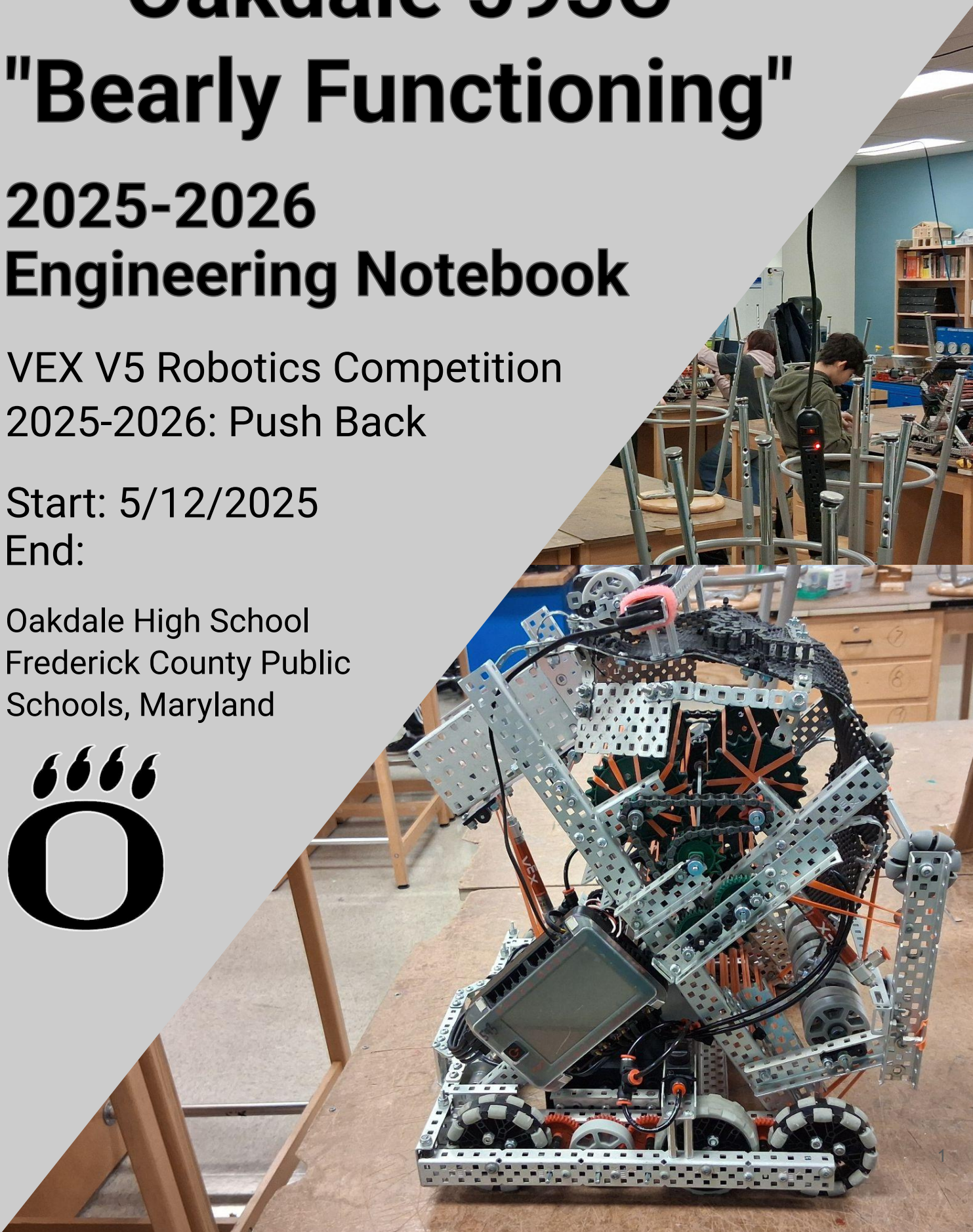
Oakdale 593C "Bearly Functioning"

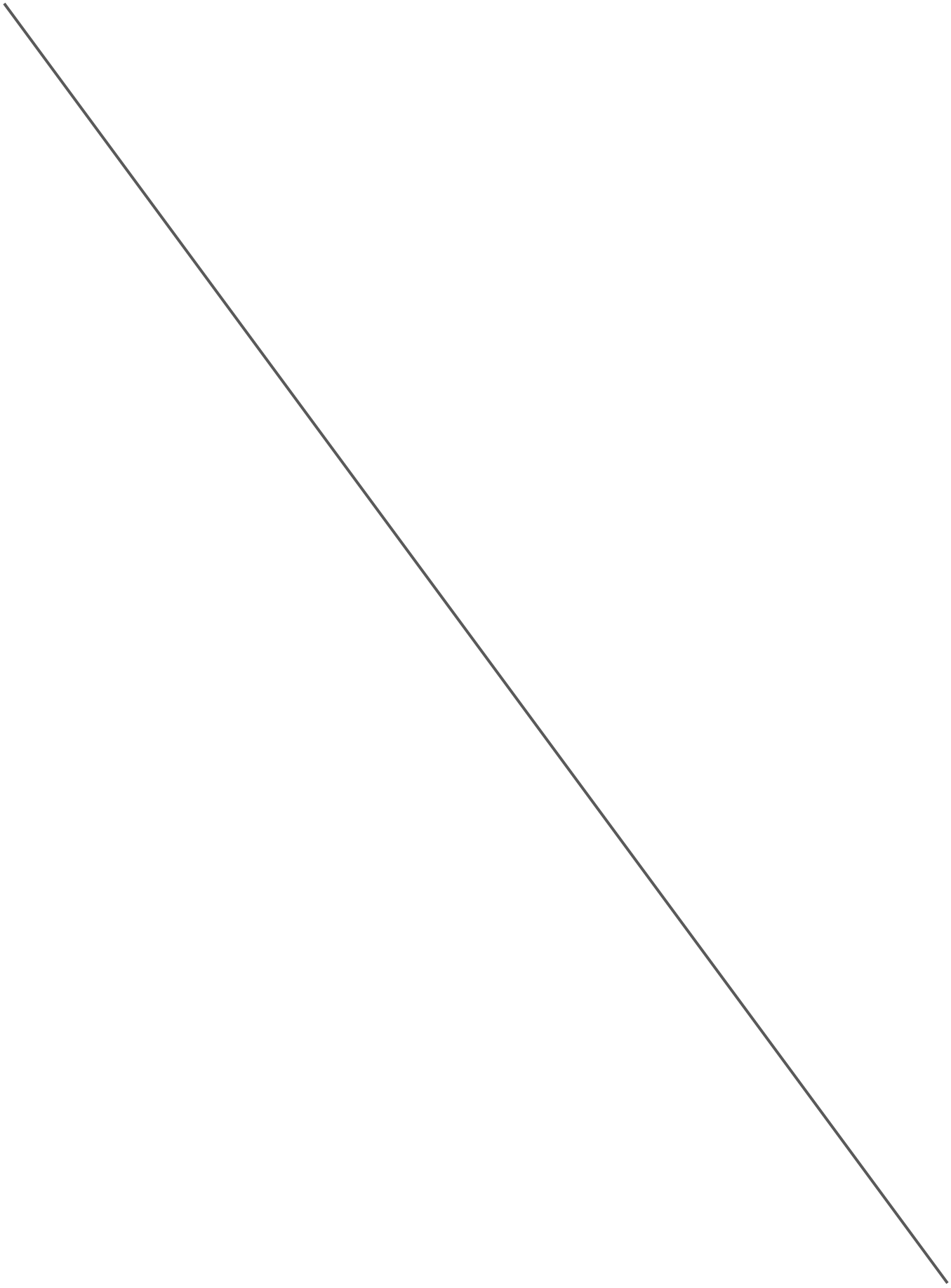
2025-2026 Engineering Notebook

VEX V5 Robotics Competition
2025-2026: Push Back

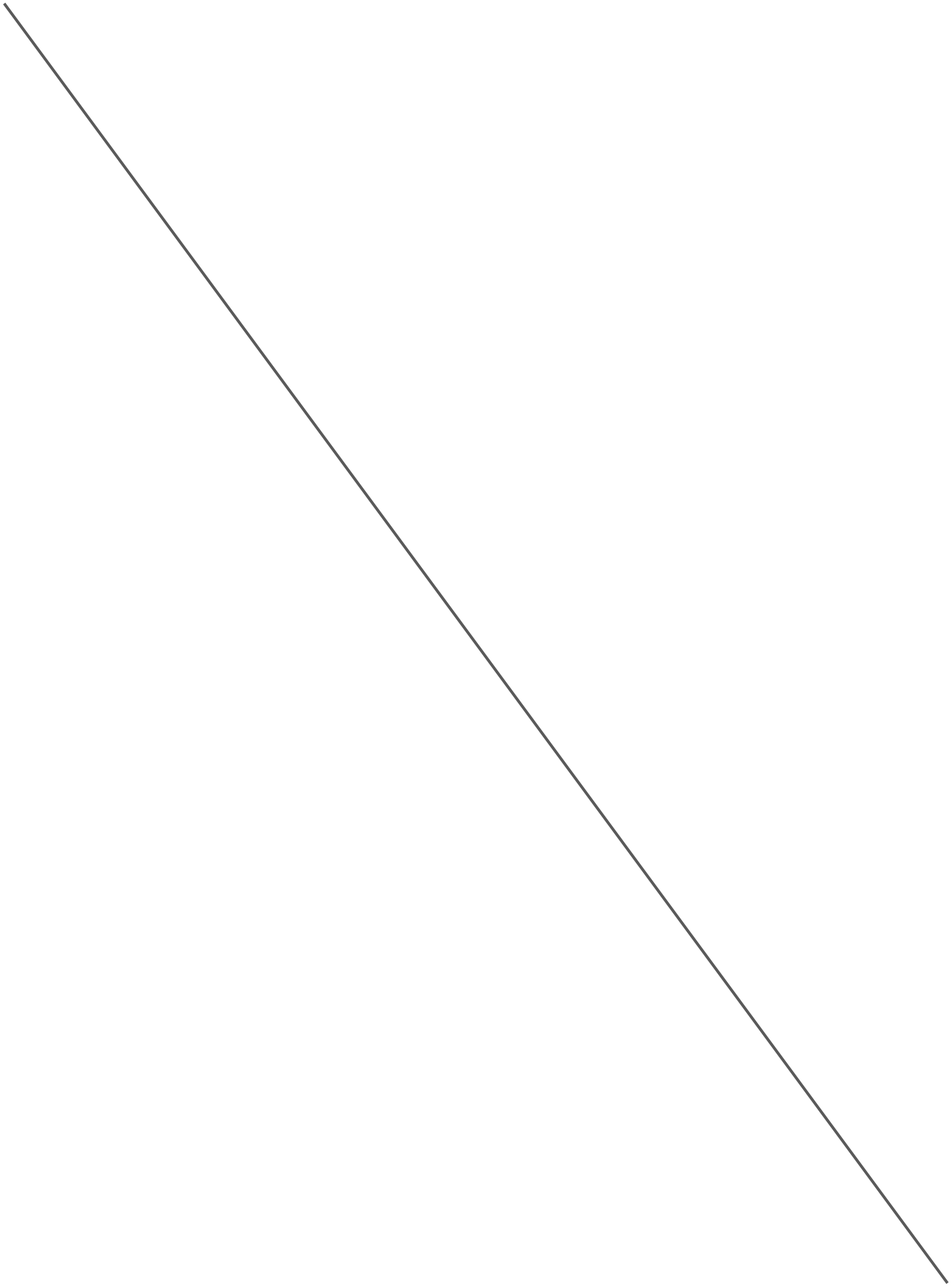
Start: 5/12/2025
End:

Oakdale High School
Frederick County Public Schools, Maryland





Innovate form goes here



Date	EDP Step	Description	Page #

Date	EDP Step	Description	Page #

Date	EDP Step	Description	Page #

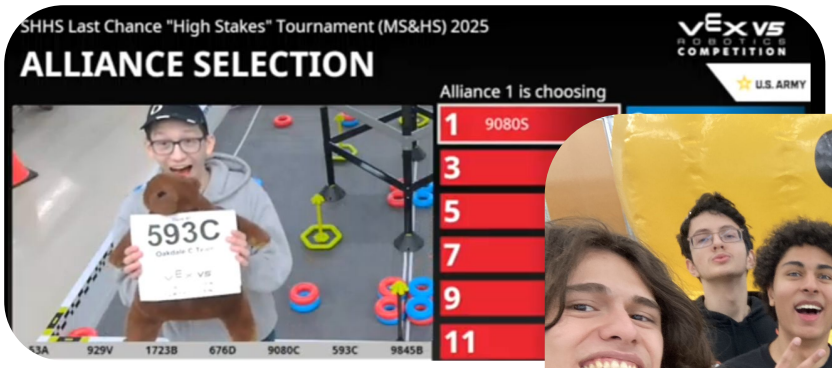
Date	EDP Step	Description	Page #

Thank you for volunteering your time to be a judge and read through our notebook! Without judges, this event, and VEX Robotics as a whole, would not be able to exist, and this amazing opportunity would not be available to the many teams competing today.

Within this notebook, you will find the full documentation of the engineering design process we followed to design a bot for this year's competition. We are proud of what we have accomplished in previous years, and are looking forward to what we can build this year.

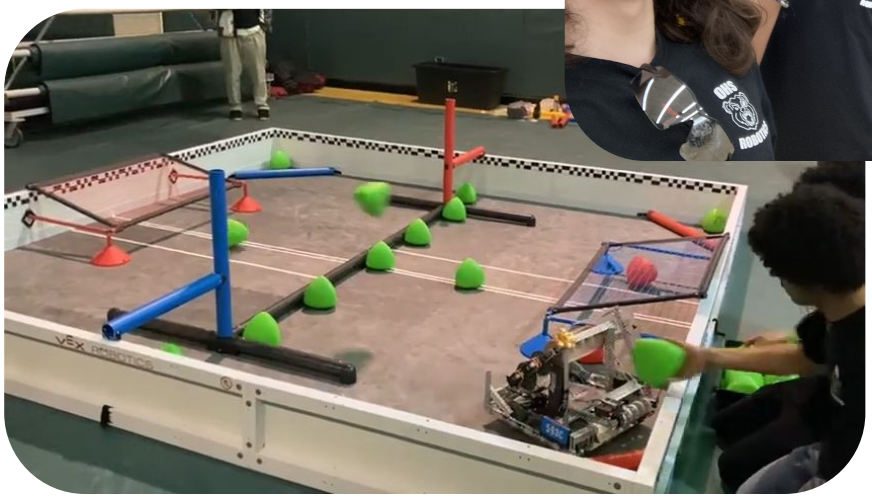
One goal of this notebook this year to allow for clearer documentation of our processes, and removing fluff - this is a notebook, not a diary. In addition, clearer citations, explanations of what went wrong, and clearer iteration processes - we are not perfect, we do borrow ideas and run into roadblocks.

Thank you! We hope you enjoy today's competition!
Below are some photos from highlights of previous seasons we enjoyed!

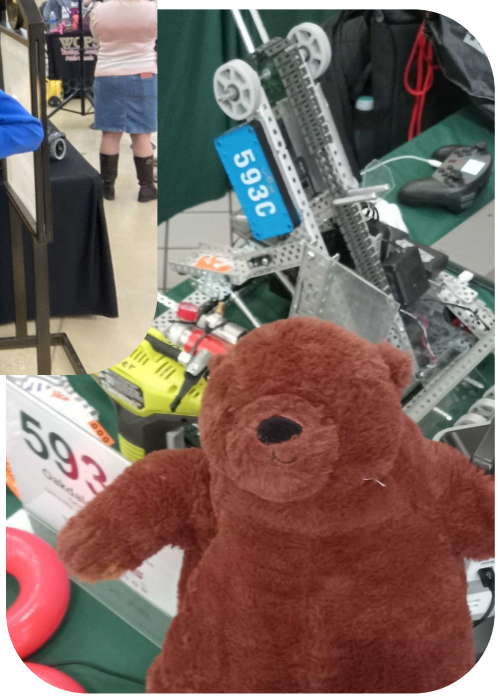


593C at SHHS, 2025!

593C at Valley Mall, 2024!



593C at States, 2024!



593C at Valley Mall, 2025!

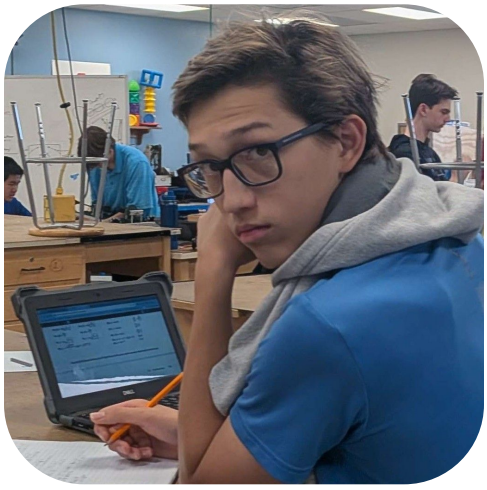
Team 593C is a semi-rookie team. We are a team of 5, with Rowan as our captain. As a team, we compete primarily at the local level, but have made it to the States level competition a few times in the past years. We are looking forward to this season!

Rana Hashemi - Sophomore
1st Year Competing
Roles: Builder, Driver
Strengths: Driving, willing to help others

Oliver Lazarus - Sophomore
1st Year Competing
Roles: Builder
Strengths: Problem solving, focus, and ability to learn

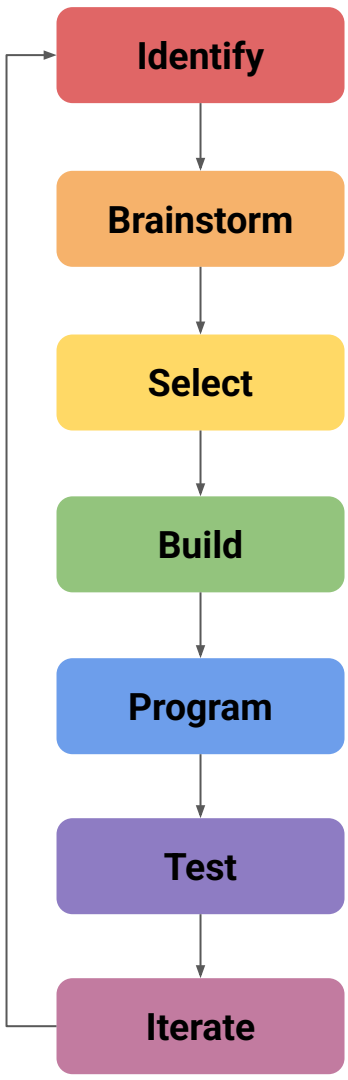
Dennis Chang - Junior
1st Year Competing
Roles: Builder, Programmer
Strengths: Brainstorming, comedic relief

Katie Chen - Junior
2nd Year Competing
Roles: Builder, Driver
Strengths: Problem solving, completing tasks for others in an attempt to be productive



Rowan Wood - Junior
3rd Year Competing
Roles: Captain, Builder, Programmer, Driver
Strengths: Experience, knowledge, too much time watching videos

This notebook utilizes colors to designate its sections, utilizing the Engineering Design Process (EDP) outlined in the VEX Engineering Notebook Rubric to guide our design process.



"Identify the Problem" - Identifies challenges of the game/design and explains how they can be solved

"Brainstorm, Diagram, or Prototype Solutions" - Create solutions to the problem, then prototype the best solutions

"Select Best Solution and Plan" Select a solution, explain why and continue to plan from there

"Build and Program the Solution" - Create the solution that was chosen, and describe the design and building process

"Build and Program the Solution" - Program the solution that was chosen, document and explain the programming process

"Test Solution" - Records testing of the solution, including test results and a summary.

"Repeat Design Process" - Records, what went right, wrong, and what we can improve. Explains how we plan to improve, too.

The Engineering Design Process is cyclic in nature, allowing for improvement of systems over time, through iteration. After identifying a problem, brainstorming, designing, and building a solution, we test our solution to identify what went right, what went wrong, and how we can improve. We then repeat the Engineering Design Process, to improve our solution. We aren't perfect!

In addition to documenting the Engineering Design Process, this notebook will also document various other functions of our team, including team management, strategy, and reflections.

Notebook — **"Notebook"** - These pages include information about the formatting or setup of the notebook, along with notes and letters.

Team — **"Team Decisions"** - These pages will document whole team decisions, including budgets, meeting dates, and team members

Daily Log — **"Daily Logs"** - These pages will include a summary of the activities completed during team meetings, as necessary
This is an engineering notebook, not a diary!

Management — **"Time and Team Management"** These pages will document the tasks each team member is in charge of, and the time allotted.

Strategy — **"Game Strategy"** - These pages will document our gameplay strategy during events, including routing during skills matches

Events — **"Event Plan and Analysis"** - These pages will analyze our performance before and at competitions, and decisions made before or during competitions.

Some text may be placed inside of a box, to emphasize those points.

Any borrowed content, such as photos or ideas, will be cited, as possible.
All pages will include a footer, with the names of the team members who created that page, the date the page was created, and the page number.

Team Goals

- Improved notebooking
  - An issue we had in previous years was not documenting our choices and designs the day of, but rather waiting a few days. This means we had no photos to include on pages.
  - Minimal photos meant walls of text, which are hard to read.
  - CAD diagrams of what we are building allows for figuring out issues before we run into them, and allow for easier explanation of what we are building.
  - Please don’t pull multiple overnighters notebooking again this season!
- Qualify for States
  - In previous years, this has been a major goal for us. Starting in 2023-24 (Over Under), the first year our C team has attempted to be competitive, we qualified for states, similarly to last year, 2024-25 (High Stakes).
  - We believe that qualifying for this competition is an important goal to have, and that doing well up to and at States will be important to prove that the continued existence of our robotics club is important!
- Focus on Skills
  - We don’t have anywhere near the amount of time other teams have to be competitive in competition matches, but we do have time to focus on skills!
  - Skills has been the primary way any one of our teams have qualified for states in the past.
- Earn an award
  - It would be really cool if we won an award this year!
  - In 2024 (Over Under), we earned a Skills Champion award at SHHS. We should shoot to do this again!

Budgeting and Sponsors

- At our first team meeting, we developed a rough budget and started putting items onto our team expenses sheet.
- We found that this year we will need ~$4,500 to successfully buy parts and pay fees necessary for a competitive team this year.

Item	Qty.	Price per Unit	Sub-Total
Field Elements	1	$589.99	$599.99
Registration	4	$150, $100	$450
Competitions	12	$100	$1200
States	4	$150	$600
Parts			$1500
		Total:	$4,349.99

- All members on our team have been requested to either find a sponsor for at least $250, or pay a $50 fee to participate.
  - If a minimum of 16 members bring in $250, this will only cover $4000. Any contributions will be welcome!
- We hope to create a flier and reach out to local businesses ASAP. We have parts, but it'll be expensive to attend more events, like we plan this year.
- We also discussed pursuing grants again this year, but we may have missed many of the submission dates for this season.

Events

We plan to attend ~4 local events before States this year! Most events will be located around Western Maryland, due to shorter travel times - we carpool to events, we don't rent buses. Below are a few of the competitions we plan to attend, screenshotted from Robotevents.

AOE Robotics Challenge

In-Person

Status: Early-Bird is Open, Standard on 27-Sep-2025
Spots Open: 50%+
Date: 8-Nov-2025
Event Region: Maryland

Event Code: RE-V5RC-25-0506
Type: Open Tournament
Location: 22401 Brick Haven Way, Clarksburg, Maryland, 20871, United States

SHHS VEX V5 Robotics Competition "Push Back" Tournament 2026
"Bring What You Have"

In-Person

Status: Early-Bird is Open, Standard on 4-Oct-2025
Spots Open: 50%+
Date: 15-Nov-2025
Event Region: Maryland

Event Code: RE-V5RC-25-1465
Type: Open Tournament
Location: 1101 South Potomac Street, Hagerstown, Maryland, 21740, United States

Will we be ready by november for these competitions?

