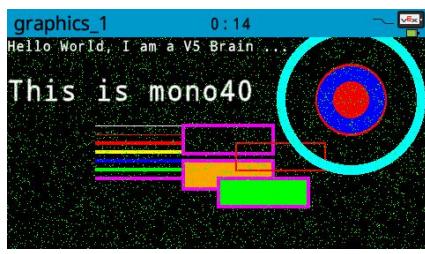
```

18 if (selectedConfig === 'libKS-mtpoint') {
19   let firstWaypoint = path[0];
20   // convert to trig bearing
21   let originAngle = ((360 - firstWaypoint.angle) + 90) % 360;
22
23   let code = `// libKS MTPoint v0.1\n`;
24   code += `Starting point: ${((firstWaypoint.x - (canvasSize / 2)) / canvasSize * 144).toFixed(2)} in, ${((firstWaypoint.y - (canvasSize / 2)) / canvasSize * -144).toFixed(2)} in\n`;
25
26   for (let i = 0; i < path.length; i++) {
27     let waypoint = path[i];
28     let translatedX = (waypoint.x - firstWaypoint.x) / canvasSize * conversionFactor;
29     let translatedY = (waypoint.y - firstWaypoint.y) / canvasSize * conversionFactor;
30
31     // Rotate point (x and y flip here for complicated reasons)
32     const rotatedX = (translatedX * Math.sin(degreesToRadians(originAngle))) +
33       translatedY * Math.cos(degreesToRadians(originAngle));
34     const rotatedY = (translatedX * Math.cos(degreesToRadians(originAngle))) -
35       translatedY * Math.sin(degreesToRadians(originAngle));
36
37     code += `chassis.moveToPoint(${rotatedX.toFixed(2)}, ${rotatedY.toFixed(2)}, ${path[i].timeout}, {forwards = ${path[i].forwards}, .maxSpeed = ${path[i].maxSpeed}, .minSpeed = ${path[i].minSpeed}}); // Point ${i + 1}\n`;
38
39     if (waypoint.includeTurn === true) {
40       // Convert angle to be relative to the path
41       let relativeAngle = (waypoint.angle - firstWaypoint.angle) % 360;
42       if (relativeAngle < 0) relativeAngle += 360;
43
44       if (path[i].angularDirection === "auto") {
45         code += `chassis.turnToHeading(${relativeAngle}, ${path[i].timeout}); // Point ${i + 1}\n`;
46       } else if (path[i].angularDirection === "clockwise" && i != 0) {
47         code += `chassis.turnToHeading(${relativeAngle}, ${path[i].timeout}, {direction = AngularDirection:CW_CLOCKWISE}); // Point ${i + 1}\n`;
48       } else if (path[i].angularDirection === "counter-clockwise" && i != 0) {
49         code += `chassis.turnToHeading(${relativeAngle}, ${path[i].timeout}, {direction = AngularDirection:CCW_COUNTERCLOCKWISE}); // Point ${i + 1}\n`;
50       }
51     }
52
53   }
54 }
55
document.getElementById('code-output').textContent = code;

```

We've been looking into adding support for **autonomous selectors**, **debugging interfaces** and **telemetry visualizations** into our codebase. We envision these features to be graphically displayed on the V5 Screen, with the possibility for inputting actions using the touchscreen or the V5 controller.

In our research, we found **3** major graphics libraries that can be used to display text, pictures, etc on the brain:

Library	Pros	Cons
LLEMU (PROS) 	<ul style="list-style-type: none"> Extremely simple to implement with minimal code required for basic text display Very lightweight with minimal processing overhead Perfect for simple debugging output and basic status displays during development 	<ul style="list-style-type: none"> No graphics capabilities beyond basic text characters Cannot display images, custom fonts, or any visual elements beyond text Appears outdated and unprofessional compared to modern interface expectations
LVGL (not vex-specific) 	<ul style="list-style-type: none"> Professional-grade GUI framework with modern, polished visual appearance Comprehensive widget library Active community support with regular updates 	<ul style="list-style-type: none"> Large learning curve requiring familiarity with object-oriented concepts Higher memory usage that may impact performance May be overkill for simple status displays or basic debugging
pros::screen (PROS) 	<ul style="list-style-type: none"> Direct pixel-level control allowing for completely custom graphics and layouts Flexibility to create unique interfaces tailored specifically to our needs 	<ul style="list-style-type: none"> Requires significant programming effort to build even basic GUI elements from scratch Time-intensive development process, especially for complex multi-view interfaces

Over the past week, we have chosen **LVGL** as our library of choice for its versatile abilities and extensive documentation that rivals VEX-targeted solutions like PROS. We've already started developing our template. Currently, it can:

- Configure our alliance colour for the match (to automatically sort rings based on colour)
- Configure which autonomous routine to run
- Check various system diagnostics (temperature, odometry coordinates, etc.)
- Show videos and animations

Each function is put as its own “page” with menu system to swap between them. To make this process simpler and remove adding needless boilerplate, we **utilize the robodash library** (<https://github.com/unwieldycat/robodash>) to add a page swap menu, with us writing the code for all the pages ourselves. Below are some screenshots and code snippets:

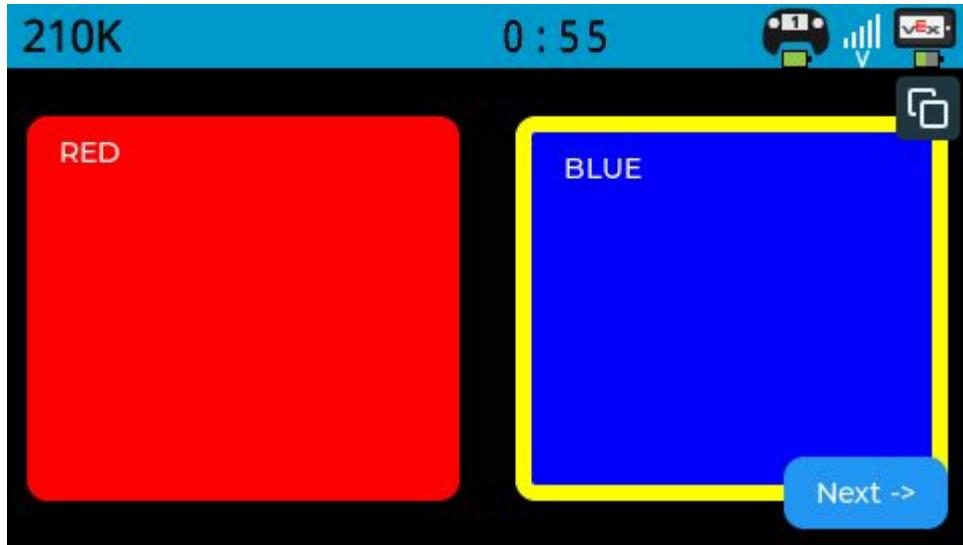
```
● ● ●
210K-HighStakes-2025 - rdconfig.cpp

23 static void btn_event_cb(lv_event_t *e) {
24     if (lv_event_get_code(e) == LV_EVENT_CLICKED) {
25         lv_obj_t *btn = lv_event_get_target(e);
26         alliance = ((btn == red_btn) ? "red" : "blue");
27         highlight_button(btn);
28
29         if (!next_btn) {
30             next_btn = lv_btn_create(lv_obj_get_parent(btn));
31             lv_obj_set_size(next_btn, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
32             lv_obj_align(next_btn, LV_ALIGN_BOTTOM_RIGHT, -10, -10);
33             lv_obj_add_event_cb(next_btn, [](lv_event_t *e) { gui_selector.focus(); }, LV_EVENT_CLICKED, NULL);
34             lv_label_set_text(lv_label_create(next_btn), "Next ->");
35         }
36         lv_obj_clear_flag(next_btn, LV_OBJ_FLAG_HIDDEN);
37         console.printf("Alliance set to: %s\n", alliance.c_str());
38     }
39 }
```