South Hagerstown Rebel Rumble (MS&HS) 2025

In-Person

Status: Early-Bird opens on 15-Oct-2025
Spots Open: -
Date: 13-Dec-2025
Event Region: Maryland

Event Code: RE-V5RC-25-1461
Type: School-Based Tournament
Location: 1101 South Potomac Street, Hagerstown, Maryland, 21740, United States

Western MD "Push Back" Valley Mall Invitational 2026 (MS&HS)
VEX V5 Robotics Competition

In-Person

Status: Opens on 12-Nov-2025
Spots Open: -
Date: 10-Jan-2026
Event Region: Maryland

Event Code: RE-V5RC-25-1464
Type: Invitational Tournament
Location: 17301 Valley Mall Rd, Hagerstown, Maryland, 21740, United States

SHHS "Push Back" SKILLS ONLY Tournament 2026
SKILLS ONLY - Blended VEX V5 Robotics Competition

Skills Only

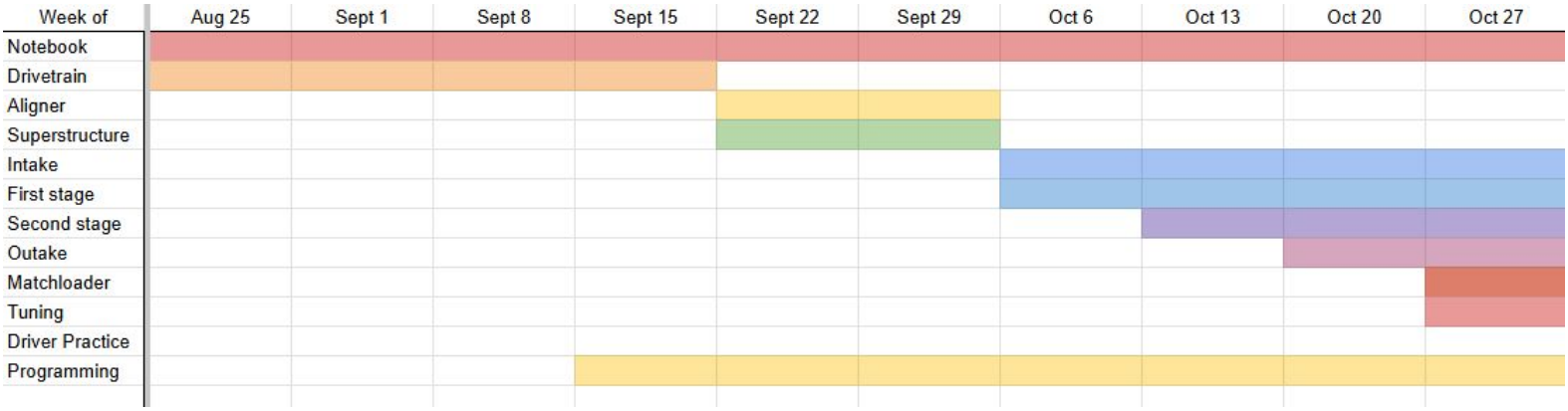
Status: Early-Bird opens on 3-Dec-2025
Spots Open: -
Date: 31-Jan-2026
Event Region: Maryland

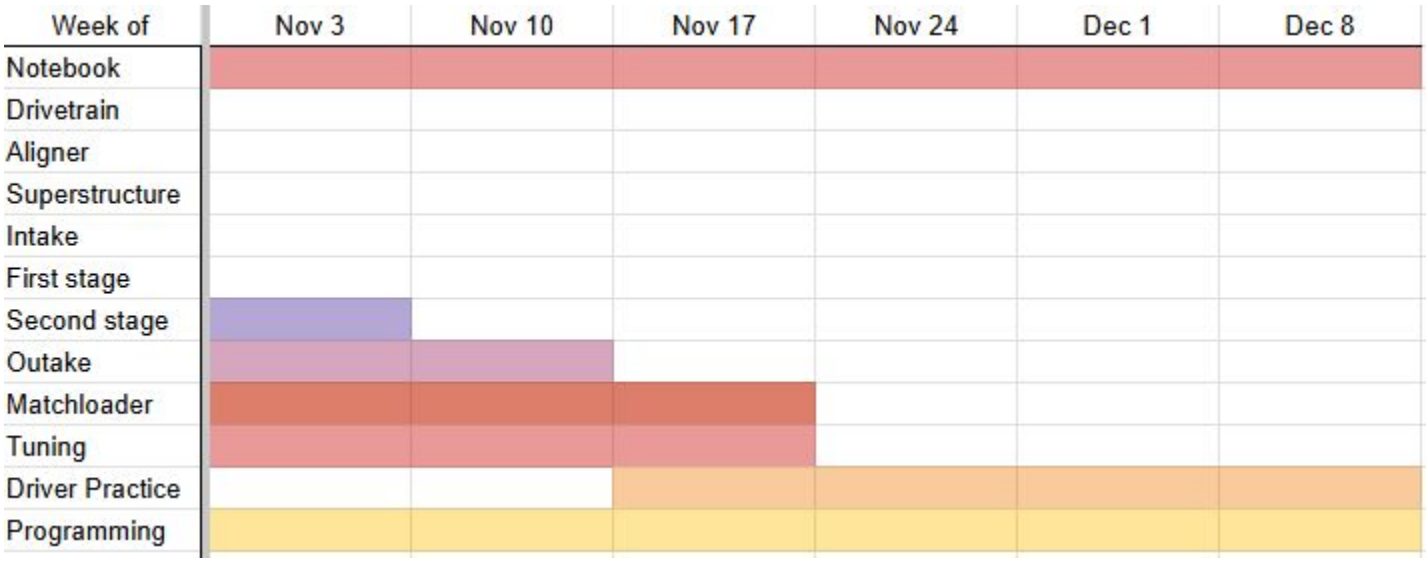
Event Code: RE-V5RC-25-1922
Type: Open Tournament
Location: 1101 South Potomac Street, Hagerstown, Maryland, 21740, United States

We are looking for a Feb competition, we worry that we will not qualify for States by January.

Time Management

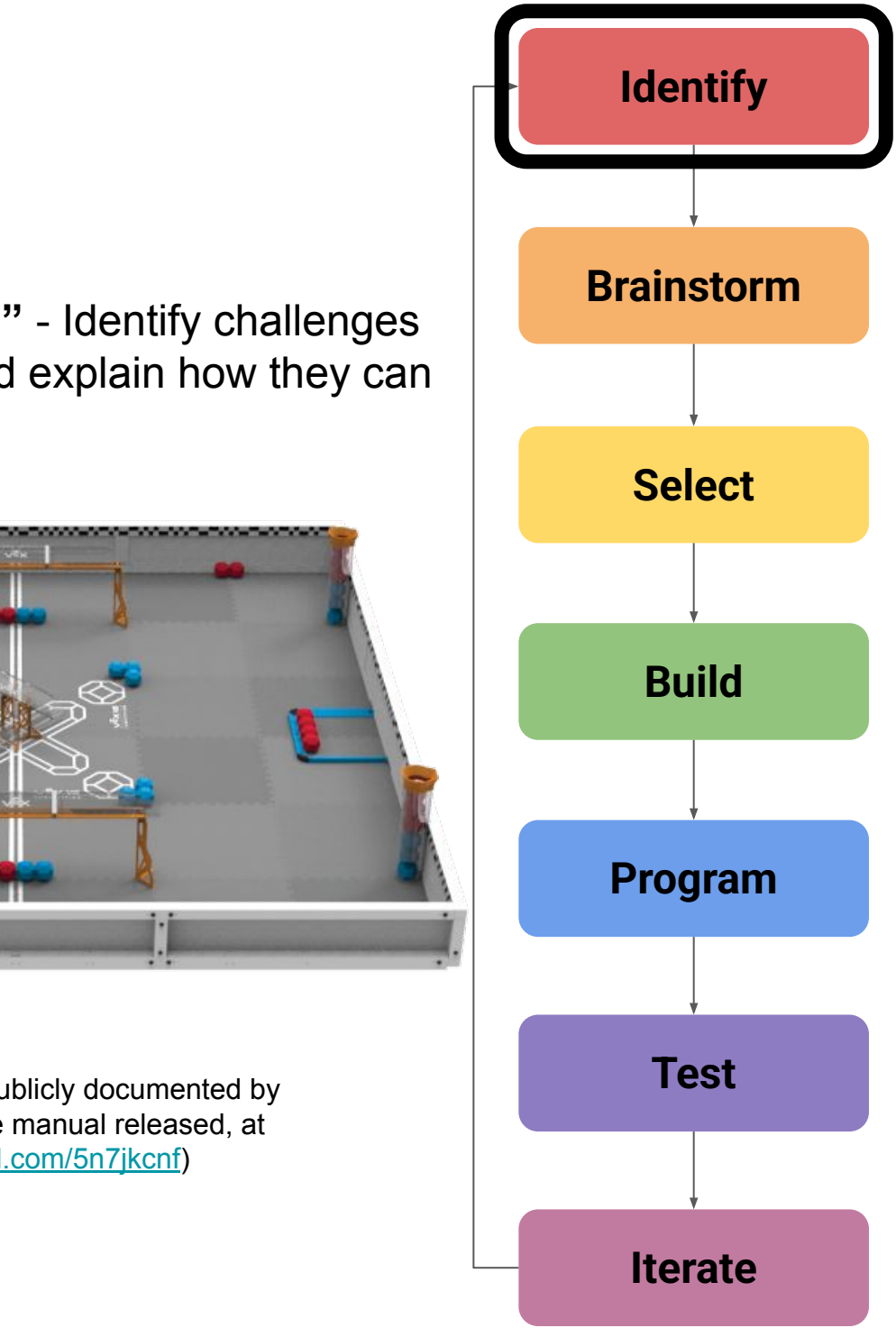
We are just 4 months away from our first competition!
We only get about 3 hours of build time a week (more like 2 hours of actual usable time), so planning outside of our meetings is a must.
Below is a Gantt Chart of how we plan to manage our time and how we're going to divy up tasks needed within the first months





Rowan and Oliver plan to tackle building, with Katie assisting
Dennis plans to program
Rana plans to practice driving the bot at any chance possible

Identify the Problem
The EDP

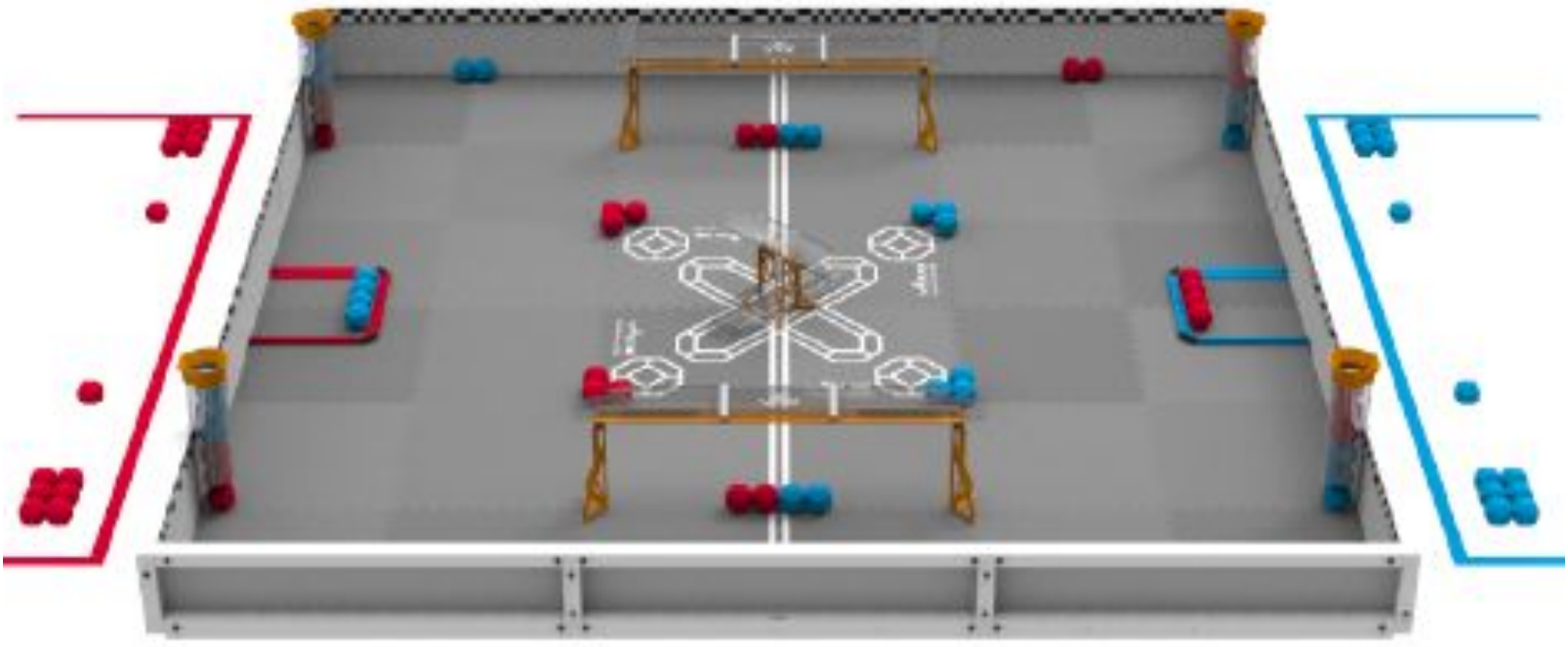


“Identify the Problem” - Identify challenges of the game/design and explain how they can be solved

(This section was publicly documented by Rowan the day the manual released, at https://tinyurl.com/5n7jkcnf)

Field Setup

- Push Back is played on a 12’ x 12’ square field, on top of 18 24” x 24” grey foam tiles. The field is surrounded by a field perimeter, which is either about 11.5” tall.
- Alliance are placed on opposing sides of the field, one side Red, one side Blue. Alliances are composed of 2 teams.
- Attached to each alliance side wall are 2 “Loaders,” which teams can load and remove blocks from
- In the middle of each alliance side wall are “parking zones”
- 2 “Long Goals” are located on either side of the field.
- 1 “High” and 1 “Low” goal are located in the center of the field
- 88 “Blocks” are in play
  - 44 Red, 44 Blue
  - 2 Preloads, each
  - 12 Match Loads, each
  - 18 on the field, each
  - 12 in Loaders, each
- An “autonomous line” spans the width of the field, in the center.



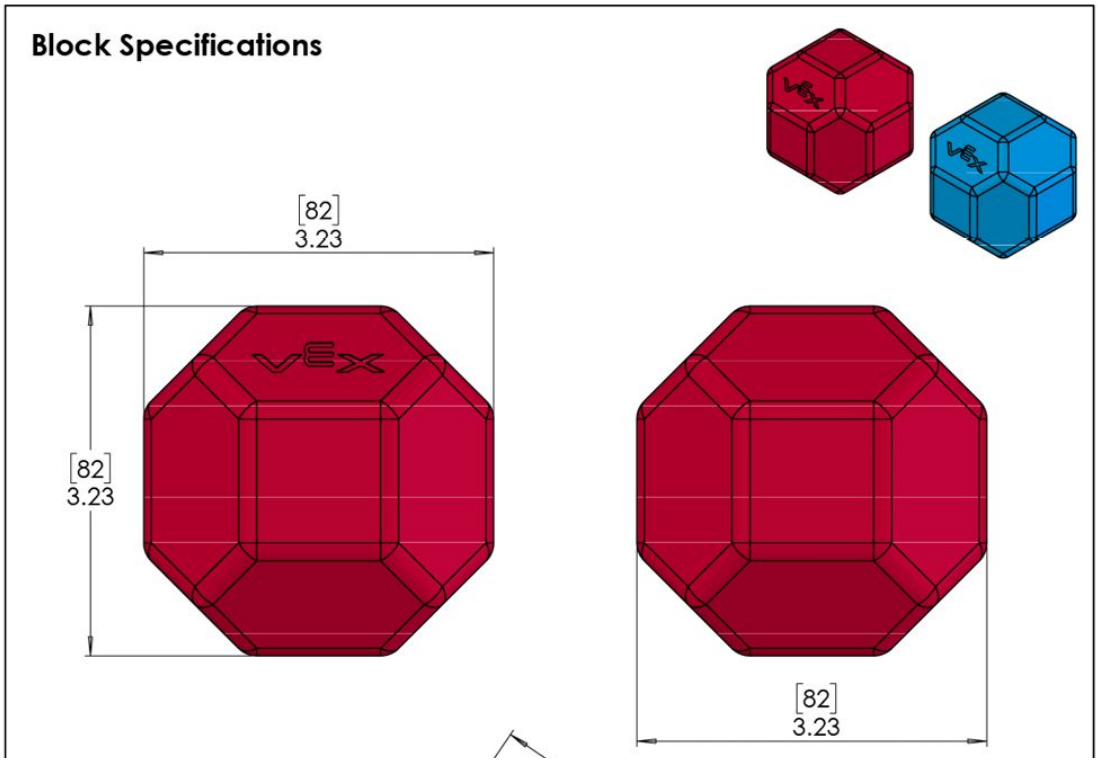
Information drawn from the 2025-2026 “Push Back” game manual, published by the VEX Robotics Game Design Committee, content used with permission.

Blocks - Game Elements

A block is a 18 sided hollow plastic object with flat faces and is the primary scoring element in Push Back.
Also known as octadecahedron, “octoballs,” “cubes,” “doohickeys,” etc. They will be referred to as “blocks” for the purpose of this notebook.

- Blocks come in two colors, Red and Blue, similar to alliances.
- Each cross-section measures approximately 3.25” (82mm) between pairs of opposing flat faces, as seen below.
- Blocks weigh approximately 40 grams, each
- Blocks are scored if fully within goal, not touching the floor, and not touching a bot of same alliance

- Worth 3 points each when scored.
- Scored blocks can be counted towards “Control Bonuses,” where they are worth an additional 6, 8, or 10 points, depending on location.



Information drawn from the 2025-2026 “Push Back” game manual, published by the VEX Robotics Game Design Committee, content used with permission.

Goals - Field Elements

These field elements are the objects that blocks are scored in.
“Goals” include the low, high, and long goals.
Each goal is comprised of clear, plastic tubes and a metal base.
Blocks can slide freely within goals.

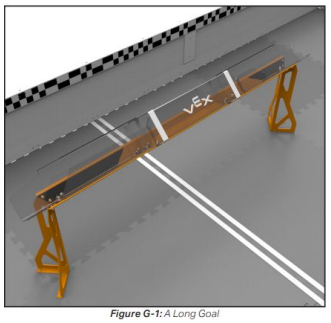


Figure G-1: A Long Goal

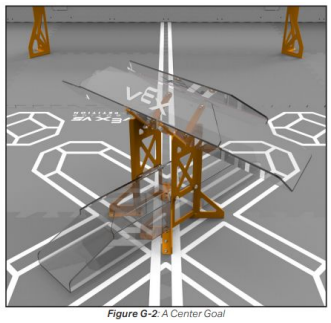


Figure G-2: A Center Goal

Long Goals

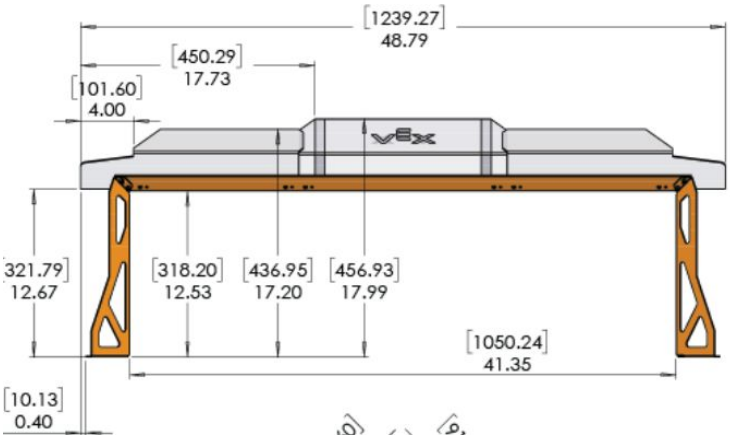
- Located on either side of the field, ~12 inches off the ground
- Can fit 15+ blocks across
- 3 blocks can fit within the central “Control Zone”, the team with the most blocks in that zone earns an additional 10 points.

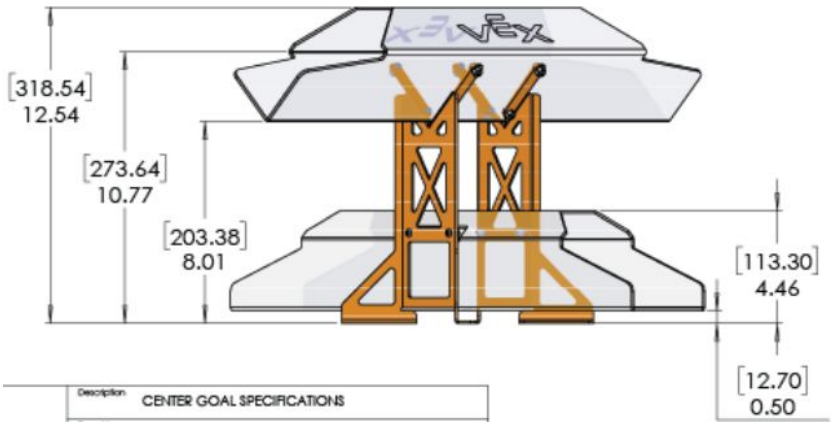
High Goal

- Located at the center of the field, ~8 in off the ground

Low Goal

- Located at the center of the field, ~0.5 in off the ground





Information drawn from the 2025-2026 “Push Back” game manual, published by the VEX Robotics Game Design Committee, content used with permission.

Park Zones - Field Elements

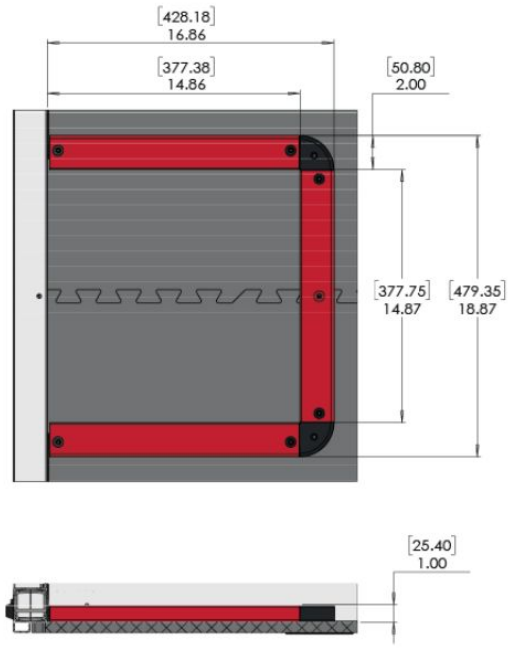
Park Zones are 18.87“ x 16.86” plastic low walls located on either side of the field. The lip is about 1 inch, and wall are slanted inwards
When starting a match, all robots must be touching their alliance park zone
When a match is ending, teams can gain bonus points for parking within their parking zone:

- 8 points for 1 parked robot
- 30 points for 2 parked robots

Robots are considered parked if:

- Robots are not touching the floor outside the park zone
- Robot is not touching any field elements other than the inside face of the field perimeter, the floor within the frame, and the park zone walls
  - Contact is not required
  - Can be resting on a block
- Robot is breaking the vertical plane of the park zone

Park zones are protected during the last 20 seconds of a match



Information drawn from the 2025-2026 “Push Back” game manual, published by the VEX Robotics Game Design Committee, content used with permission.

Important Terms

- **Holding** - A robot status. A robot is considered to be holding if it meets any of the following criteria during a match:
  - **Trapping** - Limiting the movement of an opponent robot to a small or confined area of the field, approximately the size of one foam field tile or less, without an avenue for escape.
  - **Pinning** - Preventing the movement of an opponent robot through contact with the field perimeter, a field or game element, or another robot.
  - **Lifting** - Controlling an opponent's movements by raising or tilting the opponent's robot off of the foam tiles.

Violation if a team is holding another bot for more than 5 sec

- **Disablement** - A penalty that forces the team affected to set down their controller for the remainder of the match
- **Disqualification** - A penalty that removes a team of all win points received in the match
- **Drive Team member** - A student who stands in the alliance station during a match
- **Field Element** – The foam field tiles, field perimeter, white tape, elevation bars, match load bars, goals, and all supporting structures or accessories
- **Autonomous Bonus** – A 10 point bonus awarded to the alliance that has earned the most points at the end of the autonomous period.
- **Match** - A set time period in which the robot performs tasks.
  - Autonomous Period - First 15 seconds, robots run a pre programmed route to earn points
  - Driver Control Period - Robot is controlled by a team member to score points

Manual Update - 1.0

The game updates periodically throughout the year to respond to developments, loopholes, and issues that arise with gameplay, and to adjust the difficulty of the game.

Since 0.1, a few important changes were made:

- Lifting's definition was updated; "Preventing a Robot that is already off of the Floor from returning to the Floor may also be considered Lifting or Trapping."
- The control zone's definition was updated to "the space between (but not including) the white tape lines, … and holds up to three (3) Blocks."
- <SC2> was updated to clarify blocks must be touching the walls of a goal to count as scored
- <SG2> and <SG3> were updated to clarify expansion limits "The Robot can never be larger than 22" wide or 22" long"
- Blocks that bounce out of a Loader while being Loaded into the Field should not be considered a Violation
- <SG10> updated, robots can not directly contact blocks within the enclosed sections of long goals
- VEX IQ pins can not be used at all
- <RSC2b> updated, 7 Blocks constitute a filled Control Zone in a Center Goal
- Blocks touching a robot at the end of a skills match are not scored
- Blocks that leave a skills match are not returned

For now, the game manual is ready for use

The next major update, 2.0, will release September 4th, 2025

Manual Update - 2.2

Game update time!
Updates to 2.2 from 1.0, which released Dec 4
A few major changes have been made to the game:

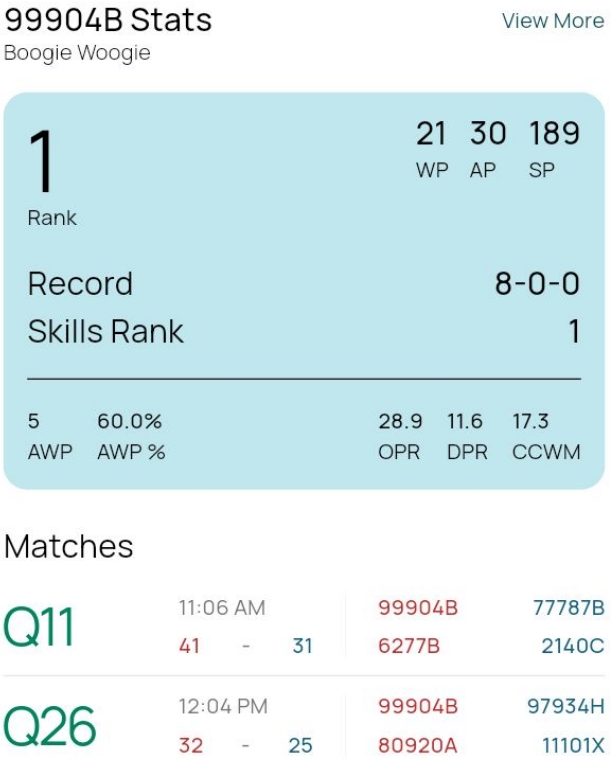
- Removing more than 3 blocks from the field will be penalized
- Match loaders can use 2 hands when loading blocks
- Blocks falling out of loaders are not considered violations
- Blocks can only be added to loaders containing 5 or fewer blocks
- Worlds AWP was released
    - 10 blocks scored across 3 goals, 3 alliance blocks removed from loaders, both robots leave park zone
- 3 drive team members per match
- **Holding reduced to a 3 count**
- Clarification of when blocks can be introduced into loaders
- Protection to robots in park zone in final 20 sec
- V5 brain needs to be clearly visible
- Blocks removed from skills field are not violations
- Robot must move to count for skills match
- Teams that participate in no qualification matches cannot be considered for judged awards
- Anchoring has a new violation
- Teams that force another team to violate a rule will be penalized
- Hold counting only starts after ref notices a hold
- **Match loads introduced 1 at a time**
- Robot skills control zone updates
- Non drive team members can’t affect matches
- More offensive robot gets benefit of the doubt
- Goalkeeping is allowed
- Event staff can photograph bots
- No speakers on bots
- No modifying v5 brain accessories
- 3-4 blocks are counted as in the long goal control zone, not just 3

Scouting Stats / Terminology

- **WP** - Win Point
  - Affects your overall ranking during qualifying matches
  - 1 WP for completing the Auton WP
  - 2 WPs for winning a match
  - Higher is better
- **AP** - Autonomous Points
  - Total points scored during the Autonomous period
  - In case of a WP tie, affects your overall ranking
  - Higher is better
- **SP** - Schedule points
  - When you beat a team, their score is added to your SP
  - In case of a WP and AP tie, SP is the tie breaker
  - Higher is better

Image credit Rowan Wood - Elapse App

- **AWP** - Auton Win Points
  - Total AWP you've scored
- **OPR** - Offensive Power Rating
  - How much you contribute to your alliance's score
  - Higher is better
- **DPR** - Defensive Power Rating
  - How much you stops your opponents from scoring
  - Lower is better
- **CCWM** - Calculated Contributed to Winning Margin
  - How much you contribute to the total winning margin
  - Higher is better



Design Brief

Problem Statement:

- Push Back is the 2025-26 VEX V5RC Competition, where teams attempt to score their alliance color blocks in goals. 88 blocks are in play, with 36 on the field and 21 in loaders. 4 start as preloads, and 24 start as match loads. Four loaders are located around the field, and blocks can be fed through them. 4 goals are located around the field, at 3 different heights. 2 are long goals, spanning a third of the field. Robots then attempt to park in a small area to score extra points at the end of a match.

Design Statement:

- We will design, build, program, test, and compete with a robot that can quickly and efficiently score game components in Push Back

Constraints:

- Power consumption is limited to 88 watts
- Robot must be build with VEX permitted parts
- Robot must start in 18” x 18” x 18”
- Robot can expand up to 22” x 22” x 22”
- No possession limit
- Score blocks in 3 levels of goals
- 12 custom plastic pieces max, must fit within 4” x 8” sheet
- **Time, Money, Resources**

Max motors, based on 88w limit

Example	A	B	C	D	E
Qty of 11W Motors:	8	7	6	5	0
Qty of 5.5W Motors:	0	2	4	6	16

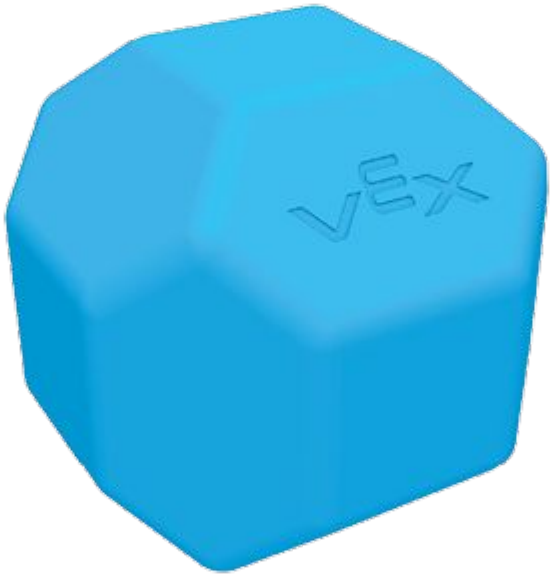
Game Strategies

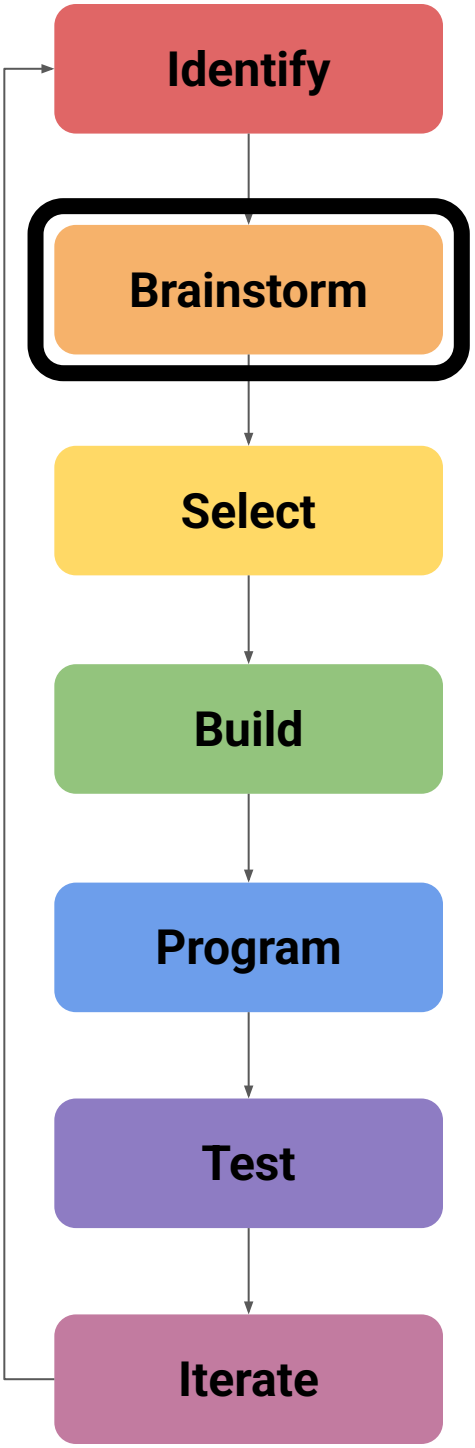
Following what we saw over the summer at events such as Mall of America, we came up with the following strategies to base our bot off of.

- Fast scoring/descoring is a must, it's hard to starve the field
  - Fast bot, fast scoring
- Match loading needs to be fast, the floor gets cleaned quickly
- Descoring is an issue, need to be able to prevent
- Double parking isn't done much, need to protect goals/ descore if opponents attempt to park
- Storing blocks in the robot seems to be a must to allow for fully filling a goal
- Control bonuses are important
- The Autonomous Bonus is a massive help in matches, attempting to score the most points during the Auton Period is a must
  - Many teams at the Mall of America Signature Event only won due to the points they won during the Auton Period
  - Outscore the opponents
- Network with other teams and talk with them! Don't hide in the pits, have conversations! Try to get into the eliminations!

Brainstorm Solutions
The EDP

“Brainstorm, Diagram, or Prototype Solutions” - Create solutions to the problem, then prototype the best solutions





Drivetrain Brainstorming

The drivetrain is the most important component of our robot, without one it is very hard to do anything while competing. Our goal is to pick the most effective drivetrain style to have a competitive robot.

A few things to keep in mind

- In years past, the most competitive teams had fast, 6 motor (66 watt) drivetrains.
- Last year, we used a 66w drivetrain, at 333rpm
- Robot traveled at 56.7 in/sec, which wasn't bad, but we still got circles ran around us
- Drivetrain should be simple to maintain, so we can quickly fix any issues that arise between matches
- Hot swappable motors would be nice, but not necessary

Hot Swappable Motors

According to rule <R28a.ii>, “may make the following modifications to the V5 Smart Motor (11W)’s … Removing or replacing the screws from the V5 Smart Motor Cap”

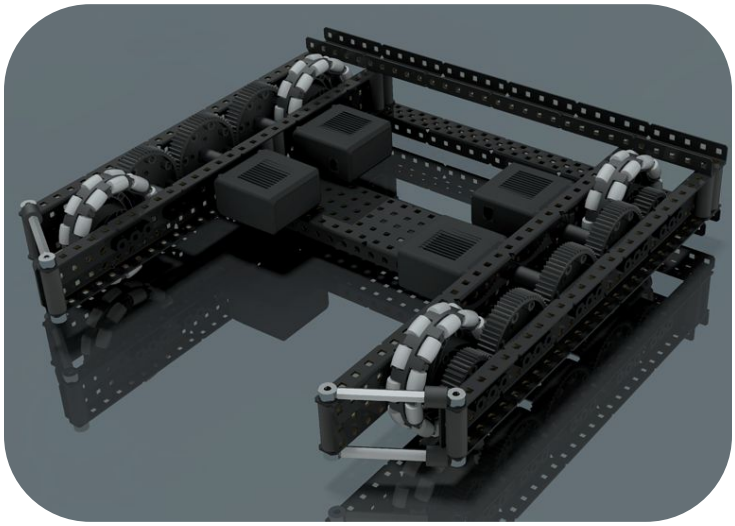
Essentially, a hot swappable motor does not have any of the linking the v5 motor cap to the motor itself, instead using zip ties and rubber bands to hold the motor in place. This allows for rapid swapping of motors between matches, so you always have cool motors going into matches; no need to worry about thermal throttling.

Image credit Ascend Robotics

Drivetrains

This video by “Engineering Girl” does a great job comparing common drivetrains, much of this section utilizes their data.
https://youtu.be/Py14YTHCth0

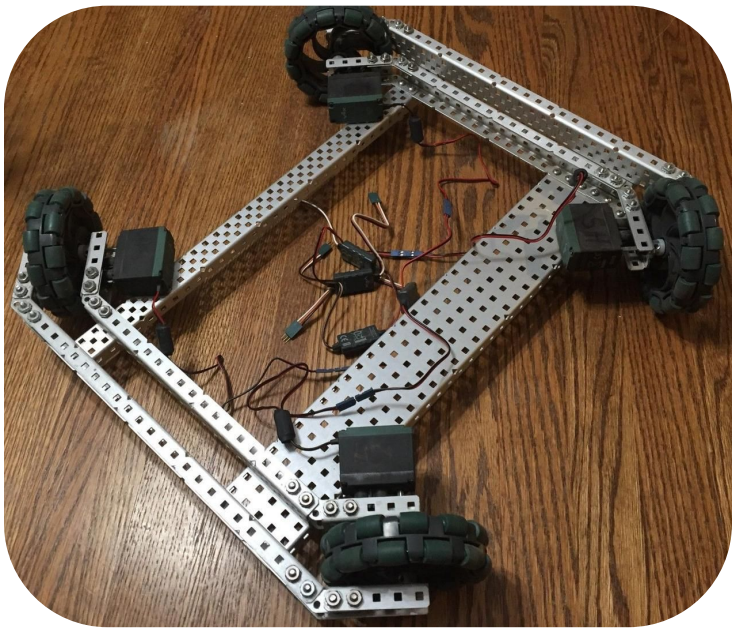
Image credit Xenon27 - VEX Forum



Tank Drive All wheels are parallel to the robot, most common drivetrain design	
Pros:	Cons:
- Fast - Consistent - Easy to build/code - Easy to maintain - Hard to shove around	- Less maneuverable - Can not strafe

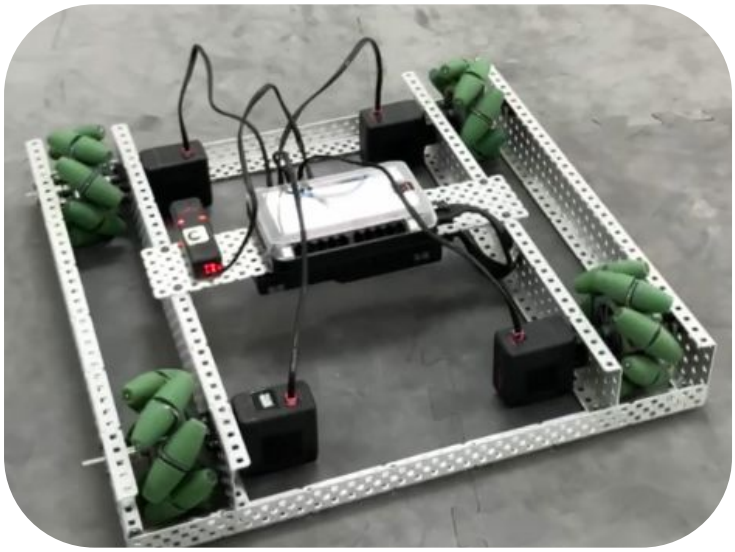
Honominic / X-Drive Wheels set at a 45 degree to the robot	
Pros:	Cons:
- Very maneuverable - Somewhat fast	- Difficult autonomous programming - Not very powerful - Hard to maintain - Easy to shove about - Not very precise in movement

Image credit Worth Point Robotics - VEX Forum



Drivetrains

Image credit SuhJae - VEX Forum



Mecanum Drive Uses mecanum wheels to allow for omnidirectional movement	
Pros:	Cons:
- Movement in all directions - Easy to build/code - Hard to shove - Much simpler than a Holonomic drive	- Low traction - Low power - Low speed

H-Drive Tank drive with sideways center wheel for sideways movement	
Pros:	Cons:
- Allows for sideways movement - Benefits of tank drive	- Uses an extra motor - Center wheel takes up valuable space - Easy to shove

Image credit Doug Klein

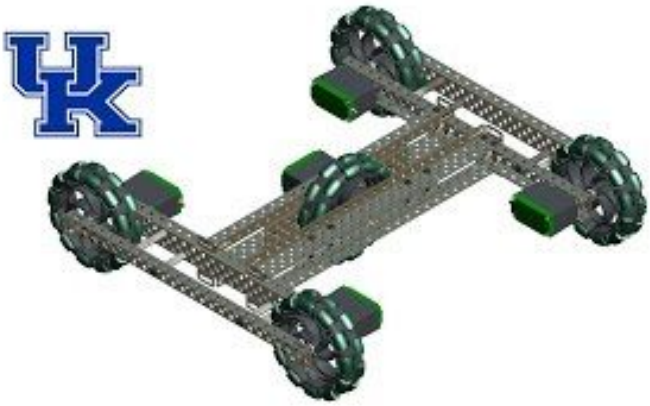
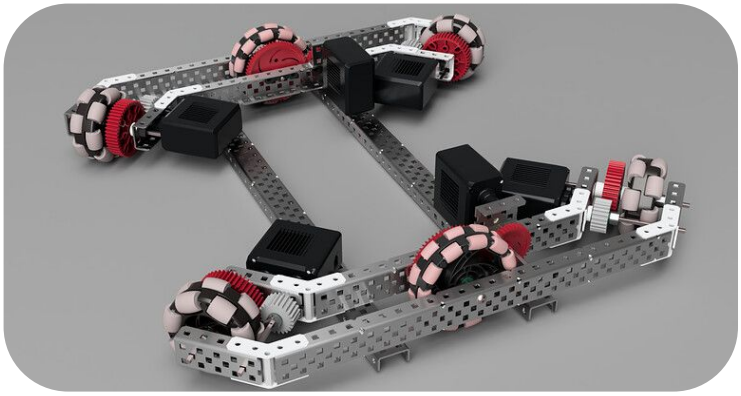


Image credit Benn - VEX Forum



Asterisk Drive X Drive with more powah	
Pros:	Cons:
- More power than X Drive - Very versatile - Very roomy	- Easy to shove - Annoying to program - Not precise

Motor Cartridges

11 watt v5 Smart Motors utilize motor cartridges to control their output speed, which outputs different amounts of torque and speed. This can then be linked to external gear ratios to control the speed of your mechanisms.

5.5w motors are limited to an output of 200rpm, or a green insert.

	RPM	Torque under 60% velocity	Torque at 100% velocity
Red cartridge	100	~2.1 Nm	~0.7 Nm
Green cartridge	200	~1.1 Nm	~0.3 Nm
Blue cartridge	600	~0.4 Nm	~0.1 Nm

Image and Data Source: VEX Library
kb.vex.com/hc/en-us/articles/360044325872-Understanding-V5-Smart-Motor-11W-Performance

Essentially:

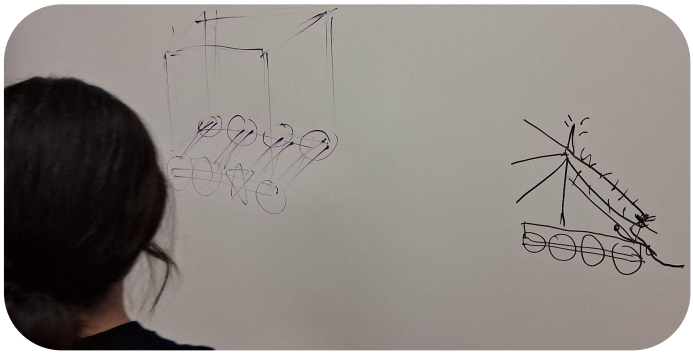
- Red: low speed, high torque
- Green: medium speed, medium torque
- Blue: high speed, low torque

This data is important to reference throughout the season so we can choose the best cartridge for each mechanism we're planning to use.

Ex: an intake needs to spin fast and is not under much load, a blue insert may be the best choice
Ex: an arm that supports the weight of the entire robot needs a lot of torque, but may not need speed : a red insert will work the best

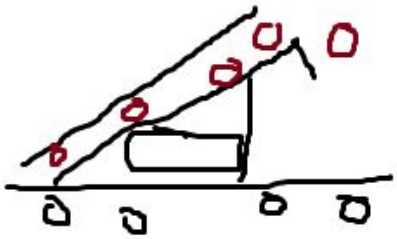
Block Pathing

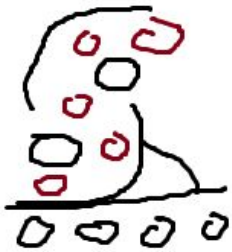
Unlike last year where we only saw a few set paths for moving elements, we have many options! We're considering a few specific options, based on theoretical paths blocks can take through our bot.



Hopper / "Shopping Cart" Allows for full control over all blocks	
- No possession limit - Starve the opponents - Overall funny bot	- Bulky - Slow scoring? - How important is block control?

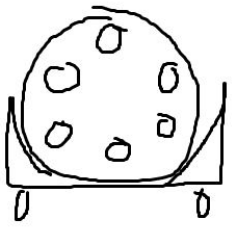
Linear Straight shot for blocks	
- Easy to build - Simple design - Fast scoring?	- Low block storage





S Linear but *fancy*	
- Store more blocks than Linear - Fast scoring?	- More complex to build - More moving parts

Revolver Blocks go spin	
- Really funny - Hold a lot of blocks - Score fast	- Very complex - How to intake/outtake?

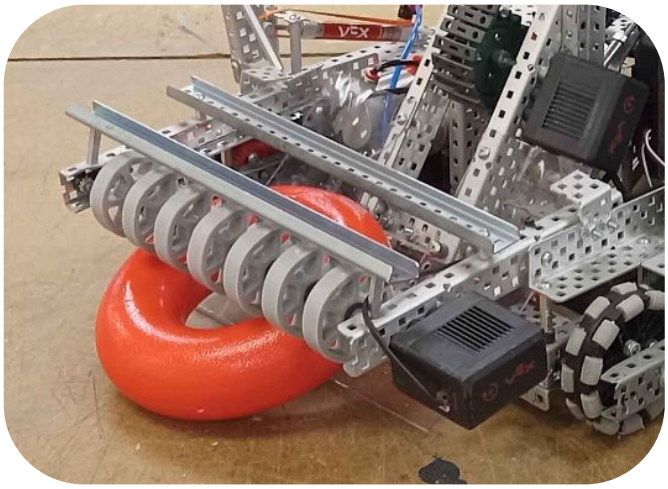


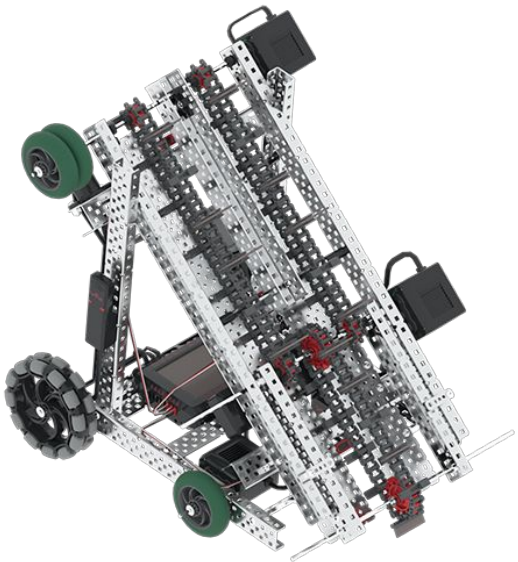
Intakes

In order to be successful at all, we need to be able to pick up blocks off the field and side loaders.

Side Intakes Image provided by VEX - Competition Super Kit	
- Can pull blocks out of wall loaders	- Need to hit blocks head on - Uses 2 motors

Over roller Image from 593C High Stakes bot	
- Wide entrance to pick up blocks - 1 motor - Easy to build	- Can not pull blocks from loaders



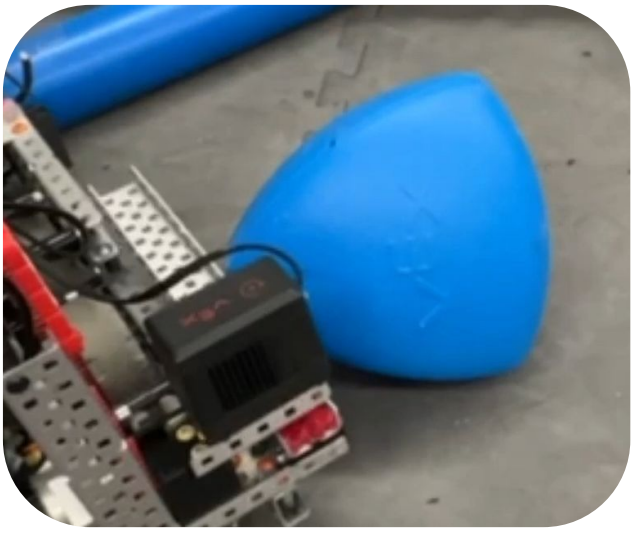


Integrated intake Image provided by VEX - Disco Hero Bot 2022-23’	
- No need for additional intake	- Can not pull blocks from loaders

Match loaders

If we choose a over block style loader , we need to pull blocks out of wall loaders

Side rollers can easily pull blocks out, but here are some ideas



Sweeper arm 9MotorGang on YouTube, used with permission	
- Easily sweep blocks out	- Only pull 1 block at a time - Uses an extra motor

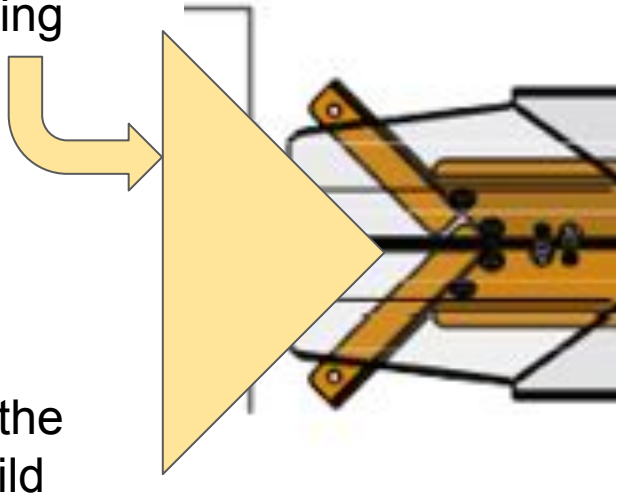
Plastic doohickey This thingamabobber	
- Lift over lip - Drops all blocks at once	- Finicky? - Additional metal/ expansion

Just a shaft™ Maybe it'll work?	
- Really easy to build - If done right drops all blocks at once	- Finicky? - One block at a time?

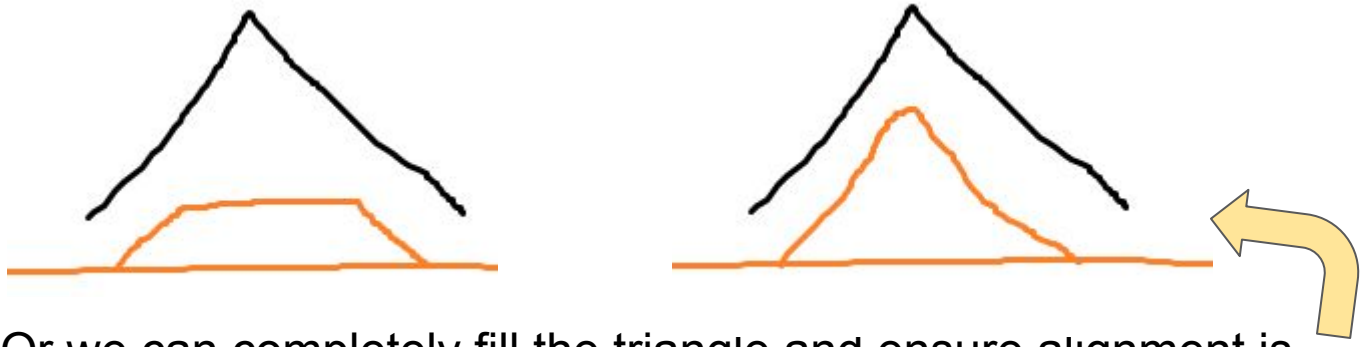
Aligner

Aligning with the goals might be a bit hard, so this should allow for better alignment

Long goals have this triangle thing built into the bottom of the leg
If we can push the bot into it we can easily score without worry about spewing blocks everywhere



If we don't want to extend past the 18" x 18" sizing box, we can build something similar to this:



Or we can completely fill the triangle and ensure alignment is always happening. With the partial triangle, if we hit the triangle at an angle, it may not perfectly align.

Currently not sure how to align with the center goal, will revisit at a later date.

Programming software

The VEX v5 system offers many programming libraries and CLIs (command line interface) to aid in programming our robot. VEX offers VEXcode v5 (https://codev5.vex.com/) as a starting point, which offers compatibility with block coding, similar to scratch, along with the C++ and Python programming languages. In addition, they offer the VEXcode VSCode extension, which allows for programming the robot with C++. Finally, students at Purdue University's Purdue ACM SIGBots created PROS (PROS Robotics Operating System or PROS), an entirely open source way to compile C++ software to a v5 brain.

VEXcode	
- Very intuitive - Plenty of resources - Block, C++, Python - Works on any computer - Supported directly by VEX	- Limited use of libraries - Proprietary

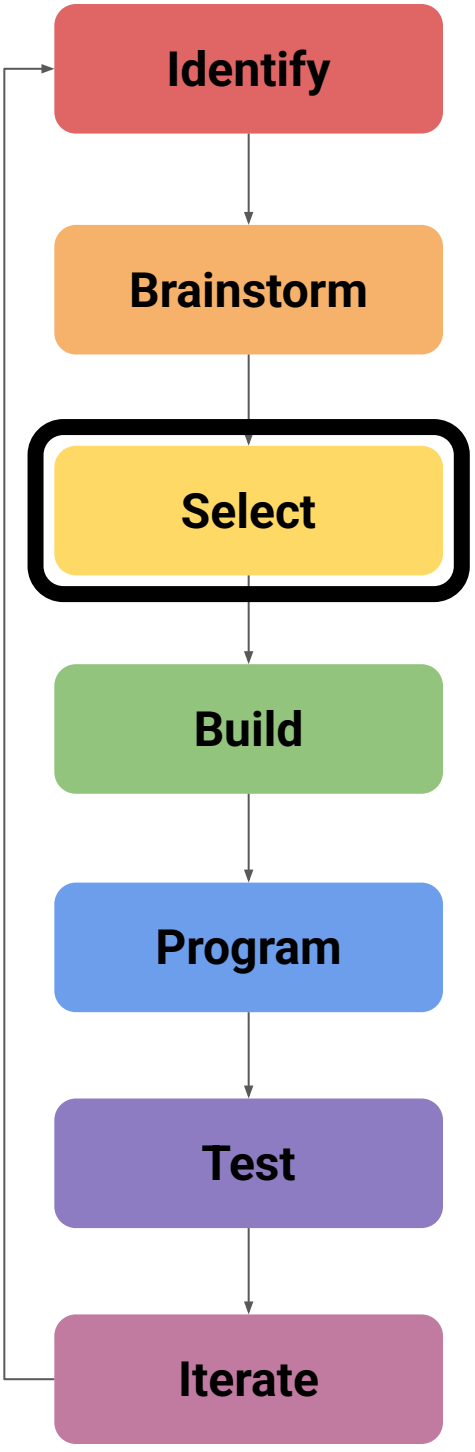
Image credit VEX Robotics

PROS	
- Plenty of community built, external libraries, including LemLib - Plenty of community resources - Full access to all of C++'s features	- Only supports C++ - Requires a personal device to run

Image credit Purdue ACM SIGBots

Select Solution
The EDP

“Select Best Solution and Plan” Select a solution, explain why and continue to plan from there



Selecting our Drivetrain

We put drivetrain selection up to a vote, where each member of the team was to rank each aspect of each drivetrain on a scale of 1-5. Total points calculated at the end, highest score wins.
Only 3 members were present when votes were tallied.
Choices were made based on research in Brainstorming phase
Votes are biased, based on past experience.
Highest possible score is 75 points.