Handles button clicks for the home screen and alliance selection screen

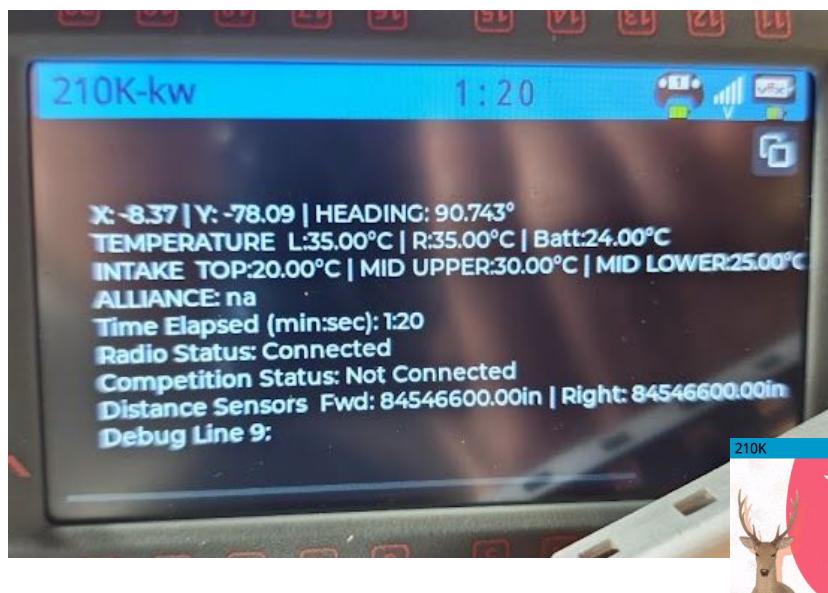
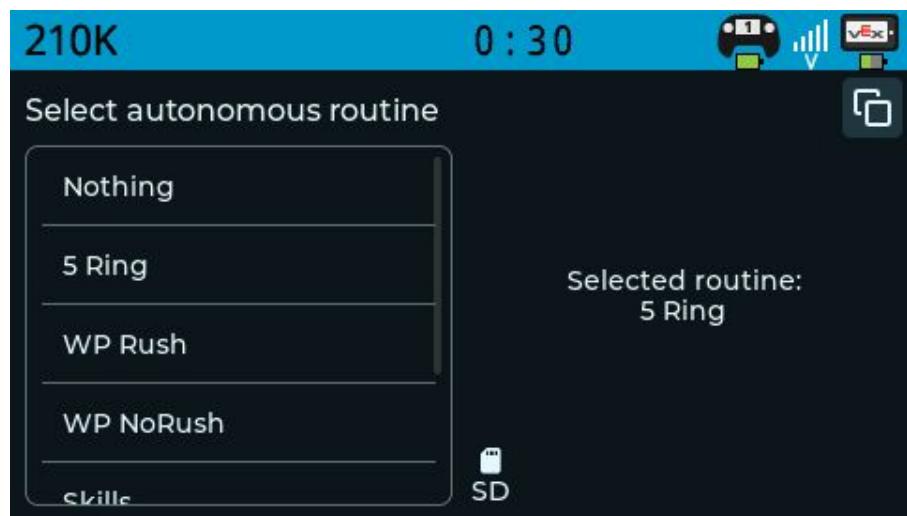


Our autonomous configuration procedure makes it easy even for **non-programmers** to set up autonomous routes. By clicking “Setup Auton” on the home page, they are guided through a series of screens shown below:



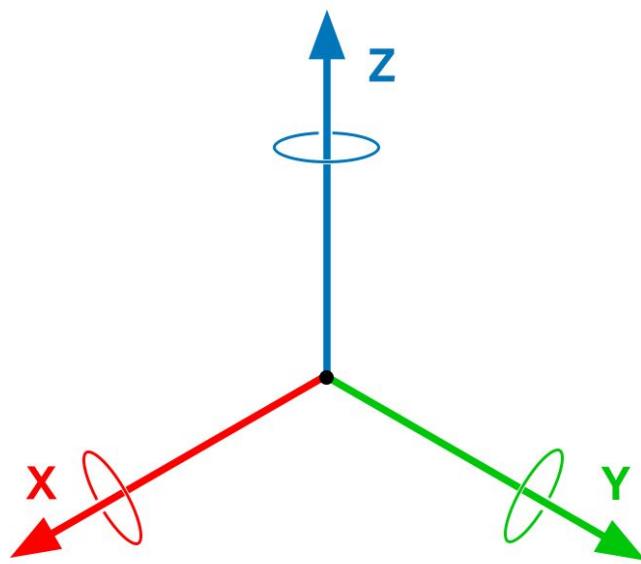
Our alliance selection mechanism. This screen will show up whenever our robot is connected to field control and it powers our autonomous mirroring system alongside our automatic colour sorting algorithm.

An intuitive autonomous routine selector that can be configured after setting the alliance colour. Selections will be automatically saved into a SD card and restored whenever the program runs.



Once the autonomous period starts, we do not need to display any crucial information on our brain, so we render a placeholder video. Special telemetry like motor temperatures are accessible via its own “Sensors” page

Continuing from last season, our team has been looking into alternative methods of calculating robot coordinates aside from physical wheels. While our tracking pods work great with our odometry code, we'd still like to have backup methods in case of failure. Last season, this was especially a priority as we still relied on our drivetrain motors for odometry. When our robot collided with other bots or field elements, its odometry coordinates got thrown off and it could not finish its autonomous movement as planned. This is one of the many investigations we performed into developing **innovative and novel autonomous systems**.



Images via VEX Knowledge base

The VEX Inertial Measurement Unit (IMU) is a sensor capable of measuring acceleration and gyroscopic motion. Odometry relies on measuring travelled distances to calculate coordinates. Acceleration cannot be directly fed into the odometry formula, however it is important to note that acceleration is the 2nd derivative of motion, since it is a measure of “the rate of change of the rate of change of displacement” (m/s/s, meters per second per second). Thus, given the instantaneous acceleration of a robot measured continuously, it is *technically* possible to double integrate the acceleration values to find the distance the robot has travelled.