Style	Speed	Manuvablity	Pushing Force	Ease of building	Footprint	Totals
Tank	5, 5, 5	3, 4, 4	5, 5, 5	5, 5, 5	5, 5, 5	71
Honominic	4, 4, 4	5, 5, 5	4, 3, 4	3, 4, 4	4, 4, 4	61
H-Drive	4, 4, 5	5, 4, 4	4, 4, 5	4, 3, 3	3, 3, 3	58
Mecanum	4, 4, 3	4, 4, 5	3, 3, 4	4, 4, 5	4, 4, 3	58
Asterisk	4, 4, 5	5, 5, 5	4, 5, 4	3, 4, 4	4, 4, 4	64

Our overall decision is to build a tank drive due to ease of building, speed, pushing force, and its footprint. Even though it may not be as maneuverable due to being unable to strafe, we believe it will be the most viable to build and integrate with the rest of our design.
We were surprised at how similar our thoughts were, but after analysing our biases, it quickly became clear why - one style which we see everyday jumped out at us, and was simple enough to easily build and understand how it works.
Rowan was also really excited to be able to build a drifting robot again this year :)

Drivetrain - Finer Details

Now that we've narrowed down our design, there are several concepts we need to narrow down, being motors in the drivetrain (4 motors or 6 motors), and size of wheels (2.75, 3.25, or 4in).

Drivetrain Motors: 4 motors vs 6 motors

4 motor drive	
• Uses less space • Uses 2 fewer motors	• Less powerful • Slower acceleration

6 motor drive	
• More powerful • Fast acceleration	• Uses more space • Uses more motors

Wheels: 2.75 in vs 3.25 in vs 4 in

2.75in wheels	
• Uses less space • Low center of gravity • Can fit 8+ wheels (more traxion)	• Don't own any • Robot is much lower to ground

3.25in wheels	
• Balanced size • Can fit 8 wheels (more traxion)	• Only own one set of 8 • Longer than 2.75 in wheels

4in wheels	
• Fast • We have many	• Larger • Only fit 6 wheels • Higher center of gravity

Overall, we decided on building a 6 motor drive for a stronger wheelbase when entering shove matches throughout our competitions, which is an issue we highly anticipate. In addition, we chose to use 8 3.25 wheels due to their happy medium; less likely to tip, and small enough to fit 4 wheels on either half for more traxion. We will use 4 omni and 4 traction wheels, to gain the turning benefit from omni wheels on the outskirts of our drivetrain, and traction wheels in the center to prevent being shoved from the side.

Drivetrain - Finer Details

We also have to choose gearing our drivetrain. Last year, we ran 333 rpm 3.25in wheels, which limited us to ~56.7 in/sec. We noticed many other competitive teams had much faster robots which could still run circles around us.

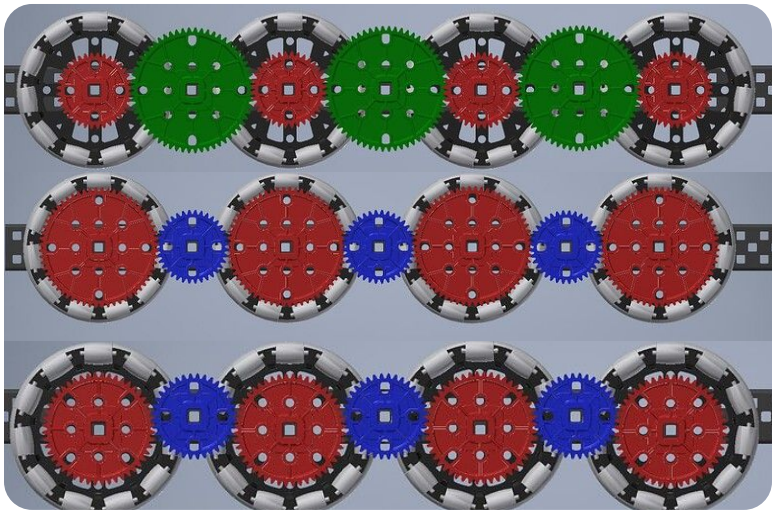
Important Formulas:

$$\frac{\text{\# of Driven Gear Teeth (Output)}}{\text{\# of Driving Gear Teeth (Input)}} = \text{Gear Ratio (Input:Output)}$$

$$\frac{\text{motor rpm} * \text{wheel diameter in} * \pi}{\text{gear ratio} * 60 \text{ sec/min}} = \text{Wheel Speed (in/sec)}$$

Based on previous brainstorming, we chose to use green cartridges for our drivetrain motors, to ensure we have enough torque without sacrificing speed.

We considered 3 drivetrain gearings, a 333 rpm 56.7 in/sec drivetrain, a 360 rpm 61.3 in/sec, and a 450 rpm 76.6 in/sec. While all are much faster than before, option 3 stood out to us the most, as the simplest to build, and meeting all our criteria.



Images credit Xenon27 - Vexforum

We finally decided to screw joint our wheels, to eliminate friction between square axles and round holes, allowing for less friction on the motors, and prolonged battery / motor life.

Selecting our Robot Structure

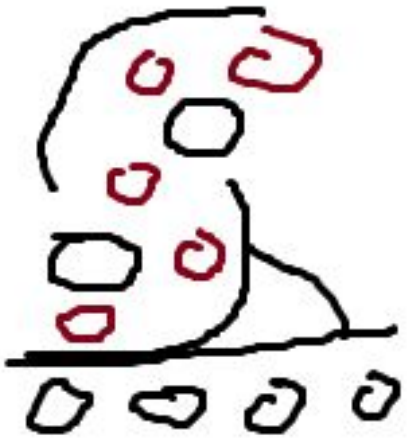
It's time to decide how our robot will be built and how it will move blocks!

Again, using a decision matrix, we're voting on the best idea we think will work

5 categories, speed of scoring, ease of control, block capacity, speed of scoring, and ease of building

Style	Speed	Control	Block capacity	Speed	Ease of building	Totals
Hopper	4, 4, 5	4, 2, 2	5, 5, 5	4, 3, 3	4, 4, 4	58
Linear	4, 4, 4	5, 5, 5	4, 3, 4	5, 4, 4	4, 4, 4	63
S	5, 5, 5	3, 4, 4	5, 4, 4	5, 5, 5	5, 4, 4	67
Revolver	4, 4, 3	4, 4, 5	3, 3, 4	4, 4, 5	4, 4, 3	58

And the votes are in, we're building a S style bot!



More sketches and whiteboard drawings to come

Final Stage

Brainstorming jumpscare! (The EDA is not linear! Jumping around steps is normal!)

How are we going to go from the S system into the goals? Since there are 2 tiers and we need to switch between the two.

Oliver proposed the following idea: A platform that can raise and lower using a pneumatic piston, with a motor at the top to hold back and push blocks into the goal.

Rowan proposed a system where the whole S shifted up and down, but that idea was swiftly thrown out due to the number of issues that could have been if we ran with that idea - pivot points could be weak, hard to build, etc.

We settled on Oliver's idea, to build later in the season.

Selecting our Intake

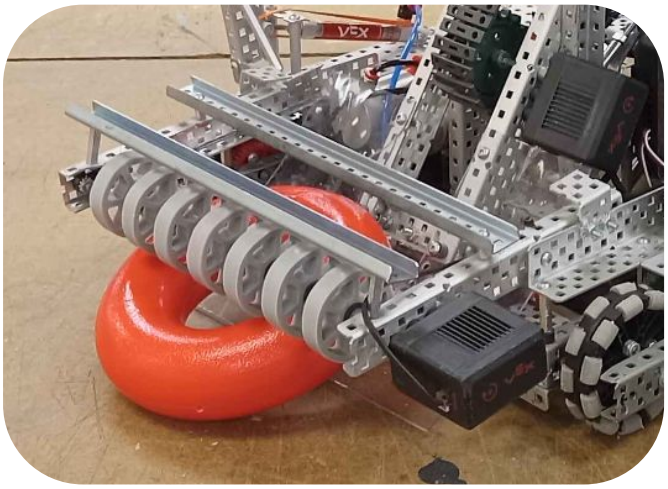
Intakes are important!

This decision matrix is scored on 4 categories, how easy it is to control, how useful the intake is, how motor efficient it is, and how easy it is to build

Style	Control	Usefulness	Motor	Ease of building	Totals
Side	4, 3, 4	5, 3, 4	3, 3, 3	2, 4, 2	40
Over	4, 5, 5	4, 5, 5	5, 5, 5	4, 4, 5	56
Integrated	4, 5, 5	4, 5, 5	5, 5, 5	3, 4, 3	53

Over intake narrowly beats integrated intake due to how easy it is to build, building a separate floating intake is much easier than trying to build around ensuring our S design can pick blocks right up off the floor.

Flexwheels or rubber band rollers can work, but we selected a flexwheel intake as that is stronger to survive head on collisions and slightly grippier against the floor - based on past experience.

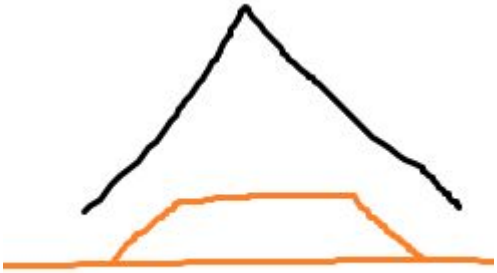
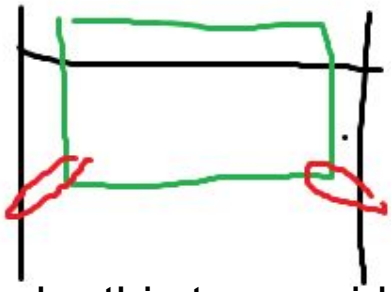
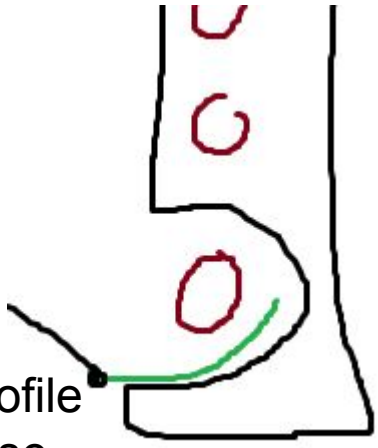


Selecting our Loader Aligner

Since we selected an over intake, we need to be able to pull blocks out of the loader easily.

Of the three options, one stuck out to us the most, no vote needed. By using the plastic intake, the plastic can bend against the back of the loader, and can funnel all blocks into our intake.
We were able to test this without building anything with just a sheet of plastic, and were able to unload an entire loader onto the ground in just a second.

This should also be easy to build, as in theory its just a sheet of plastic and a pivot. Now ensuring it's within sizing and can move out of the way will be the hard part. But that's a future us problem!

ligner to be this trapezoid aligner, as we thought it would be the most low profile while still allowing for self centering and ease of use.
Hopefully it's easy to build with some 45 degree gussets!

Programming

It sounds like we're going to use a hybrid of the two programming systems!

System	Ease of use	Upkeep	Familiarity	External Tools	Totals
VEXcode	5,5	3, 4	5, 5	3, 3	33
PROS	4, 4	5, 5	4, 3	5, 5	36

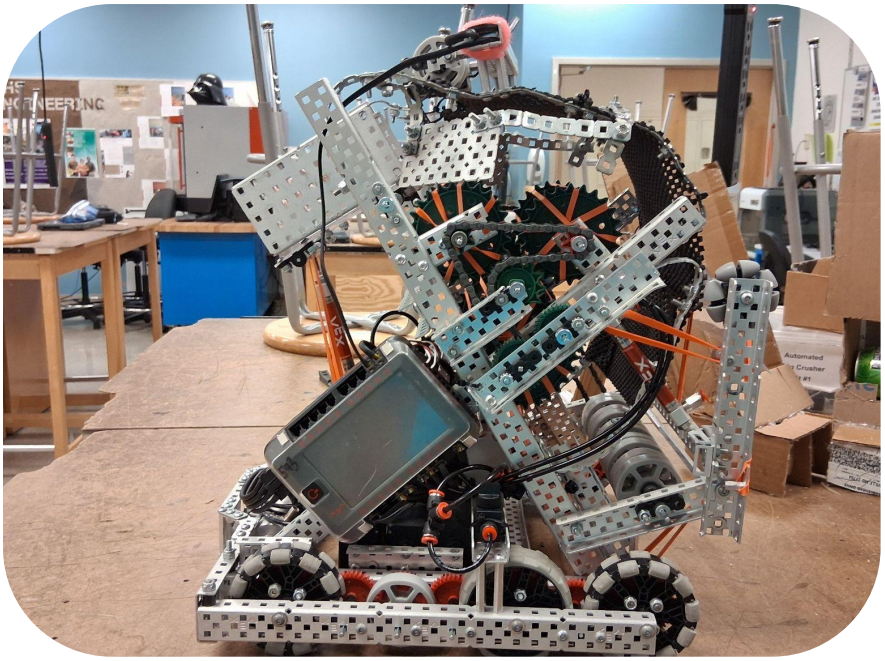
Rowan and Dennis selected our programming system.
While PROS did score higher, we don't know that we'll be able to get it set up and ready to go by our first comp, and also teach other members how the system works

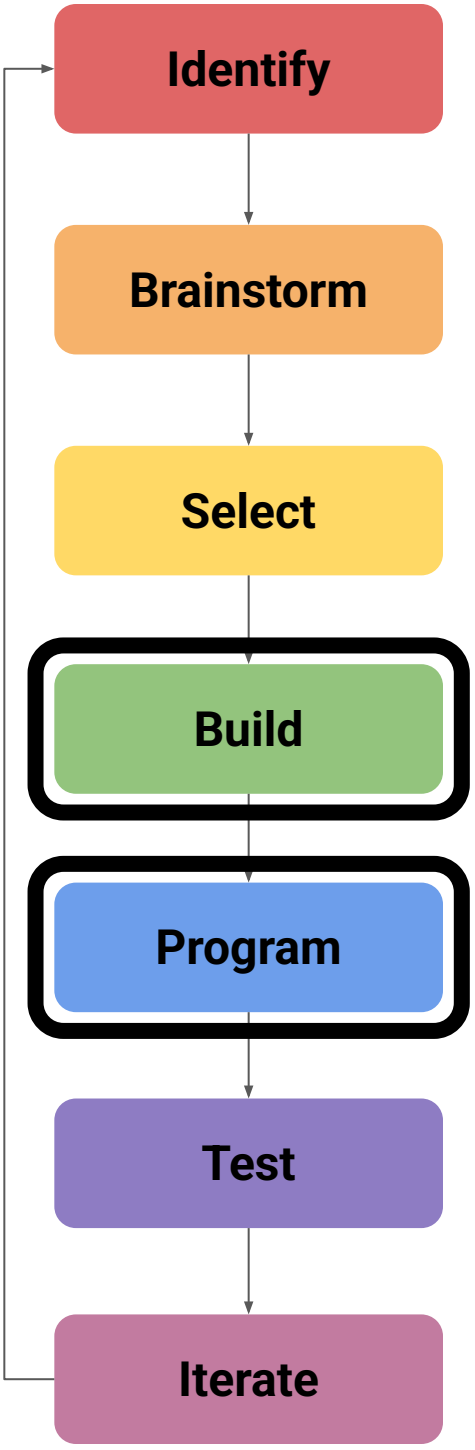
Since most members have at least used Scratch before, we're going to get our robot functioning first, then we'll switch over to make programming autonomous routes much easier.

PROS supports LemLib, which is a very useful system for easily programming auto routes, as it is a much cleaner implementation of odometry and PID than anything we could make, despite us understanding how the system works.

Build and Program The EDP

“Build and Program the Solution” - Create the solution that was chosen, and describe the design and building process. Program the solution that was chosen, document and explain the programming process

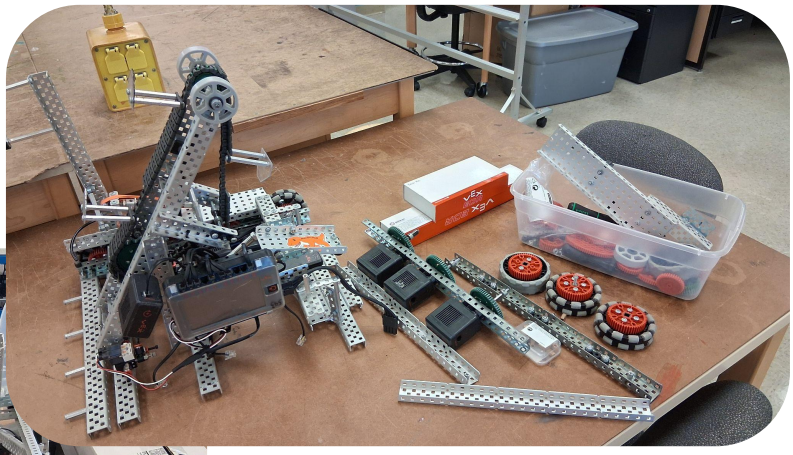


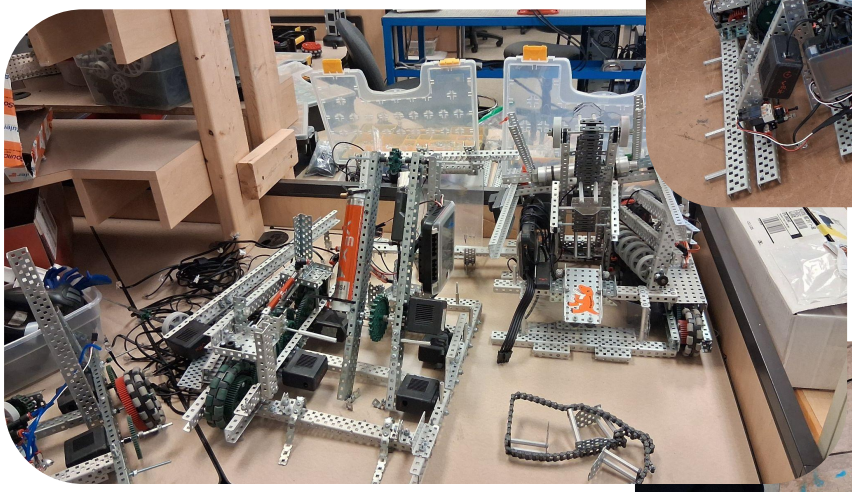


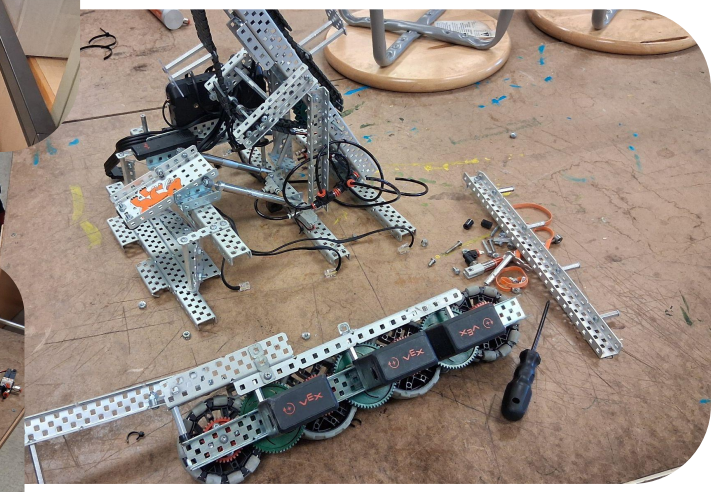
Teardown

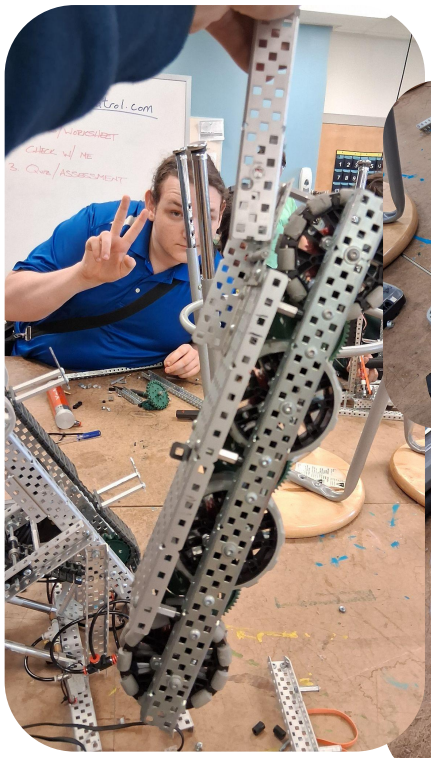
Before we can build anything, we need to tear down our old bots from last year.

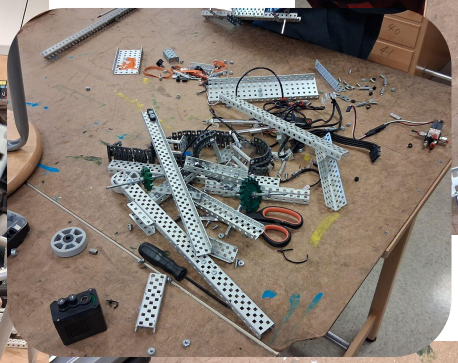
Many of the parts from last year we can reuse, and will do so!

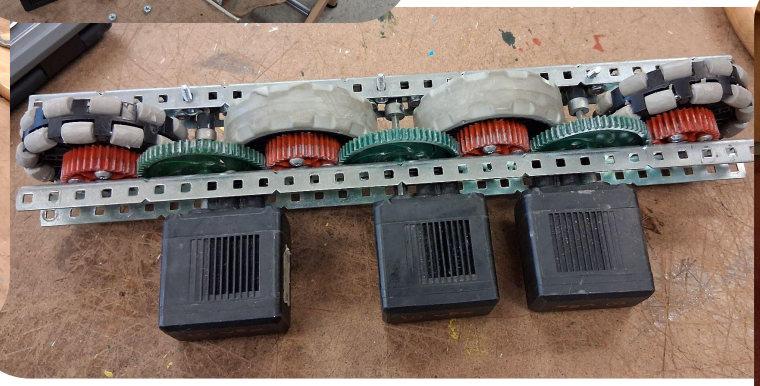






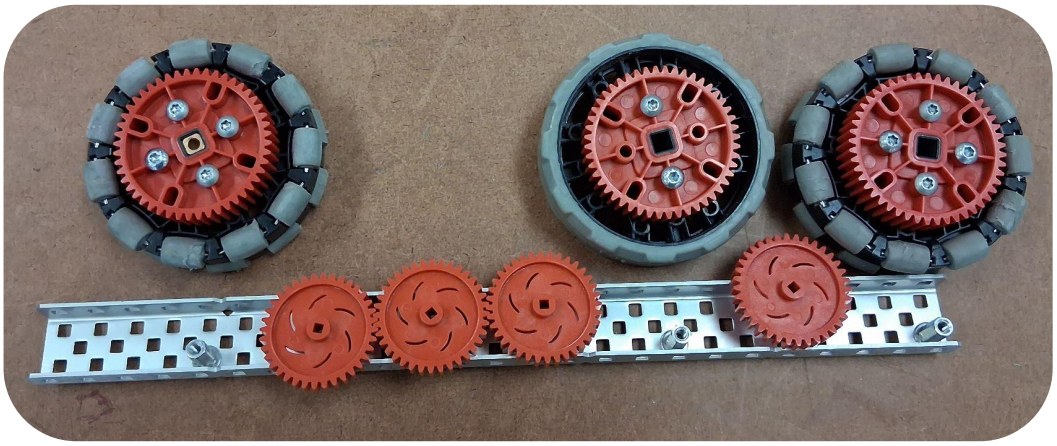




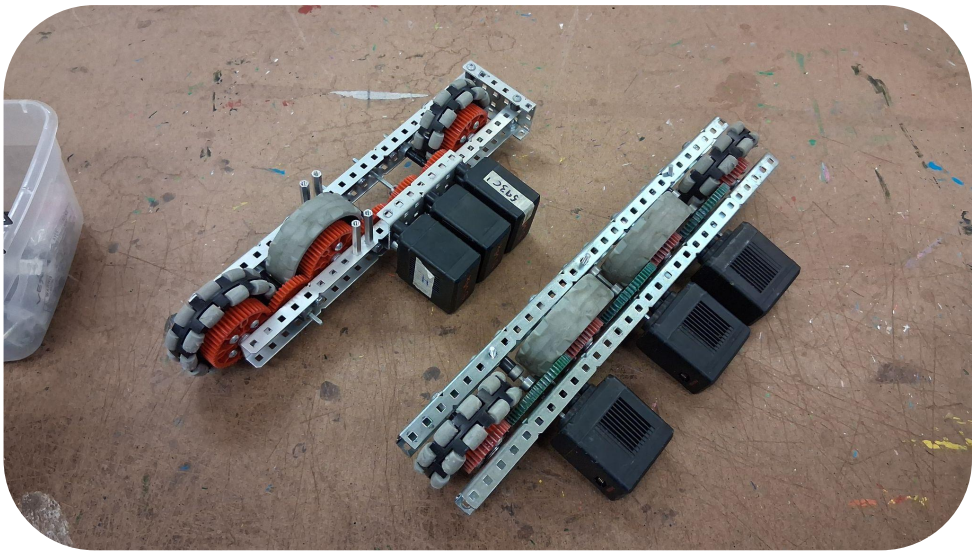


Drivetrain

It's construction time!
We selected a 600 rpm geared down 36:48, which allows us a theoretical max speed of 76.6 in/sec.
We screw jointed our wheels onto the structure, essentially using these brass inserts in place of the regular silver square inserts. Paired with a 2.5 in screw and a standoff, we can create a wheel that spins with very little friction. Low friction means better performance!



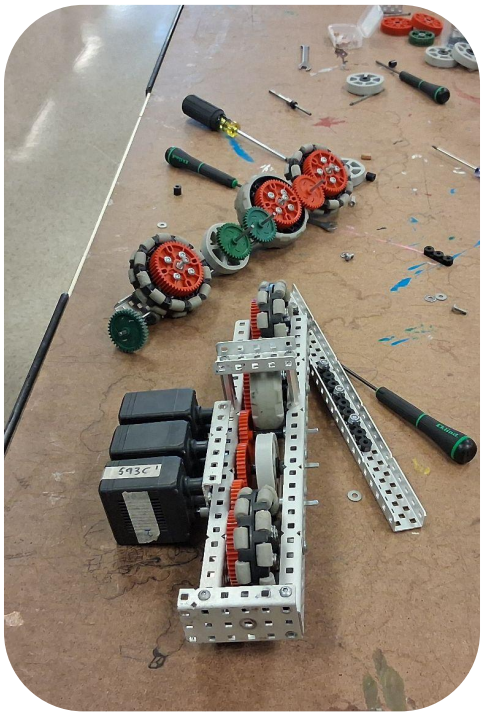
Here is our layout of drivetrain. The series of three gears allows for motor mounting, And also a slightly shorter wheelbase. We wanted a smaller wheelbase to allow for more maneuverability around tight corners on the field.



After mounting the wheels and motors, here is a comparison of last years drivetrain (right) vs our new one. Motors are located near the back to allow more room in the front for an intake!

Drivetrain (cont)

As we continued building, we ran into a few issues, namely friction and the fact that our structure wasn't perfectly square.
These issues were still easily fixed with some rebuilding!

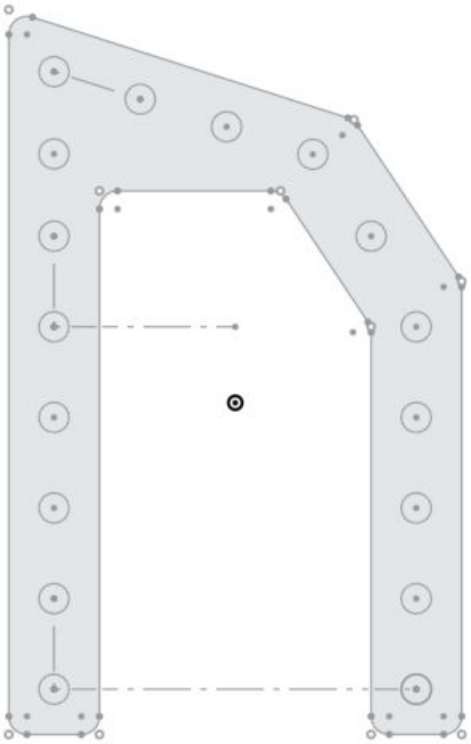


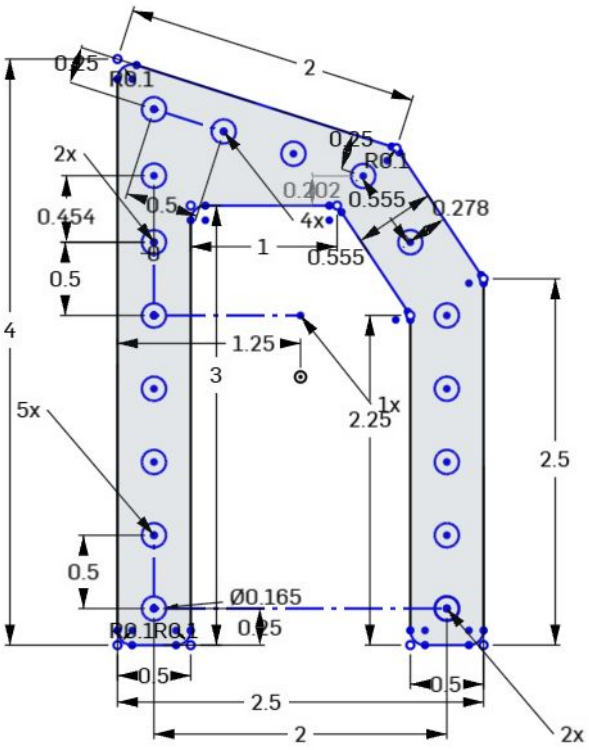
Axles just barely fit between the wheels, as we tried as hard as we could to cram all the wheels in place. A small flexwheel was placed in the gap where a wheel was missing, to allow for easier movement over the parking zone barrier - we won't be able to get stuck on the barrier.

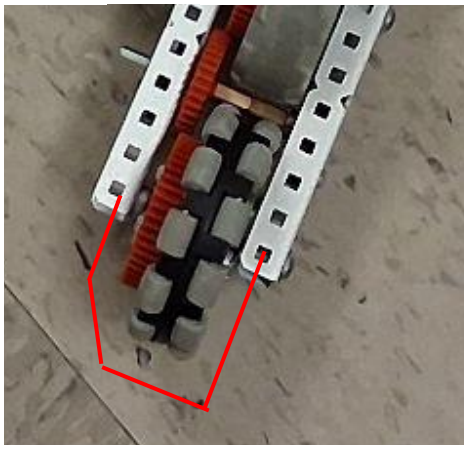
We placed a vertical 3 wide c channel across the back of the drivetrain, and another cross brace across the middle of the drivetrain. We plan to cut out plastic parts to protect the front exposed wheel, and to funnel blocks into the intake.

Drivetrain (cont)

Here is the part we plan to cut out of plastic on our school's Wazer water jet, when we have a chance. It's a bit of a mess, but the paper model we made fits the drivetrain well. The circles are cutouts for standoffs, to reinforce the plastic and provide a funneling surface! The first 3 holes will be attached to the drivetrain.

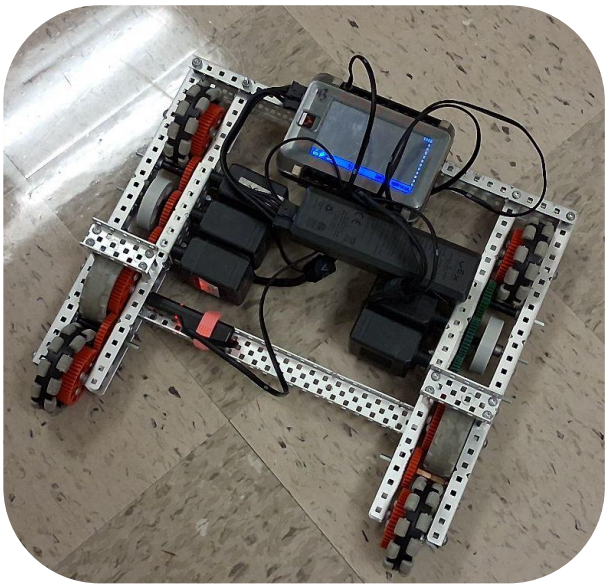


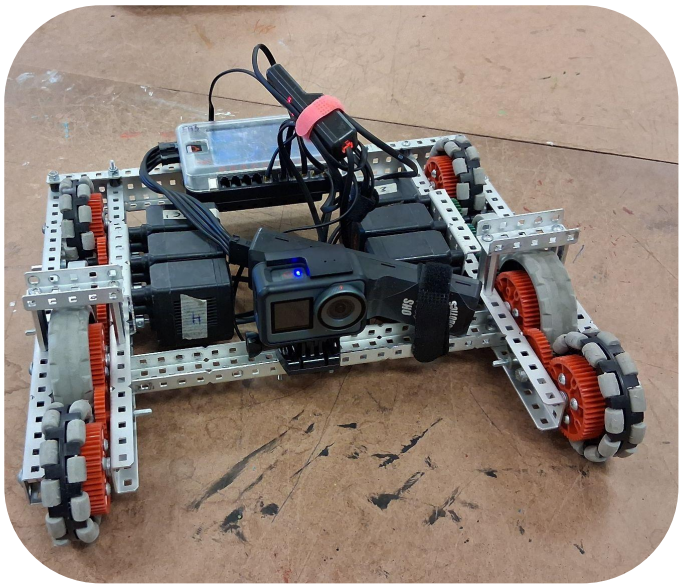


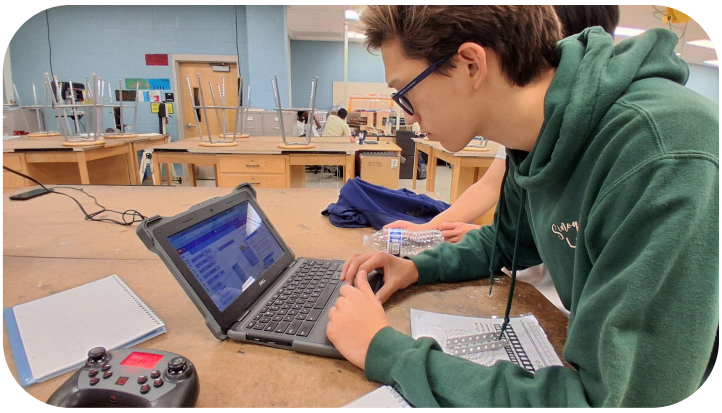


Drivetrain (cont)

After assembling our drivetrain, we attached all the necessary electronics (brain, motors, radio, battery), and programmed to be able to drive around! We also attached a camera, check our YouTube and Instagram accounts for those videos! @oakdale593



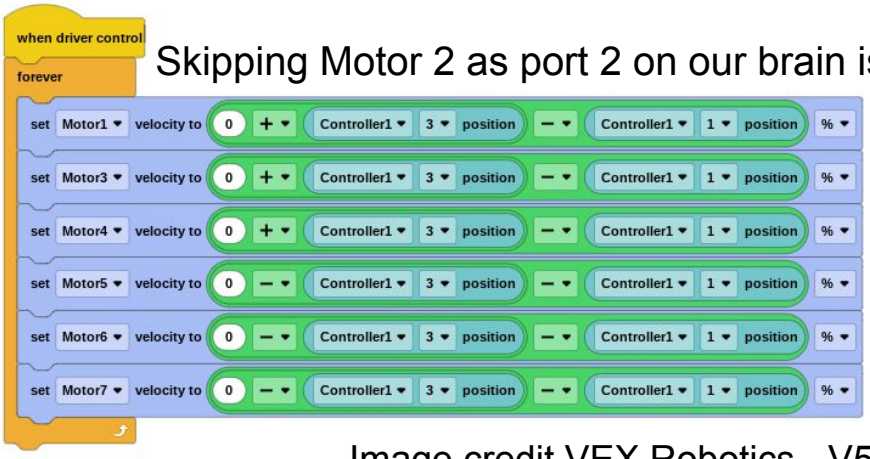




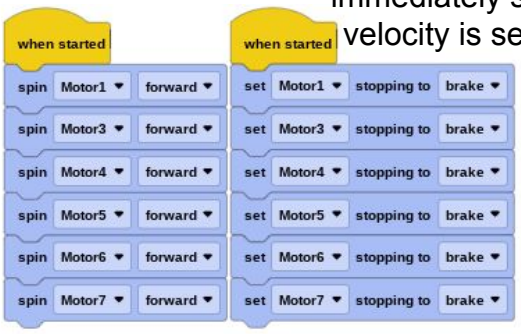
We held some driver practice, pushing around High Stake goals

Programming - Blocks

Now that we've gotten our drivetrain built, we need to program it. We're starting with using VEXCode, for readability



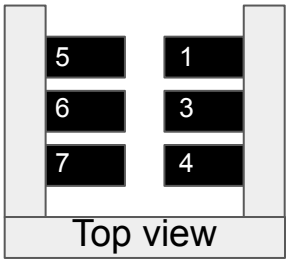
Skipping Motor 2 as port 2 on our brain is fried



Spin motors forward, immediately stop if velocity is set to 0

Image credit VEX Robotics - V5 Controller

Each motor is assigned a port, from front to back, as seen here ---------->



In our block code, the 

block pulls the value of "axis 3," or the vertical position of the left stick, while the other "1 position" block pulls the horizontal position of the right stick. By adding and subtracting these values, then assigning them to the motors movement velocity, we can control the speed of our robot proportionally to how the sticks are used.

Since VEXcode doesn't support motor groups of more than 2 motors, this is a little jank, but it works.
We're also able to drift the bot!

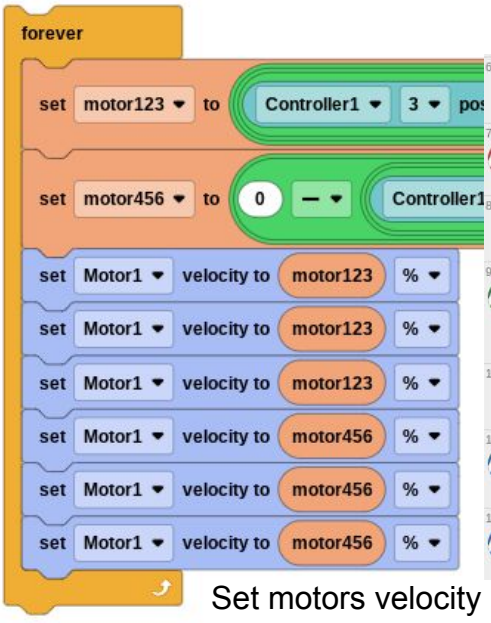
VEX joysticks provide a range of values from -127 to +127, but the motors only accept a range of -100% to 100%, so the outer edges of the joysticks are stuck at 100% power, but that shouldn't be an issue, if we want full speed we'll still be able to reach full speed!

Programming - Control Scheme

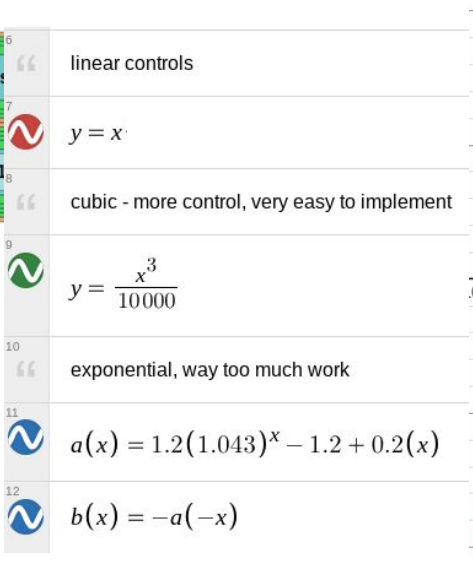
One issue with our current control scheme is how hard it is to control at low speeds due to the fact that speed scales linearly - the speed of the bot doubles much faster at low speeds and it's hard to make precise turns or movements.

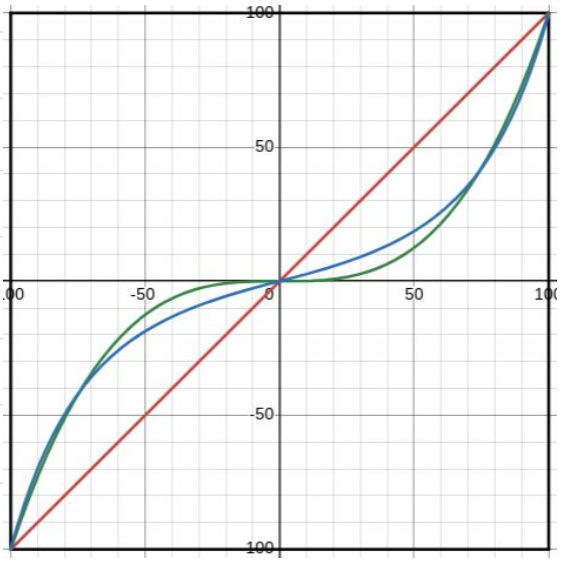
If we change the way inputs are scaled, such as using exponential or cubic functions, small movements at lower speeds are much easier to control

Below is a chart of how these controls will affect the velocity of the bot (y-axis) against the stick position (x-axis), and a cubic implementation of a control system. It's a bit janky due to VEXcode not supporting exponents, but it works.

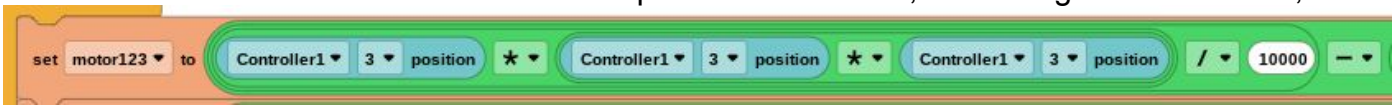


Set motors velocity



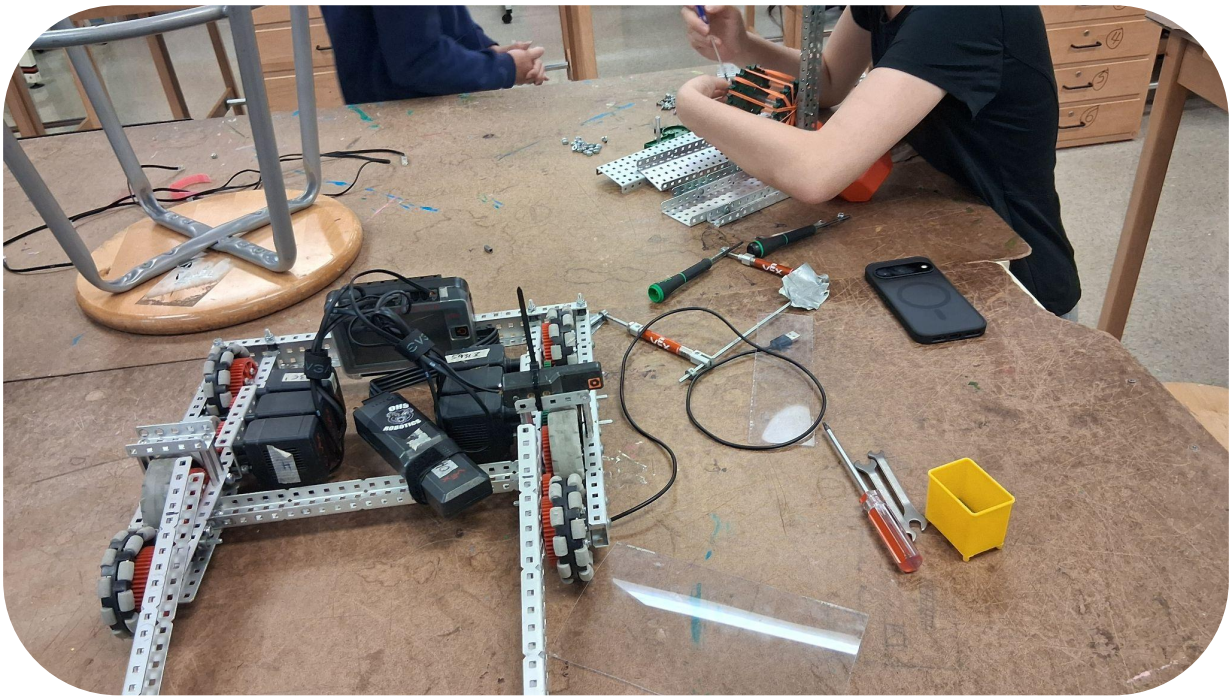


Take inputs from controller, run through cubic function, set to variable



Intake and first stage

For preliminary intake and stage prepwork, we made a model to make sure our idea of using rubber bands on sprockets would actually work, in addition to trying to figure out how our intake would be situated

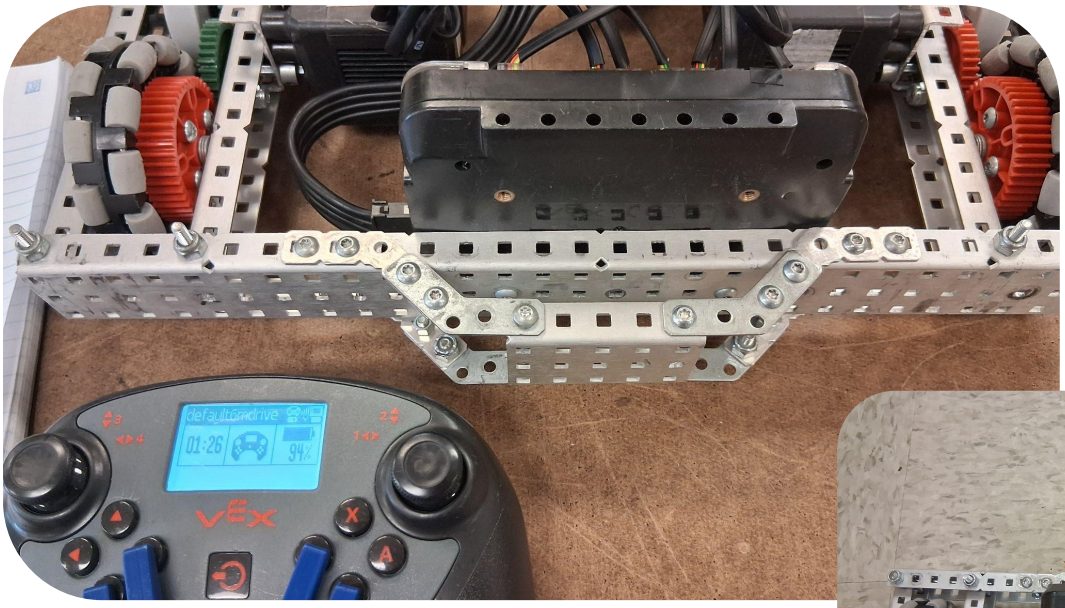


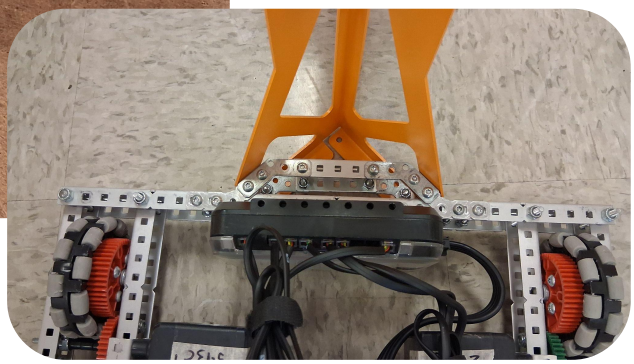
Much time was wasted with this model, but it seemed to work! We're moving on with the S design!

We considered directly attaching our model to the drivebase, but threw that idea out after seeing how bulky it was, so we moved on to building our superstructure.

Aligner

Since we had a functional drivetrain, we thought we should tackle our aligner. Using 8 (or so) 45 degree gussets, we were able to attach a 3 wide chunk of aluminium to the back of the chassis, which perfectly fit into the v at the back of the long goals.

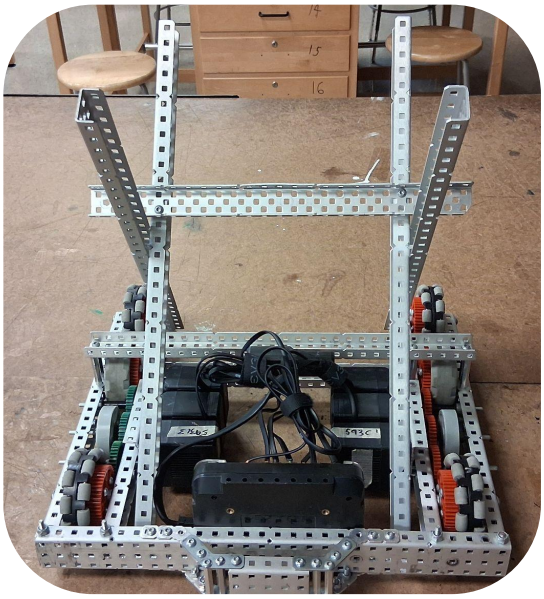


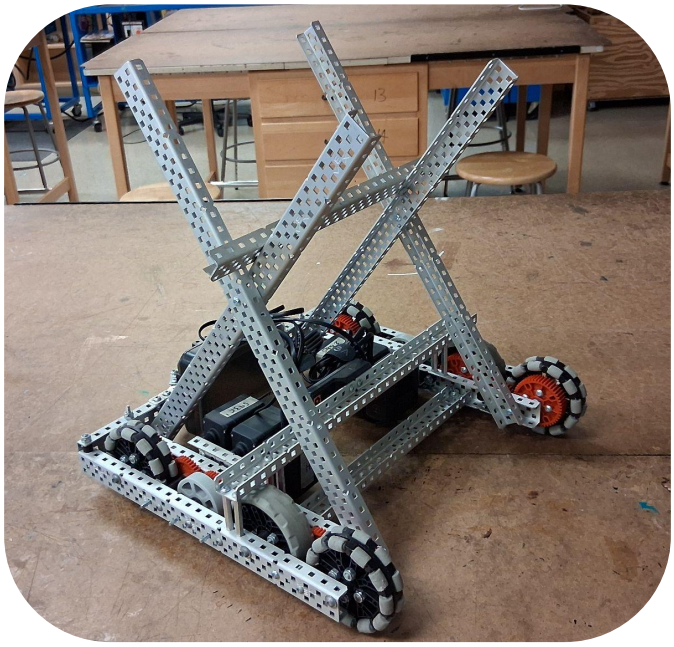


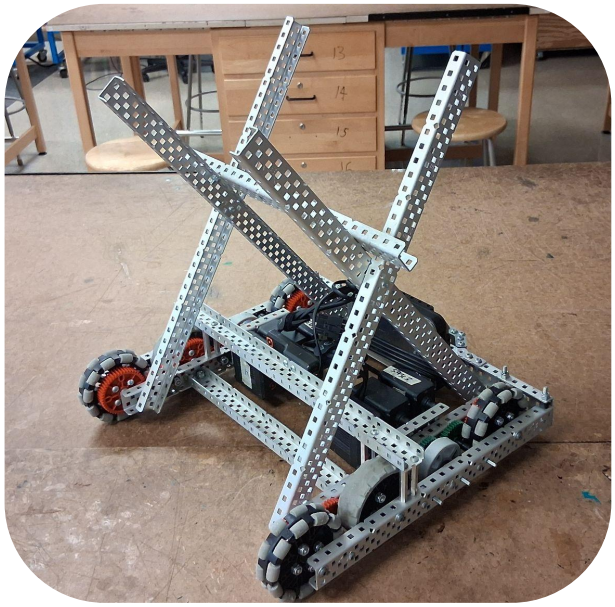
In preliminary testing, this aligner seemed to hold up, and worked to align the bot if we drove full speed into the goal, even at slight angles.
We'll see how this preforms over time!

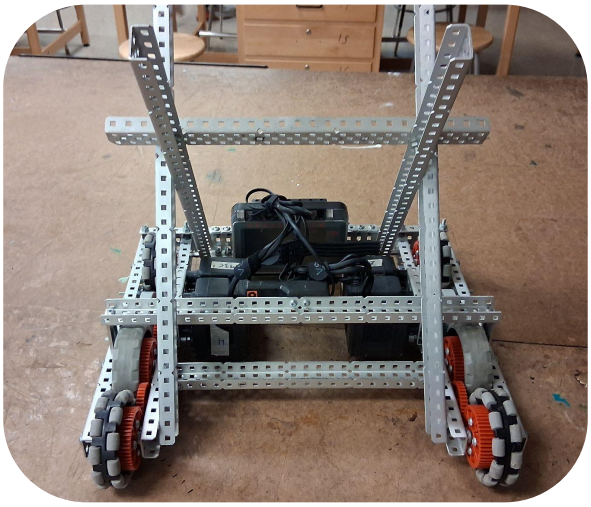
Superstructure

To attach all of our subsystems to, we needed a structure. We had an idea of a crossed superstructure to hold our s shape in shape, so we assembled that using some c channel!
The structure is a bit wobbly, we'll need to add more anchors, but this should be a good starting point.