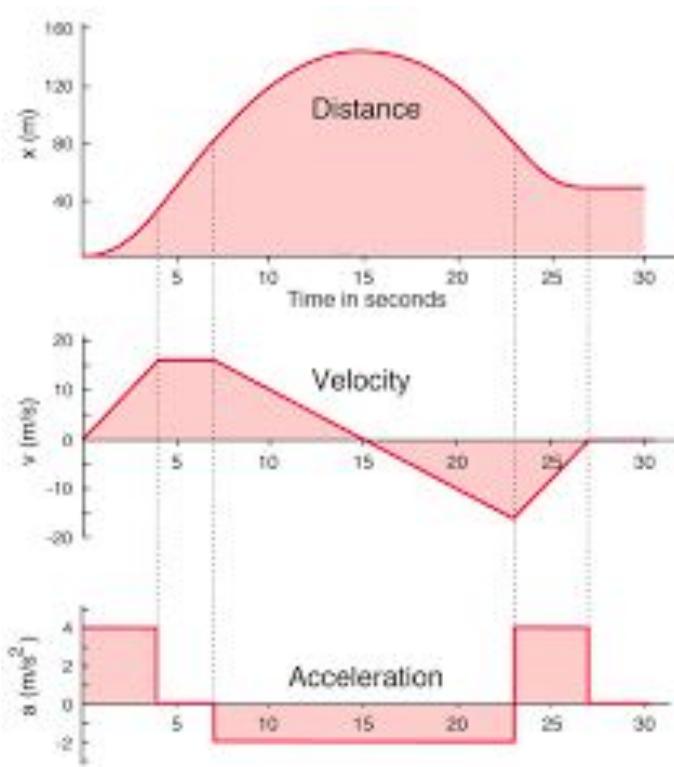
Our investigation started with the usage of a traditional odometry tracking system. We had our robot move forward 100 inches, which we tracked using our physical wheel movements and our PID. Using a SD card, we logged down our current odometry position and the instantaneous acceleration measured through the IMU at an interval of 10ms. Using various kinematics equations, we were able to use a spreadsheet to model the acceleration curve for our specific robot to its displacement.

```

39 int timer = 0;
40 std::deque<double> buffer;
41 int windowSize = 20; // Adjust this value as needed
42 FILE *save_file;
43 save_file = fopen("/usd/log.txt", "w");
44 while (true) {
45     pros::c::imu_accel_s_t accel = inertial.get_accel();
46     double filteredAccelY = strait::median_filter(buffer, accel.y, windowSize);
47     printf("%d,%f,%f\n", timer, accel.y, chassis.getPose().y);
48     fprintf(save_file, "%d,%f,%f\n", timer, filteredAccelY, chassis.getPose().y);
49     timer += 1;
50     pros::delay(1);
51     if (timer > 10000) {
52         fclose(save_file);
53         break;
54     }
55 }

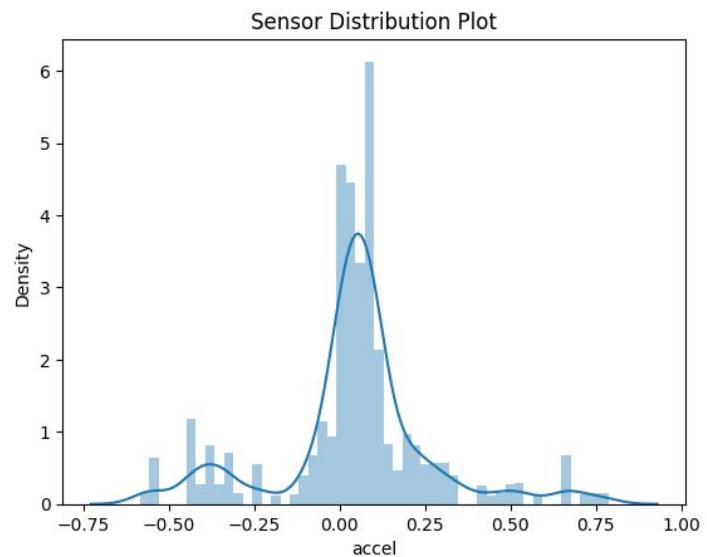
```

Every 1ms, the code will poll the latest acceleration value for the IMU and create a log file on a SD card, we've also setup a basic median filter, which will take the median of all gathered values so far and discard any extraneous readouts that stray too far from the past recorded values and are evident outliers. The filtered values are saved both to the file and to a live console readout on my laptop, where I can verify everything is working is intended. Once 10 seconds have elapsed since the robot started moving, the log is saved and can be imported into spreadsheet software as comma-separated values for external processing.



Firstly to verify the sensor was reporting accurate data, we plotted its density. It shows telltale signs of following a normal distribution curve, which is what we'd expect from an inertial sensor.

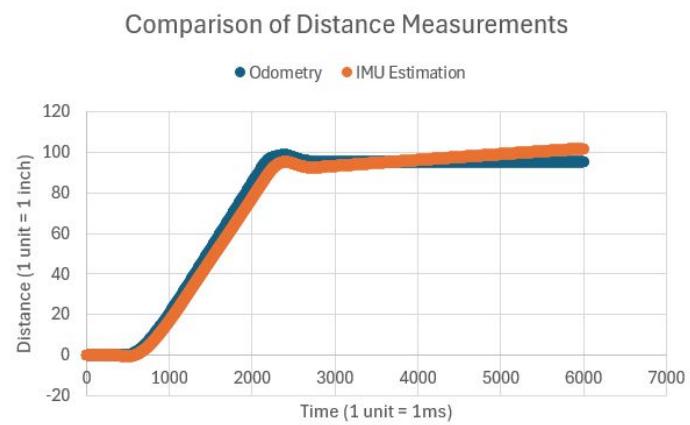
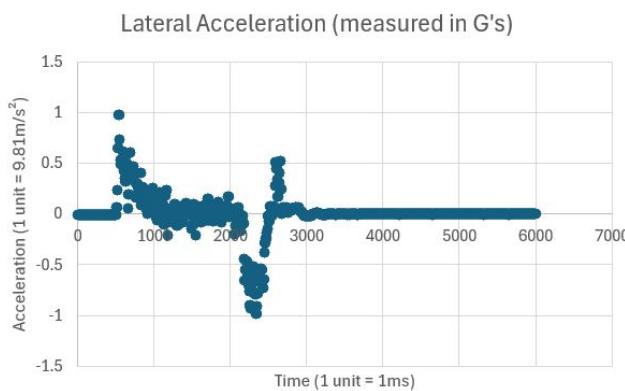
time	accel	dist	velocity(m/s)	V through D
0	-0.0111	-0.001	-0.00010878	-4.2828E-06
1	-0.0111	-0.001	-0.00021757	-1.2848E-05
2	-0.0111	-0.0009	-0.00032635	-2.5697E-05
3	-0.0111	-0.0009	-0.00043513	-4.2828E-05
4	-0.0111	-0.0009	-0.00054392	-6.4242E-05
5	-0.0111	-0.0009	-0.0006527	-8.9939E-05
6	-0.0111	-0.0009	-0.00076148	-0.00011992
7	-0.0111	-0.0009	-0.00087026	-0.00015418
8	-0.0089	-0.0009	-0.0009575	-0.00019188
9	-0.0089	-0.0009	-0.00104473	-0.00023301
10	-0.0089	-0.0009	-0.00113196	-0.00027757
11	-0.0089	-0.0009	-0.00121919	-0.00032557
12	-0.0089	-0.0009	-0.00130642	-0.00037701



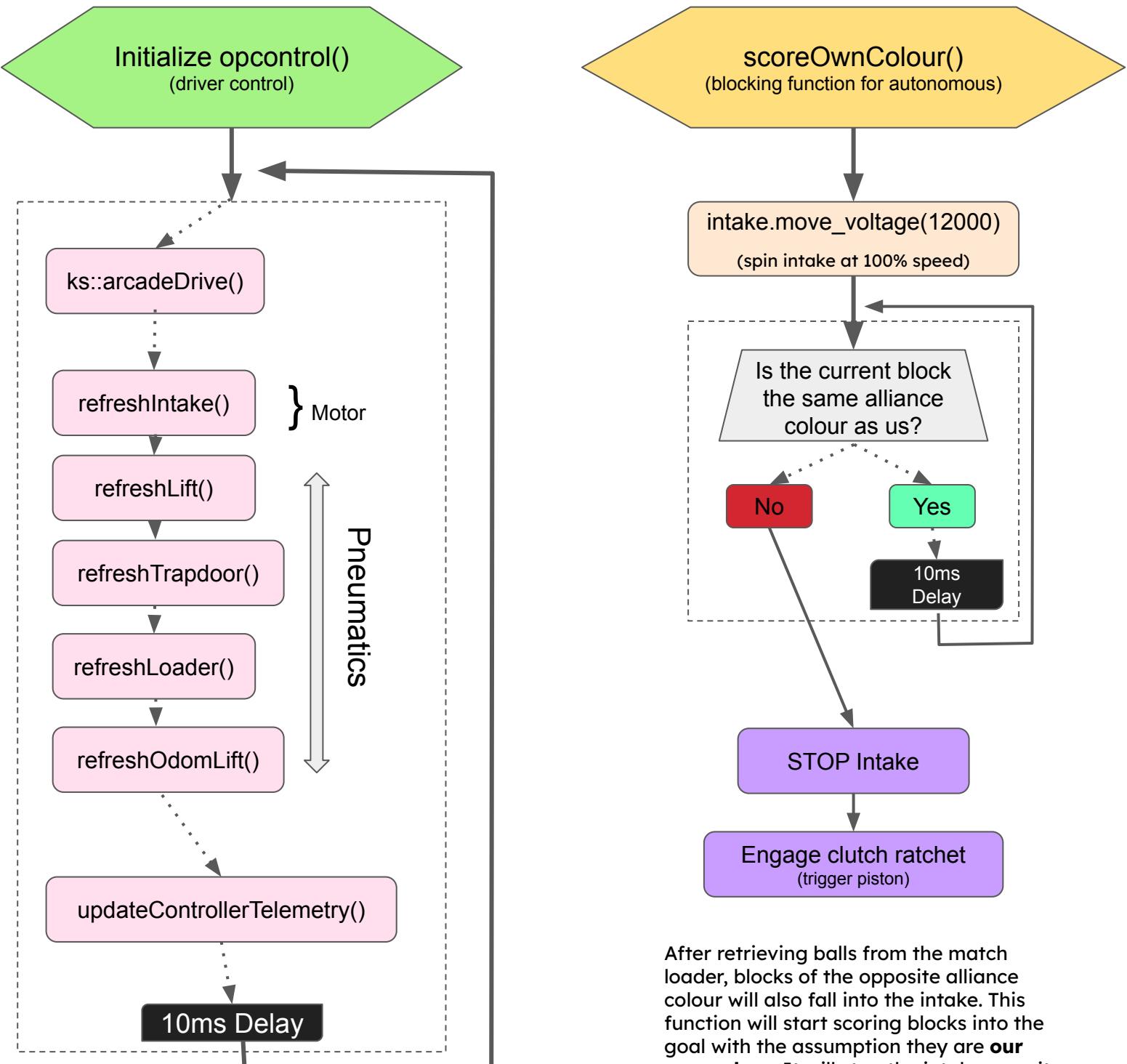
We use some tricky maths to account for the fact that these measurements take place over an interval. Using the kinematics formula $v = u+at$, we can recognize that the instantaneous velocity for a given time interval is equal to the sum of all previous acceleration divided by the time interval since our current velocity is also based on previous acceleration.

The acceleration readings are in the amount of G's ($1G = 9.81m/s^2$) and the time interval between readings is 1ms ($1s = 1000ms$). The instantaneous velocity calculation becomes $v = (\text{sum of previous acceleration}) * 9.81m/s^2 / 1000ms$.

Now, we can draw a relation between our instantaneous velocity and time, which forms a curve. We can simply integrate the area under this curve using **Rectangle Approximation** to find our distance travelled up to that point by finding its product with a constant. Our estimated distance (orange) was within 5 inches of our actual distance reported by our odometry wheels. The biggest challenge facing our accuracy was our acceleration changing despite our bot staying still at the end, which we hope to resolve with a Kalman Filter.



These flowcharts outline the control loop for our **driver control** code and our **autonomous scoring macro** that prevents scoring blocks for the opposite alliance.



Each **function** contains control logic for its corresponding subsystem. When run, it checks for some controller input and adjusts the state of the subsystem accordingly. (e.g. pneumatics functions are binary toggles)

After retrieving balls from the match loader, blocks of the opposite alliance colour will also fall into the intake. This function will start scoring blocks into the goal with the assumption they are **our own colour**. It will stop the intake once it detects a oppositely-coloured block **to prevent scoring it**.

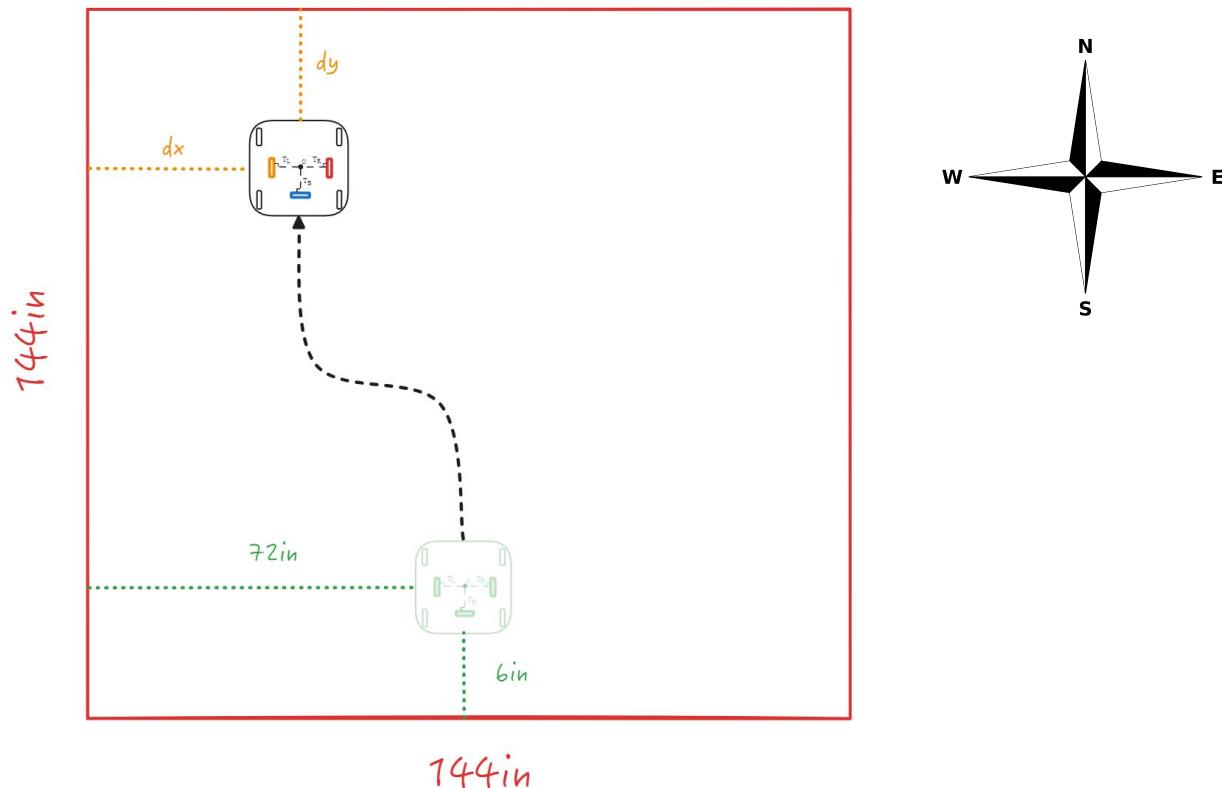
During the autonomous skills challenge, we notice that our usage of odometry is not perfect. Partially due to inconsistency with the IMU and rotation sensors, our odometry coordinates are not fully accurate past the 30-second mark in our routine.

One solution is to use distance sensors to “reset” our local position relative to our starting position by determining our absolute position on the field.

Assume our robot’s starting position is the green \times . It will travel autonomously until it reaches its current position, indicated by Δ . If two distance sensors are mounted perpendicular to each other on the chassis, we can determine its distance to the closest field walls, dx and dy . The field walls are 144in x 144in, so $(dx, 144 - dy)$ must represent the coordinates of our bot relative to the southwestern corner of the field (cardinal directions are only used to simplify this explanation).

Since we also know the position of the starting position (\times) relative to this same corner, we can deduce that the position of the robot relative to \times is $(72 - dx)$ in the horizontal component, and $[(144 - dy) - 6]$ in the vertical component.

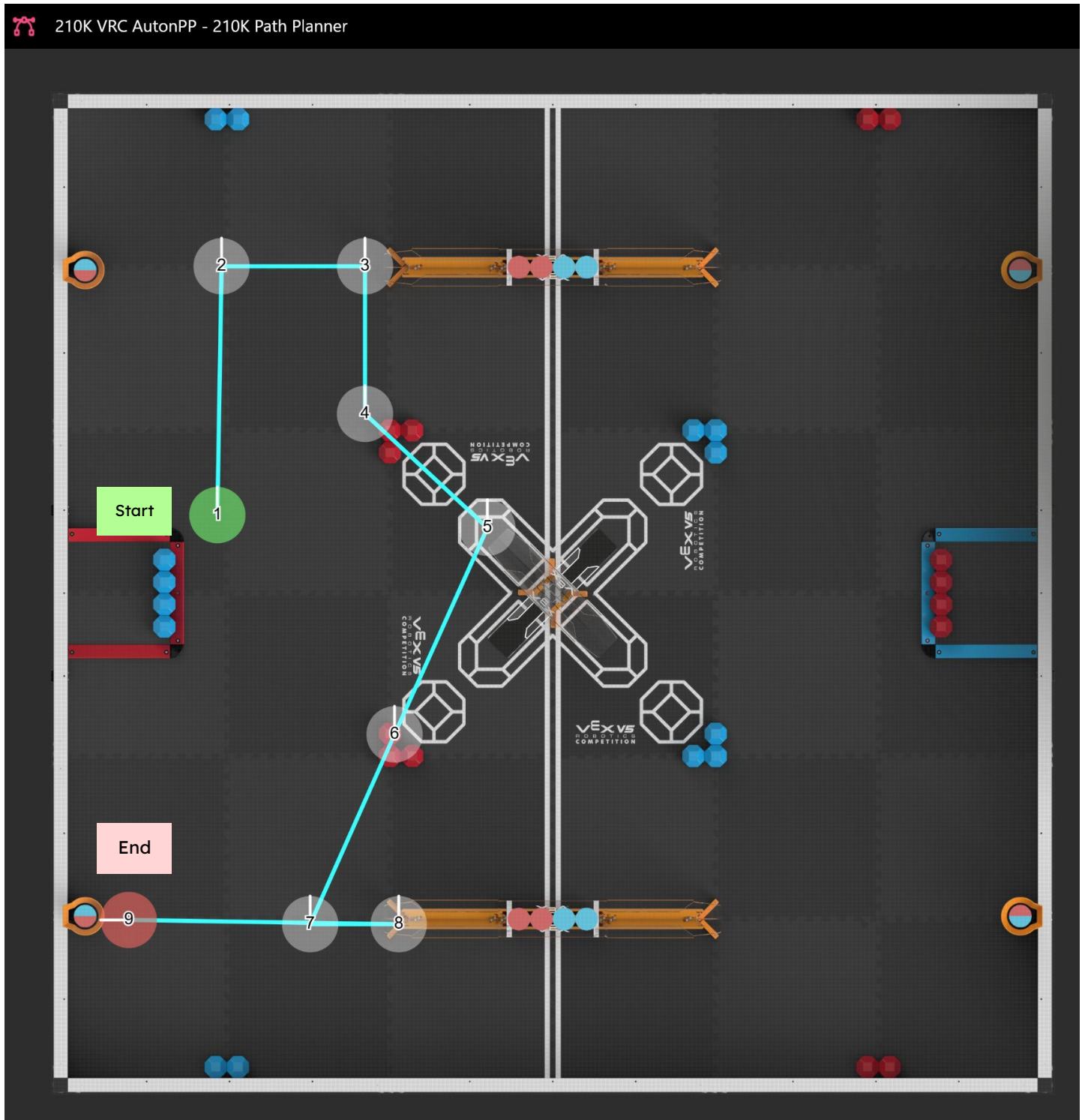
This process can be applied to any corner of the field given we know how far \times is from that corner, easily allowing us to reset our robot’s position with two constants.



Top-down view of the VEX competition field. The field perimeters and their lengths are labelled in red.

A **major factor** in Push Back which will likely decide the outcome of a match is the scoring capability of autonomous routines. Having a high-scoring autonomous will increase the likelihood of winning a match through the **autonomous bonus** and improve rankings by achieving the **autonomous win point**.

Below is our routine that achieves AWP **by ourselves** (7 blocks scored across 3 goals and 3 blocks removed from match loader).



Autonomous Win Points (AWPs) play a critical role in overall tournament performance, especially during qualification matches where securing early ranking points can significantly improve standings. Achieving many AWPs requires the robot to meet a specific set of autonomous objectives consistently and efficiently.

Today, our goal was to develop and tune a **Solo AWP (SAWP)** routine, which demands handling a higher number of game elements compared to the standard AWP sequence used in local competitions. This version involves more precise movements, increased cycle speed, and tighter synchronization between intake, indexing, and shooting mechanisms.

- Today, we worked extensively on fine-tuning the pathing and timing of this routine
- However, despite the significant effort invested, the results were **inconsistent and unreliable**
- Often missed critical alignment points or failed to complete the routine within the allotted time. These inconsistencies made the SAWP path unsuitable for use in an actual match at this stage.

Given our **time constraints** leading up to the next competition, we concluded that it would be more effective to **pivot toward a simplified autonomous path**. This adjusted version would prioritize reliability and repeatability over complexity, allowing us to secure at least a standard AWP consistently.

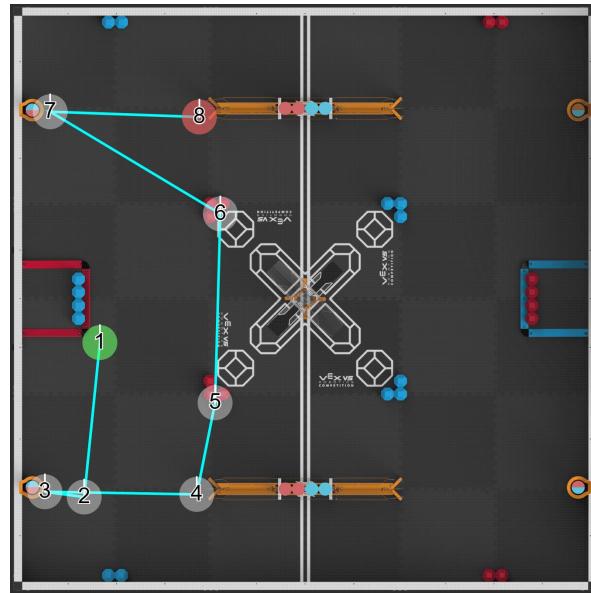


Figure 1: Target SAWP path for a 13 ball SAWP