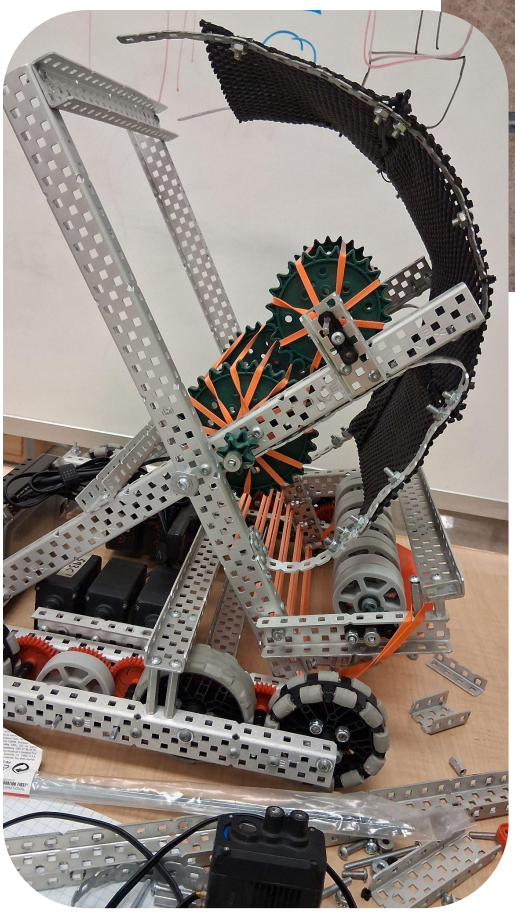


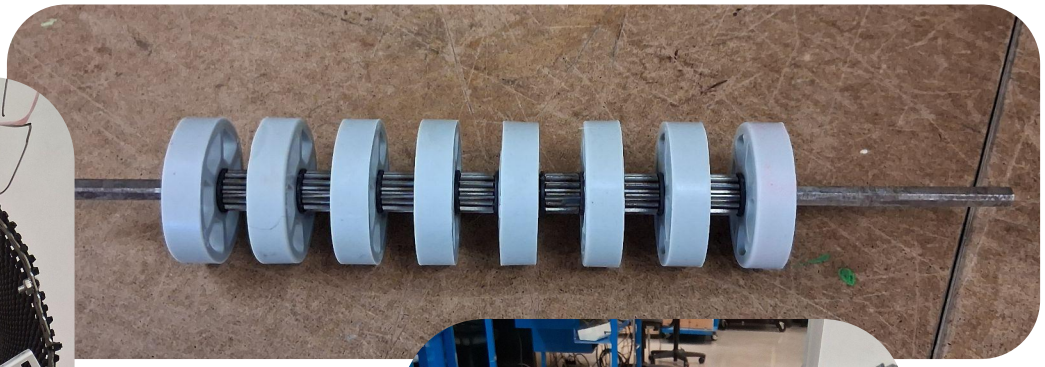


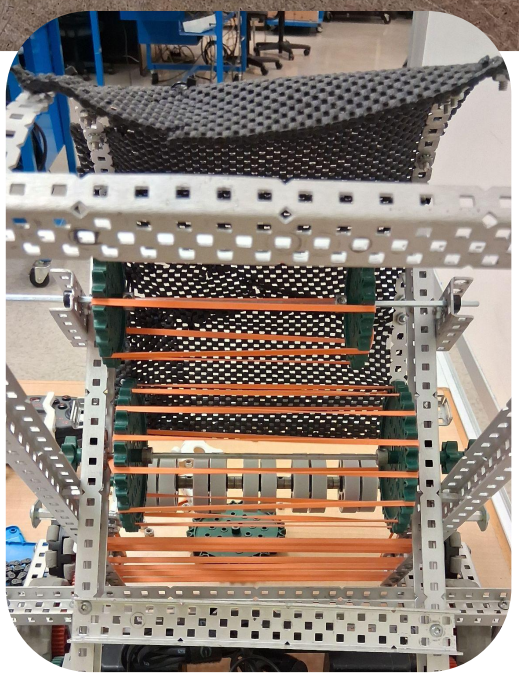


Intake & First Stage

Some time has passed, and we have continued work!
Our intake consists of a low strength axle with flex wheels, spun by a chain. We iterated several designs, to find the lightest intake that still worked, and did not flex. We found adding an extra bar over the flexwheels prevented the flex we didn't want.
In addition, we mounted our first two rubber band rollers to the superstructure, and created a surface for the blocks to rub against with some one by metal sheets and some anti slip mesh. Rubber bands also compose the bottom of our intake, as they are grippy enough to grab the blocks, but can also flex to keep pressure on the blocks. We are currently unsure of how to drive the intake though.

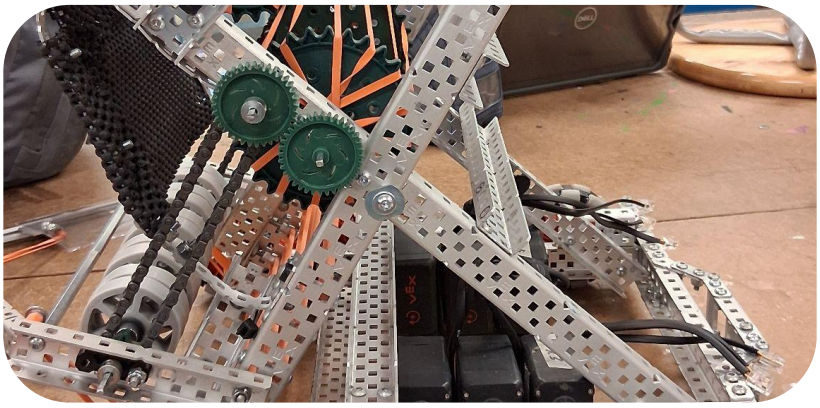




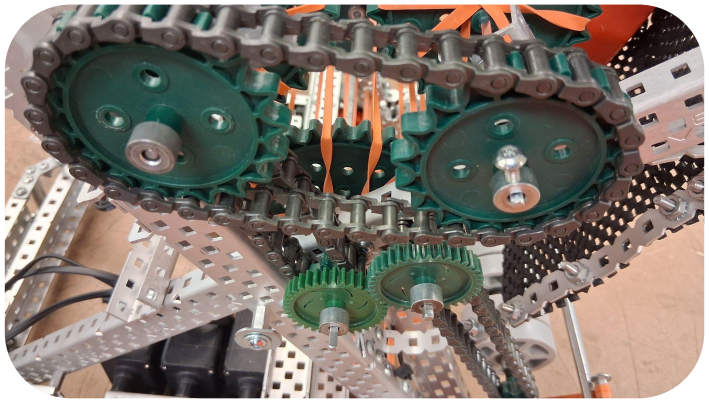


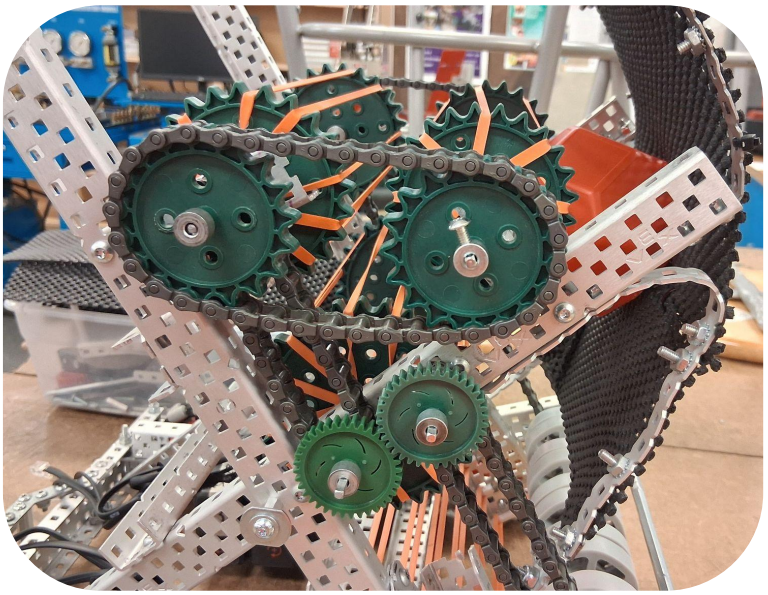
Intake & First Stage (cont)

We figured that if we cantilever some gears, we can reverse the direction of the intake, allowing for all wheels to suck in blocks!



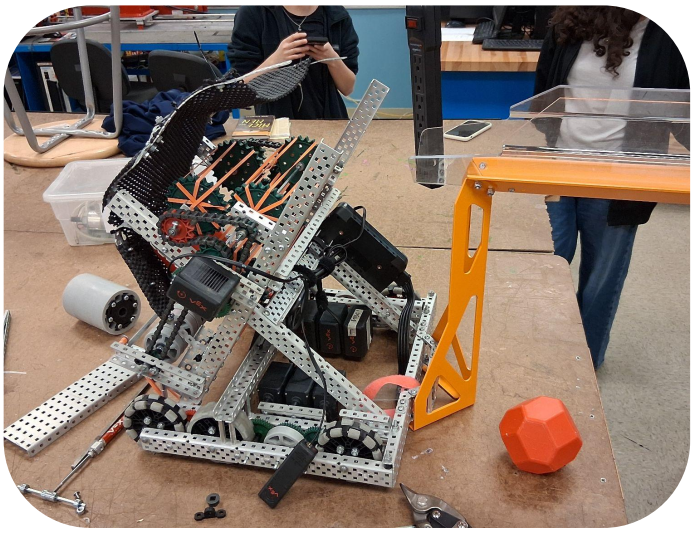
There's a lot of friction that needs to be fixed, but for now this works!
In addition, we attached a third rubber band roller, but this chain routing is not cutting it, we need to iterate and find a better chain route/ motor mounting spot.
These cantilevered sprockets are really not helping either!

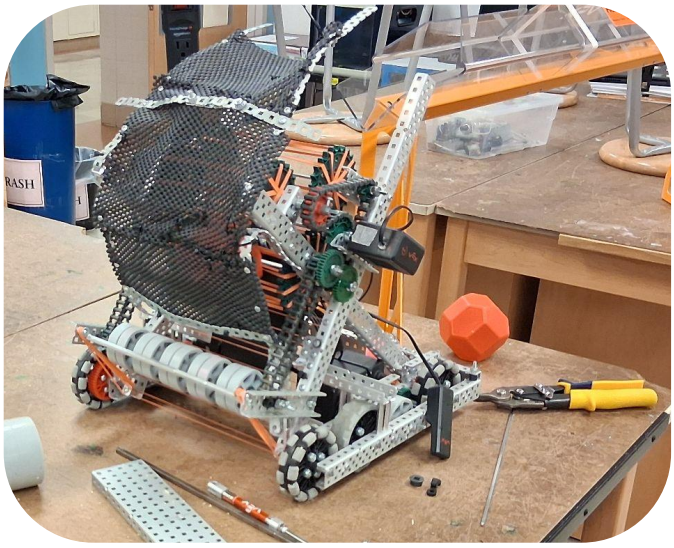




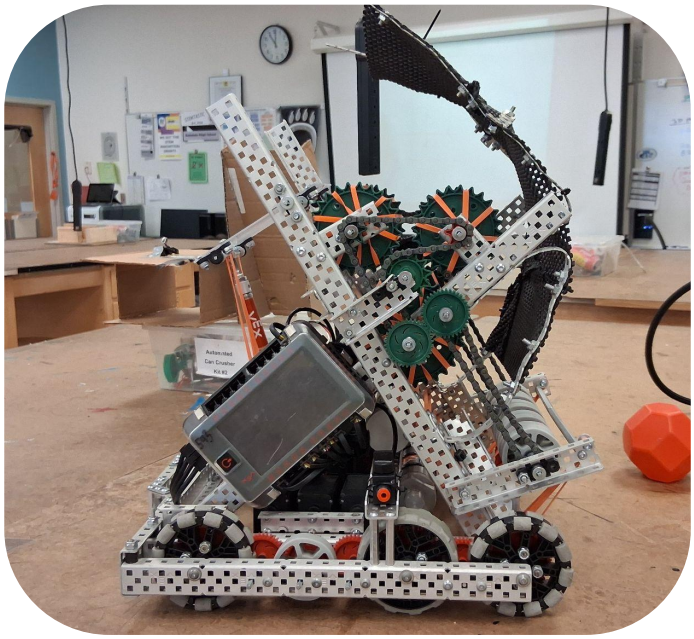
Intake & Second Stage (cont)

We rerouted the chain, and mounted our motor!





This system seems to work well, and intakes blocks off the floor. We also changed the intake slightly, as the chain was sucking the intake up - by using rubber bands to pull it down we prevented this clogging issue. We also opted for a smaller intake stabilization method with standoffs instead of a c channel.
Oliver attached a plastic sheet with a piston to allow for adjustment of where blocks are directed when scoring, which should allow for easier scoring down the line.
Chain routing was adjusted/rebuilt to allow for a little less friction.

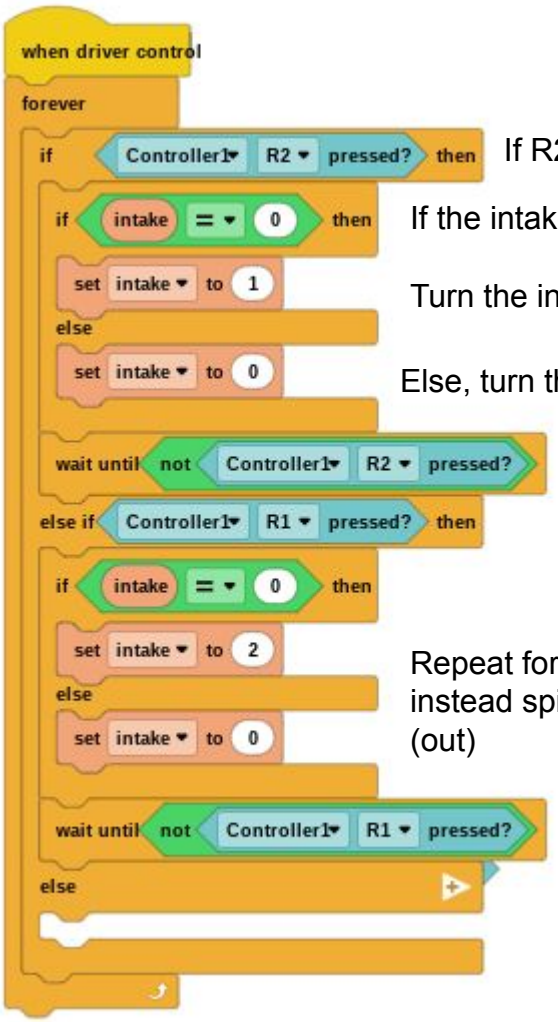


Programming - Intake

Since our intake and second stages are chained together, we can really easily control them with button presses.

We want to be able to stop the intake at any point with either right trigger button if the intake is spinning, and control direction with the trigger buttons.

If we create a new variable, we can change that variable to different values based on how we want the motor to spin.



If R2 pressed (bottom right trigger)

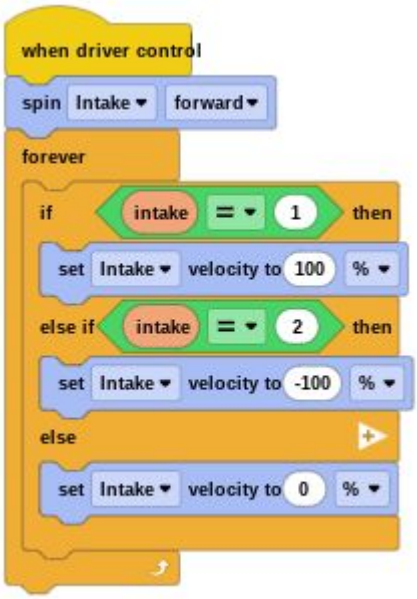
If the intake variable is 0 (intake off)

Turn the intake on (forward)

Else, turn the intake off

Wait for the button to be released so the intake doesn't turn on and off repetitively

Repeat for R1 trigger (top right) but instead spin the intake backwards (out)



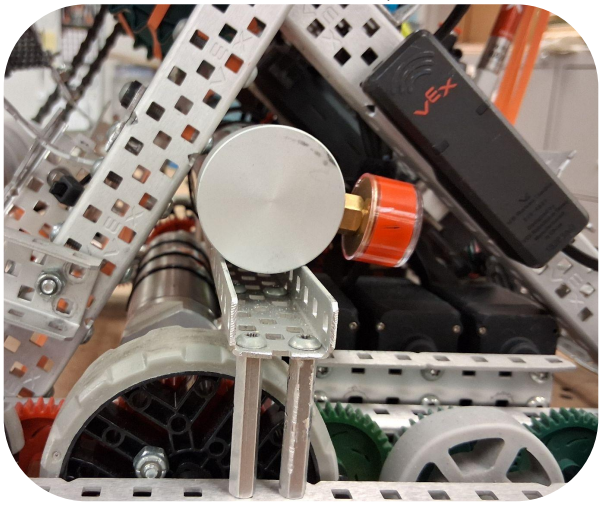
Spin forward

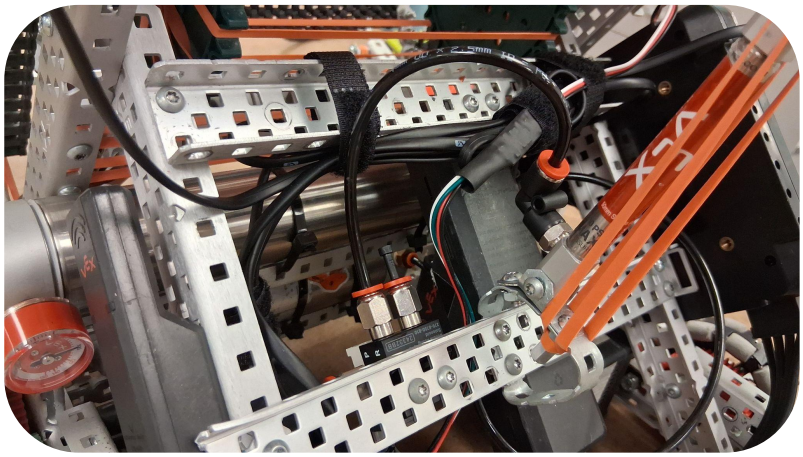
Spin backward

Don't spin

Pneumatics

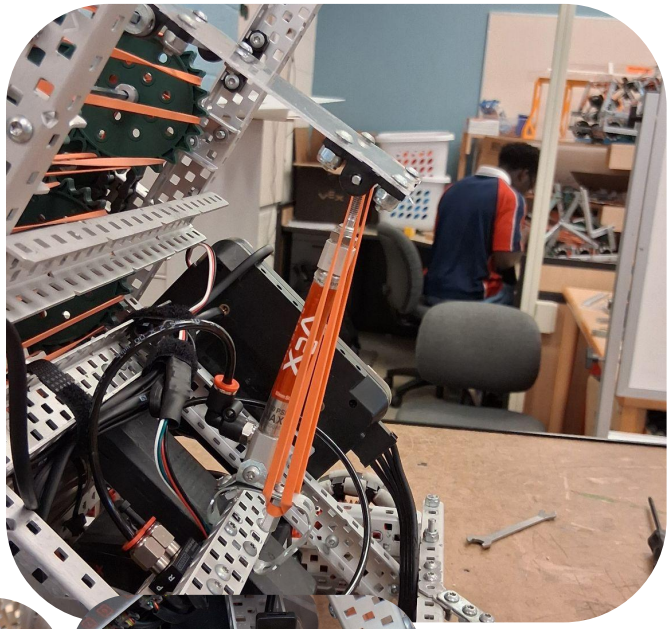
Since such was the time to start using pneumatics, we installed and secured tanks, solenoids, and wires!

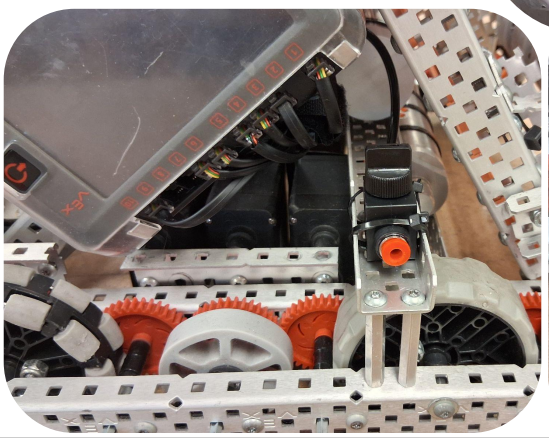


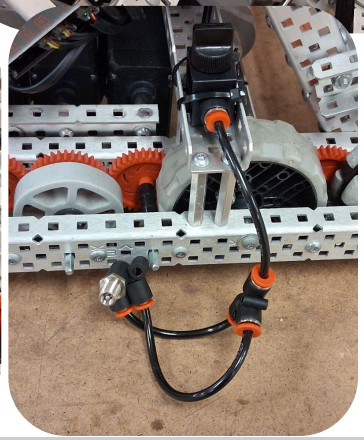


In addition, we started trying to attach pneumatics to the match loader to ensure we stayed within sizing, as possible.

One cool thing we learned from another MD team was to use one of these fittings, pipe it into a shutoff valve, and Be able to turn on/off the valve to prevent the loss of any air when filling tanks!

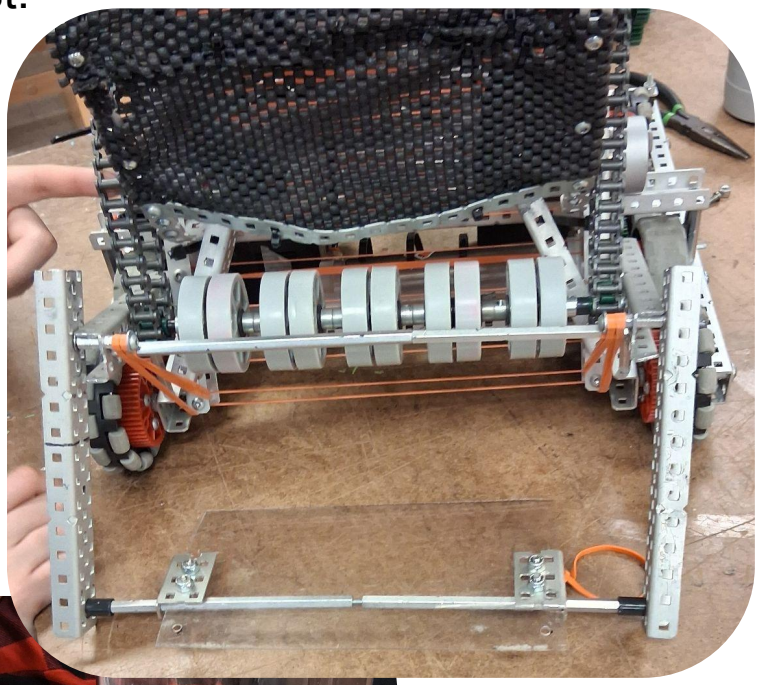




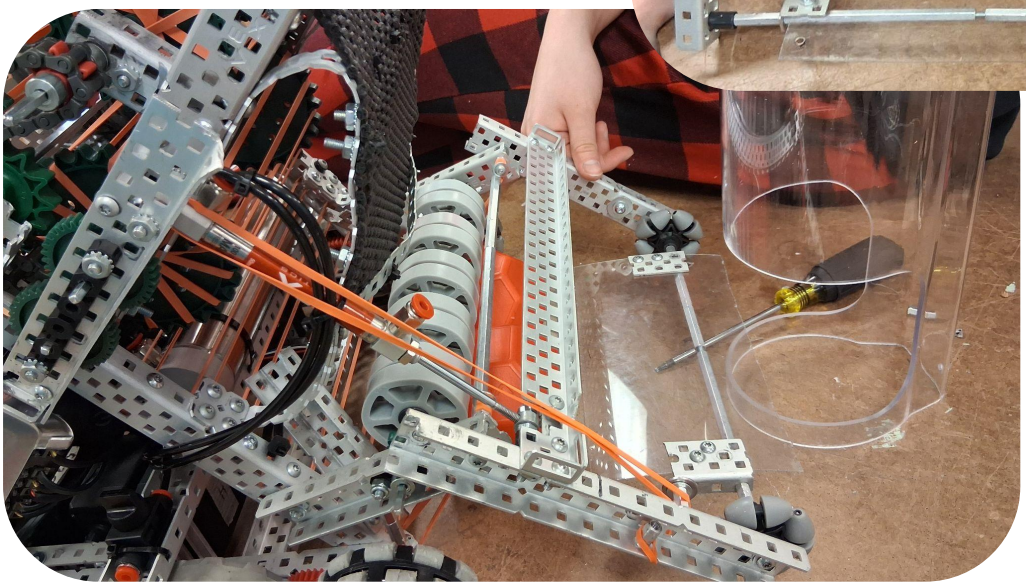


Match Loader

Working on this over the months was fun!
Our match loader is essentially a sheet of plastic on an arm we jam into the loaders to bring blocks out of the loader.
We were going to attach it to the drive base, but found attaching it to the intake would be the easiest.
It seems to work to dump blocks directly into the intake, and was easy enough to build and install.

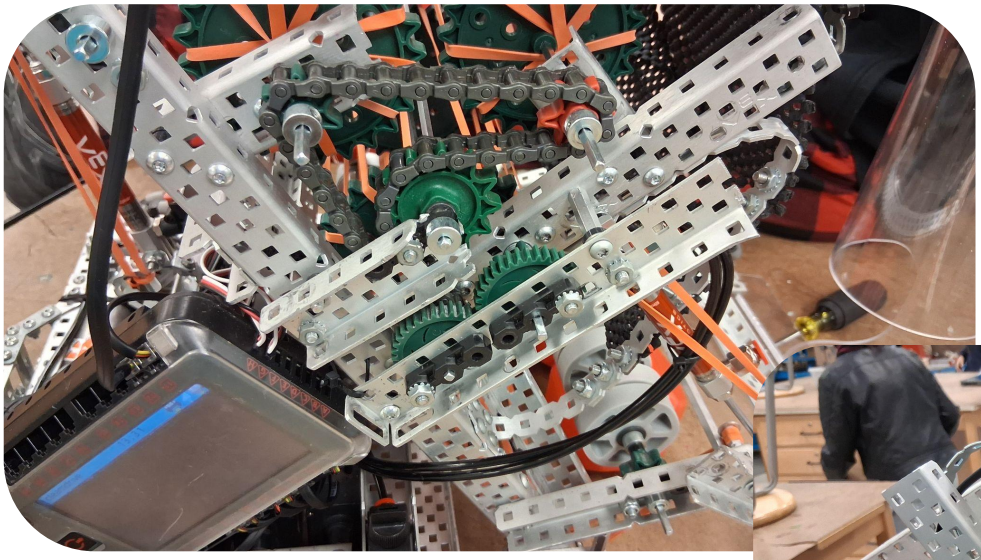


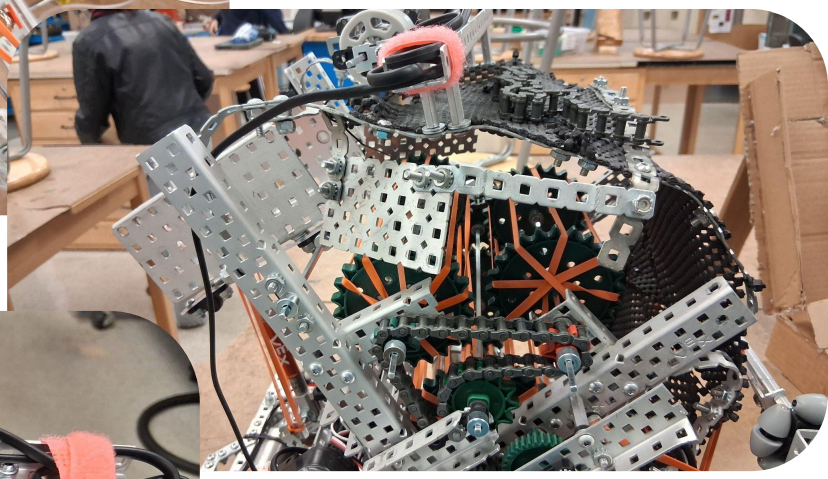
Around this time we also finally removed the cantilever from the intake gears, as that was adding a lot of friction.

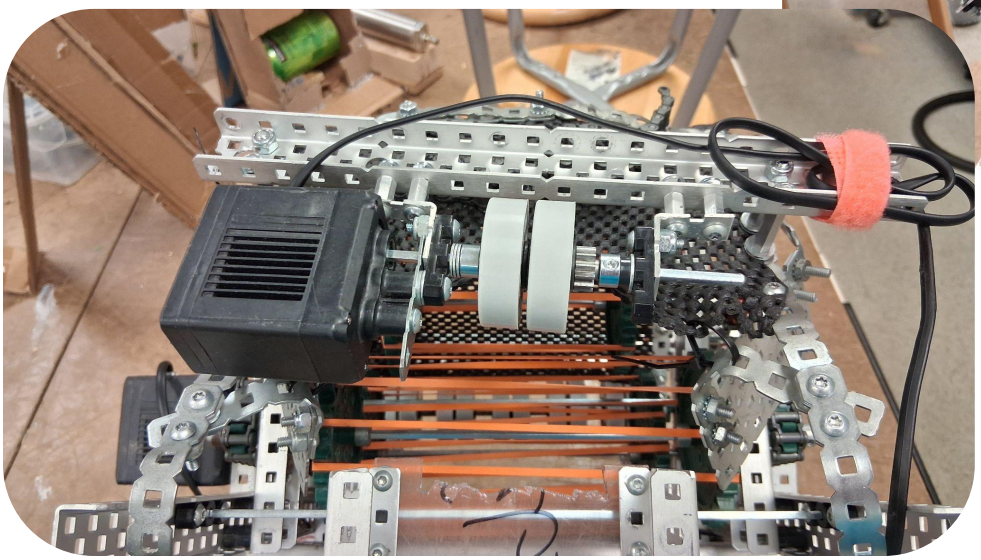


Outtake & Stages

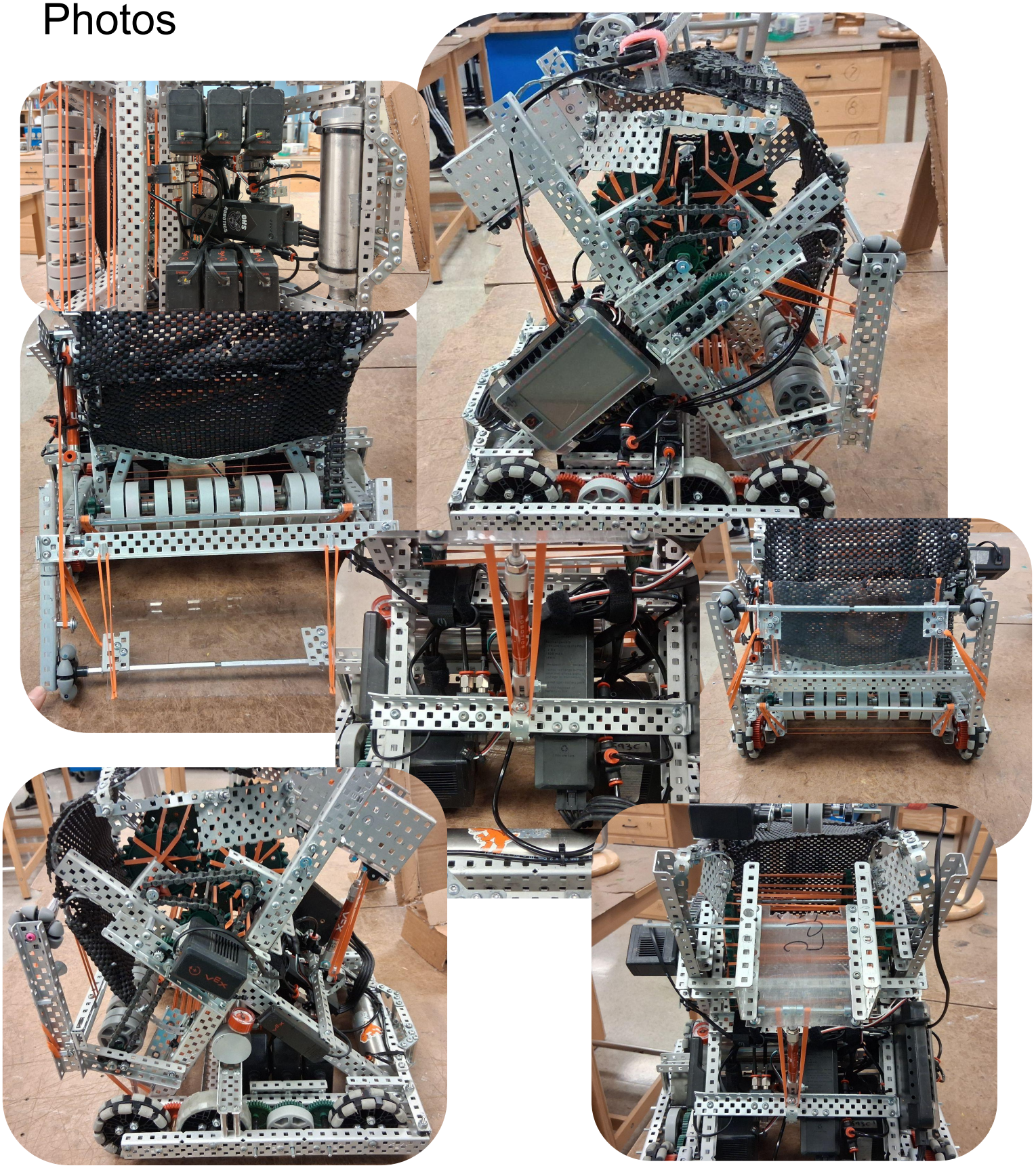
We need a way to hold blocks within the bot, but still be able to score onto the goals, so we devised this outtake system with an extra motor and flexwheels. The flexwheel's grip hold the blocks in place, and can still push blocks out! Oh, did we mention we fixed the cantilevered gears, and found smaller sprockets?





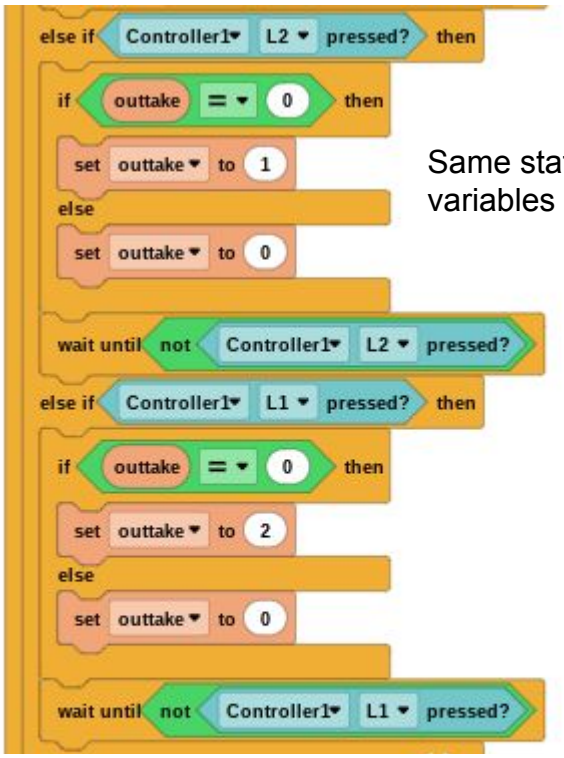


Photos

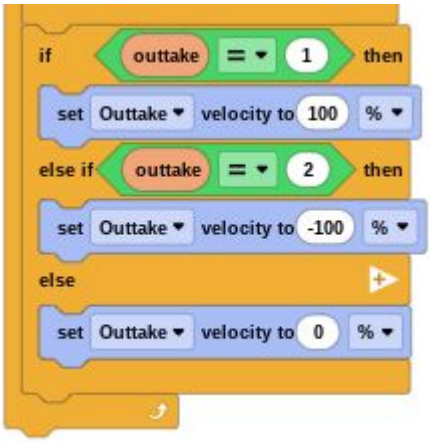


Programming - Outtake

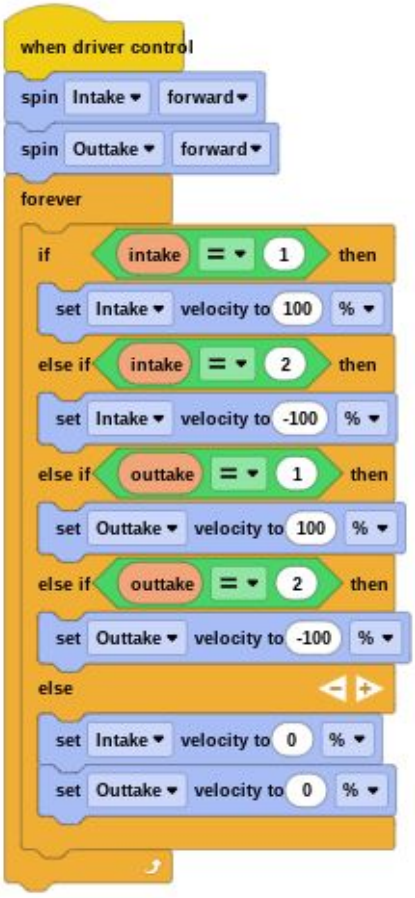
We want out outtake to be controlled the same way as our intake, so we can just duplicate that code and change some values! See the previous “Programming - Intake” page for more reasoning behind this control scheme.



Same statement as before, but variables and motors are changed



We did have a few issues with getting the two systems to play nice with these if statements, as if we used a 4 statement else if block (see right) for controlling the motors with variables, only one motor would spin at a time. As the if else block checks for the first statement that is true then moving on, only the first statement would be run, so the outtake could never turn on if the intake was on. This was fixed with a second if statement.



Programming - Pneumatics

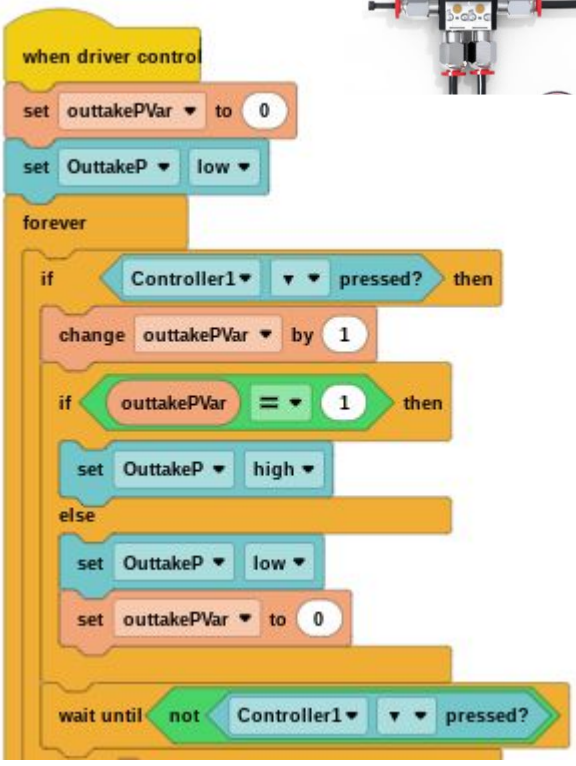
We plan to use two pistons on our bot at the moment, one to control the height of the outtake, and one to control the position of the match loader. These Pneumatics are controlled by solenoids, over a 3wire connection.

3wire is a simple way to power sensors and outputs, as all that is needed is 3 wires, power, ground, and data, hence the name. Seen right is a 3wire cable plugged into the brain, with the solenoid attached.

Images credit VEX Library - RECF

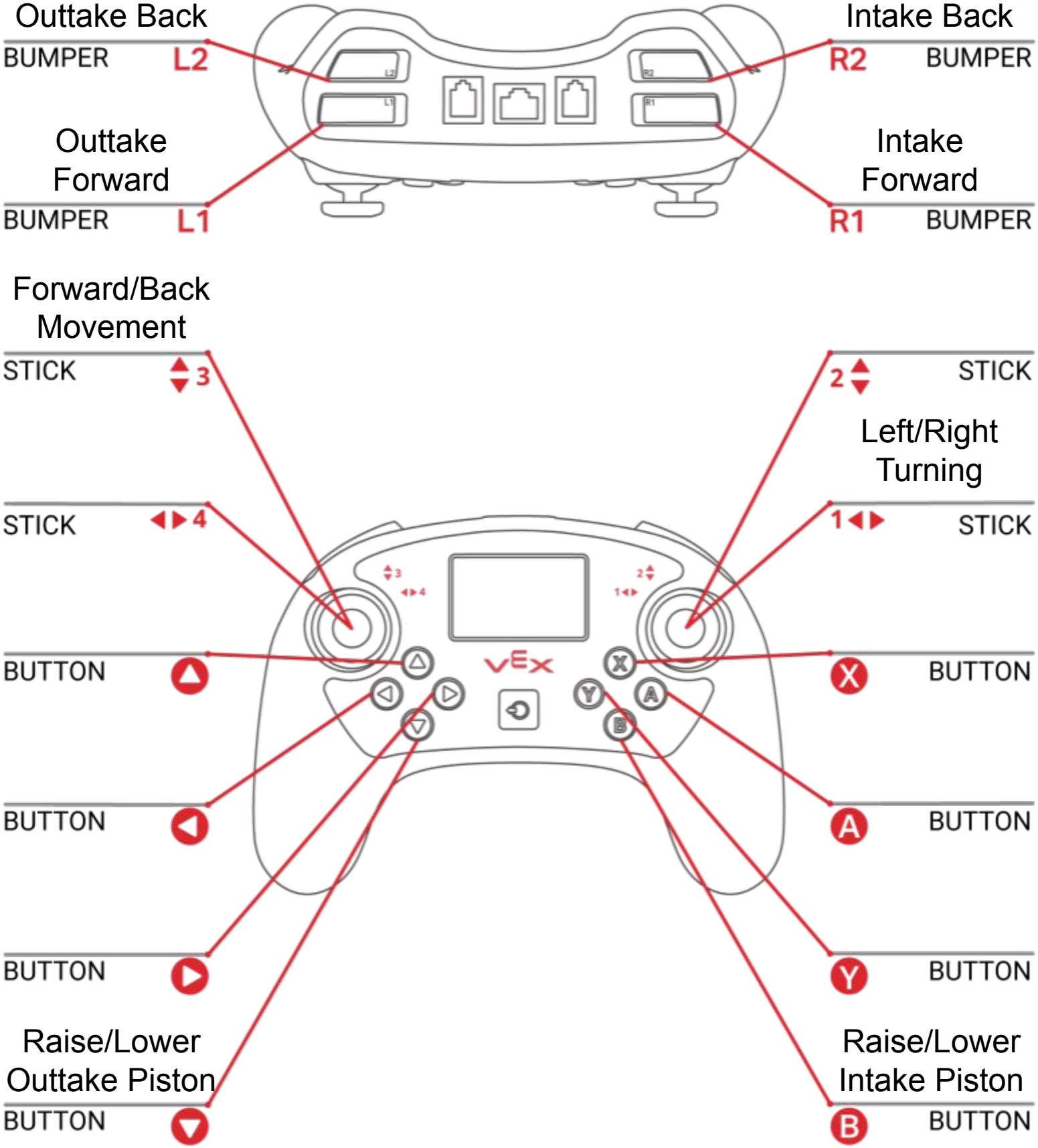
We can control if the solenoid is on or off, also known as “HIGH” or “LOW” using a DigitalOut device in VEXCode.

When a button on the controller is pressed, we change a variable by 1. If the variable = 1, then turn the solenoid on, but if it doesn’t and the button was pressed, turn the solenoid off, and set the variable back to 0.



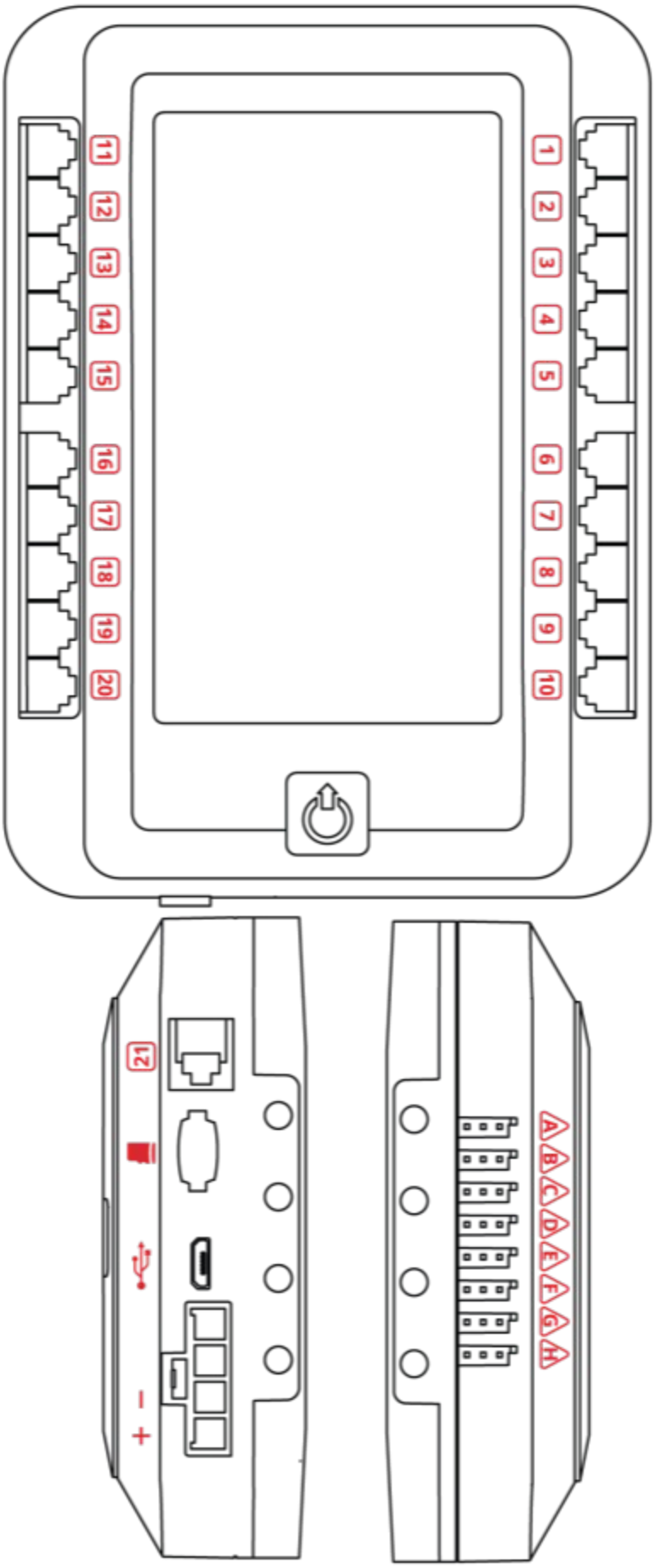
This is just duplicated for the second piston, it works exactly the same way, just with a different button.

Our solenoids are plugged into ports G and H, as they were the easiest ports on the brain to access, and the solenoids are mounted in various points across the bot.


Outtake Back
BUMPER L2
Intake Back
R2 BUMPER
Outtake
Forward
BUMPER L1
Intake
Forward
R1 BUMPER
Forward/Back
Movement
STICK 3
2 STICK
Left/Right
Turning
STICK 4
1 STICK
BUTTON
X BUTTON
BUTTON
A BUTTON
BUTTON
Y BUTTON
Raise/Lower
Outtake Piston
BUTTON
Raise/Lower
Intake Piston
B BUTTON

PORT 11
Outtake
PORT 12
PORT 13
PORT 14
PORT 15
PORT 16
PORT 17
PORT 18
PORT 19
PORT 20

PORT 21

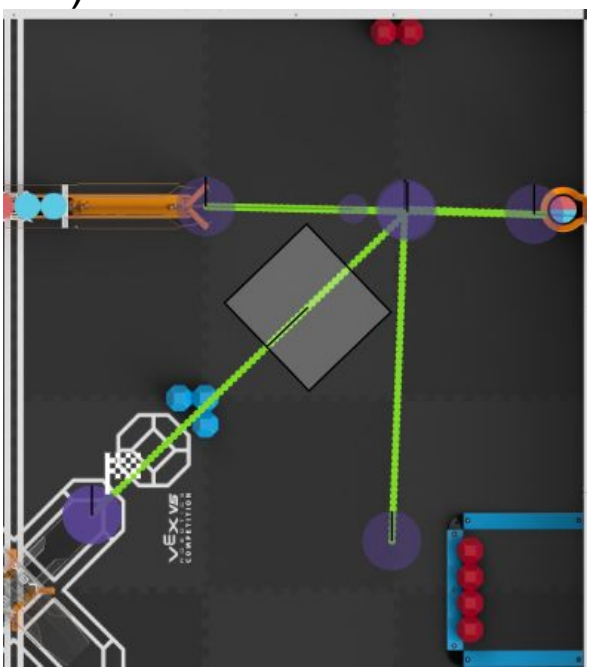


Drive 1
1 PORT
2 PORT
Drive 3
3 PORT
Drive 4
4 PORT
Drive 4
5 PORT
Drive 5
6 PORT
Drive 6
7 PORT
8 PORT
Radio
9 PORT
Intake
10 PORT

A PORT
B PORT
C PORT
D PORT
E PORT
F PORT
Intake Piston
G PORT
Outtake Piston
H PORT

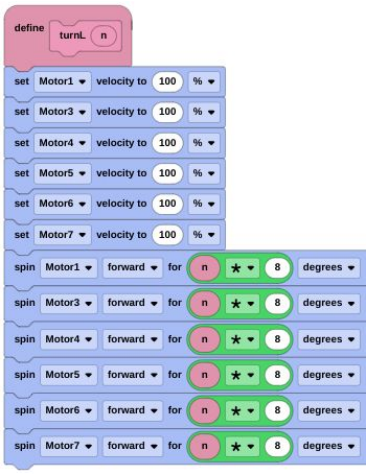
Programming - Autonomous

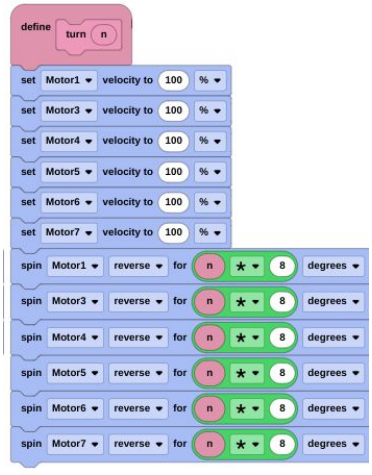
We choose this path due to it would be easy to code and it would score both long and middle goals. (Seen here)

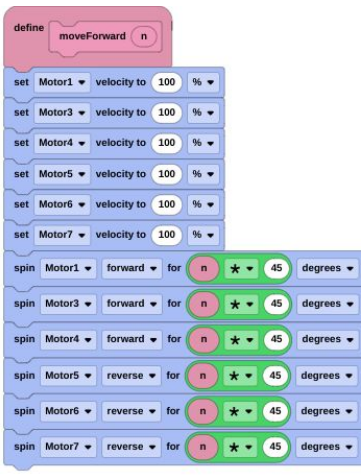


The “moveForward” function takes “n”, n being the amount of inches we want to robot to move forward. We calculated that the robot moves 8 inches forward with one full rotation of the motors. Making moving the motor turn 45 degrees for every 1 inch we want to move the robot.

The turn functions takes “n”, n being the amount of degree we want to turn. We calculated the robot moves 45 degrees with one full rotation of the motors. Making the motor turn 8 degrees for every 1 degree we want to rotate the robot .

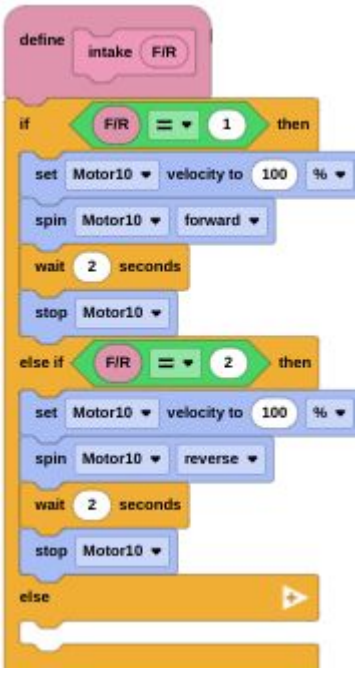


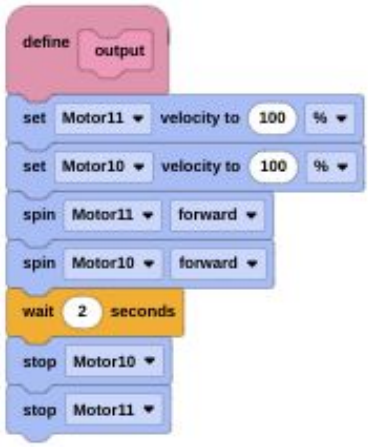




Programming - Autonomous

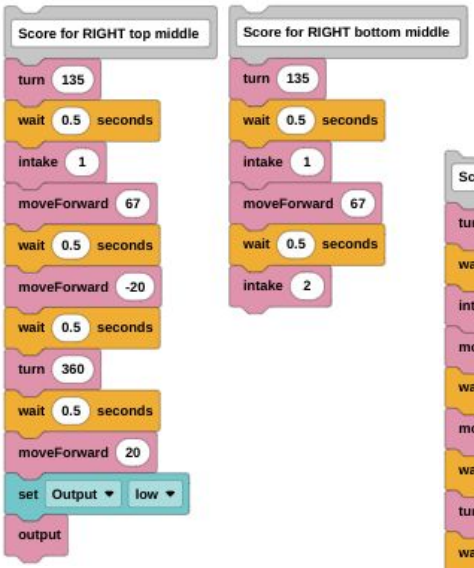
The intake Function takes “F/R”, F/R meaning forward or reverse for the intake motors. If F/R = 1 the intake will spin forward and if F/R = 2 the intake will reverse so we can score the lower middle goal.

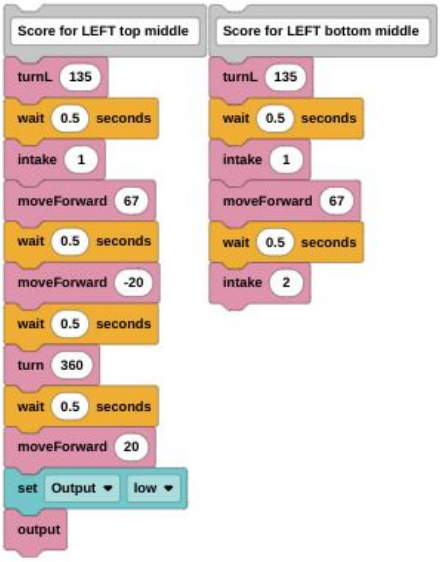




The output Function spins the intake motors while spin a top motor to score the long and top middle goals.

Depending if we want to score the bottom or top middle goal. The robot would turn towards the middle goals with its output or intake facing the goal.

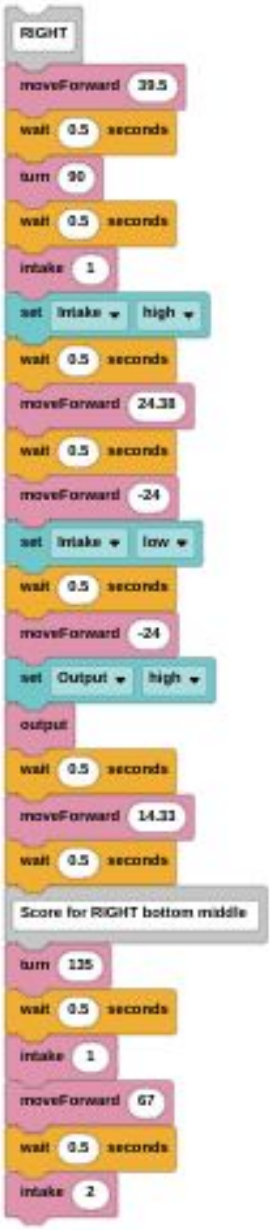


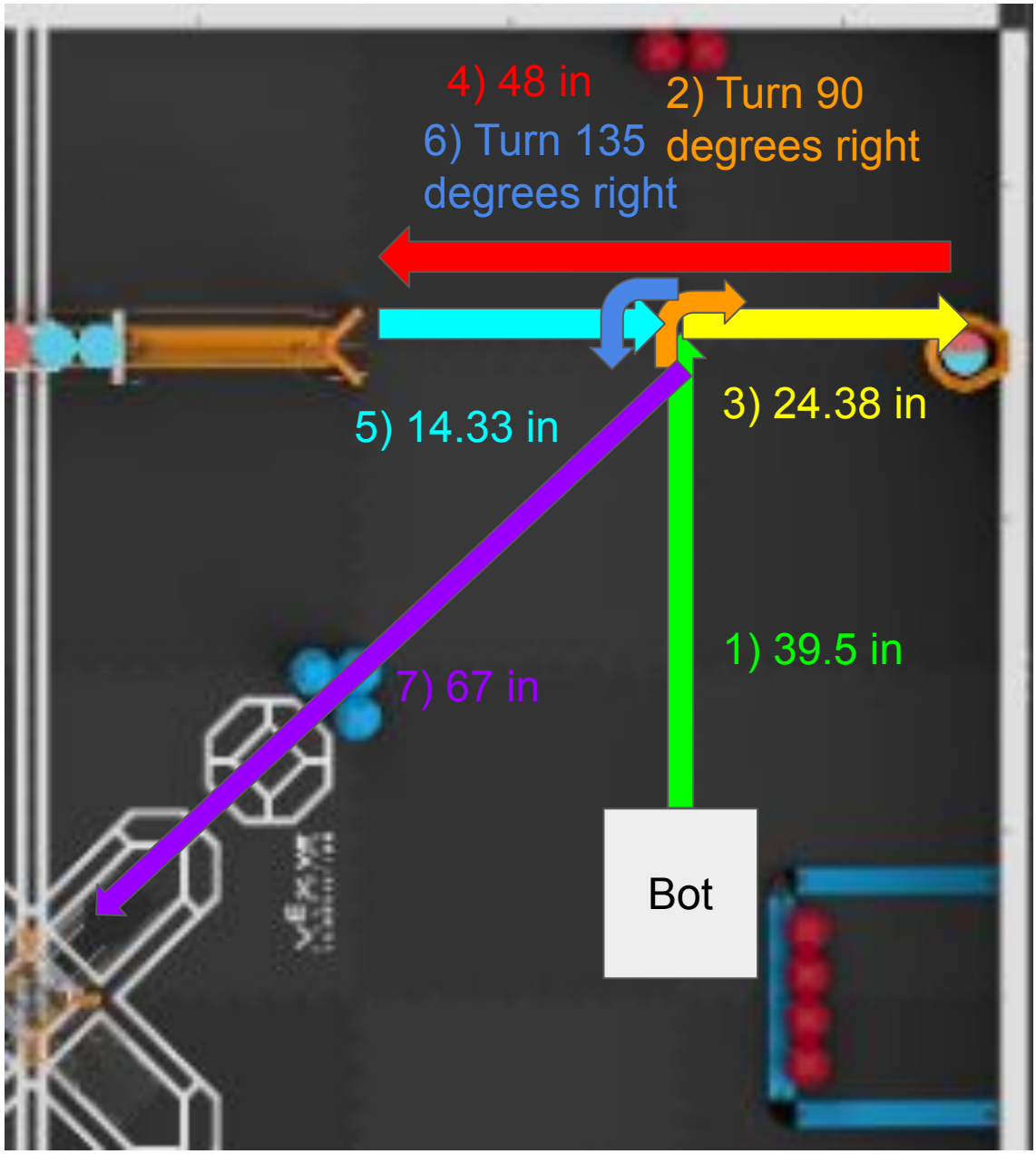


Depending if we want to score the top middle, we make the output piston lower to score and if we want to score the bottom middle we reverse the intake.

Programming - Autonomous

This is the general path our robot will be taking

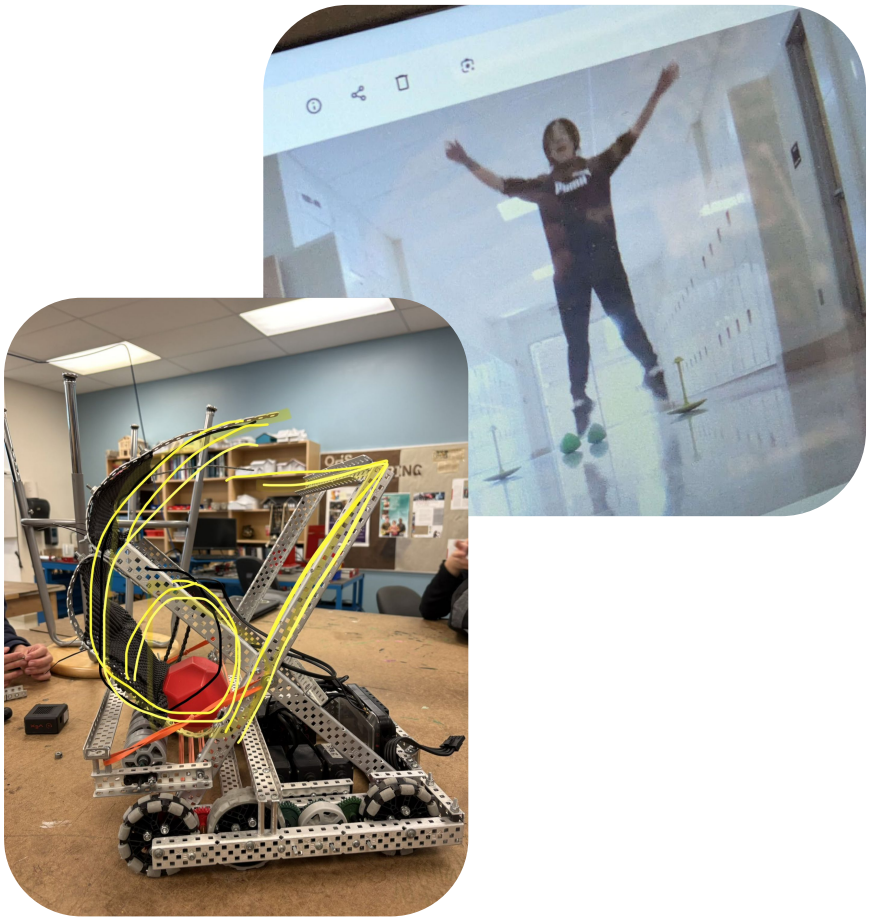


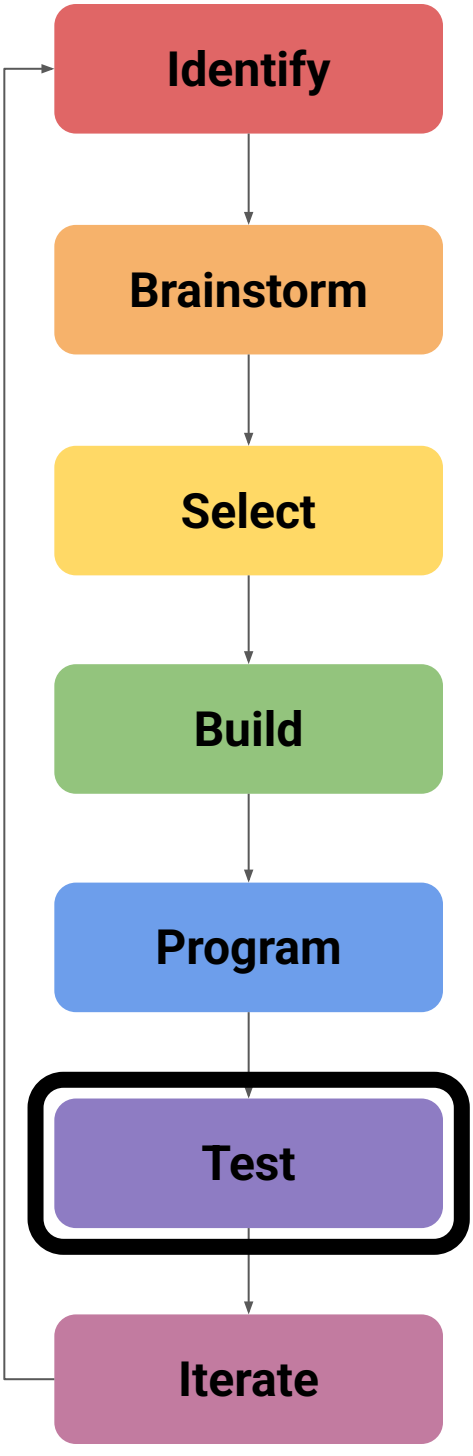


The robot moves 39.5 inches forward to center to the loader, turns 90 degrees, then moves 24.38 inches to drop the intake and collect blocks. It then backs up 48 inches, retracts the intake, and scores in the long goal. Next, it backs up 14.33 inches, turns 135 degrees right to face the middle goal, moves 67 inches forward to collect three center blocks and reverses the intake to score in the bottom middle goal.

Test Solution
The EDP

“Test Solution” - Records testing of the solution, including test results and a summary



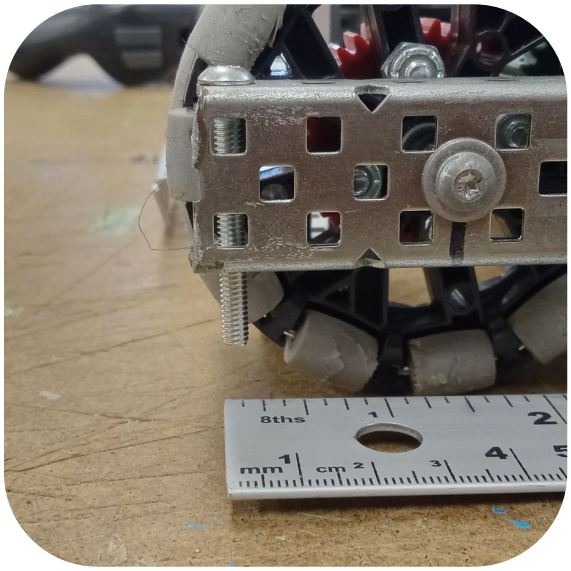


Testing effectiveness of robot

We are unable to effectively test our robot in scrimmage matches, as there are no other teams that have effective robots currently.

However, we can test for slop within our drivetrain, precision, and scoring viability.

First, testing for slop.
We placed a screw through the drivetrain, and pulled the drivetrain as far forwards as it would go, before the motors caught it. We then placed a ruler beneath the screw, and pushed the drivetrain as far back as it would go.



From this test, we find that our drivetrain has ~0.5in of slop, or area where the robot is not very precise, due to spacing between teeth of gears outside and inside motors, and around axles.

This data means our robot's positional accuracy will always be ± 0.25 inches from where we expect it to be, which we will need to account for in auton. This can be proven by movement tests

Testing Accuracy

For the setup, we programmed our robot to travel 1, 3 and 6 feet forwards, stop, then return back to the start, to test for positional accuracy.

	Test 1	Test 2	Test 3
1 ft	1/16 in ahead	1/8 in behind	1/16 in ahead
3 ft	1/4 in ahead	1/4 in behind	1/2 in ahead
6 ft	1/2 in ahead	3/16 in behind	1/16 behind

This test shows that our robot is more precise than we realized, it never reached worst case scenario for slop in testing.
While this may not be the best test, as the robot was traveling at 75% velocity without any turns, it shows that short autonomous programs are possible to be completed accurately.

Testing Scoring

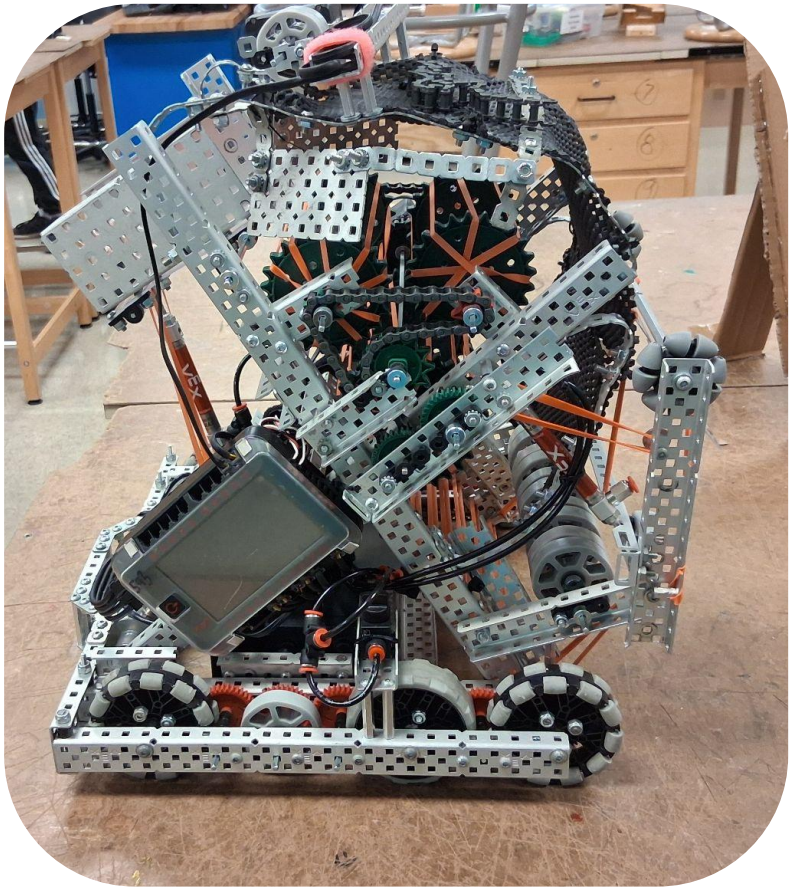
To test scoring percentage, we tested various angles where we picked up blocks off the floor

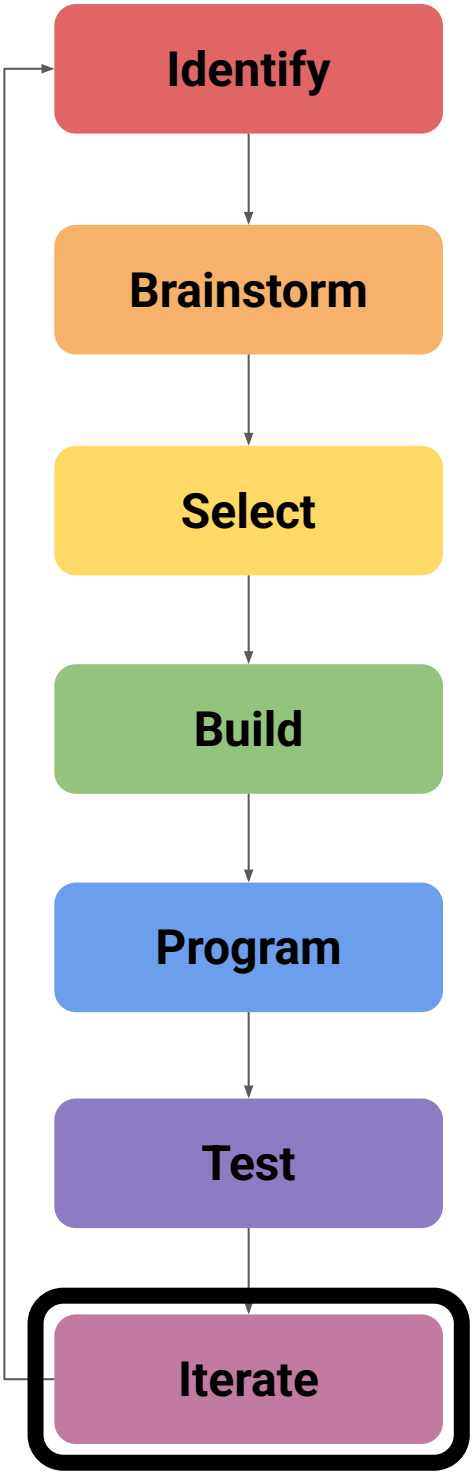
	Test 1	Test 2	Test 3
Side of inyake	Fail	Fail	Fail
Center	Success	Success	Success
2 at once	Success	Success	Fail
3 in a row	Success	Success	Fail

Overall, our system works well head on, but gets stuck if blocks end up where we don't expect them to be. This is something to improve on in the future!

Iterate
The EDP

“Repeat Design Process” - Records what went right, wrong, and what we can improve.





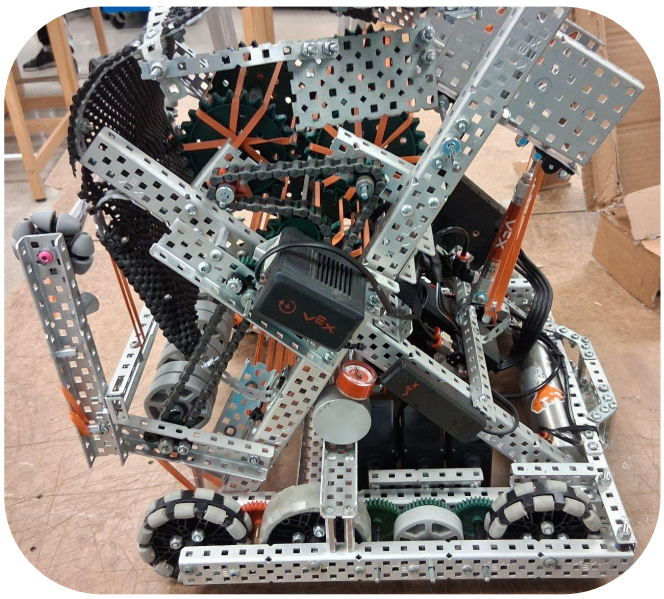
Iteration

Iteration is present throughout the entire design process, and is documented throughout. We've encountered many issues we've had to solve; friction, sizing issues, pneumatic troubles, cut and fitment, gaping holes in plans etc. We've had to return to the drawing board hundreds of times throughout building our robot, sometimes without even noticing it.

A massive thing we do need to work on throughout our iteration cycles, is documentation. We'll admit it, we are not that good at it. Much of the iteration is cut out, because we found it boring to document, or forgot to take photos.

While this is not the exact bot we planned on so many months ago, the general idea we started with o stayed much the same in general. Many of the subsystems we had planned originally we found just would not work, and had to be improved on or replaced with better ideas.

There are several things we would like to improve on, namely being our second stage, we believe that will be the biggest failing point for us, due to overheating motors and blocks getting stuck.



Rebel Rumble Match Strategy

Our first competition is this Saturday! We really are not prepared, but that's ok, this is only just the first competition.

A big part of our strategy is to try to score as much as we can while blocking our opponents, as we don't really have a viable way to descore. We're also trying to tune our autonomous program ASAP, as being able to score that Auto Bonus or AWP will be huge in trying to do well.

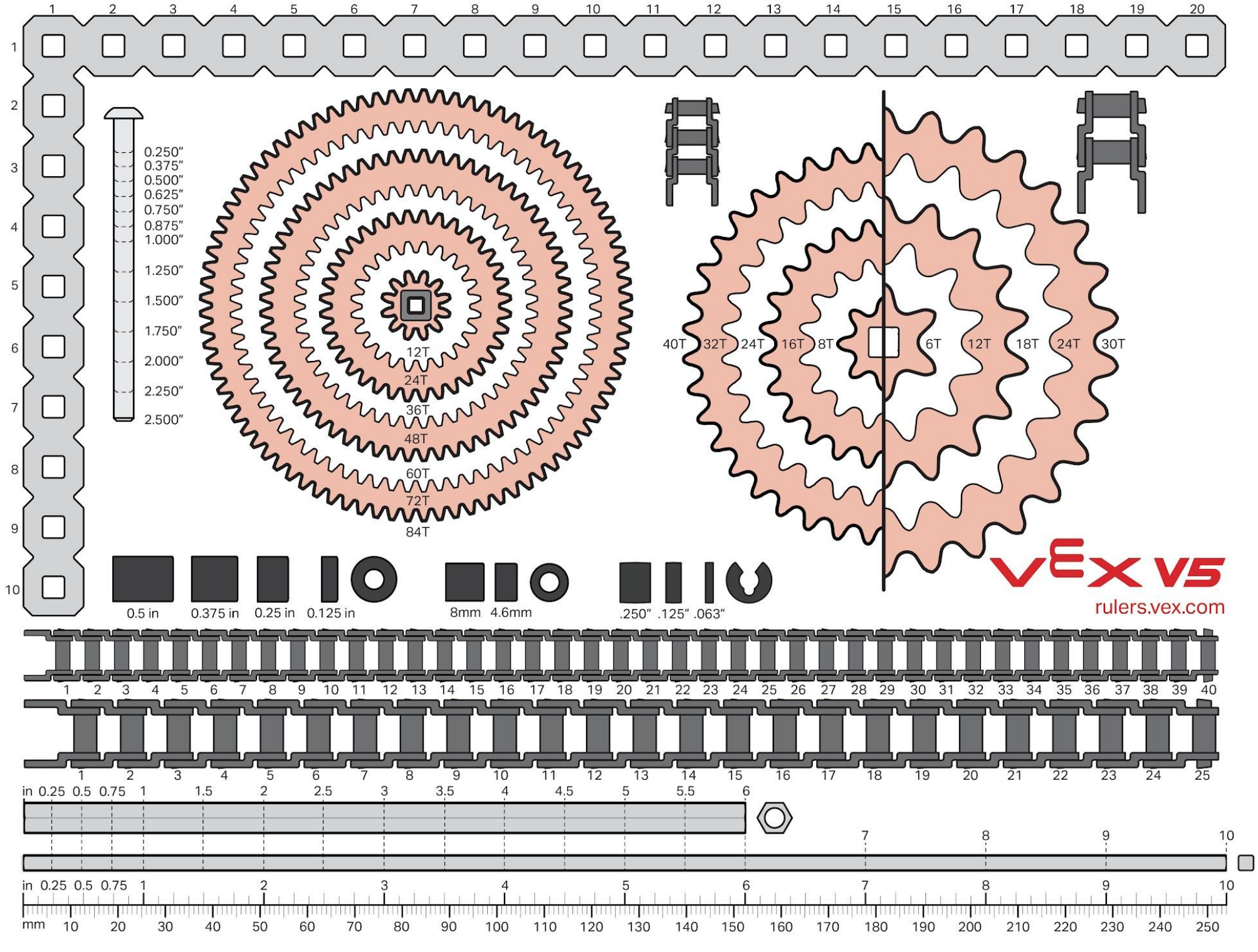
We also have a goal to work with our alliance to ensure they have a functioning robot, and are able to score. Talking strategy beforehand is very important!

Skills will be a large part of this competition for us, as we see it as the only way to make it to states. We are not optimistic about earning any awards, but believe if we do well enough in skills again we may make it to states. Since there is not enough time at home, we may have to program our skills route while at the competition. That will not be fun.

This is our first competition, but we're just here to have some fun!

Let's do this!

Dennis C.
Katie C.
Rana H.
Oliver L.
Rowan W.


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10
0.250″
0.375″
0.500″
0.625″
0.750″
0.875″
1.000″
1.250″
1.500″
1.750″
2.000″
2.250″
2.500″
12T
24T
36T
48T
60T
72T
84T
40T 32T 24T 16T 8T 6T 12T 18T 24T 30T
0.5 in
0.375 in
0.25 in
0.125 in
8mm
4.6mm
.250″
.125″
.063″
VEX V5
rulers.vex.com
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
in 0.25 0.5 0.75 1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6
7 8 9 10
in 0.25 0.5 0.75 1 2 3 4 5 6 7 8 9 10
mm 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250

V5RC Team 593C
Oakdale High School
2025-2026, Push Back

Rowan W.
Katie C.
Dennis C.

Rana H.
Oliver L.

Oakdale 593C Engineering Notebook Template

- Welcome to the Oakdale 593C Engineering Notebook!
- This template was designed to be easy to use for all members of 593C, and to be easily readable.
- The goal of this notebook template was to create an easy way of logging data, yet looks more unique than the notebooks provided by VEX.
- Another goal was to prevent walls of text that you need to pull a magnifying glass out to view, much like previous years notebooks.
- Design philosophy is to integrate colors and images to break up walls of text to make it easier to read
- Writer names and entry dates will be visible on every page to demonstrate chronological documentation!

Name: **Date**: