



Longmont, Colorado

Digital Engineering Notebook

2024-2025 High Stakes

Date updated: 05/08/25

Published Resources:

During the season we created many resources to showcase what we have learned in VEX robotics. We have created the following videos at the time of this notebook's publication to showcase what we have learned.

[Notebook Link](#)

[Notebook Formatting \(Make a copy!\)](#)

[Robot Explanation](#)

Software:

[VEX Programming Youtube Playlist](#)

[2654E Echo GitHub Programming Repository](#)

[Path Planner](#)

[Look at alexDickhans on GitHub for other repositories](#)

Hardware:

[VEX Hardware Youtube Playlist](#)

[Robot Reveal Video Playlist](#)

[Drive Gearing & Gear Ratios: Carl Richter/2654E](#)

Skills:

[Skills Runs Playlist](#)

Contact Us!

- [Alex Dickhans](#) (Programming/Team Organization/Leadership)
 - Discord: alex_2654e
 - adickhans@gmail.com
- [Carl Richter](#) (Design/Build)
 - Discord: carl_2654e
 - carlrichter32@gmail.com
- Matt Dickhans (Driver)
 - Discord: matt_2654e
 - mdickhans@gmail.com

License

[2654E Engineering Notebook](#) © 2024-2025 by [Alex Dickhans](#), [Carl Richter](#), Matt Dickhans is licensed under
[Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International](#) 

Date 02/20/2025

Event Name Colorado States & Worlds

Innovate Award Submission Information Form

Instructions for team: Please fill out all information, printing clearly. This form should be included immediately after the Engineering Notebook's cover page. In the case of physical notebooks, this form can be printed out and placed in the notebook. For digital notebooks, this form can be scanned in and included. 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: 2654E

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

Unique tier 3 hanging mechanism with an effective and efficient integration into other robot systems. Advanced macros and software control to enhance the performance of our high hang mechanism.

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

[Brainstorm/Background Research: High Hang](#) (Pg. 243-244)
[Identify/Design: Mechanism Limitations and Constraints](#) (Pg. 245)
[Brainstorming/Design: Hang Prototyping](#) (Pg. 246)
[Design: Initial Hang CAD \(C.3.1\)](#) (Pg. 247)
[Evaluate/Brainstorming: Hang Concept Reconsideration & Changes](#) (Pg. 257)
[Design: CAD Day 2 \(C.3.2\)](#) (Pg. 263-267)
[Design: CAD Day 5 \(C.3.2\)](#) (Pg. 273-277)
[Build: Hang String Selection](#) (Pg. 287)
[Build: Day 6 \(R.2.1.6\)](#) (Pg. 290-291)
[Testing/Identify: Hang \(R.2.1.6\)](#) (Pg. 292)
[Reevaluate: Hang String Selection \(R.2.1.6\)](#) (Pg. 293)
[Design: Hang Macro](#) (Pg. 300-302)
[Build: Hang Improvements \(R.2.1.7\)/\(C.3.2\)](#) (Pg. 303)
[Testing: Hang Macro](#) (Pg. 304)
[Build: Hang String Release \(R.2.1.11\)](#) (Pg. 307)
[Build: Hang Changes \(R.2.1.14\)](#) (Pg. 312)
[Testing: Hang Macro Revised](#) (Pg. 313-315)
[Evaluate: Robot 2 Subsystem Analysis \(R.2.1.19\)](#) (Pg. 356-359)
[Brainstorm/Identify: Robot 3 Requirements](#) (Pg. 360)
[Background Research: Subsystem Options](#) (Pg. 361-364)
[Design: Ideal Robot vs. Physical Limitations](#) (Pg. 365)
[Design: CAD Day 2 \(C.4.1\)](#) (Pg. 368)
[Design: Robot 3 Subsystems](#) (Pg. 371)
[Design: CAD Day 3 \(C.4.1\)](#) (Pg. 376)

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

In order to increase our skills potential, we needed a way to score more points as we had nearly maxed out what was possible with just rings. Developing a T3 hang allowed us to achieve a world record skills score in mid January that is still #1 in the world. Our implementation also allows us to maintain a competitive, lightweight match robot with effective wall stake scoring, a fast drivetrain, and the opportunity to score an additional 12 points that can significantly alter the matches outcome. Two full iterations of our T3 robot have led to perhaps the fastest and most viable implementation of a high hang this season with highly integrated and thought out robot subsystems, structures, automation, and strategies.

Page Intentionally Left Blank

Table of Contents

Notebooking Introduction	1
05/01/24 Team Biography	2
05/01/24 Team Goals	3
05/02/24 Notebook Formatting	4
05/02/24 Notebooking Accountability	5
05/03/24 Design Process	6
05/03/24 Design Matrix/Pros Cons Lists	7
05/03/24 Time Management	9
Game Analysis	10
05/04/24 Identify: VEX Competition Overview	11
05/04/24 Identify: High Stakes Overview	12
05/05/24 Identify: Point Breakdown	13
05/05/24 Identify: Game Manual Analysis	14
05/07/24 Identify: Scoring Rules	15
05/07/24 Identify: Safety Rules	20
05/07/24 Identify: General Game Rules	21
05/08/24 Identify: Specific Game Rules	27
05/09/24 Identify: Robot Rules	31
05/10/24 Identify: Robot Skills Rules	37
05/10/24 Identify: Programming Objectives and Challenges	40
Background Research	41
05/11/24 Background Research: Round Up Analysis	42
05/11/24 Background Research: Turning Point Analysis	43
05/11/24 Background Research: Important Characteristics of the Chassis	44
05/11/24 Background Research: Chassis Types	46
05/11/24 Background Research: Prior Chassis Analysis	50
05/11/24 Background Research: Chassis Programming	53
05/12/24 Background Research: Chassis Decision Matrix	54
05/13/24 Background Research: Ring Manipulation	55
05/14/24 Background Research: Lift Mechanisms	57
05/14/24 Background Research: Motors & Pneumatics	58
05/14/24 Background Research: Feedback Loops	61
05/15/24 Background Research: Feedforward Loops	65
05/15/24 Background Research: Wheel Options	67
05/17/24 Background Research: Gears vs. Chain	70
05/22/24 Background Research: Programming Tools for VEX	71
05/24/24 Background Research: Other Programming Tools	73

<u>05/25/24</u>	Background Research: Programming Tools Setup	76
<u>05/26/24</u>	Background Research: CAD Software Options	77
<u>05/28/24</u>	Background Research: CAD Selection & Setup	78
<u>06/01/24</u>	Background Research: Robot Localization: Ideology	80
Robot 1		84
<u>06/02/24</u>	Time Management: Summer Timeline	85
<u>06/03/24</u>	Brainstorming: Initial Strategy/Potential Designs	86
<u>06/04/24</u>	Identify: Robot 1 Design Considerations	87
<u>06/11/24</u>	Brainstorming: Concept 1 Drawing	88
<u>06/12/24</u>	Brainstorming: Concept 2 Drawing	89
<u>06/12/24</u>	Brainstorming: Concept 3 Drawing	90
<u>06/14/24</u>	Select Solution: C.1.1 Concept Decision	91
<u>06/18/24</u>	Brainstorming: Lift System Calculations	93
<u>06/23/24</u>	Design: 2DOF Lift Inverse Kinematics	99
<u>06/23/24</u>	Design: Drive Base Specs	102
<u>06/27/24</u>	Identify: June 25th Game Manual Update Analysis	104
<u>07/02/24</u>	Brainstorm: Programming Software Architecture	104
<u>07/12/24</u>	Design: CAD Day 1&2 (C.1.1/C.1.2)	108
<u>07/17/24</u>	Design: CAD Day 3 (C.1.3)	110
<u>07/17/24</u>	Design: CAD Day 4 (C.1.3)	111
<u>07/18/24</u>	Evaluate: Initial Concept Reconsideration	112
<u>08/01/24</u>	Time Management: Revised Timeline	113
<u>08/05/24</u>	Analysis: Minnesota Signature Event	114
<u>08/08/24</u>	Brainstorming: C.2.1 Concept One	118
<u>08/10/24</u>	Brainstorming: C.2.1 Concept Two	118
<u>08/14/24</u>	Brainstorming: C.2.1 Concept Three	120
<u>08/15/24</u>	Select Solution: C.2.1 Concept Decision	121
<u>08/15/24</u>	Identify & Design: C.2.1 Requirements	122
<u>08/15/24</u>	Background Research: Robot Localization: Literature Review	126
<u>08/16/24</u>	Design: Robot Localization	131
<u>08/17/24</u>	Design: Sensor Models	136
<u>08/17/24</u>	Design: CAD Day 1 (C.2.1)	140
<u>08/18/24</u>	Design: CAD Day 2 (C.2.1)	141
<u>08/19/24</u>	Design: CAD Day 3 (C.2.1)	142
<u>08/20/24</u>	Design: CAD Day 4 (C.2.1)	144
<u>08/21/24</u>	Design: CAD Day 5 (C.2.1)	145
<u>08/22/24</u>	Build: Day 1 (R.1.0.0)	149
<u>08/23/24</u>	Build: Day 2 (R.1.0.0)	150

08/24/24	Build: Day 3 (R.1.0.1)	151
08/25/24	Build: Day 4 (R.1.1.0)	152
08/26/24	Build: Day 5 (R.1.1.0)	153
08/27/24	Testing/Identify: Problems (R.1.1.0)	155
08/27/24	Build: Minor Improvements (R.1.1.2)	156
08/27/24	Testing: State Machine/Programming Environment	156
08/29/24	Identify: Wall Stake Optimization (R.1.1.2)	158
08/29/24	Select Solution/Design/BUILD: Wall Stake Optimization (continued) (R.1.2.0)/(C.2.2)	159
08/31/24	Build/Testing: Improving Subsystems (R.1.2.6)	161
09/01/24	Design/Testing: Macro Design	163
09/04/24	Background Research: Motion Control	167
09/05/24	Build: Intake Upgrades (R.1.2.7)	170
09/05/24	Testing: Scoring Consistency (R.1.2.7)	171
09/06/24	Testing: Localization	172
09/06/24	Identify: Sep. 3 Game Manual Update Analysis	175
09/08/24	Design: Autonomous Path Planning	177
09/08/24	Design: Skills Pathing	182
09/09/24	Design: Vision Guided Motion Control	184
09/10/24	Design: 2D Spline Motion Profiling	187
09/12/24	Design: Ramsete Path Following	191
09/16/24	Design: Intake Optimizations (R.1.2.9)	193
09/17/24	Time Management: 1st Tournament Timeline	194
09/18/24	Design: AWP Autonomous Pathing	195
09/19/24	Testing: Skills Path	196
09/21/24	Time Management: Comp Dates & Priorities	199
09/21/24	Strategy: Skills Repathing	200
09/22/24	Time Management: Season Timeline	201
09/22/24	Strategy: Competition Analysis	202
09/23/24	Testing: Skills Repathing	204
09/28/24	Testing: Original Skills Path	205
09/28/24	Testing: AWP Path	206
10/08/24	Design: Hang & Front Arm Updates (R.1.2.11)	207
10/18/24	Analysis: Butter Nexus League (10/15/24) Skills and Matches	208
10/21/24	Design: Color Sorting	212
10/21/24	Design: Alliance Color Determination	213
10/21/24	Design: Eliminations Autonomous Routine	214
10/23/24	Strategy: Match Play	215
10/23/24	Testing: Eliminations Autonomous Routine	217

10/23/24	Testing: Erie Pre-Competition Skills Testing	218
10/30/24	Testing/Evaluate: Erie Competition Analysis	219
11/04/24	Testing/Evaluate: CEC Competition Analysis	222
11/04/24	Time Management: Robot Rebuild	225
11/05/24	Identify/Define/Brainstorm/Design: Localization Testing Tool	226
11/06/24	Testing: Localization Performance/Tooling Testing	230
11/07/24	Design/Testing: Faster 2D Motion Profiling	232
11/08/24	Evaluate: Robot 1 Deconstruction (R.1.2.11)	241
Robot 2		242
09/30/24	Strategy: Skills Reanalysis	243
10/05/24	Brainstorm/Background Research: High Hang	244
10/07/24	Identify/Design: Mechanism Limitations and Constraints	246
10/18/24	Brainstorming/Design: Hang Prototyping	247
10/20/24	Design: Initial Hang CAD (C.3.1)	248
10/22/24	Brainstorming: Lift Gearing Calculations	249
11/01/24	Evaluate/Testing: Skills Time Distribution	250
11/02/24	Analysis/Brainstorming: R.1.2.11 Subsystems	251
11/02/24	Evaluate/Brainstorming: Hang Concept Reconsideration & Changes	258
11/03/24	Design/Planning: Design Methodology	259
11/03/24	Design: CAD Day 1 (C.3.2)	260
11/04/24	Design: CAD Day 2 (C.3.2)	264
11/05/24	Design: CAD Day 3 (C.3.2)	269
11/06/24	Design: CAD Day 4 (C.3.2)	271
11/07/24	Design: CAD Day 5 (C.3.2)	274
11/08/24	Background Research: System Identification	279
11/09/24	Build: Day 1 (R.2.1.1)	282
11/10/24	Build: Day 2 (R.2.1.2)	283
11/11/24	Build: Day 3 (R.2.1.2)	285
11/12/24	Build: Day 4 (R.2.1.2) / CAD (C.3.2)	286
11/12/24	Build: Hang String Selection	288
11/13/24	Build: Day 5 (R.2.1.2)	289
11/13/24	Analysis/Design: Skills Pathing	290
11/18/24	Build: Day 6 (R.2.1.6)	291
11/19/24	Testing/Identify: Hang (R.2.1.6)	293
11/19/24	Reevaluate: Hang String Selection (R.2.1.6)	294
11/19/24	Testing: Intake, Drivetrain, Wall Stakes (R.2.1.6)	295
11/20/24	Design: New Negative Eliminations Autonomous	296
11/20/24	Design: Drivetrain System Identification	297

11/22/24	Design: Hang Macro	301
11/22/24	Build: Hang Improvements (R.2.1.7)/(C.3.2)	304
11/22/24	Testing: Hang Macro	305
11/23/24	Build/Design: Wall Stake Mechanism Improvements (R.2.1.10)/(C.3.2)	306
11/24/24	Build: Hang String Release (R.2.1.11)	308
12/01/24	Testing: Drivetrain System Identification	309
12/02/24	Testing: New Negative Eliminations Auton	311
12/04/24	Analysis: Butter Nexus League Finals (12/03/24)	312
12/05/24	Build: Hang Changes (R.2.1.14)	313
12/09/24	Testing: Hang Macro Revised	314
12/09/24	Testing: RAMSETE/Feedforward Constant Improvement/System Identification	317
12/11/24	Build: Bottom Intake Modifications (R.2.1.15)	319
12/13/24	Testing: Skills Repath Testing (R.2.1.15)	320
12/17/24	Analysis: Kent Denver Matches (12/14/24)	322
12/17/24	Analysis: Kent Denver Skills (12/14/24)	323
12/18/24	Time Management: Kalahari Timeline	325
12/27/24	Identify/Design: Wireless Debugger Tool	326
01/05/25	Analysis: Sugar Rush Signature Event	331
01/07/25	Build: Hardware Improvements (R.2.1.17)	332
01/09/25	Testing: Autonomous Error Analysis	333
01/10/25	Testing: Skills Tuning Day 1 (R.2.1.17)	336
01/12/25	Strategy: Kalahari Autonomous Planning	337
01/14/25	Build & Test: Passive Ring Mechanism (R.2.1.18)	341
01/20/25	Testing: Kalahari Skills Tuning	342
01/22/25	Testing: Kalahari Autonomous Tuning	343
01/28/25	Analysis: Kalahari Matches (01/24-25/25)	344
01/29/25	Analysis: Kalahari Skills (01/24-25/25)	345
02/04/25	Build: Intake Optimizations (R.2.1.19)	347
02/04/25	Design: Skills Path End Repath (R.2.1.19)	348
02/11/25	Analysis: Silver Creek Matches (02/08/25)	349
02/11/25	Analysis: Silver Creek Skills (02/08/25)	350
02/11/25	Analysis: Pikes Peak Matches (02/09-10/25)	352
02/11/25	Analysis: Pikes Peak Skills (02/09-10/25)	353
Robot 3		354
02/13/24	Time Management: Robot Rebuild	355
02/13/25	Evaluate: Robot 2 Subsystem Analysis (R.2.1.19)	356
02/13/25	Brainstorm/Identify: Robot 3 Requirements	360
02/14/25	Background Research: Subsystem Options	361

<u>02/15/25</u>	<u>Design: Ideal Robot vs. Physical Limitations</u>	<u>365</u>
<u>02/16/25</u>	<u>Design: CAD Day 1 (C.4.1)</u>	<u>366</u>
<u>02/17/25</u>	<u>Design: CAD Day 2 (C.4.1)</u>	<u>368</u>
<u>02/17/25</u>	<u>Design: Robot 3 Subsystems</u>	<u>371</u>
<u>02/18/25</u>	<u>Design: CAD Day 3 (C.4.1)</u>	<u>373</u>
<u>02/19/25</u>	<u>Design: CAD Day 4 (C.4.1)</u>	<u>376</u>
<u>02/20/25</u>	<u>Strategy: Autonomous Reevaluation</u>	<u>379</u>
<u>Appendix A:</u>	<u>Community Contributions</u>	<u>388</u>
	<u>Section Goals</u>	<u>388</u>
	<u>Mentoring and Leadership—Longmont High School Robotics</u>	<u>389</u>
	<u>Public Programming Contributions</u>	<u>391</u>
<u>Appendix B:</u>	<u>External Sources</u>	<u>392</u>
	<u>Links</u>	<u>393</u>

NOTEBOOKING INTRODUCTION

05/01/24 Team Biography

Designed by: Matt	Witnessed by: Carl	Witnessed on: 05/01/2024
-------------------	--------------------	--------------------------

Hello! We are team 2654E Echo from Longmont High School in Longmont, Colorado. As a team we have many years of robotics experience. This season we hope to not only compete well, but also learn a lot from the VEX robotics program and our competitors, and enjoy the season as much as we can. Beyond just us, we would also like to help the competitive robotics community in our area.

<p>Matt Dickhans</p> <p>Age: 16</p> <p>Grade: Junior</p> <p>Experience: 9 Years</p> <p>Strengths: Driving, strategy, and building.</p> <p>Primary Role: Driver and Builder</p>	<p>Carl Richter</p> <p>Age: 17</p> <p>Grade: Senior</p> <p>Experience: 6 Years</p> <p>Strengths: CAD: (Fusion 360 & SOLIDWORKS), project management, and developing unique solutions.</p> <p>Primary Role: Designer and Builder</p> <p>Website: carlrichter.co</p>
<p>Alex Dickhans</p> <p>Age: 17</p> <p>Grade: Senior</p> <p>Experience: 10 Years</p> <p>Strengths: Programming (C++, Rust, Dart, GUI/Robot), strategy, and time management.</p> <p>Primary Role: Programmer</p> <p>Website: alex.dickhans.net</p>	<p>Mutual Roles</p> <p>Drive team, notebooking, strategy, overall robot concepts, time management, and scouting.</p>

05/01/24 Team Goals

Designed by: Matt	Witnessed by: Alex	Witnessed on: 05/01/2024
-------------------	--------------------	--------------------------

Goal: Establish long term goals and set out milestones to meet throughout the season

We are setting out goals this season to be able to look back on throughout the season to maintain a level of standard that we want across all aspects of play. We will be splitting these goals into 5 categories: Design, Build, Programming, Match Play and Performance.

Notebooking

- Create documentation that would make replication our entire design process throughout the season possible
- Document all important details on time
- Maintain notebooking accountability
- Release this notebook as a universal technical document for VEX

Design Process

- Design a well rounded robot that can score points in all the different ways
- Design the robot in a reasonable time frame
- Include any important details for build in the CAD
- Thoroughly design robots using a CAD program
- Use effective time management for each new robot/ project

Programming

- Maintain easy and quick to use codebase
- Keep programming skills at the top of the rankings
- Achieve high consistency (>90%) for autonomous and programming skills

Match Play

- Practice driving and maneuvers
- Have all team members ready to make sound calls and quick decisions
- Consistently maintain strong overall strategic decisions and adapt throughout the season

Performance

- Win at least 80% of our matches
- Keep ourselves in the top 10 skills teams
- Get to the Dome at worlds

05/02/24 Notebook Formatting

Designed by: Carl

Witnessed by: Alex

Witnessed on: 05/02/2024

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

General Page Set Up:

- Entry title: The title of each page will be in the format (Design Process Stage: Description of Task).
 - Appears when anything in the entry bar changes or the topic being discussed changes.
 - The title always appears at the top of the page; any space left over on the previous page will not be utilized.
- Entry bar: This bar always appears after the title and includes the date of the entry and the contributors to the task.
 - “Designed by” means the person making the entry; they will also be working on the specified task
 - “Witnessed by” means anyone who reviews the entry in the notebook. This should always happen on the day that the entry is written.
- Goal: A short statement about goals for a page/meeting.
- Content: Documentation of what was accomplished during a meeting.

Documentation Summary

Will be Documented	Will NOT be Documented
<ul style="list-style-type: none"> ● Significant changes in the design or programming of the robot ● Strategic discussion throughout the season, especially after tournaments and game manual updates ● Testing and reflections about robot components ● Time management systems such calendars and gantt charts 	<ul style="list-style-type: none"> ● Minor changes in code, build, etc... ● Cosmetic robot decorations or other elements that don't relate to the design process ● The bulk of our programing as we anticipate several thousand lines of code (140+ pages) <ul style="list-style-type: none"> ○ However, all steps necessary to make code (language- and platform-agnostic) will be documented

We noticed that using colors to represent different stages in the design process could be confusing due to each team's unique style. To ensure clarity for everyone, we will rely on page titles to identify each stage of the design process. Colors will only be used as side bars for each page aligning with specific sections.

05/02/24 Notebooking Accountability

Designed by: Alex	Witnessed by: Carl	Witnessed on: 05/02/24
-------------------	--------------------	------------------------

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

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

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

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

Accountability notice: Alex Dickhans completed the notebooking late (goal 2) on 5/16/2024

For our goals, we will shorten them for the purposes of staying concise

1. Wrong person documented
2. Notebooking completed late
3. Not witnessed in time

Conclusion:

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

05/03/24 Design Process

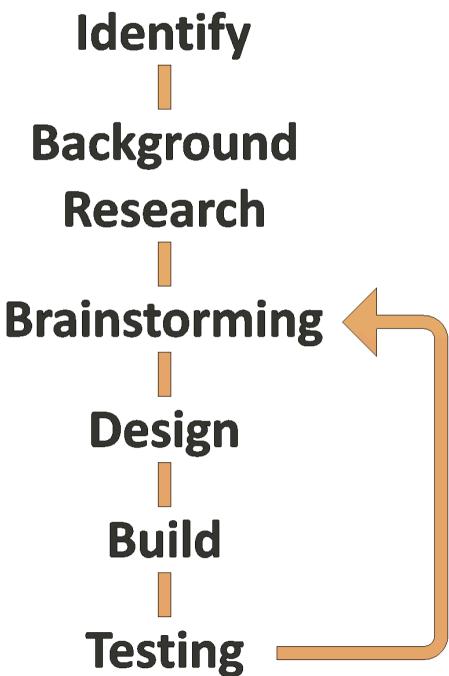
Designed by: Carl

Witnessed by: Alex

Witnessed on: 05/03/24

Goal: Describe the design process and how we have chosen to use it.

- Identify: Describe and clearly identify the challenge or problem that needs to be solved; we can't solve a problem if we don't know what it is.
- Background research: Gather relevant information about other teams, competitions, and from previous games. This will help us ensure that our robots are competitive against other designs, and we can take inspiration to help improve our robot.
- Brainstorming: Brainstorm 3 solutions that meet our requirements and compare each solution using tables and decision matrices.
- Design: Utilize CAD to create the selected solution. Designing before building helps reduce unanticipated complications in the construction process.
- Build: Construct the robot/subsystem that was CADed.
- Testing/Evaluation: Rigorous testing of a system to identify problems and oversights. We need to identify and resolve as many problems as possible in order to have a reliable and competitive robot.



More often than not, the original design does not work first try. If this happens, we will go back to an earlier phase in the design process. Small problems may simply mean going back to the design or build stage, while more significant problems may necessitate revisiting the brainstorming stage.

05/03/24 Design Matrix/Pros Cons Lists

Designed by: Alex	Witnessed by: Carl	Witnessed on: 05/03/24
-------------------	--------------------	------------------------

Goal: Describe the importance of effectively weighing different design options using Design Matrices and Pros Cons lists allow.

Effectively weighing different design options is crucial in any design process, as throughout the season we need to properly document how we make decisions. This will help us reevaluate decisions we make later in the season along with help us think through design decisions thoroughly. Two powerful tools for this purpose are Design Matrices and Pros and Cons lists.

Design Matrices:

Definition:

A Design Matrix is a structured tool that allows for the comparison of different design options based on multiple weighted criteria. Our design matrix has columns for each design decisions with rows for the criteria:

Criteria:	Weight	Choice		Choice		Choice	
		Score	Weighted Score	Score	Weighted Score	Score	Weighted Score
Criteria	1	2	2	3	3	6	6
Criteria	2	2	4	3	6	6	12
Criteria	3	2	6	3	9	6	18
Criteria	17	2	34	3	51	6	102
Criteria	18	2	36	3	54	6	108
Total Score:			82		123		246

Importance:

- Comprehensive Comparison
 - Side-by-side comparison of various design options
 - Allows effective weighing of many options against many criteria
- Objective Evaluation
 - Minimizes bias through quantifiable data
 - Ensures that decisions are based on quantifiable data
- Identifying Trade-offs
 - They help in identifying trade-offs between different design options. For example, one design might be more lighter but less durable, while another might be highly durable but heavier. A Design Matrix makes these trade-offs explicit, aiding in balanced decision-making.

Pros and Cons Lists:

Definition:

A Pros and Cons list is a straightforward method of listing the positive and negative aspects of each design option. It is less structured than a Design Matrix but also can be valuable in certain scenarios.

Importance:

- Simplified Decision-Making
 - Pros and Cons lists simplify the decision-making process by breaking down each design option into its advantages and disadvantages.
 - This straightforward approach makes it easier to grasp the key points quickly.
- Flexibility
 - Highly flexible and can be used in various contexts without the need for detailed criteria and quantitative data.
 - This makes them a practical tool for quick, simple analysis of different options.

Example Design Option	
Description	Describe the design decision
Pros	<ul style="list-style-type: none"> ● Big pro 1 ● Big pro 2 ● Pro 1 ● Pro 2 ● <i>Small pro 1</i> ● <i>Small pro 2</i>
Cons	<ul style="list-style-type: none"> ● Big con 1 ● Big con 2 ● Con 1 ● Con 2 ● <i>Small con 1</i> ● <i>Small con 2</i>

Conclusion:

- Both Pro/cons lists and Design Matrices play a vital role in design process
- Design Matrices
 - Structured and objective
 - Many different options
 - Very detailed
- Pros/Cons lists
 - Flexible
 - Simple
 - Very effective

Both Design Matrices and Pros and Cons lists play a vital role in our design process. Design Matrices offer a structured and objective way to compare different options, making them ideal for detailed evaluations with lots of criteria. On the other hand, Pros and Cons lists are a simple, flexible, and effective method. Leveraging these tools, we can make informed decisions that balance various factors and lead to the most suitable design.

05/03/24 Time Management

Designed by: Matt, Carl

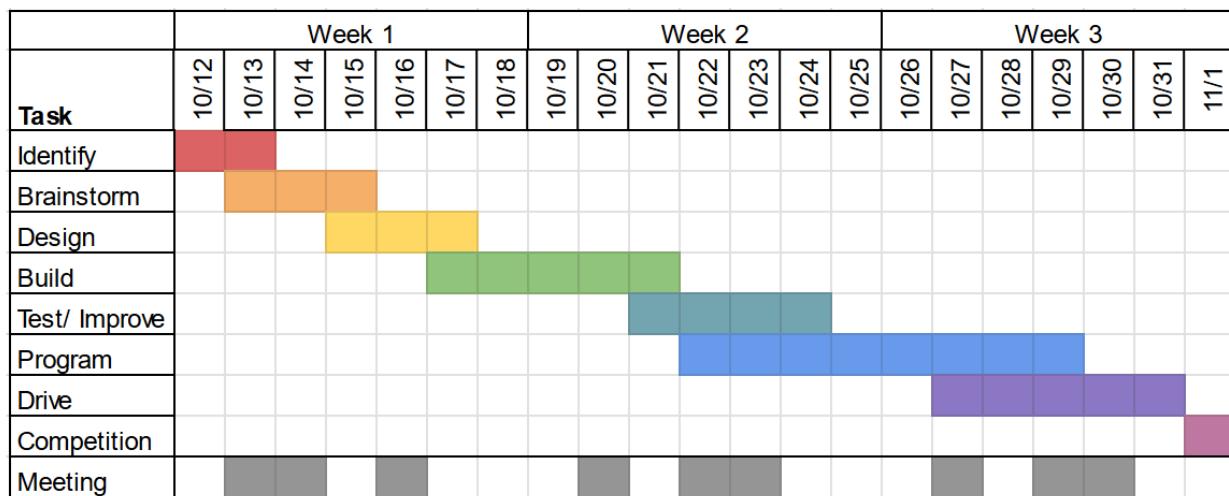
Witnessed by: Alex

Witnessed on: 05/03/24

Goal: Describe the importance of effective time management; lay out a system that can be used to manage time throughout the season.

In order to ensure that we are adequately prepared in all ways before each tournament, we will use Gantt charts to help organize our time with tasks set out before tournaments. At this point we do not know when most of our tournaments will be or how our robots will perform, meaning planning for the entire season would have limited benefit.

As opposed to a traditional calendar, Gantt charts such as the one below clearly show task progression and can be customized by length easily. Another benefit of Gantt charts is that colors can be used to show additional details about each task such as who is responsible for completing it.



GAME ANALYSIS

05/04/24 Identify: VEX Competition Overview

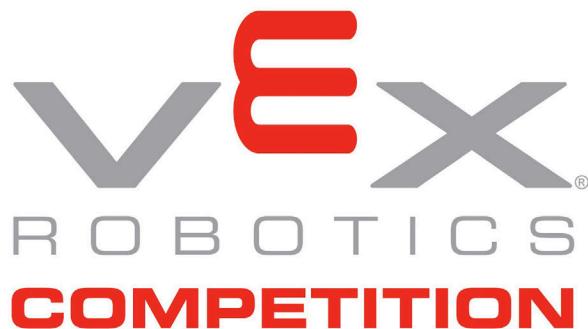
Designed by: Carl

Witnessed by: Alex

Witnessed on: 05/04/24

Goal: Provide a brief explanation of what VEX is and describe the competition as a whole.

The VEX Robotics Competition is a competitive robotics challenge, with participants ranging from elementary schoolers to college students. This competition helps prepare students for the future with a unique skill set, including teamwork, problem-solving skills, and time/resource management. VEX provides a unique experience that helps prepare students for our swiftly changing world in a way that might not be possible through traditional methods.



VEX 123 Grades Pre-K+	VEX GO Grades 3+	VEX IQ Grades 6+	VEX EXP Grades 9+	VEX V5 Grades 9+
VEX 123 is an interactive, programmable robot that takes Computer Science and Computational Thinking off of the screen and brings them into the hands of elementary students.	VEX GO is an affordable STEM construction system that taps into children's natural inquisitiveness. VEX GO utilizes the VEX IQ plastic construction system and adapts it for elementary students.	VEX IQ is a snap-together robotics system designed from the ground up to provide novice users the chance to find success quickly, while still being able to constantly challenge more advanced users.	The VEX EXP Ecosystem promotes high-quality STEM education that is essential, relevant, and continual. Educators also get professional development, curriculum and support.	The VEX V5 system includes versatile elements that take the frustration out of engineering for novice users, while still providing experienced users with endless design possibilities and build challenges.
Buy 123	Buy GO	Buy IQ	Buy EXP	Buy V5
Learn More	Learn More	Learn More	Learn More	Learn More

(Image source: vexrobotics.com)

05/04/24

Identify: High Stakes Overview

Designed by: Alex

Witnessed by: Carl

Witnessed on: 05/04/24

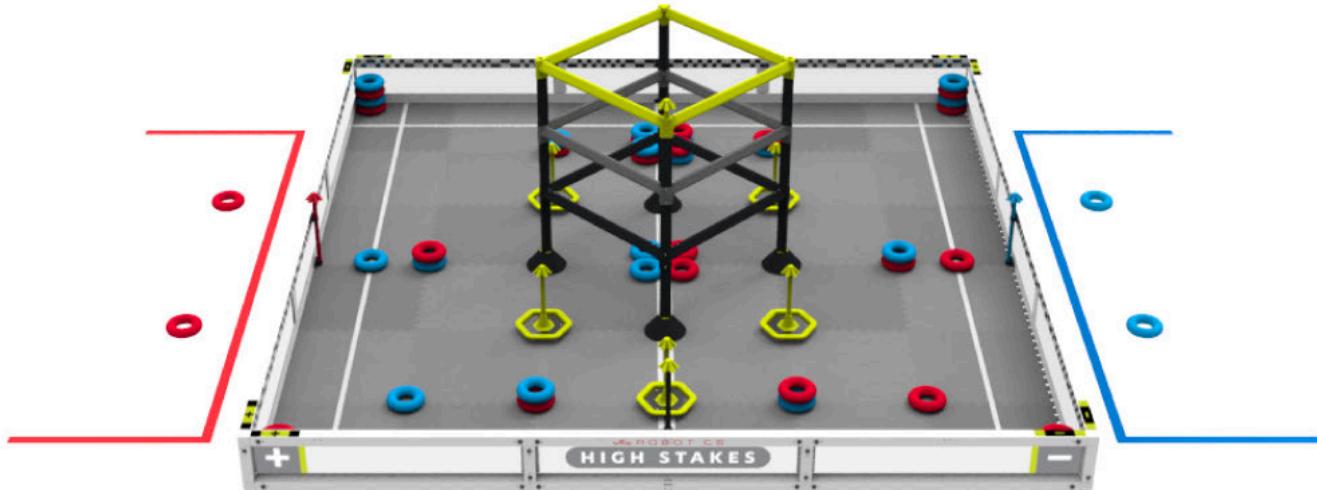
Goal: Understand at a high level how the High Stakes game is played.

VEX Tournament Overview

In the VEX robotics program they release a new game each year. For the 2024-2025 season, the competition is named High Stakes. This game has 2 main components, skills and tournament. The tournament portion of the competition is made up of matches. In these matches teams are trying to maximize the amount of win points they can get to increase their ranking. Matches have a 15 second autonomous period, where robots move on their own and try to score as many points as possible along with the autonomous win point; one of the three win points that can be attained in a match. Then the driver control period starts where drivers control their robots to get more points than the other alliance. If a team beats the other alliance or gets more points, they are awarded with 2 win points, if they tie both are awarded with 1 win points, and for the losing alliance, each team is awarded 0 win points. In skills, robots compete by themselves to maximize the points they can score. At each event, the team is ranked on their combined programming and driver skills scores they can achieve in each 1 minute long robot skills match.

High Stakes Overview

This year VEX competition High Stakes is played on a 12 foot by 12 foot field with 5 mobile goal stakes, 2 neutral stakes, and 2 alliance stakes. To score on these stakes the game includes 48 blue or red colored rings, 24 for each alliance. This game also has the highest hanging structure ever implemented in a VEX robotics competition. This hanging structure is made up of multiple tiers that the robots will have to climb in between to get to the top rung and score the most points. Additionally, there are scoring modifier corners that change the value of the rings on the goals.



05/05/24 Identify: Point Breakdown

Designed by: Carl	Witnessed by: Matt	Witnessed on: 05/05/24
-------------------	--------------------	------------------------

Goal: Identify how points can be scored; discuss maximum point values and estimated efficiency of scoring in different ways.

Name	Point Value	Maximum Points	Estimated Points Per Second
Autonomous Bonus	6	6	0.4
Each Ring Scored on a Goal	1	24	0.5
Each Top Ring on a Goal	3	15	1.5
Each Ring Scored on a Stake	1	13	0.3
Each Top Ring on a Stake	3	9	1
Climb - Level 1	3	3	1
Climb - Level 2	6	6	1.5
Climb - Level 3	12	12	2

To create this table, all different tasks and point values were sourced directly from the game manual. The maximum points were found for each category using the built in score calculator for VRC Hub. To estimate points per second (efficiency of scoring) we leverage our experience from previous games to determine how long each task might take, and multiplied it by its point value. This year, based on our analysis, it will be very time effective to create a hang mechanism, because there are a lot of points that robots can complete relatively quickly. The downside is this action can only be completed once, and the mechanism to complete this task will likely take up a lot of space and add complexity to the robot. Doing this task also has no impact on the autonomous bonus, so it will be very beneficial to make a robot or mechanism that can also score rings on a variety of the stakes.

Note: Because this point analysis is based on our assumptions, it is NOT accurate, and is just meant to provide a base understanding for the values of each category.

Note: Due to corner scoring modifiers in SC6, rings on Mobile Goals may have different point values. Our analysis on this rule is below.

05/05/24

Identify: Game Manual Analysis

Designed by: Alex

Witnessed by: Carl

Witnessed on: 05/05/24

Goal: Read and understand the game manual in its entirety. Identify key limitations and applications of rules.

Rule Analysis Template

For each of the rules we will copy all necessary parts into the “Rules Definition” layed out below. After that we will give how we interpret how we think it will be ruled in the game, and then we will discuss the strategic and/or design impacts that the rule will have on the game. The contents of each box are laid our below:

Rule Definition:

Rule definition from the rule book. Sometimes we will omit red boxes or violation notes for conciseness.

Interpretation:

How we think the rule will be implemented at competitions and in inspection. Additionally, if the rule is especially self explanatory, we will not do our own interpretation of it.

Impact:

In the impact we will analyze the strategic impact, the design impact or both. We will analyze how this rule will affect both the strategic decisions that we and other teams might make, and the ways this will affect design decisions for teams. If the rules have no impact on design, there will be no impact, but we will review them.

All important definitions will be described and interpreted in the rules that it applies to.

05/07/24 Identify: Scoring Rules

Designed by: Alex, Matt	Witnessed by: Carl	Witnessed on: 05/07/24
-------------------------	--------------------	------------------------

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

- a. This 5 second delay is intended to be the only permitted “benefit of the doubt” for last-second scoring actions. If an object or *Robot* is still in motion and “too close to call” between two states at the 5-second mark, then the less advantageous of the two states should be awarded to the *Robot(s)* in question. For example:
 - i. A *Robot* which has *Climbed* on the *Ladder* but is slowly drooping down, and crosses a *Level* threshold right at 5 seconds, would be considered in the lower of the two *Levels*.
 - ii. A *Ring* which slowly slides out of a *Robot's* mechanism and lands on a *Stake* right at 5 seconds would not be considered *Scored*.
- b. At the end of the *Match*, the on-screen timer displayed by Tournament Manager will hold the current *Match* information and “0:00” for 5 seconds before moving to queue the next *Match*. This should be the primary 5-second visual cue used by *Teams* and *Head Referees*.
- c. This 5 second delay is only intended to be a “benefit of the doubt” grace period, not an extra 5 seconds of *Match* time. *Robots* which are designed to strategically exploit this grace period will receive a *Minor Violation*, and any post-*Match* movement will not be included in score calculation (i.e., the *Match* will be scored as it was at 0:00).

Interpretation: This rule is primarily there to ensure teams that have hanging mechanisms that swing, or that are slowly falling down still get some points for their hangs. Additionally this prevents mechanisms that are entirely passive from climbing the bars after time.

Impact: This rule will mean that robots have to design around a faster hang that can be completed right at the end of the match, instead of designing a mechanism slightly slower that might hang after the time expires.

<SC2> Scoring of the **Autonomous Bonus** is evaluated immediately after the **Autonomous Period** ends (i.e., once all **Scoring Objects**, **Field Elements**, and **Robots** on the **Field** come to rest).

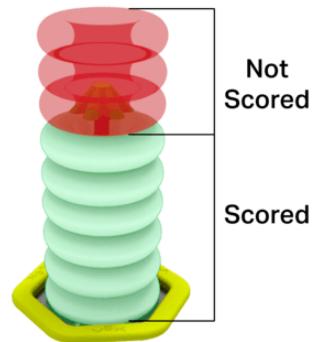
- a. **Climb** points and **Corner** modifiers are not included in the calculation of an **Alliance's** score for the purposes of determining the **Autonomous Bonus**.
- b. If the **Autonomous Period** ends in a tie, including a zero-to-zero tie, each **Alliance** will receive an **Autonomous Bonus** of three (3) points.
- c. Any rule **Violations**, Major or Minor, during the **Autonomous Period** will result in the **Autonomous Bonus** being awarded to the other **Alliance**. If both **Alliances** violate rules during the **Autonomous Period**, no **Autonomous Bonus** will be awarded.

Interpretation: The autonomous win is awarded to the team that gets more of their colored rings and scores them onto the goals.

Impact: The corner modifiers are uneven on the field, and difficult to strategically take advantage of in autonomous mode. Additionally, in autonomous it only makes sense for a team to score their color, which may not be what teams score in the match because of the corner modifiers. Additionally, this may be a time where teams take advantage of the alliance stakes, and the neutral stakes to leave the goals more available.

<SC3> A **Ring** is considered **Scored on a Stake** if it meets the following criteria:

- a. The **Ring** is not contacting a **Robot** from the same color **Alliance** as the **Ring**.
- b. The **Ring** is not contacting a gray foam tile.
- c. The **Ring** is “encircling” a **Stake**. In this context, “encircling” means that any part of the **Stake** is at least partially within the volume defined by the inner edges of the **Ring**.
- d. The **Stake** does not exceed its total permitted number of **Rings** (see definition of **Stake**). In the event of too many **Rings** on a **Stake**, the “highest” **Rings** will be removed.



Interpretation:

This rule generally means that refs will use common sense to determine if a ring is scored on a stake. Tipped over stakes might not have all their rings count.

Impact:

Because rings also have to be slightly pushed on a Tipping Point style intake that drops rings on will likely be ineffective.

<SC4> A *Ring* is considered a **Top Ring** if it meets the following criteria:

- The *Ring* is *Scored* on a *Stake* (i.e., meets all criteria in **<SC3>**).
- The *Ring* is the furthest *Scored Ring* from a given *Stake's* base (i.e., *Mobile Goal* base or *Field Perimeter* wall).
- There is no minimum number of *Rings* required; if only one *Ring* is *Scored* on a *Stake*, then it is still considered that *Stake's Top Ring*.

Note: A Ring that is considered a Top Ring does not also receive points for being Scored on a Stake; i.e., that Ring is worth 3 points, not a total of "3 + 1" points.

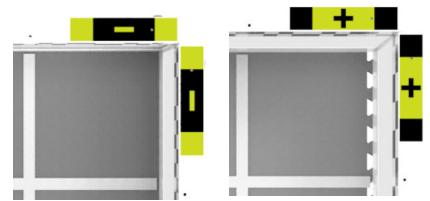
Note 2: If a Top Ring cannot be determined, but the two Rings in question are of the same color, then either of them may be considered the Top Ring. If the two Rings in question are of opposite colors, then that Stake will have no Top Rings.

Interpretation: The rings scored on the top of the post, again using common sense, are worth 3 points.

Impact: Maintaining control of the top rings, either by scoring or descoring, will be important capabilities to have on a robot.

<SC5> A *Mobile Goal* is considered **Placed in a Corner** if it meets the following criteria:

- The *Mobile Goal's* base is contacting the *Corner* (i.e., the *Floor* and/or white tape line).
- It is "upright." For the purposes of this definition, a *Mobile Goal* is considered "upright" if no contact is being made between its *Stake* (and/or any *Rings* on this *Stake*) and the *Floor* or *Field Perimeter*.
- Contact with a *Robot* is irrelevant, as long as all other criteria are met.



Interpretation: This rule means refs will determine if a Mobile Goal is placed in the corner zone and if there are 3 ways in which a mobile goal is determined if it is placed.

Impact: We must keep the mobile goals vertical while placing in the corner zones.

<SC6> A *Mobile Goal* that has been *Placed* will result in the following *Corner modifiers* to its *Scored Rings*:

- a. *Placed* in a *Positive Corner*
 - i. Values of all *Scored Rings* on the *Mobile Goal* will be doubled. *Scored Rings* will receive two (2) points, and *Scored Top Rings* will receive six (6) points.
- b. *Placed* in a *Negative Corner*
 - i. Values of all *Scored Rings* on the *Mobile Goal* will be set to zero points.
 - ii. For each *Ring*, an equivalent amount of points will be removed from that *Alliance's* other *Scored Rings*. *Scored Rings* will remove (1) point, and *Scored Top Rings* will remove three (3) points.
 - iii. This negator only applies to an *Alliance's* “*Ring* points.” Points received for *Climbing* and the *Autonomous Bonus* cannot be removed.

Example	Before Negative Corner	After Negative Corner	Notes
1	 Stake 1: +3, +1, +1	 Stake 1: +3, +1, +1	<i>Stake 2 was initially worth 5 points for the Blue Alliance, but is now worth negative 5 points after being moved into the Negative Corner.</i>
	Blue: +6 Points	Blue: +5 Points	Blue: +1 Point
2	 Stake 1: +3, +1	 Stake 1: +3, +1	<i>Even though the net total is -1, you cannot have negative total points.</i>
	Blue: +4 Points	Blue: +5 Points	Blue: -5 Points
			Blue: 0 Points

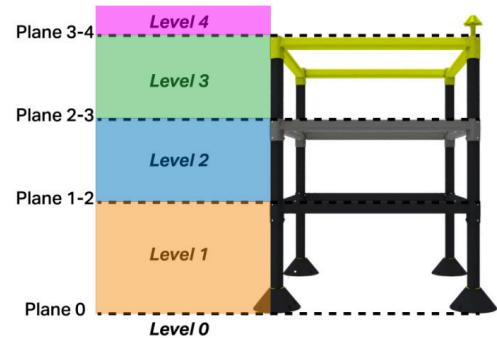
Note: The impact of *Corner modifiers* is subject to change in any of the major Game Manual updates (June 25, 2024; September 3, 2024; January 28, 2025; and/or April 2, 2025).

Interpretation: This means that rings on mobile goals that are considered “scored” in a corner zone will either double or subtract points depending on the corner it is in.

Impact: Make sure we can pick up mobile goals out of the corners and make sure we can easily score in the corners too.

<SC7> A *Robot* is considered to have *Climbed to a Level* if it meets the following criteria:

- a. The *Robot* is contacting the *Ladder*.
- b. The *Robot* is not contacting any other *Field Elements*, including the gray foam tiles.
- c. The *Robot* is not contacting any *Mobile Goals*.
- d. The *Robot's* lowest point is past that *Level's* minimum height from the gray foam tiles.
- e. Each *Level's* height corresponds to the top edge of a rung of the *Ladder*. For example, a *Level 1 Climb* represents a *Robot* whose lowest point is above the foam tiles, but not higher than the first rung of the *Ladder*.



Interpretation: This is the criteria for determining what counts as hanging.

Impact: Ensure that the robot does not contact the field tiles when hanging and that the lowest part of the robot is past the level's minimum height.

<SC8> An **Autonomous Win Point** is awarded to any *Alliance* that ends the *Autonomous Period* with the following tasks completed, and that has not broken any rules during the *Autonomous Period*:

- At least three (3) *Scored Rings*
- A minimum of two (2) *Stakes* with at least(1) *Ring Scored*
- Neither *Robot* contacting / breaking the plane of the *Starting Line*
- One (1) *Robot* contacting the *Ladder*

Interpretation: This is the requirements that you need to meet during the autonomous period to get awarded the AWP.

Impact: We need to ensure that we can meet all of these objectives with our robot and ensure that the autonomous code is consistent and fast to meet all of the requirements in time.

05/07/24 Identify: Safety Rules

Designed by: Alex	Witnessed by: Carl	Witnessed on: 05/07/24
-------------------	--------------------	------------------------

<S1> Be safe out there. If at any time the *Robot* operation or *Team* actions are deemed unsafe or have damaged a *Field Element*, *Scoring Object*, or the *Field*, the offending *Team* may receive a *Disablement* and/or *Disqualification* at the discretion of the *Head Referee*. The *Robot* will require re-inspection as described in rule <R3> before it may take the field again.

Interpretation: When interacting with your robot or other teams in the pits or other places in the event, safety always comes first.

Impact: Power tools in the pits or at our lab should be used with as much caution as possible. Also when designing robot mechanisms, a high priority should be placed on making them safe to use.

<S2> Students must be accompanied by an Adult. No *Student* may attend a VEX V5 Robotics Competition event without a responsible *Adult* supervising them. The *Adult* must obey all rules and be careful to not violate *Student*-centered policies, but must be present for the full duration of the event in the case of an emergency. *Violations* of this rule may result in removal from the event.

Interpretation: At events to maintain safety an adult mentor must always be present to ensure team's safety.

Impact: When we go to signature events and worlds, we must always travel with an adult mentor or coach so that we are safe.

<S3> Stay inside the field. If a *Robot* is completely out-of-bounds (outside the *Field*), it will receive a *Disablement* for the remainder of the *Match*.

Interpretation: This rule is intended to ensure robots are safe at the event. A robot that is out of the field in an inherent danger to other teams and people at the event.

Impact: Making sure that the robot stays inside the field at all times during a match will be a top priority not only to ensure competitive success, but make sure that other teams and people at the event are safe.

<S4> Wear safety glasses. All *Drive Team Members* must wear safety glasses or glasses with side shields while in the *Alliance Stations* during *Matches*. While in the pit area, it is highly recommended that all *Team* members wear safety glasses.

Interpretation: To ensure safety in the drivers box, teams must wear safety glasses at all times.

Impact: We will wear safety glasses at the field. To ensure that we are used to this slight change we will make sure that we bring high-quality safety glasses to all events.

05/07/24 Identify: General Game Rules

Designed by: Alex, Carl	Witnessed by: Matt	Witnessed on: 05/07/24
-------------------------	--------------------	------------------------

<G1> Treat everyone with respect. All *Teams* are expected to conduct themselves in a respectful and professional manner while competing in VEX V5 Robotics Competition events. If a *Team* or any of its members (*Students* or any *Adults* associated with the *Team*) are disrespectful or uncivil to event staff, volunteers, or fellow competitors, they may receive a *Disqualification* from a current or upcoming *Match*. *Team* conduct pertaining to **<G1>** may also impact a *Team's* eligibility for judged awards. Repeated or extreme violations of **<G1>** could result in a *Team* being *Disqualified* from an entire event, depending on the severity of the situation.

Interpretation: Ensure that all team and robot actions throughout the event are respectful and considerate. This rule applies to parents, so any unruly behavior from parents will punish the team.

Impact: We will make sure that we are as respectful as we can be at the event. This will be one of our top priorities at all the events that we go to.

<G2> V5RC is a student-centered program. *Adults* may assist *Students* in urgent situations, but *Adults* may never work on or code a *Robot* without *Students* on that *Team* being present and actively participating. *Students* must be prepared to demonstrate an active understanding of their *Robot's* construction and code to judges or event staff.

Interpretation: Adults may not help students directly with their robot

Impact: Adult involvement must be limited in our team and other teams. All important decisions, notebooking, robot construction, programming, and other parts of our team must be done by students. This rule will have little effect on our team because we are already student built, but will ensure that the competition is fair for everyone who competes

<G3> Use common sense. When reading and applying the various rules in this document, please remember that common sense always applies in the VEX V5 Robotics Competition.

Interpretation: Use common sense when interpreting the rules and don't take typos and small errors to an unreasonable extent.

Impact: When evaluating if a robot action or mechanism is legal or stretching the rules, make sure you apply common sense.

<G4> The Robot must represent the skill level of the Team. Each *Team* must include *Drive Team Members*, *Coder(s)*, *Designer(s)*, and *Builder(s)*. Many also include notebooker(s). No *Student* may fulfill any of these roles for more than one VEX V5 Robotics Competition *Team* in a given competition season. *Students* may have more than one role on the *Team*, e.g., the *Designer* may also be the *Builder*, the *Coder* and a *Drive Team Member*.

- a. *Team* members may move from one *Team* to another for non-strategic reasons outside of the *Team's* control.
 - i. Examples of permissible moves may include, but are not limited to, illness, changing schools, conflicts within a *Team*, or combining/splitting *Teams*.
 - ii. Examples of strategic moves in *Violation* of this rule may include, but are not limited to, one *Coder* “switching” *Teams* in order to write the same program for multiple *Robots*, or one *Student* writing the Engineering Notebook for multiple *Teams*.
 - iii. If a *Student* leaves a *Team* to join another *Team*, <G4> still applies to the *Students* remaining on the previous *Team*. For example, if a *Coder* leaves a *Team*, then that *Team's* *Robot* must still represent the skill level of the *Team* without that *Coder*. One way to accomplish this would be to ensure that the *Coder* teaches or trains a “replacement” *Coder* in their absence.
- b. When a *Team* qualifies for a Championship event (e.g., States, Nationals, Worlds, etc.) the *Students* on the *Team* attending the Championship event are expected to be the same *Students* on the *Team* that was awarded the spot. *Students* can be added as support to the *Team*, but may not be added as *Drive Team Members* or *Coders* for the *Team*.
 - i. An exception is allowed if only one member of the *Team* is able to attend the event. The *Team* can make a single substitution of a *Drive Team Member* or *Coder* for the Championship event with another *Student*, even if that *Student* has competed on a different *Team*. This *Student* will now be on this new *Team* and may not substitute back to the original *Team* during the season.

Interpretation: The robot that a team creates must be something that the team is able to create.

Impact: This rule means that students from other programs, or coaches are not allowed to directly contribute to a team’s robot. This rule makes sure the game is fair to teams that are student built and put in the time.

<G5> Robots begin the Match in the starting volume. At the beginning of a *Match*, each *Robot* must be smaller than a volume of 18" (457.2 mm) long by 18" (457.2 mm) wide by 18" (457.2 mm) tall.

Interpretation: Robots have a size box restriction that they must be able to fit into at the start of the match

Impact: Robot’s must be designed so they are compact and can fit into the box at the beginning of the match.

<G6> Keep your Robots together. *Robots* may not intentionally detach parts during the *Match* or leave mechanisms on the *Field*.

*Note: Parts which become detached unintentionally are a *Minor Violation*, are no longer considered “part of a *Robot*,” and should be ignored for the purposes of any rules which involve *Robot* contact or location (e.g., Scoring) or *Robot* size.*

Interpretation: Robots must not have mechanisms that would be left on the field.

Impact: You can't leave mechanisms, for example on the goal, to protect the scored rings. This makes the game more interesting and fluid because descoring can happen.

<G7> Don't clamp your Robot to the Field. *Robots* may not intentionally grasp, grapple, or attach to any *Field Elements* other than the *Ladder*. 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 both unintentionally damaging the *Field* and/or from anchoring themselves to the *Field* in locations other than the *Ladder*.

Interpretation: Don't use a part of your robot to strongly grasp onto parts of the field other than the ladder for climbing

Impact: Robots must not clamp onto other parts of the field for a strategic advantage. Robot design must also make sure that it can't unintentionally clamp to the field. This stops strategies that clamp to the edge of the field to stop people from moving the mobile goals outside of the corner modifier zones.

<G8> Only Drive Team Members, and only in the Alliance Station. During a *Match*, each *Team* may have up to three (3) *Drive Team Members* in their *Alliance Station*, and all *Drive Team Members* must remain in their *Alliance Station* for the duration of the *Match*.

Drive Team Members are prohibited from any of the following actions during a *Match*:

- a. Bringing/using any sort of communication devices into the *Alliance Station*. Non-headphone devices with communication features turned off (e.g., a phone in airplane mode) are allowed.
- b. Standing on any sort of object during a *Match*, regardless of whether the *Field* is on the floor or elevated.
- c. Bringing/using additional materials to simplify the game challenge during a *Match*.
- d. To ensure that *Drive Team Members* are aware of verbal calls or warnings during a *Match* (as an application of rules <T1>, <G1>, <S1>, and <G3>), powered headphones, earbuds, and passive earpieces connected to electronic devices cannot be worn/used in the *Alliance Station* except as required by an officially approved accommodation request.

Interpretation: In the drivers box, it is only the 3 people on your team in there that should be interacting with the driver or the robot.

Impact: We only have 3 people on our team, so everyone will be in the driver's box, and we will also make sure to turn on airplane mode on our phones, to make sure that we are not breaking the rules.

<G9> Hands out of the field. *Drive Team Members* are prohibited from making intentional contact with any *Scoring Objects*, *Field Elements*, or *Robots* during a *Match*, apart from the contact specified in <G9a>.

- a. During the *Driver Controlled Period*, *Drive Team Members* may only touch their own *Robot* if the *Robot* has not moved at all during the *Match*. Touching the *Robot* in this case is permitted only for the following reasons:
 - i. Turning the *Robot* on or off
 - ii. Plugging in a battery
 - iii. Plugging in a V5 *Robot* Radio
 - iv. Touching the V5 *Robot* Brain screen, such as to start a program
- b. *Drive Team Members* are not permitted to break the plane of the *Field Perimeter* at any time during the *Match*, apart from the actions described above, or while reintroducing *Scoring Objects* to the *Field* as described in rule <SG4>
- c. Transitive contact, such as contact with the *Field Perimeter* that causes the *Field Perimeter* to contact *Field Elements* or *Scoring Objects* inside of the *Field*, could be considered a *Violation* of this rule.

Interpretation: This rule ensures both the safety of the game, and the integrity of robot actions during a match by limiting how humans are allowed to interact with their robot.

Impact: The only time that people are allowed to put their hands in the field, are debugging small issues with the robot(only if it has not moved), or adding match loads into the field.

<G10> Controllers must stay connected to the field. Prior to the beginning of each *Match*, *Drive Team Members* must plug their V5 Controller into the field's control system. This cable must remain plugged in for the duration of the *Match*, and may not be removed until the "all-clear" has been given for *Drive Team Members* to retrieve their *Robots*. See <T23>for more information regarding field control system options.

Interpretation: Controllers for the robots must always be connected to the field for the duration of the match so that the integrity of the competition is protected by the match controller.

Impact: Robots must always follow the instructions from the match controller, such as the beginning and end of the autonomous and driver periods.

<G11> Autonomous means "no humans." During the *Autonomous Period*, *Drive Team Members* are not permitted to interact with the *Robots* in any way, directly or indirectly. This could include, but is not limited to:

- Activating any controls on their V5 Controllers
- Unplugging or otherwise manually interfering with the field connection in any way
- Manually triggering sensors (including the Vision Sensor) in any way, even without touching them

Interpretation: During autonomous there must be no driver interactions with the robot.

Impact: Autonomous periods must operate without any driver assistance.

<G12> All rules still apply in the Autonomous Period. Teams are responsible for the actions of their Robots at all times, including during the Autonomous Period. Any Violations, Major or Minor, during the Autonomous Period will result in the Autonomous Bonus being awarded to the other Alliance. If both Alliances violate rules during the Autonomous Period, no Autonomous Bonus will be awarded.

Interpretation: Robots must still follow the robot rules during autonomous control of the robot, but more benefit is given to the offending team because autonomous control can often be difficult.

Impact: Robots must not be attempting to break rules during autonomous, but if rules are broken, the impact will be less than if the same action were completed during driver, often only affecting the winners of the autonomous bonus.

<G13> Don't destroy other Robots. But, be prepared to encounter defense. Strategies aimed solely at the destruction, damage, tipping over, or Entanglement of opposing Robots are not part of the ethos of the VEX V5 Robotics Competition and are not allowed.

- a. V5RC High Stakes is intended to be an offensive game. Teams that partake in solely defensive or destructive strategies will not have the protections implied by <G13>(see <G14>). However, defensive play which does not involve destructive or illegal strategies is still within the spirit of this rule.
- b. V5RC High Stakes is also intended to be an interactive game. Some incidental tipping, Entanglement, and damage may occur as a part of normal gameplay without Violation. It will be up to the Head Referee's discretion whether the interaction was incidental or intentional.
- c. A Team is responsible for the actions of its Robot at all times, including the Autonomous Period. This applies both to Teams that are driving recklessly or potentially causing damage, and to Teams that drive around with a small wheel base. A Team should design its Robot such that it is not easily tipped over or damaged by minor contact.

Interpretation: Robots may not intentionally cause damage to the field or other robots; destructive mechanisms are prohibited.

Impact: We cannot use dangerous mechanisms. Our robots must be durable and robust.

<G14> Offensive Robots get the “benefit of the doubt.” In a case where Head Referees are forced to make a judgment call regarding a destructive interaction between a defensive and offensive Robot, or an interaction which results in a questionable Violation, referees will decide in favor of the offensive Robot.

Interpretation: In any aggressive interaction between two teams, the defensive robot is more likely to be impacted if a violation occurs.

Impact: More offensive robots are much safer to use in matches. We may need to take extra precautions when playing defense.

<G15> You can't force an opponent into a penalty. Intentional strategies that cause an opponent to break a rule are not permitted, and will not result in a *Violation* for the opposing *Alliance*.

Impact: We can't have mechanisms or strategies that break this rule. For example, we can't try to pull our opponents over the white line in autonomous.

<G16> No Holding for more than a 5-count. A *Robot* may not *Hold* an opposing *Robot* for more than a 5-count during the *Driver Controlled Period*.

For the purposes of this rule, a “count” is defined as an interval of time that is approximately one second in duration, and “counted-out” by *Head Referees* verbally.

A *Holding* count is over when at least one of the following conditions is met:

- a. The two *Robots* are separated by at least two (2) feet (approximately one foam tile).
- b. Either *Robot* has moved at least two (2) feet away (approximately 1 tile) from the location where the *Trapping* or *Pinning* count began.
 - i. In the case of *Lifting*, this location is measured from where the *Lifted Robot* is released, not from where the *Lifting* began.
- c. The *Holding Robot* becomes *Trapped* or *Pinned* by a different *Robot*.
 - i. In this case, the original count would end, and a new count would begin for the newly Held *Robot*.
- d. In the case of *Trapping*, if an avenue of escape becomes available due to changing circumstances in the *Match*.

After a *Holding* count ends, a *Robot* may not resume *Holding* the same *Robot* again for another 5-count. If a *Team* resumes *Holding* the same *Robot* within that 5-count, the original count will resume from where it ended.

Interpretation: Teams can't limit an opponents movement for more than 5 seconds.

Impact: Again, an offensive robot is protected from defense, however short/strategic holds can have a good impact. Additionally, mechanism such as wedges on the side of a robot may cause “pushing” to turn into “lifting”.

<G17> Use Scoring Objects to play the game. *Scoring Objects* may not be used to accomplish actions that would be otherwise illegal if they were attempted by *Robot* mechanisms. Examples include, but are not limited to:

- Interfering with an opponent's Autonomous routine per <SG8>
- Interfering with an opponent's *Climb* per <SG9>

Impact: We can't use game elements as gloves or loopholes.

05/08/24 Identify: Specific Game Rules

Designed by: Alex

Witnessed by: Carl

Witnessed on: 05/08/24

<SG1> Starting a Match. Prior to the start of each *Match*, the *Robot* must be placed such that it is:

- Contacting / “breaking the plane” of their *Alliance’s Starting Line*. See Figure SG1-1.
- Not contacting any *Scoring Objects* other than a maximum of one (1) preload. See rule <SG5>.
- Not contacting any other *Robots*.
- Completely stationary (i.e., no motors or other mechanisms in motion).

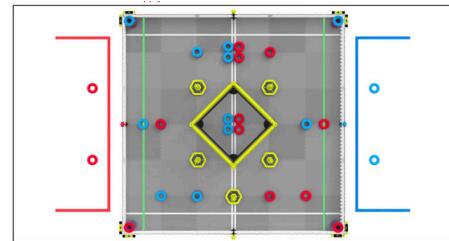


Figure SG1-1: An overhead view of the Field, with the Starting Lines highlighted green.

Interpretation: Before autonomous robots must be set up in regular positions on the autonomous line.

Impact: Robots have to move so that at the end of autonomous they are not breaking this plane anymore for the autonomous win point, so all teams have to move.

<SG2> Horizontal expansion is limited. Once the *Match* begins, *Robots* may expand beyond the 18" x 18" starting size, within the following criteria:

- Robots* may never exceed an overall footprint of 24" x 18". For reference, 24" is roughly the width of a foam field tile.
- From the *Robot’s* perspective, they may only expand in one “X/Y” direction (i.e., from a single “side” of the *Robot*). This “side” must be identified and measured during *Robot* inspection. See the figures below.
- Vertical expansion is addressed separately by rule <SG3>. *Robots* may expand both horizontally and vertically; the top of the *Robot* is not considered a “side” in the context of this rule.

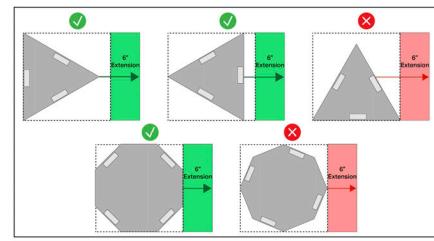


Figure SG2-1: An overhead view of the Field, with the Starting Lines highlighted green.

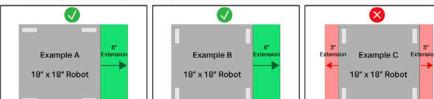


Figure SG2-2: An overhead view of the Field, with the Starting Lines highlighted green.

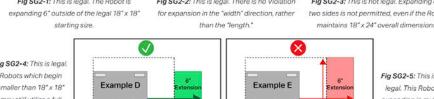


Figure SG2-3: An overhead view of the Field, with the Starting Lines highlighted green.

Note: Horizontal expansion is measured from the Robot’s perspective; i.e., it does “rotate with the Robot.” Robots that tip over, or rotate while Climbing, are still restricted to expanding from the chosen “side” that was measured during inspection.



Figure SG2-4: An overhead view of the Field, with the Starting Lines highlighted green.

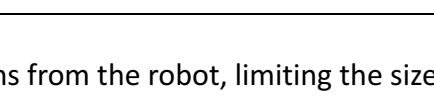


Figure SG2-5: An overhead view of the Field, with the Starting Lines highlighted green.

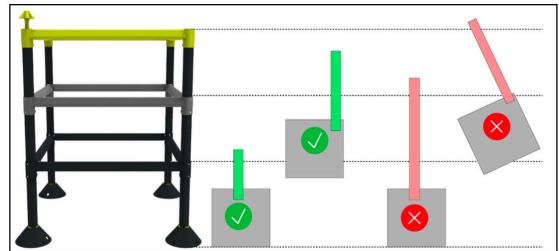
Interpretation: This rule prevents the robot from having multiple extrusions from the robot, limiting the size of the robot very easily and making sure extra robot manipulators don’t become too large.

Additional note: the top photos illustrate well that ONLY one side on a robot may expand throughout the match

Impact: This will prevent robots from enlarging too much during the match, and in the design stage it will make it so robot manipulators that come out of the robot must only go out one way. This rule also means that the hang mechanism must come out the same side of the robot that the other manipulator comes out with, making it more advantageous to combine them into one manipulator.

<SG3> Vertical expansion is limited. Once the *Match* begins, *Robots* may expand vertically, but may never be “breaking the plane” of more than two *Levels* of the *Ladder* at any given time. For the purposes of this rule, the *Floor* is considered a *Level*.

- For a *Robot* that is on the *Floor* (i.e., not *Climbing*), this is effectively a height limit of 32”, the distance between the *Floor* and the top of the middle rung of the *Ladder*.
- This vertical limit is measured from the perspective of the *Field*; i.e., it does not “rotate with the *Robot*.”



Interpretation: This rule prevents any robots from skipping rungs on the ladder when climbing to ensure that the robot hanging mechanisms are more interesting.

Impact: Robots must spend extra time climbing because they can't climb all the way at once. This will affect how strategically beneficial it is to climb.

<SG4> Keep Scoring Objects in the field. *Teams* may not intentionally or strategically remove *Scoring Objects* from the field. *Rings* that leave the *Field* during *Match* play, intentionally or unintentionally, will be given to *Drive Team Members* from the same color *Alliance* as the *Ring*. These *Drive Team Members* may gently place them into the field such that they satisfy the following conditions:

- Contacting the *Field Perimeter* wall on the side that coincides with their *Alliance Station*.
- Contacting the *Floor*.
- Not contacting a *Mobile Goal*.
- Not contacting a *Robot*.
- Not contacting a *Corner*.

Interpretation: Teams must not remove field elements for strategic elements to ensure that teams have adequate objects still in play on the field.

Impact: Teams can't remove objects to starve their opponent of points, but if objects accidentally fall out of the field, teams aren't punished.

<SG5> Each Robot gets one Ring as a preload. Prior to the start of each *Match*, each preload that is used must be placed such that it is:

- Contacting one *Robot* of the same *Alliance* color as the preload.
- Not contacting the same *Robot* as another preload.

Interpretation: Teams can utilize their alliance preloads, one for each robot.

Impact: At the beginning of the match, teams are already loaded with one Ring, which makes it easier to achieve the AWP because the robots have to get less rings throughout the match.

<SG6> Possession is limited to two Rings and one Mobile Goal. Robots may not have *Possession* of more than two (2) *Rings* and one (1) *Mobile Goal* at once. *Robots* in *Violation* of this rule must immediately stop all actions except for attempting to remove the excess *Scoring Objects*.

- a. A *Robot* that is *Violating* this rule while *Possessing* any of the opposing *Alliance's Rings* may not participate in further gameplay other than removing the excess *Scoring Objects*.
 - i. If they are unable to remove the excess *Scoring Objects*, then they must return to a legal starting position (as described by <SG1>). They will not be eligible to receive points for *Climbing*. Any offensive or defensive interactions with *Mobile Goals*, *Stakes*, and *Corners* will be included in *Match Affecting* consideration.
- b. *Rings* on a *Stake* are not included in a *Robot's Possession* count. For the purposes of this rule, "on a *Stake*" means that the *Ring* meets the criteria for a *Scored Ring*, even if it is being contacted by a *Robot*.
- c. Plowing multiple *Mobile Goals* is permitted. However, *Plowing* an additional *Mobile Goal* while also *Possessing* one is considered a *Violation* of this rule due to the extremely high likelihood of accidental/implied *Possession*. *Teams* which employ *Plowing* strategies are encouraged to clearly demonstrate that none of the *Mobile Goals* Are being *Possessed*, e.g., by using a flat face of the *Robot* with no active mechanisms.

Violation Notes:

Any egregious or clearly intentional *Violation* by an *Alliance* who wins the *Match* will be considered *Match Affecting*.

Interpretation: This rule limits how many items teams are allowed to carry at once.

Impact: This means that at any given time, there will always be a goal that is free on the field. Additionally, this means that if a team is trying to switch goals to fill up another goal, they have to leave the goal that they filled somewhere on the field, which is susceptible to descoring. This part of the game will keep matches moving and make scoring on the neutral stakes, which are harder to de-score from, much more beneficial to filling up the goals earlier in the match.

For the ring possession limit teams, the designs for ring scoring mechanisms must ensure that they only possess one at a time. Also, rings on goals do not count, so it will likely make it so that teams focus on mechanisms that have many small cycles to score rings instead of mechanisms that collect a lot of rings and then score them all at once.

<SG7> Don't cross the Autonomous Line. During the *Autonomous Period*, *Robots* may not contact foam tiles, *Scoring Objects*, or *Field Elements* which are on the opposing *Alliance's* side of the *Autonomous Line*.

Interpretation: Robots must stay isolated to their side of the field to ensure that they don't mess up their opponents' autonomous.

Impact: Teams must make sure that if they are rushing the middle line in autonomous they are not going to be dragged across the field.

<SG8> Engage with the Autonomous Line at your own risk. Any *Robot* who engages with *Scoring Objects* and/or *Wall Stakes* on the *Autonomous Line* should be aware that opponent *Robots* may also choose to do the same. Per <G11> and <G12>, *Teams* are responsible for the actions of their *Robots* at all times.

Interpretation: Teams that attempt to interact with the objects on the autonomous line are not fully safe from opponent interference.

Impact: Teams must keep in mind when planning their autonomous routine that ones that interact with the middle line are likely to be messed up with another team, so they must be designed to recover from this.

<SG9> Don't remove opponents from the Ladder. There are no rules explicitly prohibiting incidental contact between *Climbing Robots*. However, if contact does occur, the principles behind rules <G13>, <G14>, and <G15> still apply. Intentional or egregious strategies aimed solely at damage or tipping are not allowed (in this context, "tipping" can be equated with "removing an opponent from the *Ladder*").

Interpretation: Removing teams from the Ladder, especially because it is a dangerous object to climb will result in consequences for teams who try it.

Impact: If we try to climb the ladder we must ensure that we aren't going to run into other robots when climbing; we are somewhat protected when climbing.

<SG10> Alliance Wall Stakes are protected. *Robots* may not directly or indirectly interact with the opponent's *Alliance Wall Stake*. This includes both *Scoring* and/or removing *Rings* of either color.

Interpretation: Team's cannot interact with the opponents Alliance Wall Stake at any point during the match.

Impact: When we score rings onto our alliance wall stake, the rings that we scored there are safe for the rest of the match.

05/09/24 Identify: Robot Rules

Designed by: Carl, Alex	Witnessed by: Matt	Witnessed on: 05/09/24
-------------------------	--------------------	------------------------

<R1> One Robot per Team. Only one (1) *Robot* will be allowed to compete per *Team* at a given event in the VEX V5 Robotics Competition. Though it is expected that *Teams* will make changes to their *Robot* at the competition, a *Team* is limited to only one (1) *Robot* at a given event. A VEX *Robot*, for the purposes of the VEX V5 Robotics Competition, has the following subsystems:

- Subsystem 1: Mobile robotic base including wheels, tracks, legs, or any other mechanism that allows the *Robot* to navigate the majority of the flat playing field surface. For a stationary *Robot*, the robotic base without wheels would be considered Subsystem 1.
- Subsystem 2: Power and control system that includes a legal VEX battery, a legal VEX control system, and associated motors for the mobile robotic base.
- Subsystem 3: Additional mechanisms (and associated motors) that allow manipulation of *Rings*, *Field Elements*, or *Climbing* the *Ladder*.

Interpretation: Teams may only use one robot and must use the same robot for the entire competition.

Impact: If we choose to make two robots, we can only bring one robot to the competition.

<R2> Robots must represent the Team's skill level. The *Robot* must be designed, built, and programmed by members of the *Team*. *Adults* are expected to mentor and teach design, building, and Programming Skills to the *Students* on the *Team*, but may not design, build, or program that *Team's Robot*. See rules **<G2>** and **<G4>**.

Interpretation: A team must be completely responsible for the design, building, programming, etc...

Impact: We will continue to develop our robot solely with team members.

<R3> Robots must pass inspection. Every *Robot* will be required to pass a full inspection before being cleared to compete. This inspection will ensure that all *Robot* rules and regulations are met. Initial inspections will take place during team registration/practice time. Noncompliance with any *Robot* design or construction rule will result in removal from *Matches* or *Disqualification* of the *Robot* at an event until the *Robot* is brought back into compliance, as described in the following subclauses.

Interpretation: All teams fully complete inspection before they are allowed to compete.

Impact: We will ensure that our robot is completely legal prior to the competition to avoid problems with inspection.

<R4> Robots must fit within an 18" x 18" x 18" volume.

- a. Compliance with this rule must be checked using the official [VEX Robotics On-Field Robot Expansion Sizing Tool](#).
- b. Any restraints used to maintain starting size (i.e., zip ties, rubber bands, etc.) must remain attached to the *Robot* for the duration of the *Match*, per <G6>.
- c. For the purposes of this rule, it can be assumed that *Robots* will be inspected and begin each *Match* on a flat standard foam field tile.

Impact: We must fit in the predefined volume at the start of a match without using detachable mechanisms. After the match begins, we must abide by <SG2>/<R5>.

<R5> Robots may only expand horizontally in one direction. *Robots* who choose to expand horizontally must demonstrably meet all criteria listed in rule <SG2>. The configuration / “expansion direction” that is measured during inspection must also be the direction used during *Match* play.

Interpretation: Implements <SG2> into the robot rules.

Impact: Already identified in <SG2>.

<R6> Robots must be safe. The following types of mechanisms and components are NOT allowed:

- a. Those that could potentially damage *Field Elements* or *Scoring Objects*.
- b. Those that could potentially damage other competing *Robots*.
- c. Those that pose an unnecessary risk of *Entanglement* with other *Robots* or *Field Elements*.
- d. Those that could pose a potential safety hazard to *Drive Team Members*, event staff, or other humans.

Interpretation: Dangerous or damaging mechanisms are prohibited.

Impact: When designing, we must be careful not to create exceptionally dangerous mechanisms.

<R7> Robots are built from the VEX V5 system. *Robots* may be built ONLY using official VEX V5 components, unless otherwise specifically noted within these rules. Product pages on the VEX Robotics website should be used as the official definitive source for determining if a product is a “V5 component.”

Impact: We can't design or build with parts not from VEX unless explicitly stated in <R8>.

<R8> Certain non-VEX components are allowed. *Robots* are allowed the following additional “non-VEX” components:

- a. Any material strictly used as a color filter or a color marker for a legal sensor, such as the VEX Light Sensor or the VEX V5 Vision Sensor.
- b. Any non-aerosol-based grease or lubricating compound, when used in extreme moderation on surfaces and locations that do NOT contact the playing field walls, foam field surface, *Scoring Objects*, or other *Robots*. Grease or lubricant applied directly to V5 Smart Motors or Smart Motor cartridges is prohibited.
- c. Anti-static compound, when used in extreme moderation (i.e., such that it does not leave residue on *Field Elements*, *Scoring Objects*, or other *Robots*).
- d. Hot glue when used to secure cable connections.
- e. An unlimited amount of rope/string, no thicker than 1/4" (6.35 mm).
- f. Commercially available items used solely for bundling or wrapping of 2-wire, 3-wire, 4-wire, or V5 Smart Cables, and/or pneumatic tubing are allowed. These items must solely be used for the purposes of cable/tubing protection, organization, or management. This includes but is not limited to electrical tape, cable carrier, cable track, etc. It is up to inspectors to determine whether a component is serving a function beyond protecting and managing cables and tubing.
- g. Non-functional 3D printed license plates, per <R9> and <R10>, are permitted. This includes any supporting structures whose sole purpose is to hold, mount, or display an official license plate.
- h. Rubber bands that are identical in length and thickness to those included in the VEX V5 product line (#32, #64, and 117B).
- i. Pneumatic components with identical SMC manufacturer part numbers to those listed on the VEX website. For more detail regarding legal pneumatic components, see the [Legal VEX Pneumatics Summary document](#).
- j. Zip ties with identical dimensions as those included in the VEX V5 product line.
- k. A Micro SD card installed in the V5 Robot Brain.

Interpretation: Teams may use specific parts as an exception to <R7>.

Impact: Using these parts can allow for more innovative solutions to problems that would not be possible with standard VEX parts.

<R9> Decorations are allowed. *Teams* may add non-functional decorations, provided that they do not affect *Robot* performance in any significant way or affect the outcome of the *Match*. These decorations must be in the spirit of the competition. Inspectors will have final say in what is considered “non-functional.” Unless otherwise specified below, non-functional decorations are governed by all standard *Robot* rules.

Impact: We aren't planning to put non-functional decorations on our robot.

<R10> Officially registered Team numbers must be displayed on Robot license plates. To participate in an official VEX V5 Robotics Competition event, a *Team* must first register on [robotevents.com](#) and receive a V5RC Team number. This *Team* number must be displayed on the *Robot* using license plates. *Teams* may choose to use the official V5RC License Plate Kit, or may create their own.

Impact: Refs will be able to better recognize teams with obvious placement of license plates.

<R11> Let go of Scoring Objects after the Match. *Robots* must be designed to permit easy removal of *Scoring Objects* from any mechanism without requiring the *Robot* to have power after a *Match*.

Impact: All object manipulators on our robot must have a way for humans to easily remove game elements from our robot.

<R12> Robots have one Brain. *Robots* must ONLY use one (1) VEX V5 Robot Brain (276-4810). Any other microcontrollers or processing devices are not allowed, even as non-functional decorations.

<R13> Motors are limited. *Robots* may use any combination of VEX V5 Smart Motors (11W) (276-4840) and EXP Smart Motors (5.5W) (276-4842), within the following criteria:

- a. The combined power of all motors (11W & 5.5W) must not exceed 88W. This limit applies to all motors on the *Robot*, even those which are not plugged in.
- b. V5 Smart Motors, and EXP Smart Motors connected to Smart Ports, are the only motors that may be used with a V5 Robot Brain. The 3-wire ports may not be used to control motors of any kind.

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

Interpretation: Teams may use any combination of V5 motors totaling to a maximum of 88W.

Impact: We must be very intentional with how we utilize motors. We can potentially use pneumatics to actuate a clamp, allowing for more power on the drivetrain.

<R14> Electrical power comes from VEX batteries only. *Robots* may use one (1) V5 Robot Battery (276-4811) to power the V5 Robot Brain.

Impact: Negligible

<R15> No modifications to electronic or pneumatic components are allowed. Motors (including the V5 Smart Motor firmware), microcontrollers (including V5 Robot Brain firmware), cables, sensors, controllers, battery packs, reservoirs, solenoids, pneumatic cylinders, and any other electrical or pneumatics component of the VEX platform may NOT be altered from their original state in ANY way.

Impact: Negligible

<R16> Most modifications to non-electrical components are allowed. Physical modifications, such as bending or cutting, of legal metal structure or plastic components are permitted.

Impact: Negligible

<R17> Robots use VEXnet. *Robots* must ONLY utilize the VEXnet system for all wireless *Robot* communication.

Impact: Negligible

<R18> Give the radio some space. The V5 Radio must be mounted such that no metal surrounds the radio symbol on the V5 Radio.

Impact: Negligible

<R19> A limited amount of custom plastic is allowed. *Robots* may use custom-made parts cut from certain types of non-shattering plastic. It must be possible to have cut all of the plastic parts on the *Robot* from a single 12" x 24" sheet, up to 0.070" thick.

Impact: Polycarbonate is a great material for constructing highly customized mechanisms or parts that need a very high strength to weight ratio. Designing parts to fit efficiently on the 12" by 24" sheet will allow us to use more.

<R20> A limited amount of tape is allowed. *Robots* may use a small amount of tape for the following purposes:

- a. To secure any connection between the ends of two (2) VEX cables.
- b. To label wires and motors.
- c. To cover the backs of license plates (i.e., hiding the “wrong color”).
- d. To prevent leaks on the threaded portions of pneumatic fittings. This is the only acceptable use of Teflon tape.
- e. In any other application that would be considered a “non-functional decoration” per *<R9>*.
- f. As an aglet at the end of rope/string to prevent fraying.

Impact: Negligible

<R21> Certain non-VEX fasteners are allowed. *Robots* may use the following commercially available hardware:

- a. #4, #6, #8, M3, M3.5, or M4 screws up to 2.5" (63.5 mm) long.
- b. Shoulder screws cannot have a shoulder length over 0.20" or a diameter over 0.176".
- c. Any commercially available nut, washer, standoff, and/or non-threaded spacer up to 2.5" (63.5 mm) long which fits these screws.

Impact: We can purchase hardware from 3rd party sources such as aluminum and nylon screws.

<R22> New VEX parts are legal. Additional VEX components released during the competition season on www.vexrobotics.com are considered legal for use unless otherwise noted.

Impact: Negligible.

<R23> Pneumatics are limited. A *Robot's* pneumatic subsystem must satisfy the following criteria:

- a. *Teams* may use a maximum of two (2) legal VEX pneumatic air reservoirs on a *Robot*. The Air Tank 200mL (included in the 276-8750 V5 Pneumatics Kit) and the legacy (pre-2023) reservoir are both considered legal reservoirs.
- b. Pneumatic devices may be charged to a maximum of 100 psi.
- c. The compressed air contained inside a pneumatic subsystem can only be used to actuate legal pneumatic devices (e.g., cylinders).

Interpretation: Teams may use a maximum of 2 reservoirs pressurized to 100 PSI.

Impact: For infrequently used mechanisms or systems that require short, linear motion we can use pistons instead of motors.

<R24> One or two Controllers per Robot. No more than two (2) VEX V5 Controllers may control a single *Robot*.

- a. No physical or electrical modification of these Controllers is allowed under any circumstances.
 - i. Attachments which assist the *Drive Team Member* in holding or manipulating buttons/joysticks on the V5 Controller are permitted, provided that they do not involve direct physical or electrical modification of the Controller itself.
- b. No other methods of controlling the *Robot* (light, sound, etc.) are permissible.
 - i. Using sensor feedback to augment driver control (such as motor encoders or the Vision Sensor) is permitted.

Impact: All robot control must occur at the field either with the controllers, or from input from the sensors on the robot.

<R25> Custom V5 Smart Cables are allowed. *Teams* who create custom cables acknowledge that incorrect wiring may have undesired results.

Impact: We will use custom cables when we need to for better wire management, however we will always robustly check our wires after doing this.

<R26> Keep the power button accessible. The on/off button on the V5 Robot Brain must be accessible without moving or lifting the *Robot*. All screens and/or lights must also be easily visible by competition personnel to assist in diagnosing *Robot* problems.

Impact: We must design adequate space around the V5 brain to allow for inspectors and us to be able to access the button to turn it off.

<R27> Use a “Competition Template” for programming. The *Robot* must be programmed to follow control directions provided by the VEXnet Field Controllers or Smart Field Control system.

During the *Autonomous Period*, *Drive Team Members* will not be allowed to use their V5 Controllers. As such, *Teams* are responsible for programming their *Robot* with custom software if they want to perform in the *Autonomous Period*. *Robots* must be programmed to follow control directions provided by the field controls (e.g., ignore wireless input during the *Autonomous Period*, *Disable* at the end of the *Driver Controlled Period*, etc.).

Interpretation: Teams must use the programming template specially designed for competition that listens to the field controller commands.

Impact: All robots automatically respond to the field controller at the right times during the match.

<R28> There is a difference between accidentally and willfully violating a Robot rule. Any *violation* of *Robot* rules, accidental or intentional, will result in a *Team* being unable to play until they pass inspection (per <R3d>).

Interpretation: Teams must not purposely violate a robot rule. Violations will be punished the same.

Impact: We won't violate robot rules, so we will ensure that our robot is compliant before every competition.

05/10/24 Identify: Robot Skills Rules

Designed by: Matt	Witnessed by: Carl	Witnessed on: 05/11/24
-------------------	--------------------	------------------------

<RSC1> All rules from “The Game” section of the manual apply to the Robot Skills Challenge, unless otherwise specified in this section.

Interpretation: All rules are the same except those stated below.

Impact: We will make sure to know the changes and implement them when doing Robot Skills.

<RSC2> *Teams* may play *Robot Skills Matches* on a first-come, first-served basis, or by a pre-scheduled method determined by the *Event Partner*. Each *Team* will get the opportunity to play up to three (3) *Driving Skills Matches* and three (3) *Autonomous Coding Skills Matches*.

Interpretation: Teams must get their skills run on their own or set up by an event partner. Each team gets 3 skills runs for each and competition should allow for that.

Impact: Make sure we get our skills runs completed during the competition or in the time that the EP set out for us.

<RSC3> *Robots* must start the *Robot Skills Match* in a legal starting position for the red *Alliance*.

- a. All *Drive Team Members* must remain in the red *Alliance Station* for the duration of the *Match*.
- b. *Robots* must meet all of the criteria listed in rule <SG1>.
- c. *Teams* may use one (1) red *Alliance* preload as described in rule <SG5>.
- d. The two (2) blue *Alliance* preload *Rings* are not used in *Robot Skills Matches*.

Interpretation: Robots must start in the red alliance position for skills along with drive team members in the red alliance box.

Impact: Make sure to practice programming and driver skills on the correct starting position.

<RSC4> Teams may only utilize the blue *Rings* as *Top Rings* on *Stakes*. Each Blue *Ring* only has a point value if:

- a. All red *Rings* in the *Match* have been *Scored* on *Stakes* and have point values.
- b. At least one red *Ring* is *Scored* below the blue *Ring* on that *Stake*.
- c. There is only one blue *Ring* on that *Stake*.
- d. No red *Rings* are *Scored* above the blue *Ring* on that *Stake*.

Interpretation: Blue rings only count as scored in skills if it is the top ring along with other scenarios.

Impact: Make sure to optimize our skills path to make sure when scoring blue rings it fits the criteria. Optimize driving, programming, and the robot so small inconsistencies don't make a difference. Reliability of all aspects of ring manipulation is crucial.

<RSC5> Any red *Ring Scored* above a blue *Ring* on the same *Stake* will not have a point value.

Interpretation: Any red ring above a blue one does not count.

Impact: Optimize our skills path to make sure we don't score any red rings on top of blue ones.

<RSC6> If any *Ring* is *Scored* on a *Stake* but does not have a point value based on rule **<RSC4>** or **<RSC5>**, no *Ring* on that *Stake* will earn points as a *Top Ring*.

Interpretation: If any ring on the mobile goal isn't considered scored according to the prior rules then no top ring will be scored. Eg. 1 blue ring is scored underneath a red ring.

Impact: Make sure that our path doesn't pick up a ring that doesn't count as scored.

<RSC7> No *Corner* Modifiers.

- a. There are no *Positive Corners* or *Negative Corners* in *Robot Skills Matches*.
- b. Each *Mobile Goal Placed* in a *Corner* will receive 5 points. Rule **<SC5>** and its note still apply, and only one *Mobile Goal* may be *Placed* in each *Corner*.

Interpretation: There are no corner multipliers in robot skills and goals in the corner have a fixed point value.

Impact: It is beneficial to put mobile goals in ALL corners of the field.

<RSC8> There is no requirement that Skills Challenge *Fields* have the same consistent modifications as the Head-to-Head *Fields*. For example, there is no requirement that all Skills Challenge *Fields* are elevated to the same height as Head-to-Head *Fields*. However, all Skills Challenge *Fields* at a single event must use the same type of field control and *Field Perimeter*, as described in rules **<T23>** and **<T24>**.

It is strongly recommended/pREFERRED that all Skills Challenge *Fields* are consistent with each other, but this may not be the case in extreme circumstances.

In order to use non-conforming Head-to-Head *Fields* for Skills Challenge runs (e.g., during lunch), the following steps should be taken:

- *Teams* must be informed that the Head-to-Head *Fields* may have some differences from the Skills Challenge *Fields* (e.g., they might not have GPS strips).
- *Teams* must be given an opportunity to select which type of *Field* they want to use, i.e. they cannot be required to use the Head-to-Head *Field* for any Skills Challenge run.

Interpretation: All skills fields must be uniform at a competition. Skills fields do not have to be elevated at competitions. Operations during lunch time may affect GPS.

Impact: If using GPS don't run skills during lunch. Skills should work with both field perimeters.

Skills Scoring	
Each Ring Scored on a Stake	1 Point
Each Top Ring on a Stake	3 Points
Climb - Level 1	3 Points
Climb - Level 2	6 Points
Climb - Level 3	12 Points
Mobile Goal Placed in a Corner	5 Points

05/10/24 Identify: Programming Objectives and Challenges

Designed by: Alex

Witnessed by: Matt

Witnessed on: 05/10/24

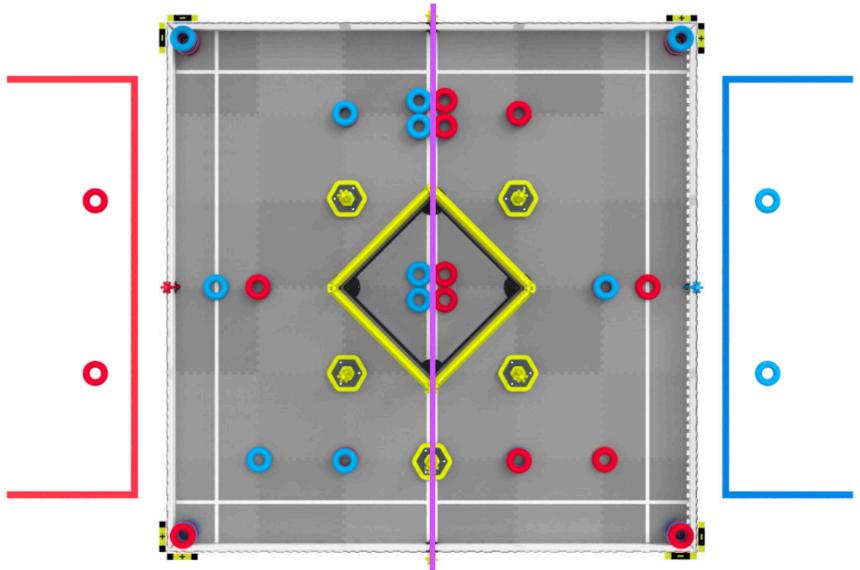
Goal: Read and understand the game manual and the field design and its impact on programming objectives, primarily focusing on requirements for programming drivetrain motion algorithms.

This year there are several features about the field design that make it especially notable. This includes the open field, irregular field mirroring, and moving scoring posts, and high accuracy requirements.

Irregular Field Mirroring (Not rotation):

This year the field is not rotated, like most years in VEX games, it is instead mirrored across the middle of the field, based on the alliance. To the right is a depiction of the mirroring line, here in purple that shows the reflection on the field. Most years this rotation effect allowed teams to use the same autonomous routine regardless of which alliance they are on in the match.

This year teams will have to have autonomous routines that are different



depending on which side of the field the robot is starting in. This is a more difficult problem that we will have to solve with both turning to different positions on the field, which the turning direction will change based on the alliance, and the paths will have to be mirrored on this line.

Scoring on Stakes:

This year the objects that we have to score on will likely be moving or in a slightly different position depending on how the field is reset. This generally leads to both the software and hardware on the robots having to be much different to accomplish this increased accuracy we will likely have to add guides to our robot to center the goals, or for a software solution we could utilize the AI vision sensor to locate where the goal is and then move to it more accurately, by achieving something closer to terminal guidance.

BACKGROUND RESEARCH

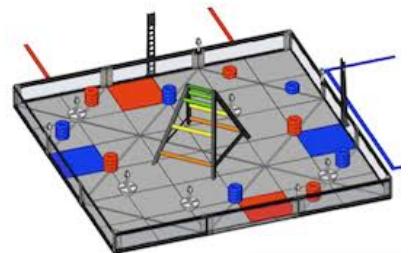
05/11/24

Background Research: Round Up Analysis

Designed by: Matt, Carl	Witnessed by: Alex	Witnessed on: 05/12/24
-------------------------	--------------------	------------------------

Goal: Examine VRC Round Up and identify successful design decisions; determine if these are still viable with the new game rules and V5 motors.

This year's game High Stakes was designed after the game VRC Round Up from 2011. These games are very similar; they both consist of plastic rings and mobile goals that you can move around the field and an elevation element in the center. There are also some key differences in these games such as the ring possession limit which was 4 in Round Up, but is now only 2 rings. Another key difference is the field setup; In Round Up the field had stacks of the same color whereas in High Stakes the rings are mixed. A unique addition in this season is the multiplier corners, which allow for fast point swings.



(Image from team254.com)

Common Robots/Meta:

- 4 Bar: Teams used 4 bar mechanisms to put rings on both mobile goals and to raise the rings onto the wall stakes on the perimeter.
- Claw or Vertical Intake: Attached to the 4 bar mechanism was commonly a claw or vertical intake to pick up the rings off the field and to descore them from mobile goals or wall stakes.

Impact of Significant Differences:

Lower Possession Limit:	Field Setup:	Multiplier Corners:
With the lower possession limit of only two rings, teams will generally need to make more cycles in order to fill up a stake. This will also require mechanisms that are much faster at picking up individual rings, however, they are also likely to be smaller and lighter because of the reduced load.	The field arrangement features a large variety of stacks of rings with different heights and colors. Having some form of mechanism to differentiate between ring colors and be flexible to different stack heights may be beneficial. This will likely result in less large claw design after considering the possession limit.	In Round Up, teams could fill up mobile goals and place them under the elevation structure, where they were protected. In this game, filling up a mobile goal and leaving it loose on the field is very risky as a team could grab it and put it in the negative zone. Leaving goals in the corner will also be very risky because if the opposing team takes one of your fully loaded goals from a + corner to a - corner, you lose 24 points.

05/11/24

Background Research: Turning Point Analysis

Designed by: Carl

Witnessed by: Matt

Witnessed on: 05/11/24

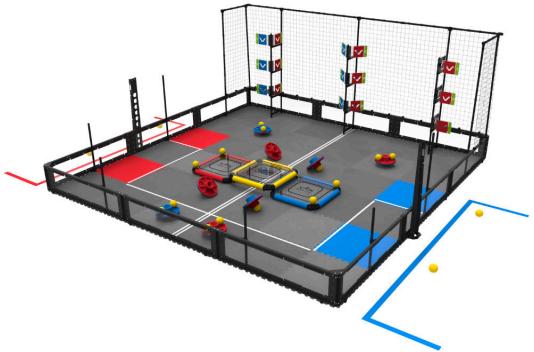
Goal: Find successful strategies and designs for VRC Turning Point and determine their relevance to the current game.

The turning point game featured flags, caps, and platform parking at the end of the match. These tasks have some similarity to this year because of the speed in order in which they are scored. The platforms are quite similar to the hang this year; while last second elevations are possible, they are lower in point value compared to the higher hangs or platform balances that take place earlier. The flags correspond to placing goals in the corner; large point swings can happen very quickly, for example, moving a mobile goal full of the opposing team's rings from the + corner to the - corner. Scoring caps on poles in tipping point is comparable to scoring rings on poles; they are not the fastest score, but these are also points that are very unlikely to be de-scored.

After watching high-level matches from turning point worlds, we noticed two main strategies that emerged:

1. Last second shots
 - a. During the match, at least one team would focus on only shooting flags, which was often mirrored by the other alliance, meaning the flags remain just about even throughout the match. The main defining element of this strategy, however, was teams would wait to the very last second of a match to shoot at the flags in an attempt to not give the other alliance a chance to de-score them. Teams utilizing the strategy were almost never able to get onto the high platform, and sometimes barely made it onto their low platform.
2. Slower, more reliable caps
 - a. Some teams opted for designs that could score caps very efficiently, however, these points almost always took longer than shooting flags. The big advantage of scoring caps was that they were significantly harder for the opposing team to de-score, meaning that in competitive matches especially, they had a more guaranteed point advantage going into the endgame. Additionally, not having to play defense on these caps generally allowed teams that went with the strategy a much longer period to get onto the high platform. The largest drawback of going for caps was that only having caps was not enough to win, and some flags were still required.

Essentially, we believe that this year there is a good chance that match play will likely include a safer strategy with scoring on the wall mounted stakes, and a more aggressive strategy by scoring on mobile goals and utilizing the point multipliers.



(Image from vexrobotics.com)

05/11/24 Background Research: Important Characteristics of the Chassis

Designed by: Carl, Matt

Witnessed by: Alex

Witnessed on: 05/11/24

Goal: Maximize our understanding of how we analyze drivetrain characteristics and interactions with other subsystems.

This information will be helpful when designing our first robot, but it will also be something that we can refer back to for future designs to help speed up the process. This abundance of pre planning will help prevent some oversights that we have had in previous years. In order to provide a clear insight into how we will design our chassis, we elected to preemptively provide the objective information and knowledge that we have obtained in previous years. This will help explain future decisions that have a lot of layers and hundreds of theoretical variations. In the future, this information will also be treated as “common knowledge”, which will help us focus on specific game development aspects.

Chassis Characteristics

Speed	Speed is a crucial factor of a good chassis and can significantly improve a robot's performance. This allows the robot to be more efficient at cycle times, maneuvering the field, and defense. Drive speed can affect the amount of points you score during autonomous and programming skills.
Pushing Power	Pushing powers is critical for playing defense against other kinds of robots. While not as needed for programming skills and autonomous it is crucial to have it to elevate your defensive play by being able to push other robots away and pin them but to also avoid that for yourself.
Strafing	Strafing is an extremely helpful method for mobility due to the sideways motion it adds to the robot. This can make it easier and faster when trying to navigate around objects without having to slow down or turn. The ability to strafe does take away drive power making the robot more susceptible to being pushed around.
Agility	Agility is a massive factor when designing a robot because the more agile you can make it the more scoring potential you get. Ideally the robot should be able to last longer than the 2 minute period meaning it won't show any signs of slowing down or overheating regardless of how much defense is being played. This is important because we want to keep an edge on our opponents who did not design their robot to last as long.

Controllability/Driver Preference	It is important to be able to control the robot. While commonly a glossed over topic many drivers and programmers take for granted the fine tuning and slop-reducing the builders have done. With the high controllability drivers can outmaneuver many other robots and be one step ahead. While programmers can be more competitive at the highest level with the reliability you need.
Mounting Space	For subsystems on the robot it is important to have mounting space and good attachment points. These are needed to craft a compact and well-made robot.
Obstacle Performance	An important aspect of each game is to effectively maneuver the field and this can consist of obstacles such as a barrier. While sometimes tricky to do and with it generally taking up more space, it is very important and can provide a huge strategic advantage.

05/11/24

Background Research: Chassis Types

Designed by: Carl	Witnessed by: Alex	Witnessed on: 05/11/24
-------------------	--------------------	------------------------

Goal: Examine different types of drivetrains and the basics of how each one functions.

X-Drive

Description	An X-Drive is a holonomic drive consisting of four Omni wheels arranged in an "X" pattern, with each wheel independently powered by a motor. By adjusting the speeds of the wheels in different combinations, the robot can move forward, backward, sideways, or rotate in place. Through more advanced software control, this chassis can rotate while driving in any direction, allowing for extremely efficient pathing.
Pros	<ul style="list-style-type: none"> • Omnidirectional • Strafing is powerful compared to other strafing drivetrains • Only 4 motors
Cons	<ul style="list-style-type: none"> • Limited speed and torque, especially on diagonals • Susceptible to getting pushed • Very little mounting space • Relatively heavy and complicated compared to other drive types

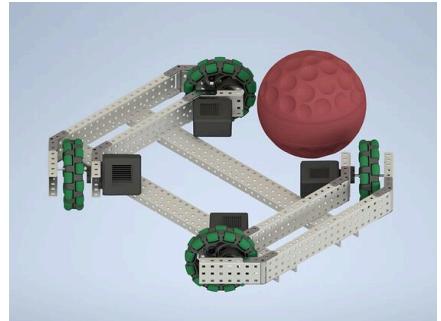


Image source: Alex Dickhans 2020



Image source: Team 81K 2020

H-Drive

Description	An H-Drive uses 5 Omni wheels arranged in an “H” pattern to move around the field. Like an X-Drive, each wheel is normally directly linked to a motor, however each side of the drive can also be linked together. This chassis can strafe by using the single wheel perpendicular to the others to push itself sideways. The singular perpendicular wheel offers little resistance to sideways pushing, making it a very rare chassis in competitive robotics.
Pros	<ul style="list-style-type: none"> ● High pushing power forward and backward ● More compact than other strafing designs
Cons	<ul style="list-style-type: none"> ● Low strafing power ● The center wheel can have different amounts of pressure on the ground due to field imperfections which can result in a loss of traction/cause the other wheels to be lifted off the ground. ● Takes 5 motors ● Center wheel prevents crossing most obstacles ● Strafing is inconsistent

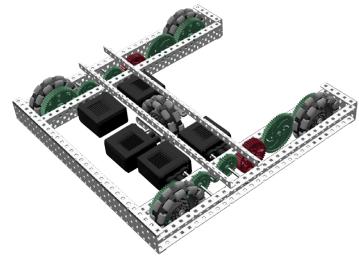
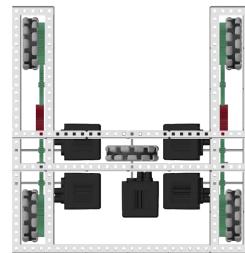


Image source: Carl Richter 2024

Tank Drive

Description	A tank drive is by far the most common type of chassis used in VEX. A tank drive uses an even number of wheels distributed between each side of the chassis. Its motion is limited to primarily forwards, backwards, and turning. Depending on the selected wheels, a tank drive can completely resist sideways motion or slip when pushed. A tank drive is very versatile for motor consumption; any even amount of 11W motors will work for this drive, and 5.5W motors can also be utilized to achieve a “5 motor” tank drive.
Pros	<ul style="list-style-type: none"> ● High pushing power forward and backward ● Very compact ● Fast speed ● Minimal frictional losses ● Preferred for driving ● Easy to build ● Lightweight ● Good obstacle performance
Cons	<ul style="list-style-type: none"> ● No strafing ● Harder to program

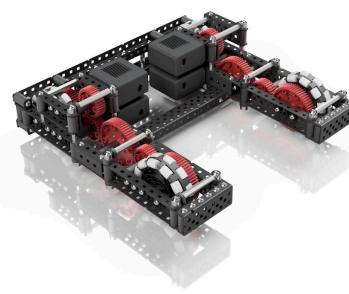


Image source: Xenon27 2022

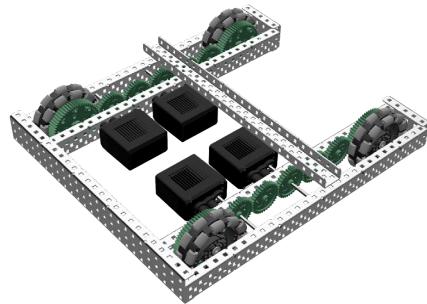


Image source: Carl Richter 2024

Mecanum Drive

Description	<p>A Mecanum Drive uses 4 specialized mecanum wheels with several rollers mounted at 45° angle to move around the field. These 4 wheels are mounted in a similar fashion to those of a tank drive, however the wheel gap needs to be much wider to accommodate the thickness of the mecanum wheels. The robot moves forwards, backward, and turns similarly to a tank drive. In order to strafe, the front and back wheels push towards or away from each other, as shown in the diagram below. This chassis also has the ability to move in any direction while turning but at a slower speed.</p>
Pros	<ul style="list-style-type: none"> ● Reasonable pushing power forward and backward ● Only 4 motors ● Easy to build ● Can cross small obstacles
Cons	<ul style="list-style-type: none"> ● Large frictional losses when turning and strafing ● Bulky wheels ● Slow

Image source: Carl Richter 2024

05/11/24 Background Research: Prior Chassis Analysis

Designed by: Carl, Matt

Witnessed by: Alex

Witnessed on: 05/11/24

Goal: Reflect on how previous chassis have performed to help reduce guesswork and bias from decision making.

2654P Tipping Point Robot 2 (Mecanum Drive)

This robot featured a 4 motor mecanum drive with an extremely low skirt to avoid driving over rings. The strafing ability made for easy programming and occasionally was useful for precise placement of the goals on the platform. This chassis proved to be decent at lower lever competition however it was incredibly susceptible to defense from tank drives, especially 6 motor ones. Another problem in this chassis came from the wheels being chained to each motor, which was a problem as the chains were prone to snapping. Furthermore, when manipulating game objects, the center of mass would shift, putting more stress on individual wheels/motors, causing the robot to move in unpredictable ways.

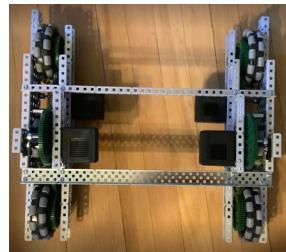


Key Points:

- 4 Motor Mecanum
- Ability to strafe
- Susceptible to defense

2654P Tipping Point Robots 3&4 (Tank Drive)

With the need for pushing power and speed, we switched to a 6 motor tank drive. 4" diameter omni wheels were chosen because of the assumption that they would be better at driving up the platform. By locking the middle wheels with screws (functionality a traction wheel), we were able to make the robot resistant from side pushes. The added speed and acceleration was tremendously helpful in increasing point scoring potential because the time driving in between tasks was much lower.

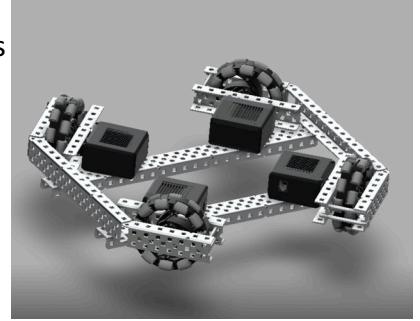


Key Points:

- 6 motor Tank Drive
- Resists sideways moving
- Strong against defense
- Fast

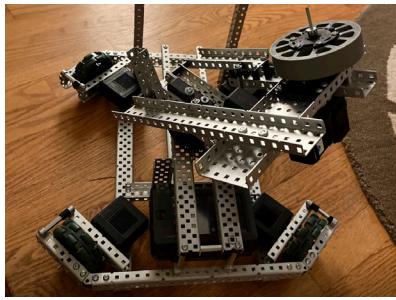
2654P Spin Up Robots 1&2 (X-Drive)

With Spin Up featuring so many game elements to be collected and one central target, we opted for an X-Drive and turret combination. The idea was to move around the field intaking discs and shooting them instantly through use of the turret. The entire concept was good in theory but proved to be too complicated to work well. The X-Drive severely reduced the robot's maneuverability/offensive and defensive capabilities, getting outperformed by all the competitive tank drives at our first tournament.



Key Points:

- 4 Motor X-Drive
- Ability to strafe
- Very susceptible to defense
- Slow and hard to drive



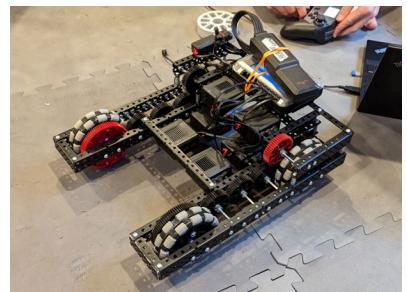
2654R Spin Up Robot 2 (Tank Drive)

With the strategy changing to more aggressive offensive and defensive driving especially in the “endgame” and to overcome some of the elements for the game we went for a tank drive. We chose to use four 4” omni wheels to overcome the short barriers around the goals. We also opted for a 6 motor drive to have enough power to be able to play effective defense during the match and to pin people in the corner during endgame. Due to the nature of this game we chose 343 rpm because we didn't need to go too fast, but we needed adequate torque for other parts of the game. This type of drive provided a good balance between speed and power and gave enough clearance for small game objects.



Key Points:

- 6 motor Tank Drive
- Strong against defense
- Fast
- Easy to drive

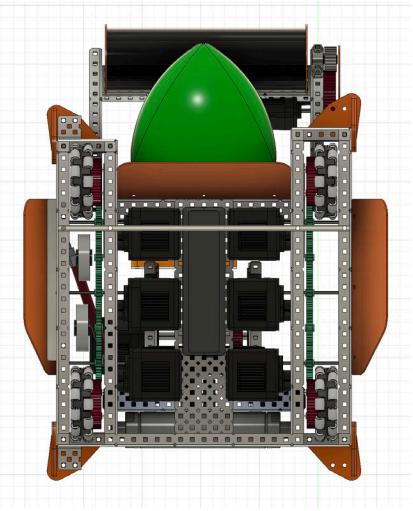
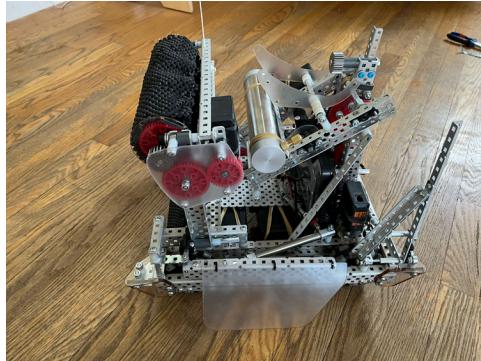


2654P Over Under Robot 5 (Tank Drive)

Over Under evolved from a primarily launching game early season to a game where shuttling one ball at a time was necessary. We utilized a 450 RPM 6 motor chassis to maneuver very quickly around the field, and a combination of 3.25 wheels and sleds to cross the barrier. To further aid barrier crossing, 2 more wheels were added on each side. This chassis performed very well and had a very small footprint while still having space for all necessary mechanisms. With the faster drive gearing, pushing power was limited, however, because the robot was light (around 12 lbs) we didn't have problems with burnout or acceleration. Additionally, this robot was also extremely effective in autonomous because its fast speed made it possible for it to interact with and score ALL possible game elements.

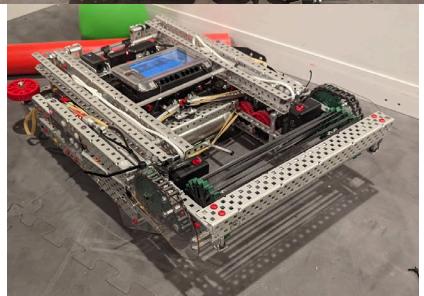
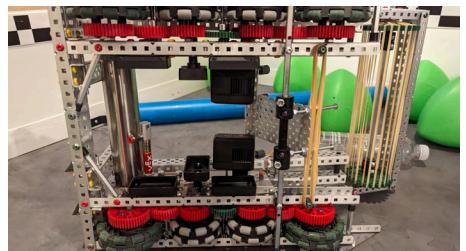
Key Points:

- 6 motor Tank Drive
- Strong against defense
- Very fast
- Light and compact



2654R Over Under Robot 4 (Tank Drive)

Over Under's strategy changed throughout the season from shooting to a high speed shuttling one ball game. This required different drives. At the end of the season when bowling and shuttling one ball was needed a faster drive was necessary. We decided to switch to a 7 motor, 450 RPM tank drive. This worked well with having the benefit of speed but also having enough power to push people around. We also had 4 wheels on each side which aided us in crossing the barrier. Although the pushing power and speed was good for this robot, in other games more motors might be necessary for other subsystems.



Key Points:

- 7 motor Tank Drive
- Very strong against defense
- Extremely fast
- Few motors for other mechanisms

(All photos are from us on our previous teams)

05/11/24

Background Research: Chassis Programming

Designed by: Alex

Witnessed by: Carl

Witnessed on: 05/11/24

Goal: Compare and discuss how different chassis options can be programmed differently and how this will affect us competitively.

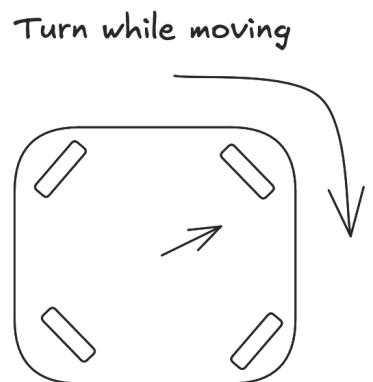
Accountability Notice: Diagram added late by Alex on 10/29/24

All the factors from the drivetrain impact how it's programmed. The main differentiating factors that determine how a robot is programmed is if it can strafe. Strafing is where the robot is able to make powered movements in more than one direction. The ability to strafe, in programming makes the algorithms to move the robot much less complicated because it makes the control mechanics much easier to control and to correct for small errors in the perpendicular angle of the robot.

Unique Movements with Strafing:

With strafing, the robot is able to do movements that are not possible with a tank drive or other non-strafing drives such as turning while moving to a point.

This allows the robot to take simpler paths while still maintaining the heading control necessary to complete the path.



Advanced Control with Tank Drive:

Even though tank drive can't strafe, it can still do complex motions that allow similar end pose control with proper programming. This requires a lot more programming and tuning, however it often means that robots can end up doing more during the same period, and having a tank drive is much more beneficial in matches.

Conclusion

- Strafing
 - Gives you many benefits for ease of coding complex movements
 - This doesn't always help though
- Tank Drive
 - Complex movements are still possible
- Balance
 - We must ensure that we maintain a balance of abilities of strafing and the other benefits of tank drive

05/12/24 Background Research: Chassis Decision Matrix

Designed by: Carl	Witnessed by: Alex	Witnessed on: 05/12/24
-------------------	--------------------	------------------------

Goal: Determine if there is a chassis that stands out from the rest or what options are worth further consideration.

In order to provide a complete judgment of the best chassis type, we chose to use a decision matrix to make sure that we took all categories into account. Each chassis is scored out of 10 points for each individual category. Each criteria has a multiplier to put emphasis on certain characteristics.

Decision Matrix:

Characteristics:	Tank Drive	X-Drive	H-Drive	Mecanum Drive	Multiplier
Speed	10	8	7	6	2
Pushing Power	10	5	8	6	1
Strafing	0	8	4	7	0.5
Agility	9	7	7	6	1
Controllability/Driver Preference	10	7	3	5	2
Mounting Space	9	3	6	7	1
Obstacle Performance	8	2	2	7	0.5
Programming	8	10	4	9	1
<i>Total Score</i>	<i>80</i>	<i>60</i>	<i>48</i>	<i>57</i>	

Note: All rankings are based on the research from previous pages and past experience.

Summary:

- Tank drive scored highest
- We decided to move forward with this design
- We are flexible and will continue to evaluate this on future robots

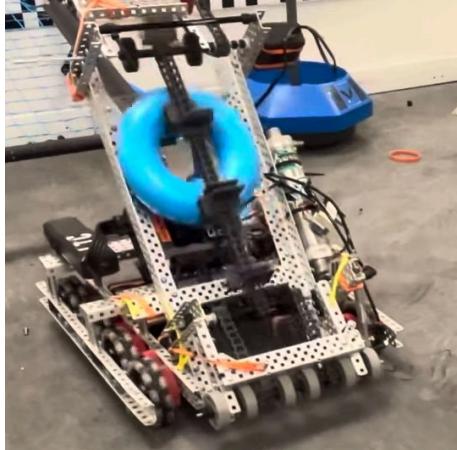
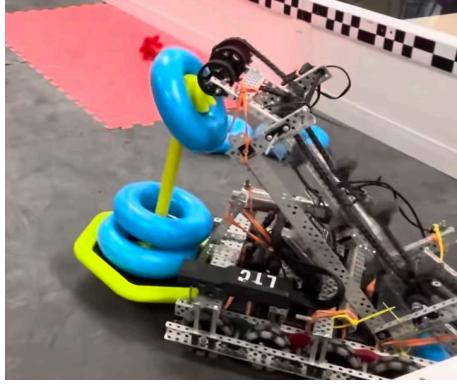
05/13/24

Background Research: Ring Manipulation

Designed by: Carl	Witnessed by: Alex	Witnessed on: 5/14/24
-------------------	--------------------	-----------------------

Goal: Maximize our understanding of how different ring holders and intakes perform and the differences between them.

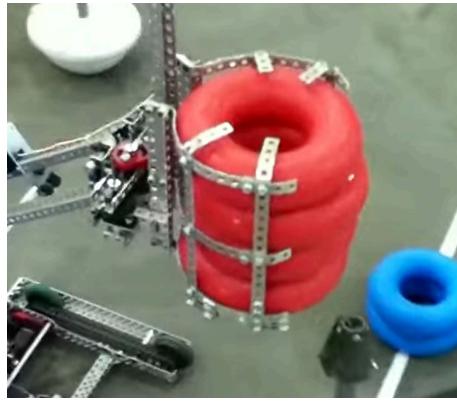
Loop/Belt Intake

Description	<p>This intake uses a loop made out of tank treads with flaps on it to interface with the center of the rings. After bringing the rings up to roughly 45° the rings are spat out on a mobile goal where they then fall onto the post. This design was very popular in Tipping Point due to its high efficiency and simplicity.</p>	
Pros	<ul style="list-style-type: none"> • High efficiency • Simple design • Easy to build • Easy to avoid possession limit 	
Cons	<ul style="list-style-type: none"> • Can ONLY score on mobile goals • No descoring capabilities • Rings require additional assistance to pass the rubber tip 	<p>(Photos from LTC Robotics on YouTube)</p>

Plunger Intake

Description	The “Plunger” intake design utilizes a narrow spike that is stabbed through the center of a ring. The mechanism then uses a piston to increase its diameter, meaning the rings can’t fall off the bottom.	
Pros	<ul style="list-style-type: none"> • Lightweight • Only 1 piston, no motor • Can score on all stakes when mounted on an appropriate lift • Much smaller than a claw 	
Cons	<ul style="list-style-type: none"> • Hard to line up with rings • Low efficiency for scoring on mobile goals compared to a loop intake. • Requires lift to work • Can’t descore 	

Claw

Description	A claw consists of two closing arms, commonly joined by meshing gears. Claws were by far the most used design in Round Up, and could very easily pick up large stacks of rings on the field and score them on a mobile goal.	
Pros	<ul style="list-style-type: none"> • Can descore • Can score on all stakes when mounted on an appropriate lift 	
Cons	<ul style="list-style-type: none"> • Limited to holding two rings by <SG6> • Can only grab rings one at a time • Releases the first ring when grabbing a second • Requires a lift • Requires precise driver control to accurately close the claw 	

05/14/24

Background Research: Lift Mechanisms

Designed by: Matt

Witnessed by: Carl

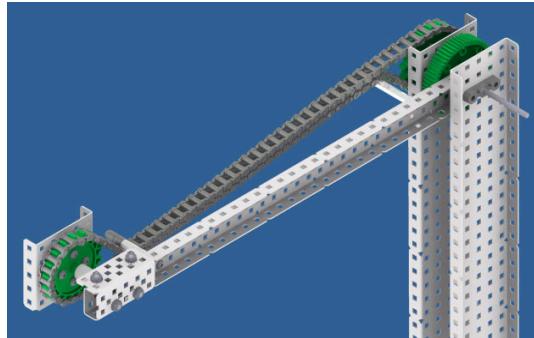
Witnessed on: 05/15/24

Goal: Maximize our understanding of how different lift systems perform and the differences between them.

4-Bar Lift

Description	A 4 bar lift mechanism includes 4 bars coming from the robot with generally the top one being fixed and the bottom one being powered by the motor. These four bars create a parallelogram, meaning the mechanism on the end will stay at the same orientation as the upright on the robot.	 <p>(Team 46B VEX Forums)</p>
Pros	<ul style="list-style-type: none"> ● High lifting power ● Can reach all stakes ● Simple ● Robust structure 	
Cons	<ul style="list-style-type: none"> ● Moves in an arc around the pivot point ● Requires more space than other lift mechanisms ● Only around 90° of movement 	

Chain Bar Lift

Description	A Chain Bar is a 2 bar mechanism that has a sprocket on both sides of the arm and is powered at the base, causing the arm to rotate. The sprocket on the upright is bolted to the upright so it remains stationary. The sprocket on the end of the arm is then forced to remain at a constant orientation, regardless of arm position, because it is linked to the other sprocket via chain.	
Pros	<ul style="list-style-type: none"> ● Over 180° of motion ● Compact ● Lightweight 	(Bobsalive VEX Forums)
Cons	<ul style="list-style-type: none"> ● Limited Load ● Hard to add band assist ● Chain is prone to snapping 	

Cascade Lift

Description	A cascade lift utilizes a long piece of chain or rope directed to a winch at the bottom. The rope is wrapped around pulleys on different stages of the lift, and when the rope is made shorter, the bottom of each stage is pulled closer to the top of the stage below it. Linear slides are used to keep the different stages moving directly vertically.	
Pros	<ul style="list-style-type: none"> ● Little horizontal space used ● Smooth motion ● Vertical motion 	
Cons	<ul style="list-style-type: none"> ● Complex ● Potential for friction ● Very heavy 	(Salim Benyoucef CAD Crowd)

05/14/24

Background Research: Motors & Pneumatics

Designed by: Carl

Witnessed by: Alex

Witnessed on: 05/14/24

Goal: Explore different types of pneumatics and look at some of their potential applications.

Motors

Motors are primarily used for applications that require large amounts of movement, continuous movement, or varying amounts of motion.

5.5W Motor



Pros:

- Compact design
- Light
- Good heat dissipation
- Allows for more appropriate power for each mechanism

Cons:

- Fixed internal gear reduction (200 RPM)
- Heavy and bulky to use two 5.5W motors instead of one 11W.
- Takes up more brain ports
- Only works with low strength shafts

11W Motor



Pros:

- Great power to size ratio
- Customisable RPM (100, 200, or 600)
- Easy to hot swap
- Only one brain port for 11W
- Works with both shaft sizes

Cons:

- Poor heat dissipation
- Hard to achieve exact desired power for a system (11W increments)
- Blocky and harder to fit than a 5.5W

(All images from vexrobotics.com)

Pistons

Pistons are typically used in applications that require simple linear motion. By nature, pneumatics can only provide 100% power or 0% power. The power outputted by a piston corresponds to the amount of pressure in the reservoir or pneumatic system. This means that the output power of the pistons will decrease as the solenoids and pistons are activated.

25mm Stroke 	Pros: <ul style="list-style-type: none"> • Very small • Great option for pin pulls • Lightweight • Low air consumption 	Cons: <ul style="list-style-type: none"> • Stroke is often too short to be useful • Smaller pistons require the same large pneumatic parts as longer pistons
50mm Stroke 	Pros: <ul style="list-style-type: none"> • Same size as the old pistons • Good balance of size and range • Easy to use and find space for 	Cons: <ul style="list-style-type: none"> • Not as easy to fit as a 25mm • More air than a 25mm • Less range of motion than a 75mm
75mm Stroke 	Pros: <ul style="list-style-type: none"> • Long actuation • Good for large mechanisms • Long range allows for better mechanical advantage 	Cons: <ul style="list-style-type: none"> • High air consumption • Hard to fit

(All images from vexrobotics.com)

05/14/24

Background Research: Feedback Loops

Designed by: Alex

Witnessed by: Matt

Witnessed on: 5/15/24

Goal: Research what feedback loops are and how they can be used in robotics programming to create more efficient motion.

What are Feedback Loops?

Feedback loops are loops that are commonly used in robotics and other applications to computationally control some output. A common household application of feedback loops is a heater control system. Heaters measure the temperature of the house using a single or group of thermometers, then use some feedback control, usually bang bang to determine if it should turn on the heater or AC to bring the temperature of the house to the desired level. In robotics, this is often applied to move an actuator, such as a robot arm to a desired setpoint. To define feedback better, they are loops that take a certain input, say a setpoint, and the current state of the system, generally from an encoder, and calculates a power percentage to apply to the motors to get them to the desired state.

What are Some Common Feedback Loops?

- PID
 - Proportional Integral Derivative
- Bang Bang
 - On/Off loops

For each of these feedback loops we will provide a short implementation in Rust to give a more solid idea.

Rust Trait Code:

In Rust you can define traits that can apply to different structs, depending on how you implement the code. We chose to make this trait to define all motion control and make these algorithms more portable with a template for the necessary functions to define. The trait code is right below.

```
pub trait MotionController {
    type Input;
    type Target;
    type Output;
    fn update(&mut self, target: Self::Target, state: Self::Input, dt: Duration) ->
    Self::Output;
}
```

PID Loop

A Proportional, Integral, Derivative (PID) feedback loop is one of the most commonly used in robotics. It is very versatile and can be used in a variety of applications, often complimenting a feedforward loop well. This loop has the following 3 main components, each multiplied by a tuning constant each loop update:

Equations:

$s(t)$ = desired position

$x(t)$ = current system position

$e(t)$ = error = $s(t) - x(t)$

$u(t)$ = control output

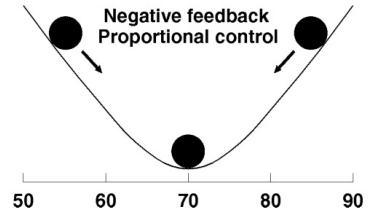
This is the formal equation often used to denote the control output given a error $e(t)$ input:

$$u(t) = e(t) * k_p + \int e(t)dt * k_i + e'(t) * k_d$$

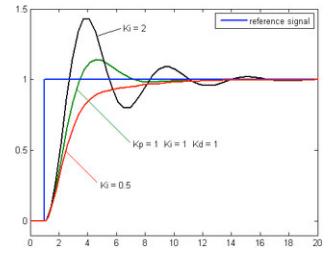
PID Loop Formal:

- Proportional:
 - $e(t) * k_p$
 - This part of the equation provides a centering control output onto the system
 - The photo on the left illustrates how the centering force affects the system.

- Integral:
 - $\int e(t)dt * k_i$
 - The integral is the term that can account for long term errors, as it gets larger over time with the smaller errors accumulating.
 - This is usually defined as the indefinite integral of the error, but it is usually accomplished by numerical integration each frame the PID is updated.
 - In this picture the red line shows what a purely integral loop would slowly correct the errors in the system.



(Photo from Fourmilab)



(Image from wikipedia)

- Derivative
 - $e'(t) * k_d$
 - The derivative component slows down sudden movements in the controller, especially as it gets close to the setpoint.
 - This is usually calculated by subtracting the current error from the previous error and dividing by time.

Code:

```
use core::time::Duration;
use crate::motion_controller::MotionController;

pub struct PidController {
    pub kp: f64,
    pub ki: f64,
    pub kd: f64,

    integral: f64,
    prev_error: f64,
}

impl MotionController for PidController {
    type Input = f64;
    type Target = f64;
    type Output = f64;

    fn update(&mut self, target: Self::Target, state: Self::Input, dt: Duration) -> Self::Output {
        // Calculate the error in the system
        let error = state - target;

        // Calculate the derivative
        let derivative = (error - self.prev_error) / dt.as_secs_f64();

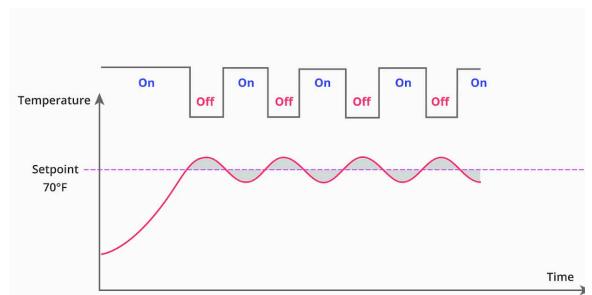
        // Accumulate the integral
        self.integral += error * dt.as_secs_f64();

        // Set the previous error
        self.prev_error = error;

        // Calculate the sum of the PID components
        error * self.kp + self.integral * self.ki + derivative * self.kd
    }
}
```

Bang Bang

Bang bang is one of the simplest methods for feedback control. It is often used in heaters because of the simple implementation. It has 2 distinct output states, usually on or off, and then it will do a simple comparison between the setpoint and the current position of the system. To the left is an illustration of how a heater would use bang bang control based on if it was greater than or less than a setpoint.



(Image from RealPars)

Code:

```
use core::time::Duration;
use crate::motion_controller::MotionController;

pub struct BangBangController {
    pub high_power: f64,
    pub low_power: f64,
}

impl MotionController for BangBangController {
    type Input = f64;
    type Target = f64;
    type Output = f64;

    fn update(&mut self, target: Self::Target, state: Self::Input, _: Duration) ->
    Self::Output {
        if state <= target {
            self.high_power
        } else {
            self.low_power
        }
    }
}
```

Conclusion

- Simple compared to PID Controller
- Comparison between target and current state
- Unstable- lacks fine control
- However, there are still use cases where Bang Bang is more than adequate

05/15/24

Background Research: Feedforward Loops

Designed by: Alex

Witnessed by: Matt

Witnessed on: 5/15/24

Goal: Research what feedforward loops are and how they can be used in robotics programming to create more efficient motion.

What are Feedforward Loops?

Feedforward loops are very different from feedback loops because they don't take into account the current state of the system. These loops take a desired position and calculate how the robot would have to apply power to get to the desired state. These loops often need a much more advanced knowledge of the robot's dynamics to work properly, however they can be extremely powerful in tandem with feedback loops to create robust and accurate operation of robot manipulators. However there is not a single type of feedforward control that is a catch all for robots. This is because with feedforward control you are calculating the control output necessary to move the manipulator at a certain speed. Therefore you have to have an advanced understanding of the dynamics of a system to be able to effectively calculate the feedforward constants properly. However there are a few fairly applicable feedforward loops that are very commonly used in robotics. These are static friction and velocity feedforward. We will go through those real quick, and then a more complex arm feedforward to give an example of how we design feedforward loops.

- Predict required input
- Use robot's system dynamics
- No catch all
- Very useful when you combine multiple feedforward loops and feedback loops

Static/Kinetic Friction Feedforward:

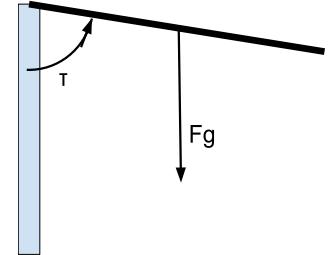
On any mechanism there is a small amount of static and kinetic friction. This friction resists motion, but it can be very easily characterized because it is very simple, and a linear dynamic. The friction on a system can be calculated by the normal force with this equation $F_k = \mu_k F_n$. However in most situations on the robot, the normal force is unknown, along with the friction coefficients, but you can still find a control input that is able to counteract this frictional force. This coefficient is often notated with kS and the control output is calculated like such $u(t) = sgn(v) * kS$ where v is the desired velocity of a system and $sgn(x)$ is the sign function.

Velocity Feedforward:

In Controls Engineering for FRC by Tyler Veness, it states that the first order model is appropriate for the impedance effect duration often seen in this size actuators. This makes velocity feedforward significantly easier to account for because the control of the motor is more directly proportional to the input voltage. This effectively means that the velocity of a system is roughly proportional to the control input, in volts on the system. This means that the control input $u(t)$ can be calculated with the equation $u(t) = v * kV$. Again, because much of the dynamics are difficult to model properly, it is often better to just add a kV tuning value to best work in the system.

Arm Feedforward:

Defining arm feedforward is more difficult than velocity or static friction because it depends much more on the system. To the left is a depiction of how the force of gravity exerts a torque on the arm. This is the torque that we are trying to resist with the feedforward calculations. The torque on the system because of gravity can be calculated with this equation $\tau = F_g \cdot r \cdot \cos(\theta)$ where theta is the angle from horizontal. The F_g and r can be simplified into a single constant kA that is again a tuned value to get a better value. This control loop could be accomplished with the following Rust code:



```
struct ArmFeedforward {
    ka: f64,
}

impl MotionController for ArmFeedforward {
    type Input = f64;
    type Target = f64;
    type Output = f64;

    fn update(&mut self, _: &Self::Target, state: &Self::Input, _: &Duration) ->
        Self::Output {
        state.cos() * self.ka
    }
}
```

What are the Limitations of Feedforward Loops?

Feedforward loops inherently cannot react to outside interference, if an arm gets out of position, or the drivetrain hits an object, a pure feedforward loop cannot react to this disturbance, however, feedforward loops can also be paired with feedback loops to provide extremely robust control. This also complements the feedback loops very well because it allows them to get closer to the control using less control input, making the resulting motion much smoother with less constant tuning.

05/15/24

Background Research: Wheel Options

Designed by: Carl	Witnessed by: Matt	Witnessed on: 5/19/24
-------------------	--------------------	-----------------------

Goal: Examine wheel options for tank drives and their different pros, cons, and use cases.

Accountability notice: Page completed late on 05/19/24 by Carl Richter

Traction Wheels		
2.75" Traction Wheel (old)	This wheel is very small and slightly narrower than the new version. This wheel lacks traction due to its small surface area and slick material. This wheel has no good attachment points for gears. There is almost no reason to use this wheel.	
2.75" Traction Wheel (new)	Improved version of the old wheel in almost all aspects including grip and attachment points, however this wheel is slightly thicker than the old version. This wheel can be used on drive bases with other 2.75" wheels, making it ideal for small drivetrains.	
3.25" Traction Wheel (old)	This wheel differs from the old traction wheels in the sense that it has attachment points for gears, making them significantly easier to use. These wheels also have a very unique grip pattern which makes them useful as offset wheels. No good use for these wheels as there are no obstacles on the field.	
3.25" Traction Wheel (new)	This wheel is a very good option for most drive bases. It has good grip due to its large surface area and plenty of attachment points for gears. Good option for a balanced drivetrain.	

4.00" Traction Wheel (old)	Along with having no mounting holes, this wheel also has almost no traction. Its large size also contributes to its undesirability. No good use for these wheels.	
4.00" Traction Wheel (new)	Similar to the other antistatic wheels, this wheel has plenty of grip and mounting holes, however it also takes up a lot of space. Limited applications for this wheel, primarily due to size.	
5.00" Traction Wheel (old)	This wheel has a lot of grip but still lacks mounting holes. This wheel is very hard to use because of its large diameter and reduced compatibility with other wheels.	

(All photos from vexrobotics.com)

Omnidirectional Wheels		
2.00" Omni Wheel (new)	The smallest variety of omni wheel that is currently competition legal, however it has very little load capacity. These characteristics make it more ideal for a tracking wheel than a drive wheel.	
2.75" Omni Wheel (old)	This wheel is fairly small, but also quite thick, requiring a 4 hole drive gap (including gear spacing). This wheel has good traction but no mounting points. Bulky rollers add a bit of bumpiness to the ride. Relatively low carrying capacity.	

2.75" Omni Wheel (new)	All around better version of the old wheel: Thinner (fits in 3 hole drive gap), smoother motion, and several mounting holes. Decent carrying capacity. Very good for making a compact drivetrain.	
3.25" Omni Wheel (old)	Good size, only 3.25" old omni to have mounting holes, decent grip, and durable. Thin enough for a 3 hole drive gap. Practical for compact drivetrains and has several compatible gear ratios.	
3.25" Omni Wheel (new)	Almost identical to the old 3.25" wheel with the addition of new mounting holes.	
4.125" Omni Wheel (old)	Exceptional grip, however still no mounting holes. Requires a four hole drive gap, and not compatible with most other 4" wheels because of its slightly larger diameter. Once again, too large to be used effectively.	
4.00" Omni Wheel (new)	Thinner version of the previous wheel with mounting holes. Good grip and high load capacity. Fits in a 3 hole drive gap, however still much too big to be used.	

(All new wheel images from vexrobotics.com, all old wheel images from kiwibots.co.nz)

05/17/24

Background Research: Gears vs. Chain

Designed by: Carl

Witnessed by: Alex

Witnessed on: 5/18/24

Goal: Explore the pros and cons of gears and chains.**Gears:****Pros:**

- Very durable/high reliability
- Low slop
- Slimmer than chain
- Low friction
- Multiple inputs to a single gear



(VEX Robotics.com)

Cons:

- Heavier
- Not space efficient over long distances
- More moving shafts than a chain drive



(Image from: cs2n.org)

Chains & Sprockets:**Pros:**

- Lightweight
- Better for long distances
- Chain can go through tight gaps
- Quick to build

**Cons:**

- High slop
- High friction
- Bulky
- Prone to snapping



(VEX Robotics.com)

05/22/24

Background Research: Programming Tools for VEX

Designed by: Alex

Witnessed by: Carl

Witnessed on: 05/23/24

Goal: Determine the advantages and disadvantages of different options for programming tools in VEX.

In VEX there are a variety of tools available to program robots. Some popular options are C++, and python, but there is also block code which is also supported by VEX, along with a C++ community supported option. There are also a couple Rust options that are community supported. To analyze these options we will do a bit of background research to determine how these different programming tools work and what their pros and cons are.

 <p>VEXCode C++ PRO</p>	<p>VEXCode C++ is the recommended way by VEX to program the VEX V5 robot. It uses VSCode as an editor with a special extension.</p>	
	<p>Advantages:</p> <ul style="list-style-type: none"> • Very quick • Most errors are caught at compile time • C++ has a strong type system • Quickest software updates for new devices 	<p>Disadvantages:</p> <ul style="list-style-type: none"> • API has very poor documentation • C++ is more difficult to learn
 <p>VEXCode Python</p>	<p>VEXCode Python is an easier to use code interface built on the micropython interpreter running on the V5 Brain that allows access to important C++ functions.</p>	
	<p>Advantages:</p> <ul style="list-style-type: none"> • Easy to program • Quick to implement solutions • Quickest API updates 	<p>Disadvantages:</p> <ul style="list-style-type: none"> • Very slow • Runtime errors • No type system • Python makes large projects very difficult to manage

 <p>Pros C++</p>	<p>PROS is the programming interface created and maintained by the Purdue Sigbots. This solution is very well documented and has a very nice API interface for programming.</p>	
	<p>Advantages:</p> <ul style="list-style-type: none"> • Easy to program • Strong type system • Most errors are caught at compile time • C++ has a strong type system 	<p>Disadvantages:</p> <ul style="list-style-type: none"> • C++ is more difficult to learn • Occasionally slow device API updating
<p>vex-rt (Rust, QUEEN)</p> 	<p>vex-rt is the solution that QUEEN built to program using PROS on VEX robots. It is currently a binding for the C pros API so it would not be ideal to use this solution.</p>	
	<p>Advantages:</p> <ul style="list-style-type: none"> • Easy to program • Strong type system • Almost all errors are caught at compile time • Rust has a very strong type system and great borrow checker 	<p>Disadvantages:</p> <ul style="list-style-type: none"> • Built off of PROS C API • Can't handle arrays • No async support • Larger binary size • Slowest API update cycles
<p>Vexide (Rust)</p> 	<p>Vexide is a community project to bring rust to the V5 system with a more sustainable and well done API. It does not rely on the pros C API in the same way that the vex-rt system does from QUEEN.</p>	
	<p>Advantages:</p> <ul style="list-style-type: none"> • Async executor • Strong type system • Almost all errors are caught at compile time • Rust has a very strong type system and great borrow checker • Second quickest API updates 	<p>Disadvantages:</p> <ul style="list-style-type: none"> • Very slow • Can't handle arrays • Little hardware testing

(All images from their respective websites)

Based on this analysis we chose to use the vexide programming interface with Rust because of the Rust language features and easier to use interface with the async features. The other options are better with hardware testing, but we are going to do a lot of our own hardware testing on vexide over the summer to ensure high competition reliability.

05/24/24

Background Research: Other Programming Tools

Designed by: Alex

Witnessed by: Carl

Witnessed on: 05/24/24

Goal: Find other useful programming tools that we can use to assist us while programming throughout the season.

For programming we need a few things to thrive this year with programming, mainly a good Integrated Development Environment (IDE) and a versioning tool.

IDE Analysis:

To thoroughly understand these options we will do research on a couple options that we can find. Additionally, switching code between IDE's is fairly easy so if we need to we might be revisiting this later in the season if we find that the IDE we are using is not working for us.

<p>Visual Studio Code</p> 	<p>Visual Studio Code, often shortened to VSCode is an open source editor maintained by Microsoft that provides a very simple and versatile editing experience. It is built off of the electron javascript interface, so it is slightly slower than native-built competitors. It is also an extremely easy to use and customizable IDE that can be used with a variety of languages in the very well built extension landscape.</p>	
<p>Advantages:</p> <ul style="list-style-type: none"> • Can be easily used with multiple languages. • Extensions make the IDE very customizable. • Free 	<p>Disadvantages:</p> <ul style="list-style-type: none"> • Slow interface with electron needing interpretation. • Not well suited for Rust or C++ out of the box. • Uses a lot of memory 	
<p>RustRover/CLion</p> 	<p>RustRover and CLion are IDE's made by IntelliJ to work specifically with Rust and C-based languages, respectively. These editors are not flexible, but are very well equipped for the languages that they are made for. For both CLion and RustRover they work very well with the respective build tools for their languages and are instead built on Java, which is much faster than electron and javascript like VSCode is, but it is very slightly slower than the native solutions such as Sublime text and Zed.</p>	
<p>Advantages:</p> <ul style="list-style-type: none"> • Very well suited for C++ or Rust • No need to install extensions 	<p>Disadvantages:</p> <ul style="list-style-type: none"> • Less extensions • Uses a lot of memory 	

	<ul style="list-style-type: none"> • Free student accounts 	<ul style="list-style-type: none"> • Overcomplicated
<u>Sublime Text</u> 	<p>Sublime Text is an editor that is built for speed and simplicity, which is built natively for each operating system which makes it faster than alternatives. This editor is designed very well and can work extremely well with C++ or Rust editing and linting.</p>	
Zed 	<p>Advantages:</p> <ul style="list-style-type: none"> • Very fast editor • Doesn't use much memory 	<p>Disadvantages:</p> <ul style="list-style-type: none"> • No free student accounts (\$100 license) • Extensions aren't extensive • Error squiggles take longer to set up
	<p>Zed is a newer code editor made in Rust, and is optimized for multi-threaded performance on newer computers. It is really well designed to work for a variety of languages with its quicker syntax trees that help with parsing significantly.</p> <p>Advantages:</p> <ul style="list-style-type: none"> • Doesn't use too much memory • Very performant • Very fast • Free • Well built for C++ and Rust 	<p>Disadvantages:</p> <ul style="list-style-type: none"> • Smaller community • Only built for mac, so Windows and Linux are much harder to use • Less extensions

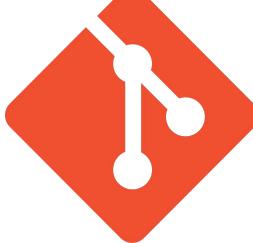
(All logos from their respective websites)

Based on our analysis from this table we chose to use Zed to start the season because of its very nice features and quick coding interface. However, if we find Zed is not working we will come back and possibly explore other IDE choices and reanalyze.

(Analysis of tools continues on next page)

Version Control Analysis:

For version control we want a solution that can keep track of our code seamlessly without much interaction from the programmer, but still have powerful tools such as branches to keep track of multiple different versions of the code at once. The 3 main public tools that are used in industry to do this are subversion, git, and mercurial. I already have extensive experience with git, but almost none with subversion or mercurial.

 <p><u>Git</u></p>	<p>Git is a tool made by Linus Torvalds to track different versions in his personal projects. It has grown to the biggest source tracking tool in the world now. Now a lot of online source control tools have been built on supporting Git as their main terminal interface. Git is what I have been using for the past 4 years to keep track of robotics sources on my past teams.</p>	
	<p>Advantages:</p> <ul style="list-style-type: none"> ● Very large user base so good resources ● Easy to use ● Lots of experience 	<p>Disadvantages:</p> <ul style="list-style-type: none"> ● Pull requests are initially hard to understand
 <p><u>Mercurial</u></p>	<p>Mercurial is a very similar source management tool to git. Mercurial doesn't allow users to change the history, unlike git which supports changes to any point in history. This gives it a small edge because previous things won't be changed.</p>	
	<p>Advantages:</p> <ul style="list-style-type: none"> ● Less complex ● Immutable history 	<p>Disadvantages:</p> <ul style="list-style-type: none"> ● No personal experience with Mercurial ● Smaller user base
 <p><u>Subversion</u></p>	<p>Subversion is a versioning tool made by apache which supports a more centralized architecture relative to git.</p>	
	<p>Advantages:</p> <ul style="list-style-type: none"> ● Less complex ● Only one centralized codebase 	<p>Disadvantages:</p> <ul style="list-style-type: none"> ● No personal experience with Subversion ● Smaller user base ● No branches

We chose to use git this year because of our experience with it in the past. Additionally it integrates well with GitHub which we also have extensive experience using. Additionally the branches and pull request features allow for very powerful source management.

Versioning Scheme:

For our repository we chose to use a Rolling Release (where we continuously commit and merge to main) scheme with a stable, well tested main branch to ensure reliable, flexible code throughout the season.

05/25/24

Background Research: Programming Tools

Setup

Designed by: Alex

Witnessed by: Carl

Witnessed on: 05/25/24

Goal: Set up the programming tools we will use this season.

Based on our prior chart we chose to use the vexide because of the Rust language being the best to maintain a large codebase. Also the async executor, lack of reliance on the PROS C API and quicker update times, gave it a large advantage over vex-rt from QUEEN. To get a project setup we needed to clone the [official example repository](#) on the vexide github. Using the following commands we then initialized a git repository in this folder using the following commands in that repository.

```
git init
git add -A
git commit -m "Initial Commit"
```

This creates a git repository that can track all of our code changes throughout the season. We then setup a repository on github using the github command line with all the default features:

```
gh repo create . --push --private --team alexDickhans -d "Team 2654E"
```

Now we have a repository that stores all our files online and will keep the data stored safely. To finish up setup we opened up the repository folder in Zed and made sure the autocomplete features were working as intended.

The screenshot shows the Zed IDE interface. On the left is a tree view of the project structure for 'Echo main'. The 'src' directory contains files like drivetrain.rs, rotation.rs, arm_feedforward.rs, bang_bang.rs, mod.rs, main.rs, motor_group.rs, and encoder.rs. Below 'src' are 'devices' (encoder.rs, mod.rs, motor.rs, motor_group.rs, rotation.rs), 'motion_controller' (arm_feedforward.rs, bang_bang.rs, mod.rs, pid.rs, drivetrain.rs), and 'main'. Other files shown include .cargo, .github, .idea, .vscode, target, envrc, .gitignore, arm7a-vexos-eabi.json, Cargo.lock, Cargo.toml, flake.nix, README.md, and rustfmt.toml. In the center, a code editor window displays the following Rust code:

```
src/main.rs: impl CompetitionRobot for Robot {
    type Error = Box<dyn Error>;
    async fn autonomous(&mut self) -> Result<(), Box<dyn Error>> {
        self.drivetrain.tank_drive(0.5, 0.5)?;
        sleep(Duration::from_secs(2)).await;
        self.drivetrain.tank_drive(0.0, 0.0)?;
        Ok(())
    }
    async fn driver(&mut self) -> Result<(), Box<dyn Error>> {
        loop {
            let drive = self.controller.left_stick.y().unwrap_or_default() as f64;
            let rotate = self.controller.right_stick.x().unwrap_or_default() as f64;
            self.drivetrain.arcade_drive(drive, rotate)?;
            sleep(Duration::from_millis(20)).await;
        }
    }
}
```

Below the code editor is a terminal window showing the output of a build command:

```
error[E0599]: no method named `cos` found for reference `&f64` in the current scope
--> src/motion_controller/arm_feedforward.rs:17:15
|
17 |         state.cos() * self.kf
|             ^^^ method not found in `&f64`
```

A warning message is also present:

```
warning: function cannot return without recursing
```

In the bottom right corner, a status bar displays: 'Installed `zed` to /usr/local/bin/zed. You can launch Zed from your terminal.'

05/26/24

Background Research: CAD Software Options

Designed by: Carl	Witnessed by: Alex	Witnessed on: 5/26/24
-------------------	--------------------	-----------------------

Goal: Explore features of different CAD softwares and how they are used in VEX.

Computer-aided design, or CAD for short, is a technology used in robotics to create, modify, and optimize robot designs. It allows users to make detailed 3D models of robotic components and systems. With CAD, we can simulate how mechanisms will move and fit together, helping to fix problems before building actual prototypes. This software allows us to optimize the design of all parts of the robot, allowing for smaller and lighter systems. When using CAD, we can also design polycarbonate pieces which can then be automatically arranged on a 2D plane, and exported as a DXF for cutting.

Program:	Advantages:	Disadvantages:
Fusion 360 	<ul style="list-style-type: none"> Intuitive user interface High support from the community Free to students High end collaboration features Good joint system for VEX parts 	<ul style="list-style-type: none"> Lack of some advanced features such as stress simulations Requires stable internet
Autodesk Inventor 	<ul style="list-style-type: none"> Highly customizable Advanced tools High end drawings/documentation 	<ul style="list-style-type: none"> Limited support from the VEX community Complicated user interface Requires stable internet Limited collaboration Resource intensive
Solidworks 	<ul style="list-style-type: none"> Industry standard Highly customizable Works well with most design programs 	<ul style="list-style-type: none"> Poor collaboration system Complicated UI Mates/joints are more time consuming Resource intensive High cost, hard to get education license
Onshape 	<ul style="list-style-type: none"> Free to students Browser based: easy to access on any device Good versioning system Simple user interface 	<ul style="list-style-type: none"> Very internet dependent Lacks simulation tools Some features not available in education edition

(All logos from respective websites)

05/28/24

Background Research: CAD Selection & Setup

Designed by: Carl

Witnessed by: Matt

Witnessed on: 05/29/24

Goal: Select a CAD program to utilize. Set up the program for optimal collaboration, organization, and design efficiency.

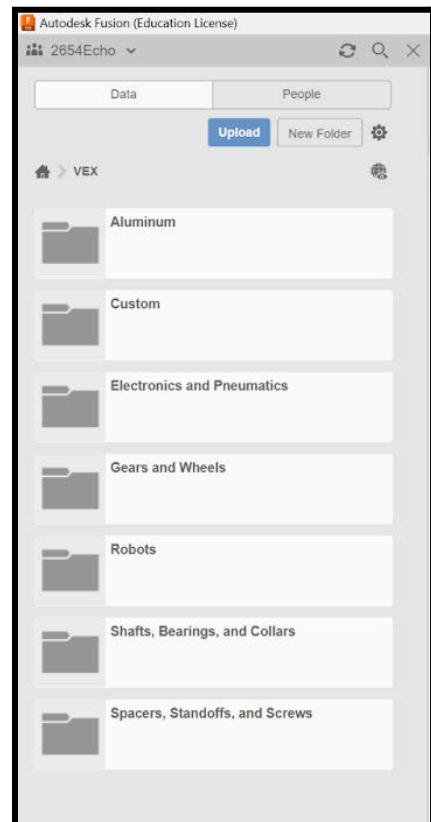
We elected to use Fusion 360 as our primary CAD software for this season, primarily because of its fantastic collaboration features and joint system. Additionally, like Onshape, Fusion's user interface is simple and easy to use, but still maintains most of the advanced features such as animations, technical drawings, and renderings.



(Image from autodesk.com)

Fusion Setup:

Prior to this season, we all had educational access to Fusion, allowing us to quickly set up a new team. To create robots, we will utilize parts from the [VRC Fusion Library v2.0.2 Release](#), available publicly on the VEX CAD Discord server. In previous years, we have simply directly imported the entire library into Fusion, keeping the organizational folders the same. Because this library includes almost every single VEX part, there are naturally a lot of different folders. This ended up creating several inefficiencies when selecting parts, largely due to the sheer amount of parts that we never used. To create a more cohesive workflow, we individually imported only parts that we use in our designs, allowing for significantly less clutter. For example, originally any particular wheel was nested within 4 folders, and can now be found in a singular folder. Furthermore, we chose to simplify some parts to save time and reduce load on our computers. This ranges from having a standardized insert for all wheels to making more common sizes of spacers and standoffs appear as default.



05/29/24 Design Numbering

Designed by: Carl	Witnessed by: Alex	Witnessed on: 05/29/24
-------------------	--------------------	------------------------

Goal: Lay out an easy to understand system to help when we are referring to a specific robot version.

Requirements:

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

Our Numbering System:

C.1.1.1

CAD or Physical Robot?

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

Rebuild/Redesign Number

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

Major Iterations

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

Minor Iterations

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

Example: R.2.1.6

This would represent our second (2) build (R), with no major changes (1), and five minor changes (6).

06/01/24 Background Research: Robot Localization: Ideology

Designed by: Alex	Witnessed by: Carl	Witnessed on: 06/02/24
-------------------	--------------------	------------------------

Goal: Define what localization is, and define some requirements and the difficulties in implementing a “perfect” localization solution.

What is Localization?

Localization is often defined as answering the question “Where am I?”. More formally this is described as a robot completing the process of determining where it is relative to the field. This is important in competitive robotics because this position knowledge can be used as feedback into different motion algorithms to allow the robot to more reliably interact with the environment, which in VEX is the field and game components. Localization is especially important when recovering from possible impacts from other robots during autonomous.

What Does Localization Currently Look Like in VEX?

Localization in VEX right now can almost always be categorized as Pose Tracking, where the robot knows where it started and continuously tracks the changes in pose to get to the end position. This is the setup referred to by “Odometry”, where the robot will use motor encoders from the drivetrain or external “tracking wheels” to get the position of the robot during the 0:15 autonomous period at the beginning of a match. This setup yields a very solid and consistent estimate of the robot’s position on the field, as long as the initial position of the robot is well defined. Many teams use this successfully, but it is susceptible to the following pitfalls and limitations.

Pitfalls of Pose Tracking:

- Can be messed up by variable field conditions including obstacles, antistatic spray, and different materials beneath the tiles
- Difficult to maintain a ground truth
 - Tuning and maintaining proper tuning properties for the diameter of odometry wheels so the field movements are scaled properly is time consuming
- Single point of failure
 - Odometry wheels add another single point of failure to the autonomous routine.
 - If you can take advantage of built in motor encoders, there is no added single point of failure and redundancy is increased because of the number of motors.

Limitations of Pose Tracking:

- Is not able to correct from minor errors in robot position
 - Initial robot position becomes extremely important because minor errors, especially in the heading can result in larger errors over time.
- Odometry wheels can slip
 - Slippage against the tiles can happen during higher acceleration motions limiting the possible motions with this solution
- Requires additional odometry wheels on the robot
 - Makes the footprint of the robot base larger, and cumbersome to fit
 - Odometry wheels will create another single point of failure in autonomous routines

Even though using Pose Tracking has the aforementioned issues, it is often enough to do complex motion control for VEX. However, our team wants to have extremely consistent and resilient autonomous and programming skills routines that exceed what is possible with traditional odometry solutions. To fully qualify what we want to accomplish with our localization solution we quantified it with the following goals:

- Global Localization
 - We want to have global localization, where the robot knows its precise location relative to the field, not its starting position
 - Relative to the field is the most useful capability, that's the goal of pose tracking odom, but it is less successful at knowing an accurate location on the field
 - Allows greater uncertainty and flexibility in the robot's starting position
 - Small errors, as small as 3 degrees can cause major issues with pose tracking at farther distances
- High Accuracy
 - We want the $E_{localization} = ||x_{actual} - x_{predicted}|| \leq 1 \text{ in}$
 - High accuracy will allow our robot to be able to grab the game objects and move around the field with a very high consistency and confidence, allowing our motion algorithms to go faster and more accurately.
- High refresh frequency
 - We want a refresh cycle where $f_{localization} \geq 100 \text{ Hz}$
 - A high refresh cycle, matching that of the motors will allow for the highest fidelity control of the robot

What Sensors are Available for Localization?

Localization is primarily made up of a sensing system and a processing system. Sensor-wise we can use a variety of sensors to detect various different parts of the field. For example you can use a line sensor to detect the lines on the field when you drive over them. To analyze our sensor options we wanted to list out all the options we have and how they could be used to determine our location on the field. After we choose which sensors we use, based on experiments using the sensors, we will design an algorithm that will best determine the robot's location from this data.

Sensor Options:

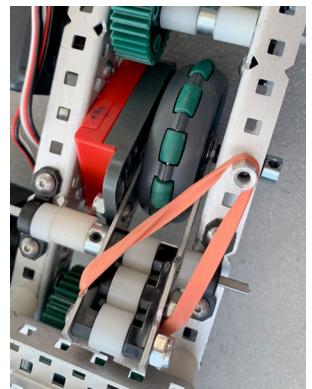
- Line sensor
 - A line sensor sends out an infrared signal and then measures the amount of infrared that is reflected.
 - This can be used, among other things, to detect when the sensor is directly above a line.
- GPS sensor
 - GPS sensor uses the “GPS” strips on the side of all the fields as a visual fiducial to determine an estimated pose for the robot.
 - This calculation of the position of the robot is done on board the sensor, so we don't have much control over the reliability of the position, however, it also uses little processing power on the brain to add this sensor.
- AI vision sensor/Vision sensor
 - The AI vision sensor could be used to find the location of objects on the field and localize the robot relative to those objects
 - These objects would usually be the climbing structure or the neutral posts
- ToF LIDAR Sensor
 - Using a time of flight lidar paired with a map of the field to attempt to localize the robot by detecting when it bounces off the walls
 - LIDAR Sensors measure the time that a light(laser) chirp takes to get to an object and back.
 - Using this solution would require us to use a solution like a particle filter that can properly account for the probability solution. This solution is more expensive so that will have to be taken into consideration.
- Inertial Measurement Unit(IMU)
 - An IMU, or inertial measurement unit detects the change in orientation.
 - This can be used to get a reliable source for the orientation of the robot.



- Rotation sensor or Encoder (Odometry wheel)
 - Using the rotation sensor or the built in motor encoders to determine how far a wheel has moved can help with position tracking.
 - This pose tracking can still be very useful for global localization because determining the small changes in distance in between global measurements are important.

(All photos from the VEX.com website except for the odometry wheel)

(Odometry wheel picture from Ryan from 4253B on the VEX forums)



Sensor Analysis:

All of these sensors are ways that you can estimate an individual axis or location source of the robot. However, some sensors, such as line sensors are not useful on their own, because they are only applicable in certain places. Others, such as odometry wheels will always get you a position, however, as previously mentioned they have consistent errors that can accumulate. Ideally we would create a solution that would be able to incorporate an arbitrary number of sensors into our localization solution, so we can make the most use of the sensors available to us. However, incorporating all the sensors is difficult to do because each sensor has its own individual uncertainty that has to be taken into account. Additionally, incorporating more sensors adds more computational complexity, especially in memory constraints on the brain. To evaluate what solutions we can use we will investigate various possible solutions to find where the robot is. Here is the list that we are looking at. We will investigate this more below in the methods research section.

ROBOT 1

06/02/24 Time Management: Summer Timeline

Designed by: Matt

Witnessed by: Carl

Witnessed on: 06/03/24

Goal: Create a general idea of how our time will be spent over the summer in regards to robotics.

This Gantt chart outlines a general plan for the summer at a fairly slow pace as we have summer jobs, and our time for robotics is limited. To help us stay on track, we have created smaller goals for each task:

Identify:

- N/A (already completed)

Brainstorm:

- Brainstorm 3 potential robot designs
- Select the best option using a decision matrix

Task	Summer Schedule													
	June				July				August					
	Week 1	Week 2	Week 3	Week 4		Week 1	Week 2	Week 3	Week 4		Week 1	Week 2	Week 3	Week 4
Identify														
Brainstorm														
Design														
Build														
Test/ Improve														
Program														
Drive														
Competition														
Competition Analysis														

Design:

- Perform multiple design iterations until we get to a solid design
- If a solid design can not be made, restart the design cycle
- Begin programming any complex features for this robot

Build:

- Build the robot from the CAD model
- Program the robot for testing

Test/ Improve:

- Test the robot on a field
- Use the design process to solve minor problems with different subsystems

Competition Analysis:

- Minnesota Signature Event strategic and design analysis

Program:

- Create and fully test multiple autonomous routines (including skills)

Drive:

- Practice filling up wall stakes and mobile goals
- Practice matches with other teams
- Practice driving skills

06/03/24 Brainstorming: Initial Strategy/Potential Designs

Designed by: Carl	Witnessed by: Alex	Witnessed on: 06/04/24
-------------------	--------------------	------------------------

Goal: Identify different strategy options and how they are limited by physical robot constraints.

As of right now, we see three different ways of playing the game. Our first option is to make a more advanced ring scoring mechanism that is capable of scoring on all stakes. This would allow us to focus on filling up the wall stakes early on, meaning we won't be at the risk of any huge descores. Towards the last 30 seconds of the match, we could start filling up a mobile goal, which we would then be able to place in the positive corner and protect for the rest of the match. The strategy would not utilize any form of hanging mechanism, which would simplify robot design significantly, and avoid any risk of descoring our mobile goal. This form of ring scoring mechanism would likely take longer to develop than the first option, but with ample time we feel that the benefit would be worth it. This strategy also has a strong correlation to some successful [Turning Point Strategies](#) previously discussed.

The second way utilizes a conveyor belt intake and a tipping point style mobile goal holder. This would allow us to very efficiently collect rings and fill up multiple goals per match. This robot design would also be very simple and allow us to have some form of high hanging mechanism for even more points, which would be extremely helpful in skills. The main disadvantage of the strategy is the fact that unattended mobile goals full of our rings can easily be pushed into the subtraction corner, which would create a 16 point swing. Towards the end of the match, in order to hang, we would need to leave our fully filled mobile goal unprotected in the positive corner, and if that goal was moved to the negative corner, it would be a 24 point loss, which is double the maximum amount of points for a tier 3 elevation. Furthermore, this type of robot design would make it very difficult or impossible to score on the wall mounted stakes because they are much higher than the mobile goal posts.

The third option combines the two previous strategies/designs into a more robot capable of completing the most tasks at the cost of lower efficiency at individual tasks. This would make us versatile and compatible with almost all types of partners and opponents. Descoring would also be a big element of this strategy as it will likely be among the fastest ways to get a huge point swing. As a result of the compromises necessary to achieve high versatility, we will inherently be slower at scoring overall which would be fairly detrimental to our skills score.

06/04/24

Identify: Robot 1 Design Considerations

Designed by: Carl	Witnessed by: Alex	Witnessed on: 06/05/24
-------------------	--------------------	------------------------

Goal: Compile a comprehensive list of features and mechanisms necessary for a robot to accomplish strategy #1.

Drivetrain:

The mobile base of our robot is the primary subsystem of our robot, and thus requires the most motors. The more the better. Because a tank drive is significantly better than any other drive as [previously mentioned](#) (Pg. 54), we will want to use it for this robot. A faster drive base has significantly better offensive capabilities than a slower one, but because pushing power and torque are inversely proportional, there is a limit to how fast it can be. We can use characteristics from the previous [robot's drivetrains](#) (Pg. 50-52) such as wheel size, number of motors, gear ratio, and robot weight to predict what drive configurations will be viable for this robot.

Mobile Goal Holder:

This system should be able to hold a mobile goal in such a way that all 6 rings can fit on it and should be able to grab from any direction. The goal holder will benefit from being extremely secure as losing the goal could be catastrophic. The goal will need to be held in such a way that it does not interfere with the loading of wall stakes, while simultaneously not requiring the loading mechanism to exceed the footprint of the robot due to <SG2>/<R5>. The goal clamp can likely be powered by pistons in a similar fashion as Tipping Point robots.

Intake:

In almost all cases, an intake is far more efficient than a claw or plunger at getting rings off of the floor and into the robot's control, however the intake is also much heavier than a plunger or claw, so putting it on the end of a tall lift would not be practical.

Power Distribution:

Obviously, we are limited to 88W for our motors by <R13>, and 2 pneumatic reservoirs with up to 100 PSI by <R23>. After that, we can use any combination of 5.5W and 11W motors, along with unlimited pistons. In order to fully maximize our motors we will first come up with a general design of the robot, allowing us to perform calculations to determine the minimum torque necessary for each mechanism. Additionally, by analyzing what mechanisms

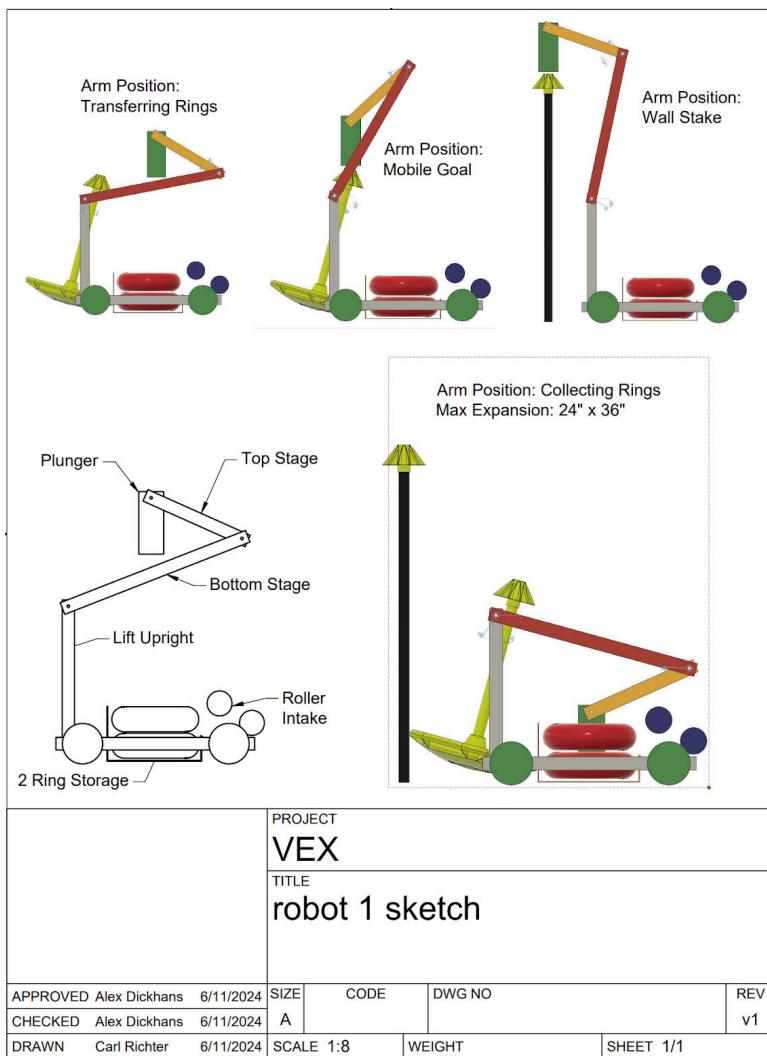
06/11/24

Brainstorming: Concept 1 Drawing

Designed by: Carl, Alex

Witnessed by: Matt

Witnessed on: 06/11/24

Goal: Conceptualize a robot to meet our design requirements.

Previous research on different subsystems:

- [Intake/ring holders](#) (Pg. 55-56)
- [Lift systems](#) (Pg. 57-58)

Explanation:

This robot features a two stage lift with a plunger mounted at the end. To minimize time spent collecting rings, we also have an intake on the front that leads to a central holding area where the plunger can grab the rings from a consistent location. We chose to use chain bars for both stages because of the wide range of positions the plunger needs to be able to get to. We chose a plunger over other claw or intake designs because of its lightweight and vertical grabbing. 6 motor drive may be possible but further calculations about the lift are needed first.

Design Matrix Scores		
Mobile Stake Scoring Speed	7/10	Concept has an intermediate step slowing down scoring
Mobile Stake Security	9/10	While scoring it always maintains control of the goal
Wall Stake Scoring Speed	8/10	This concept prioritizes wall stakes speed
Descore Speed	0/10	This robot concept cannot descore
Descore Safety	0/10	This robot concept cannot descore
Robot Simplicity	3/10	Many moving parts
Robot Weight	5/10	Very complex -high weight

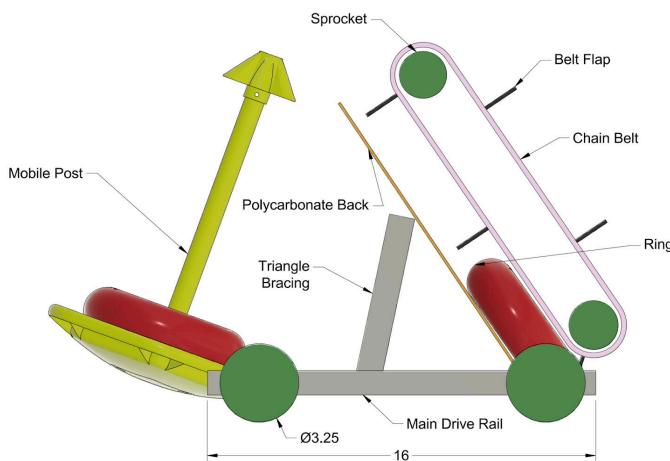
06/12/24 Brainstorming: Concept 2 Drawing

Designed by: Carl, Alex

Witnessed by: Matt

Witnessed on: 06/12/24

Goal: Conceptualize a robot to go along with strategy #2.



Explanation:

This design is very simple in the sense that it has very few moving components and only has a few functions. The benefit of this design would be robustness, quick construction time, and extremely high efficiency at scoring onto mobile posts. We could easily accommodate a 6 motor drive and 1 motor intake while still having 11W left over for other subsystems. The main downside of this robot is its lack of ability to hang or score on wall stakes.

Design Matrix Scores		
Mobile Stake Scoring Speed	9/10	Fast, efficient path for rings
Mobile Stake Security	9/10	Goal clamp secures goal while moving and scoring
Wall Stake Scoring Speed	0/10	This robot does not score on wall stakes
Descore Speed	0/10	This robot does not have the ability to descore
Descore Safety	0/10	This robot does not have the ability to descore
Robot Simplicity	9/10	This design is very simple with relatively few high-risk single points of failure
Robot Weight	10/10	This Simplicity leads to having a very lightweight robot

Previous research on different subsystems:

- [Intake/ring holders](#) (Pg. 55-56)
- [Past Drivetrains](#) (Pg. 50-52)

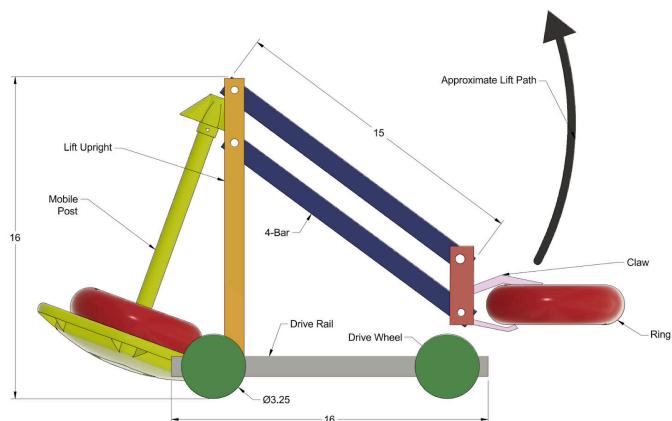
06/12/24 Brainstorming: Concept 3 Drawing

Designed by: Carl, Alex

Witnessed by: Matt

Witnessed on: 06/12/24

Goal: Conceptualize a robot to go along with strategy #3.



Explanation:

This robot is very simple in concept, using one four bar lift with a claw on the end to lift one ring at a time onto any type of post. With a claw, we think we will be able to grab scored rings at any position on a goal and lift it up along with all of the rings above it. This concept only uses 7 motors assuming 6 motor drive, 1 motor lift, and pneumatic claw and goal holder, so it could be combined with concept #2 in some capacity to create a more functional robot. This integration would come at the cost of additional complexity and weight.

Previous research on different subsystems:

- [Intake/ring holders](#) (Pg. 55-56)
- [Lift systems](#) (Pg. 57-58)

Design Matrix Scores		
Mobile Stake Scoring Speed	4/10	Lots of movement from the arm to complete scoring
Mobile Stake Security	2/10	Goal clamp secures goal while moving and scoring
Wall Stake Scoring Speed	3/10	This robot has to pick up rings one at a time to score on the wall stakes
Descore Speed	8/10	This concept is able to descore from the goal effectively with its movable claw
Descore Safety	8/10	This concept is able to descore safely because it retains control of rings throughout the entire descore process
Robot Simplicity	9/10	Almost no high-risk single points of failure in this concept and few moving parts
Robot Weight	9/10	This design is very simple, therefore it leads to a low robot weight

06/14/24

Select Solution: C.1.1 Concept Decision

Designed by: Alex, Carl	Witnessed by: Matt	Witnessed on: 06/15/24
-------------------------	--------------------	------------------------

Goal: Identify the robot concept we want to move forward with for the first robot design.

Through our brainstorming we came up with 3 initial design concepts for the first robot design, a two stage pick-and-place arm concept, a ring conveyor belt intake concept, and a simple claw concept. While all of these designs have very strong merits, each one also has disadvantages that we must acknowledge and consider when choosing the best robot design to explore over the summer. To properly do this analysis a complete design matrix is necessary. We will score the robot concepts on the following criteria out of 10 based on our best judgment, which is also justified in each of the above sections about brainstorming the robot designs. If the robot is incapable of completing any of the actions, it will be given a score of 0 in that criteria and it will be noted.

Criteria	Description	Weight
Mobile Stake Scoring Speed	We will judge this based on how fast we think this design will be able to consistently score on the Mobile Goal stakes during a match.	4
Mobile Stake Security	We will judge this criteria based on how secure the goal will be both while we score and when we are moving the goal around the field.	4
Wall Stake Scoring Speed	We will judge this based on how fast this robot will be able to score the wall stakes relative to how we think the other concepts will.	6
Descoring Speed	Descoring is critical in this game and doing it quick enough to be an adequate strategy is an important part of each concept's scoring.	3
Descoring Safety	If Rings that we descore go out of the field we can receive violations for that so ensuring all descored rings stay inside the field is crucial.	5
Robot Simplicity	Robot simplicity is always important because it reduces the number of points of failure and the significance of them, which often makes a much more competitive robot because reliability is key to being consistent in matches.	6
	In this category we will judge how heavy we think the robot	3

Robot Weight	design will be, lighter is better and therefore a higher score. However we don't think a heavier robot will be too much of a detriment because of the necessity to have more complex designs this year.	
--------------	---	--

Based on these weights and our prior scoring analysis we made a [decision matrix](#) (Pg. 7) to weigh our decision:

		Concept 1		Concept 2		Concept 3	
Criteria:	Weight	Score	Weighted Score	Score	Weighted Score	Score	Weighted Score
Goal Stake Scoring Speed	4	7	28	9	36	4	16
Goal Stake Security	6	9	54	9	54	2	12
Wall Stake Scoring Speed	6	8	48	0	0	3	18
Descoring Speed	2	0	0	0	0	8	16
Descoring Safety	5	0	0	0	0	8	40
Robot Simplicity	2	3	6	9	18	9	18
Robot Weight	2	5	10	10	20	9	18
Total Score:			146		128		138

In our decision matrix, Concept 1 (the dual lift mechanism) was identified as the best solution, receiving the highest weighted score of 146. The other concepts, Concept 2 and Concept 3, received scores of 128 and 138 respectively. To fully design this concept we must do some calculations to ensure this robot concept is fully feasible. Primarily we must do some calculations on the power requirements of the lift mechanism to ensure the robot can be created within the motor limits of <R13> from the game manual. Additionally we must fully CAD out the robot to ensure no parts will collide with other parts of the robot during scoring motions.

06/18/24

Brainstorming: Lift System Calculations

Designed by: Carl

Witnessed by: Alex

Witnessed on: 06/19/24

Goal: Identify the speed and torque requirements for the lift based on the concept CAD.
Calculate the minimum amount of power for each stage.

In order to create an accurate CAD model that can function effectively when built, we decided to calculate the necessary gear ratio and motor power for each stage of the lift. For this design to be most competitive, we will need to score a minimum of two rings every two seconds. For this to happen, the arm would need to move between positions in around 0.7 seconds (assuming 0.5 seconds to get the rings onto a stake).

Output torque of a 5.5W motor:

$$\omega = RPM_{motor} \cdot \frac{2\pi}{60} = \frac{200rev}{min} \cdot \frac{min}{60sec} \cdot \frac{2\pi rad}{rev} = \frac{21rad}{sec}$$

$$\tau_{5.5W} = \frac{P}{\omega} = \frac{5.5kgm^2}{sec^3} \cdot \frac{sec}{21rad} = 0.26Nm$$

where $\tau_{5.5W}$ is the output torque of a 5.5, P is power, and ω is angular velocity

Top Stage:

Givens/Ideal Values (for placing rings on a mobile goal):

$$gravity (g) = 9.8m/sec^2$$

$$plunger mass (m_p) \approx 1.3lb \rightarrow 0.6kg$$

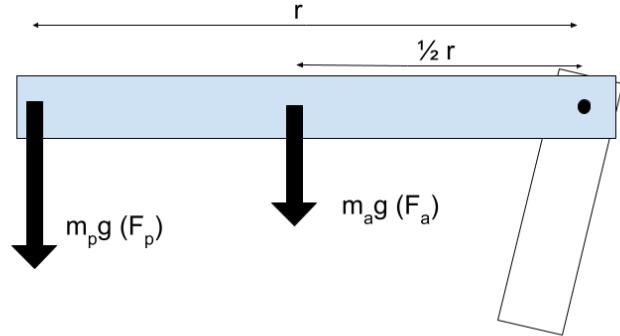
$$arm mass (m_a) \approx 0.7lb \rightarrow 0.3kg$$

$$arm length (r) = 8.0in \rightarrow 0.2m$$

$$desired time for motion (t) = 0.7sec$$

$$total angle adjustment (\theta) \approx 100^\circ$$

$$desired accel and decel time (t_{accel}/t_{decel}) = 0.2sec$$



Target maximum angular velocity (how fast the arm should rotate based on the desired time for motion):

$$\theta_{rad} = \theta_{deg} \cdot \frac{\pi}{180} = 100^\circ \cdot \frac{\pi}{180^\circ} = 1.75rad$$

$$\theta_{rad} = \overline{\omega} t$$

$$\theta_{rad} = \frac{1}{2}\omega_{max} t_{accel} + \omega_{max} t_{constant} + \frac{1}{2}\omega_{max} t_{decel}$$

$$1.75rad = \frac{1}{2}\omega_{max} \cdot 0.2sec + \omega_{max} \cdot 0.3sec + \frac{1}{2}\omega_{max} \cdot 0.2sec = 0.5sec \cdot \omega_{max}$$

$$\omega_{max} = \frac{1.75rad}{0.5sec} = \frac{3.5rad}{sec}$$

Target RPM (conversion of units from previous calculation):

$$RPM_{target} = \omega \left(\frac{60\text{sec}}{\text{min}} \cdot \frac{\text{rev}}{2\pi\text{rad}} \right) = \frac{3.5\text{rad}}{\text{sec}} \cdot \frac{60\text{sec}}{\text{min}} \cdot \frac{\text{rev}}{2\pi\text{rad}} = \frac{33\text{rev}}{\text{min}}$$

Max static torque (the highest amount of torque that the weight of the arm transfers to the pivot):

$$F_a = m_a g = 0.3\text{kg} \cdot \frac{9.8\text{m}}{\text{sec}^2} = 2.94\text{N}$$

$$\tau_a = r \perp F_a \rightarrow \tau_a = \frac{r}{2} \cdot F_a = \frac{0.2\text{m}}{2} \cdot 2.94\text{N} = 0.29\text{Nm}$$

$$F_p = m_p g = 0.6\text{kg} \cdot \frac{9.8\text{m}}{\text{sec}^2} = 5.88\text{N}$$

$$\tau_p = r \perp F_p = 0.2\text{m} \cdot 5.88\text{N} = 1.18\text{Nm}$$

$$\tau_{static} = \tau_a + \tau_p = 0.29\text{Nm} + 1.18\text{Nm} = 1.47\text{Nm}$$

where F is the force of gravity pushing down, τ_a is the torque exerted onto the pivot by the arm itself, and τ_p is the torque exerted onto the pivot by the plunger

Moment of inertia (the arms resistance to being accelerated):

$$I_a = \frac{1}{3}m_a r^2 = \frac{1}{3} \cdot \frac{0.3\text{kg}}{1} \cdot \frac{0.2^2\text{m}^2}{1} = 0.004\text{kgm}^2$$

$$I_p = \frac{1}{3}m_p r^2 = \frac{0.6\text{kg}}{1} \cdot \frac{0.2^2\text{m}^2}{1} = 0.024\text{kgm}^2$$

$$I_{total} = I_a + I_p = 0.004\text{kgm}^2 + 0.024\text{kgm}^2 = 0.028\text{kgm}^2$$

where I_a and I_p are the moments of inertia for the plunger and arm respectively

Torque for arm acceleration (torque needed to accelerate the arm to full speed in 0.2 sec WITHOUT gravity):

$$\alpha = \frac{\Delta\omega}{\Delta t} = \frac{3.5\text{rad}}{\text{sec}} \cdot \frac{1}{0.2\text{sec}} = \frac{17.5\text{rad}}{\text{sec}^2}$$

$$\tau_{accel} = I\alpha = \frac{0.028\text{kgm}^2}{1} \cdot \frac{17.5\text{rad}}{\text{sec}^2} = 0.49\text{Nm}$$

where α is angular acceleration

Total torque necessary (torque needed to accelerate the arm to full speed in 0.2 sec WITH gravity):

$$\tau_{total} = \tau_{accel} + \tau_{static} = 0.49\text{Nm} + 1.47\text{Nm} = 1.96\text{Nm}$$

Gear ratio needed to produce necessary torque (with a 5.5W motor):

$$gear\ ratio = \frac{\tau_{5.5W}}{\tau_{total}} \cdot \frac{0.26\text{Nm}}{1.96\text{Nm}} = \frac{1}{7.5}$$

Max angular velocity (how fast the arm spins with the gear ratio above):

$$\omega_{max} = RPM_{motor} \cdot gear\ ratio \cdot 360^\circ = \frac{200\text{rev}}{\text{min}} \cdot \frac{1}{7.5} \cdot \frac{360^\circ}{\text{rev}} \cdot \frac{\text{min}}{60\text{sec}} = \frac{160^\circ}{\text{sec}}$$

Degrees traveled during acceleration and deceleration phases of the arms movement:

$$\alpha = \frac{\omega_{max}}{t_{accel}} = \frac{160^\circ}{sec} \cdot \frac{1}{0.2sec} = \frac{800^\circ}{sec^2}$$

$$\theta_{accel} = \frac{1}{2}\alpha t^2 = \frac{1}{2} \cdot \frac{800^\circ}{sec^2} \cdot \frac{0.2^2 sec^2}{1} = 16^\circ$$

$$\theta_{decel} = \theta_{accel} = 16^\circ$$

Time for arm to move 100° (how much the arm would need to rotate to score on a mobile stake):

$$\theta_{constant} = \theta - \theta_{decel} - \theta_{accel} = 100^\circ - 16^\circ - 16^\circ = 68^\circ$$

$$t_{constant} = \frac{\theta_{constant}}{\omega_{max}} = \frac{68^\circ}{1} \cdot \frac{sec}{160^\circ} = 0.425sec$$

$$t_{total} = t_{constant} + t_{accel} + t_{decel} = 0.425sec + 0.2sec + 0.2sec = 0.8sec$$

Time for arm to move 140° (for wall stake):

$$\theta_{constant} = \theta - \theta_{decel} - \theta_{accel} = 140^\circ - 16^\circ - 16^\circ = 108^\circ$$

$$t_{constant} = \frac{\theta_{constant}}{\omega_{max}} = \frac{108^\circ}{1} \cdot \frac{sec}{160^\circ} = 0.675sec$$

$$t_{total} = t_{constant} + t_{accel} + t_{decel} = 0.675sec + 0.2sec + 0.2sec = 1.1sec$$

Bottom Stage:

Givens (all other variables remain the same as the top stage):

$$gravity (g) = 9.8m/sec^2$$

$$top\ stage\ mass (m_{ts}) \approx 2.0lb \rightarrow 0.9kg$$

$$arm\ mass (m_a) \approx 1lb \rightarrow 0.45kg$$

$$arm\ length (r) = 15in \rightarrow 0.38m$$

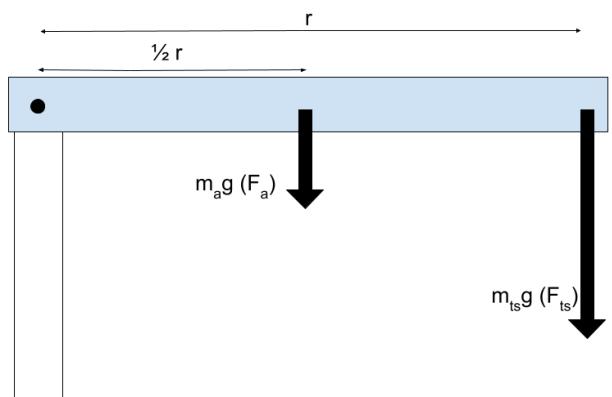
$$desired\ time\ for\ goal\ motion (t) = 0.8sec$$

$$desired\ time\ for\ stake\ motion (t) = 1.1sec$$

$$total\ angle\ adjustment\ for\ goals (\theta) \approx 80^\circ$$

$$total\ angle\ adjustment\ for\ stakes (\theta) \approx 140^\circ$$

$$desired\ accel\ and\ decel\ time (t_{accel}/t_{decel}) = 0.2sec$$



Average angular velocity required for mobile goals (rotation speed needed to match the speed of the top stage):

$$\theta_{rad} = \theta_{deg} \cdot \frac{\pi}{180} = 80^\circ \cdot \frac{\pi}{180^\circ} = 1.4rad$$

$$\overline{\omega}_{goal} = \frac{\theta_{rad}}{t} = \frac{1.4rad}{0.8sec} = \frac{1.75rad}{sec}$$

Average angular velocity required for wall stakes (rotation speed needed to match the speed of the top stage):

$$\theta_{rad} = \theta_{deg} \cdot \frac{\pi}{180} = 140^\circ \cdot \frac{\pi}{180^\circ} = 2.4\text{rad}$$

$$\bar{\omega}_{wall} = \frac{\theta_{rad}}{t} = \frac{2.4\text{rad}}{1.1\text{sec}} = \frac{2.2\text{rad}}{\text{sec}}$$

Because the wall stakes require a larger angular velocity, we will optimize the arm to run at 100% speed for the wall stakes which means that for mobile goals the bottom stage of the arm will not need to go at full speed.

Target maximum angular velocity (how fast the arm should rotate based on the desired time for motion):

$$\theta_{rad} = \bar{\omega}t$$

$$\theta_{rad} = \frac{1}{2}\omega_{max} t_{accel} + \omega_{max} t_{constant} + \frac{1}{2}\omega_{max} t_{decel}$$

$$2.4\text{rad} = \frac{1}{2}\omega_{max} \cdot 0.2\text{sec} + \omega_{max} \cdot 0.7\text{sec} + \frac{1}{2}\omega_{max} \cdot 0.2\text{sec} = 0.9\text{sec} \cdot \omega_{max}$$

$$\omega_{max} = \frac{2.4\text{rad}}{0.9\text{sec}} = \frac{2.7\text{rad}}{\text{sec}}$$

Target RPM (conversion of units from previous calculation):

$$RPM_{target} = \omega \left(\frac{60\text{sec}}{\text{min}} \cdot \frac{\text{rev}}{2\pi\text{rad}} \right) = \frac{2.7\text{rad}}{\text{sec}} \cdot \frac{60\text{sec}}{\text{min}} \cdot \frac{\text{rev}}{2\pi\text{rad}} = \frac{26\text{rev}}{\text{min}}$$

Max static torque (the highest amount of torque that the weight of the arm transfers to the pivot):

$$F_a = m_a g = 0.45\text{kg} \cdot \frac{9.8m}{\text{sec}^2} = 4.41\text{N}$$

$$\tau_a = r_{\perp} F_a \rightarrow \tau_a = \frac{r}{2} \cdot F_a = \frac{0.38m}{2} \cdot 4.41\text{N} = 0.84\text{Nm}$$

$$F_{ts} = m_{ts}g = 0.9\text{kg} \cdot \frac{9.8m}{\text{sec}^2} = 8.82\text{N}$$

$$\tau_{ts} = r_{\perp} F_{ts} = 0.38m \cdot 8.82\text{N} = 3.35\text{Nm}$$

$$\tau_{static} = \tau_a + \tau_{ts} = 0.84\text{Nm} + 3.35\text{Nm} = 4.19\text{Nm}$$

Moment of inertia (the arms resistance to being accelerated):

$$I_a = \frac{1}{3}m_a r^2 = \frac{1}{3} \cdot \frac{0.45\text{kg}}{1} \cdot \frac{0.38^2\text{m}^2}{1} = 0.02\text{kgm}^2$$

$$I_p = \frac{1}{3}m_p r^2 = \frac{0.9\text{kg}}{1} \cdot \frac{0.38^2\text{m}^2}{1} = 0.043\text{kgm}^2$$

$$I_{total} = I_a + I_p = 0.02\text{kgm}^2 + 0.043\text{kgm}^2 = 0.063\text{kgm}^2$$

Torque for arm acceleration (torque needed to accelerate the arm to full speed in 0.2 sec WITHOUT gravity):

$$\alpha = \frac{\Delta\omega}{\Delta t} = \frac{2.2\text{rad}}{\text{sec}} \cdot \frac{1}{0.2\text{sec}} = \frac{11\text{rad}}{\text{sec}^2}$$

$$\tau_{accel} = I\alpha = \frac{0.063\text{kgm}^2}{1} \cdot \frac{11\text{rad}}{\text{sec}^2} = 0.69\text{Nm}$$

Total torque necessary (torque needed to accelerate the arm to full speed in 0.2 sec WITH gravity):

$$\tau_{total} = \tau_{accel} + \tau_{static} = 0.69Nm + 4.19Nm = 4.88Nm$$

Gear ratio needed to produce necessary torque (with a 5.5W motor):

$$gear\ ratio = \frac{\tau_{5.5W}}{\tau_{total}} \cdot \frac{0.26Nm}{4.88Nm} = \frac{1}{19}$$

Max angular velocity (how fast the arm spins with the gear ratio above):

$$\omega_{max} = RPM_{motor} \cdot gear\ ratio \cdot 360^\circ = \frac{200rev}{min} \cdot \frac{1}{19} \cdot \frac{360^\circ}{rev} \cdot \frac{min}{60sec} = \frac{63^\circ}{sec}$$

$$\omega_{max} = \frac{63^\circ}{sec} \cdot \frac{\pi rad}{180^\circ} = \frac{1.1rad}{sec}$$

As previously calculated, ω_{max} should be around $\frac{2.7rad}{sec}$, meaning we need around 245% the current speed, however, we still want to maintain the current torque. To partially get around this issue, we can use rubber bands or some other form of tensioned spring to help counteract the weight of the lift. We estimate that we will be able to comfortably reduce τ_{static} by around 50%.

Max angular velocity (with 5.5W motor and rubber bands):

$$\tau_{total} = \tau_{accel} + \tau_{static} \cdot 0.5 = 0.69Nm + 4.19Nm \cdot 0.5 = 2.79Nm$$

$$gear\ ratio = \frac{\tau_{5.5W}}{\tau_{total}} \cdot \frac{0.26Nm}{2.79Nm} = \frac{1}{10.7} \text{ (rounded to } \frac{1}{10} \text{ as } \frac{1}{10.7} \text{ is not reasonably achievable)}$$

$$\omega_{max} = RPM_{motor} \cdot gear\ ratio \cdot 360^\circ = \frac{200rev}{min} \cdot \frac{1}{10} \cdot \frac{360^\circ}{rev} \cdot \frac{min}{60sec} = \frac{120^\circ}{sec}$$

$$\omega_{max} = \frac{120^\circ}{sec} \cdot \frac{\pi rad}{180^\circ} = \frac{2.1rad}{sec} \text{ (still not fast enough)}$$

Max angular velocity (with 11W motor and rubber bands):

$$gear\ ratio = \frac{\tau_{5.5W}}{\tau_{total}} \cdot \frac{0.52Nm}{2.79Nm} = \frac{1}{6} \text{ (rounded up from } \frac{1}{5.4} \text{)}$$

$$\omega_{max} = RPM_{motor} \cdot gear\ ratio \cdot 360^\circ = \frac{200rev}{min} \cdot \frac{1}{6} \cdot \frac{360^\circ}{rev} \cdot \frac{min}{60sec} = \frac{200^\circ}{sec}$$

$$\omega_{max} = \frac{200^\circ}{sec} \cdot \frac{\pi rad}{180^\circ} = \frac{3.5rad}{sec}$$

Degrees traveled during acceleration and deceleration phases:

$$\alpha = \frac{\omega_{max}}{t_{accel}} = \frac{200^\circ}{sec} \cdot \frac{1}{0.2sec} = \frac{1000^\circ}{sec^2}$$

$$\theta_{accel} = \frac{1}{2}\alpha t^2 = \frac{1}{2} \cdot \frac{1000^\circ}{sec^2} \cdot \frac{0.2^2 sec^2}{1} = 20^\circ$$

$$\theta_{decel} = \theta_{accel} = 20^\circ$$

Time for arm to move 80° (for mobile goal):

$$\theta_{constant} = \theta - \theta_{decel} - \theta_{accel} = 80^\circ - 20^\circ - 20^\circ = 40^\circ$$

$$t_{constant} = \frac{\theta_{constant}}{\omega_{max}} = \frac{40^\circ}{1} \cdot \frac{sec}{200^\circ} = 0.2sec$$

$$t_{total} = t_{constant} + t_{accel} + t_{decel} = 0.2sec + 0.2sec + 0.2sec = 0.6sec$$

Time for arm to move 140° (for wall stake):

$$\theta_{constant} = \theta - \theta_{decel} - \theta_{accel} = 140^\circ - 20^\circ - 20^\circ = 100^\circ$$

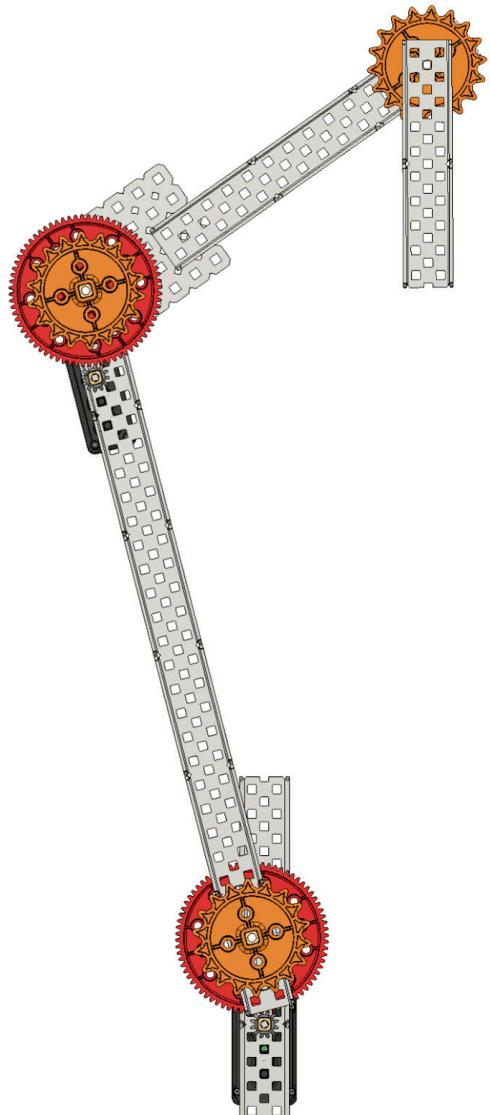
$$t_{constant} = \frac{\theta_{constant}}{\omega_{max}} = \frac{100^\circ}{1} \cdot \frac{sec}{200^\circ} = 0.5sec$$

$$t_{total} = t_{constant} + t_{accel} + t_{decel} = 0.5sec + 0.2sec + 0.2sec = 0.9sec$$

Summary:

Based on the previous calculations, our lift design will be 16.5W. The bottom stage will utilize one 11W motor with a ratio of 1:6 with a 200 RPM cartridge or 1:3 with a 100 RPM cartridge. The bottom stage will utilize rubber bands to cancel out some of the weight of the lift, putting less strain on the motor. Additionally, because of the 11W motor, the bottom stage is capable of completing both motions much faster than the upper stage, meaning it could be geared down even more if it doesn't have enough torque. The top stage was calculated to need a 1:7.5 gear ratio, however, it could very easily be 1:7 if we figure out how to put rubber bands on the top stage. If rubber band mounts are not possible, we will either need to use a 1:8 ratio or an 11W motor.

Even after doing these calculations, it is still difficult to completely predict how much torque each stage needs, meaning our design should be very easy to modify. On the right is a brief CAD model of roughly how we expect our arm to look with gears, motors, and C channels.



06/23/24

Design: 2DOF Lift Inverse Kinematics

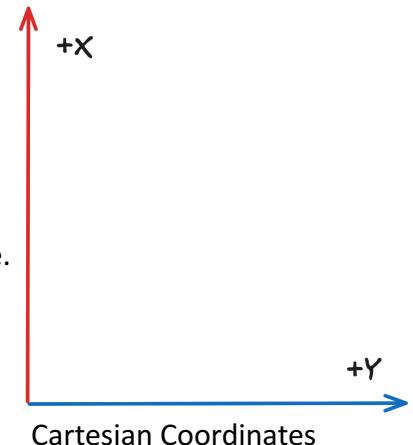
Designed by: Alex

Witnessed by: Carl

Witnessed on: 06/24/24

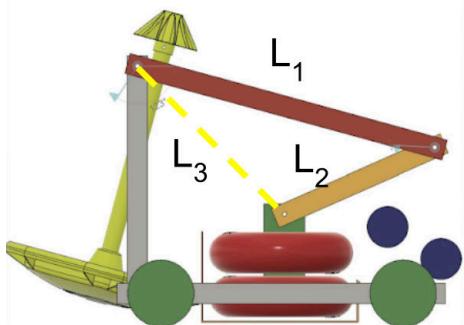
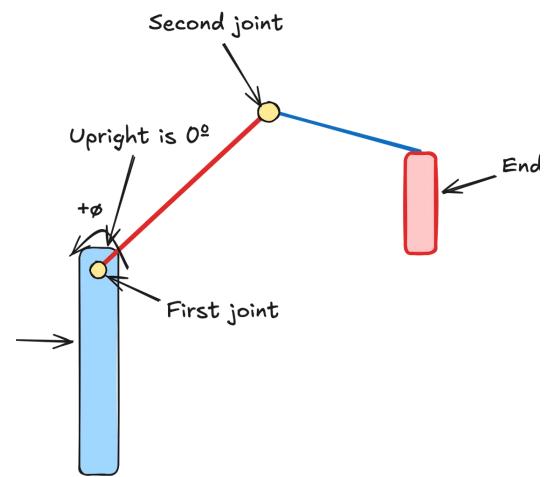
Goal: Design an algorithm to calculate the joint positions for the plunger arm from cartesian coordinates.

In order to control the arm on the robot in a meaningful way we must determine an algorithm to calculate the necessary joint commands given a cartesian coordinate input. Cartesian coordinates, commonly referred to as x-y coordinates, is a human understandable input frame that will make it easy to specify and change goal points for the end effector in a local frame of reference. However these cartesian coordinates are not directly transferable to the end effector positions, and we must do math to determine the end position of the effector given a desired cartesian coordinate.



Joint Representation:

Robots are often described by joints. In our system we have 2 joints, one between the bottom stage and the upright and one connecting the top and bottom stages. On our arm each joint can be explained with the distance from the last joint and a static zero for the arm to start at. We can use this description of the arm to calculate possible joint angle positions to achieve the cartesian goal state using inverse kinematics. Additionally because we are in the 2d plane and we have 2 degrees of freedom (DOF) on the arm, the solution to calculate the necessary joint angles is closed form, which makes it extremely easy to calculate on the brain. During our research we found a [site by Alan Zucconi](#) that describes the calculations necessary to find the joint angles.

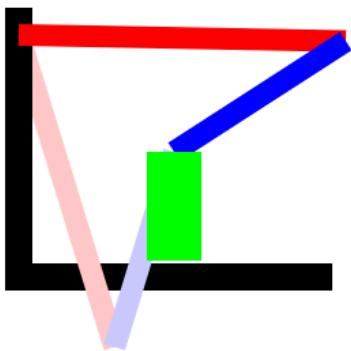
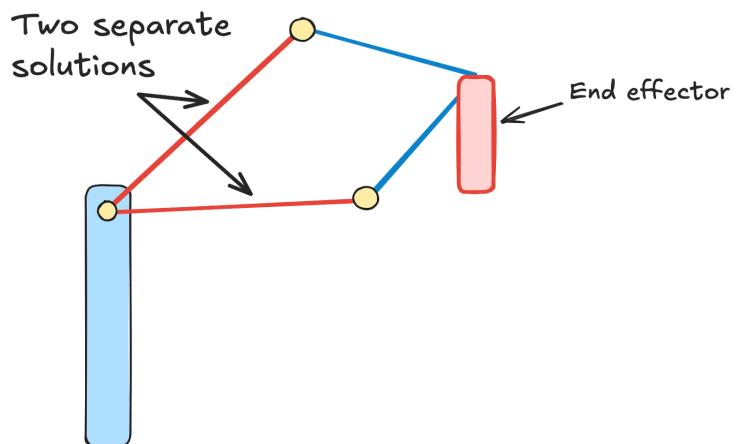


$$A_1 = \text{First angle} = \arctan\left(\frac{\text{goalY}}{\text{goalX}}\right) + \frac{\pi}{2} + \arccos\left(\frac{L_3^2 + L_1^2 - L_2^2}{2bL_1}\right)$$

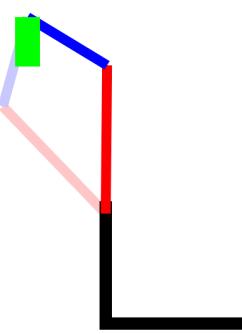
$$A_2 = \text{Second angle} = \pi - \arccos\left(\frac{L_2^2 + L_1^2 - L_3^2}{2L_1L_2}\right)$$

Picking optimal solutions

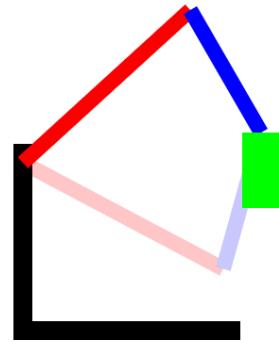
These calculations don't consider all the solutions that exist for the arm, to do this you have to use the other solutions found using inverse trigonometric functions. We implemented all of this in a [p5.js script](#) to visualize and test out different inverse kinematics solutions. We used this to ensure that our calculations worked on a simulation of the real robot, which also enabled us to test out cartesian joint animations. These illustrations from the simulator show the inverse kinematics following the cartesian inputs from the mouse. Additionally the darker solution in the simulation is the one that we think the robot should select because it interferes the least with the path the rings need to take.



Picking up rings



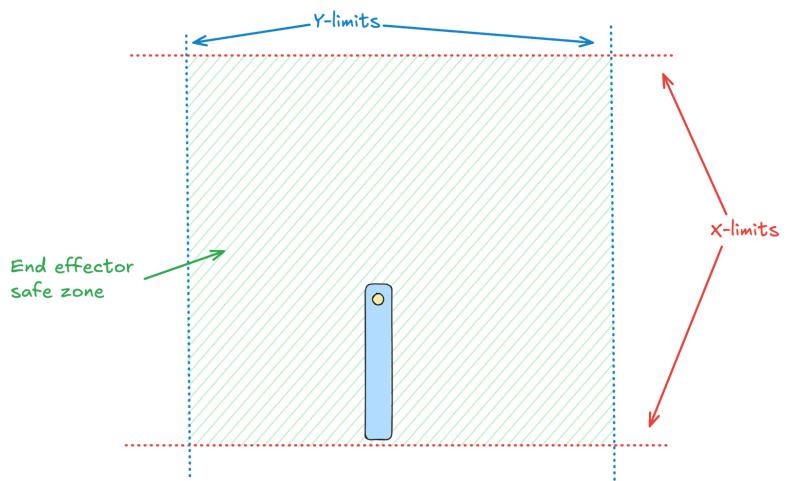
Placing on neutral stake



Placing rings on goal

Size Limitations

This year the size constraints must be kept at all points during the match. This is especially applicable with our arm because with how long the arm is it is able to reach beyond the vertical or horizontal expansion limits in rule <SG3>/<SG4>. To ensure the arm stays within the size constraints we will limit the maximum end effector locations in cartesian space. We will use minimum and maximum x- and y-coordinates to achieve this. A graphic of how this algorithm will work is to the right.

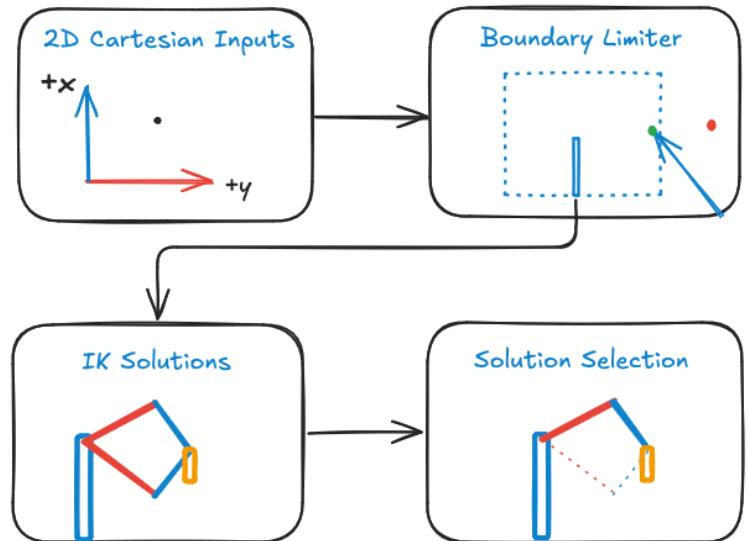


Complete Algorithm:

Our algorithm takes in 2D cartesian inputs, puts a boundary limit on them to ensure the robot's end effector stays inside the size constraints for the robot rules. After that we use inverse kinematics to find all the possible solutions for the inverse kinematics of the arm, followed by selecting the solutions based on which one allows for the smoothest transition of the rings. We send these joint angles to the robot allowing us fine cartesian control of the arm.

Conclusion:

The simulator and calculations we made have been a large step in leveraging the full control that the 2DOF arm allows. Testing in the simulator provides a safer and quicker way to implement motion commands on the arm, and proves that in an ideal environment our inverse kinematics algorithms will be capable of moving the arm to cartesian goal states. This capability will help us move the arm on our robot more effectively and deliberately when we try to score and grab rings in different directions.



06/23/24

Design: Drive Base Specs

Designed by: Carl

Witnessed by: Alex

Witnessed on: 6/24/24

Goal: Determine the maximum amount of drive motors that we can have and what the best gear ratio is.

With the lift system requiring 16.5W, we believe that by putting a 5.5W on the intake and utilizing pneumatics for all the other mechanisms, we will be able to run a 66W drivetrain. This will help provide a huge advantage during matches, already discussed [prior chassis analysis](#) (Pg. 50-52). We will be using a tank drive for this robot because of its dominance in almost every category in our [decision matrix](#) (Pg. 7). In order to achieve the optimal balance of speed and torque on our drivetrain, we will need to create a gear train that changes the RPM between the motor and the wheel. For these ratios, we will be using gears instead of chains because of how compact and robust they are (previous research: [gears vs. chain](#) (Pg. 70)). Another benefit that a gear system offers is multiple outputs from a central input and vice versa, meaning if we get tipped and only one wheel is touching the ground, we will still have the full power of the drivetrain being exerted into the floor. By nature, using a higher RPM input motor will mean that a smaller input gear and a larger wheel gear will be necessary to achieve a desired RPM. If we were to use 600 RPM motor cartridges it would help make the drivetrain significantly smaller and more compact. This concept has been integral for compact drivetrains in previous years, going from a [bulky drivetrain](#) (Pg. 50) in Tipping Point to a [compact and integrated chassis](#) (Pg. 52) in Over Under. To help when looking for gear ratios for the drive base and in general, we developed a quick reference table in google sheets (shown below).

		Input Gears						
		12	24	36	48	60	72	84
Output Gears	12	100	200	300	400	500	600	700
	24	50	100	150	200	250	300	350
	36	33	67	100	133	167	200	233
	48	25	50	75	100	125	150	175
	60	20	40	60	80	100	120	140
	72	17	33	50	67	83	100	117
	84	14	29	43	57	71	86	100
		Input Gears						
		12	24	36	48	60	72	84
Output Gears	12	200	400	600	800	1000	1200	1400
	24	100	200	300	400	500	600	700
	36	67	133	200	267	333	400	467
	48	50	100	150	200	250	300	350
	60	40	80	120	160	200	240	280
	72	33	67	100	133	167	200	233
	84	29	57	86	114	143	171	200
		Input Gears						
		12	24	36	48	60	72	84
Output Gears	12	600	1200	1800	2400	3000	3600	4200
	24	300	600	900	1200	1500	1800	2100
	36	200	400	600	800	1000	1200	1400
	48	150	300	450	600	750	900	1050
	60	120	240	360	480	600	720	840
	72	100	200	300	400	500	600	700
	84	86	171	257	343	429	514	600

Because we have built many robots in previous years, we have a pretty good idea of how much power and speed different gearings have. Listed below are stats from drivetrains that we have used in previous years along with similar gear ratios that we haven't used yet.

Past Robots (600 RPM Motors)		
RPM and Wheel Size	Max Speed (in/sec)	Robot Weight (lbs)
450 on 2.75"	65	15
360 on 3.25"	61	16
450 on 3.25"	77	12
343 on 4"	72	14

Other Options		
RPM and Wheel Size	Max Speed (in/sec)	Max Robot Weight (lbs)
480 on 2.75"	69	14
400 on 3.25"	68	14
300 on 4"	63	16

Wheel size is irrelevant for torque and speed if the gear ratio is appropriately adjusted. For this robot we believe that by using 3.25" wheels we will be able to achieve a good balance of grip and small size, leaving us with 3 potential gearings: 36:60 360 RPM, 48:72 400 RPM, and 36:48 450 RPM. Because of our analysis on lift speed, we know that it will not be the limiting factor in scoring, therefore a more torque-focused drivetrain would allow us to achieve game tasks, and counter defense better. Furthermore, having an extremely fast chassis would also make us more likely to tip over when the lift is up, so we believe having a 360 RPM chassis with 3.25" wheels is ideal.

06/27/24 Identify: June 25th Game Manual Update Analysis

Designed by: Matt	Witnessed by: Carl	Witnessed on: 06/29/24
-------------------	--------------------	------------------------

Goal: Analyze the impact of this update on strategy and design.

Changed the Field layout such that Positive Corners and Negative Corners are now on the same side of the Field, rather than catercornered.

Impact: This will affect which side of the field we put the mobile goals but will not have a difference on autons, strategy, and robot.

Added a new rule, <SC9>, that adds a 2-point bonus per Climb for whichever Alliance has a Rings Scored on the High Stake at the end of a Match

Impact: This rule incentivises high hangs and more advanced mechanisms, however we need to consider if the 4 points we could potentially gain are worth the added weight and complexity.

Added a new rule, <SG11>, to add a 10 second protection to Positive Corners at the end of a Match

Impact: This rule makes hang more applicable because it allows teams to have more time to hang at the end of the match because they don't need to be protecting their corner. This will impact the strategy and allow robots to have higher, slower hangs (<10 sec) rather than faster lower hangs (<3 sec).

Added an additional Violation Note to <SG4> to state that a Team will receive a Major Violation for removing three (3) or more Rings from the Field in a single Match

Impact: This rule will have some impact on the game, we just need to make sure when we are descoring and scoring wall stakes we don't accidentally remove them out of the field, and we should stop descoring if we have already removed two rings from play. This also factors into robot design, we will want to be able to control the rings as they come off of the posts.

07/02/24 Brainstorm: Programming Software Architecture

Designed by: Alex	Witnessed by: Carl	Witnessed on: 07/03/24
-------------------	--------------------	------------------------

Goal: Identify key requirements for potential programming software architectures.

Requirements:

To make effective software, especially at the scale of the codebase that we are using can be very difficult. A way to mitigate issues is to lay out requirements for the code and brainstorm solutions to implement these effectively:

- Maintain asynchronous localization
- Control asynchronous subsystems in tandem with each other
- Keep close timings(100Hz frequency, 1ms consistency)
- Be easy to maintain
- Maintain safety (No race conditions/non-thread safe code)

Considering the current robot design with several highly integrated subsystems(Intake, arm, plunger), the need for robust state management becomes apparent. Additionally, we need robust communication layers to ensure all code running on the robot can quickly get the information it needs to run. Furthermore, ensuring proper type- and thread- safety will be crucial to constantly putting high quality code onto the robot, and must be done to limit single-point failures during skills and tournament matches. This is why we think implementing and testing a robust state management system must be done to ensure tournament robustness, timing, and safety goals with our code.

Codebase Guidelines:

A good way to organize thoughts and ensure consistency in projects is to create a set of guidelines to follow when writing code. This is often a practice of larger companies but even organizing these thoughts in smaller settings can help us create more high-quality and consistent code throughout the season.

1. Limit number of concurrent instances of the same objects
 - a. Use multi-threading safe solutions where necessary but keeping control of an object in only one context is best.
2. Use dynamic allocation only where necessary
 - a. Statically allocated arrays with fixed size at compile time are generally more efficient and can be more verbose.

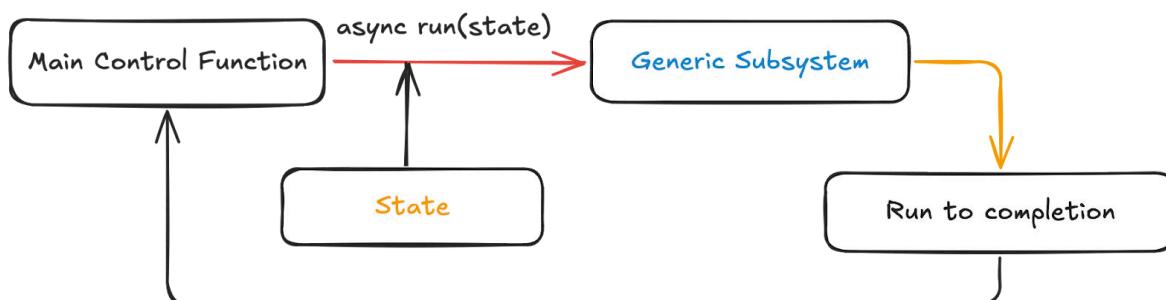
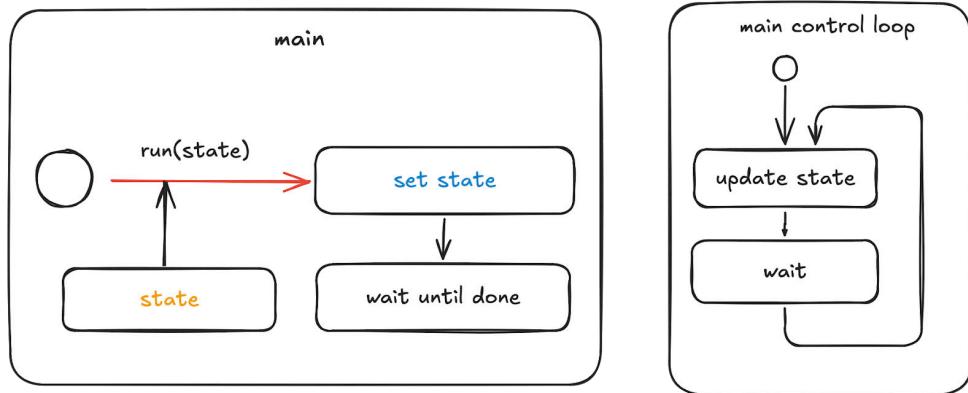
3. Don't use *unsafe* blocks
 - a. In our code there is little to no reason to use *unsafe* code blocks
 - b. Utilizing *unsafe* blocks stops the compiler from checking code for mistakes which can lead to many critical errors.
4. Follow official rust style guidelines
 - a. Follow the [official rust style guidelines](#)
 - b. This will ensure that stylistically our code will stay consistent throughout the season

Codebase Layout:

Another key component of maintaining a large codebase is planning out the structure for major pieces before you create them to ensure decisions are well designed. In our code base and with the state machine architecture it is especially important to design code that is able to scale well.

State Machine Architecture:

We thought the best way to plan our state machine architecture would be to start with a flowchart and ensure that we can retain the highest simplicity possible. Our initial idea for the flowchart is to the right. It allows us to easily change states and maintain the states over time. However, it requires many more function calls than necessary and it requires a control loop to be running all the time, when it only really needs to run when there is a state active. To remedy this we thought of the following structure that allows us to take advantage of the vexide asynchronous features and run the update only when it is in a state. We can also use the rust borrow checker to ensure that only one state is running at a time by passing in a mutable reference to the drivetrain on run(state). This makes sure that only one thing can be using the same resources at once.



The state machine in its current state is very simple to use, for example, in the code below we create and run a state to completion in only one line of code. This shows the state machine being used to start a pure pursuit state at the beginning of autonomous

```
async fn autonomous(&mut self) {
    self.drivetrain.run(PurePursuit::new(args)).await;
}
```

We can also use async features such as the select!() macro and the join!() macro to allow us to race states and run all states to completion. For example this code runs both teleop tasks to completion in driver:

```
async fn driver(&mut self) {
    let drive_state = self.drivetrain.run(TankDrive::new(&self.controller));
    let arm_state = self.lift.run(TeleopArm::new(&self.controller));

    join!(arm_state, drive_state).await;
}
```

Localization:

With our current implementation of a state controller, each state's update takes in an input and sends an output. In the drivetrain state controller we can store the localization object. This will allow us to get data from the localization and send it directly to the state that needs to do the processing. The fine localization details will be discussed much further in Localization Implementation section

Summary

In this section, we planned out how we will program the robot this year. By thoroughly thinking through the structure of the code we have this season, we will more effectively be able to program and expand the code we write throughout the entire season. We created a state machine that effectively uses the language features, such as the borrow checker and the asynchronous executor to write exceptionally efficient state-machine code.

07/12/24 Design: CAD Day 1&2 (C.1.1/C.1.2)

Designed by: Carl

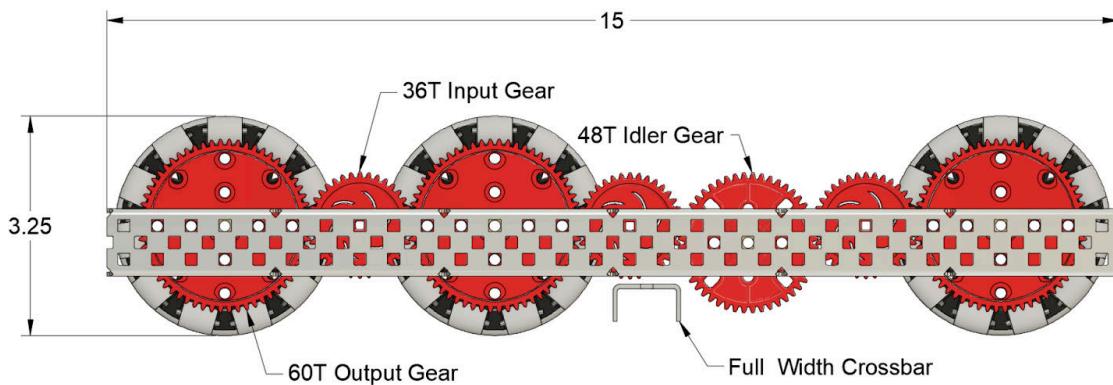
Witnessed by: Alex

Witnessed on: 7/13/24

Goal: Start designing the robot in CAD and address any concerns with the design.

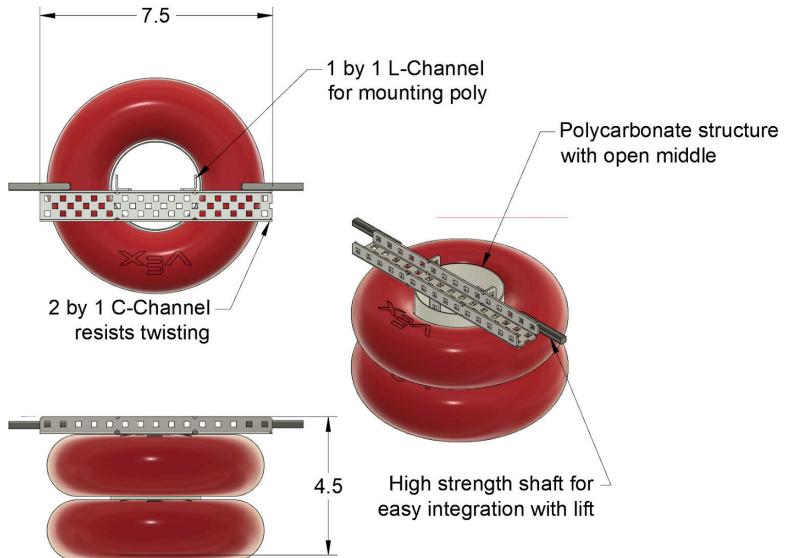
Drivetrain:

To start the robot, we first designed one side of the chassis using three 3.25" omni wheels and a 36:60 360 RPM gear ratio. The drive features three 36T gears per side which will all ideally be used as input gears. We made the chassis 15 inches or 30 holes long, meaning we still have an extra 3" of our starting size length so the intake and other mechanisms can stick out. Because of the gear and wheel sizes, the center wheel had to be offset slightly to one side. This is not inherently a problem but as we are designing the rest of the robot we will need to ensure that heavier components like the brain and battery are positioned in such a way that the weight is evenly distributed between all the wheels for better performance and lower resistance when turning. Below is a side view of the chassis.



Plunger:

The plunger is one of the trickier parts of our robot because of the fact that it needs to have a diameter of less than 3" so it can fit inside of the rings, but it also needs to fit the rubber tip of a stake which only compresses to around 2.5". On top of that, the plunger needs to have an actuating element at the bottom to grab rings. Because we do not have access to parts and this concept is mostly unproven, we will not be developing the actuation mechanism yet, however we will ensure that the design could facilitate one.



Arm:

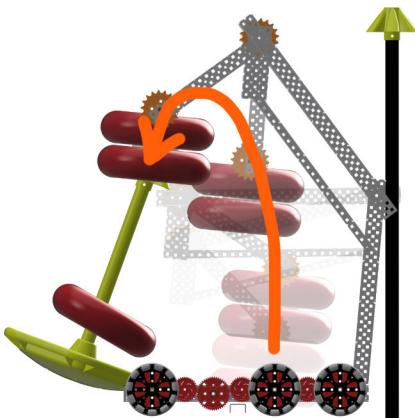
Quickly after beginning the top stage, it became obvious that a chain bar for both stages was not going to be possible because of its thickness. For all of our other models of the lift, we had been completely worrying about the ZX plane while mostly neglecting the Y. The primary problem was that the lift needed to be driven by motors on the outside, But because the ring needs to fit in between the two sides of the lift, each side of the lift would need to fit in a 3 to 4 hole gap. To help with this, we chose to replace the bottom stage of the lift with a 4-bar, which allows it to fit in the desired width and adds a little bit more robustness, with the main drawback being a little bit of extra weight. This is the current CAD model:



Currently, we have two main concerns:

1. The location of the ring staging area, as we think fitting in drive motors will be really hard
2. It is not possible to score on any kind of wall-mounted stake while in possession of a goal (see above images). These two problems would severely hinder the rest of the build and gameplay, so they need to be addressed.

To solve the problem of the motors, we decided to move the staging area above the chassis so that all the motors could fit neatly underneath the rings. To solve the second problem, we want to try moving the uprights to the front of the robot and changing the arm geometry slightly to accommodate this change.



07/17/24

Design: CAD Day 3 (C.1.3)

Designed by: Carl

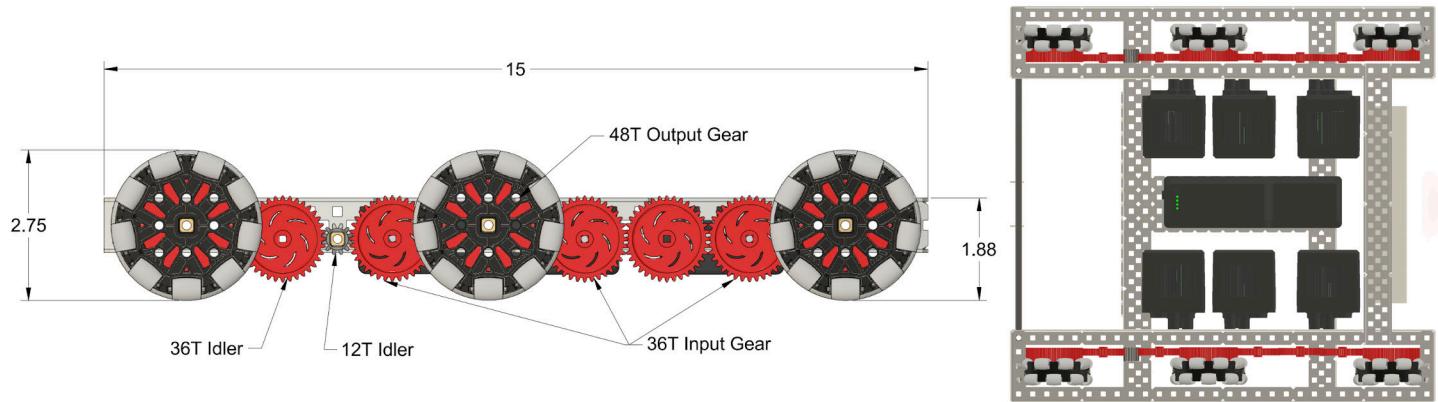
Witnessed by: Alex

Witnessed on: 7/18/24

Goal: Create a new robot model with a more compact drivetrain and begin intake work.

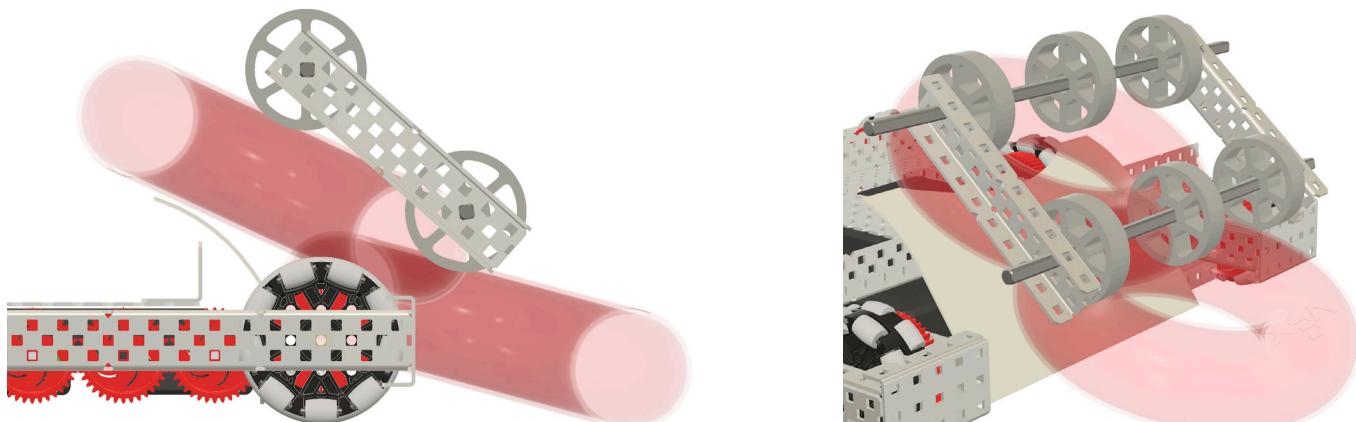
Drivetrain:

When working with the previous robot model (C.1.2), we noticed that the drivetrain was bigger than it could be and with our complex design, that was not acceptable. To rectify this, we will attempt to make a new CAD model (C.1.3) using the smaller 2.75" wheels. The gear ratio closest to C.1.2 on 2.75" wheels is 36:48 450 RPM. Additionally, with the new chassis we are aiming to lower the drive motors, making more space for ring storage and thus shortening the intake.



Intake:

For the intake, we prioritized simple design with minimal moving parts. We are using 2" flex wheels because they are much more durable and have a much lower entanglement risk than rubber band rollers. The intake has two stages; a rigidly mounted upper roller and a lower roller that pivots on the top roller's shaft, allowing the intake to maintain constant pressure on the rings while allowing them to change angles as they are picked up off of the tiles. For the intake back, we opted for a bent piece of polycarbonate as it was much lighter and lower friction than metal, while also being incredibly customizable.



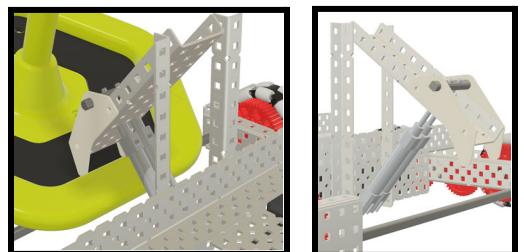
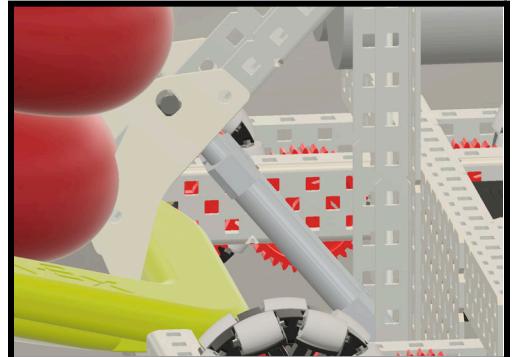
07/17/24 Design: CAD Day 4 (C.1.3)

Designed by: Carl	Witnessed by:	Witnessed on:
-------------------	---------------	---------------

Goal: Complete C.1.3.

Back Clamp:

As of right now, the only mobile stake manipulator design we have thought about or seen is some variation of a tipping point goal clamp where pistons (typically two) are used to interact with the closest inside edge of a mobile goal. By applying force, the manipulator tilts the goal while lifting it slightly off of the floor. In addition to not requiring a motor, this style of grabber is also very compact, making it a viable option for our robot. On our back clamp, we decided to use two 25mm pistons because of their small size and relatively low air use. These pistons then open and close a clamp optimized for the small space we had available.



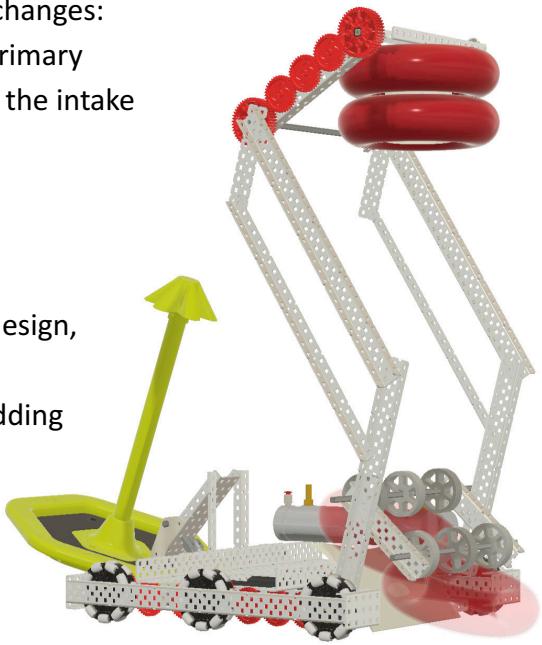
Lift System:

Our lift follows the same general concept as the old one with a few changes:

1. 15° angled lift towers to allow for more forward reach. The primary purpose of this is to allow us to score on wall stakes without the intake contacting the base of the poles.
2. More efficient C-channel placement for a simpler structure.
3. Gears instead of chains on top stage for durability.

Despite these changes and considerable tweaking around with the design, we still have several issues that have not been solved:

1. No space for lift motors or gearing on either stage without adding high complexity or a much larger footprint.
2. Lack of bracing: The bottom halves of the lift cannot be connected because the top stage moves between them. This likely will make the lift too flimsy to reliably place rings on posts.
3. The high strength shaft on the back of the lift interferes with rings on the mobile goal.



07/18/24 Evaluate: Initial Concept Reconsideration

Designed by: Carl	Witnessed by: Matt	Witnessed on: 07/20/24
-------------------	--------------------	------------------------

Goal: Evaluate C.1.3 and the original concept to determine if it is viable to keep.

Good Reasons to Keep Design:

- Promising neutral stake scoring
- Allows for 6 motor drive
- We already have solved many difficult problems from this design

Good Reasons to Change Design:

- Very limited structure for the lift
- Efficient filling of mobile posts is key for skills and matches
- This design is bulky and large
- Plunger will likely be difficult to design and execute
- Many high-risk single points of failure

Conclusion:

- Our current design is well thought out
 - Cons: It's heavy and hard to fit in size
- Considering new designs would allow us to possibly find a lighter, simpler design

After considering the pros and cons of our current design, we have decided to look into new solutions. Our current design would be efficient at wall stakes and supports a 6 motor drive, but it has major limitations. The lift structure is weak, the design is bulky and close to size limits, and the plunger will be hard to design and build. More importantly, we have found better design options that can fill mobile posts and even wall stakes more efficiently. Despite the time spent on the current design, switching to a new design will lead to a more effective and competitive robot.

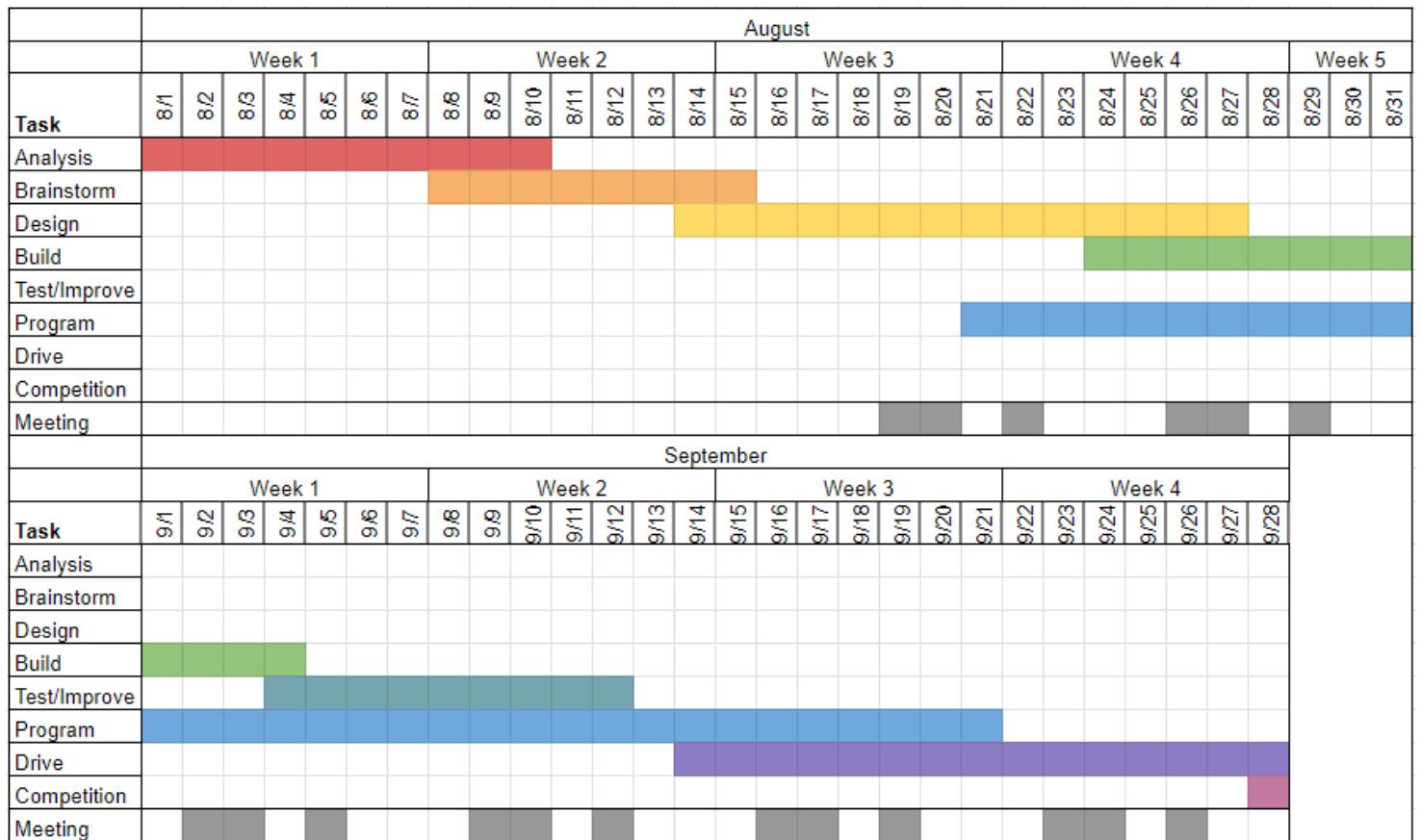
08/01/24

Time Management: Revised Timeline

Designed by: Matt	Witnessed by: Alex	Witnessed on: 08/02/24
-------------------	--------------------	------------------------

Goal: Create a revised timeline for the new design cycle.

This Gantt chart outlines a general plan for the next design cycle, giving us time for each part of the design project. This design cycle is preparing us for our tournament on the 28th of September. In order to maximize our preparedness for the tournament, we condensed our build and design phases to allow for ample programming, testing, and drive time.



08/05/24 Analysis: Minnesota Signature Event

Designed by: Carl	Witnessed by: Alex	Witnessed on: 08/06/24
-------------------	--------------------	------------------------

Goal: Watch some of the higher level matches and identify key strategies and game altering errors and moves.

Accountability notice: Notable Designs completed late on 08/13/24

Notable Strategies:

1. Early Corner Hold:

In this strategy, teams were filling up their mobile stakes in autonomous and early driver control, then rushing for the closest "+" corner. When in the corner, they could hold off other teams, defending the corner for the rest of the match.



2. Goal Hoarding:

Similar to the first strategy, teams would rush a full mobile goal and then go to a "+" corner. During this time, their alliance partner would fill up a second mobile and then deposit it in the same corner. This strategy would allow one member of the alliance to get the last free mobile goal and fill it up with rings.



3. Late Hanging:

With the introduction of <SG11> in the [June 25th update](#) (Pg. 104), the goals in the positive corners are protected in the last 10 seconds, giving teams a chance to hang. This strategy was often combined with the 1st one for consistently high performance.



4. Opportunistic Descoring:

This strategy is where the teams opportunistically find goals with many rings from the opposing alliance to move from the "+" corner or the field into the "-" corner for a net point gain in the match. This strategy was most effective in the last few seconds of a match when teams would drop their goal to climb.



(all images from Minnesota Signature Event livestream)

Links to Matches:

Day 1 Stream: [Minnesota Signature Event on Vimeo](#)

Day 2 Stream: [Minnesota Signature Event on Vimeo](#)

Finals Matches: [Minnesota Signature Event: Finals](#)

Notable Designs:

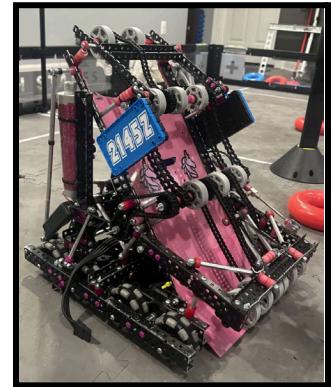
1. Flex Wheel Intake + Claw (2775V/2145Z):

This design essentially combines two of our original concepts for this year into one robot. Very straight forward in terms of functionality as both mechanisms operate independently of each other.

Here are some pros and cons of this design:



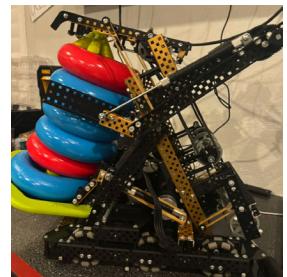
Pros:	Cons:
<ul style="list-style-type: none"> • Simple • Can score in all ways • 6 motor drive • Very fast at mobile stakes • Easy to build 	<ul style="list-style-type: none"> • Slow at wall stakes • Claw requires precise driving • No descoring mechanism



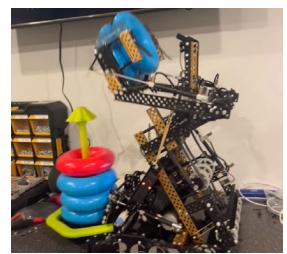
(Image directly from 2145Z)

2. Backpack (360X):

This design by 360X utilizes the mobile goal as a way to store rings during the match which gives more room for strategic maneuvers. The motor driven arm with a piston actuated passive ring holder makes scoring 2 rings on any wall stake possible and fast.

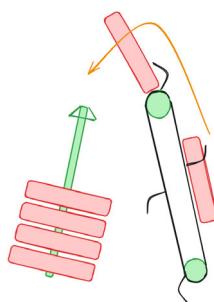


Pros:	Cons:
<ul style="list-style-type: none"> • Fast at wall stakes • 6 motor drive • Small footprint 	<ul style="list-style-type: none"> • Very complex • Heavy • Hard to design and build • Unreliable



(360X MOA Recap)

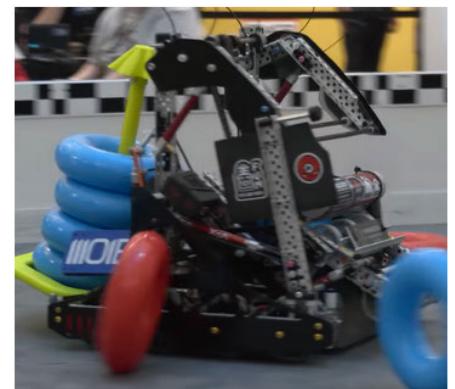
3. Chain & Standoff Intake:



Easily the most common and reliable design, every robot in finals and most in eliminations had designs very similar to this. This concept is essentially just a chain belt with rigid hooks on it typically combined with a flex wheel bottom stage. By having the hooks on the bottom/inside of the intake, rings can be reliably slammed on the top of the goal without the need for a hood.

(Diagram by Carl R.)

Pros:	Cons:
<ul style="list-style-type: none"> Extremely simple Lightweight 6 or even 7 motor drive Super reliable Easy to build Can do alliance stakes quickly Space for hang 	<ul style="list-style-type: none"> No descoring abilities Can't score on wall stakes Can't get a very high skills score Can jam with lots of rings

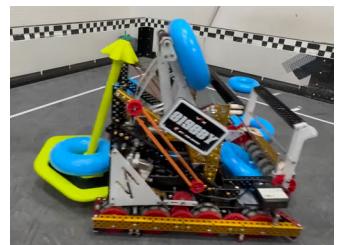
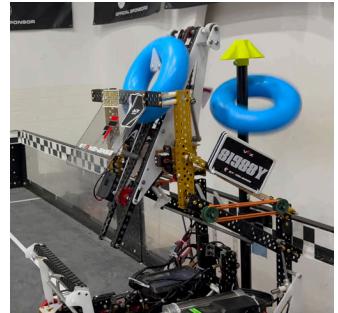


(11101B Pits & Parts)

4. Chain & Standoff Intake on Lift (1233H/81988Y):

This design is very similar to (3) but adds an arm for the top stage of intake, enabling scoring on wall stakes. With the correct number and spacing of hooks, it can score two rings on wall stakes quickly, but NOT while in possession of a mobile base.

Pros:	Cons:
<ul style="list-style-type: none"> Simple Lightweight Reliable Fast at mobile stakes Fast at wall stakes Good for skills 	<ul style="list-style-type: none"> No descoring abilities Must drop mobile stake to score on wall posts Low motor count on drive



(81988Y Reveal)

5. Snail Intake (8110W):

This intake is functionally the same as the last two, however near the middle it also has a piece of polycarbonate on a hinge that acts as a 1 way door for rings. This allows the intake to reverse while it has a ring in it, which then forces the ring into the storage compartment on the front.



Pros:	Cons:
<ul style="list-style-type: none"> • Lots of moving parts • Fast at mobile posts • Can do 2 rings at a time on wall stakes <u>while</u> in possession of a mobile stake 	<ul style="list-style-type: none"> • No descoring abilities • Heavy • Prone to jams



(8110W Pits & Parts)

Sources and Citations:

1. 2775V Jackson Area Robotics Pits & Parts: [2775V Jackson Area Robotics | Pits & Parts | High Stakes](#)
2. 360X MOA Recap (youtube): [360X MOA Recap - Vex High Stakes](#)
360X Full Circle Pits & Parts: [360X Full Circle | Pits & Parts | High Stakes](#)
3. 11101B Barcbots Getting There Pits & Parts: [11101B Barcbots Getting There | Pits & Parts | High Stakes](#)
4. 81988Y High Stakes Reveal: [\[VEX High Stakes\] 81988Y HIGH STAKES REVEAL](#)
5. 1233H MOA Reveal: [1233H MOA REVEAL | High Stakes](#)
6. 8110W Whisper Pits & Parts: [8110W Whisper | Pits & Parts | High Stakes](#)

08/08/24

Brainstorming: C.2.1 Concept One

Designed by: Carl

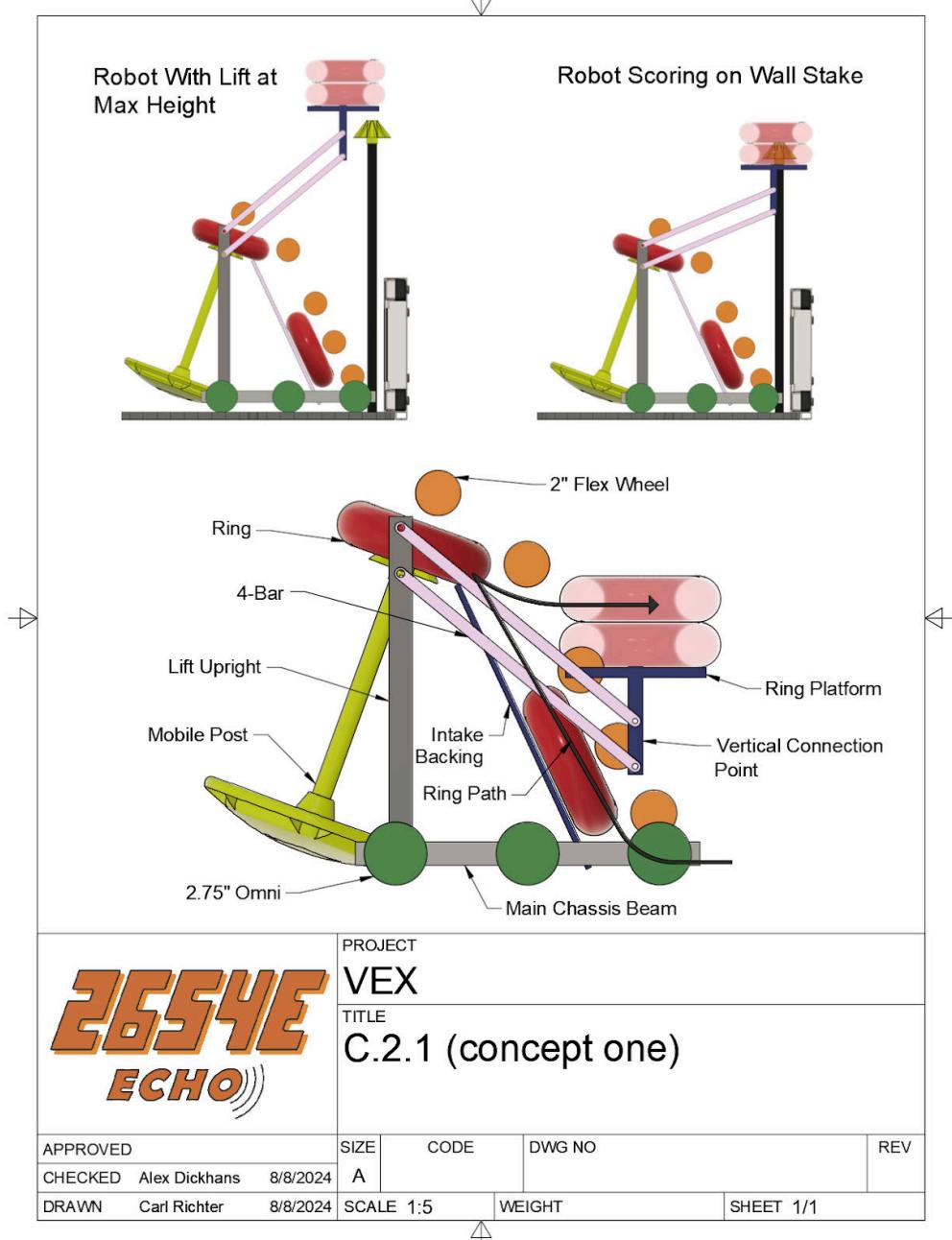
Witnessed by: Matt

Witnessed on: 08/09/24

Goal: Create a visual representation of how a snail style intake bot would look and move.

Explanation:

This is our attempt at creating a “Snail Bot”. It consists of a long 4-bar lift (chosen because of its simplicity and strength) that is fed by a flex wheel intake (because a chain intake doesn’t have gaps for the rings to exit). This robot still has all of the functionality of a standard intake robot but also has the ability to score two rings at a time on the posts. This robot is inherently very complex as there needs to be a ring redirection system, passive grabbing mechanism on the ring platform, and 4-5 different intake rollers.



08/10/24

Brainstorming: C.2.1 Concept Two

Designed by: Carl

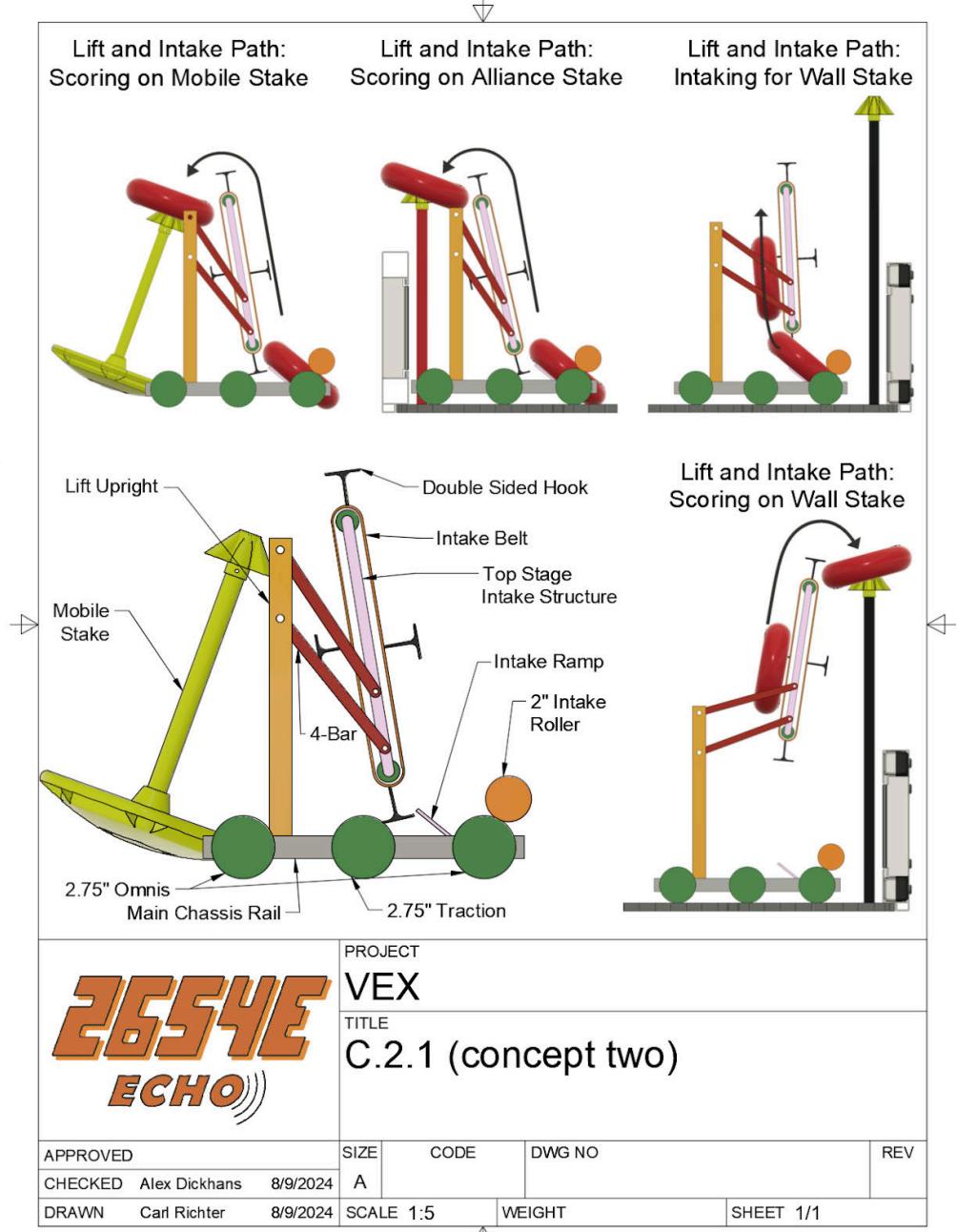
Witnessed by: Matt

Witnessed on: 08/11/24

Goal: Brainstorm a chain & standoff intake that can score on wall stakes while in possession of a mobile stake.

Explanation:

After completing our [Minnesota Sig Analysis](#) (Pg. 114-116), we realized that a chain intake on top of a lift was one of the simpler designs that could score on all of the stakes, however, it still has a glaring problem. In order to score wall stakes, we would need to release the mobile goal in the back clamp, and as we previously discussed, that can create the opportunity for big descores against us. The big thing prohibiting reaching over the mobile goal to get to the stake is the sizing limits described in <SG2> and <R5>. Our concept utilizes the intake's ability to spin in both directions with double sided hooks to make it possible to score on posts in front of the robot. Additionally, by utilizing a four bar with an offset linkage, we can make the intake tilt forward when the lift raises.



08/14/24

Brainstorming: C.2.1 Concept Three

Designed by: Carl

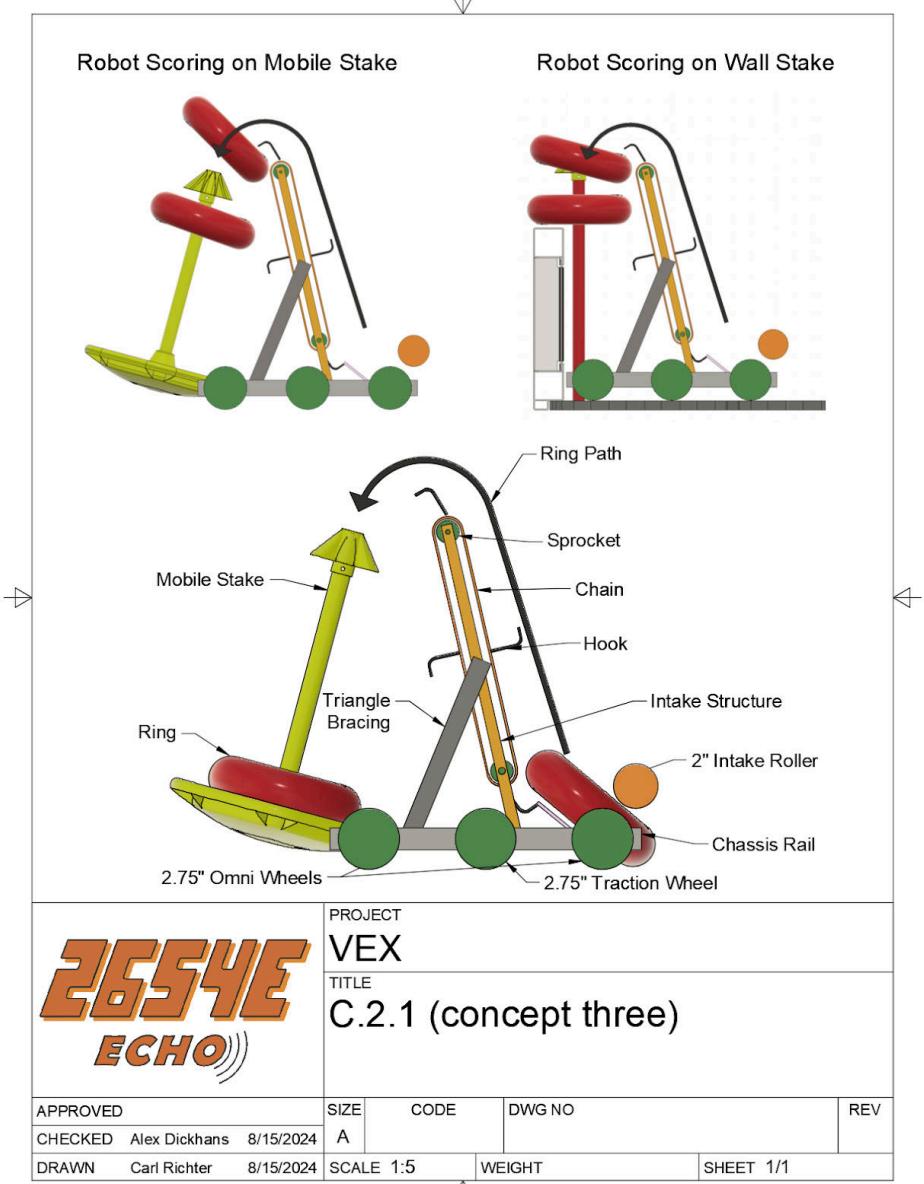
Witnessed by: Matt

Witnessed on: 08/14/24

Goal: Design a simple chain and standoff intake for comparison with the other options.

Explanation:

Sometimes, a super simple robot is also viable as shown by our [MOA Analysis](#) (Pg. 114-116). This style of robot is similar to Concept 2, but without wall stake capabilities. This robot design has been proven to work well but has its limitations, especially in the skills challenge. We also anticipate that as other designs evolve, this one will slowly become less competitive and potentially completely obsolete.



08/15/24

Select Solution: C.2.1 Concept Decision

Designed by: Alex

Witnessed by: Matt

Witnessed on: 08/15/24

Goal: Select the best concept using a decision matrix.

For this design matrix we kept many of the same criteria from the last design [concept decision](#) (Pg. 91-92). However with this design matrix we will not be considered descoring safety because none of the robot concepts we designed are capable of descoring. We also kept the weights similar to our last decision matrix because in our analysis they were good.

Concept 1		Concept 2		Concept 3	
Criteria:	Weight	Score	Weighted Score	Score	Weighted Score
Goal Stake Scoring Speed	5	8	40	8	40
Goal Stake Security	6	9	54	9	54
Wall Stake Scoring Speed	6	6	36	8	48
Descoring Speed	2	0	0	0	0
Robot Simplicity	2	3	6	5	10
Robot Weight	2	5	10	8	16
Total Score:			146		168
					137

Conclusion:

In the design matrix the design concept 2, with a lifted intake had the highest score with 168. We think that this is the best design because it is very fast to score on both the mobile stakes and the wall stakes. We think that this is very important because it will allow us to get a very high score in skills and be competitive in matches. Additionally, concept 2 was much smaller and lighter than concept one, making it a more ideal choice.

08/15/24

Identify & Design: C.2.1 Requirements

Designed by: Carl

Witnessed by: Matt

Witnessed on: 08/15/24

Goal: Set goals for the CAD and define some constraints for the robot.

Motor Distribution Rationale:

With our selected design, we have five subsystems: lower intake, upper intake, lift, drivetrain, and goal clamp. The goal clamp is easier to operate with pistons, so no motors are needed for it. The lift mechanism cannot be pneumatic since it needs to stop at more than two positions. All other mechanisms must be motorized for continuous motion. The lift could use a 5.5W motor (with good rubber band assistance), but it would still be slower. While the drivetrain could use more motors, beyond a certain point, the robot doesn't benefit from additional speed as the intake can't keep up. We observed this at the Minnesota Signature, and believe a faster intake would improve our skills score significantly. To achieve this speed without losing torque, we need to dedicate 11W per stage, leaving 60.5W for the drivetrain. However, to power it evenly, we can only use 55W (27.5W on each side), leaving an extra 5.5W. We could use this 5.5W for a mechanism to remove rings from the corner or to increase the lift's power.

Arguments for Adding to Lift:	Arguments for Using for Corner Mechanism:
<ul style="list-style-type: none"> Lift would be faster and less prone to burn out Corner mechanism is extra weight Corner mechanism could be pneumatic 	<ul style="list-style-type: none"> Would significantly increase the amount of usable rings Necessary for a competitive autonomous Using a motor for this is lighter than pneumatics

To determine if the lift needs extra power, we will perform some rough calculations.

Lift Speed Calculations:

Givens:

$$\text{gravity } (g) = 9.8 \text{ m/sec}^2$$

$$\text{time } (t) \leq 1 \text{ sec}$$

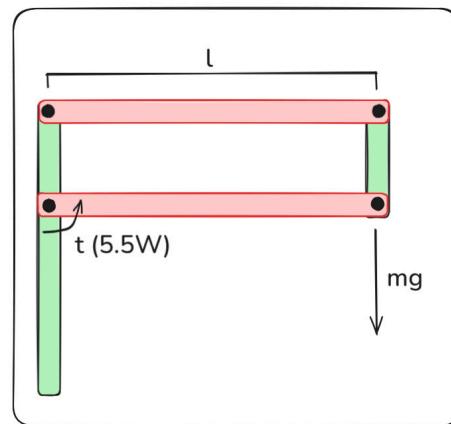
$$\text{angle change } (\theta) = 55^\circ$$

$$\text{intake mass } (m) = 1.5 \text{ kg}$$

$$\text{arm length } (l) = 0.2 \text{ m}$$

$$\text{motor torque } (\tau_{5.5W}) = 0.26 \text{ Nm}$$

$$\text{desired accel and decel time } (t_{\text{accel}}/t_{\text{decel}}) = 0.2 \text{ sec}$$



Target maximum angular velocity (based on the time (t) that we want for the lift to raise):

$$\theta_{rad} = \theta_{deg} \cdot \frac{\pi}{180} = 55^\circ \cdot \frac{\pi}{180^\circ} = 0.96 \text{ rad}$$

$$\theta_{rad} = \overline{\omega t} = \frac{1}{2}\omega_{max} t_{accel} + \omega_{max} t_{constant} + \frac{1}{2}\omega_{max} t_{decel}$$

$$0.96 \text{ rad} = \frac{1}{2}\omega_{max} \cdot 0.2 \text{ sec} + \omega_{max} \cdot 0.6 \text{ sec} + \frac{1}{2}\omega_{max} \cdot 0.2 \text{ sec} = 0.8 \text{ sec} \cdot \omega_{max}$$

$$\omega_{max} = \frac{0.96 \text{ rad}}{0.8 \text{ sec}} = \frac{1.2 \text{ rad}}{\text{sec}}$$

where ω_{max} is the maximum angular velocity of the arm

Target RPM (conversion of units from previous calculation):

$$RPM_{target} = \omega \left(\frac{60 \text{ sec}}{\text{min}} \cdot \frac{\text{rev}}{2\pi \text{ rad}} \right) = \frac{1.2 \text{ rad}}{\text{sec}} \cdot \frac{60 \text{ sec}}{\text{min}} \cdot \frac{\text{rev}}{2\pi \text{ rad}} = \frac{11.5 \text{ rev}}{\text{min}}$$

Max static torque (the highest amount of torque that the weight of the arm transfers to the pivot):

$$F = mg = 1.5 \text{ kg} \cdot \frac{9.8 \text{ m}}{\text{sec}^2} = 14.7 \text{ N}$$

$$\tau_{static} = lF = 0.2 \text{ m} \cdot 14.7 \text{ N} = 2.94 \text{ Nm}$$

where F is force

Moment of inertia (the lifts resistance to being accelerated):

$$I = ml^2 = \frac{1.5 \text{ kg}}{1} \cdot \frac{0.2^2 \text{ m}^2}{1} = 0.06 \text{ kg m}^2$$

Torque for arm acceleration (torque needed to accelerate the lift to full speed in 0.2 sec WITHOUT gravity):

$$\alpha = \frac{\Delta\omega}{\Delta t} = \frac{1.2 \text{ rad}}{\text{sec}} \cdot \frac{1}{0.2 \text{ sec}} = \frac{6 \text{ rad}}{\text{sec}^2}$$

$$\tau_{accel} = I\alpha = \frac{0.06 \text{ kg m}^2}{1} \cdot \frac{6 \text{ rad}}{\text{sec}^2} = 0.36 \text{ Nm}$$

where α is angular acceleration

Total torque necessary (torque needed to accelerate the lift to full speed in 0.2 sec WITH gravity):

$$\tau_{total} = \tau_{accel} + \tau_{static} = 0.36 \text{ Nm} + 2.94 \text{ Nm} = 3.3 \text{ Nm}$$

Gear ratio needed to produce necessary torque with 50% band assistance:

$$gear ratio = \frac{\tau_{5.5W}}{\tau_{total} \cdot 0.5} \cdot \frac{0.26 \text{ Nm}}{3.3 \text{ Nm} \cdot 0.5} \approx \frac{1}{6}$$

Max angular velocity (how fast the arm spins with the gear ratio above):

$$\omega_{max} = RPM_{motor} \cdot gear ratio \cdot 360^\circ = \frac{200 \text{ rev}}{\text{min}} \cdot \frac{1}{6} \cdot \frac{360^\circ}{\text{rev}} \cdot \frac{\text{min}}{60 \text{ sec}} = \frac{200^\circ}{\text{sec}}$$

This is more than enough speed as the lift only needs to achieve 50°/sec. We will likely use a 1:9 gear ratio which works out to be 133°/sec, which is still more than fast enough and creates a sizable margin for error from the calculations. This also means that we can use a 5.5W motor for our descoring mechanism.

Finalized Power Distribution:

Chassis: 55W

Lift: 5.5W

Top Intake: 11W

Bottom Intake: 11W

Corner Mechanism: 5.5W

Back Clamp: 2 Pistons

Chassis:

Similar to our previous CAD, we will be utilizing a tank drive largely due to its speed and overall competitiveness ([Drivetrain Types](#) (Pg. 46-49)). In the past we have always used fully drifting drivebases using only omni wheels ([Past Chassis Analysis](#) (Pg. 50-52)), which has provided great handling in driver and made for an offensive powerhouse. This year, we will be subject to intense defense and constant interaction with other robots as detailed in our [MOA Analysis](#) (Pg. 115-117). To help counter defense, we will use traction wheels instead of our center omni wheels. This will also help make our autonomous more consistent by significantly reducing uncertainty in our position caused by drift. To further the effects of these wheels we will offset them downwards slightly to put more force on them.

To help us design a small robot, we will be using 2.75" wheels.

Because the chassis only has 55W compared to the "standard" 66W, it will have less power assuming the same gear ratio. With the limited gear ratios we have, 450 RPM is still the best option, and the pushing power is still in line with some of our faster robots in the past. This means that if we keep our weight down (<15lbs), we should be fine.

Past Robots (600 RPM Motors)		
RPM and Wheel Size	Max Speed (in/sec)	Robot Weight (lbs)
450 on 2.75"	65	15
360 on 3.25"	61	16
450 on 3.25"	77	12
343 on 4"	72	14

Lift:

- The lift needs to be rigid and must have the correct geometry to ensure that it can move the intake to all three locations
- The lift tower must be shorter than the first hanging rung (16.1")
- The lift gearing must be possible to adjust (start with a 1:9 ratio)

Top Intake:

- Lightweight, ideally less than 3lbs
- Gearing should be easy to change but should start at 300-600 RPM
- Double sided hooks need to be spaced two rings can be held on one side of the intake
- Rotation sensor would be ideal

Bottom Intake:

- Protected against other robots
- As light as possible
- Few moving parts
- Faster or the same speed of the top stage

Corner Mechanism:

- Out of the way of other mechanisms
- Be able to grab specific rings from the corner stacks
- Light so actuation can be as fast as possible
- Within the horizontal expansion limits defined by <SG2>
- Retract fully within the footprint of the robot
- On the front of the robot

Back Clamp:

- Steal proof
- Should be able to grab the mobile posts from any angle
- Low air consumption (at least 10 grabs per match+)
- No backwards expansion

Other Robot Goals:

- Height: 16"
- Length: 14"
- Width: 14"
- Wall/alliance stake guide on front and back
- Sub 15 pounds
- Low brain and battery mounts for optimized center of gravity
- Make all of the different components accessible and easy to maintain

08/15/24 Background Research: Robot Localization: Literature Review

Designed by: Alex	Witnessed by: Carl	Witnessed on: 08/16/24
-------------------	--------------------	------------------------

Goal: Use what we determined in the ideology part of localization research to find solution(s) that we can test to determine the best algorithm.

What are the Goals for Localization?

Before we start looking at solutions we would like to review our requirements for our localization system. Additionally we would like to start the discussion of testing methods for the different localization methods that are options. The following are the goals we would like to achieve, with a rough idea of how we could measure them or evaluate the papers we are reading.

- Global Localization
 - We want our robot to be able to determine its position on the field without knowledge of the initial location of the robot
 - This will allow us to recover from errors over time, instead of accumulating small errors.
- High accuracy
 - We want our odometry to be accurate to < 1 in and maintain this at 100 Hz
- High speed
 - We want the odometry to refresh at a rate of 100 Hz to match the maximum speed we can send data to the motors
- Redundant
 - We want our robot to be able to recover from a single or multiple point of failure in localization without suffering accuracy too much
 - This is very difficult but it will allow us to reduce the number of failures during competition drastically
- Use as many sensors as possible
 - Using all the sensors available properly will hopefully increase our localization accuracy, along with our redundancy at competitions if we can detect these well. Removing single points of failures in our system will be especially helpful as it becomes more complex.

To do this analysis we will do a short literature review of applicable solutions to VEX and analyze the data and usability of the results from these papers.

Monte Carlo Localization for Mobile Robots (Particle Filter)

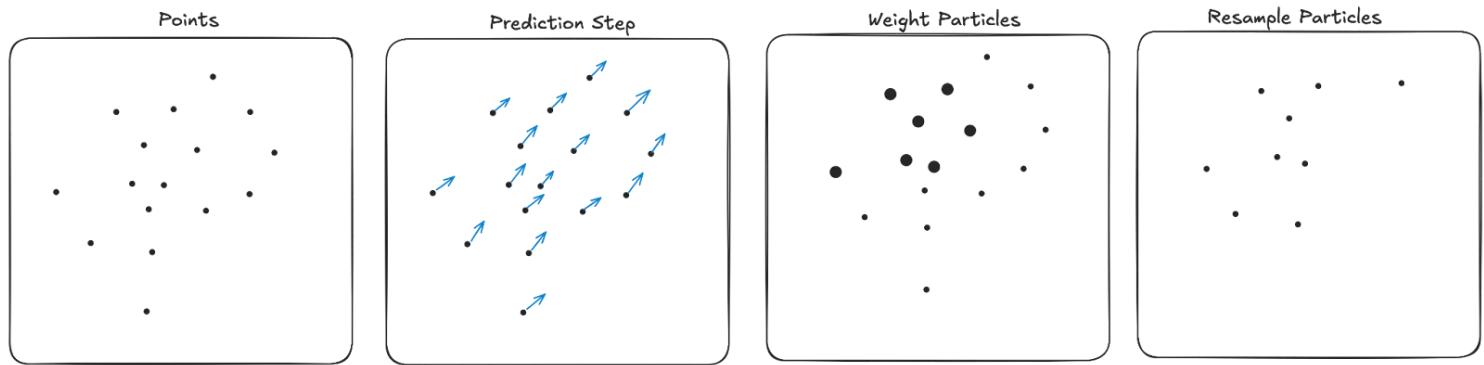
In this paper, Frank Dellaert, et al. describes the methods to localize a robot platform using a particle distribution to represent the probability densities of the robot position. This solution allows for the model to account for different probability distributions than filters such as a kalman filter would be able to properly represent. In this paper they describe how they were able to apply this technique to a moving robot with range finders that are similar to the LIDAR ToF “Distance Sensor” available to us in VEX, but this solution could easily incorporate multiple different sensor sources, such as the GPS sensor or IMU if you are able to implement those into the filter.

Method:

Monte Carlo localization is a Bayes filter, which means that it recursively updates the estimate of the robot’s position, only given the past estimate, and the current control input. The particle filter or Monte Carlo Localization has 2 steps every update, prediction and update parts.

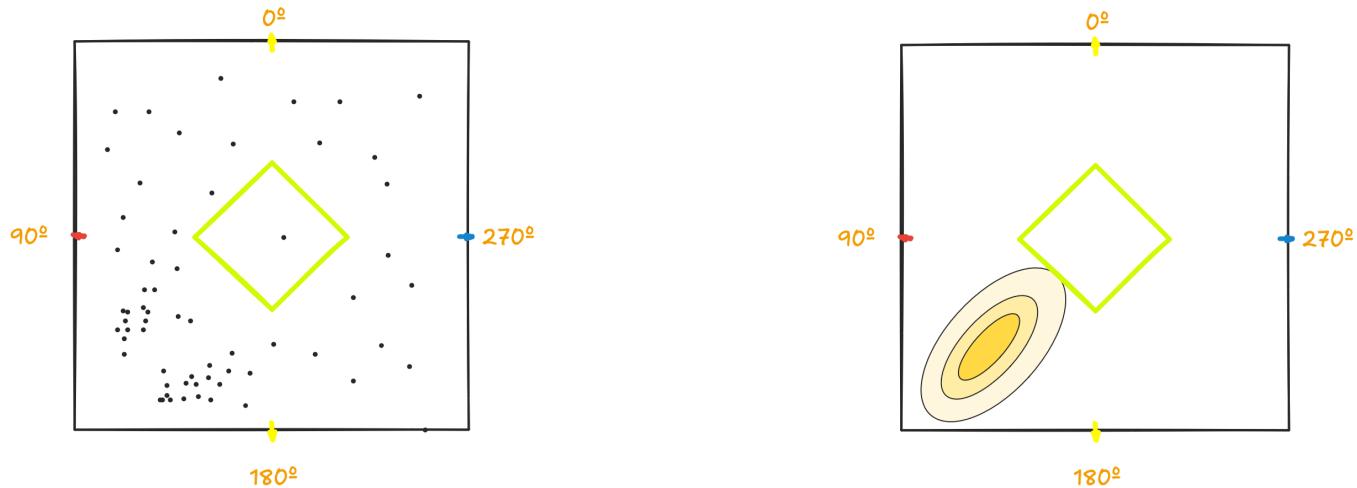
1. Prediction phase:
 - a. In this step the filter iterates through each particle and predicts how the robot could have moved, given the control input or sensor readings, such as drivetrain encoders.
2. Update phase:
 - a. In this step the filter weighs each particle using the probability that you would get this sensor measurement given that the robot is at that position, then there is a resampling step to move particles to places with higher weights

Illustration:



Key Benefits:

Non-gaussian probability distributions vs. gaussian probability distributions



In the left image you can see the density of the particles, represented by the black dots and on the right is a gaussian covariant 2d distribution represented with the different colors indicating the probability. On the left, there are 2 points of high density, with some points still scattered around the area, while on the right the distribution is a lot more regular, and could not properly represent the data we get back from distance sensors. This is one of the key benefits of a particle filter is that it can incorporate data from multiple nonlinear sensors.

Drawbacks:

This filter is much more resource intensive than other types of filters, such as a kalman filter because it has to maintain state over many more points where a kalman filter maintains the state belief in a more coherent and easier to compute method.

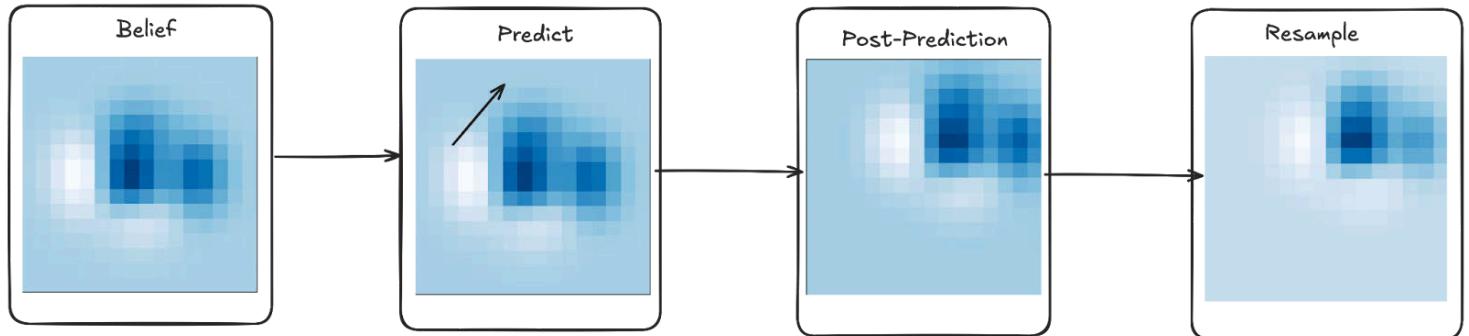
Markov Localization for Reliable Robot Navigation and People Detection

Where the Particle filter represents the state of the robot using points on a uniform grid. It works very similarly to the particle filter, employing many of the same methods to model the probability distribution at each point. This filter was also developed to model distributions that the kalman filter was not able to represent because of the gaussian nature of the predicted distributions. The grid based nature discretizes these complex distributions in a grid format that is better suited for the complex distributions that occur with distance sensors.

Method:

1. Prediction Phase:
 - a. Translate the grid by how far the robot was predicted to move
2. Update Phase:
 - a. Use sensor readings to get a new prediction at each point
 - b. Reweigh the predictions.

Illustration:



Key Benefits:

- Allows uniform sampling of potential states

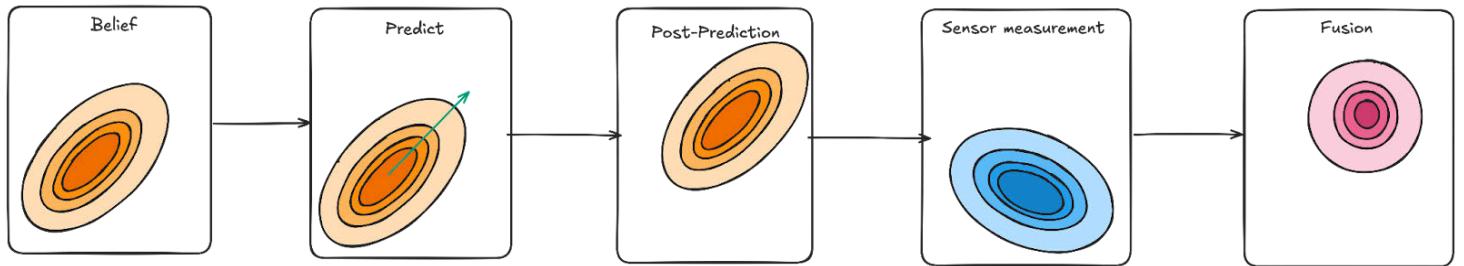
Drawbacks:

- Computes probability for points that are unlikely
- Discretization of points limits accuracy
- Difficult to transfer distribution

Kalman Filter

Similar to other filters the kalman filter is a bayes filter, however this filter represents the probability distribution with a covariance matrix. It stores the belief of the bayes filter in the mean and covariance estimate. It then updates the covariance matrix.

Illustration:



Key benefits:

- Lower computation complexity
 - Allows us to use more of the brains resources for stuff such as motion planning
- More deterministic
 - All factors are determined by deterministic code that will run the same with the same inputs

Drawbacks:

- No good easy way to add distance sensors
- Doesn't work with line sensors
- Needs more detailed sensor models

Conclusion:

These derivatives of the bayes model from literature provide a strong basis for us to start our localization implementation. Each one operates very uniquely, they all represent the belief of where the robot is in very different ways. Because of this they each have unique benefits and drawbacks to consider when making a filter. Currently based off of our review either grid-based or particle-based localization seems to fit our needs best. They both allow more flexible sensor models for complex systems such as line sensors and distance sensors for the walls, but still allow you to incorporate sensors such as the GPS that are easier to use. However we do not have the resources to implement and test both solutions for our use case, so we decided to use the particle filter because of its non-discrete nature.

08/16/24

Design: Robot Localization

Designed by: Alex

Witnessed by: Carl

Witnessed on: 08/16/24

Goal: Implement the Particle Filter based on our research in the Localization Literature Review

Particle Filtering

In our previous Localization literature review we decided to implement a particle filter. We briefly discussed how the particle filter works on a high level but here we will discuss the nitty-gritty implementation details we went through to implement particle filtering in our code. The main 2 parts of particle filters are the prediction and update step, but each step has many individual substeps. We broke this down into reasonable substeps below:

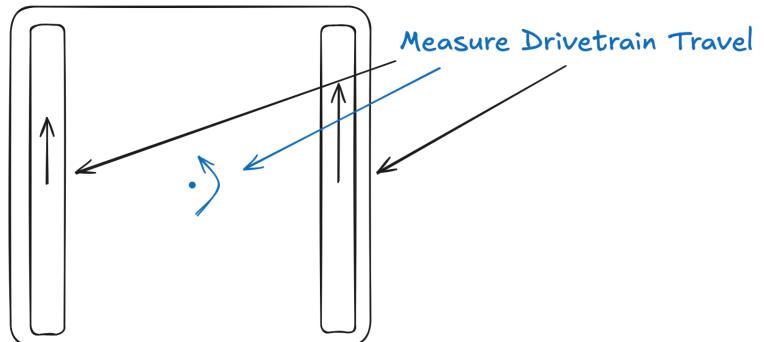
1. Prediction Step
 - a. Measure prediction sensor inputs
 - b. For each particle
 - i. Add noise to sensor inputs
 - ii. Calculate translation
 - iii. Add translation to point
2. Update Step
 - a. Determine if update is needed
 - i. If we haven't traveled far enough updating will potentially decrease state accuracy
 - b. For each particle
 - i. Calculate the weight by summing the probabilities for each of the sensors
 - c. Resample
 - i. Use systematic sampling to get N new points from the weighted belief

Prediction Step

In the prediction step we have 2 steps. The first step is straightforward, we measure the movement of the drive using sensor inputs such as motor encoders or odometry wheels. We run the `.update()` method every 10ms to get and store new sensor readings at the highest possible rate.

```
pub async fn update(&mut self) {
    // Update the deltas for each side of the drive
    this frame
    self.left_delta = self.left_side.update().await;
    self.right_delta =
    self.right_side.update().await;

    // Update the heading once per frame
    self.heading = self.orientation();
```



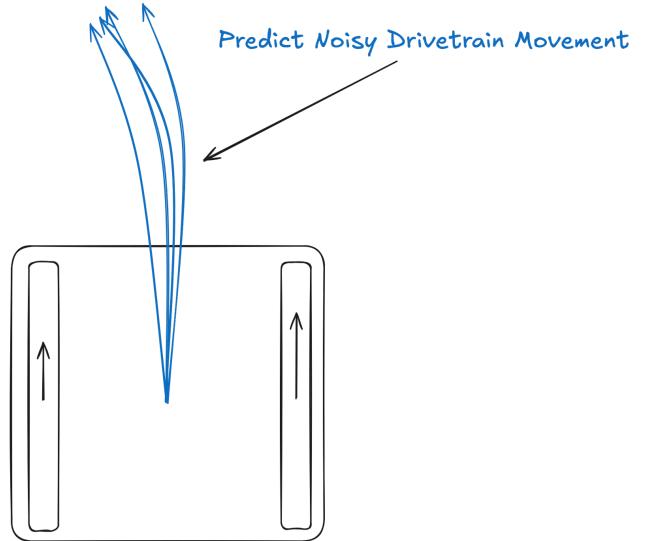
We then use the `.predict()` method for each of the particles to move each particle with some noise. The fact that we add noise to the sensor readings is very confusing, we have clean sensor inputs, but we add noise to ensure variation in the points we are sampling from. This noise would seem to make the predicted movement of the robot also noisy. However because of the number of points we will have (often in the hundreds) these small, consistent noise additions at each point will approximate the sensor inputs. This noise will help in the update step because it thrives on stronger deviations in measurements. This is something we will often experiment with later, because maintaining a strong deviation in the particle cloud belief will allow us to rule out more areas of the field more effectively. The code for the prediction is below:

```
pub fn predict(&mut self) -> StateRepresentation {
    // Calculate noise for each side of the drive
    let left_noisy = self
        .rng
        .sample(Normal::new(self.left_delta, self.drive_noise * self.left_delta).unwrap());
    let right_noisy = self
        .rng
        .sample(Normal::new(self.right_delta, self.drive_noise * self.right_delta).unwrap());

    // Because we are calculating from the center of the drive the mean is the displacement of
    // the center of rotation
    let mean = left_noisy + right_noisy / 2.0;

    // Calculate a local displacement vector from the current position of the robot
    let local = Rotation2::new(
        -self.heading.angle() + self.rng.sample(Normal::new(0.0, self.angle_noise).unwrap()),
    ) * Vector2::new(-mean, 0.0);

    // Create a state representation without a angle(z) change
    // We do this because the IMU is more than accurate over the entire match
    StateRepresentation::new(local.x, local.y, 0.0)
}
```



Update Step

The update step is significantly more complicated, as illustrated in the right picture from “Monte Carlo Localization for Mobile Robots” by Dallert et al. with the steps from $S'(k)$ to $S(k)$, we have to weigh in all the inputs from each of the sensors. This resamples the points to better reflect the robot’s true position. This resampling step is extremely important to maintain a strong position of belief.

However if this runs too often when the robot stays in the same spot it will converge the probability distribution too far and it will have poor variation in the points once it starts moving. The max frequency we can update the particle filter is 100Hz but when we are stationary we think it would be helpful to experiment with reducing it down to as low as 2Hz or not at all to ensure it doesn’t converge too fast. However if we are moving it is extremely important to keep updating our belief so we added a feature that overrides the frequency slowdown and updates after the robot moves a preset distance.

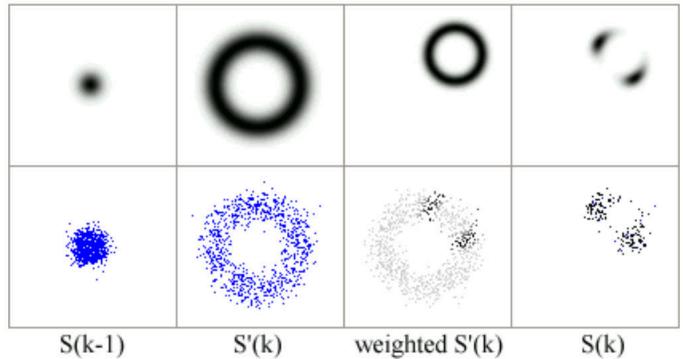
```
// Only run if it's been longer than the minimum time or moved more than 2 inches
if (Instant::now() - self.last_update_time) < self.min_update_interval
    && self.dist_since_update < self.mi
n_update_distance.get::<meter>()
    || self.sensors.is_empty() {
    return;
}
```

We then calculate the weights for each of the particles, given by their individual sensor models. We use a *Sensor* trait to represent this functionality in code. However not every sensor is always going to get a reading and sometimes we may want to discard a reading, which we represent with Option::None. We do have to consider this in our summing operation, which we do using the `.filter_map()` function and then we use the language feature `.sum()` to get the sum of all the weights for all of the sensors.

```
// Update step
let mut weights = [0.0; D];

// weight particles
for (i, particle) in self.particles.iter().enumerate() {
    weights[i] = self
        .sensors
        .iter()
        .filter_map(|sensor| sensor.p(&particle))
        .sum();
}
```

One of the hardest parts of the kalman filter is properly resampling the belief from the weight of the particles. Ideally the particles with the highest weight would be the most resampled and you could use a simple random sample to find the best points. However, a particle filter is known to be more computationally complex and not



provide the strongest diversity in samples. If you do stratified sampling there is more variation because it doesn't get quite as much from the most strongly weighted particle.

```
// Calculate average weight and random variable for resample
let avg_weight = weights.iter().sum::<f64>() / weights.len() as f64;
let sample_rand = self.rng.sample(Uniform::new(0.0, avg_weight));

// Clone the particles to be memory safe with resample
let old_particles = self.particles.clone();

let mut sum = sample_rand;

// Sample an adequate number of points
for i in 0..weights.len() {
    // Start with a sum and index for the while loop
    let mut weight_sum = weights[0];
    let mut particle_index = 0;

    // Increase until it gets to the right index
    while weight_sum < sum && particle_index < weights.len() {
        particle_index += 1;
        weight_sum += weights[particle_index];
    }

    // Resample the particle
    self.particles[i] = old_particles[particle_index];

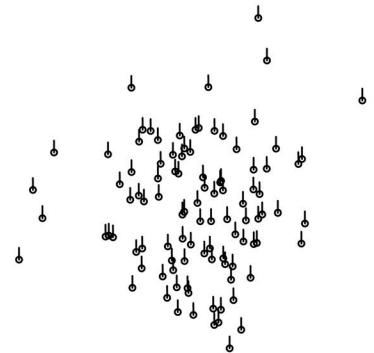
    // Increase the sum
    sum += avg_weight;
}
```

Testing

We do not have hardware ready to test the particle filter yet, however we can use motors to simulate movement with the encoders and a “dummy” sensor to weigh the particles appropriately. We graphed particles on a field tool, however with live and wireless communication we could only represent relatively few points(5-10). This will still allow us to easily visualize the if and how the particle filter works:

Particle initialization:

To the right is a diagram of 100 particles being randomly initialized with a normal distribution around the center and an identity covariance matrix. This shows that the points are generated closer to the center and there are a couple points on the edges. This is very similar to how we will initialize the belief of the robot before matches start.



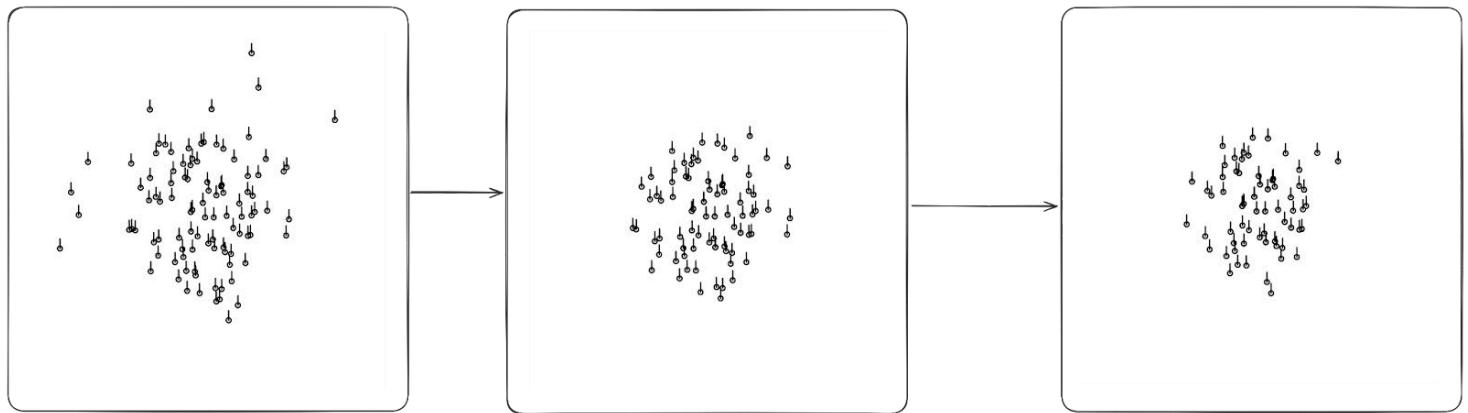
Prediction phase:

In this test we initialized the robot's belief with 10 points exactly on the center. We then use the motor to drive forwards towards the top of the field. It is clearly visible that the noise was added and accumulated into a nice distribution at the end.



Update Phase:

To test this we will initialize a large divergent randomly initialized particles similar to the first test picture and then wait until it converges onto the center where the "dummy" sensor has the highest probability. In the picture below the particles converge on the dummy sensor probability.



Conclusion

Following the guidance from the paper Monte Carlo Localization for Mobile Robots by Dallert et al. we implemented and tested a particle filter. The next step to complete an operational particle filter is creating sensor models to determine the weights for each of the sensors and testing it on a real robot. These sensors include many types such as line sensors, distance sensors, and the game positioning sensor (GPS). Having this reliable localization will allow us to create motion algorithms that are able to move more repeatably and reliably during skills and matches.

08/17/24 Design: Sensor Models

Designed by: Alex	Witnessed by: Carl	Witnessed on: 08/18/24
-------------------	--------------------	------------------------

Goal: Implement sensor models for every sensor we need based on a strong numerical analysis.

In this section we are going to implement the sensor models for the sensors we researched in [Robot Localization: Ideology](#) (Pg. 80-83). We will use the background research we collected here and our own sensor measurements to establish sensor models we can use in our particle filter.

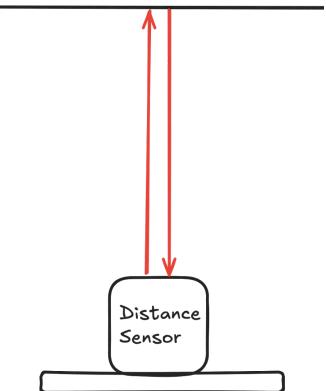
What is a Sensor Model?

A sensor model is code and math that describes the probability of a sensor given a specific state the robot is in. These sensor models are extremely important for the update step of the particle filter because it is what determines the weights of each of the particles, which then determines what happens in the resampling step. It is extremely important to ensure the sensor models accurately reflect the behavior of the sensor including potential errors.

Distance Sensor:

Implementation concerns:

The distance sensor is a Time of Flight(ToF) based LIDAR sensor. This means that it sends out laser pulses and measures the response time of the signal. This provides a very narrow field of view and long range, however it also means that the measurements that we get from the sensor are noisy. Additionally these sensors can detect any object in its way in the field as well. Another consideration for the distant sensor is that it has a maximum distance of about 2 meters before it stops reading distances from the wall effectively.

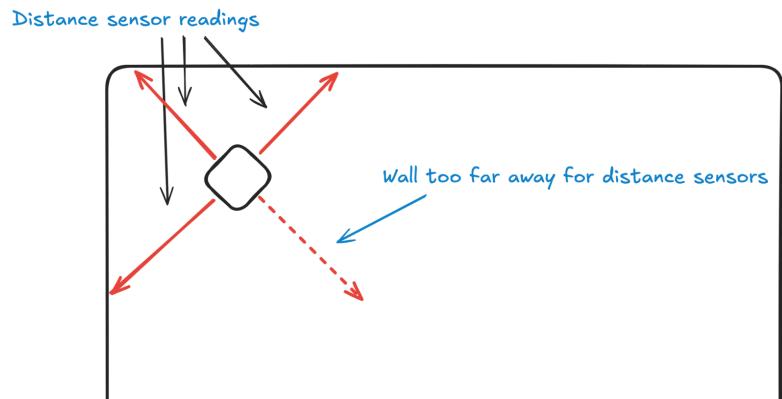


Measurement confidence:

On the VEX website for the distance sensor it states “Below 200mm the accuracy is approximately +/-15mm, above 200mm the accuracy is approximately 5%.” To interpret this into our filter we have to make a probability function using this information. Generally when statistics are listed in this way it is the 95% confidence interval which is about 2 standard deviations away from the mean reading for the sensor. Therefore the standard deviation of the distance sensor readings when the distance is in the range 200mm must be close to 2.5%. We don’t need to spend much time near the wall so we do not need to consider this in our standard deviation. This means that we can use a normal PDF with this standard deviation to determine the confidence. However this is not the only value the sensor provides. We can also take advantage of the “confidence” and “size” variables. On the website the size can be used to determine if the sensor is detecting something such as the wall. The confidence is a value from 0-64 giving the “confidence” of the distance sensor in the measurements. For the time being we will just divide the standard deviation by the confidence over 64 to properly represent the errors in measurement.

Prediction:

For a normal PDF comparison to work we must calculate the predicted value for the distance sensor at the locations in the field. To do this we will trace rays from the sensor’s location on the robot at the particle to the walls of the field. To do this we will use ray-line intersections on each of the walls. We will then take the shortest measurement from each of the lines and compare that result to the measured value.



Line Sensor

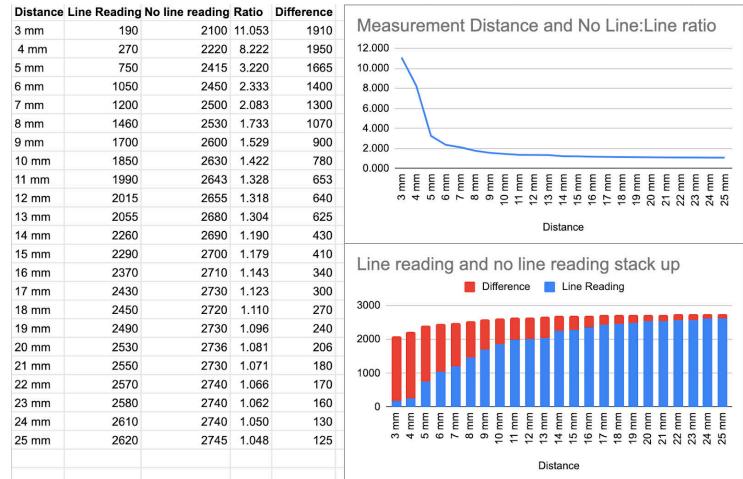
Description:

The line sensor uses an infrared emitter and then measures the reflection from the field tiles. When the sensor is over the field tapes it changes the value and is very easy to reliably detect when the sensor is over a line on the field. We can use this in the particle filter to weight the particles which match the current sensor readings with where the particle is predicted to be. In the field from VEX they supply us with the exact locations of these lines. In the illustration on the left it shows when the line sensor reads a line where the probability distribution will be the highest in green. On the other hand, when the sensor does not read a line it is more likely to be in the red lined parts of the field.

Measurement confidence:

For the line sensor confidence we don't have a large number of values to read from or significant knowledge about the error characteristics. We will have tuning variables to determine small extraneous probabilities when the sensor is getting false positives or false negatives.

Another key consideration is how far the line sensor is from the tiles. The closer it is to the tiles, the better the difference will be from the no tile to the reflection. To ensure that we have an optimal distance from the tiles we measured the response on and off the lines at distances from 3-25 mm in intervals of 1 mm to find a reasonable range of values to put the sensor at to get optimal readings. We then plotted the data as a ratio between the higher value (Off the lines) and the lower value (on the lines).



For any sensor reading we want the readings to be significantly different so they can easily be differentiated with a simple threshold value in software. A decent ratio for this would be above 1.5 high value:low value. In our graph it is clear that any distance above the ground of 3-9 mm can have these easily distinguishable differences, which is clearly demonstrated in the graph on the bottom right.

Prediction:

For the prediction we will use distance from a point to a line for each of the field tapes. If the location of the sensor is close enough to a line we determine that we should be over the line and compare the predicted and measured value. However, there is no strong probabilistic basis for what these values should be. To find adequate values we decided that we should have a different probability for each possible case of the predicted and measured values in a match statement. We chose high probability values of 0.9 for each of the cases where they matched, and 0.1 for the cases where there were errors somewhere in the process.

```
match (measured, predicted) {
  (true, true) => Some(0.9),
  (true, false) => Some(0.1),
  (false, true) => Some(0.1),
  (false, false) => Some(0.9)
}
```

Game Positioning Sensor

Description:

The GPS sensor uses computer vision and accelerometers, with a kalman filter for sensor fusion. This sensor provides a global source of localization information as a pose(position and orientation) of the sensor relative to the GPS field tapes on the side of the field. However, this sensor is prone to errors when the robot is moving, because of the lag in vision processing. Additionally because of vision artifacts it is often known to have inaccurate sensor readings even when staying in the same spot, so we must account for this issue in our particle filter.

Measurement confidence:

To predict the p-value for this sensor we will use a normal distribution with the distance from the mean being the only input. This sensor also has an error prediction, given in the Mean Square Error(MSE) of the sensor pose prediction. This value does not give us the full belief of the internal kalman filter, which is represented as a covariance, but we can still use the error value in our filter. Additionally, in the .status() flag from the sensor it contains a bit (0b0000 0100 0000 0000) that is true when the camera is unable to get a position reading from the GPS field tapes. We will implement this in our filter by ignoring the values from this sensor when the flag is high, as these inaccurate predictions could cause a divergence of the probability belief from the ground truth.

Prediction:

There is no prediction necessary for this sensor as it gives us a global location of where the robot is and we can compare the location prediction from the sensor to the current particle position without a prediction at the current location.

Conclusion

The implementation of the sensor models is key to ensure a reliable and accurate particle filter that will be able to work in a variety of scenarios. The next step for the particle filter is full integration testing of the particle filter on a real field with realistic robot conditions. Another implementation detail we need to consider is a “ground truth” source for location on the field, which we can investigate in another entry. Based on these sensor models we desire the following sensors on the design:

- 3 Distance sensors (pointed at the wall)
- 1 Line sensor (0.12"-0.35" above the ground)
- 1 GPS sensor (in line with GPS wall pattern)
- 1 AI vision sensor (pointed backwards to detect mobile and alliance stakes)
- 1 Inertial sensor (mounted on vibration dampeners, used for orientation)

08/17/24

Design: CAD Day 1 (C.2.1)

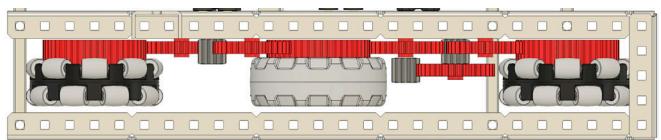
Designed by: Carl

Witnessed by: Alex

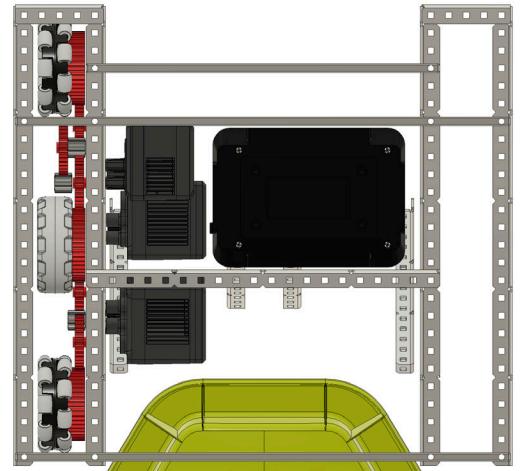
Witnessed on: 3/18/24

Goal: Begin work on the drivetrain and lift uprights.**Drivetrain:**

In order to fit our robot in the desired 14" by 14" size, we decided to go with a chassis 26 holes long, allowing for an extra inch for the intake or back clamp. The drivetrain width is dictated by the mobile stake which requires a gap of at least 9" or 18 holes between the drive rails. Because we are using the antistatic wheels, we can achieve a gap of 3 holes between the drive rails.



In terms of the gearing, we are using a ratio of 36:48 on 2.75" wheels as we [previously discussed](#) (Pg. 122-125), but because of the short length we wanted to achieve, it was necessary to offset some of the gears. These floating gears will get supported by polycarbonate but we will CAD these parts later as they are very time consuming to change. We will be using two high strength shafts as crossbars because of their high strength and very small size that allows us to put them under the drive gears. Additionally, the back shaft will serve as a leverage point for the back clamp. These offset gears also allow for easier integration of the 5.5W motors, which spin at 200 RPM compared to the 600 RPM of the 11W's. This means that we need to gear the 5.5W 3:1 before connecting it to the rest of the drive. We decided to do this through a compound gear ratio where the 5.5W drives a 36T gear at 200 RPM, which then turns a 12T linked to the rest of the drivetrain at 600 RPM. To fit this in the drivetrain, there is a freely rotating 12T idler gear on the 5.5's shaft to transfer the power to the front wheel.

**Lift:**

Almost an exact recreation of the original concept drawing for the robot, the lift was designed using 2x1x26 C-Channels as uprights and 1x1x17 L-Channels for the arm for reduced weight. These uprights will be attached to the chassis with polycarbonate parts to be designed later.



08/18/24

Design: CAD Day 2 (C.2.1)

Designed by: Carl

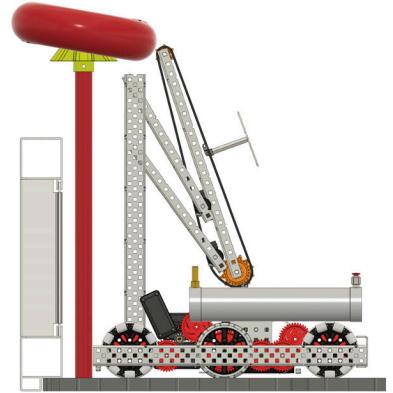
Witnessed by: Alex

Witnessed on: 8/19/24

Goal: Complete the top stage of the intake and optimize its geometry for mobile stakes, wall stakes, and alliance stakes.

**Top Stage Intake:**

For the top stage, we will be using a chain with 4 double sided hooks running between two sprockets. For the top sprocket, we went with a 6T to keep the height at a minimum, and at the bottom we used a 12T for a smoother transition of the rings between the different stages.

**Arm Geometry Fine Tuning:**

While the arm closely matches the concept drawing, it couldn't be exact due to the 0.5" hole increments. By adding HS pillow blocks to the back of the intake, we can precisely adjust the spacing in both CAD and on the physical robot. Additionally, standoffs, which have a similar diameter to HS shafts but are much lighter, are ideal for attaching the lift arms to the intake. When building, we will also add a hard stop to the lift to prevent it from lowering too far and allowing the lift motor to rest.

Bottom Stage Intake Polycarbonate:

Roughly designing the intake polycarbonate was crucial for optimizing the arm's geometry to get a smooth transition of the ring between stages. The polycarbonate is a simple rectangular shape, matching the width of a ring, with a cutout for the top stage's hooks. It is attached at the bottom to a high-strength shaft and at the top to a standoff, with zip ties for minimal weight.



08/19/24

Design: CAD Day 3 (C.2.1)

Designed by: Carl

Witnessed by: Alex

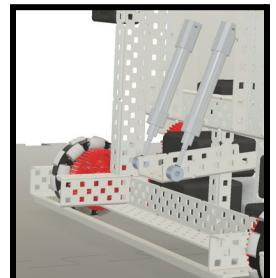
Witnessed on: 8/20/24

Goal: Finish the goal clamp, intake, lift gearing, and mount the brain and battery.



Goal Clamp:

The goal clamp utilizes the already existing HS shaft for the lever point of the goal. Five holes behind the HS shaft we have a 2x2x20 L-channel to hold the goal as the clamp pushes the lip of the goal into it. The L channel also gives us a convenient place to attach 1x1 L-channels vertically for the clamp's 3x1x7 C-channel to pivot on. The pistons chosen have a 25mm stroke and are attached to the clamp with shaft collars and to the inner uprights with screw joints. These 3x1x18 C-channel inside uprights are positioned in such a way that they can also be used as support on the inside of the lift.



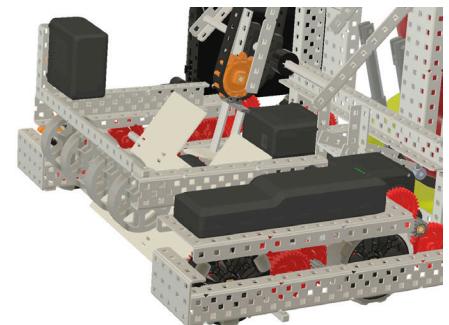
Bottom Stage Intake:

For the roller, we opted to use four 2" flex wheels because of their light weight, small size, and reduced risk of entanglement. The motor is mounted directly above the intake because it is the only place that is somewhat protected from other robots and out of the path of the rings. The arms of the intake pivot on pillow blocks mounted on 5x1 C-channels above each side of the drive and are linked at the front with a drilled HS shaft. The motor is currently linked to the roller via chain and sprockets, but we hope to implement a more robust solution in the build.



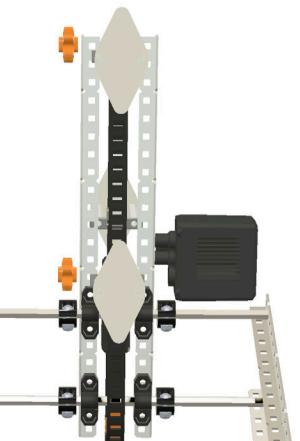
Brain & Battery Mount:

A low center of gravity is very important so we don't incidentally tip over during a match. Because the brain and battery are some of our heaviest components, we mounted them as low as possible without adding more structure. Additionally, ensuring the brain screen and power button were easily accessible was a key factor in determining their placement.



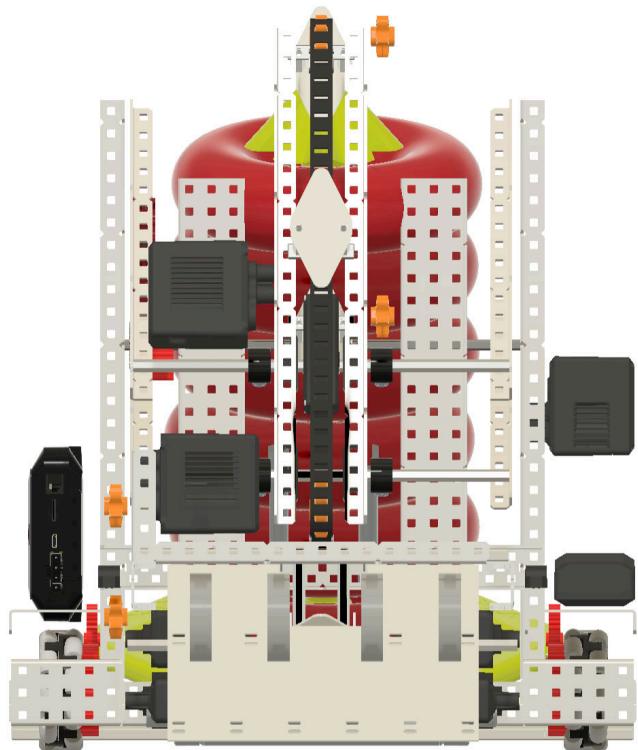
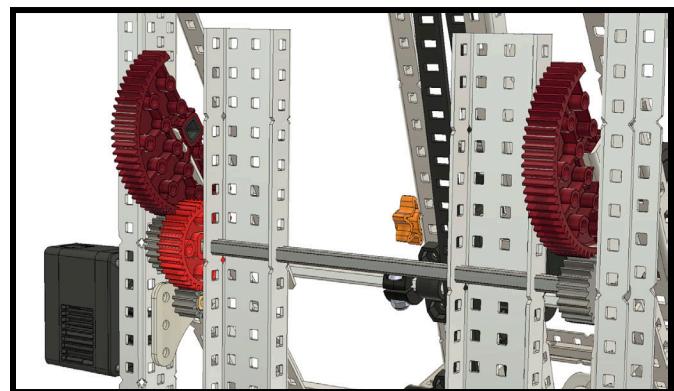
Top Stage Motor Mount:

The motor for the top stage is mounted directly to the 1x1 L-channel as low as possible. The power is then transferred to the top sprocket on the second stage by a simple chain. Inside, we have a rotation sensor so the intake knows where it is when the code starts.



Lift Gearing:

The lift uses compound gearing with two consecutive 1:3 ratios, resulting in a total reduction of 1:9. Because of the rings on the mobile goal and the intake hooks, it was impossible to put any kind of bracing in between the two lift halves, which would typically result in a very weak lift. To solve this issue, we use a high strength shaft after the first 1:3 reduction to transmit power to the other half of the lift, ensuring that both sides stay in sync. Furthermore, the 72T gears mounted to the bottom arms needed to be cut in order to not interfere with the upper arms.



08/20/24

Design: CAD Day 4 (C.2.1)

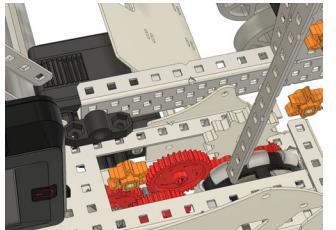
Designed by: Carl

Witnessed by: Alex

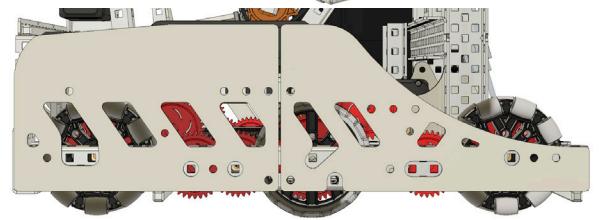
Witnessed on: 8/21/24

Goal: Add front arm and design polycarbonate.**Front Arm:**

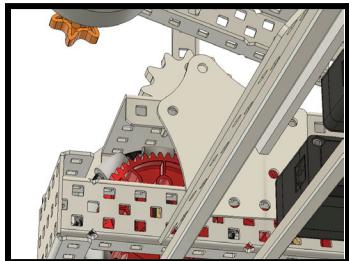
Integrating the arm into the robot was challenging due to the desired mounting point and limited motor space. We mounted the pivot point on extensions from the side panels to fit the arm, but the intake prevented direct motor attachment. A string drive was considered but dismissed due to its complexity and entanglement risk. Instead, we mounted the 5.5W motor behind the intake polycarbonate and chained it to the arm, requiring a custom polycarbonate sprocket screwed directly to the arm.

**Polycarbonate Parts:****Side Panels:**

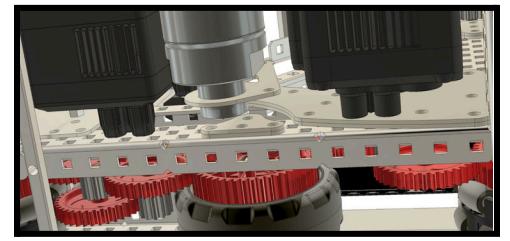
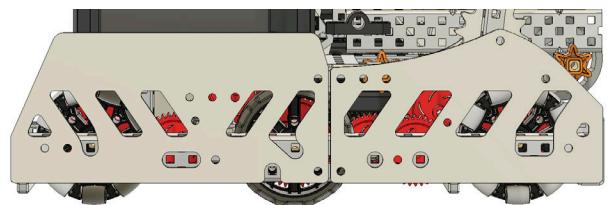
- Protects the brain and battery
- Divided into two parts for maintenance and accessibility
- Cutouts for quick access to the drivetrain
- Support for floating drivetrain gears

**Inside Motor Mounts:**

- Holds 11W drive motors
- Mounting for lift uprights
- Mount for triangle bracing

**Forward Inside Polycarbonate:**

- Intake standoff mount
- Inner mount for front arm
- Forward idler gear mount

**Other Parts:**

- Pneumatics tank mount (light solution to attaching pneumatics tank in the ideal spot)
- Center wheel offset (double stacked poly that drops the center traction 1/16")
- Top triangle mount (connects the lift uprights to the top of the triangle bracing)
- Custom sprocket (for front arm)
- Hooks (for intake top stage)
- Intake ramp (polycarbonate back of the intake's first stage)

08/21/24

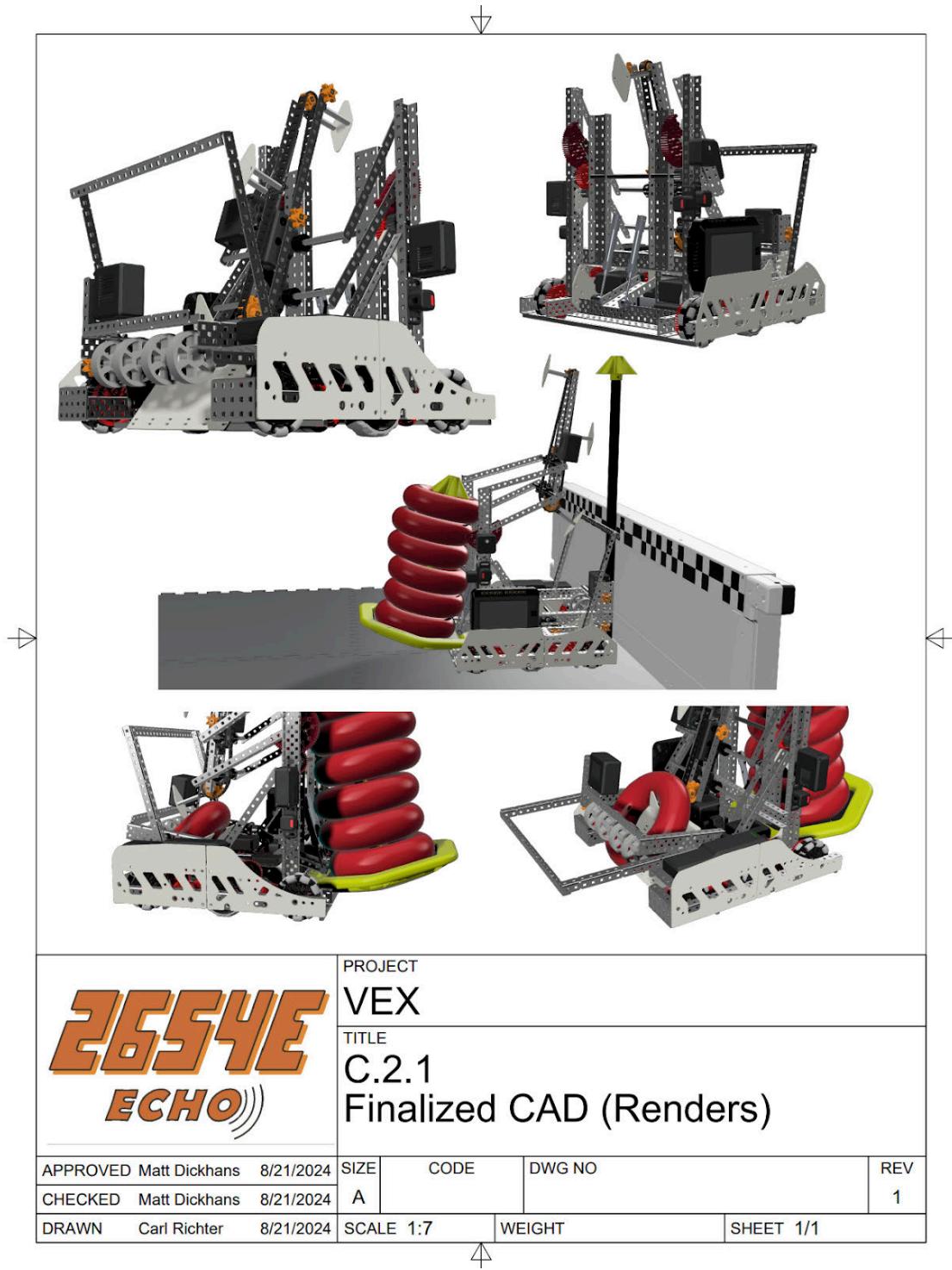
Design: CAD Day 5 (C.2.1)

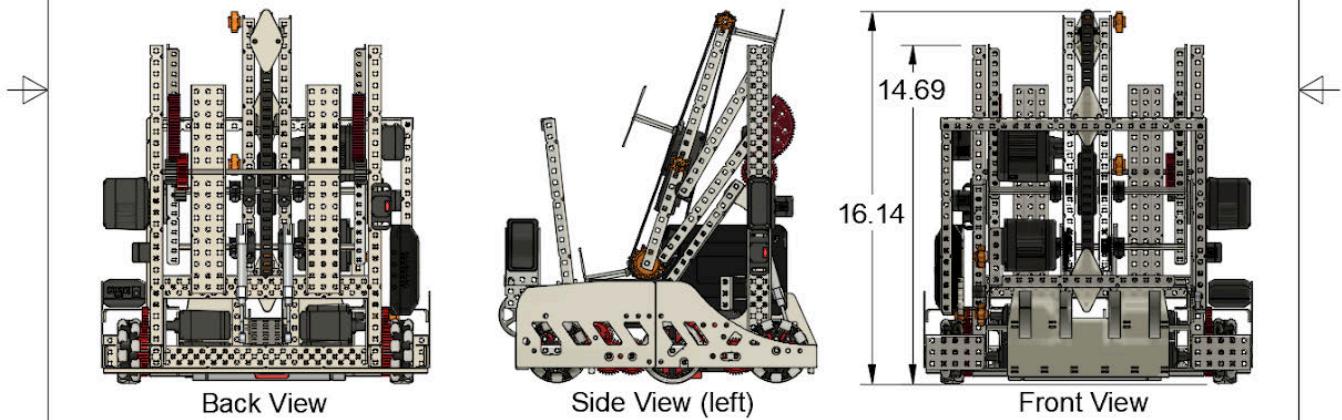
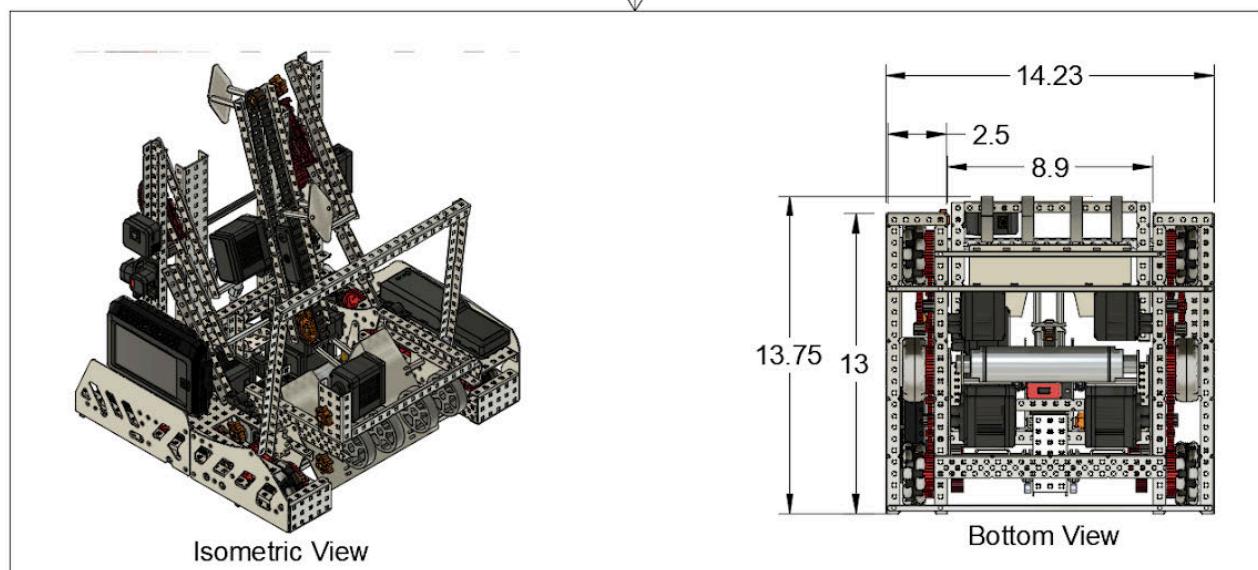
Designed by: Carl

Witnessed by: Alex

Witnessed on: 8/22/24

Goal: Add sensors and measure their position relative to a center point. Create a polycarbonate layout for laser cutting. Finalize CAD for build.





2654E
ECHO

PROJECT
VEX
TITLE
C.2.1
Finalized CAD (Technical Drawing)

APPROVED	Matt Dickhans	8/21/2024	SIZE A	CODE	DWG NO	REV 1
----------	---------------	-----------	-----------	------	--------	----------

CHECKED	Matt Dickhans	8/21/2024
---------	---------------	-----------

DRAWN	Carl Richter	8/21/2024
-------	--------------	-----------

SCALE	1:7	WEIGHT	SHEET	1/1
-------	-----	--------	-------	-----

PARTS LIST

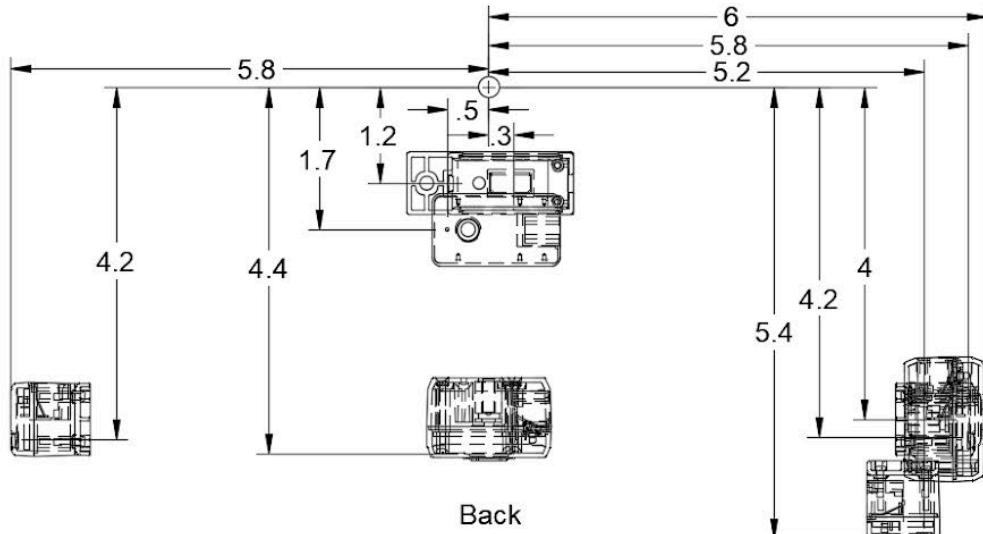
ITEM	QTY	PART NUMBER
1	4	HOOK POLY
2	1	INTAKE POLY
3	4	UPRIGHT BASE POLY
4	8	WHEEL OFFSET POLY
5	2	FRONT INSIDE GEAR OFFSET POLY
6	1	SIDE PANEL RIGHT FRONT POLY
7	1	SIDE PANEL RIGHT BACK POLY
8	1	SIDE PANEL LEFT FRONT POLY
9	1	SIDE PANEL LEFT BACK POLY
10	4	12T POLY SPROCKET
11	4	TANK HOLDER POLY

PROJECT
VEX
TITLE
C.2.1 poly layout
Arrangement for Laser Cutting

APPROVED	Alex Dickhans	8/21/2024	SIZE	CODE	DWG NO	REV
CHECKED	Matt Dickhans	8/21/2024	A			1
DRAWN	Carl Richter	8/21/2024	SCALE	1:4	WEIGHT	SHEET 1/1

Top View

Front



Back

Sensor Distances (Inches)		
Sensor Title	Y Offset	X Offset
Line	-.3	-1.2
Inertial	.5	-1.7
AI	0	-4.4
GPS	-6	-4
Distance (right)	-5.8	-4.2
Distance (back)	-5.2	-5.4
Distance (left)	5.8	-4.2



PROJECT
VEX
TITLE
C.2.1 (Sensors)

APPROVED Alex Dickhans 8/21/2024

SIZE

CODE

DWG NO

REV

CHECKED Alex Dickhans 8/21/2024

A

DRAWN Carl Richter 8/21/2024

SCALE 1:2

WEIGHT

SHEET 1/1

08/22/24 Build: Day 1 (R.1.0.0)

Designed by: Carl

Witnessed by: Alex

Witnessed on: 8/23/24

Goal: Prepare custom parts for the drivetrain and begin assembly.

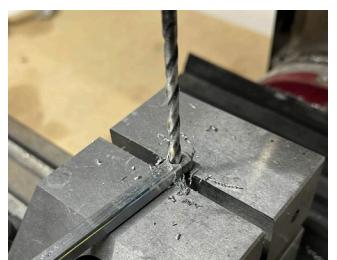
Shortened Motor Caps:

To fit the 11W drive motors between the inner drive rail and the 1x1 L-channels supporting the back clamp, we reduced their depth by about $\frac{1}{8}$ ". We removed the steel inserts from the motor caps, sanded the caps on a belt sander, and then sanded the back of the steel inserts to size. Since the lip holding the inserts was removed, the motors now need to be secured with screws inside the motor cap going outward.



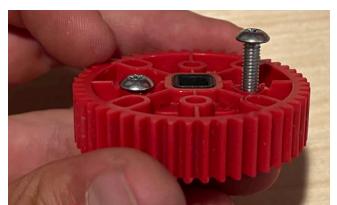
Drilled High Strength Shafts:

A key element of our design is accurately cut and drilled HS shafts. We precisely measured the position of each hole, punched, and drilled them out using a drill press.



Drive Gear Modifications:

To fit our wheels and gears within the 1.5" gap between the drive rails, they must be attached directly without spacing. The screws connecting the wheels need to be flush with the gear, hence the slight recess in the gear (pictured left). Additionally, the gears mounting to the omni wheels require a bevel to accommodate the rollers.



Drivetrain Progress:

Thus far, we have not made any deviations from the CAD, with the exception of adding screws, nuts, bearing flats, and other hardware that was not added to the CAD. While building, we ensured perfect squareness of the structure to reduce future issues. Furthermore, we also utilized a build technique commonly referred to as "boxing" (bottom right) to reinforce high stress points and prevent twisting of the drive rails.



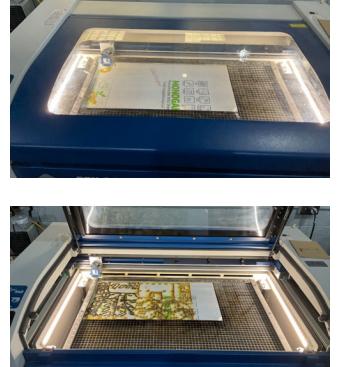
08/23/24 Build: Day 2 (R.1.0.0)

Designed by: Matt	Witnessed by: Carl	Witnessed on: 08/25/24
-------------------	--------------------	------------------------

Goal: Cut out polycarbonate and continue building the drivetrain and mount lift uprights.

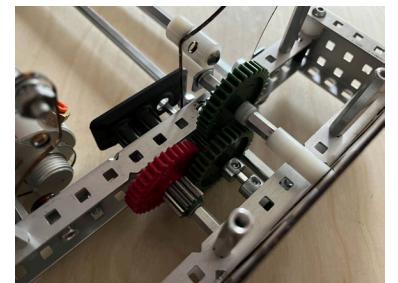
Laser Cutting:

For laser cutting, the first step is adding all of the polycarbonate parts into one document and arranging them onto a 12"x24" plane ([Pg. 147](#)). The layout can then be exported as a DXF to Adobe Illustrator for laser cutting. Once imported into the software, we change the speed, power, and frequency, ensure scaling and position is correct, and get the document ready to cut. We then cut the polycarbonate and clean it off using soap and water.



Screw Joints:

To minimize the amount of friction in the drivetrain, we use screw joints on non-driven gears and wheels. In a screw joint, the gear/wheel with circle inserts spins around a screw instead of a shaft to reduce friction. To do this, we use one 0.5" standoff attached to the inside of the drivetrain, and a screw from the outside rail that gets locked in with another standoff. Screw joints also have the added benefit of physically attaching the drive rails to each other, adding significant strength.



Pneumatic Tank Mounting:

To keep our center of gravity as low and as centered as possible, we mounted the tank at the bottom and center of our robot using polycarbonate parts from the CAD.



Polycarbonate and Shoulder Screws:

With the extensive design and CAD process for this robot, we have not had to deviate from the CAD. This includes the polycarbonate that we designed which is nicely integrating into the robot. To ensure that these pieces are in the correct spot we use shoulder screws to ensure that all the holes are centered and the robot is square.



Progress:

Today, we finished framing the drivetrain, uprights, and attached essential polycarbonate, allowing us to start adding other components to the robot, including the lift, lower intake, and back clamp.

08/24/24 Build: Day 3 (R.1.0.1)

Designed by: Matt, Carl

Witnessed by: Alex

Witnessed on: 8/25/24

Goal: Build and test back clamp, build and test the front arm and add lift gearing.

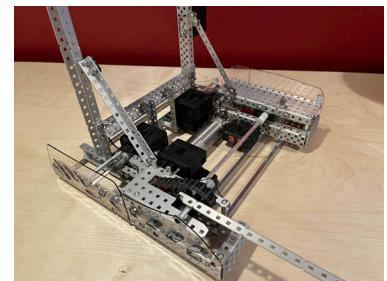
Front Arm:

In order to make the front arm gearing fit, the front right 5x1 needed to have some material removed. The uppermost cut is for the 12T sprocket and the arm, the center pocket fits the 6T driven by the 5.5W, and the bottom cutout is for the brain.



Upright Triangles:

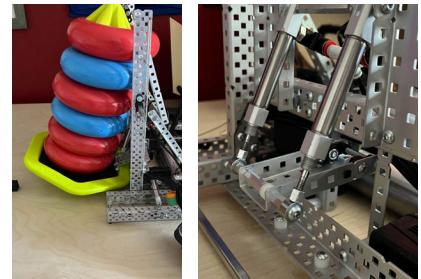
To maximize the strength of the uprights, we use triangle bracing to support them. We have a 1x1 L-channel connected to the top of the upright, going down to the 5x1 C-channel over the drive, creating a triangle providing strength to the upright. To help achieve perfect alignment, we used shoulder screws to attach the triangle bracing to their respective polycarbonate parts.



Back Clamp:

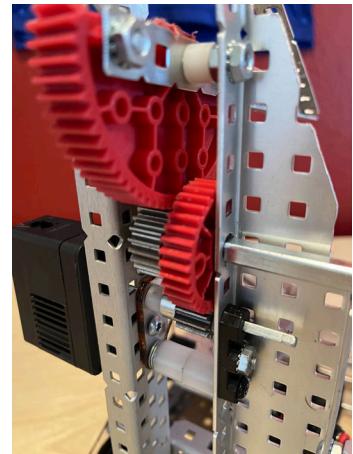
The back clamp was built exactly to the CAD with only a few minor changes:

1. The use of hardware and fasteners to attach everything together. We also boxed all three of the pivots to prevent any bending under stress.
2. The addition of two 0.25" spacers on the 2x2 L-channel to go under the lip of the goal, effectively locking it in place when it is tilted.
3. Lowering of the back HS shaft by 0.125" to decrease the tilt angle to be more similar to the CAD.



Lift Gearing:

The lift was also executed following the CAD very closely, with the only changes being slightly thinning the 36T gear, and adding steel 1x8 plates to attach the bottom arms to the 72T gears.



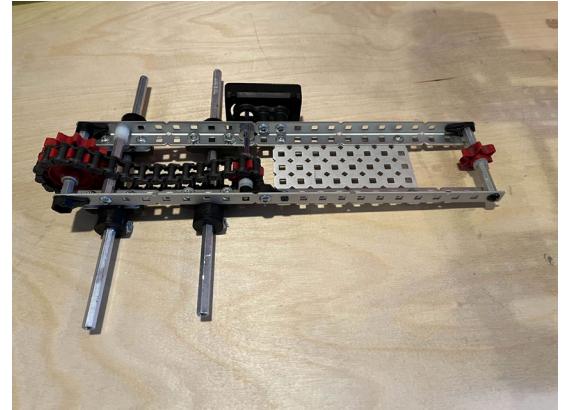
08/25/24 Build: Day 4 (R.1.1.0)

Designed by: Matt	Witnessed by: Alex	Witnessed on: 8/26/24
-------------------	--------------------	-----------------------

Goal: Build and Test Top and Bottom Stages of the Intake.

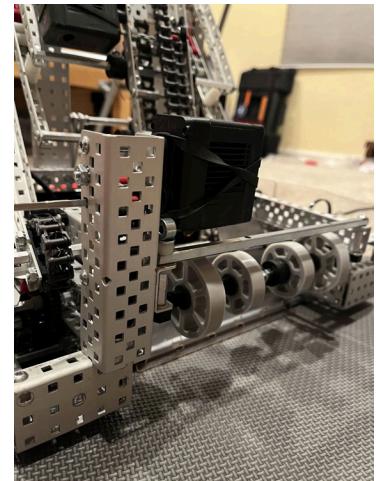
Intake Top Stage:

Instead of mounting the driven sprocket outside the 1x1 structure, we opted to mount it on the inside, forcing us to build the structure a hole wider. While changing the structure, we also decided to add a 5x8 plate to help us square the 1x1s. To do this, we connected the original 18T at the bottom to a second 18T using a circular insert, allowing the assembly to spin freely on the screw; the 6T was simply put in between the 1x1s. Another thing we changed was adding more support to the top by adding some sheet metal.



Intake Bottom Stage:

To protect the chain from breaking in vigorous interactions with other robots, we used a U-channel with both sprockets and chain enclosed inside. This differs from the CAD where we had a 3 wide C-channel to hold the chain. After some testing with the bottom stage, we found that the pick up was not working due to having the flex wheels too close to the intake ramp, resulting in the flex wheels pulling the top of rings while the bottom of the rings would get stuck on the ramp as shown in the image. To fix this, we moved the intake forward onto the polycarbonate holding the front arm. We also took this opportunity to switch the chain for a lower friction, more robust gear train.



08/26/24

Build: Day 5 (R.1.1.0)

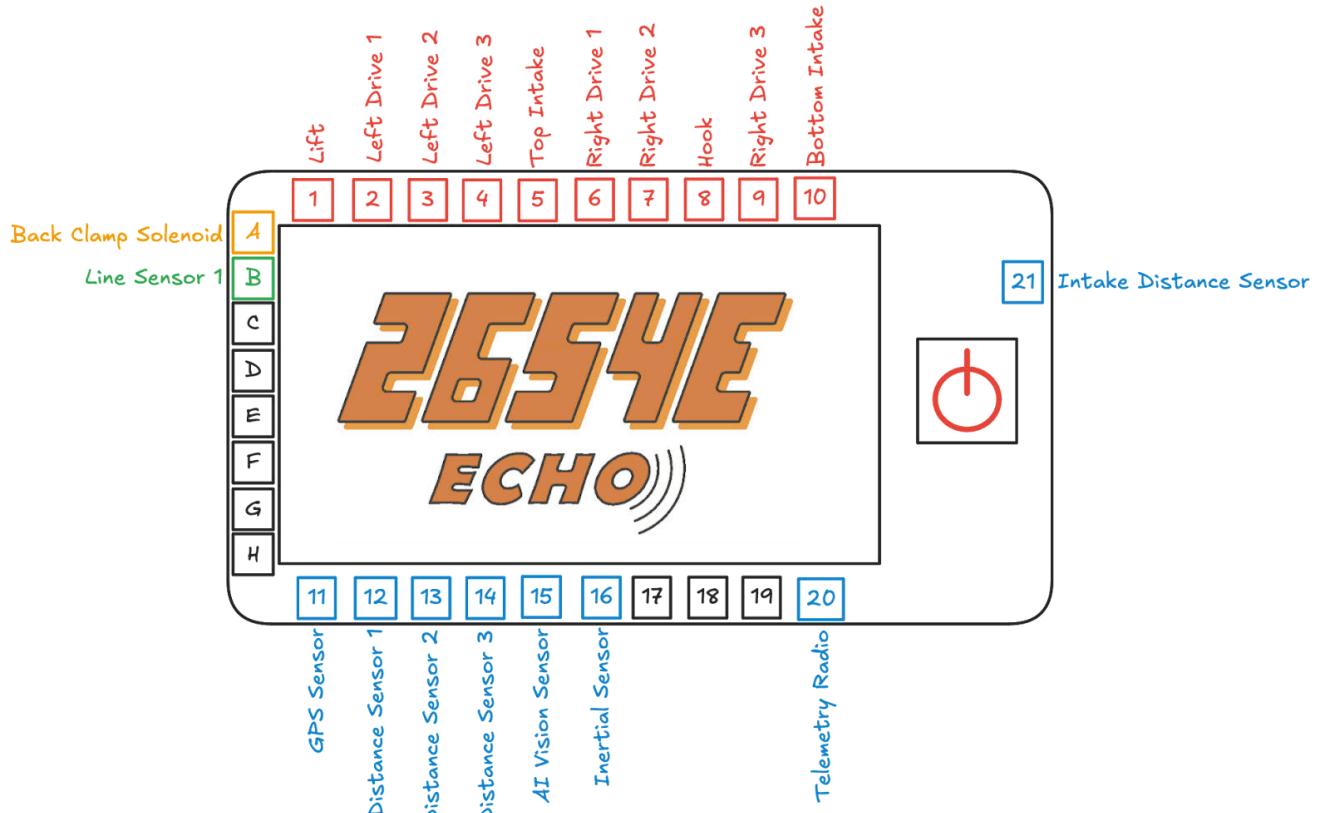
Designed by: Alex, Carl

Witnessed by: Matt

Witnessed on: 8/28/24

Goal: Wire Motors and Pneumatics, Add Sensors and Finish up Smaller Details.**Wiring:**

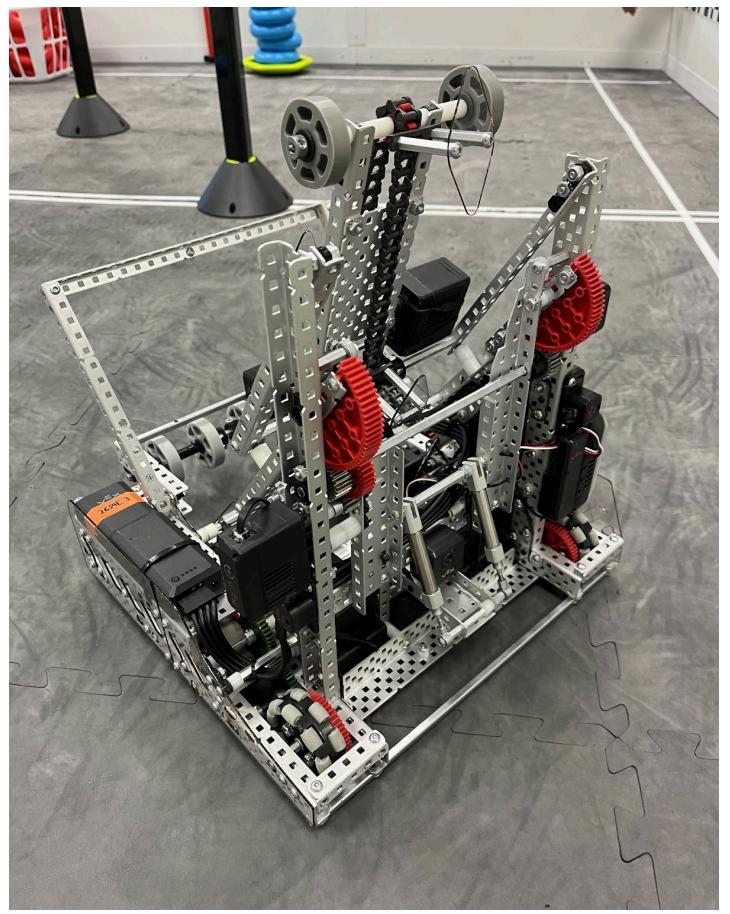
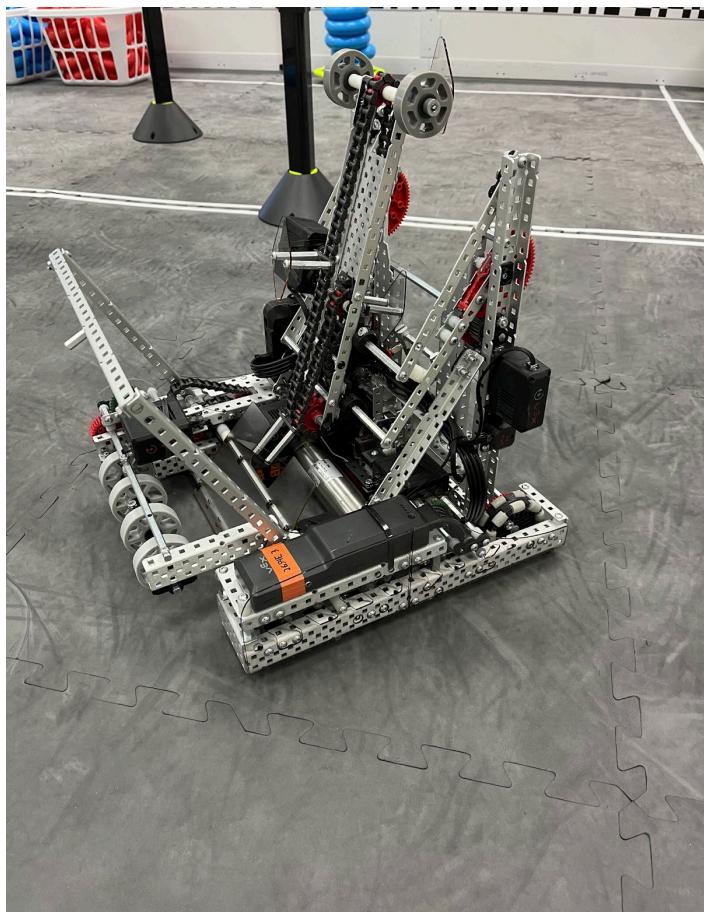
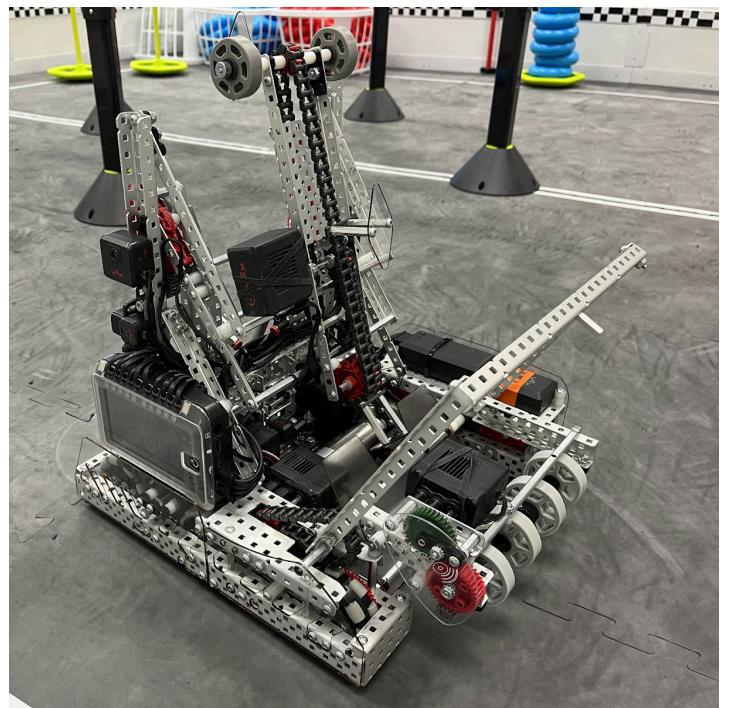
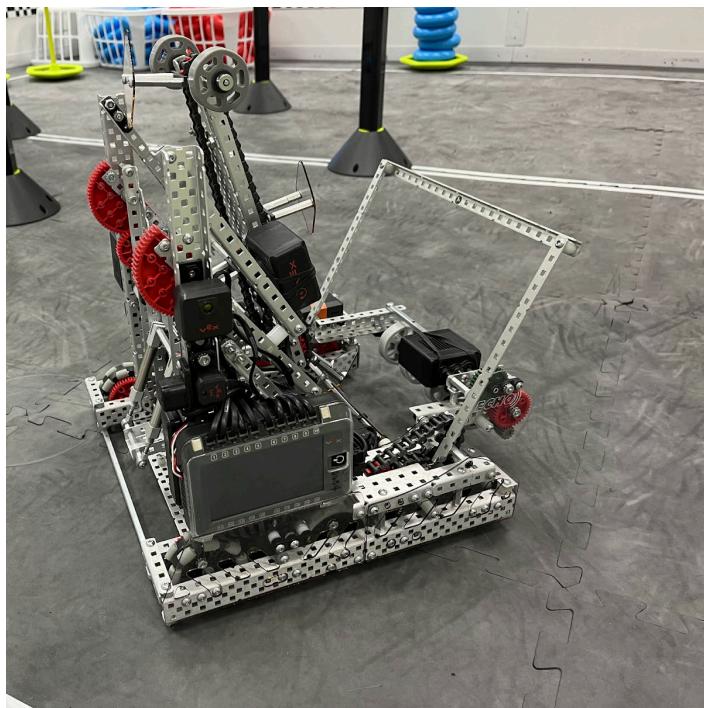
To simplify we made this diagram, and there were no deviations from it in the actual wiring. Another thing to note is that we prioritized short wire routing for better cable management.

**Sensors:**

We mounted the 3 distance sensors and 1 GPS to the robot exactly as CADed. We put the Inertial Sensor on rubber links to reduce vibrations. The vibration reduction is crucial because it reduces the amount of noise that the IMU receives from other mechanisms on the robot such as the intake and the lift moving. We found in past years testing that this damper system can reduce inconsistent drift from about 2-3 degrees per skills run to less than a degree per skills run which is high enough accuracy for our needs.



Finished Build Images:



08/27/24 Testing/Identify: Problems (R.1.1.0)

Designed by: Carl	Witnessed by: Alex	Witnessed on: 8/28/24
-------------------	--------------------	-----------------------

Goal: Compile a list of problems with our robot as we test it on the field. Identify what works well.

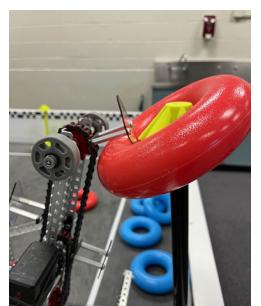
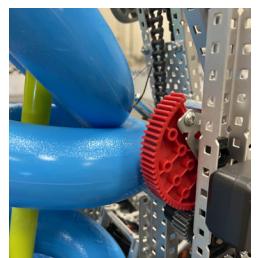
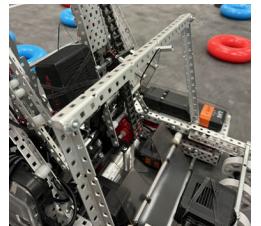
High Functionality:

1. Scoring on mobile stakes
 - a. Very efficient, limited jams
2. Drivetrain
 - a. Good acceleration, high resistance to sideways pushing, stable base, no overheating



Inadequate Functionality:

1. Pneumatics tank ground contact
 - a. A small oversight with the original tank holder poly resulted in not enough tolerance; this caused slight contact between the tank and the tiles.
2. Front arm arm can get stuck on intake
 - a. Occasionally the arm wanders from its intended position, even with motor hold, as the acceleration forces from the drivetrain are too extreme; this results in it interfering with the intake and causing a jam.
3. Intake hooks can get stuck on bottom rung
 - a. When the hooks on the top intake stage point directly up, they hit the bottom rung, causing the robot to temporarily get stuck.
4. Mobile goals are hard to grab
 - a. There is currently nothing to force mobile stakes into the correct position for grabbing.
 - b. Mobile stakes are designed to stop at the 2x2 L-channel, but constantly go over it.
5. Rings get stuck on lift HS shaft
 - a. Relatively rare issue and not catastrophic.
 - b. The intake can sometimes snag rings on the mobile post if they are on top of the HS shaft.
6. Scoring on alliance and wall stakes is extremely unreliable
 - a. The robot can not effectively score on any wall mounted stakes.
 - b. We believe this is caused by rings not having enough time to completely fling around as the wall mounted stakes are higher than the top of the intake.



08/27/24

Build: Minor Improvements (R.1.1.2)

Designed by: Carl

Witnessed by: Alex

Witnessed on: 8/28/24

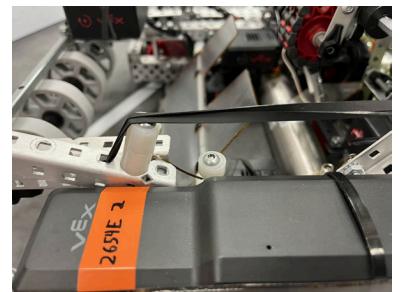
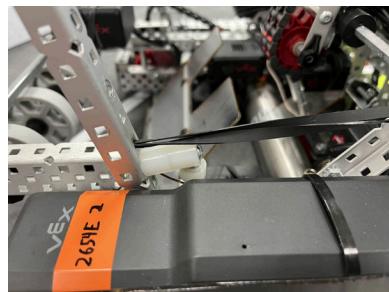
Goal: Address the problems with the back clamp and front arm.

As mentioned on the previous page, we have several issues that need addressing; we want to prioritize the goal clamp and front arm because they are both big issues which we believe can be easily solved.

Front Arm Hardstop:

To prevent the front arm from wandering into the intake top stage, we simply added a hard stop.

This is a very simple solution and can hold the arm in a consistent position when accompanied with a slight bit of rubber band force.



Goal Clamp Wall:

We integrated a taller wall on the back clamp to eliminate the problem of the goal sliding too far back, however, we still need to add more guides to fully align the goal. We want to utilize polycarbonate for these guides so they can conform to the goal as much as possible; we will need CAD and laser cut these parts in order to implement them.



08/27/24 Testing: State Machine/Programming Environment

Designed by: Alex	Witnessed by: Carl	Witnessed on: 08/29/24
-------------------	--------------------	------------------------

Goal: Reanalyse our decision of programming environment after programming the code for this robot.

When programming the first robot we found that there were many issues with the asynchronous structure we settled on, and the support for vexide. When programming with futures, the underlying type behind asynchronous code, you always have to ensure the futures are always being *awaited*. When they aren't being awaited, the async scheduler won't run the future at all. This makes it so that we can't schedule a future easily from something such as a controller button press. This made it very difficult for us to plan for all the complex sequences we needed to complete with our robot.

Problem: Vexide Stability

Vexide, the rust programming architecture we were using, is entirely community made and supported. This means that it could break at any future VEXos update and there is less stability and testing than other programming environments like PROS with C++. For the stability reasons we have decided to shift away from vexide and to PROS, because it was our second best option. With this change we will also be changing to CLion to have better language support while editing code.

State Machine

To complete the shift to PROS, we will have to port all the code from Rust to C++, which primarily includes our state machine. We wanted to take inspiration from the FRC [WPILib command based-programming](#) because it is very flexible, and works very well with command compositions, which allows compositions such as sequences for complex autonomous routines. We took inspiration from this architecture to make our own command scheduler that better fit our needs. Additionally, we made this a [public library](#) that has [extensive documentation](#) on the inner workings.

08/29/24

Identify: Wall Stake Optimization (R.1.1.2)

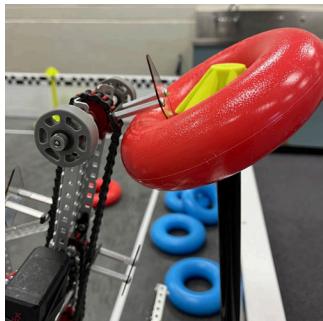
Designed by: Carl

Witnessed by: Alex

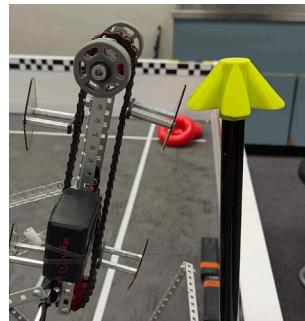
Witnessed on: 8/30/24

Goal: Identify exactly how the lift needs to be adjusted to score on wall stakes consistently and come up with 3 solutions to achieve this.

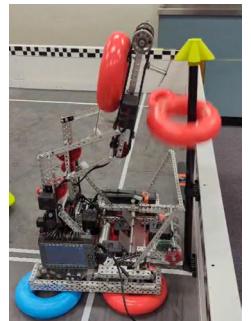
We believe the robot can't score on wall stakes because the intake is too low relative to the stake's top. To test this, we elevated the chassis, effectively giving us a 2" taller lift.



(before extra height)



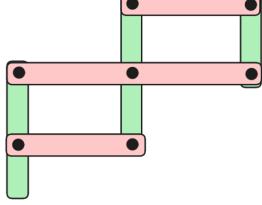
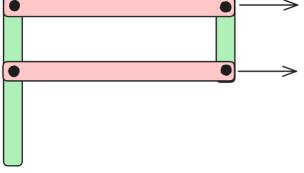
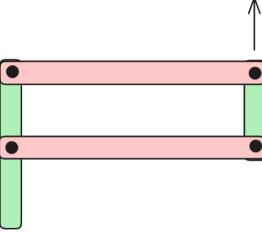
(after extra height)



(testing setup)

After testing with the robot raised, it became very apparent that this was the problem, which we further verified with slow-mo video review. To fix the issue, we will explore 3 options for raising the height of the top stage.

Possible Solutions for Scoring on Wall Stakes

6-Bar	Longer 4-Bar	Longer Top Intake Stage
Pros: <ul style="list-style-type: none"> - High reach - More vertical less horizontal movement - Can fit under bottom rung Cons: <ul style="list-style-type: none"> - Complex - Higher play in the lift - Heaviest option - Hardest implantation 	Pros: <ul style="list-style-type: none"> - Very simple - Limited slop - Lightweight - Fits under bottom rung Cons: <ul style="list-style-type: none"> - Limited options for geometry - More forward reach 	Pros: <ul style="list-style-type: none"> - Least work to implement - Lightweight - Simple Cons: <ul style="list-style-type: none"> - Would hit the bottom rung - Could make mobile stake performance worse 

08/29/24 Select Solution/Design/Build: Wall Stake Optimization (continued) (R.1.2.0)/(C.2.2)

Designed by: Carl	Witnessed by: Alex	Witnessed on: 8/30/24
-------------------	--------------------	-----------------------

Goal: Select the best option for changing the lift, design in CAD, and implement it onto the robot.

Select Solution:

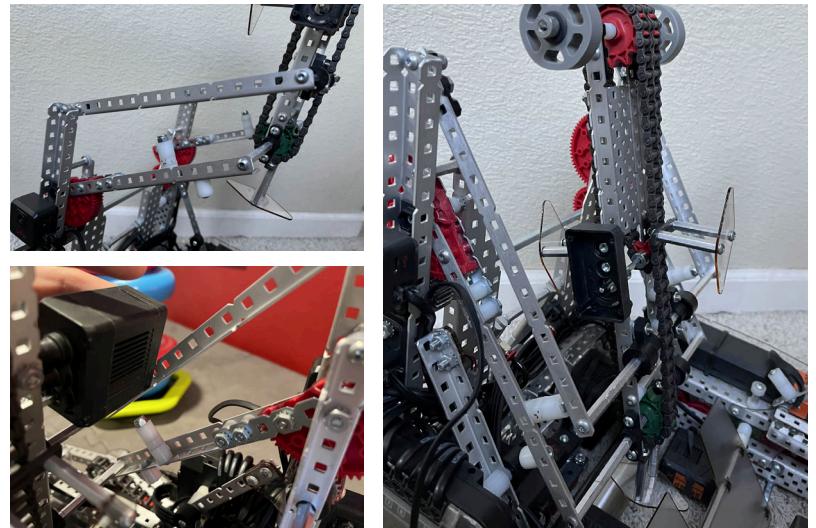
To improve the robots wall stake scoring abilities, we need to change the lift to bring the top of the intake higher. On the previous page, we came up with three ways of doing this. The first option that we eliminated was “Longer Top Stage” because it would prevent us from being able to go under the climbing structure. A longer 4-bar is much simpler and more stable than a 6-bar, with no huge disadvantages. Our main concern with a 4-bar design was the limited options for viable arm geometries, but with CAD, we hope to find one that works.

4-Bar Updated CAD:



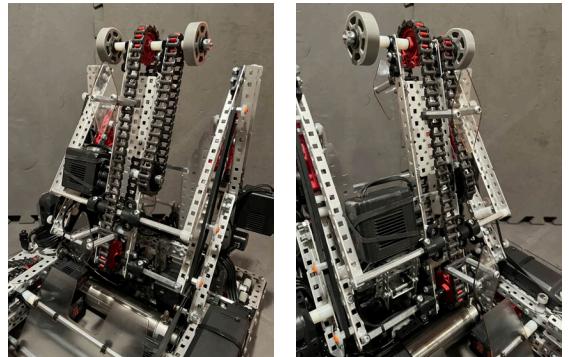
Physical Implementation:

The physical implementation of the lift mechanism was fairly simple, just re-cutting longer arms and changing the pivot points on the intake. Because the new lift geometry features a pivot point in the center of the bottom sprocket, we needed to drill it out to allow the intake to spin freely. Additionally, we could not add the motor for the top stage intake because the motor prevents the arm from going down all the way.



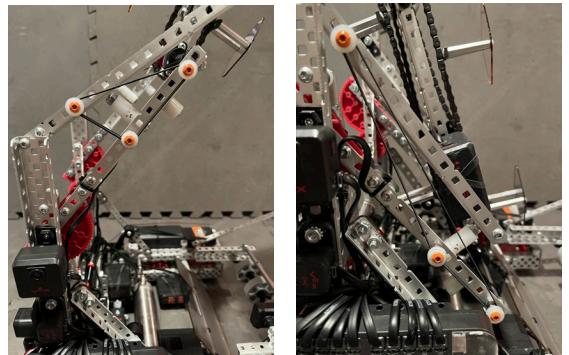
Top Stage Rebuild:

In order to mount the 11W on the top stage, we needed to fully rebuild the top stage to be at least 0.5" narrower. We also noticed when manually moving the longer lift, the top intake stage had more wobble than we would like. To solve this, we will replace the bottom standoff pivot with a drilled HS shaft. Finally, because our previous intake had more than enough torque, we decided to increase the speed by 33% to fully maximize the motors strength.



Triangle Banding:

By rebuilding the lift to be longer, we subsequently generate a higher torque on the pivot as $\tau = r_{\perp} \cdot m \cdot g$. This means that the arm takes longer to reach its top speed, which is less than ideal for scoring efficiency. To remedy this situation, we decided to add rubber bands to help cancel out the lift's weight. We found that by putting the band pegs in a triangular arrangement, the band assistance was far more even through the lift's range of motion.



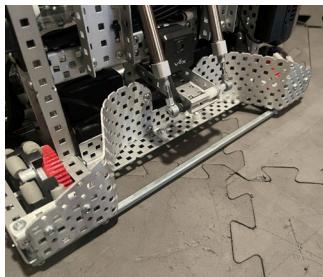
08/31/24 Build/Testing: Improving Subsystems (R.1.2.6)

Designed by: Carl

Witnessed by: Alex

Witnessed on: 09/01/24

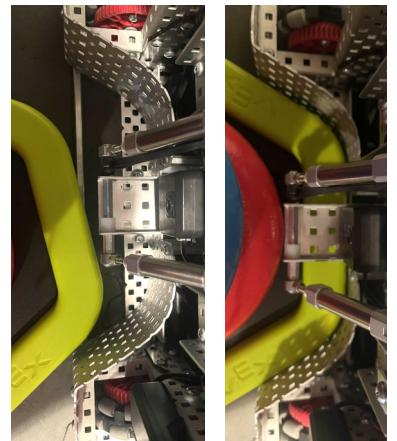
Goal: Optimize performance of all subsystems as much as possible to prepare the robot for more vigorous testing.



Goal Clamp Guides:

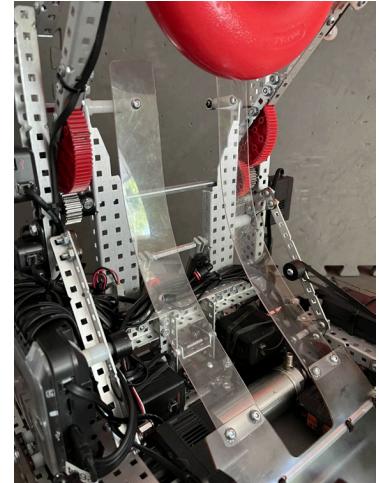
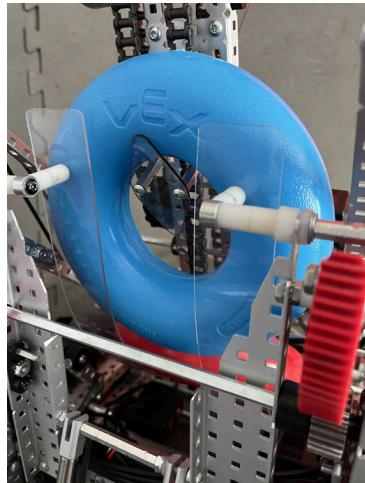
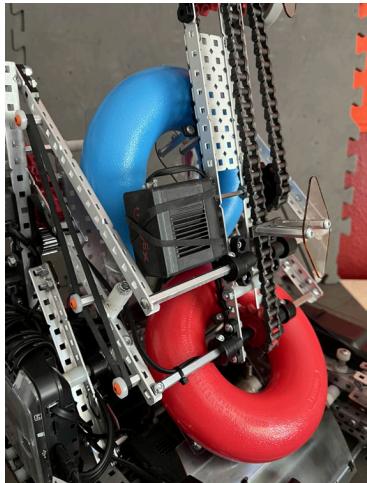
We have already made [small improvements](#) (Pg. 156) to the goal clamp, however, these modifications only restricted the goals from moving too far into the robot, but did nothing to funnel the goals into the center of the robot. For our funnels, we were originally going to

use polycarbonate because of its highly customizable shape that would allow us to create a shape to perfectly conform to the mobile goal. After further consideration, we realized we could achieve a very similar shape that was much stronger by using bent aluminum plates. These plates were added to the robot and can generally align a mobile stake perfectly, even if it's 3-4" off the robot's centerline.



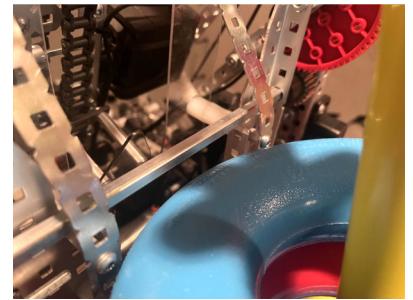
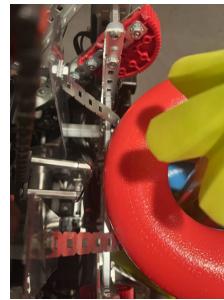
Top Stage Intake Backing:

Even when testing the 2nd loading position of the top stage in a very controlled setting with very few external factors, it was not uncommon for rings to fall off the intake and get stuck in the robot. This would have serious implications if it were to happen in a match, rendering the intake essentially useless. To eliminate the chance of this happening, we added 12x1.5" polycarbonate strips on either side of the chain/hooks. These strips were bent to conform to the ring's path as closely as possible, making losing a ring when loading the top stage very unlikely, and a catastrophic jam almost impossible.



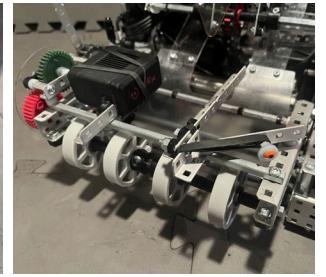
Ring Guides onto Mobile Stakes:

A problem already acknowledged on [Pg. 155](#) where the rings going on to a mobile stake would get stuck on the lift's HS shaft, causing a jam. This problem was easily fixed with the addition of two steel 0.5" wide strips between the tops of the intake backing and a screw just below the HS shaft.



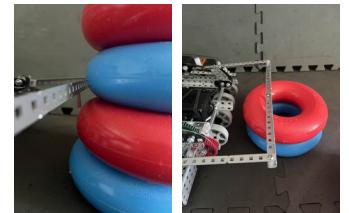
Intake Triangle Bracing, Contact Arm, & Mobile Stake Aligner:

While testing intaking, a ring would occasionally hit a hook at the wrong time and get launched out the front of the intake. This was fixed by adding a pivoting 1x1 with slight downwards banding. This keeps the rings in the correct spot allowing the rings to reliably transfer between intake stages. To help keep the intake square, we added a second triangle brace to the HS shaft, and to help line up with wall stakes, we added a polycarbonate funnel.

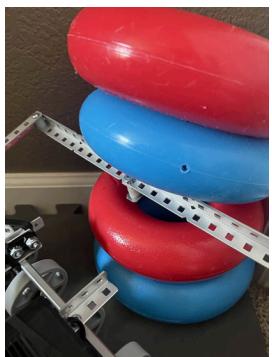


Front Arm Upgrades:

Our previous front arm only had a 0.75" standoff to pull rings out of the corner which was not effective. This was primarily due to the arm only being able to remove the top ring as the standoffs would prevent the arm from going in between the stacked rings.



To allow the arm to get rings at any height in the corner, we added a wedge shape built with polycarbonate and a steel 1x4. This lets the arm slide in between the rings and either lift one up with the polycarbonate part or pull the bottom one out with standoffs mounted below.



(Testing with basic prototype)



(Final design)

09/01/24 Design/Testing: Macro Design

Designed by: Alex

Witnessed by: Carl

Witnessed on: 09/03/24

Goal: Define, Design, and Test macros for robot R.1.2.6

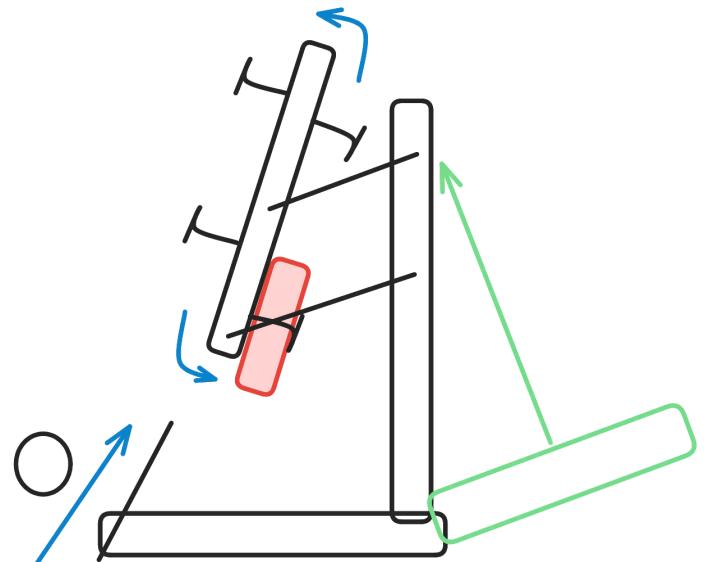
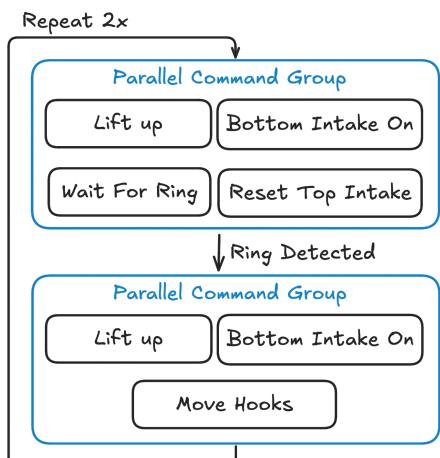
Macros are small pre-programmed tasks that the driver can dispatch, usually at the press of a button. These are very useful on this robot specifically because of the many lift positions and precise location of the top intake. To make the most useful macros we have to establish the specific needs of the tele-op program:

- Intake onto the back of the intake for the wall stakes
- Intake onto the front of the intake for alliance stakes and mobile goals
- Score on wall stakes
- Score on mobile goals
- Holding position (default)

For each of these tasks we will need to create a macro. Ideally each one would only require one button, to reduce the mental stress on the driver, however with some macros needing multiple smaller independent actions we might need separate buttons for. For each of the macros we will go through and analyze and design a macro for each state.

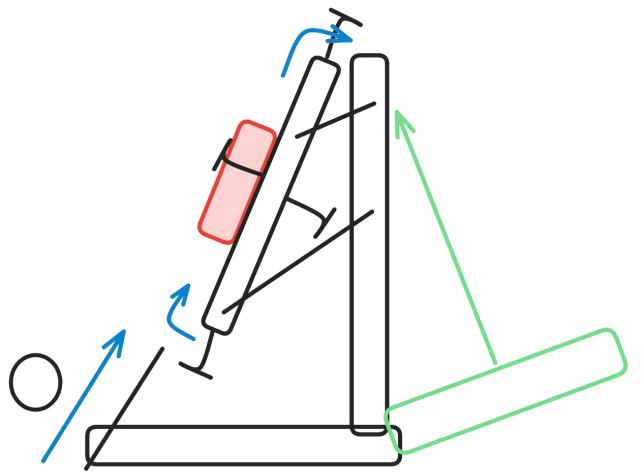
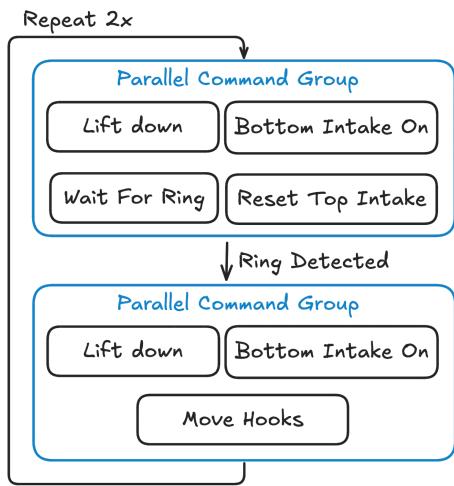
Load Back Intake

For this state we want to intake rings into the robot, detect them, and load them onto the back of the top stage. We will use the distance sensor to detect when rings are in the bottom stake of the intake. For this macro, we lift the arm, wait for the rings, and then move the hooks one by one.



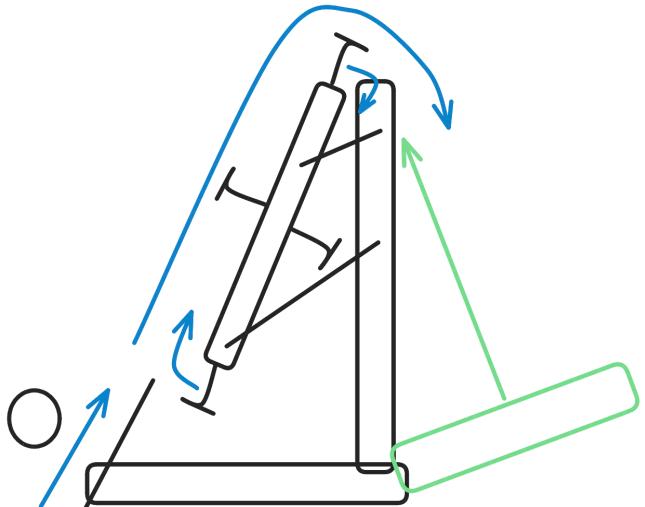
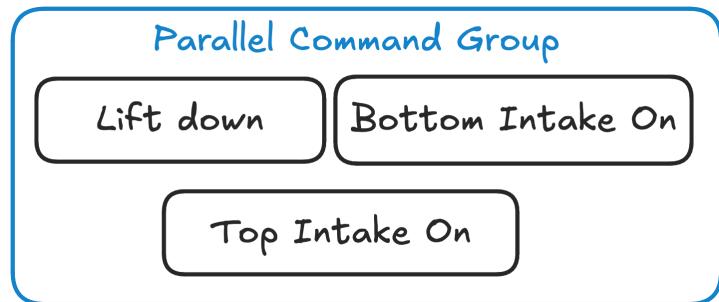
Load Front Intake

In this state we keep the lift all the way down, and then index the rings in the same way we do for the back intake to wall stakes, but for the other side of the intake. This is useful to load up the intake to put them on another goal, or to hoard rings.



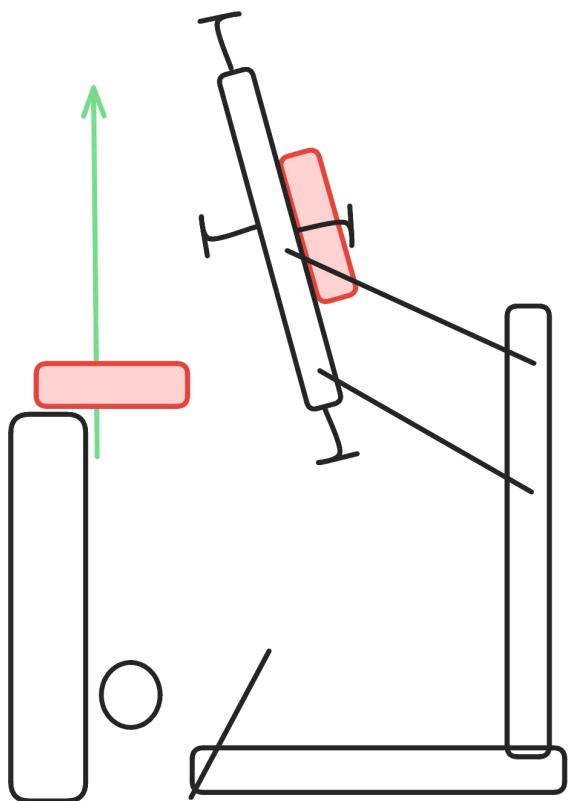
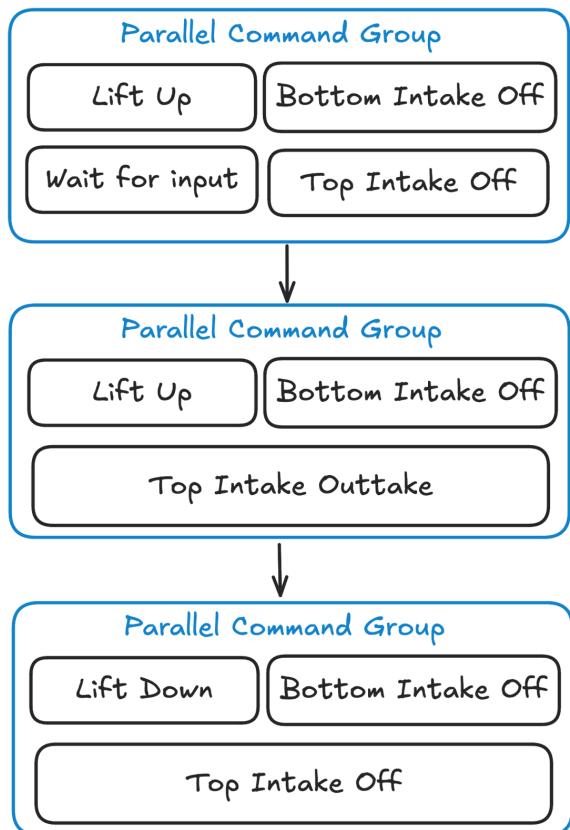
Load Goal/Alliance Stakes

We use this macro to score on the alliance stakes and mobile goals by having the lift down and both intakes spinning at full speed.



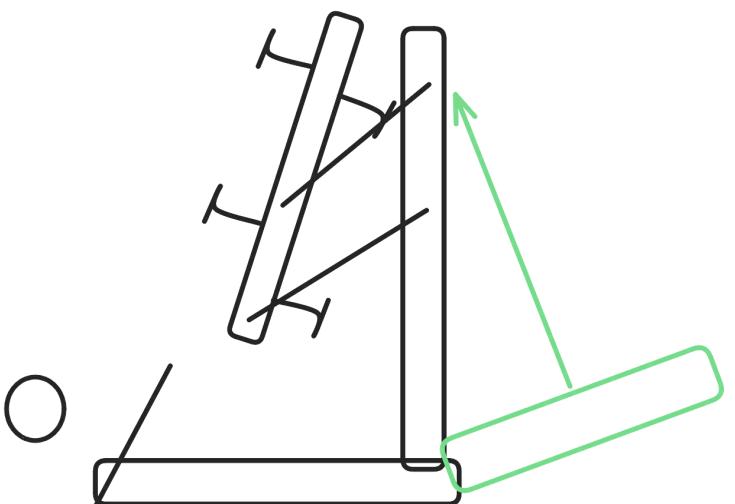
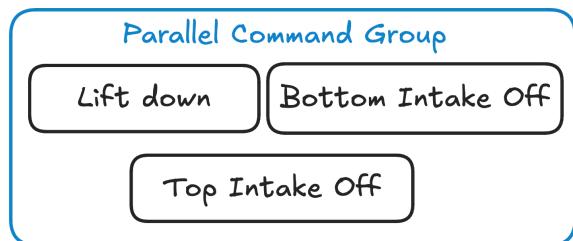
Score on Neutral Stakes

For this state we want the lift to go up, wait for a user input, then throw the rings onto the goal. We then want the lift to go down while outtaking to avoid getting jammed on the rings already scored on the post.



Holding Position

This will be the default position for the robot. Both intakes are off and the lift is all the way down.



Testing

We tested all of them rigorously, for each of the ones that didn't work as intended we created notes below with what we changed.

- Holding position
 - We realized that we had to reset the hook position to make sure that it fits under the hanging structure nicely.
- Scoring on neutral stakes
 - We realized that this shouldn't always move the intake backwards as it's going down and now we only outtake if we use the score button on the stake.
- For loading the front and back
 - The intake needs to wait slightly after moving the top intake so that it doesn't detect the rings that it just intaked. This makes sure that the intake doesn't double intake the same ring and cause indexing issues.

09/04/24

Background Research: Motion Control

Designed by: Alex

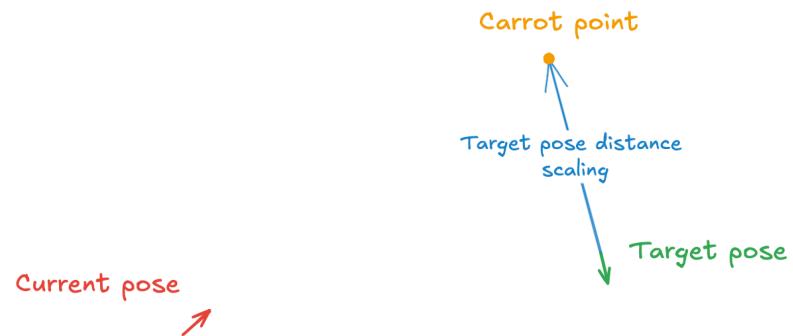
Witnessed by: Carl

Witnessed on: 09/04/24

Goal: Research different algorithms to move the robot drivetrain during Programming controlled periods

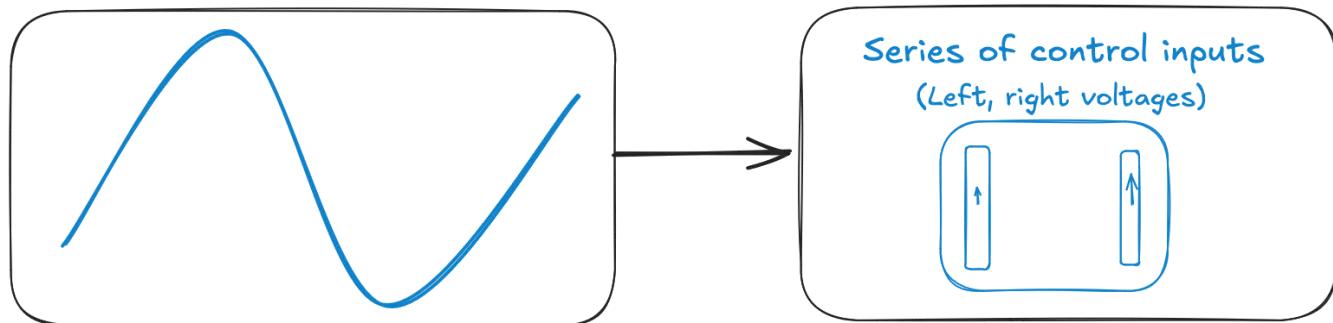
Boomerang

Boomerang is a newer motion control algorithm that many teams had success in Over Under. This algorithm is specifically designed and exclusively used in competitive robotics to the best we can tell. This algorithm works by having a “carrot” point that the robot always tries to drive to. This point is scaled out from the target point in the direction of the desired angle of the target pose. The scale of how far the “carrot” point is from the target point is scaled based off of the distance from the target point. This algorithm is very good at getting to the target pose on drivetrains where the omni wheels drift a lot, and especially preformant on high-speed movements. However this algorithm is not time-determinant, meaning the time the same path takes to run can change from run to run, making timing things much more difficult. Additionally, chaining motions together requires much more tuning than other options can provide.



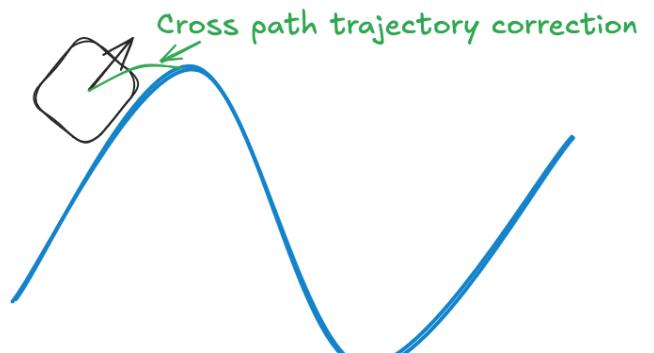
2D Motion Profiling

Pure 2D motion profiling predicts what the control inputs on the actuators on the robot would have to move to complete a desired motion. This allows for very complex motions that complete very quickly, however this method is very prone to propagating errors because it is unable to correct for errors that happen because it doesn't have any feedback provided to the algorithm and only uses pre-generated pathing. With additions, it is able to correct for some errors.



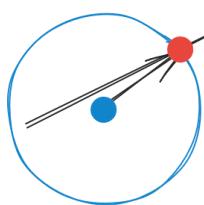
RAMSETE

RAMSETE is an algorithm that can take in motion-profiled paths and reduce the cross path error and angular error on paths. This algorithm is very well suited for motion profiled paths and it retains the high speed and time-deterministic properties of motion profiling, while being able to correct for errors in the movement of the robot. This algorithm is very complex because of both requiring motion profiling and another motion control algorithm, but it can make up for that with its plentiful speed and cross-track correction.

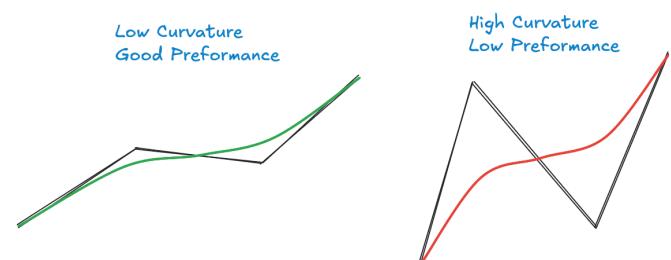


Pure Pursuit

Pure pursuit is a purely path following algorithm that was created to follow paths in cars and other steered mechanisms. It can be transferred to skid-steer robots like the ones we use and follow paths very consistently. This method works by

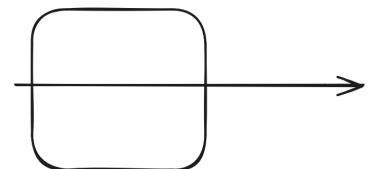


having a predefined path, and then drawing a circle around the robot. The robot then tries to drive to the farthest intersection of the line and the circle. This allows the robot to drive on a smooth path close to the line, and have fairly good accuracy. However, this method is not good at higher-curvature, faster movements, this method has been found to produce very undesirable results with poor path following.



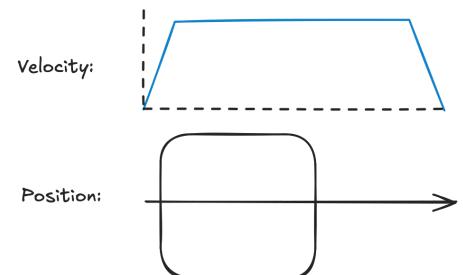
PID Straight Movements

PID straight movements use PID closed loop control to move the drivetrain to the end of the path. We researched PID loops in [feedback control](#) (Pg. 61-64). This solution generally uses 2 PIDs, one for the signed distance of the drive and the other is the angle PID to keep it driving straight. This solution doesn't limit the acceleration so it rocks the robot a lot, and is also limited to straight lines making it less capable than the other options.



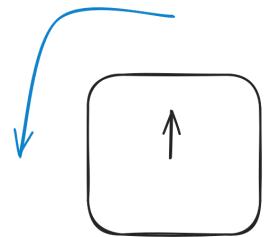
1D Motion Profiling

1D motion profiling moves the robot in a straight line through predicting the necessary velocities that the path needs to be at each point. This can also be used with PIDs for feedback which can make this solution more accurate. This solution limits the acceleration, reducing the problems listed in PID straight movements. However, it is still limited by the straight lines like PID.



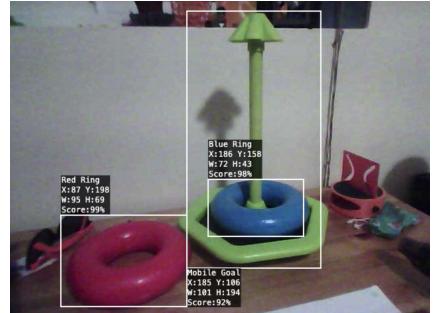
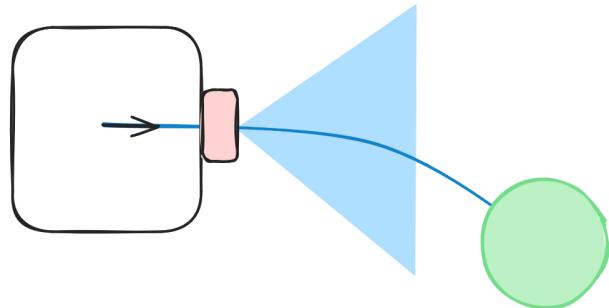
PID Rotation Control

PID rotation control uses a PID to turn the robot to a certain angle. This is very limited in scope, but is very useful to combine with other movements to make a more complicated series of movements more easily possible by moving and then turning to positions.



Vision Controlled Guidance

Vision controlled guidance uses a combination of some feedback, such as a PID loop with input from a Camera. We think it would be useful this year to use the vision sensor to detect the goals and move relative to them to ensure that it always grabs the goal as accurately as possible to increase our consistency.



Conclusion

For motion control we do not have to only select one option, we can create multiple implementations and use them each where they are most useful. This year we want to start by implementing 2d motion profiling and RAMSETE combined to do complex paths with correction, vision controlled guidance for the goals, 1d motion profiling to allow us to do straight movements, and PID rotation control to allow us to turn in place easily. As we implement these solutions we will create entries that go into more depth

09/05/24 Build: Intake Upgrades (R.1.2.7)

Designed by: Carl

Witnessed by: Matt

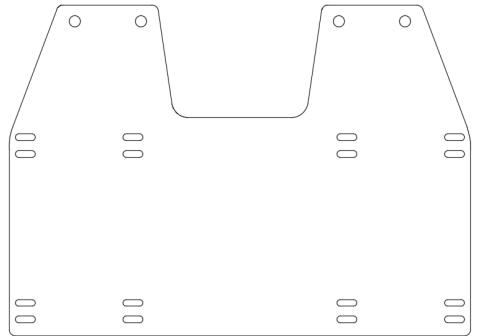
Witnessed on: 09/05/24

Goal: Optimize the bottom stage of the intake for collection while stopped or moving slowly.

Our current intake, although being a huge improvement from the original, still struggles to intake rings when the robot is not moving forward. After examining other teams intakes, we noticed a few things that could be making our intake dysfunctional:

1. Too steep of an angle for the polycarbonate at the back of the intake
2. Polycarbonate ramp: much more friction with rings when compared to Delrin/Acetal
3. Too low of a hardstop for our arm; this creates unnecessary pressure, pushing the rings into the ground

We will attempt to implement the last two elements into our design as they are the easiest to change and would not affect the transition of rings to the top stage. We also took this opportunity to make minor adjustments to the polycarbonate part: Zip tie holes moved to the outside, holes to attach the strips for the upper stage, and removal of some unnecessary material.



Modification Images:



Changing the ramp material and hardstop position improved the intaking performance, but did not fully solve the issue. By moving the HS shaft holding the bottom of the intake plastic forward one hole, we were able to decrease the ramp angle, which ultimately allowed us to intake rings even with no forward movement. Additionally, we slightly altered the spacing on the intake roller to eliminate dead spots.

09/05/24 Testing: Scoring Consistency (R.1.2.7)

Designed by: Carl	Witnessed by: Matt	Witnessed on: 09/05/24
-------------------	--------------------	------------------------

Goal: Conduct testing of scoring on different stakes and determine whether top stage intake modifications are needed.

Testing Methods:

Wall Stake: Fill up a single wall stake with 6 rings collected from around the field. Repeat 5 times.

Mobile Stake: Fill up a mobile stake with 6 rings collected while driving the robot. Repeat 5 times.

Alliance Stake: Fill up an alliance stake with 2 rings collected from around the field. Repeat 5 times.

Scoring Consistency						
Scoring Object	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Consistency
Wall Stake	6/6	6/6	6/6	6/6	5/6	97%
Mobile Stake	5/6	6/6	5/6	5/6	6/6	90%
Alliance Stake	2/2	2/2	2/2	2/2	2/2	100%

Failure Notes:

Wall Stake: The only failure in testing happened in the last trial with the last ring. In 3 of the other trials, the top ring was not fully on the stake, yet would still be “scored” under <SC3>.



Mobile Stake: All failures happened at different points when filling the goal, and were normally linked to harsh robot movements and turns.

Alliance Stake: Very consistent, no errors or additional notes.

Summary:

Wall stakes and alliance stakes were very consistent, however, we still need to further examine exactly why mobile stakes fail and how these failures can be prevented.

09/06/24 Testing: Localization

Designed by: Alex

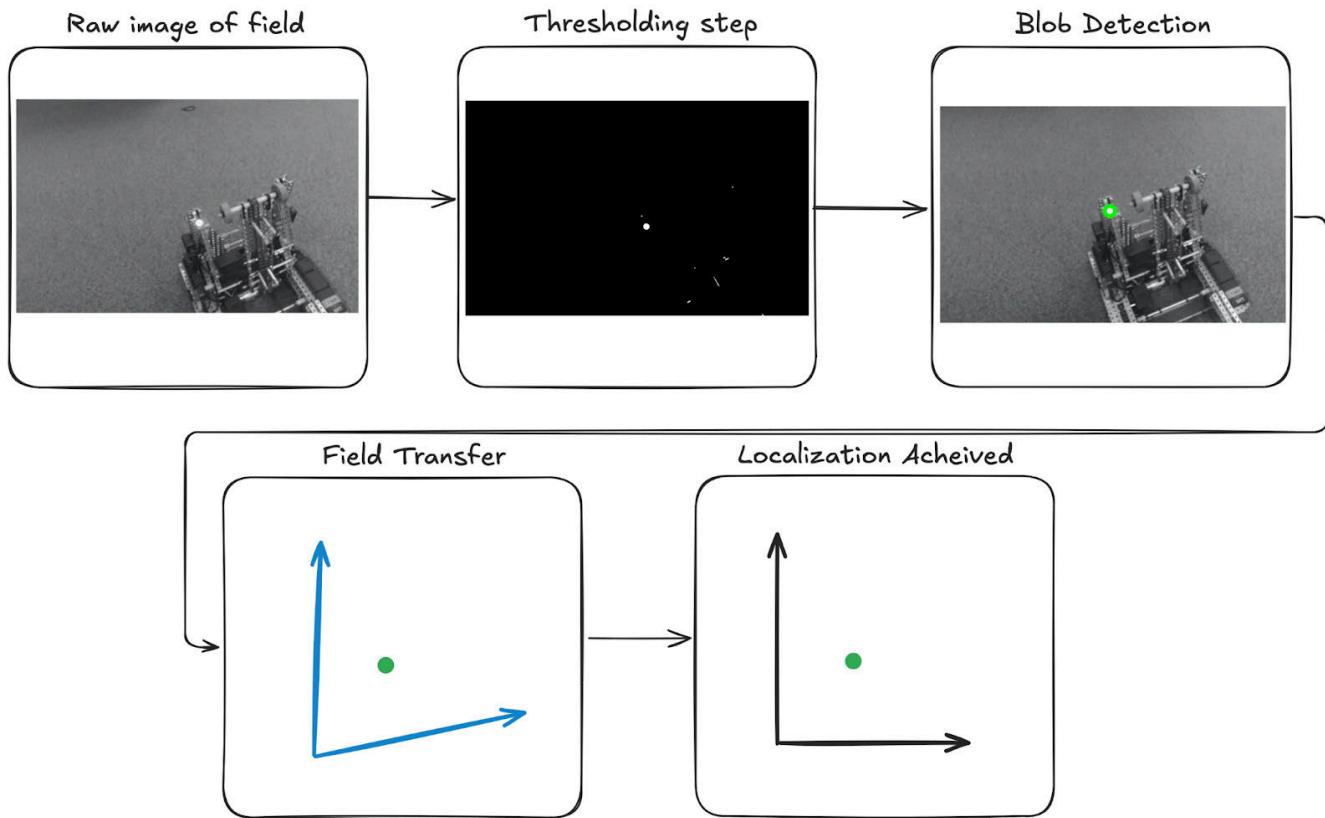
Witnessed by: Carl

Witnessed on: 09/10/24

Goal: Verify our implementation of the particle filter.

Ground Truth

To effectively analyze the particle filter we need to have a “ground truth” that can tell us where the robot is for comparison. For our ground truth localization it needs to be similar accuracy to the particle filter, but be able to get a reliable location at all times. To get a source of global localization we decided to use a downward facing camera above the field with an LED on the robot that the camera can detect and localize in the image. We can then send this measured position to the path planner application to graph where the robot’s predicted location is compared to the ground truth. The following is a flowchart of how the image pipeline runs to measure the location of the robot on the field.



Timeline Issue

We started the implementation of this ground truth localization system, however, on September 3, we decided that we needed to move onto other localization tasks to ensure that we met code testing timeline goals. We plan to finish the ground truth localization later and test our particle filter against that to ensure better tracking accuracy.

Without the ground truth localization it will be harder to test empirically, however we can still put it onto the field and use the telemetry radio we made 2 years ago to wirelessly transmit predicted locations. We can measure this compared to where the lines in the tiles are on the field using our eyes for a good approximation to compare the localization against.

Testing/Results:

When testing on the robot we had a couple issues that needed to be resolved before we were able to have a fully working localization. First, the auto type system in "line.h" caused it to assume the type was a CwizeBinaryResult, however, when we explicitly set the type as an Eigen::Vector2d, it fixed the multiplication result. This was also due to an issue in the distance sensor. We fixed both of these and the offsets were correct.

We tested the calculated distance and line sensor values and then used the predicted values versus the actual values. The predicted values were within 2% of the actual values. This was a very good result and proves that our prediction methods are very accurate.

Another issue we had was the normal_pdf implementation having a standard deviation that was too narrow, causing the particles to be weighted too low. To fix this, we decided that we should have a wider standard deviation to allow certain less probable particles to survive for the time being.

We started with 100 particles in our filter, however after analysis we decided that it would be better if we could have more particles. With 100 particles there is a particle every 196 in², and if they were all spread out evenly throughout the field, one particle for every 14inx14in square. We think increasing the amount of particles to 500 (39.2 in²) or 1000 (19.6 in²) would increase the performance of the particle significantly. This could use up to 8kB of memory, which the V5 brain has 128MB of, so we think that this allocation of memory is appropriate. However, we will have to analyze if the V5 Brain's Cortex-A9 processor has enough processing power(1.3GHz) to calculate the p-values for all of the particles.

Optimization

With our naive implementation of all of the algorithms, the full kalman filter with 1000 particles takes 50ms to run, giving us a maximum update frequency of 20Hz, which is well below our goals. To fix this we will have to optimize the filter to make it run with less computational power.

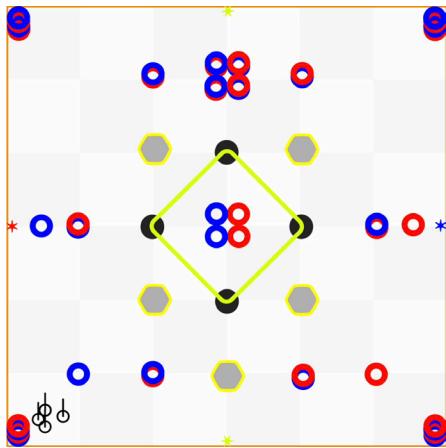
In optimizing the particle filter we had many setbacks, especially as we were trying to predict so many particles at once; keeping the complexity of the algorithm to a minimum was pertinent to remaining within our time

constraints. By adding profiling steps we were able to narrow down the noise generation of the particles as one of the most expensive steps. To fix this, we changed from the default random engine, to the "subtract_with_carry_engine" to reduce computational complexity. This nearly cut the computational complexity in half. We also considered changing from doubles(64-bit precision, floating point numbers) to floats(32-bit precision) to reduce the number of random bits the computer would need to generate. This reduces the accuracy of our algorithms by a very small amount (7 decimal places), so this change would be negligible and therefore it would be acceptable.

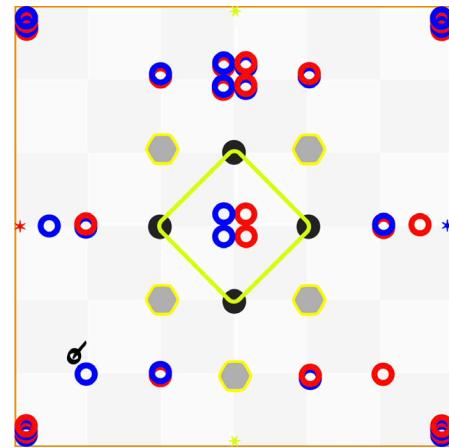
Another thing we had to change was the sampling. In our resampling based on the weight, we calculated the particle based on the weights an index too high. We reduced the index by one and the higher weight particles were weighed properly then. We did not have to optimize this step because even for 1000 particles it only took about 60 microseconds to complete, which is much lower than the computation time for the other steps.

Retesting

When we retested on the field we found that the algorithm was easily able to localize the robot given a uniform, random starting belief with just time of flight and line sensors. We were able to test this using our field visualizer, with 10 randomly selected particles from the 500-particle belief. This allowed us to see the entire probability cloud over time without having to send and graph all 500 particles every frame.



Initial Particle Distribution (Seeding initialization)



Particle distribution converges

Listed in Time/particle	
Particle Time	
Change	Negative
None	60 ns
Float	40 ns
Random Engine	20 ns
Line Optimization	6 ns
Reducing Particles	6 ns

09/06/24 Identify: Sep. 3 Game Manual Update Analysis

Designed by: Carl

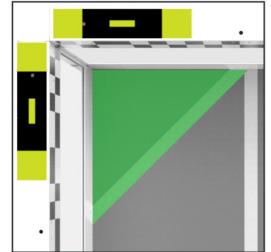
Witnessed by: Alex

Witnessed on: 09/07/24

Goal: Analyze the major impacts of this update on strategy and design.

Field tape layout changed so that Corners are now triangular

Impact: Mobile stakes may not be able to be placed in the corners consistently if there are rings present. This will affect matchplay and potentially skills pathing/movements.



Changed the starting Ring configuration; the Ring stacks that begin in the right Corner along each Alliance Station wall have been flipped

Impact: In autonomous, we only need to remove the first and third rings, no matter which corner we start in.

Added Significant Q&A boxes and bullets throughout the manual

Impact: We should look at the Q&As throughout the game manual and take note of impactful responses.

Add Violation notes and clarifications to <S1>, <S2>, <G1>, <G2>, <G4>, <R28>, and <T1> to clarify intent

Impact: We should all re-read the game manual to ensure we are up to date on these updates.

Updated <SC8> to include the Autonomous Win Point tasks for World Championship qualifying Events:

1. At least four (4) Scored Rings of the Alliance's color.
2. A minimum of three (3) Stakes on the Alliance's side of the Autonomous Line with at least one (1) Ring of the Alliance's color Scored.
3. At least one Ring of the Alliance's color Scored on the Alliance's Wall Stake.
4. Neither Robot contacting / breaking the plane of the Starting Line.
5. At least one (1) Robot contacting the Ladder.

Impact: At signature events and our state championship, we should optimize our autonomous routines for the new tasks. We should also coordinate with alliance partners well before the match to ensure compatibility.

Rewrote <SG4> to only apply to opponent's Scoring Objects

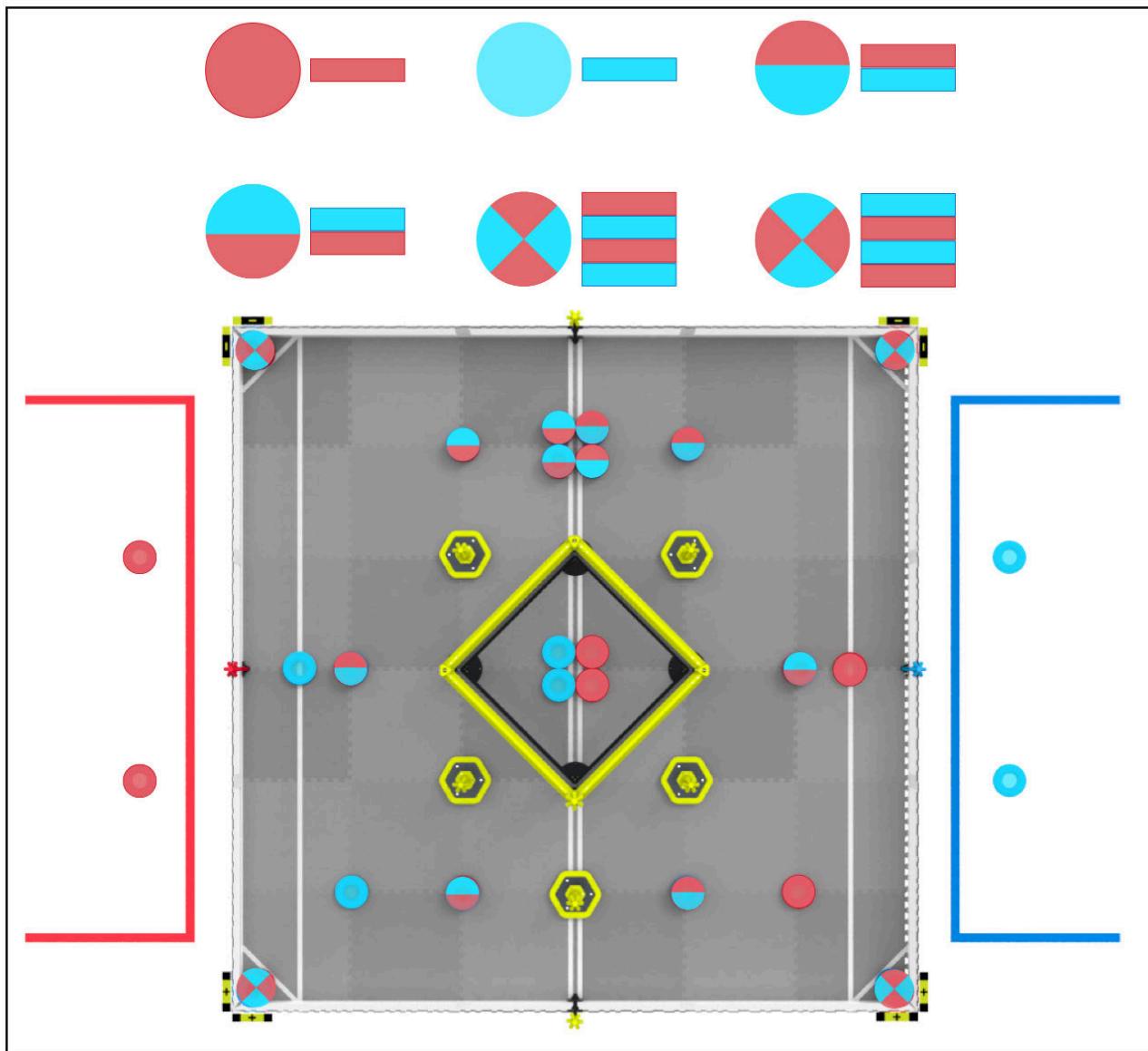
Impact: Scoring on wall stakes no longer carries risk of disqualification, however, descoring them is still risky.

Updated <SG11> to change the Positive Corner protection period to fifteen (15) seconds

Impact: With the increased endgame, hanging will become more viable and negative corners might be used more frequently.

Note: This is not a complete list of updates, only ones that we believe significantly impact design or strategy.

Updated Field Image:



09/08/24 Design: Autonomous Path Planning

Designed by: Alex

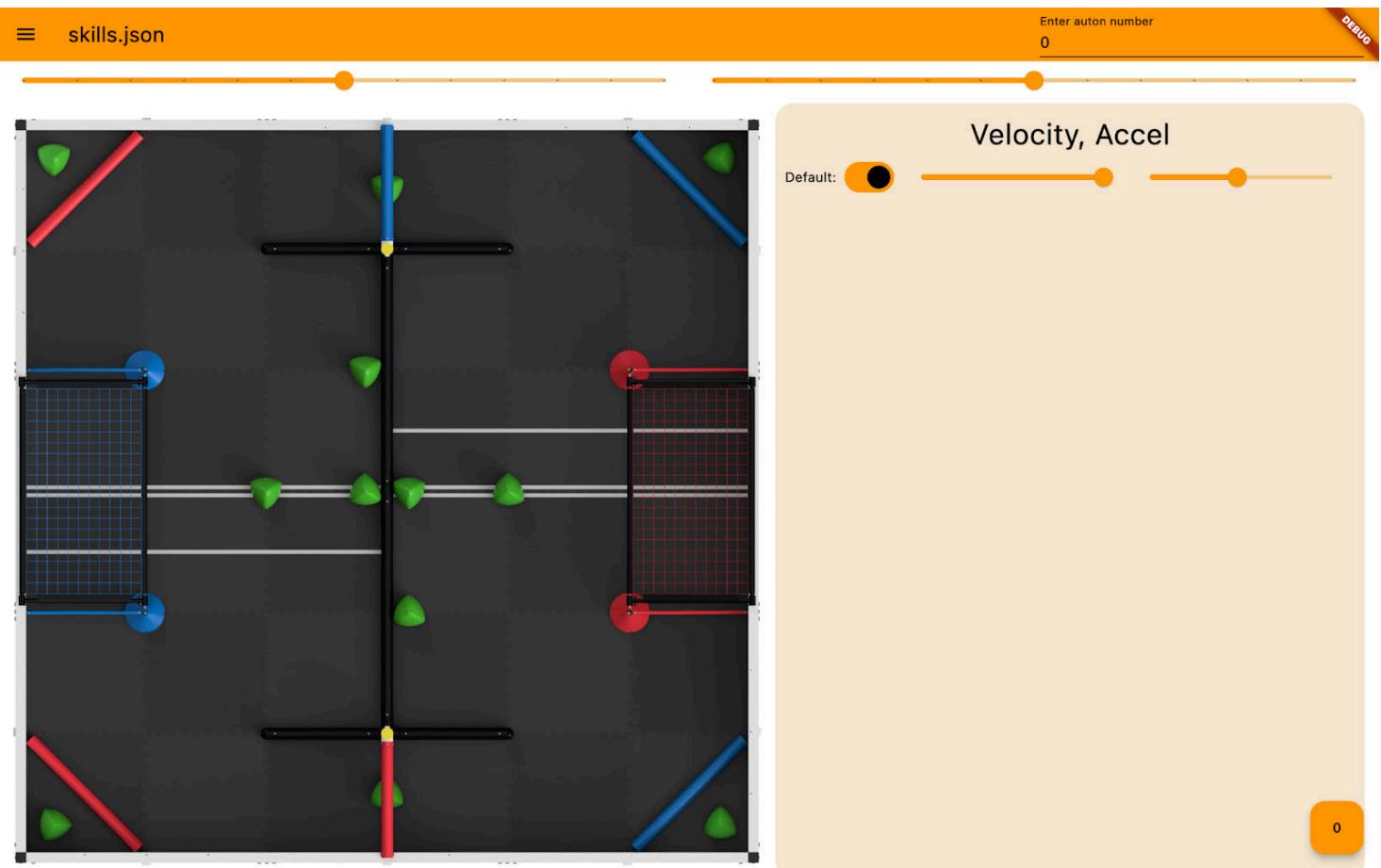
Witnessed by: Carl

Witnessed on: 09/09/24

Goal: Create or modify an app to create paths for our autonomous.

Path Planner Selection

To effectively create paths for autonomous and skills, we need some way to visualize the paths to ensure that the robot is going to the right spot. We think that it would be best to do this in some sort of app that could help us visualize where the robot is going during the path. We think the best solution would be to adapt the [open-source path planner](#) we made last year in flutter. This is the optimal decision for us because we already have plenty of experience in this application because we made it ourselves, and we know how to modify it to suit our needs. This is what the path planner looked like last year:



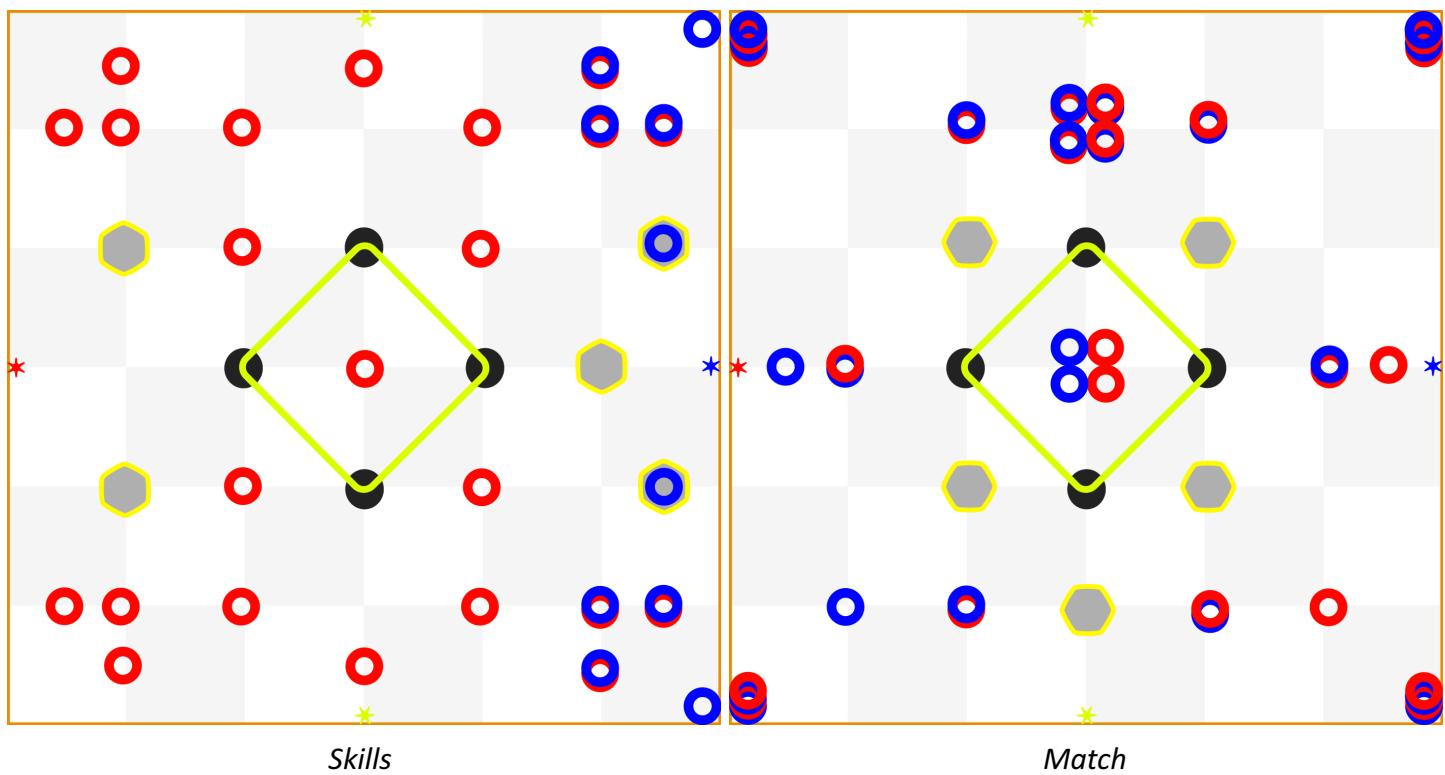
Path Planner Modification

This year we need to modify the path planner to better suit our needs. This will require completing the following tasks to adapt it for this year:

- Update field image
- Enforce smooth splines
- Simulate movements and Predict time
- Robust windows support
- Adjust coordinates

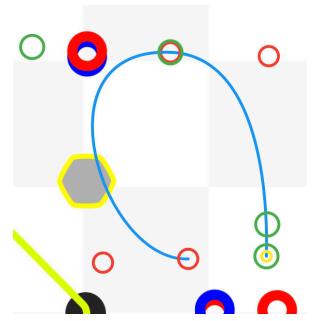
Update Field Image

First, we have to update the field image to work for this year. To do this we have to create a cleaner field image so that we can easily print it in the notebook and see where the ring stacks are. We created a field image in inkscape for our path planner application, we ended up with these images for both skills and matches:



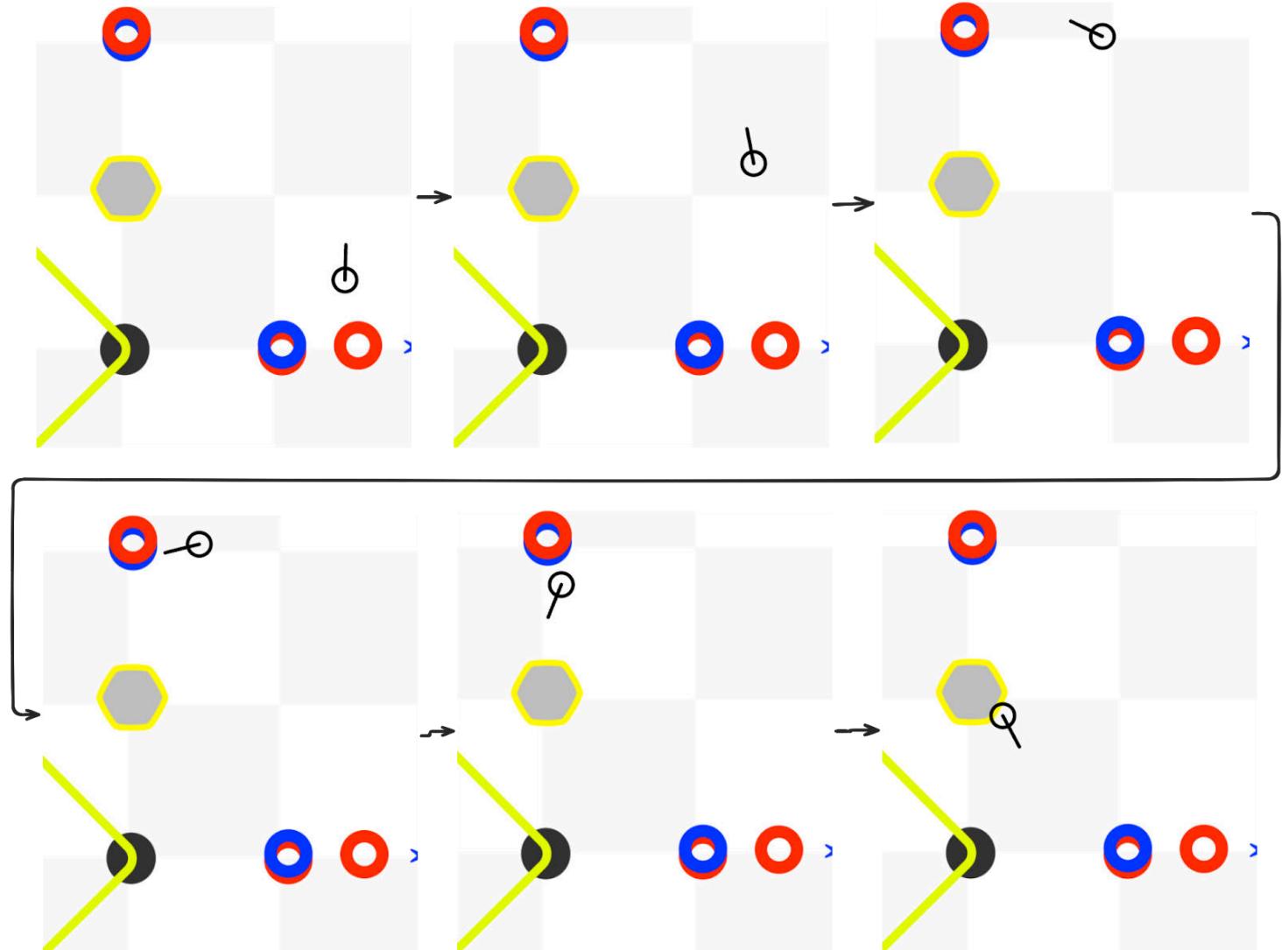
Enforce Smooth Splines

The current path planner is designed to work with last year's game and our old motion profiling algorithm. We are still planning to use splines, so most of the path planning aspect can remain the same, however this year we want the ability to ensure the smoothness of the splines. We will do this in the path planner by making sure that the spine ends are always pointed at the same angle in between splines. Additionally we can make the end points always move to the same spot. Now when we create or adjust splines they are always enforced to be smooth:



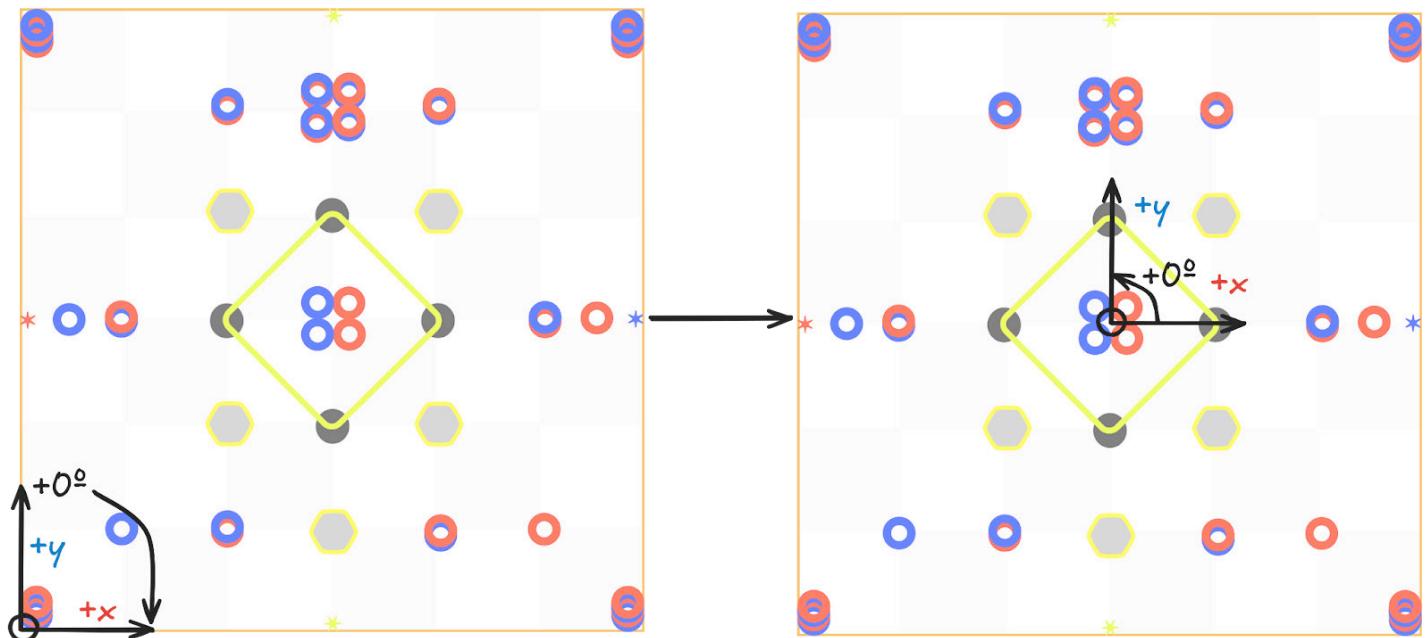
Simulate Movements

Another thing we want this year is a way to simulate how long the movements would take. This will help us create better time-optimized autonomous trajectories. To do this we will run the rust motion profiling library that we made live to simulate the time motions take. We then use a flutter rust bridge to call the rust motion profiling functions from the Dart flutter interface. In the end this makes the motion profiling animate paths like this:



Adjust Coordinate Systems

Another thing we want to change this year is to move to another coordinate system. Last year we used a coordinate system with the origin in the nearest left corner to the driver station with $+y$ being forward and $+x$ being right. We also had angles that pointed in the $+y$ direction and positive clockwise movements, because we thought that would be intuitive. However this year to make kinematics easier we want to use trigonometric angles and coordinate systems, which is $+x$ being 0° , $+y$ being 90° counterclockwise, and $+0^\circ$ being counterclockwise. This makes kinematics easier because coordinate rotations become much easier because trigonometric functions become much simpler to think about in this coordinate system. Additionally, the [field mirroring issues](#) (Pg. 40) we identified become easier to solve with this coordinate system because we can mirror paths simply by inverting the angles and y values.

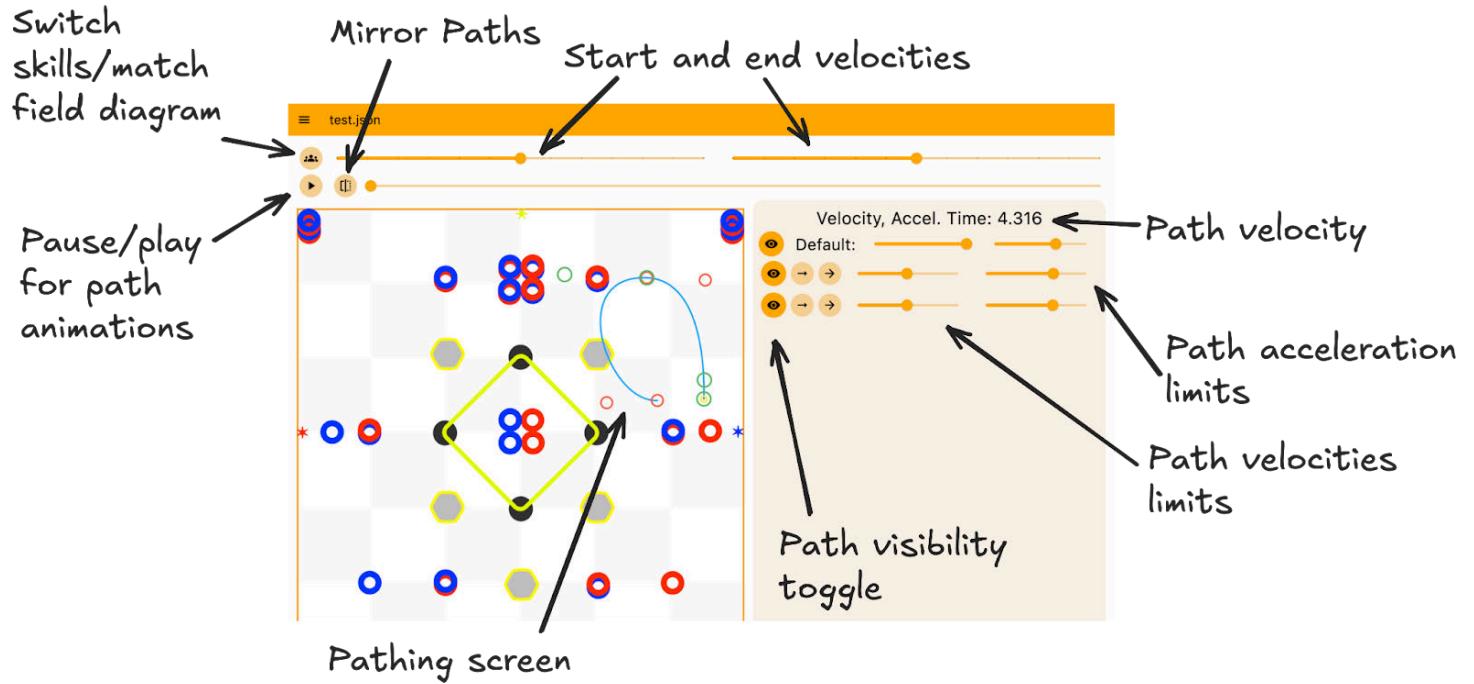


Robust Windows Support

Last year we made this path planner only work on MacOS because that's primarily what we used. However, this year we want to add windows support so that more of our teammates can also use the path planner and help out with making autonomous paths. The main thing that needs to change is the keybinds because they use the command key which only works on MacOS, but if we add keybinds that allow for the use of the control key on Windows we can support multiple operations systems at once.

Conclusion

In the end we added several features to our path planner, and it is going to be much easier to use for this year. This is a diagram of all the functions that our path planner provides.



09/08/24 Design: Skills Pathing

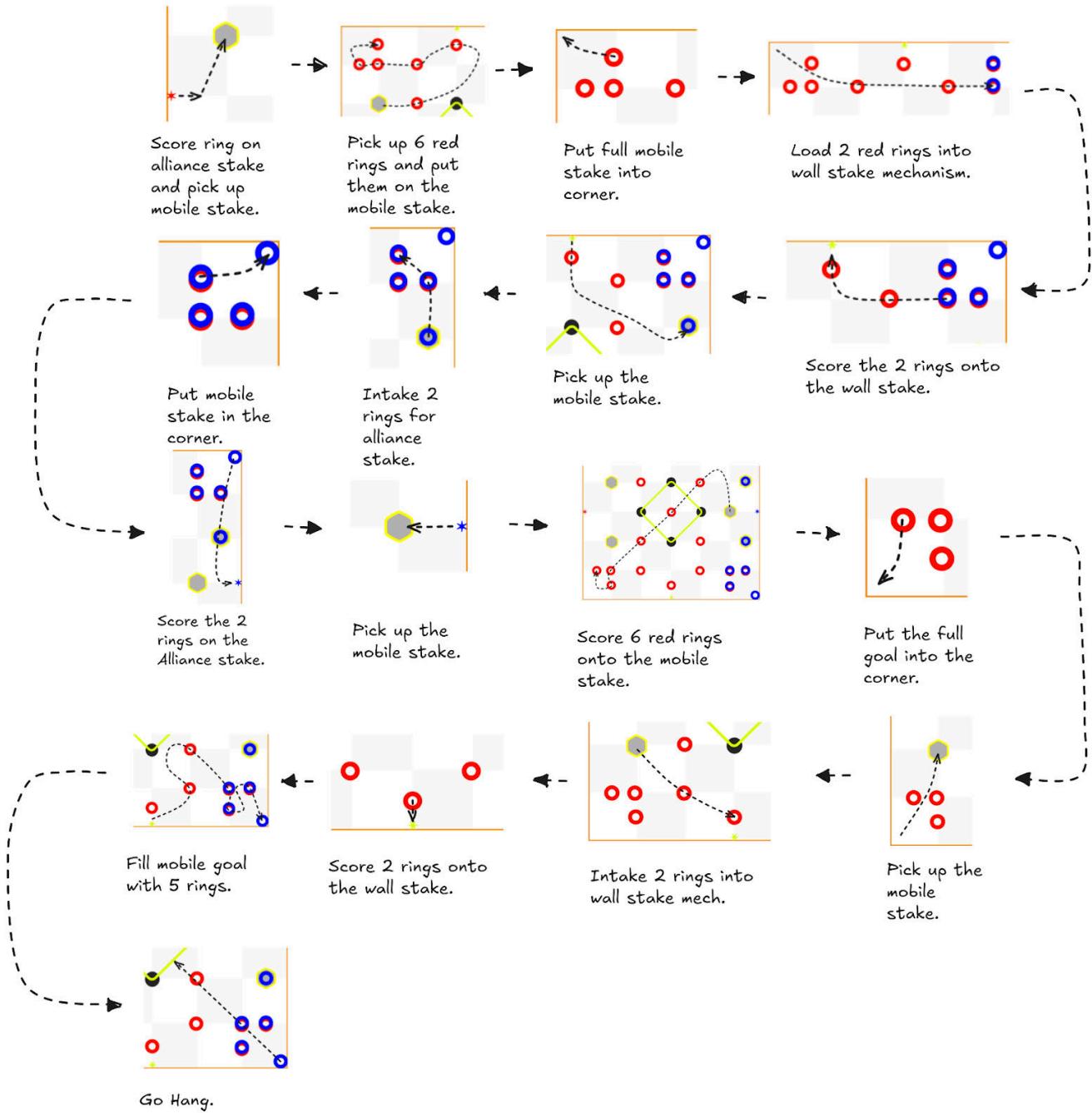
Designed by: Matt

Witnessed by: Alex

Witnessed on: 09/10/24

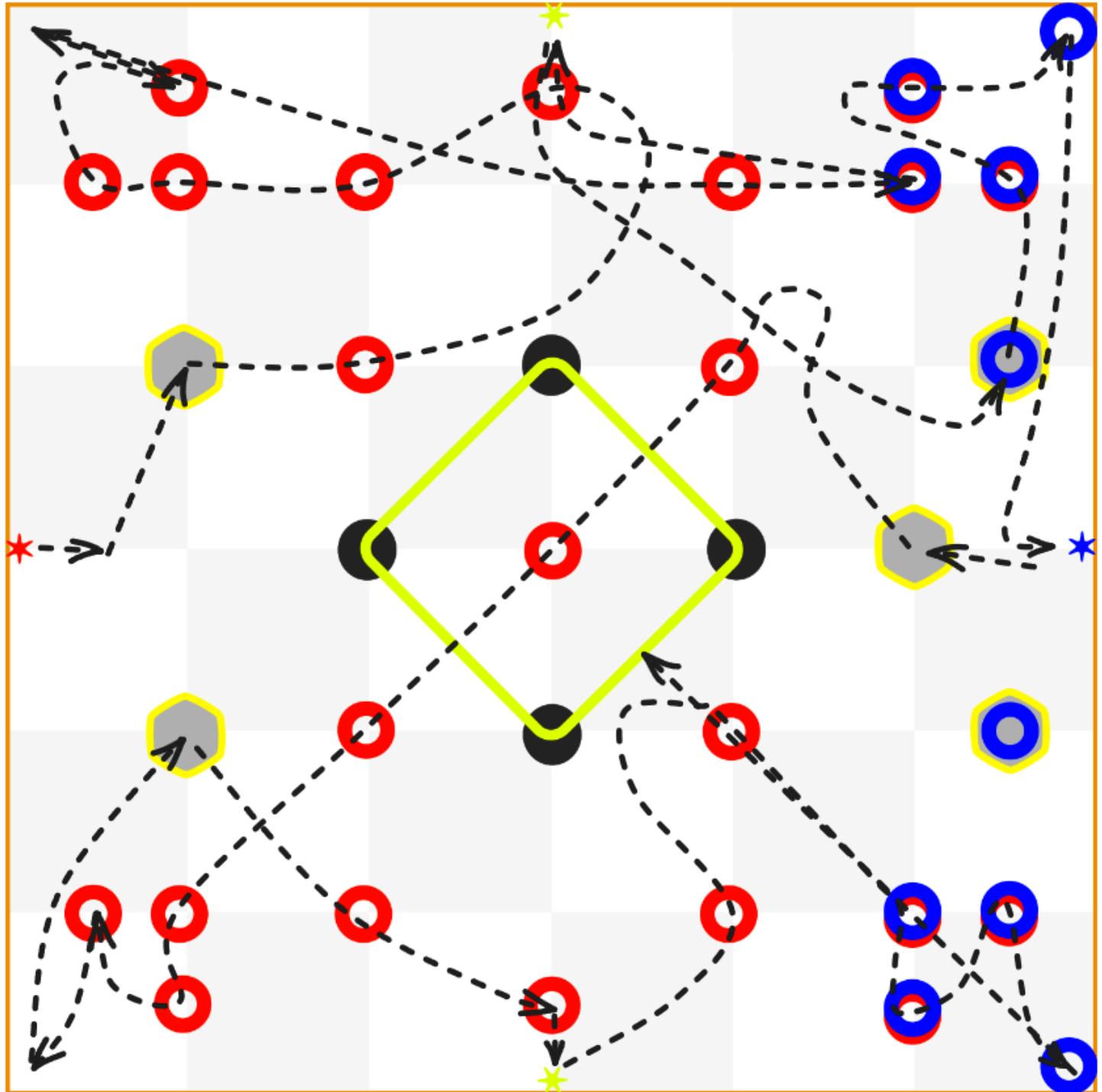
Goal: Determine the most optimal skills path.

To meet our goal of getting in the top 10 of world skills, it's crucial that we utilize the full 60 seconds to maximize our points. To make sure our path fits in time, we need to ensure that each segment of the path is optimized. We have thought through every step making sure it is the fastest path possible.



Full Path:

This is the final version of the path, it is able to get every red ring on the field including the pre-load, and score at least one ring on every stake. The path scores a total of 59 points and is estimated to take 55 seconds. With this path, we try to minimize the time spent driving around reducing unnecessary movement and time.



09/09/24 Design: Vision Guided Motion Control

Designed by: Alex

Witnessed by: Carl

Witnessed on: 09/10/24

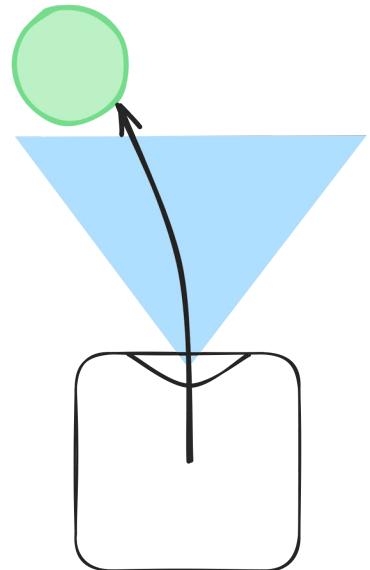
Goal: Design a motion control algorithm to clamp onto goals using vision controlled guidance for higher programming consistency.

In our Background Research on [Motion Control](#) (Pg. 169), we researched various different motion control algorithms that we could use in our Autonomous and Programming skills routines. One of the options we identified was vision controlled guidance. This uses the vision sensor to find goals or other objects to pursue them using vision feedback. On our current robot (R.1.2.7) we have an AI vision sensor mounted on the back of the robot to track goals, but we have to implement an algorithm to track them properly.

Algorithm

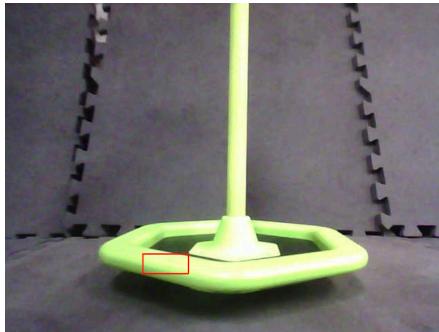
We were thinking that we could use a PID controller with input from the detected camera angle to the goal as the input and a target at the center of the camera.

We will use the color detection in the AI vision sensor to detect the goals as opposed to the AI detection model because of the faster data acquisition speed, allowing us to run a tighter PID loop. We will also have the drivetrain slowly drive backwards at a constant voltage so we can grab the goal. To compartmentalize this we will put it into a Command very similar to the drivetrain rotation algorithm.

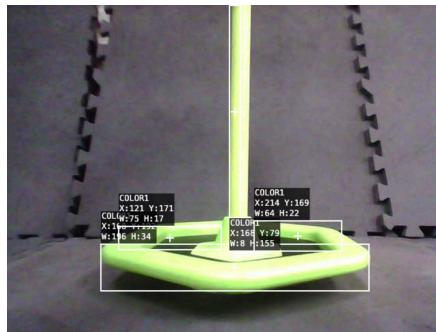


Vision Sensor Tuning

To tune the vision sensor we used the online VEXCODE V5 Block code interface because that is currently the only AI Vision sensor tuning utility at the time of writing this. We selected the yellow-green base of the goal to have the largest colored area to detect the goal with, set the color of the first ID, and then increased the hue range until it was just detecting the base of the goal reliably. We then imported this into the repo to make commands.



Selecting base of the goal

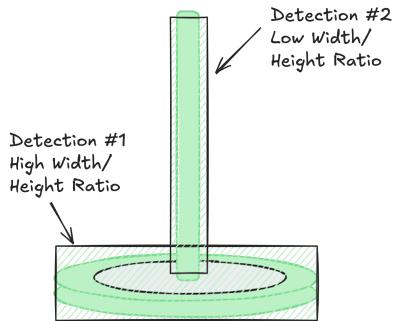


Resulting camera detection



Upon moving closer to the goal

The camera results show that the camera can easily detect the goal, however it can still detect it up close because the colors are still very obvious, so we can't just wait until we have no detection. However, we can use the change in the width/height of the detected objects to see if we are close to the goal. In the right image if it were to detect the pole it would have a very low width to height ratio and we can use this change to determine that we are close enough to the goal to grab onto it.



Command Programming

In the execute function for this state, we update the pid with the goal angle from the vision sensor, and a static voltage to keep the robot moving backwards. We multiplied the static voltage by the cosine of the goal angle to ensure that if the robot is too far away from the target the static voltage won't overpower the turning PID power. We also limit the maximum turning power to ensure that the static voltage is always moving the robot backwards.

```
void execute() override {
    Angle angle = 0.0;

    if (const auto goalAngle = drivetrain->getGoalAngle(); goalAngle.has_value()) {
        angle = goalAngle.value();
    }

    const auto output = std::ranges::clamp(pid.update(-angle.getValue()), -1.0, 1.0);
    drivetrain->setPct(static_voltage * cos(angle) - output, static_voltage *
cos(angle) + output);
}
```

The isFinished function of the Command checks the largest detected object's "aspect ratio" against the 3.0 constant to see if it is detecting something other than the base of the goal. Once this happens it signals that it is done and the next state of the sequence is scheduled.

```
bool isFinished() override {
    if (const auto aspectRatio = drivetrain->getLargestObjectAspectRatio();
    aspectRatio.has_value()) {
        return aspectRatio.value() < 3.0;
    }
    return true;
}
```

Testing

Method:

1. Put goal behind the robot
2. Turn on auto targeting
3. Repeat 5 times

In our testing the targeting code worked very well, however we found that the aspect ratio code to detect if it was close to the goal ended up being inconsistent. However if we were to add a sensor to the back goal clamp to detect when there was a goal ready to grab we could use the pole detection and still be able to move accurately towards the mobile stake. This is why we think it would be best to add a line sensor to the intake to detect when a mobile stake is in the right position to grab. In our preliminary testing the line sensor was able to detect that there was a goal within close range(< 1in) without the need for physical contact or additional moving mechanisms which are more prone to failure. Based on this testing we will reevaluate the goal pursuing code once a line sensor is able to detect when there is a mobile stake ready to grab.

Trial	Properly Targeted?
1	NO
2	YES
3	YES
4	YES
5	NO

09/10/24 Design: 2D Spline Motion Profiling

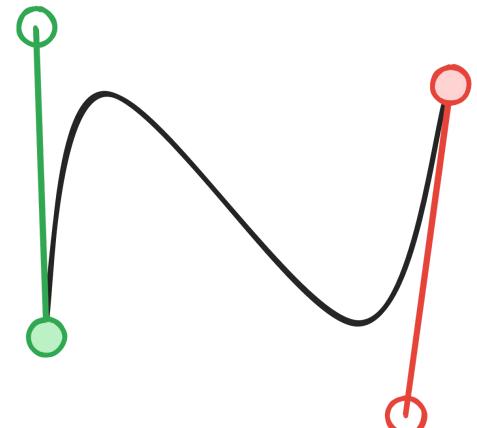
Designed by: Alex

Witnessed by: Carl

Witnessed on: 09/13/24

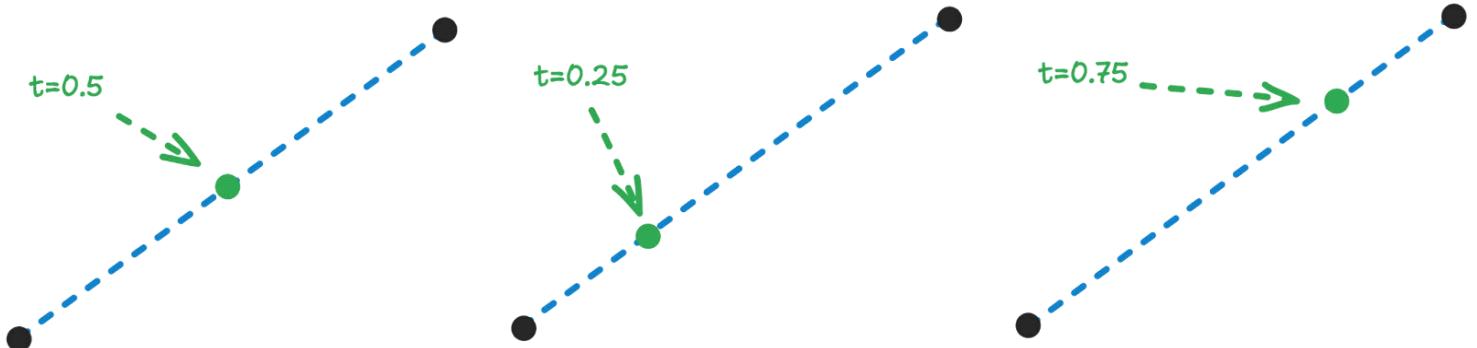
Goal: Create a 2D motion profiling algorithm for Beziers(third-order splines).

In our case, 2D Motion profiling refers to the algorithm that creates a series of linear and angular velocities, and the pose at each time necessary to complete the input movements. We are profiling motions over the splines because they allow for very flexible paths, with beginning and ending orientation specified, and the ability to easily chain complex movements. These splines are constructed with beginning and end points and 2 control arms. In the illustration on the left, the beginning point is illustrated with a filled green circle, the control arm and point indicated with the unfilled circle and line, and the same with the red end points.

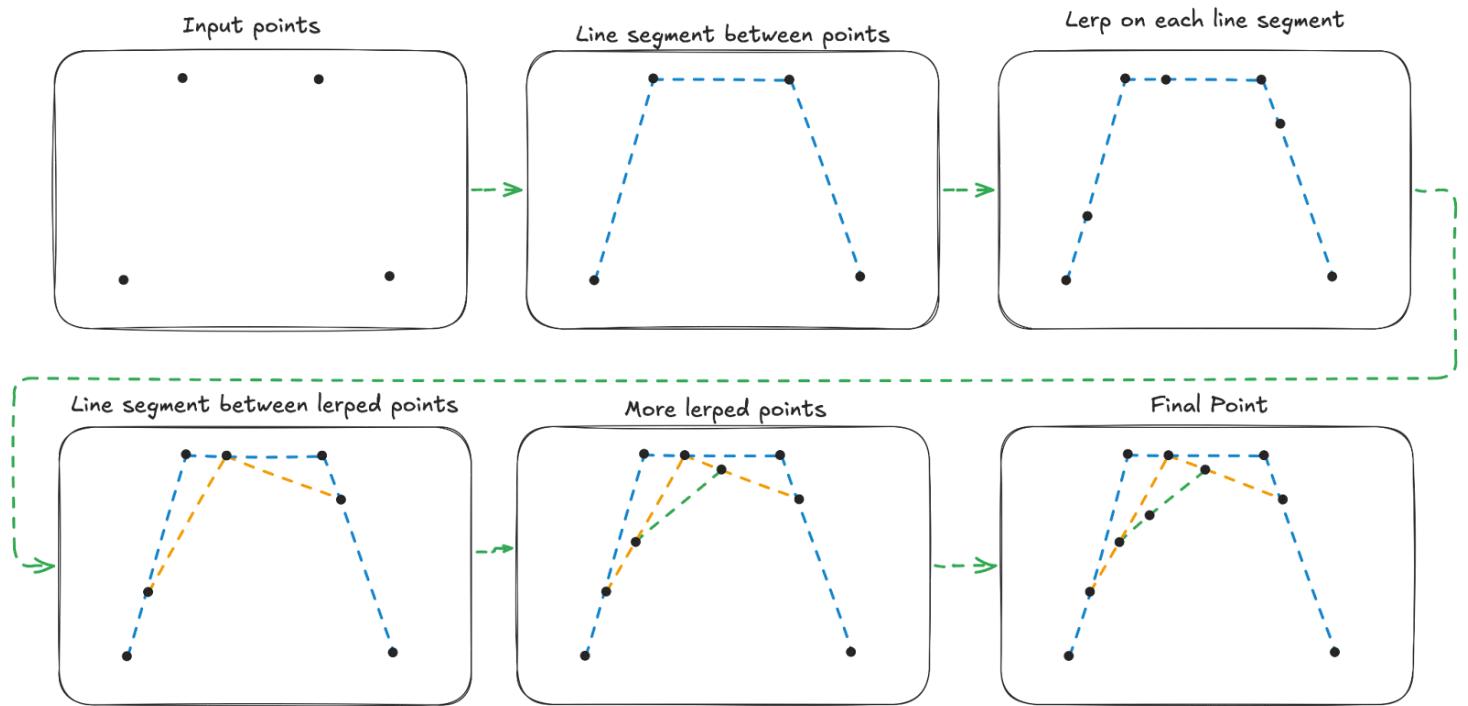


Spline Math

Splines are a variety of mathematical curves that we use to describe the paths that we want the robot to follow. These curves are constructed with input points as shown in the figure to the top right, that illustrates a spline defined by 4 points (3rd order). There are many ways to calculate the points that allow the spline given the inputs, but the following method, while slightly inefficient, is easy to visualize.



This method is based on Linear Interpolation, which is often shortened to “lerps”. This calculation calculates the x and y position of a point on a line between two points, with a t value that specifies where on the line the point should be.



This method calculates splines by recursively applying lerps to determine the point at a given parameter t . For each step, it computes lerp points between control points, then iteratively repeats the process by calculating new lerp points between the newly created points. This continues until a single point remains, which represents the position on the spline at t .

$$p(t) = [t^3 \ t^2 \ t \ 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

$$p'(t) = [t^2 \ t \ 1] \begin{bmatrix} -3 & 9 & -9 & 3 \\ 6 & -12 & 6 & 0 \\ -3 & 3 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

$$p''(t) = [t \ 1] \begin{bmatrix} -6 & 18 & -18 & 6 \\ 6 & -12 & 6 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

We can deconstruct the individual lerps into a parametric equation that we can use to create a Matrix that would allow us to more easily and cheaply do spline calculations (derivation is omitted for brevity). Additionally, we want to get the first and second derivatives to calculate the angle and curvature at each point on the path.

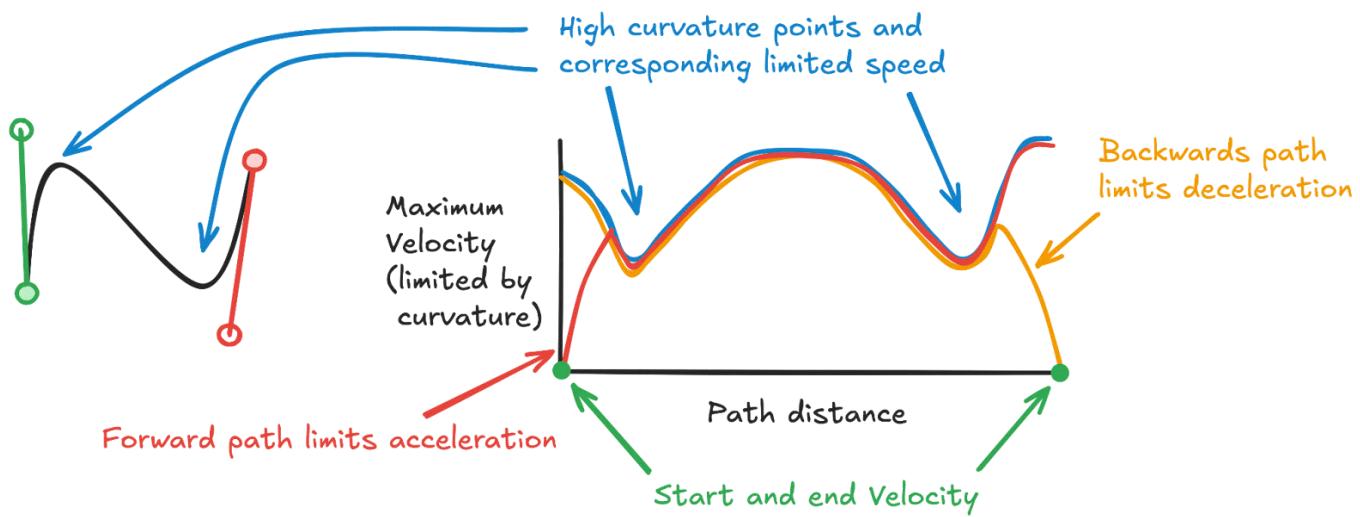
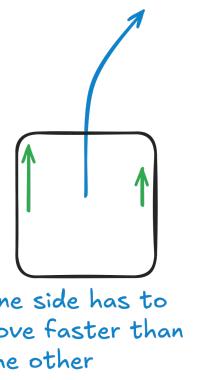
We also need to calculate the total distance of the path which can be done with this formula::

$$d = \int_0^1 \|p'(t)\| dt$$

This formula allows us to calculate the total distance, however we also need to calculate the distance from the start at various points on the spline. We can do this by changing the ending of the integral to various percentages on the spline. As we calculate the distances on the spline, we will put the distance and t value into a linear interpolator so we can estimate t from distance later (This will be important in curvature calculations). Although the integral would in theory give us a perfect estimation of the distance, in practice there is no known closed form solution for 3rd dimensional spline curves, so we have to use numerical integration methods on the brain to solve for distance. We chose to use rectangular approximation to solve for the distance of the spline, as it was the simplest method. Although it is slightly inaccurate, with enough samples (~50-100) the error should be negligible.

Linear Velocity Calculation

First, we have to calculate the linear velocity at each time in the path. However, we also need to simultaneously calculate how long the path will take and account for the robot slowing down to be able to take curves and stay within the kinodynamic constraints of the robot. This is necessary because when turning the robot cannot have the full linear speed of the robot without having some of the wheels move faster than the maximum speed of either side of the drive. This maximum linear speed also changes throughout the path as the curvature of the path changes over time. To account for this and ensure that the robot is always staying within the acceleration constraints, we chose to use a forward/backwards pass approach to limit the acceleration over the total length of the path. This step is extremely important as it ensures the code doesn't violate any kinematic constraints of the robot.

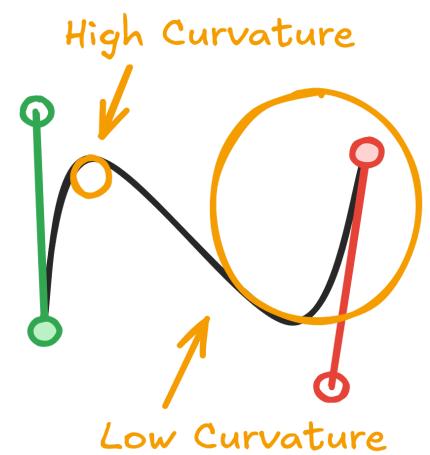


Angular Velocity Calculation

Given the velocity at each point on the path we can multiply the curvature of the spline by the velocity of the robot. Curvature, the inverse of radius, gives us the rate that the spline is changing directions, which can be represented as a circle. We use the following formula to calculate the curvature at the t value(a number from 0-1 giving the progress over the path):

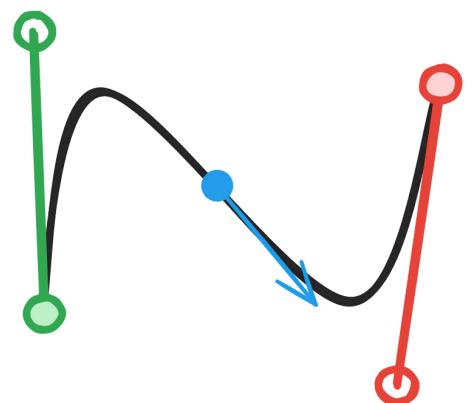
$$C(t) = \frac{p'(t) \times p''(t)}{\|p'(t)\|}$$

We then multiply curvature($C(t)$) by the linear velocity calculated at each point to get the angular velocity at each point. We use this calculation in our path following algorithm to ensure the robot is turning at the correct rate to follow the path.



Pose Calculation

Pose in robotics is defined as a position and orientation. On our 2d field model this means a x, and y value, with a Θ angle value. To calculate the position on the spline is a fairly simple task, using the t value from the distance interpolator from the motion profiling, then plugging that back into the $p(t)$ equation shown earlier. We then calculate the angle of the robot from the normalized $p'(t)$, which gives us a unit vector in the direction we want to travel.



Conclusion

Now that we have calculated the spline motion profile, we can use it in a path following algorithm to more effectively follow complex paths at faster speeds. Our algorithm is composed of spline calculations, a linear velocity profile, and angular velocity, and pose calculations. We use all of these together to calculate how the robot has to move to follow the spline input paths. These spline paths will allow us to intake and score rings faster in our autonomous skills and autonomous routines, which will help us meet our strategic goals.

09/12/24 Design: Ramsete Path Following

Designed by: Alex

Witnessed by: Carl

Witnessed on: 09/13/24

Goal: Implement the RAMSETE path following algorithm to follow the paths generated by 2D motion profiling.

Implementation

Motion profiling allows us to get the necessary predicted control outputs of the drivetrain, however, it doesn't provide a way to correct deviations from the paths. We implemented the Ramsete described implementation in 8.8 - Ramsete unicycle controller in [Controls Engineering in FRC](#). Ramsete follows a motion profiled path, such as the one we implemented in [2D Spline Motion Profiling](#) (Pg. 187-190). First we take in the desired linear velocity (v_d), desired angular velocity (ω_d), and desired pose (x_d) from the motion profile at that time. From our [Particle Filter](#) (Pg. 131-135) we get a current pose of the robot (x), which allows us to do feedback.

$$e = R(x_d - x)$$

where R is the matrix to rotate the error of the robot in the global frame to the local robot frame. We use this equation to find the current forward and cross-track error of the robot. The values in this matrix (e_x , e_y , e_θ) will be used later in Ramsete.

$$k = 2\zeta \sqrt{w_d^2 + bv_d^2}$$

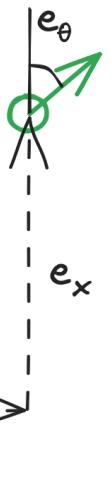
First, we start by calculating the intermediate k value that we will use later in the filter. b and ζ are tuning variables that we will discuss in detail later.

$$v = v_d \cos(e_\theta) + ke_x$$

In this equation, we calculate the commanded linear velocity of the robot. We adjust the linear velocity of the robot to move faster when it is pointed in the same direction as the desired pose, and slower or backwards when it is pointing in the other direction. In this equation, we also add the k value described earlier to have a correcting motion on the robot for distance error.

$$\omega = \omega_d + ke_\theta + bv_d \sin(e_\theta) e_y$$

Here, we calculate the commanded angular velocity, based on the desired angular velocity, the tuning variable, and a cross track error calculation.



After these calculations, we are left with commanded linear and angular velocity, which can't directly be sent to the robot. We must calculate the control outputs as left and right voltages from this velocity by first calculating the velocities of each wheel and sending them through a simple feedforward loop to create velocity from the inputs.

$$v_l = v - \omega * \frac{w}{2}$$

$$v_r = v + \omega * \frac{w}{2}$$

where w is the track width of the drivetrain (the distance between the sides of the drivetrain).

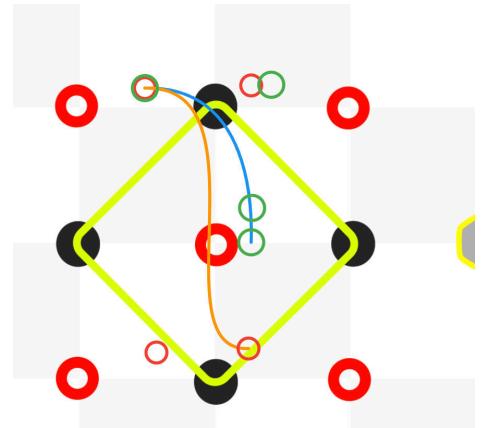
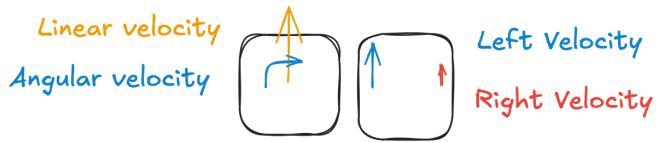
With these equations we calculate the commanded left and right linear velocities of the drivetrain. Now we use a [feedforward equation](#) (Pg. 65-66) to calculate the commanded voltage from the commanded velocities. We use the following equation to solve for the voltage:

$$f_{\text{drivetrain}}(v, a) = v * kV + a * kA + kS * \text{sgn}(v)$$

where kV , kA , and kS are all tunable scalars for velocity, acceleration, and static friction, respectively. We use this equation for calculating the velocity for VEX motors because it is an extremely good model to approximate the correct voltage for the motors, given that the back current is negligible for VEX motors (During experimental verification this assumption was shown to be good *enough* for VEX motors). We take the value calculated by the feedforward function from the control inputs to move the drive with a voltage command.

Testing

To test our algorithm we created a test path on the field in our path planner that does a forward(green) movement and a reverse(orange) movement to test the path following both directions of movement. We disabled the sensor models for localization, and set the drive noise to zero to allow us to move without considering the noise these models may add. We will do integration testing with localization later when we do skills tuning. We will initialize the robots belief as a point at $(0,0)$ and a heading at 0° . This will allow us to test the path correction capabilities as it should turn off to the right to correct the crosstrack error that exists at the beginning of the path. This path was tested on the real robot with a starting b and ζ tunable parameters set to zero to ensure the motion profiling implementation was working. After this verification we started increasing the b parameter, which is a roughly proportional value, slowly to get a ballpark value where the robot starts to oscillate. We would then increase the ζ , a roughly dampening value, to decrease the size and number of oscillations we have. When we tried to do this we ran into results much different than we expected. This was due to another `CWiseBinaryOp` happening on *auto* values in our code. Once we changed this to an `Eigen::Vector2f`, we were able to get the expected results and continue tuning the parameters. After this rough tuning we got values of 0.5 and 20 for b and ζ , respectively.



09/16/24 Design: Intake Optimizations (R.1.2.9)

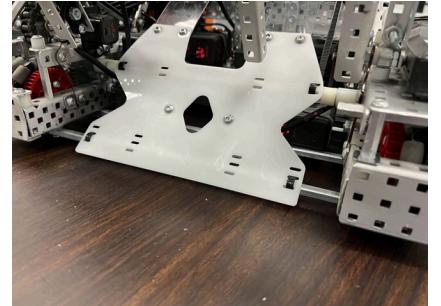
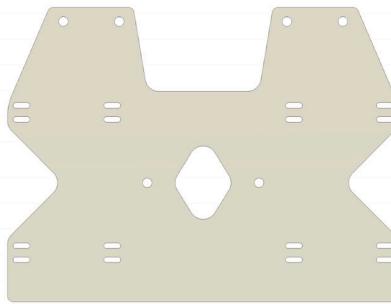
Designed by: Carl	Witnessed by: Alex	Witnessed on: 09/17/24
-------------------	--------------------	------------------------

Goal: Improve the efficiency of the intake's bottom stage.

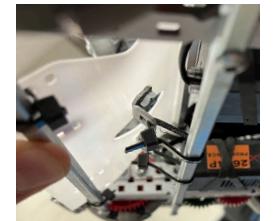
Despite our previous [improvements of this subsystem](#) (Pg. 170), the bottom stage was still not fast enough. This was again a result of the intake's slope being too steep. Here are two additional iterations of the intake acetal aimed at creating a curved surface for a more gradual transition:

2nd Iteration:

This part was designed to be forcibly bent using two screws to pull the center back via zip-ties to the drive crossbar. We created cutouts on the part to reduce the force required to bend it.



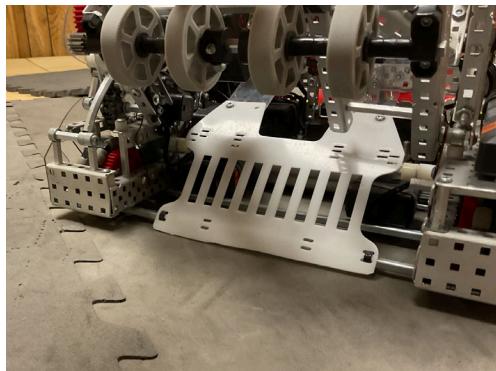
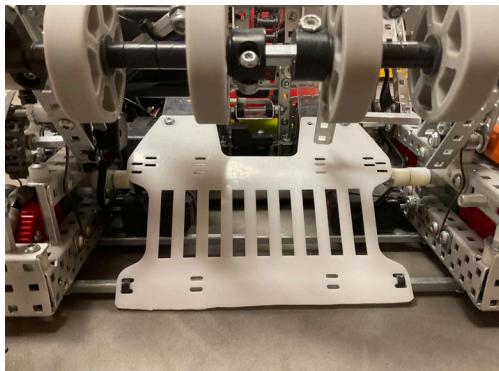
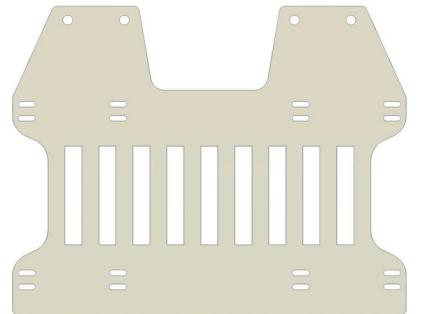
This new part significantly improved the speed at which rings were lifted off the tiles, however, the exposed screw heads impeded the rings ability to travel further up the intake. Additionally, the shape of the cutouts concentrated the bend in a very small space, causing the brittle acetal to snap easily.



3rd Iteration (current):

This iteration aimed at distributing the bend over a longer distance, significantly improving the part's strength. The additional flexibility of the part allowed rings to bend the ramp as they went up, meaning there was no need for screws or other methods of bending.

The performance of this part met our expectations and we will continue with this design for the time being.



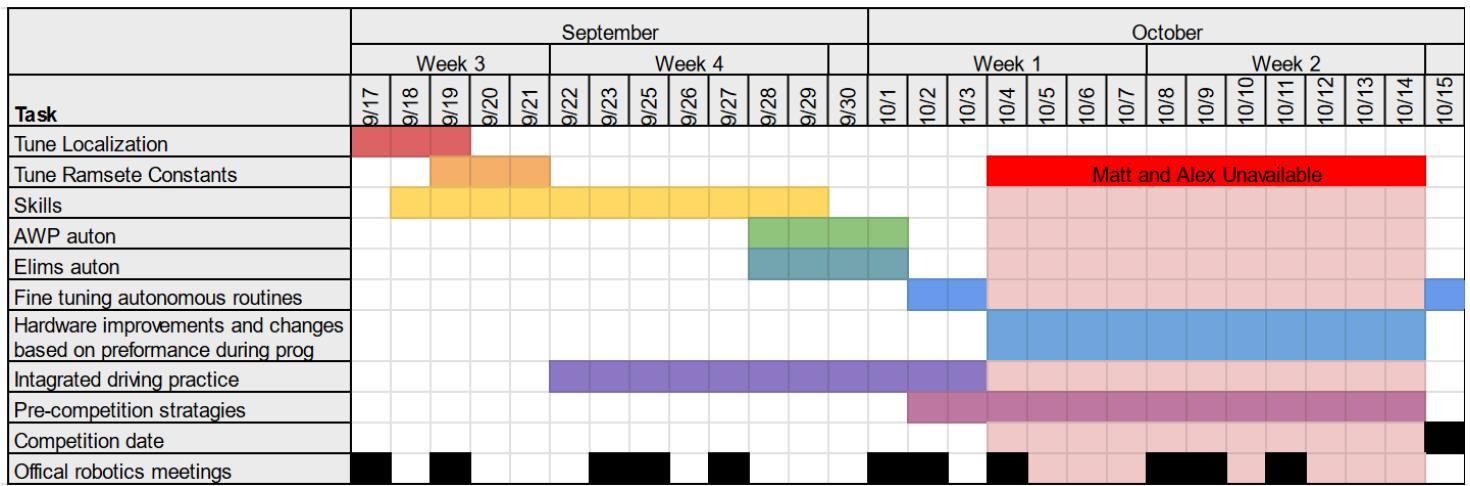
09/17/24 Time Management: 1st Tournament Timeline

Designed by: Carl	Witnessed by: Alex	Witnessed on: 9/18/24
-------------------	--------------------	-----------------------

Goal: Revise our timeline for the first tournament.

Adjusted Timeline:

The tournament we were originally planning to go to on the 28th of September has since been canceled. This means we have until our league on October 15 to prepare. Here is an updated Gantt chart featuring more detailed tasks and deadlines.



Rationale:

Before the competition, we aim to fully eliminate any physical problems that could affect the redundancy of our skills or autonomous. Our autonomous routines should all be completed by October 4th due to Alex and Matt being unavailable 10 days prior to the competition.

Any hardware modifications will be completed between the 4th and the 10th or during the weekend, maximizing programming and driving time while we have field access.

09/18/24 Design: AWP Autonomous Pathing

Designed by: Alex

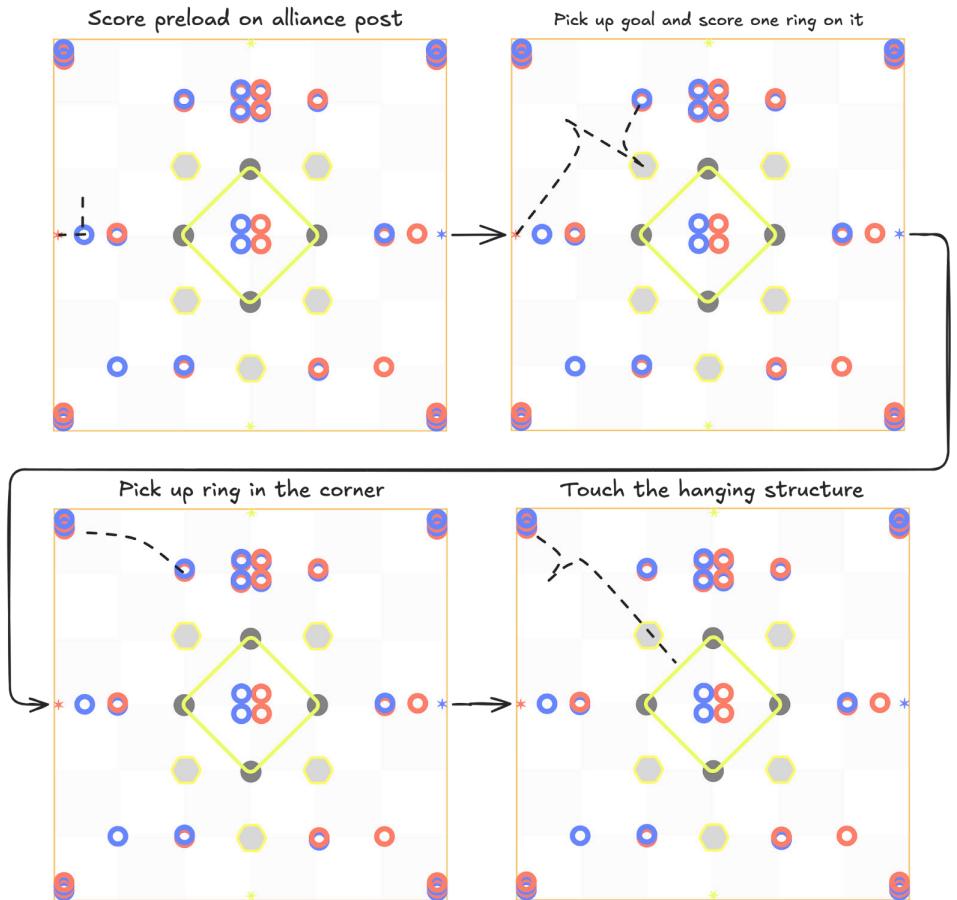
Witnessed by: Carl

Witnessed on: 09/20/24

Goal: Create an autonomous path to complete the win point by ourselves in autonomous.

The autonomous win point is crucial to our standings in the tournament rankings, so being able to get the autonomous win point by ourselves is crucial to maintaining a high ranking. For this reason, we want to create an autonomous routine that is reliable at achieving the [autonomous win point goals](#) (Pg. 19), which for local events is scoring 3 rings on stakes, 2 stakes with scored rings, touching the hanging structure, and both teams off the autonomous line. For this autonomous routine we are prioritizing simplicity to ensure we reliably get the autonomous win point. For this reason, we should only do what is required to reduce the chance of an error. Another consideration is allowing flexibility for our alliance. We would like to be able to run with any team, so we think it could be beneficial to have an autonomous for both the positive and negative corner that achieves the AWP objectives. We think the following path is ideal to prioritize the main autonomous win point objectives:

This auton first starts near the alliance stake, pushes the ring near the alliance stake out of the way, and scores on the alliance stake. Then we pick up the goal nearest to our negative corner on the field and put 2 rings on it from the stack in our negative corner and one of the red ones on the line. We then move to the hang structure and touch it with our lift.



09/19/24 Testing: Skills Path

Designed by: Alex	Witnessed by: Carl	Witnessed on: 09/20/24
-------------------	--------------------	------------------------

Goal: Test the full integration of skills path.

Localization Tuning

When we tested localization on the field we noticed that after we stopped the robot slowly drifted back to the correct position on the field. This was an issue because it indicated that somewhere in our particle filter the particles weren't being pruned enough during the driving and it was accumulating error. To fix this we decreased the drive noise parameter and the standard deviation of the distance sensor readings. After we made this change, the position of the robot converged much faster, so it was properly up to date while it was driving. This change made it so that the position was consistent enough that it could score on the neutral wall stakes without any terminal guidance (vision sensor).

Ramsete Tuning

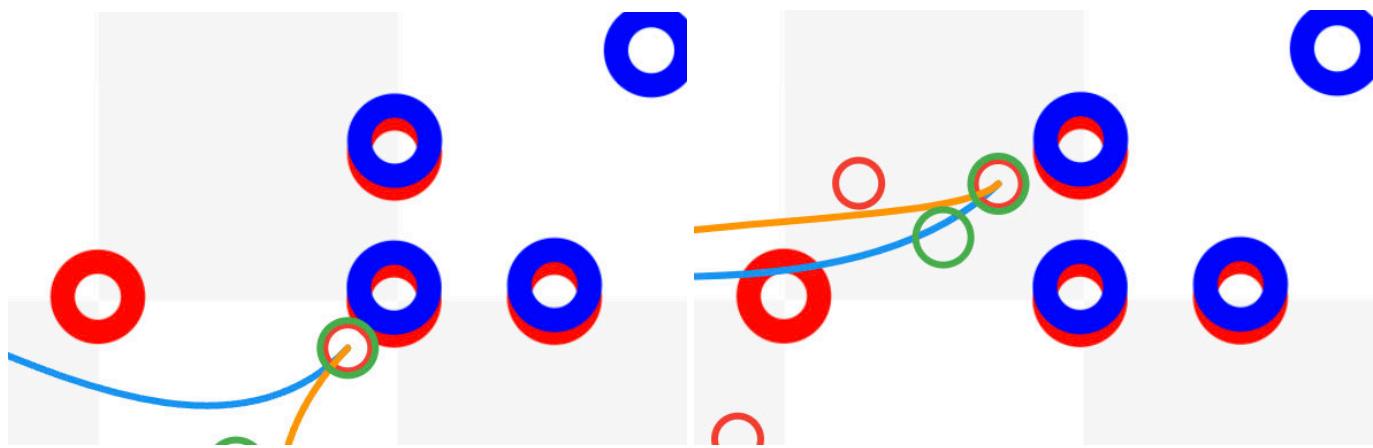
For Ramsete, we had 2 primary issues: oscillation issues on paths and inconsistent spins. To fix the oscillations, we reduced the b tuning value described in [implementing Ramsete](#) (Pg. 191-192), this resulted in significantly decreased oscillations in our paths, and resulted in sub-inch consistency in our Ramsete path following at slow-medium speeds. At high speeds, the drivetrain is saturated, which means that both sides of the drivetrain are already going at full speed. Currently, we can't think of a way to fix this at high-speeds, so we will have to make our paths slightly slower to avoid this actuator saturation problem. The inconsistent spins on the robot were related to our calculation of the angular error. In our implementation of the *angleDifference* function, original shown below on the left, we use the *fmod* operator from the `<cmath>` standard library. This function's result changes sign based on the input sign, which is inconsistent with what we expected the function to return. This was fixed by adding another *fmod* operation that ensures the input to this function is a positive *float*, ensuring that the angle is always within the $[-180^\circ, 180^\circ]$ values we expect from this function. After testing the function on the right, it fixed the abnormal spins that we were getting in Ramsete movements.

```
Angle angleDifference(const Angle x, const
Angle y) {
    return fmod(x.Convert(radian) -
y.Convert(radian) + M_PI, M_TWOPi) - M_PI;
}
```

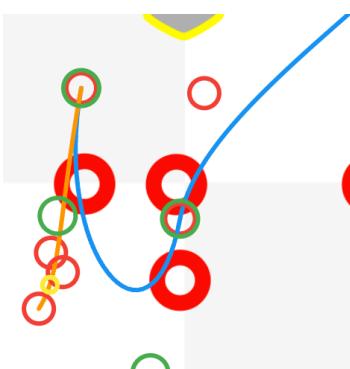
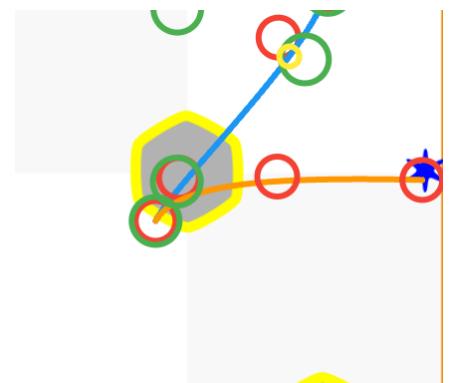
```
Angle angleDifference(const Angle x, const
Angle y) {
    return fmod(fmod(x.Convert(radian) -
y.Convert(radian), M_TWOPi) + M_TWOPi +
M_PI, M_TWOPi) - M_PI;
}
```

Skills Path Changes

We largely kept the path from [Design: Skills Pathing](#) (Pg. 182-183) above, with a few minor changes to make it more effective on the real robot. First, we had to drop the first goal we grabbed in the corner after we filled it. In the original skills route, we didn't drop it before getting another goal, so we chose to drop it in the red negative corner near where we collect the rings. Next, we changed which rings we get for the neutral stake so that we don't knock them as far in and block the corner from being scored. We first get the ring on the top left, and then we get the ring on the bottom left, instead of getting the one far to the right.



Another change we had to make was pushing the goal in front of the blue alliance stake out of the way so that we could turn in the tight space necessary. We found that pushing the goal made this consistent, instead of trying to squeeze in and turn, leaving the goal in an inconsistent position.



Another change we made was in the red positive corner of the field, where we would do a "bounce" type motion before, we wanted to change it to a smooth motion, similar to what we did in the red negative corner because that was very smooth and worked well.

For the rest of the path, the planned path matched what we did on the real robot, with small deviations to grab rings at slightly different angles.

Results

At the end of the day we were able to consistently score around 50 points, which is around the high programming score of 53. During our testing the neutral stakes were around 95% consistent with the programming, which goes to suggest that the localization and path following are within the goal of 1 in consistency and accuracy, given the precision needed in this maneuver.



Conclusion

- **Localization tuning**
 - Increased drive noise - allowed larger variety of particles
 - Zeroed minimum particle weight - culled lower-weighting particles sooner
- **Ramsete**
 - Constant tuning
 - Ensure consistently signed angle error
- **Pathing**
 - Make movements require less precision
 - Less turn movements

Based on the very successful testing today, we think that we can complete the skills path to a point we are satisfied with after the Monday meeting. This is much ahead of the intended timeline for this change, we think we should create a new path before the tournament and analyze if it is worth our time to implement it. With this success, the consistency, and high-score of the programming skills, we believe that it would be beneficial to have the programming skills run during driver control. With this being said, however, we do think it would be useful to have an override button during driver in case it messes up to improve reliability.

09/21/24 Time Management: Comp Dates & Priorities

Designed by: Carl	Witnessed by: Alex	Witnessed on: 9/22/24
-------------------	--------------------	-----------------------

Goal: Create an understanding of the tournaments we will be going to and how competitive we think they will be.

By creating a google sheet with tournament names and dates, we can automatically count the gap we have to prepare for any given tournament using the formula “=ABS(DAYS(C#,C#))”. Additionally, for all tournaments, we rank what we believe the competition level will likely be leveraging our experience to do so. Both of these numbers will have a large influence when building our schedule for the season.

Notes:	Tournament Name:	Date:	Preparation Days:	Expected Competitiveness:	Notes:
Waitlist	N/A (Current Date)	09/21/24	N/A	N/A	
	Butter Nexus League III (Q1)	10/15/24	24	1	
	SVVSD Erie High	10/26/24	11	2	
	Butter Nexus League III (Q2)	10/29/24	3	1	
	Butter Nexus League III (Q3)	11/12/24	14	2	
	Butter Nexus League III (F1)	12/03/24	21	2	
	Roosevelt	01/18/25	46	2	
	Kalahari Classic (Day 1)	01/24/25	6	5	
	Kalahari Classic (Day 2)	01/25/25	1	5	
	SVVSD Silver Creek	02/08/25	14	2	
by qualification	AFCEA Pikes Peak (Day 1)	02/10/25	2	4	
	AFCEA Pikes Peak (Day 2)	02/11/25	1	4	
	State Championships (Day 1)	03/08/25	25	3	rebuild here
	State Championships (Day 2)	03/09/25	1	3	
	Worlds Championship (Day 1)	05/06/25	58	5	
by qualification	Worlds Championship (Day 2)	05/07/25	1	5	
	Worlds Championship (Day 3)	05/08/25	1	5	

09/21/24

Strategy: Skills Repathing

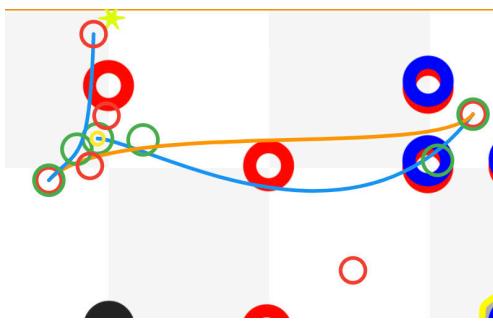
Designed by: Alex

Witnessed by: Carl

Witnessed on: 09/22/24

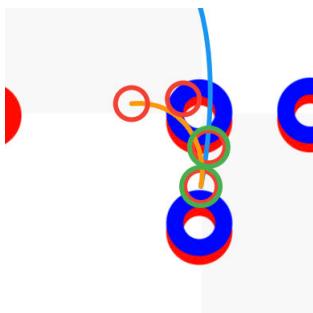
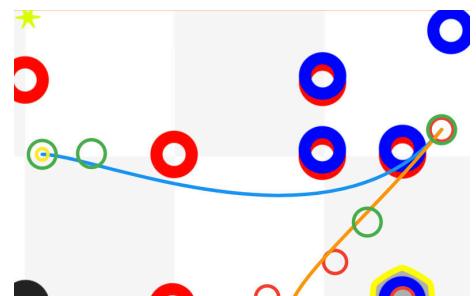
Goal: Analyze skills pathing to analyze the benefits of switching possible paths.

Based on the success of [skills path testing](#) (Pg. 196-198) we think that it would be beneficial to reconsider the skills path that we will use. Currently, without time optimization, we have 3-5 extra seconds in skills, so we have considerable time to add extra parts. So far our path scores all but 1-2 of the red rings, meaning we have to start scoring the blue rings to increase our score. The blue rings can only be scored as the top rings, with a red ring on the goal below it. This means that we can't score on the goals with blue rings already on them, but we can score more on the neutral stakes to give us more scoring capacity. Therefore, we must heavily consider the possibility that we miss a ring in our scoring calculations. We think that without going too far out of the current path, we could add the following rings to our path to increase our score by 4 points by scoring 3 blue rings. We will do this with only one extra scoring movement, at the first neutral stake we will score twice, and the other blue rings are the rings that we get from the stacks of red blue rings that we already collect the bottom red from. We have made the following adjustments to the skills path:



First, we score another set of red and blue rings on the neutral stake. This allows us to get a blue top ring and also a red ring below it, giving us 2 more points and all the reds left on the field scored.

Next, where we got the red ring at the bottom of the stack for the blue alliance stake, we are going to get both rings and score them on the alliance stake. This also opens up the corner for when we push the goal with the blue ring on it into the corner, giving us more consistency with that movement.



Finally, we get two of the red rings, the top left and bottom one in the photo to the left. Then we collect the red and the blue rings to the right and score them on the goal.

09/22/24 Time Management: Season Timeline

Designed by: Carl	Witnessed by: Alex	Witnessed on: 9/23/24
-------------------	--------------------	-----------------------

Goal: Create a timeline for our entire season with scheduled rebuilds.

Member Unavailability:		
Name:	Dates:	Tournaments Missed:
Carl	11/12/24–11/16/24 11/23/24–12/02/24	11/12/24 League
Alex and Matt	10/04/24–10/14/24 12/26/24–1/02/25	None

Timeline (Based off of [Pg. 191](#)):

Season Timeline:		Robot 1		Robot 2		Robot 3		Carl Unavailable				Alex & Matt Unavailable					
	Sep.	October (10)				November (11)				December (12)				January (1)			
Task	Week 4	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4
Design																	
Disassembly																	
Build																	
Test																	
Programming																	
Competition				15th	26&29th		12th			3rd					18th	24th	
		February (2)				March (3)				April (4)				May (5)			
Task	Week 1	Week 2	Week 3	Week 4		Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4
Design																	
Disassembly																	
Build																	
Test																	
Programming																	
Competition			8&10th				1st								6th		

Methods:

Our rebuild schedule is based on what tournaments we have big gaps in between, how competitive we think the following tournament will be, and balancing gaps between rebuilds. For both robot rebuilds, we favored a shorter window to rebuild, but before less competitive tournaments. This means that for Kalahari Signature and Worlds, our robots will be competition tested and thoroughly optimized. Additionally, because we do not have enough parts to build 2 robots at a time, we are forced to disassemble the previous robot before constructing the new one, emphasizing the need for a complete CAD.

09/22/24 Strategy: Competition Analysis

Designed by: Matt	Witnessed by: Carl	Witnessed on: 09/25/24
-------------------	--------------------	------------------------

Goal: Analyze the competitions that happened this weekend and the designs in them. See if there are any novel strategies or designs to implement.

Important Events:

- **Highlander Summit Signature Event (DAY 1: [Highlander Summit NJIT - 9/21/2024](#), Day 2: [Highlander Summit NJIT - 9/22/2024](#))**
- **Howling Halloween At The Creek (<https://www.youtube.com/live/HfrAYMHnWgg>)**

Notable Strategies:

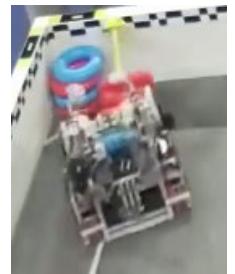
1. Early Corner Hold: Similar to MOA, teams opted to quickly put at least one mobile goal in the “+” corner to ensure ownership. Different from MOA, however, instead of staying in the corner for the remainder of the match, teams would come out to score on wall stakes or put opposing mobile goals in the negative corner.



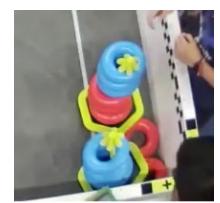
2. Tipping Own Goals: A new strategy that we saw at the Highlander Signature Event was teams tipping goals with their alliance rings over. This is due to the rule change on [September 3rd Update](#) (Pg. 175-176) allowing rings to be scored even when contacting the tiles. Tipping their own goals made them difficult to pick up, reducing the fear of the opposing team putting it in the negative corner.



3. Negative Plays: With teams coming out of the corners to play wall stakes, we saw an increasing amount of teams gaining control of opponents' mobile goals and putting them in the negative corner. This was especially common when teams went to hang or do wall stakes.



4. Opponents rings in corner: A rare strategy we saw in some matches was teams would put opposing teams rings in the corner then would put a mobile goal on top. This made it hard for teams to get the mobile goals out and to get their rings to score.



(all images taken directly from event live streams)

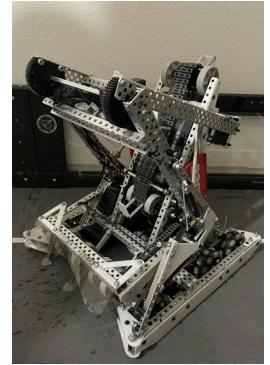
Notable Designs:

1. Hook with Arm “Lady Brown” (18522R/229V):

This design is a hook intake that can score on both the mobile stakes and alliance stakes. It also includes a wall stake mech that can divert one ring at a time into a motorized arm. This design was initially developed by 18522R but improved by 229V.

Here are some pros and cons of this design:

Pros:	Cons:
<ul style="list-style-type: none"> • Simple • Compact • Easy to build • Can score in all ways • 6 motor drive • Very fast at mobile stakes • Very fast at wall stakes 	<ul style="list-style-type: none"> • Can only hold one ring in wall stake mech • Arm Reach • Mobile stake is easy to steal • 1 motor intake



(Image from 229V)

Note: We didn't encounter any other designs significantly different from what we already [analyzed](#) (Pg. 115-117)

09/23/24 Testing: Skills Repathing

Designed by: Alex	Witnessed by: Carl	Witnessed on: 09/23/24
-------------------	--------------------	------------------------

Goal: Test skills and note all inconsistencies with the plan.

After [repathing skills](#) (Pg. 200), we are going to retest the path to ensure it works with the changes we made.

Increasing Speed

In our repathing analysis we thought that there would be plenty of time for the changes we made, however, that turned out to be quite wrong requiring at least 5 seconds of optimizations to fit in time. First, we sped up how fast the robot would move while going on straight movements. After increasing the path speeds to a max speed of 50 inches per second, we had cut about 2.5 seconds from the run. Next, we could reduce the amount of time the robot spent waiting for turns in the run. This cut off an additional 0.5 seconds, giving us 2 more seconds of time optimizations needed. The next change we made was to put the lift up before we get to the neutral stakes to score on them, saving about 1 second from scoring on the neutral stake, leaving only an additional second to cut out. The final change we had to make was to modify high-curvature movements in our paths. When the robot does a high curvature movement, it needs to slow down the robot's speed so the drivetrain is not oversaturated (one side of the drive moving over the maximum speed allowed by the drivetrain). By making the movements in the path smoother, it allows for the robot to better maintain speed during movements, giving us another second of time, giving us plenty to complete the path.

Consistency

When testing the skills path, even pre-time optimizations, we noticed that it was not very consistent at scoring all the red rings, therefore losing us points when scoring the blue rings. Out of the 15 runs that we did on 9/23, we were never able to intake or score all of the red rings correctly. It is for this reason that we concluded that we would need to change the path to go slower and increase the consistency for blue rings to be feasible.

Conclusion

- **Path barely fits in time**
 - To make the path fit in time we had to cut many corners
 - On the edge of what our robot and path following algorithms are capable of
- **Reliability**
 - Corner-cutting was needed to fit in time
 - Because of this the run suffered reliability issues

This is why we think it would be more time effective for us to tune a programming skills routine with our previous path so that we get to a super high level of consistency.

09/28/24 Testing: Original Skills Path

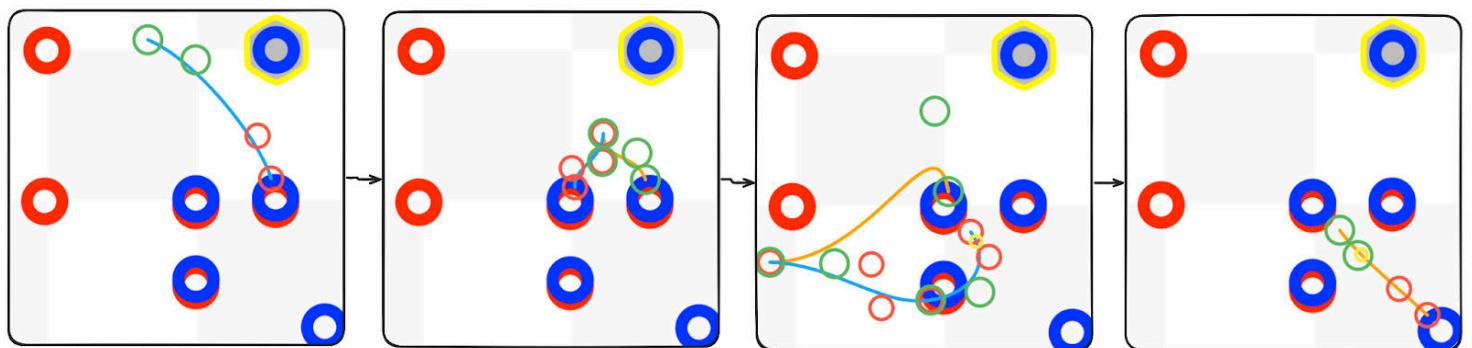
Designed by: Alex

Witnessed by: Carl

Witnessed on: 09/30/24

Goal: Test skills and note all inconsistencies with the plan.

To test the original skills path we used our [Git version control](#) (Pg. 75) to branch the new path changes and revert to the current main branch, which had the stable skills path backed up. After reverting, the code worked the same as it had before. To increase the score slightly we wanted to change how it got the rings in the blue negative corner so we would reliably collect all of the red rings. Before we were trying to knock the blue rings off the top of the red rings, but we decided it would be a better approach if we specifically picked the red ring under each of them because it would increase the consistency and reduce the chance that we intake a blue ring.



This caused a few extra seconds, so we had to go back to do the same time optimization that we discussed in the [previous skills repathing](#) (Pg. 204). We did everything but speeding up the paths, and that reduced just enough time to fit the hang into the end of the run. Through our testing these changes were enough to fit a passive hang in time, however we couldn't test it as the hang is not completed yet, however the run has been tested thoroughly and all other parts remain working. The next step will be testing the path with hang once it has been built.

09/28/24 Testing: AWP Path

Designed by: Alex

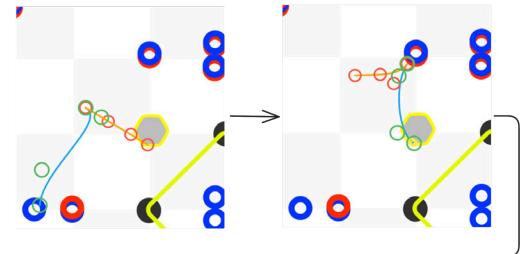
Witnessed by: Carl

Witnessed on: 09/30/24

Goal: Test skills and note all inconsistencies with the plan.

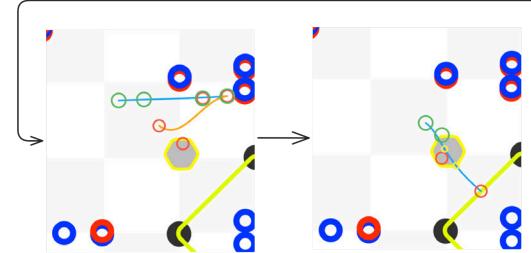
Pathing

This path's (shown upper left) testing went right to plan without any significant modifications. The largest problem we had to fix was the location of the goal grab, which we fixed by moving the point, illustrated in the lower left.



Path Mirroring

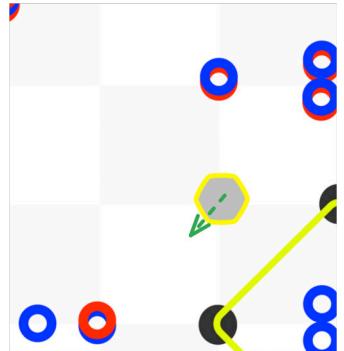
After we tuned the auton for the red side of the field we wanted to test the side flipping capabilities in our [path planner](#). To do this we have a red and blue flipped version of the path that we generate at runtime by flipping all the y values. Upon testing, the robot seemed to move in the same path on the blue side. Currently, we store both the red and blue alliance paths on the brain, limiting the number of auton paths we have severely, which is something that we should look into for next steps once we start using more autons.



Testing

Method:

1. Run the same autonomous 5 times
2. Alternate sides of the field
3. Only change if it fails



Alliance	AWP Tasks Successful	Changes
Red	All tasks successful	None
Blue	All tasks successful	None
Red	Goal grab failed	Moved goal grab position to be closer to + corner
Blue	All tasks successful	None
Red	All tasks successful	None

10/08/24 Design: Hang & Front Arm Updates (R.1.2.11)

Designed by: Carl, Alex

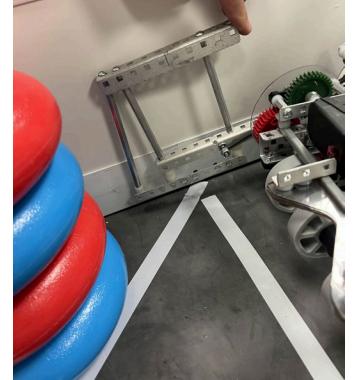
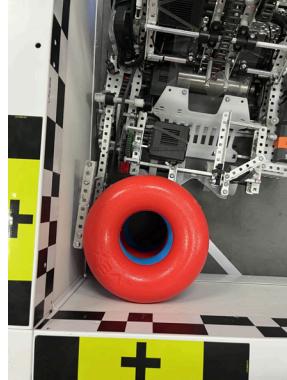
Witnessed by: Matt

Witnessed on: 10/12/24

Goal: Change the front arm to a corner sweeper and add a tier 1 hang.

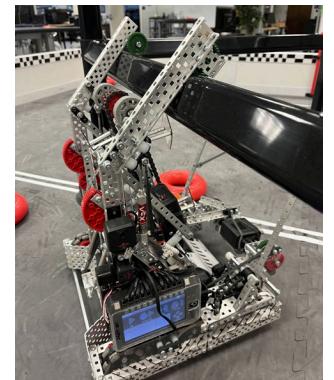
Corner Sweeper

As we were practicing driving, it became readily apparent that our current corner arm was not efficient enough. This was primarily due to the fact that it could only remove one ring at a time. By removing the left arm and crossbar, we were able to create a single arm that can fully sweep the corner in less than 2 seconds.



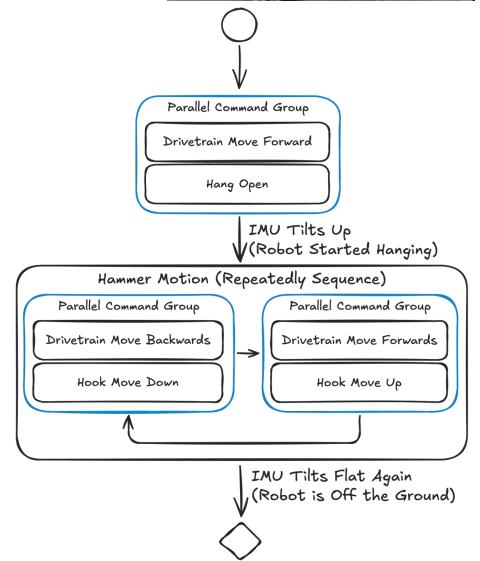
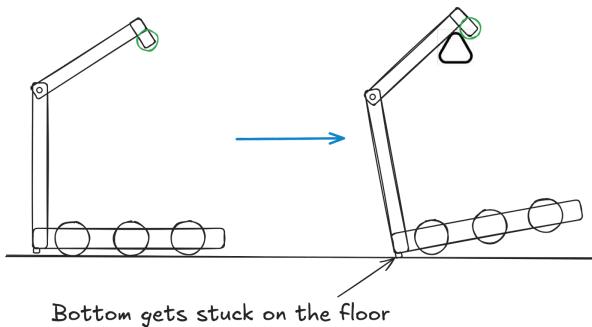
Hang

For this robot, we believe that a hang is a fast and consistent way to score points in matches while also being an easy way to boost our skills score. Because this robot was not designed with any form of winch system, we have no way to power a high hang. However, by utilizing small piston actuated arms mounted to the lift uprights, we can utilize the momentum of the robot to elevate by approximately $\frac{3}{16}$ of an inch. The construction of the hang was intended to be as lightweight as possible while also being easy to modify in order to change the balancing point.



Hang Programming

Our hang can get stuck on the back high strength shaft while it is hanging, as shown in the figure below. We get around this by driving into the hang at a very consistent speed that makes it hard for us to get stuck permanently while hanging. We then use our front arm and drive to make the most vibrations possible to overcome the friction between the robot and the tiles. To do this most efficiently we made a macro that is shown on the right.



10/18/24 Analysis: Butter Nexus League (10/15/24) Skills and Matches

Designed by: Alex, Matt

Witnessed by: Carl

Witnessed on: 10/21/24

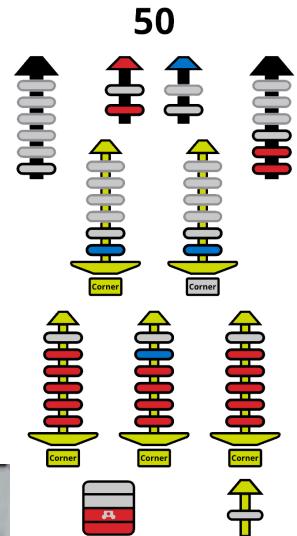
Goal: Analyze how we performed in skills and matches at the first league event.

Skills

Overall, our skills scored fairly high at this event with our total score being 100 (50 autonomous coding skills, 50 driver skills) putting us 4th in the world. This was well below our theoretical max of 120 (60 in both), therefore we would like to analyze each run to determine what happened so we can improve for the next event.

Driver Skills #1: 50 points

For our first run of the day we chose to run a driver skills, so if the program messed up we could override it for the remainder of the run. In this run we were successfully able to grab all the mobile stakes and fill them with rings, only missing 2, except on the last goal the blue ring from the corner was pushed in front of where the robot needed to go and it was intaked instead of a red ring, losing 3 points. Additionally, in this run we missed the blue alliance stake and the first wall stake, reducing our score by another 7 points. Overall, with every goal grabbed and every ring intaked, this was a successful run, however there are still multiple places we could improve for the next competition with just programming.



Missed wall stake



Missed blue alliance



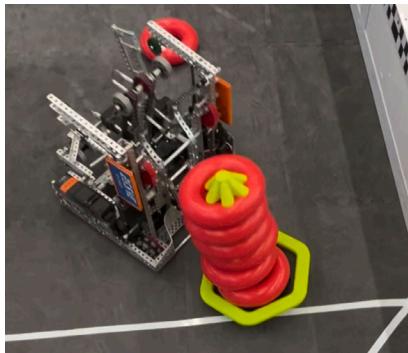
Blue ring in the way

Prog Skills #1: 50 points

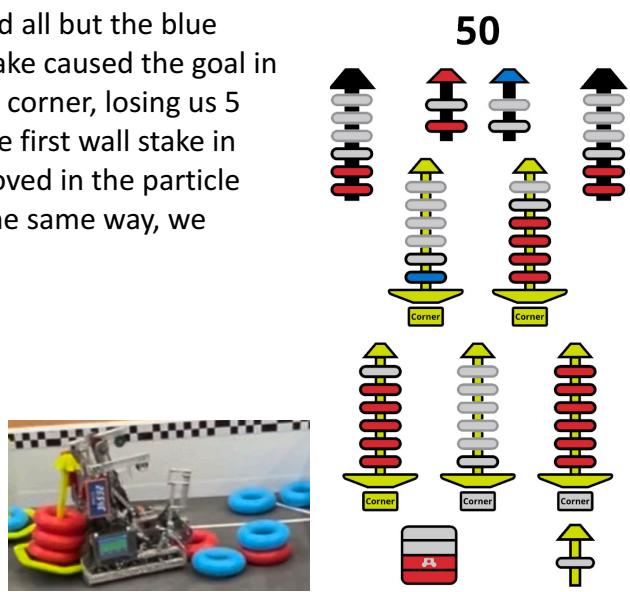
In this run we again scored a 50, however in this run we scored all but the blue alliance stake losing 3 points, however the goal clamp and intake caused the goal in the close right corner to not drop into a scored position in the corner, losing us 5 points. Notably in this run we didn't change the position of the first wall stake in code, so there are some inconsistencies that need to be improved in the particle filtering or RAMSETE. However, as the alliance stake missed the same way, we changed that by 2 cm for the next run.



Miss blue alliance stake



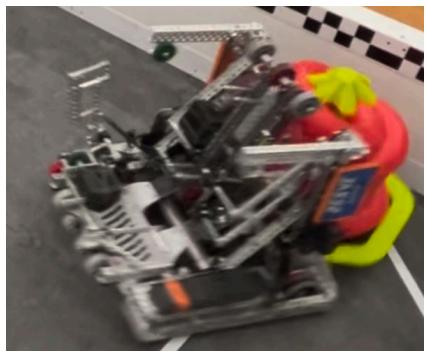
Goal dropped late



Blue ring in the way

Driver Skills #2: 50 points

In this run we scored 50 again, however we missed the corner again, and we got a blue ring on the top again. But in this run, we scored on all the stakes, which means our corrections worked. This run failed on the blue the same way with that ring in front of the red ring, even though we tried to fix it. In this run the outtake snagged on the 5th ring on the goal as shown in the following picture. This run was done on a different field which had issues with the field bowing out without the field straps. We should consider this issue in our algorithms and test with this case to ensure it works in this potential case to ensure our redundancy is high.



5th ring stuck on the hook

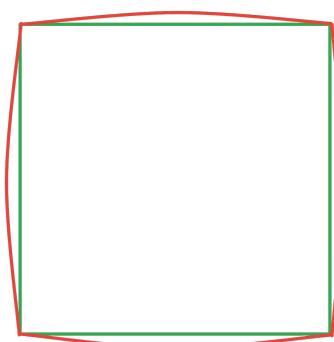
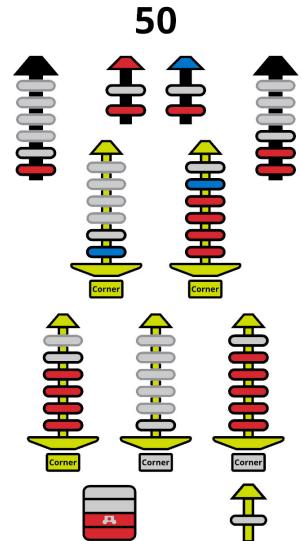
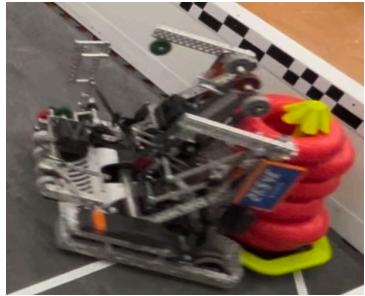


Illustration of bowed field (in red)

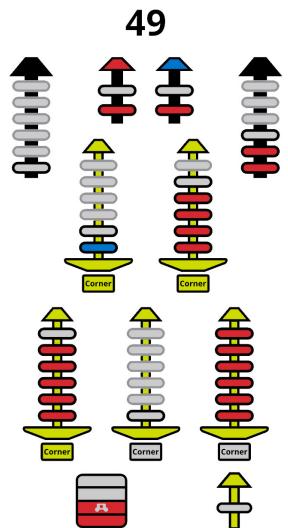


Prog Skills #2: 49 points

In this run we had the same issues with missing the first stake, and the corner for the last goal. There wasn't anything especially notable that happened in this run that was different from our other runs, although it does indicate that we need to improve our wall stake consistency.



Hook stuck on 6th ring



Overall at the competition our skills scored extremely reliably, however it was lower scoring, but we think we can pretty easily fix this run for the next league day.

Skills Next Steps:

- Fix corner goals
 - Goals getting snagged by the intake after the corner/not falling down soon enough
 - To fix this we will use our de-jam state that moves the lift up while out-taking to make it impossible to stay stuck on the goal.
- Fix wall stakes
 - We missed a couple wall stakes, mostly the alliance stake, so we will work on tuning the position of these targets in the next week.

Matches

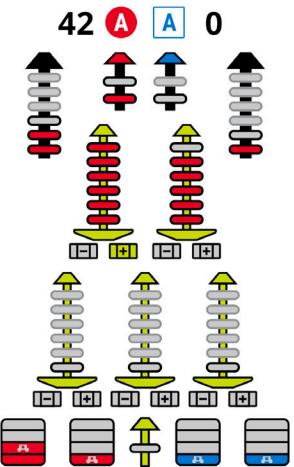
We went 1-1-0 in qualifications for the first league day, with 1 autonomous win point. We are currently ranked 7th at the competition. While matches were relatively successful, there were a few critical mistakes that needed to be addressed.

Qualification 2: Win

In this match we got an autonomous win point by running our AWP code which is able to push our alliance off of the starting line. We filled our goal with 6 rings and went to the positive corner, waited until our alliance finished filling up their goal then switched places. This allowed us to utilize our wall stake ability to increase our lead. At the end of the match, we executed a quick hang for three additional points.



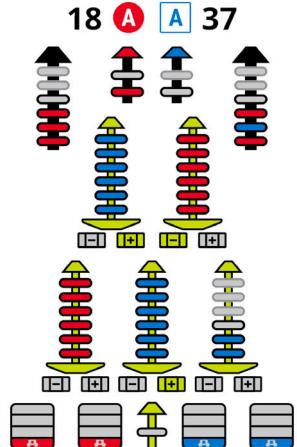
Pushing Alliance over starting line



Qualification 6: Loss

In this match we were not able to get the autonomous win point due to an unknown issue that resulted in our autonomous running out of time. At the start of the match, we were able to quickly fill up a mobile stake along with our alliance partners. While our alliance held a goal in the positive corner we filled up the wall stakes making sure we maintained control over the top ring. Towards the end of the match our alliance left the corner and our opponents put theirs in their place. When we set our stake down to hang, it got quickly put into the negative. This all resulted in a loss for that match.

Both of these scenarios could have been prevented by more thorough communication with our partners and taking less unnecessary risks when we are leading in points.



Conclusion

- Continue to tune skills
 - Many small areas of improvement to consistency
 - Work on dropping goals
- Maintain corners during matches
 - Don't take unnecessary risks
- Watch out of negative corner plays
 - Include dropping the mobile stake in our hang macro

Additionally, in both matches and skills we accidentally intaked rings of the wrong color, which either caused us to score them on our mobile stake or waste significant time ejecting them, highlighting the need for automated ejection and sorting capabilities.

10/21/24 Design: Color Sorting

Designed by: Alex

Witnessed by: Carl

Witnessed on: 10/21/24

Goal: Create an algorithm to sort out rings of the opposite color in matches.

Problem

As concluded from our [Butter Nexus League analysis](#) (Pg. 208-211), we think it is important to add a ring color sorting feature to our codebase. To do this we first need to add a sensor to detect the color of the rings. However, we do not have the [optical sensor](#) so we will have to find a way to detect colors without it. To do this we want to repurpose the AI vision sensor on the back of our robot that we no longer use by turning it around, allowing it to sense when rings are in the intake.

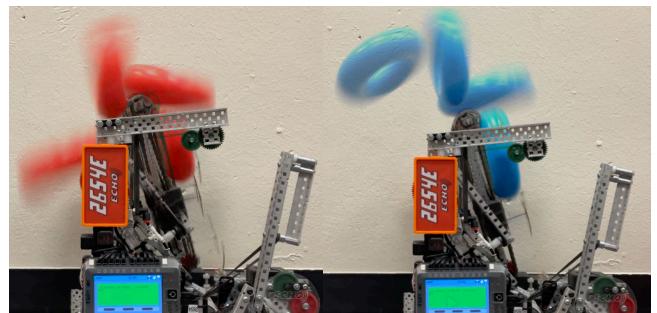
Sensors

To fit the vision sensor on the robot, we simply reversed the original AI vision sensor so it sees into the back of the intake. We then use the distance sensor on the intake as a trigger and request the largest red or blue object from the vision sensor. We assume the largest object is the color currently in the intake and then we track when the ring goes out of range and assume that is the color on the intake.



Algorithm

For ejection we used triggers on the distance sensor in the intake to detect when there was a ring at the bottom of the intake. We will wait until the ring is on the intake's top stage and away from the sensor to determine which hook it's on. Then, we schedule the intake to eject this ring once it gets to the top by driving the intake backwards slightly. This causes the ring to fly off because it maintains inertia and continues straight without the hook to redirect it onto the stake.



Overlay images of blue ring ejection vs red ring

Testing Data

All testing went to plan, with the vision sensor and distance sensor able to accurately detect which ring is on the intake and on which hook, later ejecting it at the right time.

Trial #:	Correctly Sorted	Incorrectly Sorted	Consistency
1	6	0	100.00%
2	6	0	100.00%
3	6	0	100.00%
4	5	1	83.33%
5	6	0	100.00%
Average	5.8	0.2	96.67%

10/21/24 Design: Alliance Color Determination

Designed by: Alex

Witnessed by: Carl

Witnessed on: 10/21/24

Goal: Design code to “find out” which side of the field the robot is on at the start of the match.

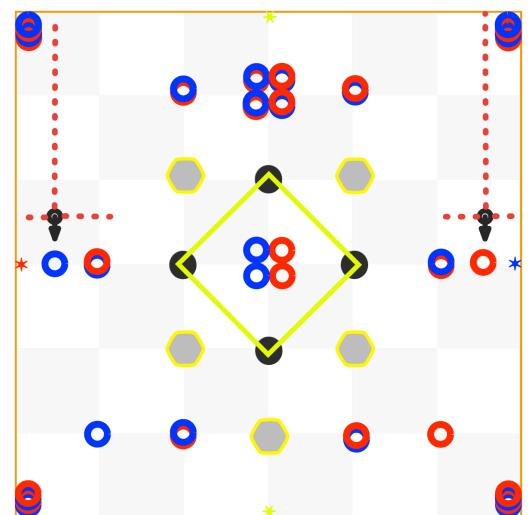
Problem

Currently, for our auton selection we list each auton with its alliance for the field mirroring. This means we are limited to having only 3 auton routines on the brain at a time, when it would be ideal if we could get somewhere around 6. To fix this we want to utilize the sensors on the robot to detect which alliance the robot is on, leading to higher reliability and easier use.

Autons	
RAWP	BAWP
RAWP_PUSH	BAWP_PUSH
RSKILLS	

Algorithm

To most effectively compare the sensor readings to the predicted locations on the field we will take advantage of the particle filter. By representing the potential red and blue starting positions as particles and comparing the weights of these particles, we can make a fairly confident guess that the higher weighted particle would represent the robot’s alliance. For example, in the picture on the right, the robot would clearly be able to differentiate between which side it is on because one particle would be much higher weighed than the other.



10/21/24 Design: Eliminations Autonomous Routine

Designed by: Matt, Alex

Witnessed by: Carl

Witnessed on: 10/22/24

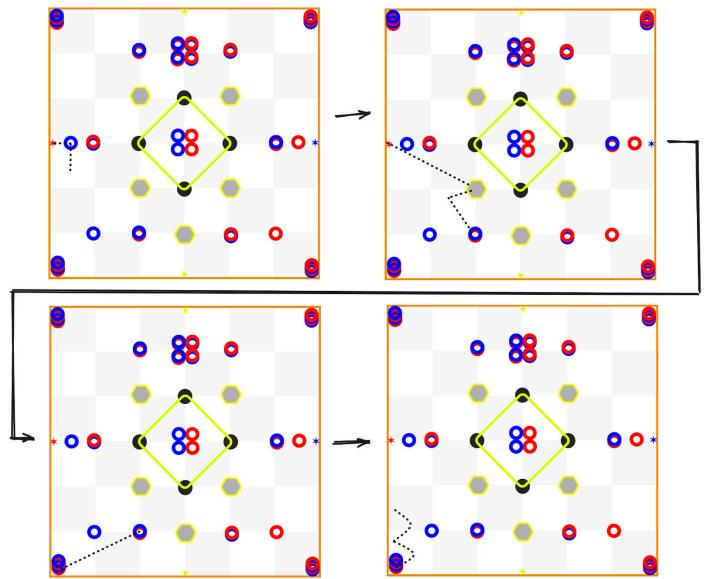
Goal: Make multiple higher scoring autonomous routines for elimination matches.

Currently, we have only developed an "[AWP autonomous](#)" (Pg. 195), which is relatively low scoring. In elimination matches the Autonomous Bonus of 6 points can easily win or lose a match, so a high-scoring autonomous is crucial. Beyond being high scoring, we would like to have the flexibility to work with our alliance and work on the positive or negative side of the field, allowing them to run the autonomous routine they are more comfortable with.

Positive Corner: 8 points

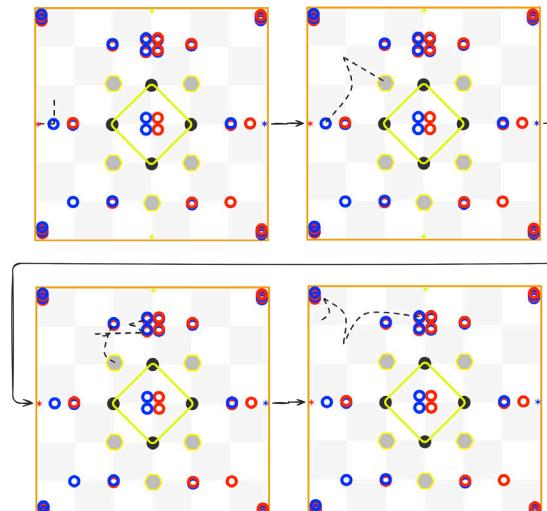
In the positive corner (bottom of picture on the right), there are significantly less rings that are available to score. This is why we are going to try and utilize the [arm on our robot](#) (Pg. 207) to sweep out the rings in the corner and the [color sorting](#) (Pg. 212) to only put our alliance color rings on the goal. In the route shown on the left, we score on the alliance stake, similar to how we would in the [AWP autonomous](#) (Pg. 195) and grab the goal, fill it with the closest rings, and finally get all of our rings from the corner.

1. Alliance stake
2. Grab goal
3. Intake Rings
4. Corner



Negative Corner: 10 points

For the negative corner we already have the AWP autonomous that gets 2 red rings, and there is another next to the center line that we are planning to get. After this, we will clear the corner in a way similar to the positive corner and also score those rings on the goal. The route is shown on the left.



1. Alliance stake
2. Grab Goal
3. Intake Rings
4. Corner

10/23/24 Strategy: Match Play

Designed by: Matt, Carl

Witnessed by: Alex

Witnessed on: 10/23/24

Goal: Develop a match strategy to ensure control in each match.

Based on our match experience at the Butter Nexus league and match analysis from Signature Events, we are developing a strategy to ensure control of the positive corner and top rings on wall stakes. Additionally, we want to implement a strategy that is consistently high scoring regardless of what our opponents do, which will help prevent losses when we are not able to fully scout our opponents. We have already determined that achieving early control of the positive corners is very closely related to winning the match ([Pg. 114](#), [Pg. 202](#), [Pg. 210](#)), so that is a huge consideration in our strategy. In order to identify the most ideal time to score on wall stakes, we chose to make a table demonstrating the pros and cons of doing tasks in different orders based on our driving practice and scrimmages with other teams.

Task Order	Pros	Cons
<ol style="list-style-type: none"> 1. Fill 6 on negative stake 2. Fill mobile goal on negative side 3. Fill 2 on positive wall stake 4. Go to positive corner 	<ul style="list-style-type: none"> ● No need to return to the negative corner. ● Minimal effort required for the remainder of the match. 	<ul style="list-style-type: none"> ● Risk of losing the positive corner.
<ol style="list-style-type: none"> 1. Fill 2 on negative wall stake 2. Fill mobile goal on negative side 3. Fill 2 on positive wall stake 4. Go to positive corner 	<ul style="list-style-type: none"> ● Control over the top stakes. ● Positive corner is easily accessible. 	<ul style="list-style-type: none"> ● Potential need to return to the negative corner. ● Rings positioned in front of the wall stake.
<ol style="list-style-type: none"> 1. Fill mobile goal on negative side 2. Fill 2 on negative wall stake 3. Fill 2 on positive wall stake 4. Go to positive corner 	<ul style="list-style-type: none"> ● Control over the top stakes. ● Rings are out of the way. ● Positive corner is easily accessible. ● Avoid returning to the negative side. 	<ul style="list-style-type: none"> ● May need to wait for the wall stake. ● May lose control of the positive corner
<ol style="list-style-type: none"> 1. Fill mobile goal 2. Go to positive corner 3. Wall stakes as needed 	<ul style="list-style-type: none"> ● Positive corner is easily accessible. 	<ul style="list-style-type: none"> ● Risk of losing positive corner ● Only 1 top stake ● Other teams may take wall stakes

Decision:

After analyzing match play and evaluating the advantages and disadvantages of each strategy we ultimately chose to implement the 3rd option. As follows:

1. Fill mobile goal on negative side
2. Fill 2 on negative wall stake
3. Fill 2 on positive wall stake
4. Go to positive corner

This strategy offers several benefits. First off, it guarantees control of at least one of the positive corners early in the match which is crucial to maintaining control and accumulating points. Additionally, it allows us to take control over both wall stakes putting us in a good position for the rest of the match. Another advantage is the efficiency; by filling up the mobile goal on the negative side before scoring wall stakes, it reduces the need to come back to the negative side of the field for rings which streamlines our movement and decreases the time spent out of the positive corner.

Drive Practice:

To implement these strategies, we needed to test them against other robots. Over the past few weeks, we've played matches against one of our sister teams (2654G, Grape Soda) to understand how other robots might impact our strategy. Additionally, we've practiced scoring after running autonomous on a fully set-up field to rehearse the task order without other robots interfering. With all this practice, we hope to be well-prepared on competition day to effectively execute our strategy and react to other robots when they intervene.

Conclusion:

- Evaluated a variety of strategies
 - Best strategy
 - Fill mobile goal on negative side
 - Fill 2 on negative wall stake
 - Fill 2 on positive wall stake
 - Go to positive corner
- Early control of the positive corner is key to match success and maintaining control
- Efficient movement throughout the field is crucial

10/23/24 Testing: Eliminations Autonomous Routine

Designed by: Alex

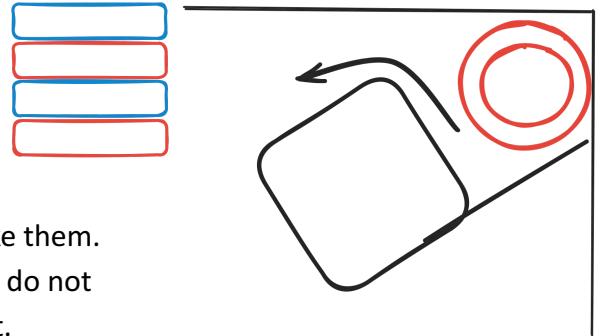
Witnessed by: Carl

Witnessed on: 10/23/24

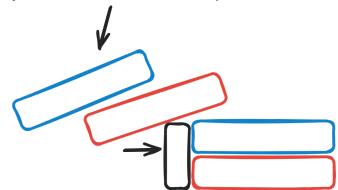
Goal: Test higher scoring autonomous routines for elimination matches.

Corner Testing

To get the rings out of the corner stack (shown right) we tried to use the sweeper to get the rings out of the corner, as shown on the right. In our testing, we tried a variety of different speeds and other variables, but the top two rings would always fall down behind the sweeper, meaning we couldn't intake them. This led us to scoring at most one of our alliance rings, which we do not think will be worth spending the extra time to score at this event.



Rings fall off the top of the stack



Negative Side Testing

For the negative side autonomous the testing went exactly to plan. The path followed the desired exactly and no changes were needed beyond removing the corners.

Positive Side Testing

The positive side was the same way, it worked without any changes to the initial path. Besides removing the corner, we added the option for touching the climb structure at the end to allow us to run this with an alliance that has a negative side AWP at competition. This behavior worked exactly as expected, allowing us flexibility with our alliance partner.

Conclusion

- Corner scoring
 - We were only able to get the bottom 2 rings
 - Scoring one ring wasn't worth the time it would take to tune at this competition.
- Negative side
 - Testing went exactly as planned
- Positive side
 - Added pole touch giving us flexibility with our alliance
 - Rest of testing went exactly as planned

Corner testing		
Trial #:	# Rings	Stack Fall
1	1	Yes
2	0	Yes
3	1	Yes
4	1	Yes
5	1	Yes
6	1	Yes
7	0	Yes
8	1	Yes
9	1	Yes
10	1	Yes

Auton Testing		
Trial #:	Positive	Negative
1	6	7
2	8	9
3	7	10
4	7	10
5	8	9
6	8	10

10/23/24 Testing: Erie Pre-Competition Skills Testing

Designed by: Alex

Witnessed by: Carl

Witnessed on: 10/23/24

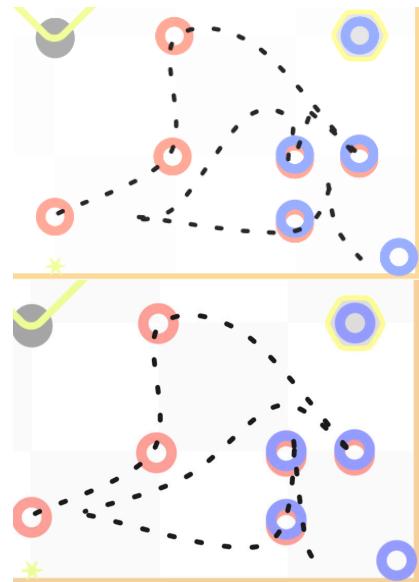
Goal: Refine the skills path for the Erie competition.

Accountability Notice: Score data inputted late (10/27/24)

In our [Butter Nexus Skills Analysis](#) (Pg. 208), we found multiple issues with our current skills path, with multiple specific ways that we could improve it. In this entry we will implement and test these changes to prepare for the Erie High School Competition.

End Path Changes

We had issues at the league event with blue rings getting in the way while we were trying to do the final motion in skills (motion shown top right). This motion would often cause the blue ring to end up as the top ring on the final goal, losing us 3 points. To fix this we made the robot move around the first ring and then intake both by going in a sweeping motion that didn't have the same issue with rings getting in the way because the intake hit the to the side. Another benefit that we found for this path is that it cuts off about a half second, giving us more time to hang and for intaking in other places.



Time Allocation

Another thing that we noticed in the league was that some rings didn't have enough time to score on the mobile stakes, losing us precious points that we should have gotten. To remedy this, we worked on time optimizations for the rest of the path, allowing us to intake rings at a slower pace, giving them more time to move through the intake, massively improving the consistency of scored rings.

Conclusion/Results

In our testing today we were able to bring our score up from around 50 to consistently scoring in the 55-59 range, with a top score of 59, which would be the highest programming skills score in the world.

- End path changes
 - Moved the path so blue rings weren't in the way of the red rings that we were trying to score
 - Reduced time by half a second
- Time allocation
 - Rings that we intake now have the chance to score on the mobile stakes
 - Strategically slowed down the robot where rings missed

Skills Scores		
Trial #:	Driver	Autonomous
1	49	54
2	50	58
3	53	59
4	55	56
5	46	53
6	54	59
7	N/A	54
8	N/A	58
9	N/A	59
10	N/A	59

10/30/24 Testing/Evaluate: Erie Competition Analysis

Designed by: Alex, Matt, Carl	Witnessed by: N/A	Witnessed on: N/A
-------------------------------	-------------------	-------------------

Goal: Document our tournament results and analyze matches and skills.

Matches

In qualification matches, we went 6-0-0 and got several win-points, allowing us to achieve 1st rank. In the elimination matches, we won our round of 16, but ended up losing in the quarter finals match.

Match	Final Score	Win Point	Auton Win	Notes
Q9	41-8	Yes ▾	Yes ▾	Alliance protected positive corner, we scored wall stakes and mobile stake
Q14	38-15	No ▾	Yes ▾	Missed alliance stake for AWP. Sorted 1 ring wrong, alliance lost corner, we gained corner back and scored wall stakes
Q19	38-5	Yes ▾	Yes ▾	Alliance protected positive corner, while we scored wall stakes
Q31	43-9	Yes ▾	Yes ▾	Got both positive corners and top ring on both wall stakes
Q38	36-16	No ▾	Yes ▾	Mobile stake grab messed up for AWP, alliance protected positive corner, we scored wall stakes and mobile stake
Q49	31-26	Yes ▾	Yes ▾	Alliance protected mobile stake, we played defence and scored wall stakes
R16 1-1	48-11	N/A ▾	Yes ▾	Got both positive corners and top ring on both wall stakes
QF 1-1	29-38	N/A ▾	No ▾	Missed alliance stake in auto, alliance got full goal in positive, we contested the other positive and lost it. Gave goal to partner to protect. Went to score on wall stakes with 20 seconds left and lost top ring in the last 2 seconds

Matches Summary:

Strengths:

- Collaboration with alliance partners - Talked to our alliance partners laying out exactly what each of us would do.
- Filling up goals quickly - We were able to efficiently fill up mobile goals in time to get to the positive corner.
- AWP - We were able to get 4/6 AWPs, ranking us much higher than other teams.

Weaknesses:

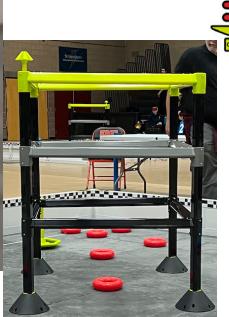
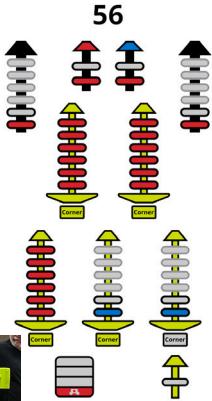
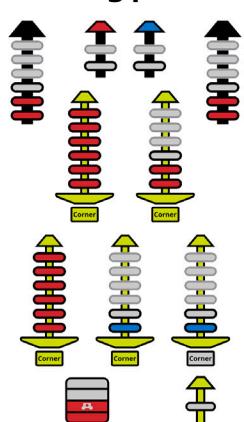
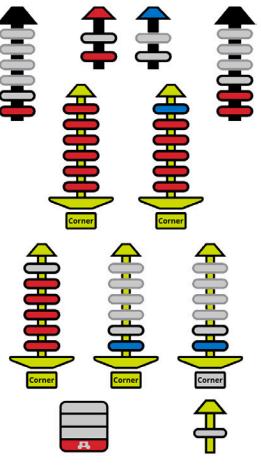
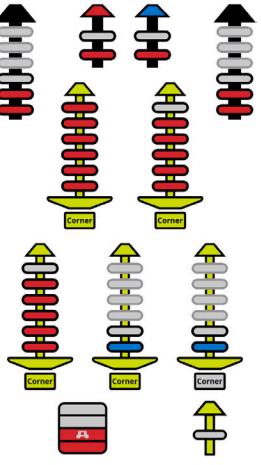
- Mobile goal strategy - We looked over the importance of top rings, so sometimes the third goal was left untouched.
- Decision making - We needed better decision making in some matches especially in our quarter-finals match.

Matches Conclusion

Overall we did very well in matches going undefeated in qualifications. In order to win more tournaments we need to refine our strategy to ensure we have more control of points and top rings.

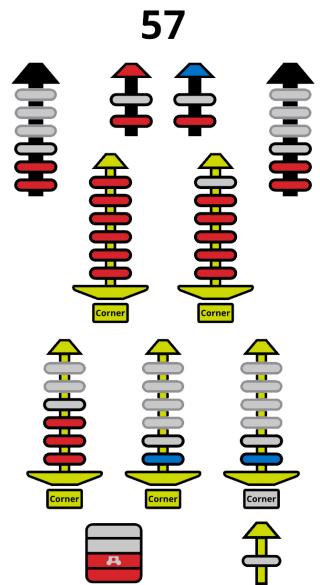
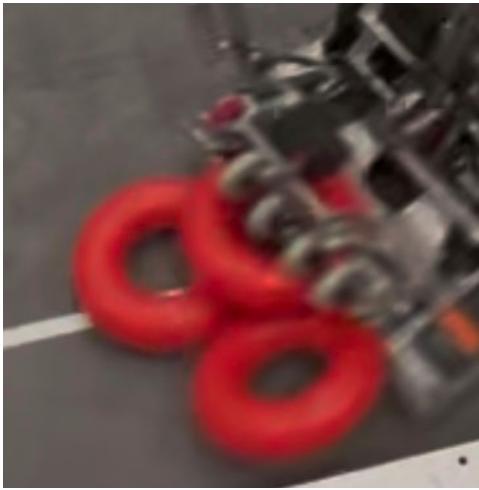
Skills

In skills, we were able to achieve a high score of 59 in programming and 57 in driving, putting us 1st in world skills. Here is a more detailed analysis of our runs:

Drive 1	Prog 1
<ul style="list-style-type: none"> Very high-scoring Missed the hang <ul style="list-style-type: none"> Due to field variances (0.75 inches) Ring fell off mech on first wall stake   <p style="text-align: center;">56</p> 	<ul style="list-style-type: none"> Lower scoring Missed both alliance stakes <ul style="list-style-type: none"> Aligned well, however still missed short  <p style="text-align: center;">51</p>  <ul style="list-style-type: none"> Missed 3 rings at the end of the run <ul style="list-style-type: none"> Ring jammed in the bottom of the intake causing the wheels to slip, reducing localization accuracy for a prolonged period of time
<p style="text-align: center;">Drive 2</p> <ul style="list-style-type: none"> Lower scoring run Missed hang <ul style="list-style-type: none"> Got stuck (Too high speed) Missed far alliance ring <ul style="list-style-type: none"> Misaligned with alliance stake Missed ring on first wall stake <ul style="list-style-type: none"> Misejection Missorted blue ring   <p style="text-align: center;">52</p> 	<p style="text-align: center;">Prog 2</p> <ul style="list-style-type: none"> World Record One ring missed on the second to last goal Only 1 point off our theoretical maximum  <p style="text-align: center;">59</p> 

Drive 3

- **Highest Driver**
- Only issue was 3 rings jammed



Skills Conclusion

Overall these scores and consistent runs allowed us to get a world record by 3 points.

Rank	Score	Autonomous Coding Skills	Driver Skills	Highest Autonomous Score Stop Time	Highest Driver Score Stop Time	Team Number	Team Name
				Time	Time		
1	116	59	57	0	0	2654E	Echo
2	113	54	59	0	0	16099A	Overclock
3	109	53	56	0	1	81988E	Extropy

11/04/24 Testing/Evaluate: CEC Competition Analysis

Designed by: Alex, Matt, Carl	Witnessed by: N/A	Witnessed on: N/A
-------------------------------	-------------------	-------------------

Goal: Document our tournament results and analyze matches and skills.

Matches

In qualifications we went 6-1-0 and got 1 win-point placing us 3rd in rankings. In elimination matches, we dominated our matches until finals where we won by a small margin but still pulled ahead.

Match	Final Score	Win Point	Auton Win	Notes
Q5	37-3	No	Yes	Got top rings on most goals, teammate defended positive corner
Q13	27-36	No	Yes	Got the positive corner, got hold of 3 mobile goal, scored on wall stakes, lost mobile stake to negative corner after we disconnected
Q17	41-19	Yes	Yes	Incorrectly sorted a ring, 3 goals with rings, lost a goal that the opposing team put in a negative corner after we were trying to flip it over
Q25	21-38	No	No	Missed alliance stake in autonomous, disconnected with 15 seconds remaining, resulting in the opposing alliance, putting our goal in the negative corner
Q38	33-10	No	Yes	Standard strategy, we're able to score opponents positive corner around 30 seconds remaining
Q47	19-12	No	Yes	Lost positive corner, we were able to get back with wall stakes and mobile stakes.
QF	45-9	N/A	Yes	Both positives and auto win, wall stakes in last 15
SF	32-12	N/A	Yes	Got both of the positive corners and won auto, wall stakes in last 15.
F	39-26	N/A	Yes	Got a positive corner early, got hold of 3 mobile goals and scored on wall stakes.

Matches Summary

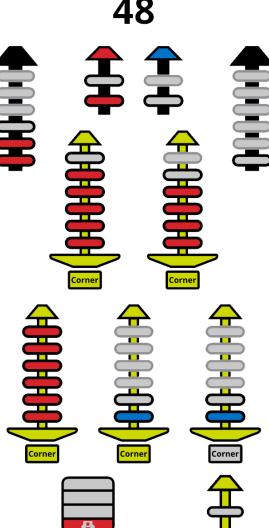
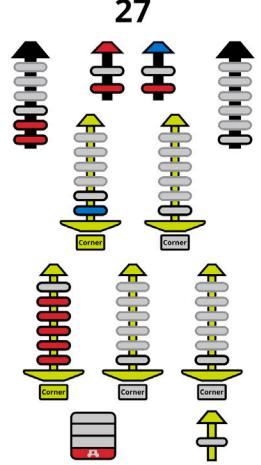
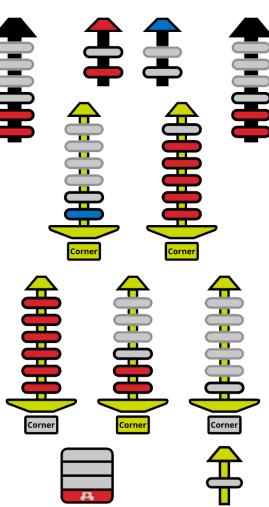
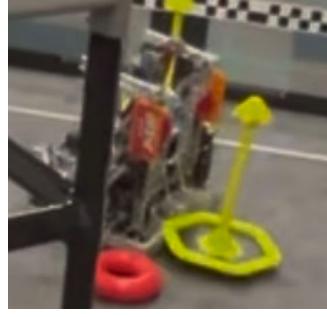
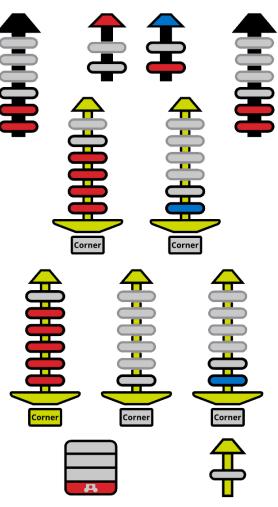
Strengths:

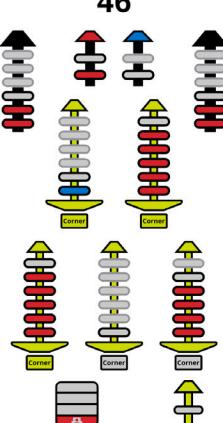
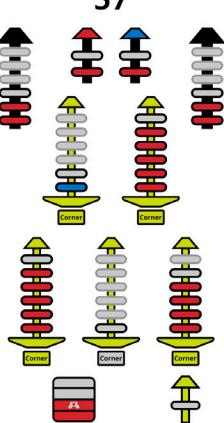
- Mobile Goal Control - In almost every match we were able to obtain 3 mobile goals giving the most top stakes.
- Auto sort - The auto-sort works very well only messing up once during the competition.
- Positive corner control - In every match we were able to get at least 1 positive corner and in eliminations sometimes 2.

Weaknesses:

- Mobile goal flip - We failed to flip a goal in one of the matches and lost it to the negative corner.
- Alliance stake missing - In auton and in matches the alliance stake was very inconsistent and would often miss.
- Top rings on wall stakes - In some matches especially final we were unable to get the top ring on all wall stakes.

Skills

Drive 1	Prog 1
<ul style="list-style-type: none"> Very low scoring Missed 2nd mobile stake grab Driver had to override All movements had severe inaccuracies  <p style="text-align: center;">48</p> 	<ul style="list-style-type: none"> Very low scoring as well This one also missed the mobile stake in the same way Intake jammed All movements had severe inaccuracies  <p style="text-align: center;">27</p> 
<p style="text-align: center;">Drive 2</p> <ul style="list-style-type: none"> Low scoring Lost localization <ul style="list-style-type: none"> Wheels slipped when getting the second alliance stake Messed up encoder readings Led to the driver having to override <p style="text-align: center;">45</p> 	<p style="text-align: center;">Prog 2</p> <ul style="list-style-type: none"> Very low scoring Missed the same goal grab All movements had severe inaccuracies  <p style="text-align: center;">29</p> 

Drive 3	Prog 3
<ul style="list-style-type: none"> • Low scoring • Intake jammed causing driver to have to override • Rest of skills went well 	<p style="text-align: center;">46</p>  <p style="text-align: center;">57</p>  <ul style="list-style-type: none"> • Increased feedforward by 3% • High score • The route missed 3 rings around the field intaking onto the goal • Overall very good run • Severe inconsistencies were no longer present

Overall these scores only got us 106 which is not higher than our previous best. This was very likely because the friction our drive experienced while it was driving was much higher than it was on different fields, which changed how the path following algorithms performed significantly leading to very noticeable differences in the results. We believe that our robot was a little closer to the tiles because the field for this tournament's skills was on a carpet. To fix this we found that increasing the [feedforward](#) (Pg. 65-66) multiplier to account for the extra friction on the tiles allowed us to reduce the inconsistencies on the tiles very greatly, giving us a good run for the one time where we had this change.

Conclusion

- Skills didn't score as high
 - Carpet under the field was the primary reason
 - Increased drag
 - After increasing feedforward the path worked as intended
 - Future steps
 - Current robot
 - Make a "carpet" mode where the robot has slightly higher feedforward constants
 - Next robot
 - Make sure no parts drag regardless of the material under the field
- Qualifications went very well
 - Only lost 1 match
 - Unpreventable loss due to disconnect during the last 15 seconds
 - Refined strategy and autons for eliminations
 - Scored ring on 3 mobile goals in auto
 - Protected the 3 mobile goals for the rest of the match
 - Protected at least 1 positive corner

11/04/24 Time Management: Robot Rebuild

Designed by: Alex

Witnessed by: Carl

Witnessed on: 11/04/24

Goal: Plan out time management for the transition from robot #1 to robot #2

To maintain our goals of being competitive in skills we would like to move on to our next robot design cycle as soon as possible to give us ample time to test. However, after the Berthoud competition was canceled we found that a league night on the 11/6 was the biggest time limitation to us rebuilding our robot before league finals. For this transition time we have the following main tasks that are dependent on one another:

Robot Transition Tasks

- Data Collection - **Must be done before deconstruction**
 - Collect data from Monte Carlo Localization and RAMSETE for analysis of the current robot
- Finalize CAD
 - Finish all parts of the CAD necessary to start building the robot
- Deconstruction
 - Deconstruct and analyze robot #1
- Data Analysis
 - Perform a detailed analysis of the data collected before the deconstruction
- Programming
 - Rewrite the programming for the new robot
- Build
 - Build the new robot
- Test/Improve
 - Refine the robot during on field testing

Gantt Chart

Task	Week 1					Week 2					Week 3					Week 4					Week 5								
	11/4	11/5	11/6	11/7	11/8	11/9	11/10	11/11	11/12	11/13	11/14	11/15	11/16	11/17	11/18	11/19	11/20	11/21	11/22	11/23	11/24	11/25	11/26	11/27	11/28	11/29	11/30	12/1	12/2
Carl Out of Town																													
Data Collection																													
Deconstruction																													
Finish CAD																													
Build																													
Test/ Improve																													
Program																													
Competition																													
Meeting																													

11/05/24 Identify/Define/Brainstorm/Design: Localization Testing Tool

Designed by: Alex

Witnessed by: Carl

Witnessed on: 11/05/24

Goal: Make an app to testing localization easier

Identify

Currently, the best tool we have to test localization is our [path planner](#) (Pg. 177-181) live, however, this is not very well suited to working with multiple recordings or replays of paths. It's for this reason that we decided that it was necessary to make a separate tool just for working on debugging the path planner.

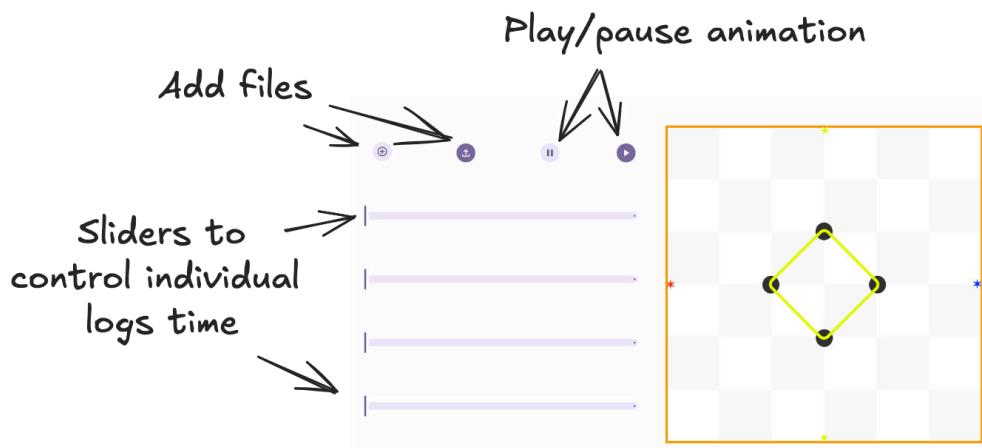
Define

This localization testing tool must:

- Be able to replay recorded localization logs
- Easily go back/forth
- Track predictions over time
- View multiple runs at once

Brainstorming

Currently, we have our path planner however, this is not built to be able to support what we have defined, so we would have to heavily modify what we have currently. This is why we think it would be beneficial to create a new app for the flexibility we would like for this project. To start we drew out a sample of the GUI that we wanted in the [figma UI design app](#):



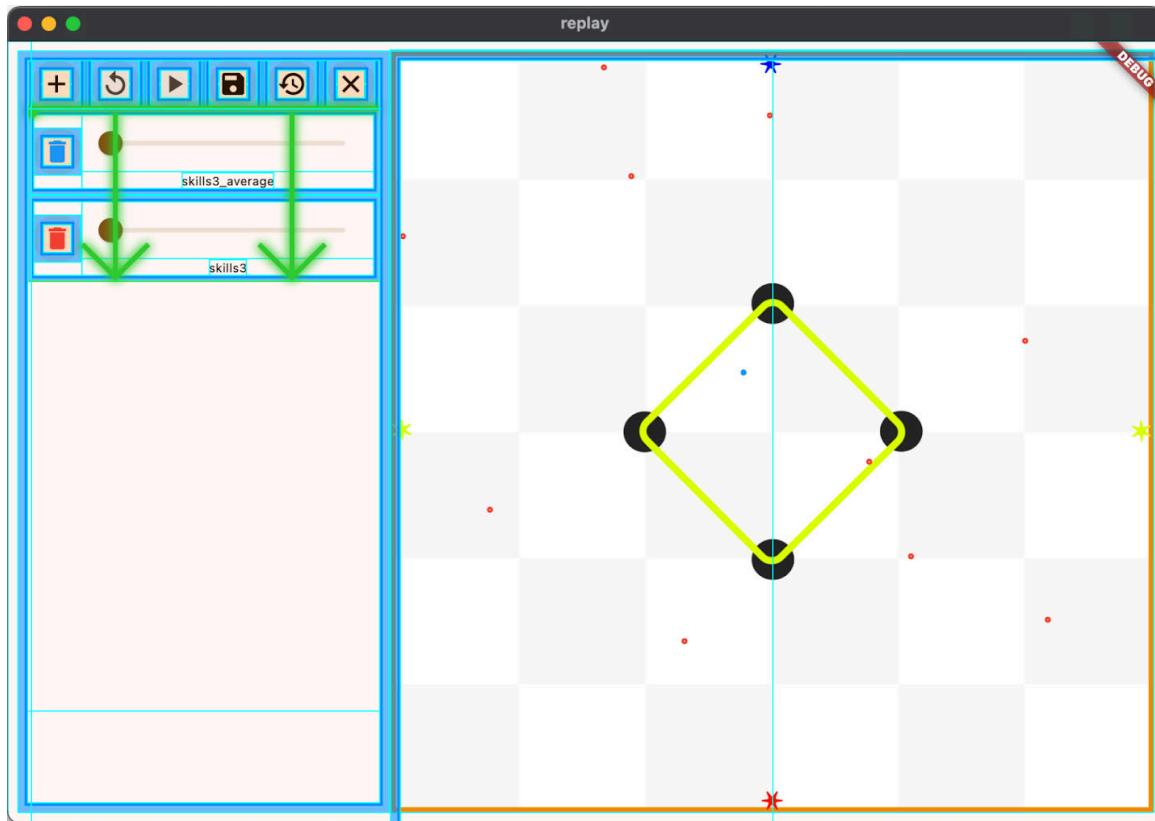
Design

To build the app we used the same GUI framework as our path planner, [Flutter](#), because of the flexibility it offers and our experience with it. To make it we had 4 main components to implement: layout, json interpreter, custom painter, and pause/play system. First we started with implementing the planned layout in flutter:

Layout

To get the layout for the app we used a Scaffold flutter layout object for the overall look of the screen. We then split up the desired layout from before into the tree structure necessary for flutter to interpret it:

- Scaffold
 - Row (Splits screen horizontally)
 - Column (For tools on the left)
 - Row (For buttons)
 - IconButton.filledTonal for play, replay, save position, restore position, and delete paths
 - ListViewBuilder (For adding each replay item to list)
 - IconButton.filledTonal (Trash button for removing from screen)
 - Slider (Change time in the path to replay)
 - Image
 - CustomPainter (Drawing particles on the field)



Json reader

We chose to store the position data in a json because we have the most experience with this data format and it is well supported in C++, python, and dart, which are the 3 primary languages that we use. We will use a data format like the following to store our position data coming from the robot, and we will discuss how we read this data and plot it on the field.

```
{
  "data": [
    {
      "time": <time in seconds>,
      "points": [
        {
          "x": <x position in meters>,
          "y": <y position in meters>,
          "t": <angle theta in radians>
        },
        ... // more data points at this timestamp
      ]
    }
    ... // more timestamps
  ]
}
```

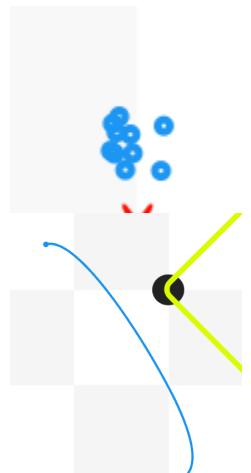
To effectively break down this structure we will use a somewhat recursive pattern such as the following:

- data
 - list
 - time
 - points
 - list
 - X
 - Y
 - T

We will use a Position class to store the X, Y, and T values for the 2d position of the robot and do the necessary functions to draw it. We then made a PositionTimestamp that stores a list of positions and the timestamp that they were collected at. Finally we store all of this in a PositionList class that we also added a member function to that allows us to get the most recent data at a specified timestamp.

CustomPainter

To draw the paths on the screen we took advantage of the Flutter CustomPainter object which allows us to draw over an object every frame. We will draw circles for each of the particles, with the index of the log changing the color allowing for easy differentiation. Using the decomposition of the json from above to get the data for the points.



For the paths we will only draw a line for paths with 1 point at each timestep. We will keep track of the past positions of this particle at each redraw and store them for each log. This allows us to draw paths of the robot's movements like this.

Play Button/Time Management

To be able to see how paths move over time we will use a play pause system to change the time on the paths. We will store the paused/play state of the buttons in a boolean and then run a routine every frame to update the current time on the slider of each path and the positions for display on the field. This allows us to animate the robot's movement (which we unfortunately cannot put in the notebook, but is on our youtube here <https://youtu.be/bH00zEN0BQI>).



Quality of life changes

- Changing the colors of the trash cans
 - This allows us to quickly identify which path is which when changing stuff
- Names
 - Under the slider also allow us to quickly identify which file is open
- X button
 - Clears paths currently drawn on the field

Conclusion

Putting this all together, we were able to create a replay tool that we can use to have better visibility into what is going right and wrong with particle filtering and path following. This tool, although we have many in our toolbox for debugging software issues, is a very important way to provide visibility into the robot's internal belief of state.

11/06/24 Testing: Localization Performance/Tooling

Testing

Designed by: Alex	Witnessed by: Carl	Witnessed on: 11/08/24
-------------------	--------------------	------------------------

Goal: Finalize localization testing.

Justification

In the paths planner occasionally we need to move the paths 2-3 inches off the actual position on the field which we suspected to be reliant on the particle filter localization of the robot leading to the robot not being able to move to the field objects. Additionally, for the initial tuning of the filter we chose parameters based on numerical calculations and reasonable assumptions. However, we have reason to assume that these calculations and assumptions are inaccurate.

Testing Methodology

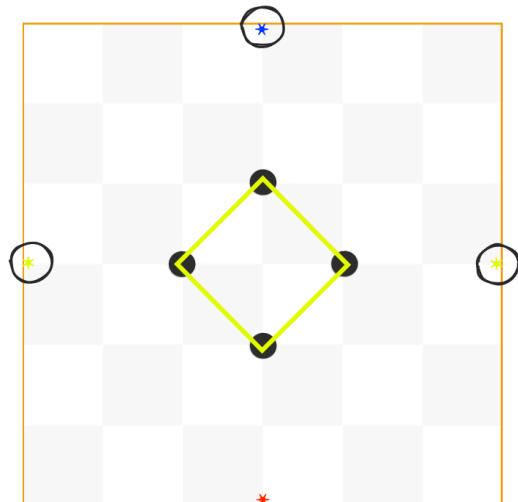
Before designing and testing a ground truth localization algorithm we wanted to solidly determine that the cause of inaccuracies in skills was the localization like we expected. To do this we would run skills as the path currently stands and analyze the accuracy around key points in the field such as the alliance and wall stakes to determine if localization is notably inaccurate around these points. After that we will analyze the paths over time and see if there are any unexpected inconsistencies in the path. We will use the new replay tool to accomplish this.

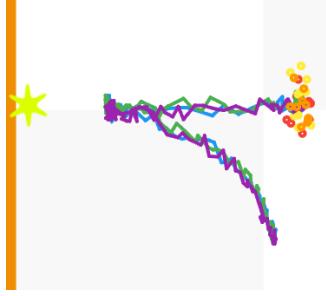
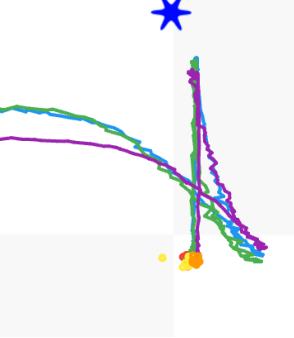
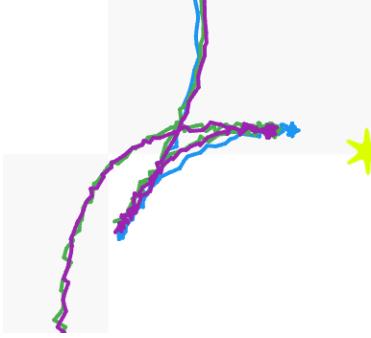
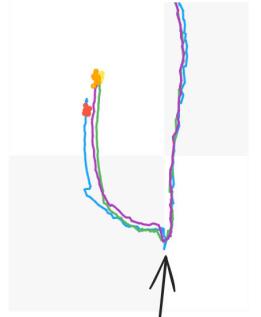
Method

1. Run skills
2. Record localization data from the runs
3. Use our new [replay tool](#) (Pg. 226) to analyze the runs
4. Return step 1

Results

Overall, the conclusion of our testing was that our current localization methods are still extremely accurate without the need for ground truth testing right now, however there are a few parameters we could tune for better performance. The skills we ran to practice at all the key points (illustrated to the right) predicted the robot's position almost exactly.



Key Point #1: Wall Stake	Key Point #2: Alliance Stake
<p>For the first wall stake we nailed the position every single time, and that is shown in all 4 of the logs as it is directly in front of the wall stake.</p> 	<p>For the alliance stake the robot was slightly off each time to the -x direction, which is reflected in the logs.</p> 
Key Point #3: Wall Stake (Right)	Inconsistency #1
<p>For the last wall stake the robot was consistently off to the -y direction.</p> 	<p>All of these points proved that our approach for localization is very consistent and accurate, however, this point showed inconsistencies in tracking. At this point the robot was unable to see the close wall, while the robot predicted that it traveled farther than it actually did, leading to this large correction once it was able to get distance sensor readings on the walls. We can easily fix this by getting a more accurate reading of the diameter of the wheels.</p>  <p>Overestimated travel</p>

Conclusion

- Unfortunately, we found ground truth localization methods much harder to implement than originally intended
 - Wanted to pin down root of the problem before spending too much time debugging localization
- We thought there was an inconsistency in the localization estimation of our robots position
- Tested that hypothesis by seeing the position estimate relative to various key points on the field
 - Concluded that localization isn't the leading cause of these inconsistencies
- Next steps
 - Tune RAMSETE parameters for higher tracking accuracy

11/07/24 Design/Testing: Faster 2D Motion Profiling

Designed by: Alex

Witnessed by: Carl

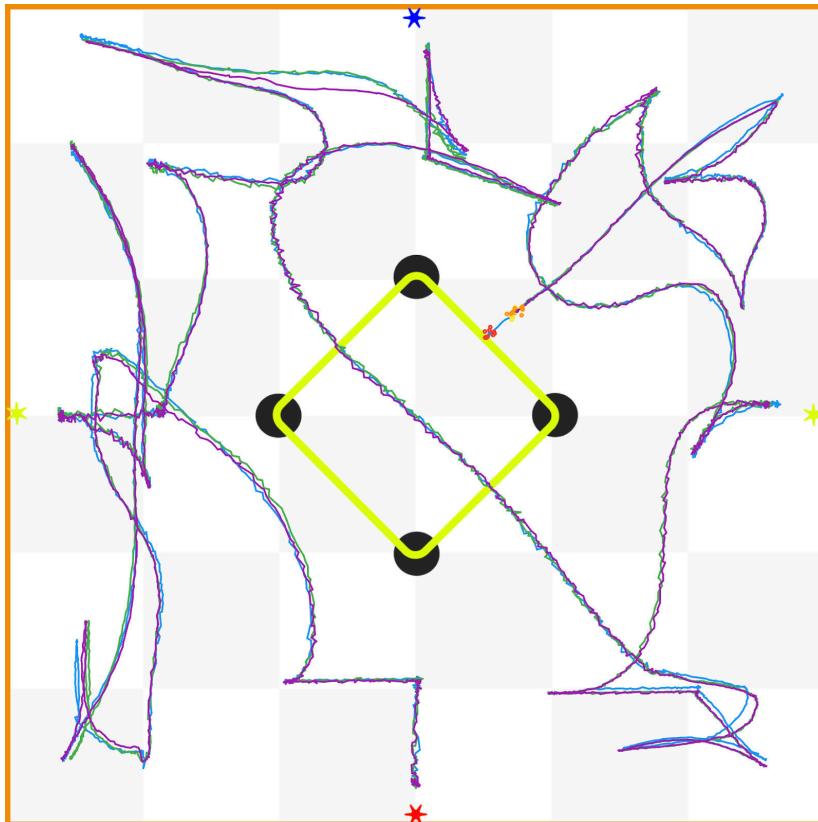
Witnessed on: 11/08/24

Goal: Tune 2D motion profiling feedforward and the tunable parameters in RAMSETE

Overall, our current methods and tuning parameters for RAMSETE are pretty well suited for creating a *consistent* path, as shown below, but we found while testing the [localization performance](#) (Pg. 230-231) that the source for inconsistencies between the desired path and the actual path followed (Referred to by the *accuracy* for the remainder of this entry) was very large and if we want to create paths for skills faster then increasing the *accuracy* of the path following is of utmost importance.

Method

1. Utilize wireless debugger V1 to record data (desired and predicted location from localization) from the robot while completing a skills path
2. Analyze run to find inconsistencies
3. Return to step 1 until no major inconsistencies are noted



3 separate skills run's localization data overlapped

Initial Testing

To start testing we will record the inputs and outputs for RAMSETE: desired position of the robot from [2D motion profiling](#) (Pg. 187-190) and the predicted location of the robot. We will use this data to analyze how we can change the [RAMSETE](#) (Pg. 191-192) path following algorithm's tunable parameters listed in the testing section to most effectively cause the robot to follow the path with minimal error. Here is the result of running the path represented by the red and blue lines:

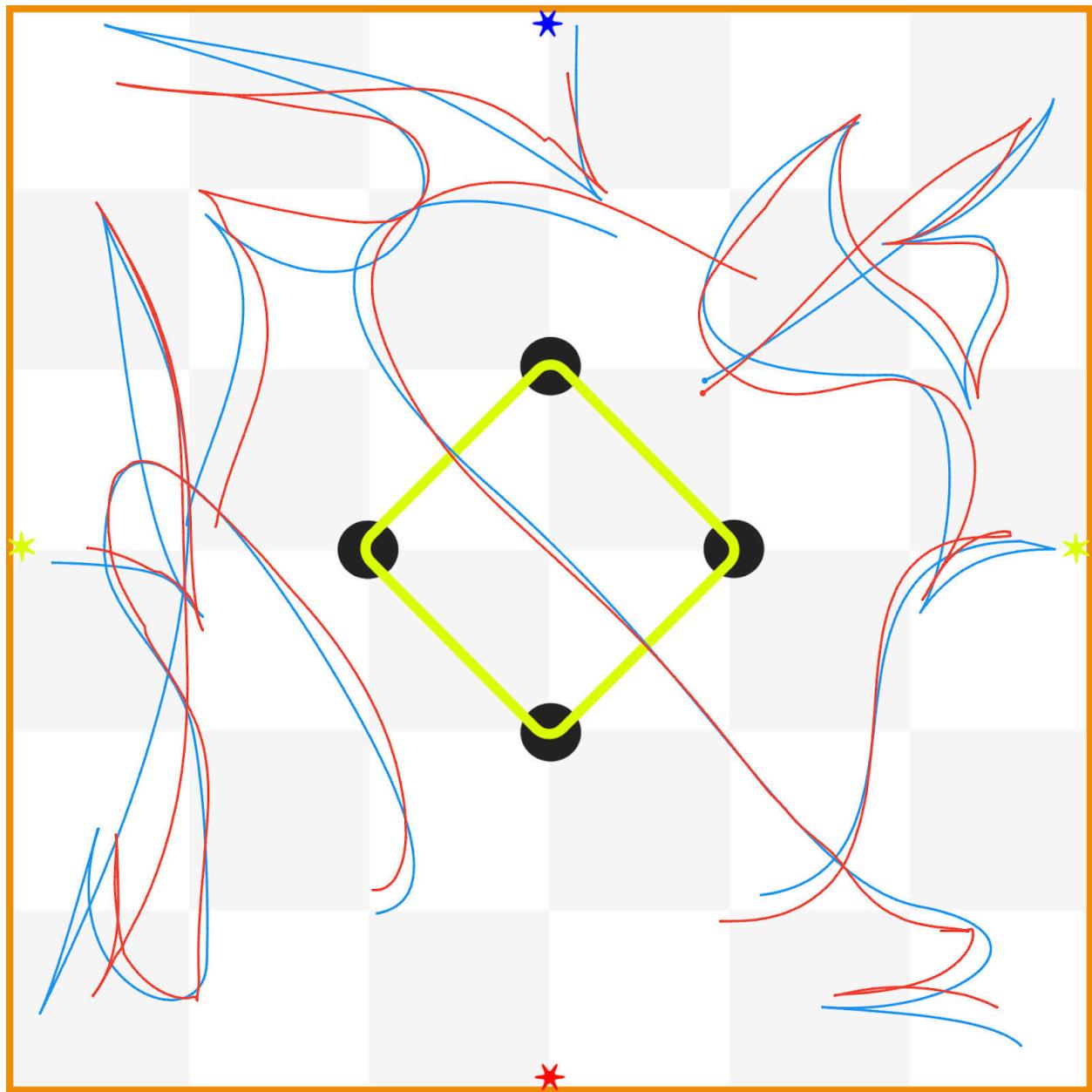
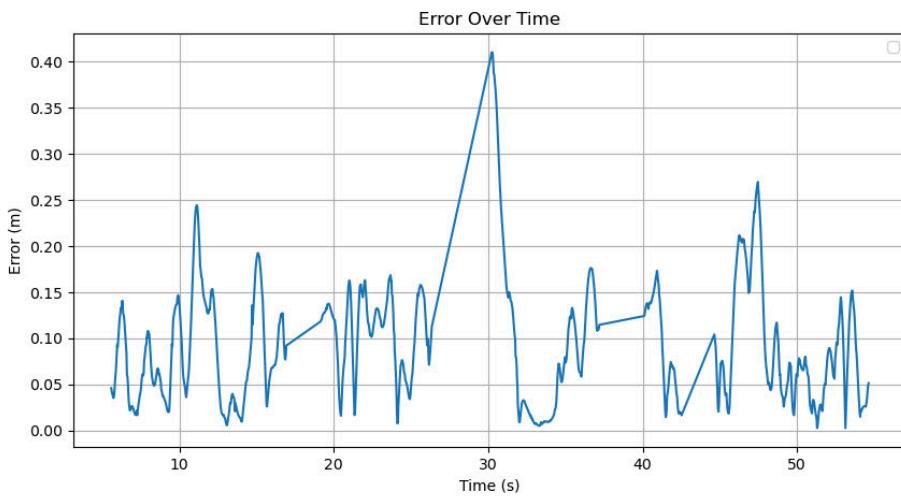


Figure: Desired (blue), actual (red) (Only 2D motion profiled movements)

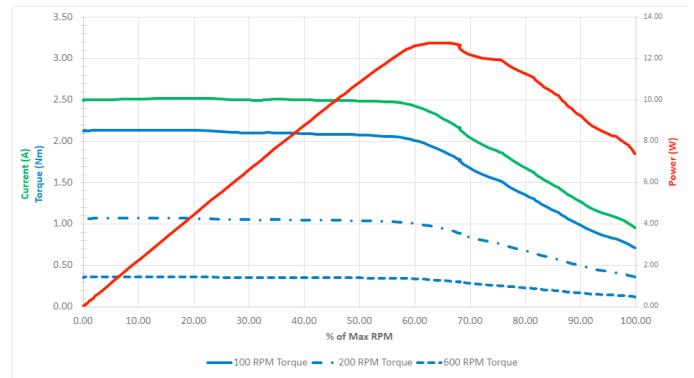
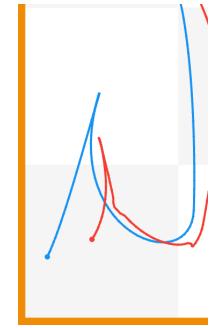
Analysis #1



Average Error: 0.09m

While overall these runs are consistent, the path following is very inaccurate. We have a couple theories for this difference, listed here:

- Infeasible paths
 - Ex: Path on the right
 - The robot can't go that close to the wall so we should never expect that tracking accuracy
- Poor feed forward constants
 - The robot consistently lagged behind the desired position from 2d motion profiling
 - The robot should always start the path with the correct acceleration and the correct speed
 - Inaccurate acceleration constraints
 - Currently we assume the robot has constant acceleration over the entirety of its speed curve, which assumes constant torque which is inaccurate to the actual motors (Torque in blue)
 - We could account for how the robot can accelerate given the torque of the drive motors, which could give us better limitations on the robots movement
 - Poorly tuned feed forward constants



Source: VEX Robotics Knowledge Base

- Currently our approach for tuning feedforward constraints relies on perfect testing situations and a long manual trial and error process
- It is *possible* to calculate better feedforward constraints automatically given recorded data of the robots movement
- Possibly we could even do this during the run to stop citations like what happened at the [CEC tournament](#) (Pg. 222-224) mid-run to adjust the feedforward constants on different underlying material
 - Extra note: while holding a goal our robot will have a higher moment of inertia therefore, it will be harder for the robot to turn
- *b* parameter is too low
 - In this run the correction from the RAMSETE controller appeared to be very small
 - To increase this tracking correction we increase the *b* parameter
 - This might increase the amount of oscillations we have and therefore require us to increase the *z* value as well which might not have the desired effects
- Slower movements
 - Currently, our desaturation algorithm ensures that under normal movement both sides of the drive will be under the max speed of this path
 - This loses us a lot of potential speed where we could have the motors desaturated to their actual max speed limit
 - This change would allow the robot to move and maintain as much speed as possible in tight turns
 - This could cause issues with the tracking error of the robot because the robot can slightly drift on high speed turns

Changes for the Next Run

- 2D motion profiling changes
 - Change desaturation algorithm to work with robot max linear speed instead of path max linear speed
 - Code changes:

bezierMotionProfiling.h (changes highlighted)

```
void calculate(const QVelocity startSpeed, const QVelocity endSpeed) {
    std::vector<QLength> distance{0.0};
    std::vector<QVelocity> velocity{std::min(limitSpeed(beziers[0].getCurvature(0.0)) *
CONFIG::MAX_SPEED, startSpeed)};
    std::vector<QAceleration> accel{0.0};

    QLength accumulatedDistance = 0.0;

    this->t.emplace_back(0.0);
```

```
for (size_t i = 0; i < beziers.size(); i++) {
    auto length = beziers[i].getDistance();
    const auto count = std::max(5, static_cast<int>(std::ceil((length / 2_in).getValue())));

    for (size_t j = 1; j <= count; j++) {
        const auto t = static_cast<double>(j)/static_cast<double>(count);

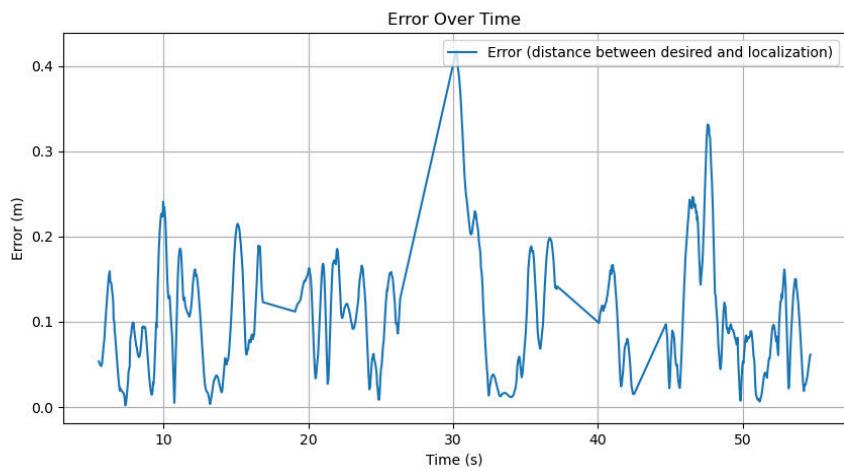
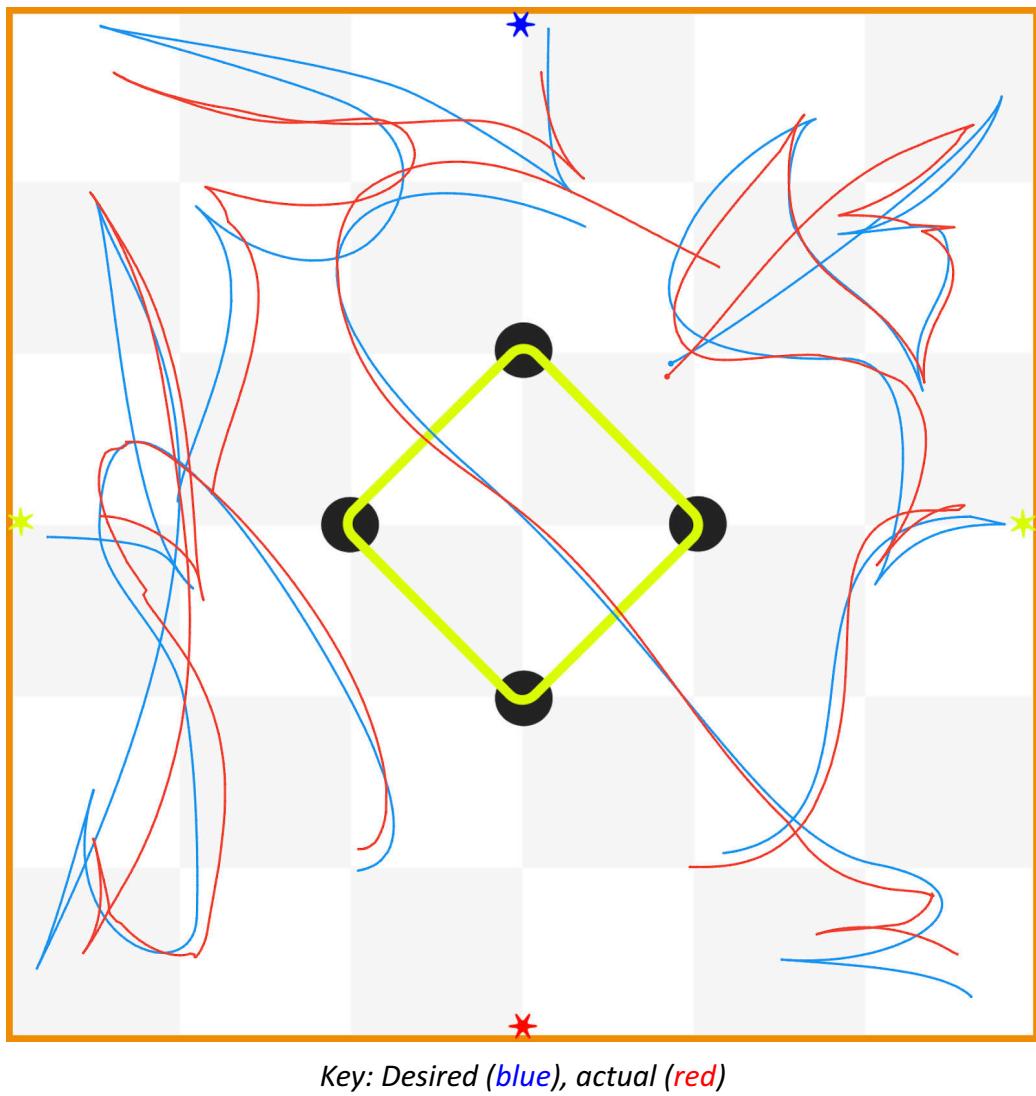
        this->t.emplace_back(static_cast<double>(i) + t);

        const auto curvature = beziers[i].getCurvature(t);

        distance.emplace_back(accumulatedDistance + beziers[i].getDistanceAtT(t));
        velocity.emplace_back(std::min(limitSpeed(curvature) * CONFIG::MAX_SPEED,
beziers[i].velocity));
        accel.emplace_back(beziers[i].acceleration);
    }

    accumulatedDistance += beziers[i].getDistance();
}
...
...
```

Testing #2



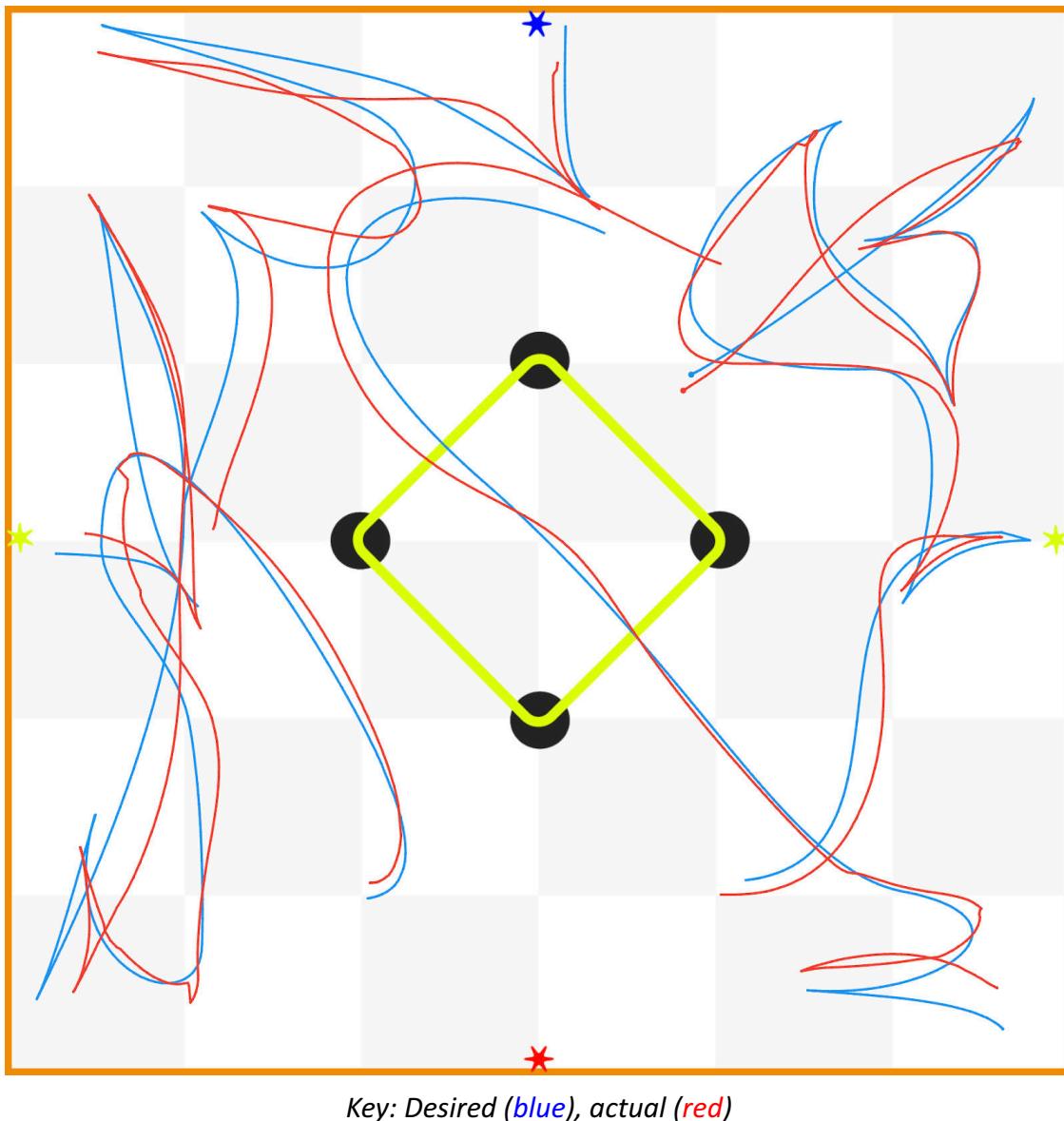
Average Error: 0.14m

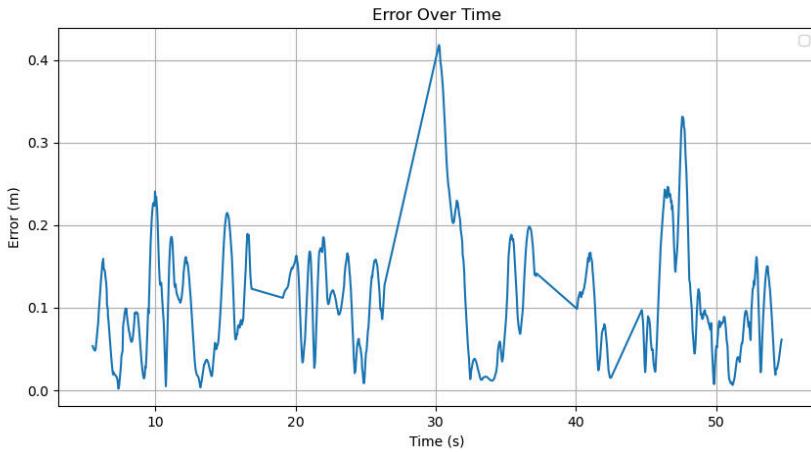
This test was very successful because our one change, modifying robot speed desaturation, while making the robot faster, (10% speed increase) didn't significantly increase error. We think that the error is likely high for all the same issues with feedforward constants and parameters, but we solved the path being too fast while keeping error relatively small.

Changes

For this run we only changed the b parameter in RAMSETE from 12 to 18 and the z parameter up to 0.9 to allow the robot to correct better for error.

Testing #3





Average Error: 0.11m

Analysis

In this test we were able to reduce the error in the path by just changing the RAMSETE tuning variables. For our next steps we need to develop a mathematical derivation to calculate the feedforward constants and derivations for better acceleration control which doesn't fit into this entry.

Summary

- Tested skills by graphing the current and desired positions from RAMSETE
 - Found a method to make the desaturation of robot controls allow the robot to move 10% faster
 - Resulted in a higher error because of faster movement
- Analyzed the effect of RAMSETE tuning variables
 - Tuned better values for b and z to reduce the tracking error of the robot with the increased speed
 - Lower tracking error before changes with slower path

11/08/24 Evaluate: Robot 1 Deconstruction (R.1.2.11)

Designed by: Carl

Witnessed by: Alex

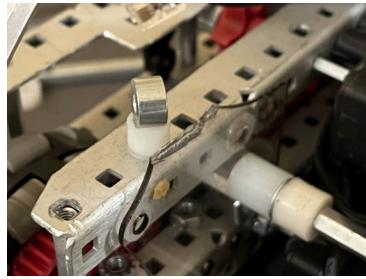
Witnessed on: 11/10/24

Goal: Deconstruct our first robot and identify any issues with the build.

Because our school has five teams competing this year, we do not have enough parts (sensors, gears, pneumatics, and aluminum structure) to have two robots at a time. The practical benefit of taking the first robot apart is that we can gain insight into the construction of the robot, which will allow us to strengthen weak areas in the rebuild. Additionally, we can reuse a lot of custom parts such as drilled HS shafts and shaved down gears to help accelerate the rebuild.

Weak Points:

- Acetal intake ramp
 - Very brittle material
 - Lots of force bending it into shape
- Intake pivot polycarbonate
 - High impact area
 - Not properly supported
- Worn-out bearing flat on top intake
 - High force, high speed
- Slight bend in back clamp pistons
 - Protect pistons on new robot



Strong Points/Good Features:

- Most aluminum structure remained square, rigid, and virtually undamaged
- Polycarbonate within the robot in mint condition
- Polycarbonate on the outside of the robot had noticeable damage however no parts were actually broken
 - This polycarbonate successfully protected the drivetrain, wiring, and electronics from damage

Summary and Takeaways:

Overall, we are extremely satisfied with the condition of our robot after two league nights, two tournaments, and 40+ hours of testing. Apart from very minor and mostly cosmetic damage, the robot was in nearly immaculate condition. We believe that employing similar building techniques on the new robot will have a good result and by implementing polycarbonate protective panels on external surfaces we can achieve an extremely resilient design.

(Note: Analysis of individual subsystems can be found in ROBOT 2)

ROBOT 2

09/30/24 Strategy: Skills Reanalysis

Designed by: Alex and Carl

Witnessed by: Matt

Witnessed on: 10/01/24

Goal: Analyze how we can feasibly score more points in skills.

Concerns With Current Robot

Based on our [Skills re-path testing](#) (Pg. 204), we think that it would be worth reevaluating our approach for scoring above 61 (Maximum score without high-hang, descoring, or using blue rings) in skills. Scoring blues has been very difficult for us to do reliably, and with that being such a large concern, we would like to reanalyze our approach to the skills challenge.

Options For Higher Score

- Blue rings
 - Worth up to about 3-4 points
 - There is already almost too much of a time crunch to score them at all
 - We believe that hanging and scoring blue rings are mutually exclusive
- De-scoring goals
 - This also takes a ton of time (8 sec total)
 - Worth up to 6 points
 - This would be a struggle to fit in a run, but also we think it would be barely possible
 - Still mutually exclusive
- High Hang
 - Also takes a ton of time (up to 10 seconds)
 - Worth 12 points
 - Still mutually exclusive

Analyzing Options

Because each solution is mutually exclusive, we made a table to analyze the possible points for each option:

Purely based on score, the high hang is the best way to go forward.

	Current	Blue Rings	Descore	High Hang
Top Blue Rings	0	3	0	0
Top Red Rings	7	4	9	7
Missed Red Rings	1	0	0	0
Normal Rings	16	20	15	17
Goals in Corner	4	4	4	4
Hang Tier	1	1	1	3
Total Score	60	64	65	70
Min Score	60	54	65	70

10/05/24 Brainstorm/Background Research: High Hang

Designed by: Carl

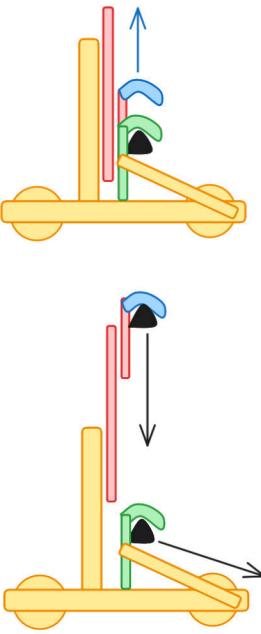
Witnessed by: Alex

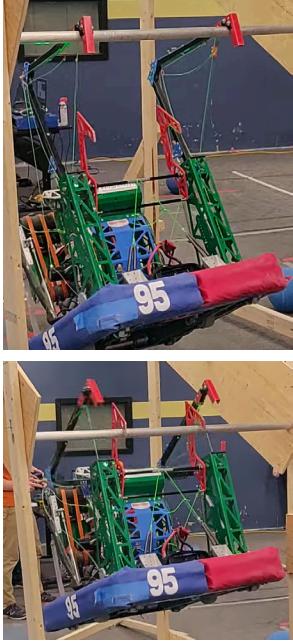
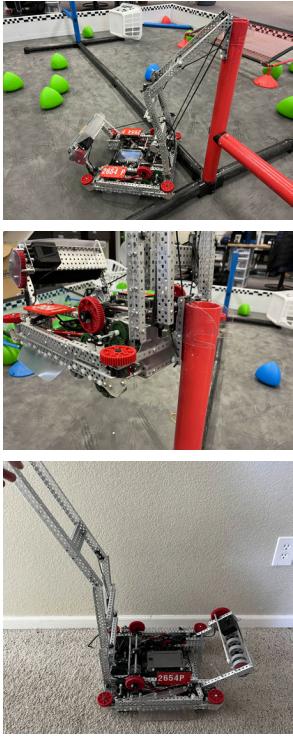
Witnessed on: 10/05/24

Goal: Research and brainstorm different options for a high hang (ideally tier 3).

Hang Options:

As mentioned [in our previous skills analysis](#) (previous page), a tier three hang (12 points) is key to achieving a maximum skills score. Although this hang is not currently necessary to get a world record, in the future we believe other teams will soon get higher scores. Hanging high in matches also provides a fast way to recover points, and is more feasible following the [September 3rd game manual update](#) (Pg. 175-176) due to the longer endgame. Here are some general hang concepts that we conceptualized/researched:

System Name	Description	Image/Diagrams	Pros & Cons
Vertical Linear Slides + Hook (Option 1) Idea Credit: Original Concept	This concept uses linear slides in a similar way as a cascade lift (Pg. 58). The linear slides would be mounted near the center of the robot, with hooks at the top to pull the robot up. There is a second pair of hooks mounted on the chassis to hold the robot in position as the lift reaches for the next bar. The lift passively raises using rubber bands and retracts using a winch with a PTO from the drivetrain.		Pros: <ul style="list-style-type: none"> • No horizontal expansion • No additional pistons or motors • Easiest to get working Cons: <ul style="list-style-type: none"> • Occupies space required for most intakes • Requires custom linear slides • Interferes with most wall stake mechanisms

System Name	Description	Image/Diagrams	Pros & Cons
Two Stage Arm + Hook (Option 2) Idea Credit: FRC Team 95: Grasshoppers	Similar to option one, this design uses a PTO from the drivetrain to power a winch. For this design, the rope pulls down on a hook attached to the top of a two stage lift. As the hook approaches the uprights, the weight of the robot is shifted onto rigidly mounted hooks as a result of pistons changing the geometry of the lift.	 <i>(The Grasshoppers FRC 95 2022 Improved Traversal Climb)</i>	<p>Pros:</p> <ul style="list-style-type: none"> Two sides can be constructed independently, <u>leaving space for ring mechanisms</u> We all had experience building similar hangs in Over Under <p>Cons:</p> <ul style="list-style-type: none"> Requires two double actuating pistons to control arms Significant design changes necessary in order to meet size constraints May not work on vertically stacked bars
Two Stage Arm (Vertical Pipe Assent) Idea Credit: Carl Richter/ 2654P 2024	This design essentially utilizes a high hang originally developed by us in Over Under to climb up the vertical post on the corner of the hanging structure. The two stage lift only allows for a tier 2 hang, but would avoid any issues associated with getting around rungs. This hang is also winch driven.	 <i>(From 2654P 2024)</i>	<p>Pros:</p> <ul style="list-style-type: none"> Very fast ascent Almost guaranteed to work Within expansion limits Lightweight <p>Cons:</p> <ul style="list-style-type: none"> Only tier two Post base and pole clamp would make a mobile stake mech implausible Hang stows in the same location as ring manipulators

10/07/24 Identify/Design: Mechanism Limitations and Constraints

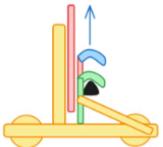
Designed by: Carl

Witnessed by: Alex

Witnessed on: 10/08/24

Goal: Identify limitations of integrating hang designs into different robot designs.

To fully characterize what is and what isn't possible with different robots, particularly ring scoring mechanisms, we created a spreadsheet.

Hang Design:	Desired Criteria:	Possible?
Option 1		
	tier 3	Yes ▾
	hook intake	Yes ▾
	flex wheel intake	Yes ▾
	single ring wall stakes	No ▾
	double ring wall stake	No ▾
	current wall stake mech	No ▾
	in horizontal expansion	Yes ▾
	in vertical expansion	Yes ▾
Option 2		
	tier 3	Yes ▾
	hook intake	Yes ▾
	flex wheel intake	Yes ▾
	single ring wall stakes	Yes ▾
	double ring wall stake	Maybe ▾
	current wall stake mech	Maybe ▾
	in horizontal expansion	Maybe ▾
	in vertical expansion	Maybe ▾
Option 3		
	tier 3	No ▾
	hook intake	Maybe ▾
	flex wheel intake	Yes ▾
	single ring wall stakes	Maybe ▾
	double ring wall stake	No ▾
	current wall stake mech	No ▾
	in horizontal expansion	Maybe ▾
	in vertical expansion	Yes ▾

After carefully considering each design and category, we concluded that it would be ideal to pursue the second option as it is the only one we think has a chance of fitting with our current wall stake design. This quality is incredibly important as it is this high efficiency scoring that has allowed us to achieve the high score we currently have. The next step in developing this hang will be to do prototyping of the concept and modify the design accordingly.

10/18/24 Brainstorming/Design: Hang Prototyping

Designed by: Carl	Witnessed by: Alex	Witnessed on: 10/18/24
-------------------	--------------------	------------------------

Goal: Identify adaptations that must be made to our chosen hang design in order for it to function in this game and with our design.

Necessary Changes	Implementation Plan
Fit within horizontal expansion constraints: <SG2>	<ul style="list-style-type: none"> No mechanisms should expand forwards (hang needs to expand backwards; expanding in multiple directions is not permitted) Create a mechanism that levels out the robot while fully supported by one rung (hang arm will need to reach back less if the robot is not leaning forwards).
Fit within vertical expansion constraints: <SG3>	<ul style="list-style-type: none"> Shorter hang arms (hang bars are closer together than they were in the FRC game). Robot should be parallel with the ground when reaching for the next rung so as to not tip below the previous rung or touch the floor.
Replace motors dedicated to hanging with a PTO from the drivetrain	<ul style="list-style-type: none"> Relocate winch near drivetrain Create a PTO that operates independently on each side of the drivetrain to reduce weight and increase space in the center of the robot.
Clamp onto a rung and level the robot	<ul style="list-style-type: none"> Create a triangular profile that matches the shape of a rung Use the 4 bar to push the bar into the profile to flatten the robot

Prototype Hang Motion with Refined Geometry:

To be able to visualize the motion of our hang, we created a quick 3-D printable model to interact with.



10/20/24

Design: Initial Hang CAD (C.3.1)

Designed by: Carl

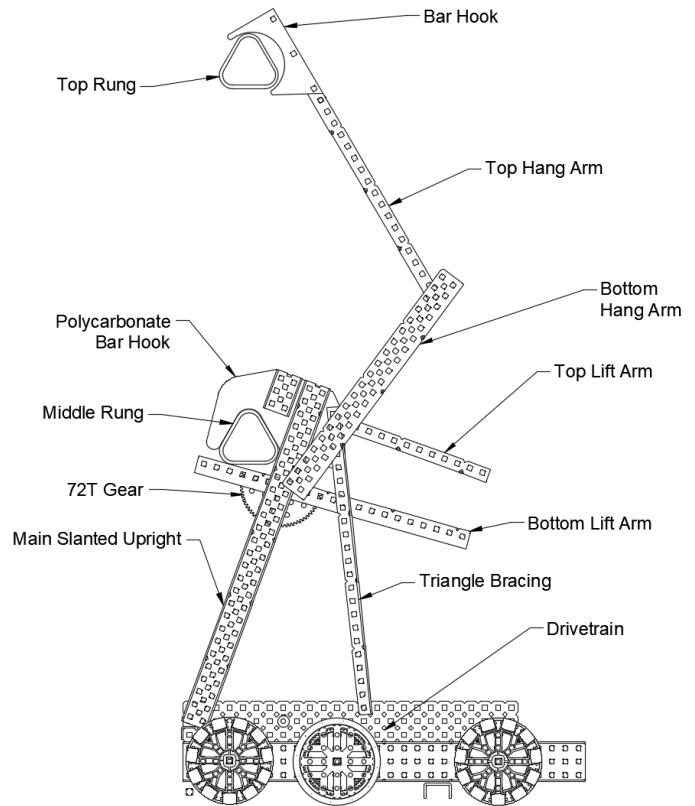
Witnessed by: Alex

Witnessed on: 10/20/24

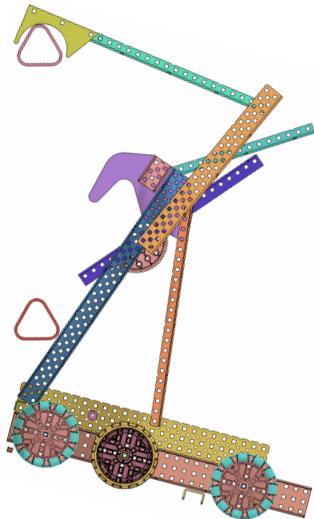
Goal: Create a CAD model of our concept hang design.

Key Features:

- Polycarbonate Hooks: They match the shape of the bar and will allow the robot to rotate with lower friction
- Main Slanted Upright: Allows the robot to smoothly move past each rung without getting stuck
- Bottom Lift Arm: Similar to the lift on [C.2.2](#) (Pg. 159), but the backwards extension allows the arm to push a Rung into the Polycarbonate Hook
- Triangle Bracing: Ensures that the slanted upright remains at the desired angle and provides mounting holes for the brain.
- Bottom Hang Arm: 2x2 U-Channel is used to prevent twisting which is crucial because the sides of our hang must be built independently due to our wall stake mechanism



Hang Images:



(Robot pulling up to the next rung)



(Transferring to fixed hook)



(Pull bar in, begin to level robot)

10/22/24

Brainstorming: Lift Gearing Calculations

Designed by: Carl

Witnessed by: Alex

Witnessed on: 10/22/24

Goal: Perform necessary calculations to determine what gear ratio we need on the lift to clamp onto a rung effectively.

Givens:

$$\text{mass } (m) = 6.8 \text{ kg}$$

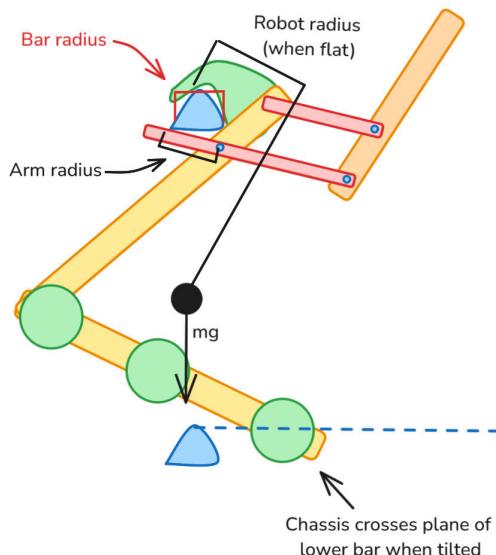
$$\text{gravity } (g) = 9.8 \frac{\text{m}}{\text{sec}^2}$$

$$\text{robot radius } (r_r) = 0.12 \text{ m}$$

$$\text{bar radius } (r_b) = 0.05 \text{ m}$$

$$\text{arm radius } (r_a) = 0.07 \text{ m}$$

Hang Forces Illustration



Standard Equations:

$$F = mg$$

$$\tau = rF$$

Derivation for Necessary Gear Ratio:

$$\tau_r - \tau_b = 0$$

$$\tau_b = \tau_r \therefore \tau_r = \tau_a$$

where τ_b , τ_r , and τ_a represent torque from their respective components

$$r_r \cdot m \cdot g = \tau_a$$

$$\tau_a \cdot b = \tau_{5.5W} \cdot 2$$

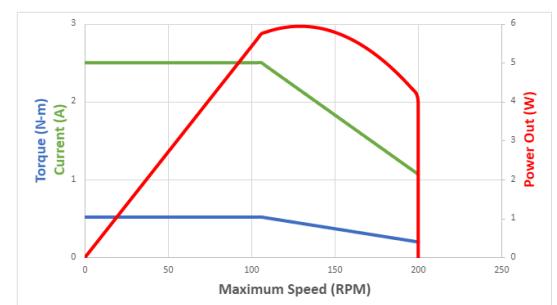
Where b is the necessary gear ratio and $\tau_{5.5W}$ is the stall torque of a 5.5W motor (0.5Nm) (image on the right).

$$\frac{2(\tau_{5.5W})}{\tau_a} = b$$

$$\frac{2(\tau_{5.5W})}{r_r \cdot m \cdot g} = b$$

Plugging in values to find final gear ratios:

$$\frac{2(\frac{0.5 \text{ kgm}^2}{\text{sec}^2})}{\frac{0.12 \text{ m}}{1} \cdot \frac{6.8 \text{ kg}}{1} \cdot \frac{9.8 \text{ m}}{\text{sec}^2}} = \frac{\frac{1 \text{ kgm}^2}{\text{sec}^2}}{\frac{8.0 \text{ kgm}^2}{\text{sec}^2}} = \frac{1}{8}$$



(vexrobotics.com)

In summary, a 1:8 gear ratio is the bare minimum, but this does NOT have a margin of error built in. In practice, we would need at least a 1:12 ratio, which would result in unacceptably long hang and wall stake times.

11/01/24 Evaluate/Testing: Skills Time Distribution

Designed by: Alex

Witnessed by: Carl

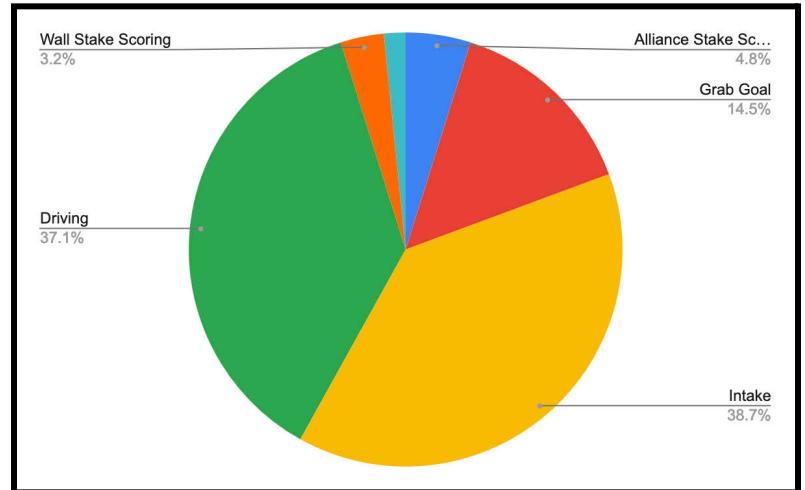
Witnessed on: 11/01/24

Goal: Create a visual representation of how we spend time in skills; use this to determine what subsystems would benefit most from additional optimizations.

To compile the data we went through the following process:

1. Watch skills video
2. Note down the times the robot was completing certain tasks
3. Compile data into Pie Chart

Task	Time
Alliance Stake Scoring	3
Grab Goal	9
Intake	24
Driving	23
Wall Stake Scoring	2
Hang	1



Analysis

After looking at the pie chart the biggest single columns are driving and intaking. For driving time we currently run the robot slower than the maximum speed, so by spending time to optimize the drive code to allow the robot to quickly move paths will be a very helpful next step, however the robot doesn't need to move much faster as it would sacrifice acceleration. On the other hand, there isn't much code that can change about intaking, so we will focus on how we can intake faster with hardware improvements. Additionally, even though the time is minor, the goal grabbing can also be easily improved with a better clamp.

- Time analysis
 - Intaking and driving take the most time
- Fixing
 - Driving
 - Speeding up in code will help more than hardware changes currently
 - Intaking
 - Focusing on intake speed will be the most effective use of our time to increase score in skills
 - Goal Clamp
 - Better goal clamp could save 2-3 seconds in skills

11/02/24 Analysis/Brainstorming: R.1.2.11 Subsystems

Designed by: Carl, Matt

Witnessed by: Alex

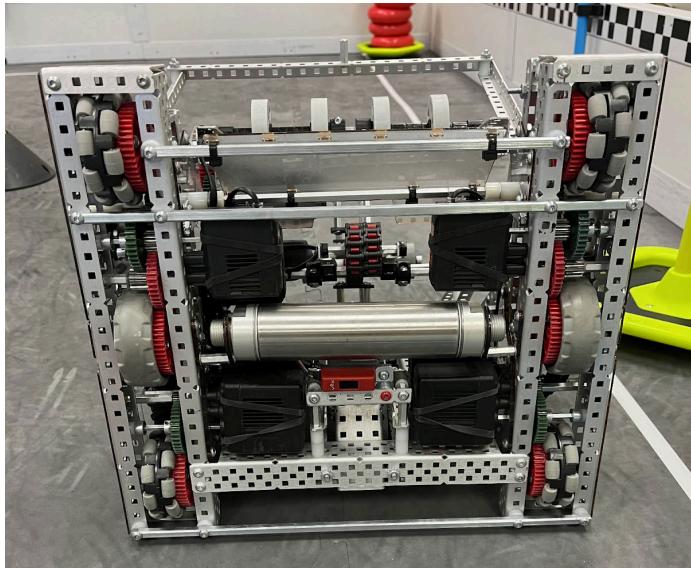
Witnessed on: 11/03/24

Goal: Analyze the performance of our previous robot's subsystems to determine what can be further optimized on the new robot.

Drivetrain:

Previous Subsystem Analysis	Optimization Ideas and Desired Traits
<p>Pros:</p> <ul style="list-style-type: none"> • Traction wheels effectively prevent sideways movement • Drivetrain is very controllable • No burnout issues • Very quick acceleration • Low center of gravity <p>Cons:</p> <ul style="list-style-type: none"> • Lots of gears meaning more weight and friction • Back HS shaft slightly drags on the ground • Drive used lots of polycarbonate 	<ul style="list-style-type: none"> • Use larger wheels to... <ul style="list-style-type: none"> a. reduce the gears needed to connect the drivetrain b. increase space beneath chassis for crossbars • Maintain a similar drive speed • Integrate drive motors into the lift structure • Reduce offset gears in the drive to reduce polycarbonate usage, friction, and weight

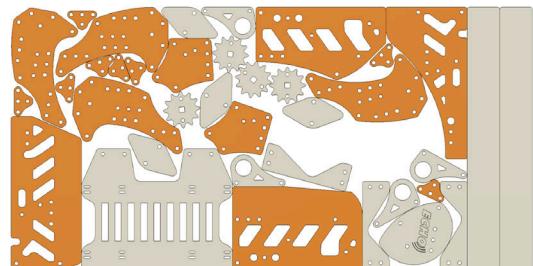
Bottom View of Drivetrain:



Offset Gears Side Profile:



Polycarbonate Used for Drive (highlighted):



Lift:

Previous Subsystem Analysis	Optimization Ideas and Desired Traits
<p>Pros:</p> <ul style="list-style-type: none"> • No burnout issues • Only one 5.5W motor • Very lightweight arms • Never jammed, never got stuck • No polycarbonate used for the entire lift <p>Cons:</p> <ul style="list-style-type: none"> • Slower than ideal • Motor is mounted relatively high • HS shaft connecting both sides prohibits moving the intake further back <ul style="list-style-type: none"> ◦ Lowers alliance stake consistency ◦ Adds high weight • Some wobble in top intake because of the standoff pivot 	<ul style="list-style-type: none"> • Use two 5.5W motors to power lift <ul style="list-style-type: none"> ◦ Almost twice as fast ◦ Each side of the lift can be controlled independently; no HS shaft needed ◦ Allows for much more flexibility in motor placement ◦ Gear ratio should be built in such a way that changing it for more speed or torque would be easy • Mount motor(s) as low as possible <ul style="list-style-type: none"> ◦ Helps with COG ◦ More space for hang • HS Shafts for both intake pivots <ul style="list-style-type: none"> ◦ Minimal slop ◦ No chance of bending

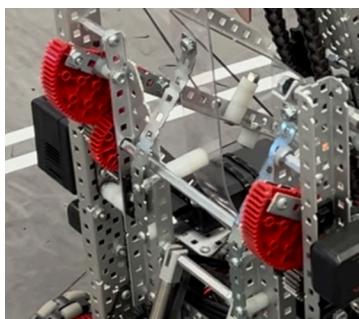
CAD of Lift Geometry:



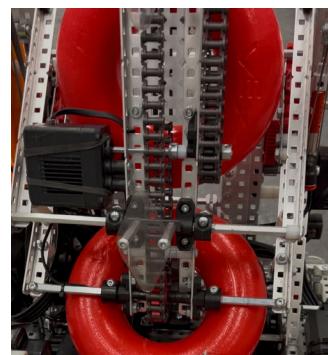
Lift Gearing & Arm Construction:



Full Lift Gearing:



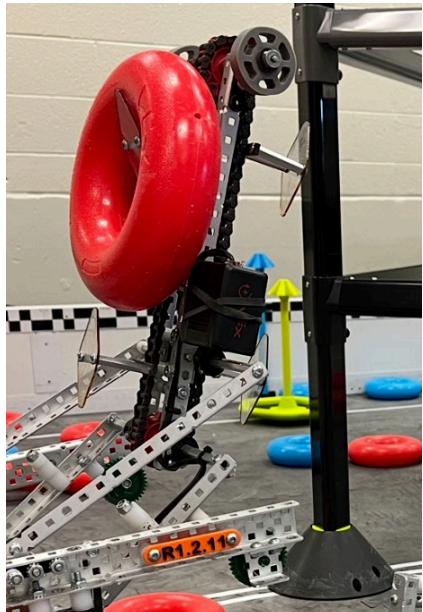
Lift Front View While Loading Wall Stake Mech:



Intake (Top Stage):

Previous Subsystem Analysis	Optimization Ideas and Desired Traits
<p>Pros:</p> <ul style="list-style-type: none"> • Fast • Reliable • Strong construction <p>Cons:</p> <ul style="list-style-type: none"> • Relatively Heavy • Sharp transition with the bottom stage • Intake slows down when intaking several rings in close concession • Intake doesn't quite fit two rings on the back so they occasionally fall off • Struggles to score on wall stakes with 5 rings <ul style="list-style-type: none"> ○ Hooks catch on scored rings • Hooks can get stuck in the indented letters on the rings 	<ul style="list-style-type: none"> • Reconsider the structure to try and remove some weight • Add a 5.5W motor to increase torque • Use hooks with a blunter edge • Lengthen the top stage by a few holes <ul style="list-style-type: none"> ○ Will help hold 2 rings more securely ○ Should be able to grab rings earlier allowing for a smoother transition between intake stages • Change 4-bar geometry to tilt the intake more at the top to move hooks away from scored rings

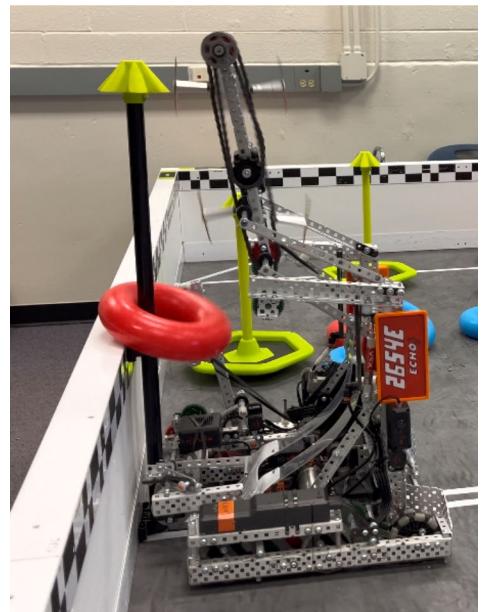
Top Stage with Ring:



Intake Scoring on Goal:



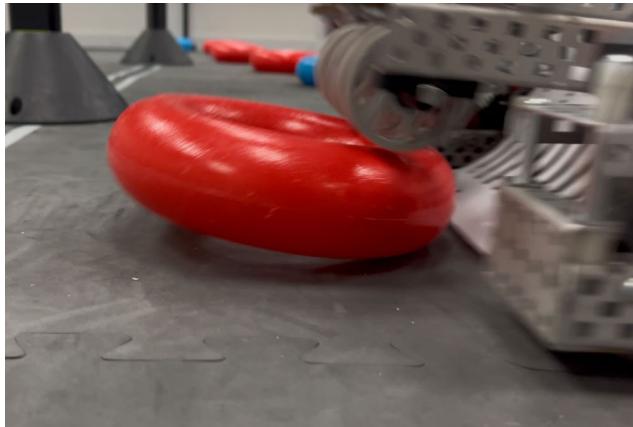
Scoring on Wall Stake:



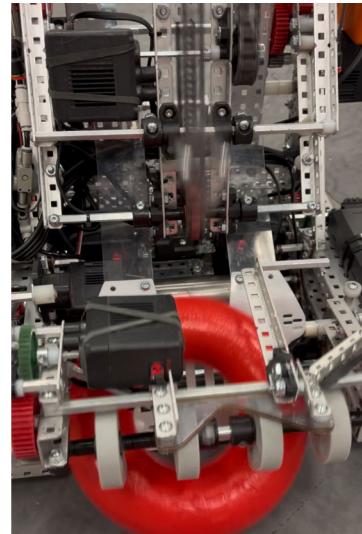
Intake (Bottom Stage):

Previous Subsystem Analysis	Optimization Ideas and Desired Traits
<p>Pros:</p> <ul style="list-style-type: none"> • Rigid and durable <ul style="list-style-type: none"> ◦ Triangle bracing, boxing, HS shafts, protected gears • Good wall stake aligner • Low polycarbonate usage <p>Cons:</p> <ul style="list-style-type: none"> • Good performance but far from optimal <ul style="list-style-type: none"> ◦ Intake back is too steep • Can't remove rings from corner • Heavy structure • Uses an 11W motor 	<ul style="list-style-type: none"> • REDUCE SLOPE ON INTAKE BACK for <ul style="list-style-type: none"> a. Smoother transition between stages b. More gradual angle change for ring c. Easier to get ring off of the ground • Intake out of the corner in auton <ul style="list-style-type: none"> ◦ Smaller wheels to help get under rings ◦ Polycarbonate plate to "cut" the stack of rings • Lighten structure, maintain durability <ul style="list-style-type: none"> ◦ 1x1 channels instead of 1x2x1's ◦ Keep using gears ◦ Box only high stress areas • Only use a 5.5W motor <ul style="list-style-type: none"> ◦ This should have sufficient torque assuming shallower intake slope, low friction, and similar speed

Struggling to Lift Ring off of the Ground:



Transition to Top Stage:



Protected Gearing:



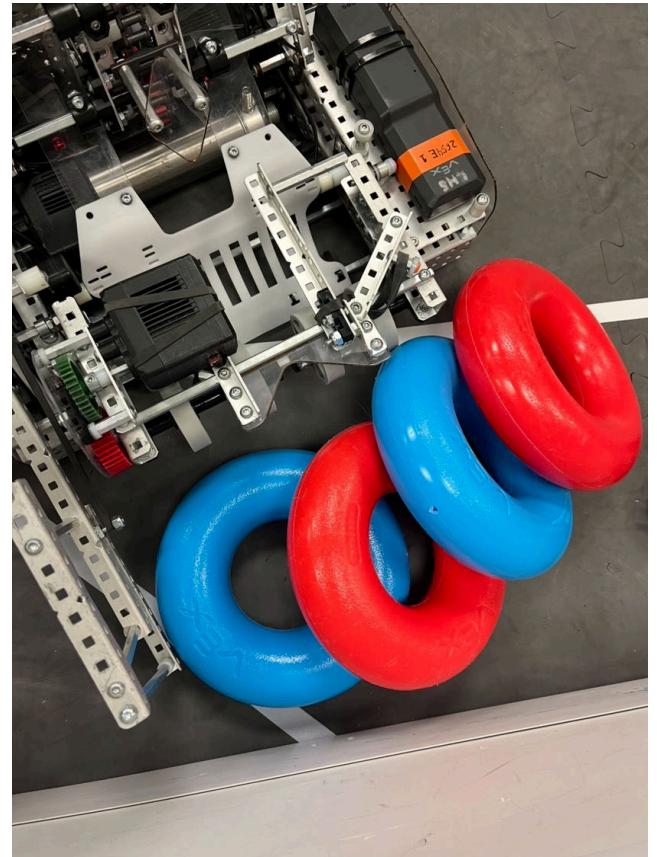
Corner Sweeper:

Previous Subsystem Analysis	Optimization Ideas and Desired Traits
<p>Pros:</p> <ul style="list-style-type: none"> • Can quickly clear a corner • Helps balance hang <p>Cons:</p> <ul style="list-style-type: none"> • Unnecessary use of a 5.5W motor <ul style="list-style-type: none"> ◦ Could be replaced with a piston • Pushed rings out to varying positions; we could not reliably intake those rings in auton • Took up a lot of space 	<ul style="list-style-type: none"> • Remove the mechanism entirely <ul style="list-style-type: none"> ◦ Won't be possible due to <SG2> as the hang needs to expand backwards ◦ With our planned intake changes, this system will not be as necessary

Sweeper Entering Corner:



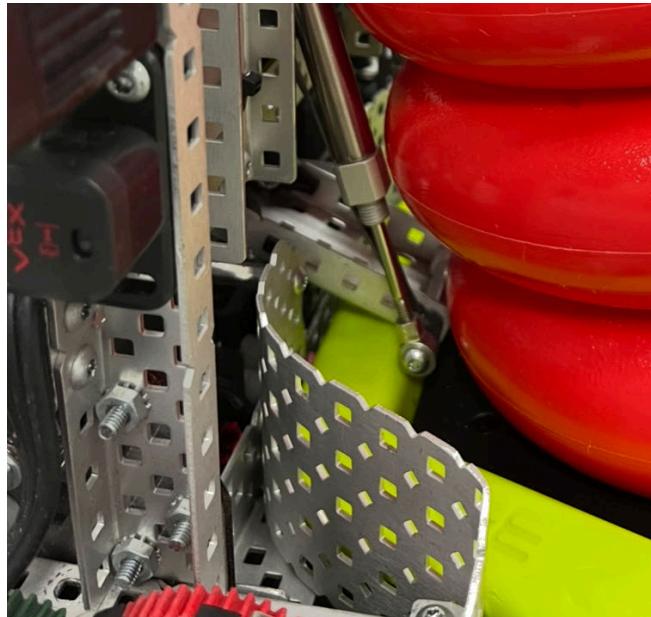
Rings Falling Unpredictably:



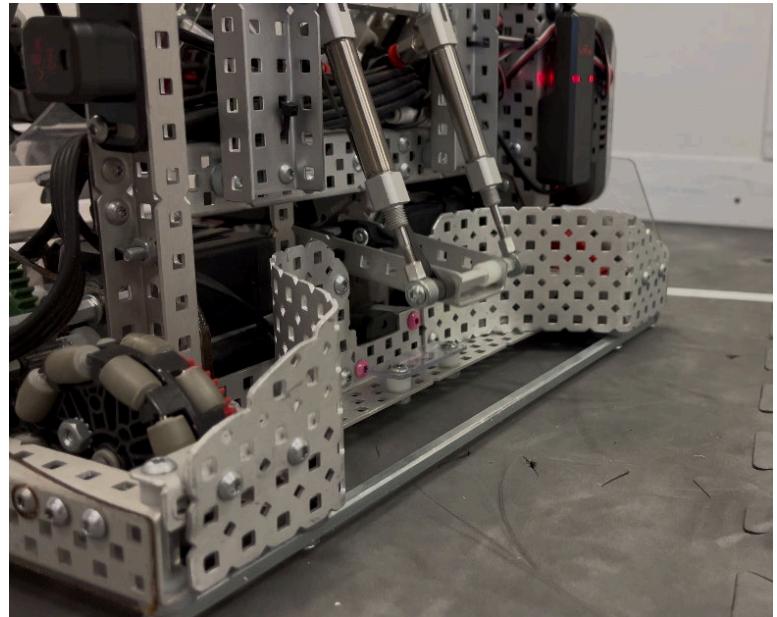
Back Clamp:

Previous Subsystem Analysis	Optimization Ideas and Desired Traits
<p>Pros:</p> <ul style="list-style-type: none"> • Holds the mobile stake very securely • Very little air usage (around 15 actuations with 1 reservoir) • Robust system <p>Cons:</p> <ul style="list-style-type: none"> • Struggled to grab goals if certain conditions were present including: <ul style="list-style-type: none"> ◦ Robot moving too fast ◦ Robot not moving at constant velocity ◦ Goal orientation too far off ◦ Goal tilted ◦ Goal was more than 0.25" away from clamp • Pistons were very exposed and sustained some damage • HS shaft rubbed on the ground 	<ul style="list-style-type: none"> • Use two 25mm pistons to save air • Position the pistons within the robot so they are less exposed • Goal clamp should open wider to help with tilted goals • HS shaft should be further off of the ground • The clamp should pull in instead of just pressing down on the edge of the goal <ul style="list-style-type: none"> ◦ This should eliminate almost all issues with grabbing goals • Clamp should be very secure <ul style="list-style-type: none"> ◦ Over centering mechanism ◦ Optimized geometry

Back Clamp Holding Mobile Stake:



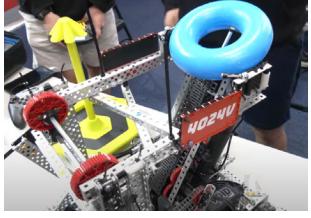
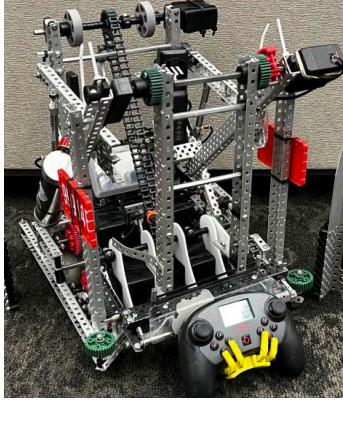
Clamp Without Goal and Low Clearance HS Shaft:



Overall Reflection/Comparative Analysis

Overall, this design was very successful in both matches and skills, performing most tasks extremely efficiently and having outstanding reliability. Since our last signature event analysis, many new wall stake designs have emerged. We want to determine if switching to one of these designs could be beneficial or if it would reduce our robots capabilities.

Mechanism Name	One Ring Wall Stake?	Two Ring Wall Stake?	Hook Intake?	Tier 1 Hang?	Tier 3 Hang Potential ?	Descore Wall Stakes?	Wall Stakes in Auton?	Small Robot Footprint?	Instant Wall Stakes?
Lady Brown	Yes	No	Yes	Yes	Maybe	Yes	Yes	Yes	Yes
Echo Mech	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes
Fish Mech	Yes	No	Yes	Yes	No	No	No	No	No
Redirect	Yes	Yes	Yes	Yes	No	No	Maybe	No	Yes

Lady Brown (4042V)	Echo Mech (2654E)	Fish Mech (99904E)	Redirect (5150Z)
<p>This design uses an arm connected to the front of the robot with the intake in between to squeeze rings in between so it can bring it up.</p> 	<p>This design has an intake on a four-bar which can load rings behind then lift it up to score.</p> 	<p>This mechanism uses a swinging arm at the front of the robot to score on the wall stake.</p> 	<p>This design reverses the intake so it goes into a basket with a 2 bar which raises to score.</p> 

With the extreme success of our robot design, we believe it is beneficial to keep a similar design especially as it IS THE ONLY DESIGN that has a small footprint, double wall stakes, and clear potential for a tier 3 elevation. Furthermore, we also have lots of experience with this design, making it even more beneficial to continue with.

11/02/24 Evaluate/Brainstorming: Hang Concept

Reconsideration & Changes

Designed by: Carl

Witnessed by: Alex

Witnessed on: 11/02/24

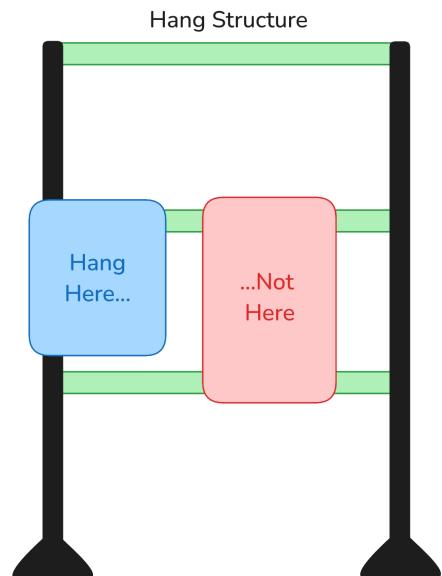
Goal: Improve our current hang concept to be faster while ideally reducing moving parts and complexity.

After further consideration of our hang mechanism, we discovered a few potential problems with our current [design](#) (Pg. 247). To further refine our design, we examined the pros and cons of our hang concept.

Pros	Cons
<ul style="list-style-type: none"> • Works with our wall stake mechanism <ul style="list-style-type: none"> ◦ Keeps our robot “optimal” ◦ Limited additional structure needed for hang • Lower stress on hang arms compared to other hang designs • PTO and winch are easy to integrate with the drive • Very easy to align 	<ul style="list-style-type: none"> • High torque needed in the lift <ul style="list-style-type: none"> ◦ Would need a lot of structure ◦ Would result in a slower lift • Would fall off at the end of a match <ul style="list-style-type: none"> ◦ Lift motors lose power resulting in the 4-bar losing clamping force ◦ Could be prevented with a piston actuated “lock” • 4-bar clamping takes time • Will be very difficult to stay in vertical expansion constraints • Pistons required

Evaluation:

Although our hang design is promising in most aspects, it still has a few issues and is relatively complex. Almost all of the issues with our design are related to leveling the robot at each individual tier. Although hanging in the center of the rungs is ideal for alignment, hanging on the side of the hang structure would allow us to leverage our robot on the vertical posts in the hang structure. By using these posts we should be able to eliminate most problems with keeping the robot flat, allowing us to comply with <SG3> with a much simpler design.



11/03/24

Design/Planning: Design Methodology

Designed by: Carl

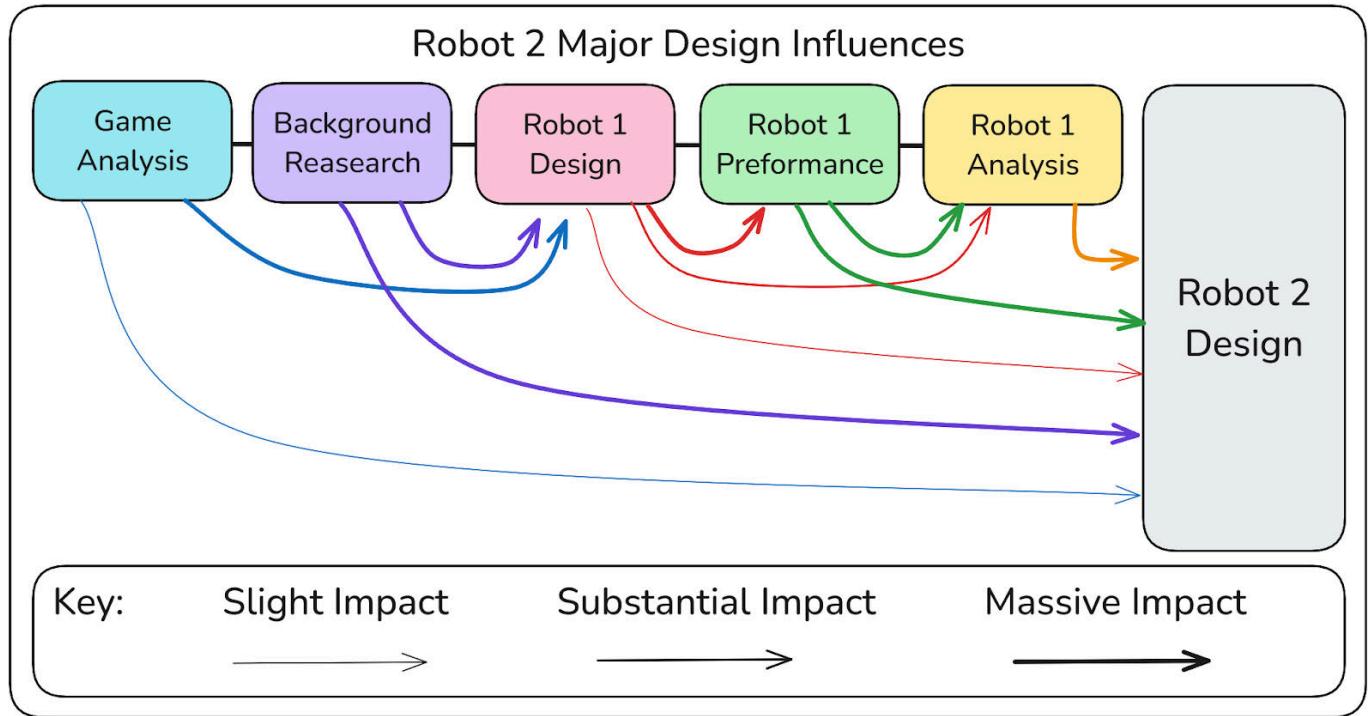
Witnessed by: Alex

Witnessed on: 11/03/24

Goal: Create a plan for designing the new robot. ([TIME MANAGEMENT ON PAGE 225](#))

Since we have already built a robot this season and conducted thorough game analysis and research, reanalyzing the game would be redundant. Our early-season research provides a solid understanding of the rules, limitations, and constraints, allowing us to save time during the development of this robot.

This knowledge has been significantly expanded over the course of the season by developing our robot, reflecting on our design, and analyzing our tournaments as well as other big events. The majority of our robot's design changes will almost certainly come from our analysis of our previous robot's strategy, build, and design, along with a few other factors that can be seen in the flowchart below:



(Note: Impact levels based on expected relevance)

In terms of determining specifics for this robot, we will be doing any necessary calculations, decision matrices, and other subsystem development in conjunction with the CAD to increase clarity and reduce the need to reference information from other pages.

11/03/24

Design: CAD Day 1 (C.3.2)

Designed by: Carl

Witnessed by: Alex

Witnessed on: 11/04/24

Goal: Start designing the drivetrain and create a basic design of the top intake.

Drivetrain Choice:

In order to accommodate an 11W lift and 22W intake, we will only have 55W available for use on our drivetrain. This is the same power as robot one's drivetrain, however it is 11W less than the standard 6 motor drive that most teams use. As highlighted [earlier](#) (Pg. 250), we had no problems with the drivetrain burning out in matches or skills runs, and the speed and acceleration was comparable to most robots.

Although the drive did not have any significant performance issues, the construction of the chassis had several key issues ([listed here](#)(Pg. 250)), all of which could be fixed by altering the gearing of the drive. To find a suitable substitute for our current ratio, we revisited the [table of gear ratios](#) (Pg. 102) that we created previously.

Chassis Stats	Old Drivetrain	New Drivetrain
Wheel Size (diameter)	2.75"	3.25"
Output RPM	450	360
Input RPM	600	600
Gear Ratio	36:48 (3:4)	36:60 (3:5)
Max Speed	65 in/sec	61 in/sec

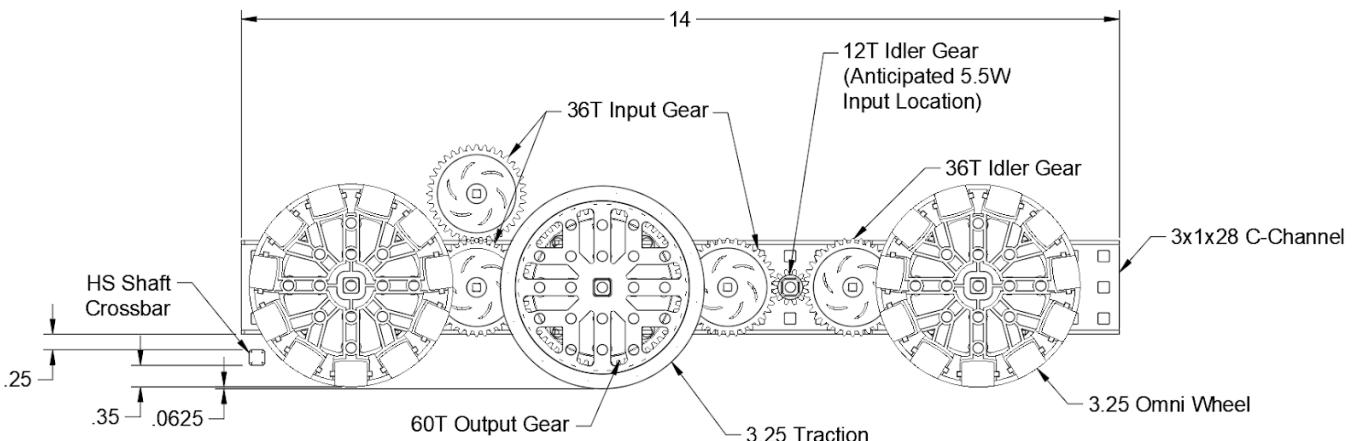
Wheel Choice: [3.25" Wheels](#) (Pg. 69)

- Better crossbar mounting
- Not too big
- High weight capacity

Gearing Choice: 36:60

- Viable speed
- We have significant experience with this gear ratio
- Fits with hole pattern nicely

Although this drive ratio is about 6% slower than our old one, the benefits it offers significantly outweigh the slight loss in speed. A quick CAD layout of our gearing demonstrates the simplicity and viability of this ratio:



Drive Gearing Layout Explanation/Rationale:

Offset Center Wheel— We will be keeping the 1/16 downwards offset on the center traction wheel as it offers significant benefits in both matches and skills. We found the previous offset distance to be nearly perfect; big enough drop so that the traction wheel carries most of the robot's weight yet doesn't cause the robot to rock on the center wheels. Instead of offsetting the center wheel downwards, we chose to offset the omni wheels upwards, meaning we will have the option to mount one of the 36T input gears above the center traction wheel should it become necessary.

Center Wheel Position— The traction wheel is intentionally offset towards the back of the robot as we anticipate the hang and mobile stake to shift the center of gravity backwards; the traction wheel will be most effective if it is directly under the CG.

3-Wide C-Channel— We made the decision to use 3x1 C-channels for the main drive rails as it increases the mounting holes available on top of and below the drive rails at the expense of a small weight penalty. By using thicker C-channels, we also significantly reduce the amount we will need to space crossbars off of the drive rails, allowing for much stronger and squarer connections.

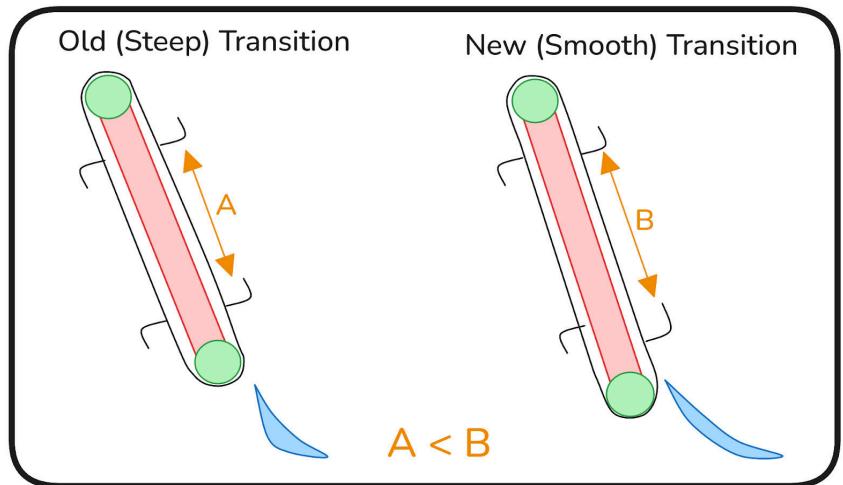
Anticipated Motor Positioning/Input Locations— Because our drivetrain has very few gears and is also relatively short, we are not able to fit all of our drive motors on the main drive rails. This means that one of the drive motors will need an additional gear in order to be connected, hence the additional 36T Input gear. As we have a 12T idler gear in the drivetrain already, it is logical to integrate the 5.5W motor there as it will be easy to gear to the correct speed with a 3:1 ratio (similar to our last robot).



HS Shaft Crossbar Mounting— Very similar to its location on the last robot; it's positioned in such a way that it could be moved up or down slightly to change the tilt of our mobile stake while still being well off the ground.

Top Stage Intake:

As determined in our analysis of [R.1.2.11](#) (Pg. 250-256), the top intake performed excellently but still has room for improvement. To give the rings enough space on each hook, we elected to extend the distance between the two sprockets to 25 holes. This change also decreases the slope necessary on the bottom stage (diagram on the right) resulting in a smoother transition of rings in between intake stages.



Motor Distribution— With a smoother transition, less power is needed on the bottom stage. The new intake will have a 16.5W top stage and a 5.5W bottom stage. This will help reduce the RPM drop when scoring on a mobile stake thus making scoring and intaking faster and more reliable.

Structural Changes— The intake will utilize one 3x1 instead of two 1x1's to save weight and increase strength.

Gearing— In theory, a slightly faster top stage would be beneficial, however, in order to not see an RPM drop when scoring, the intake needs more torque than the previous one had. We are limited to using only 6T and 12T sprockets as they are the only sprockets smaller than the belt's path. We are also limited to 100, 200, and 600 RPM motors. To the right is a table showing possible output speeds for our intake using a one stage sprocket & chain reduction.

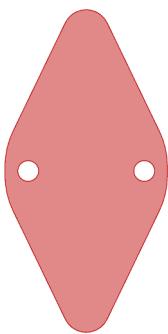
Motor RPM	Gear Ratio	Output RPM
100	6:12	50
100	6:6	100
100	12:6	200
200	6:12	100
200	6:6	200
200	12:6	400
600	6:12	300
600	6:6	600
600	12:6	1200

As none of the gear ratios offer a speed that is slightly faster than the previous, we will continue to use a 400 RPM intake. This is also beneficial as it allows us to easily integrate the 200 RPM 5.5W motor.

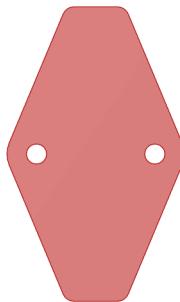
Other Changes— As we have extended the intake's top stage, in order to achieve a similar lift geometry the bottom 12T sprocket no longer needs to spin on the HS shaft supporting the intake. This means that the bottom sprocket can now be screw-joined for a lower friction joint. We will be keeping the offset pillow block on the top HS shaft pivot as it will allow us to easily tune the exact position of the top intake after the robot is built.

Both motors are mounted as low as possible to keep the CG as low as possible which will make the robot more stable.

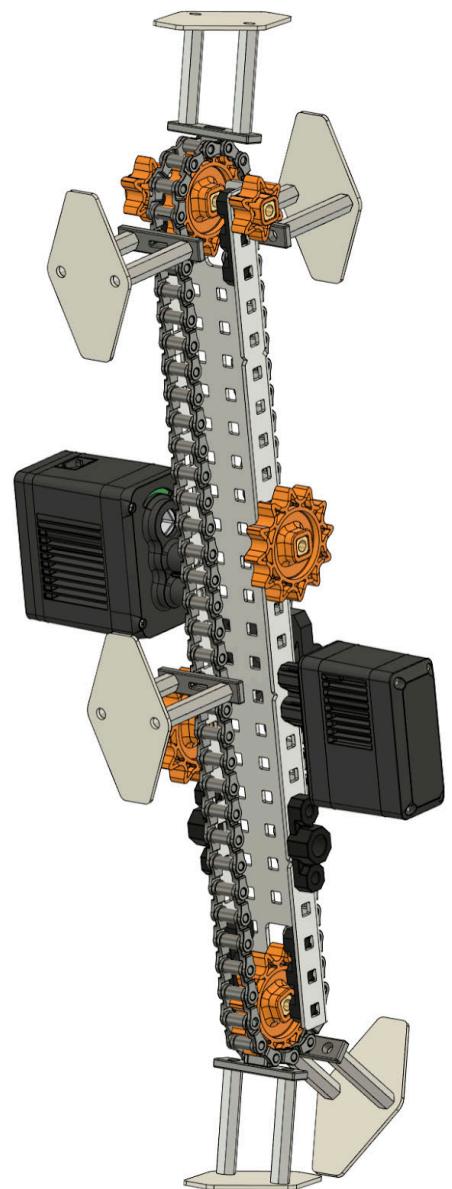
We elected to flatten the tips of our hooks as it is something that several other teams are doing and we believe it may have a positive impact on accuracy. We can test if this is true after we build it by changing them out for our old hooks.

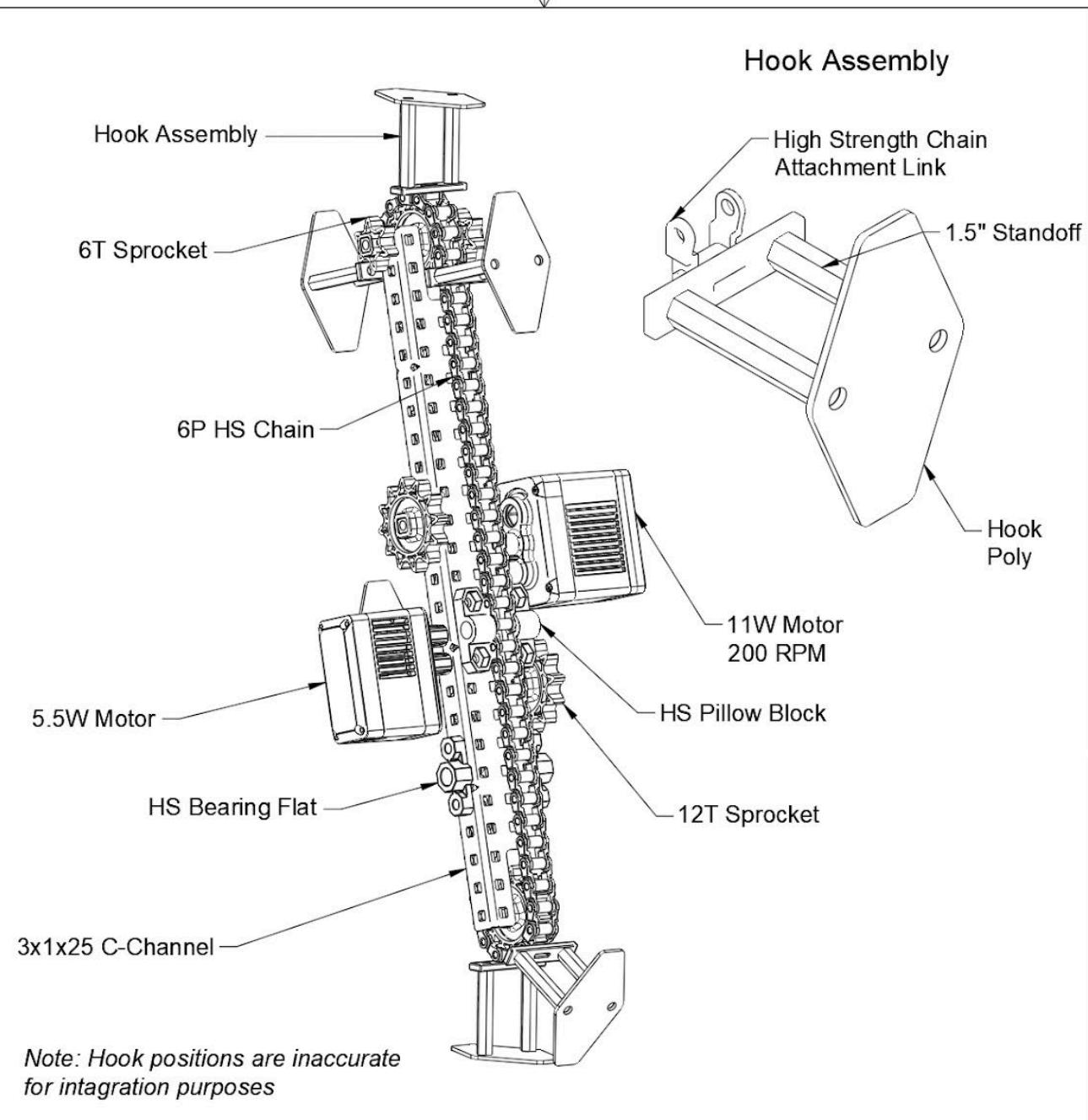


Old Hook



New Hook





2654E ECHO		PROJECT VEX		
TITLE C.3.2 Top Intake				
APPROVED	Matt Dickhans	11/3/2024	SIZE	CODE
CHECKED	Matt Dickhans	11/3/2024	A	DWG NO
DRAWN	Carl Richter	11/3/2024	SCALE 2:5	WEIGHT
				REV 1
				SHEET 1/1

11/04/24

Design: CAD Day 2 (C.3.2)

Designed by: Carl

Witnessed by: Alex

Witnessed on: 11/04/24

Goal: Finalize the drive layout with PTO/hang winch, add uprights, and input the top stage intake into the main CAD.

Drivetrain Completion:

Completing the gearing was fairly straightforward as we had already determined the [layout of the gears](#) (Pg. 259).

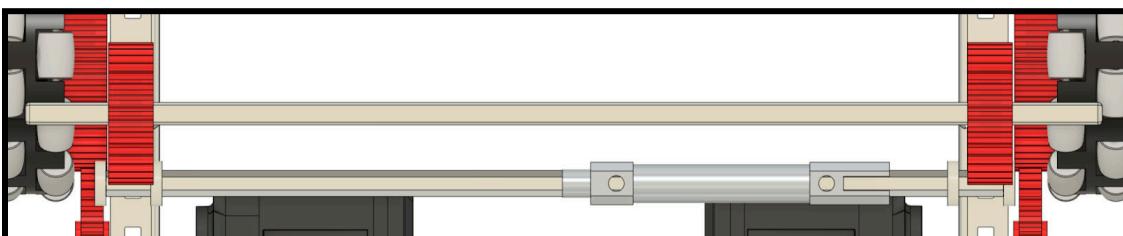
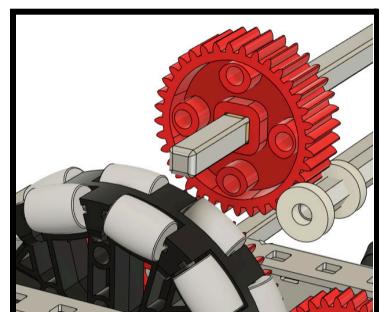
- Front Crossbar

- The front crossbar was mounted as far forward as possible while still being under the drivetrain gears, providing a solid mount for the intake back. The 2x1x28 C-channel needed to be slightly spaced away from the drive rails to keep clear of the gears. This distance is small enough that we will still be able to use shoulder screws, allowing us to keep the drivebase square when building.

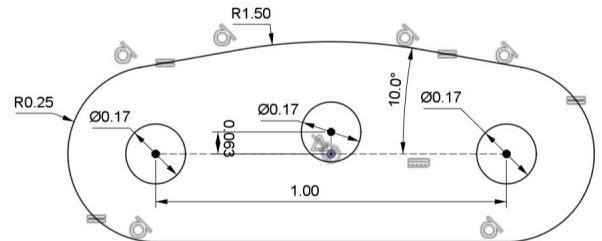


- PTO (Power Take Off)

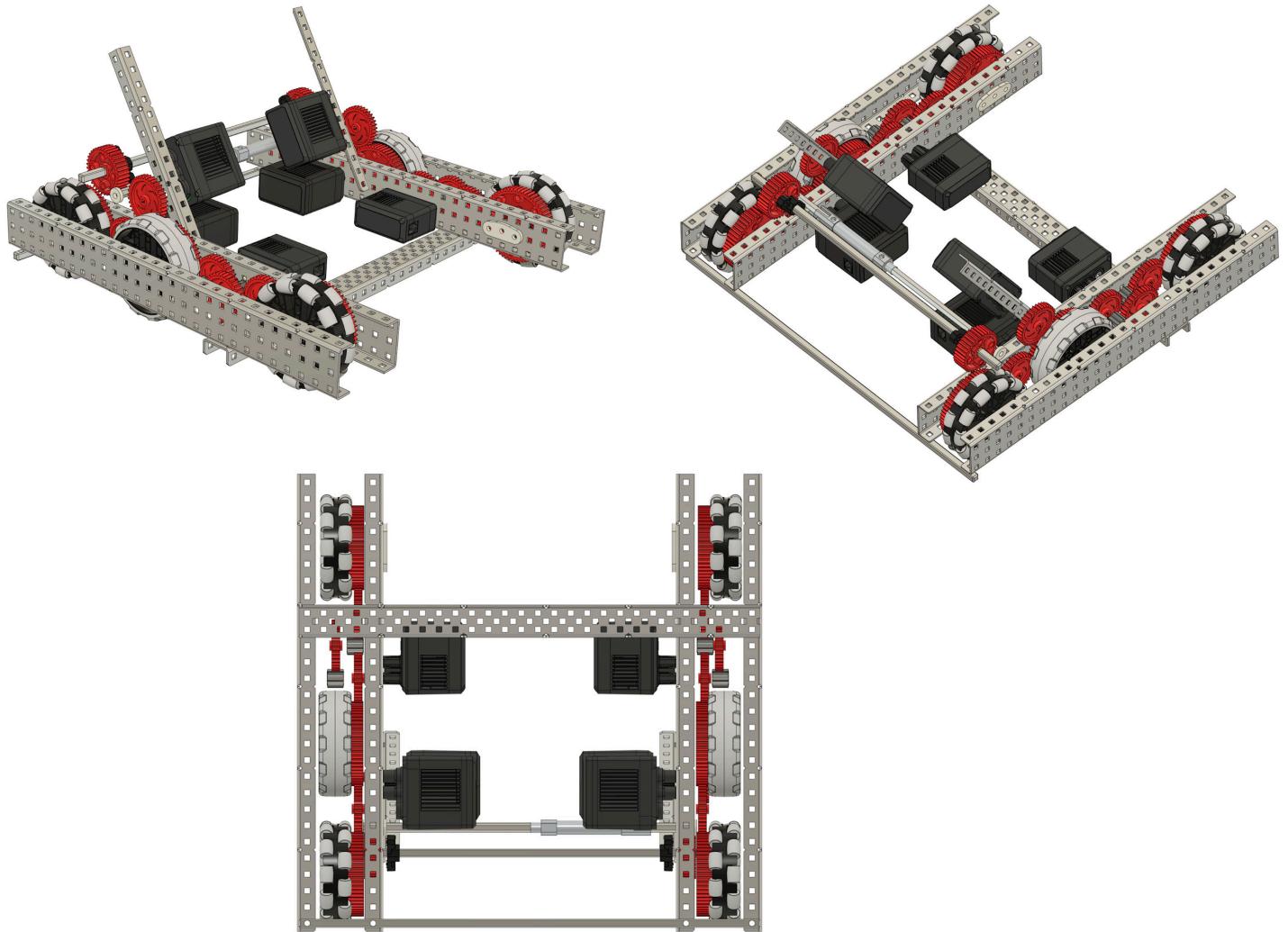
- A PTO is a mechanical device that “borrows” power from a certain subsystem on a robot
- The PTO works by sliding a gear on the auxiliary mechanism into the powered system (usually with a piston). When the gear is slid, the mechanisms become linked together.
- We have significant experience using PTO mechanisms for hanging from Over Under
 - Our previous PTOs have all been 77W drives all using 600 RPM HS shafts as the winch
 - As we had a lot of success with this winch speed on robots of similar weights, we think we have the best chance of achieving the correct winch RPM by using a similar setup
 - This setup also uses a 36T gear on the PTO/winch shaft, and is much easier to implement than other gear sizes
- To be out of the path of the intake hooks, the PTO needs to be mounted as far back as possible
- Single 25mm piston used for actuating the PTO as it is light and uses minimal air



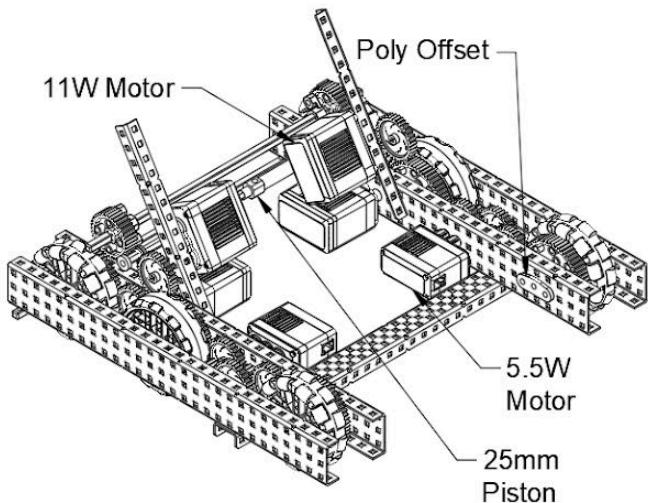
- Motor Placement
 - 11W motor on back 36T gear
 - 5.5W motor mounted on the 12T idler to allow for easy implementation of the compound gearing
 - Upper 11W motor moved to 1x1 L-channel above the center wheel as the previous mounting point interfered with the PTO
 - This 1x1 is intended to be used as a triangle brace for the lift upright
- Polycarbonate Wheel Offsets
 - 1/16" upwards offset (same as last robot) on all omni wheels
 - Part designed to minimize weak (thin) points and to nest efficiently



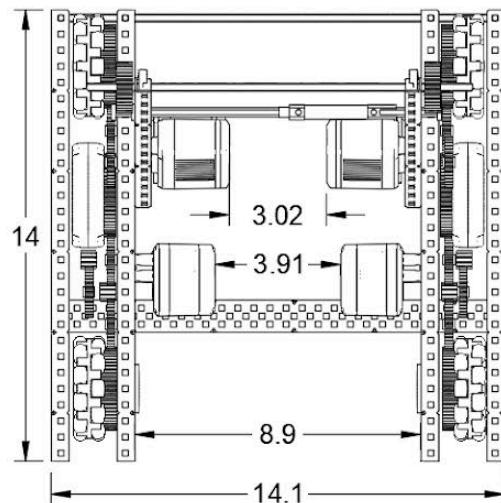
Progress Images:



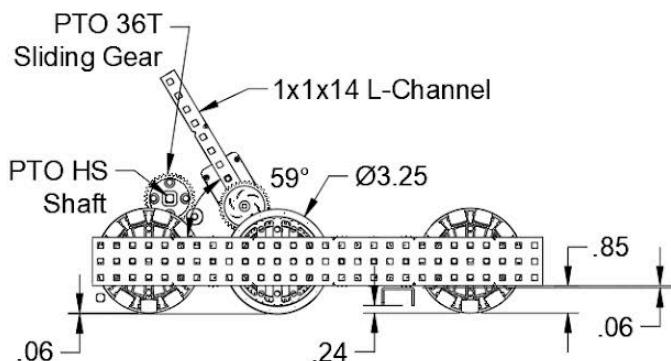
Isometric View



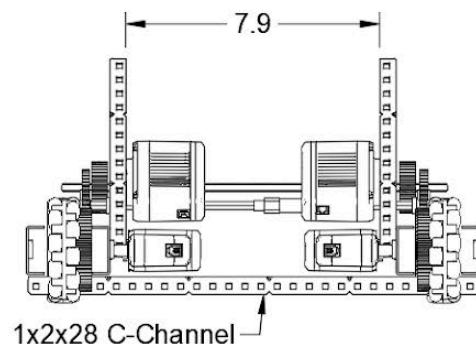
Top View



Left View



Front View



2654E
ECHO

PROJECT

VEX

TITLE

**C.3.2 Drivetrain
CAD Day 2**

APPROVED Matt Dickhans 11/4/2024

SIZE

CODE

DWG NO

REV

CHECKED Matt Dickhans 11/4/2024

A

1

DRAWN Carl Richter 11/4/2024

SCALE 1:4

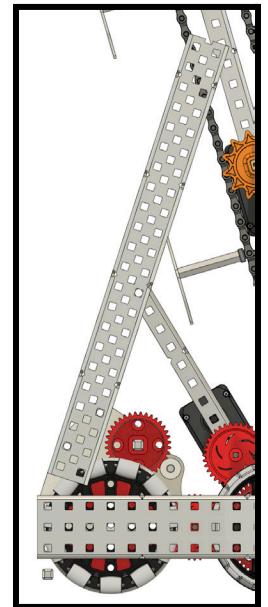
WEIGHT

SHEET 1/1

Upright Mounting & Positioning:

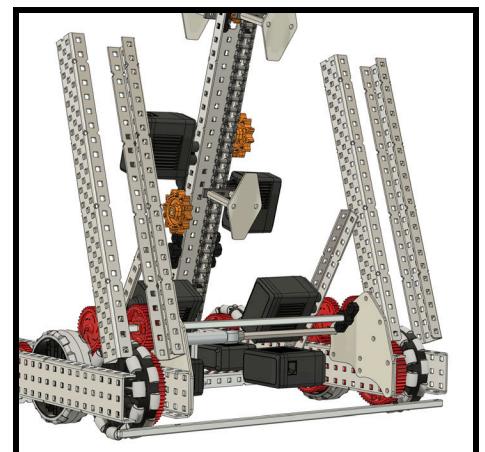
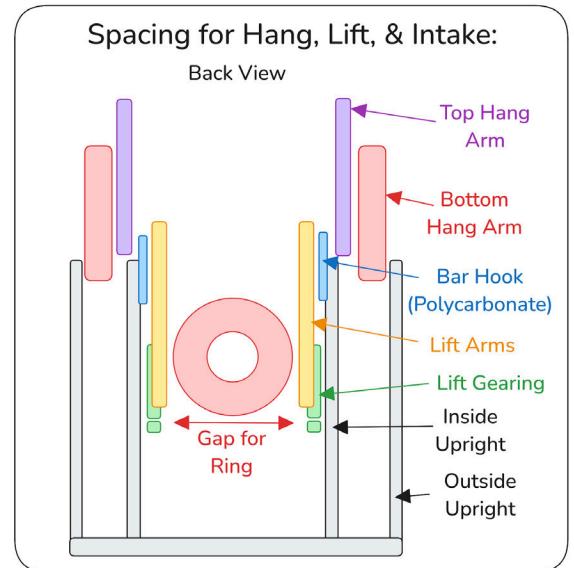
The main uprights on this robot will be mounted at an angle for several reasons:

- Built in Guides
 - The uprights will help guide the robot past the previous rung as the robot moves to the next tier
- Improved Packaging
 - By angling the uprights, the hang's hooks and arms will not stick out the back of the robot
- Vertical Expansion
 - This adjustment will cause the robot to tilt slightly upward when the rigid hooks engage with a rung. This tilt simplifies compliance with <SG3> during rung transitions because, even if the chassis rotates forward, it remains above the previous rung



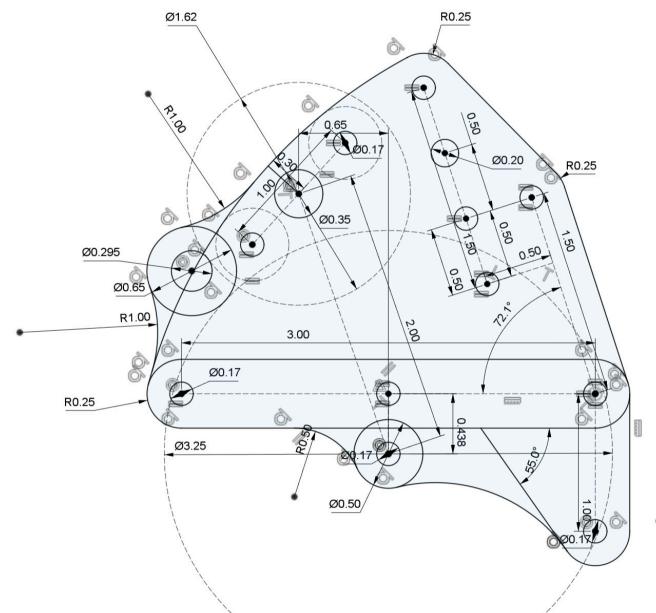
We will utilize two uprights on each side of the robot for the following reasons:

- Hang Integration
 - The bottom hang arm needs to be mounted outside of the inside upright on each side in order to fit with the lift. As this joint will have a lot of stress, it needs to be supported from both sides
 - The outer upright will provide a place for the vertical post of the hang structure to rest against
- Lift Gearing
 - Two uprights will allow us to cantilever the lift gears more effectively
 - Opens up more options for lift motor placement
- Mounting Options and Packaging/Structure
 - Increases space to mount different parts such as sensors
 - Protects key components
 - Durable impact point at the corner of our robot
 - Protects gears, PTO, and likely the brain and battery
 - Allows us to use lighter, weaker, components inside the robot
 - Significantly reduces risk of snagging or entanglement

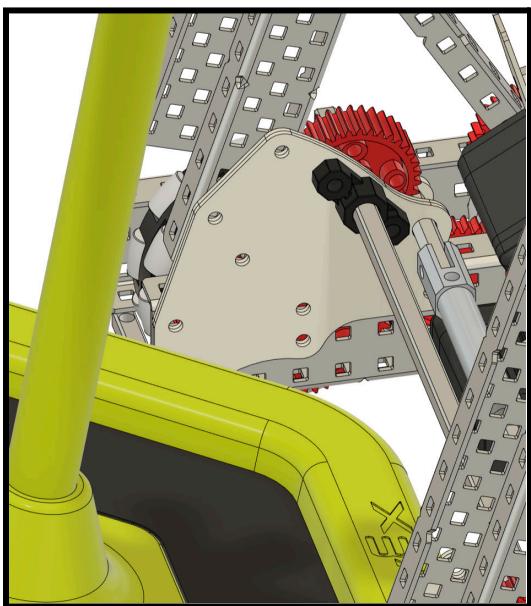


Polycarbonate Upright Mounts:

- Light and strong way to attach uprights
 - In addition to the uprights, these parts also...
 - Mount PTO HS shaft
 - Mount PTO piston
 - Provide a smoother edge for the mobile stake to center against
 - Help offset the inside of the back omni upwards
 - Allows for precise and intentional placement of all components



Progress Images:



11/05/24

Design: CAD Day 3 (C.3.2)

Designed by: Carl

Witnessed by: Alex

Witnessed on: 11/06/24

Goal: Find a working lift geometry and determine and implement the necessary gear ratio.

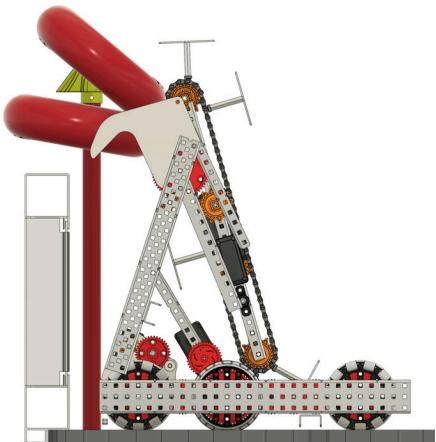
Lift Design Requirements:

- The intake must be positioned to allow scoring in ideal locations for all three types of stakes: mobile, alliance, and wall stakes. It should also be capable of scoring on mobile and alliance stakes while at its lowest position
- To prevent hooks from catching on previously scored rings, the intake must tilt significantly when scoring wall stakes
- The entire intake and lift assembly must fit below the lowest hang rung
- The lift should have minimal slop to ensure precise scoring. High-strength shafts are required at pivot points instead of standoffs
- The lift gearing must be adjustable and maintain torque at least equal to the previous design

Lift Geometry Tuning:

The lift geometry took a lot of trial and error to get working. This was largely due to the considerable effort that went into ensuring that the top intake roller was in the right position to properly score on alliance stakes AND getting as much intake tilt as possible for wall stakes. Getting one of these traits was easy, however, achieving both simultaneously proved very difficult. After further consideration, we elected to mount the top arm on the rung hook for the hang, allowing for extremely small adjustments in the geometry to hone in on the optimal configuration. Additionally, mounting the arm on this polycarbonate was very low profile and should help when adding the hang.

Scoring on Alliance Stake



Scoring on Mobile Stake



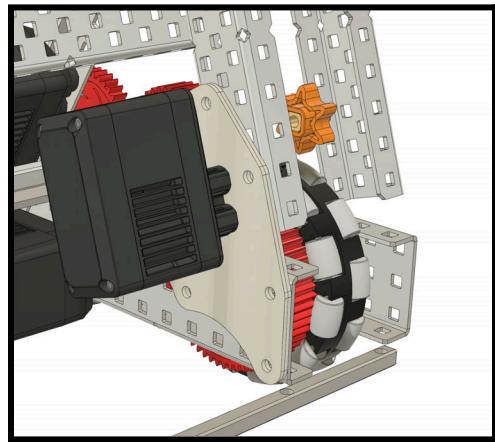
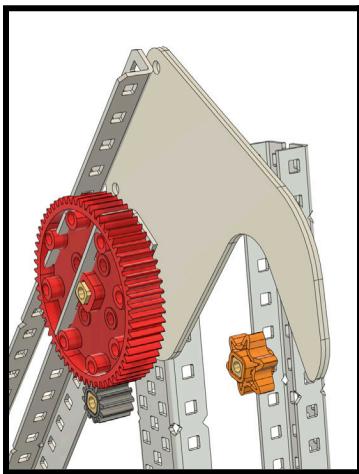
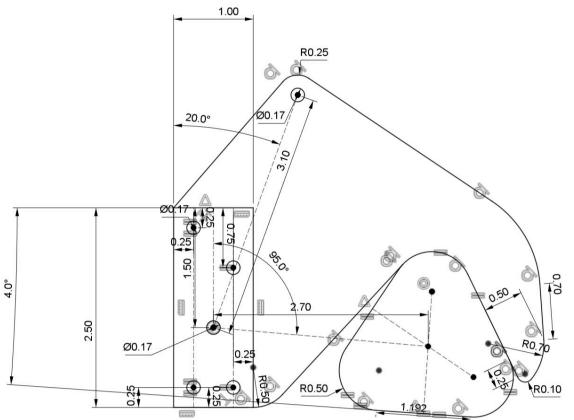
Scoring on Wall Stake



Lift Gearing:

Our previous lift utilized a gear ratio of 1:9 with a single 5.5W 200 RPM motor. That means with the two 5.5W motors we have allocated to the lift, we would need a minimum ratio of 1:4%. Even with the use of compound gear ratios, this reduction would be very difficult to achieve as it is not a round number. Furthermore, using a compound gear ratio would make adjusting the gear ratio incredibly difficult which is NOT acceptable with our untested hang concept.

A ratio of 1:5, however, is very easy to implement—only requiring a one stage gear reduction—and very easy to change to a 1:4, 1:6, or even a 1:7 ratio. Here is our implementation of this ratio:



Both the lift arms and the gears are cantilevered. Typically this is not good to do in high torque situations, however, because the gears are very close to the uprights this shouldn't cause problems. The lift motors are mounted as low as possible—just above the mobile stake—to lower the robot's CG. The power is transferred up to the lift gearing with chain and sprockets which will eventually be protected more to prevent them from incidental contact.

11/06/24

Design: CAD Day 4 (C.3.2)

Designed by: Carl

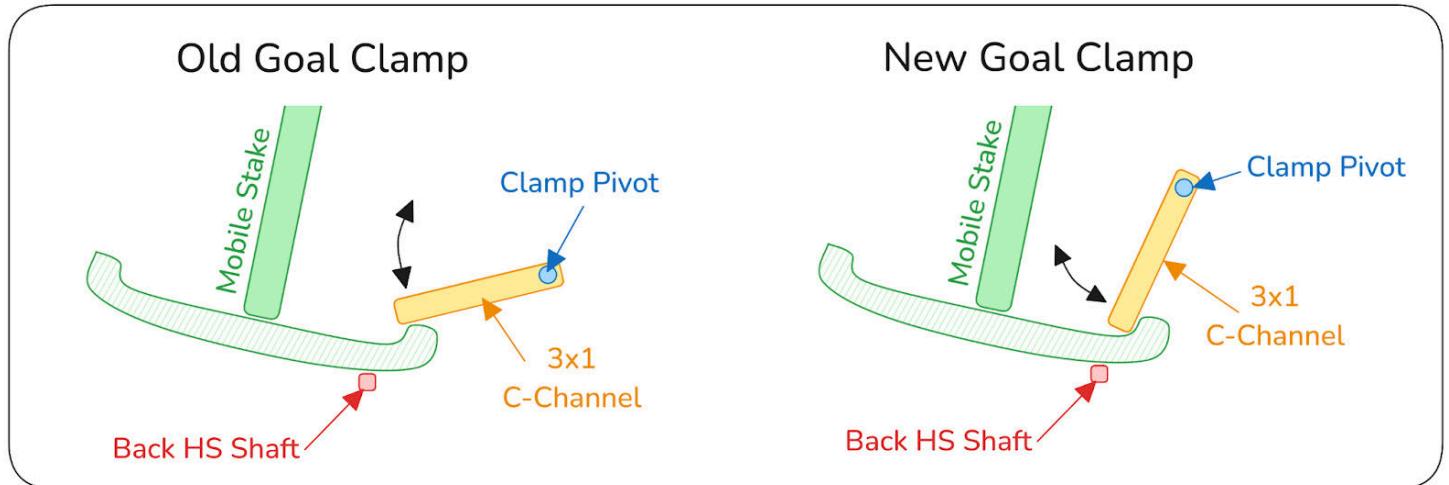
Witnessed by: Alex

Witnessed on: 11/07/24

Goal: Design a bottom-stage intake and integrate it into the main system, add a mobile stake clamp, and determine mounting positions for the brain, battery, and pneumatic reservoir.

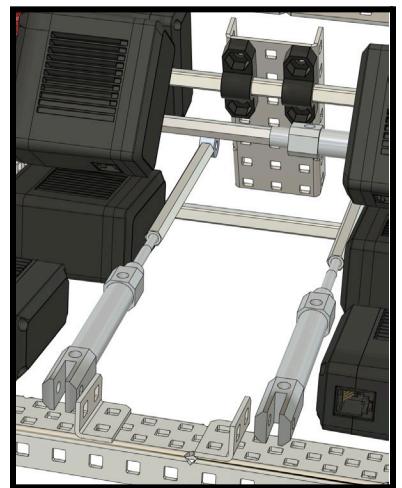
Back Clamp:

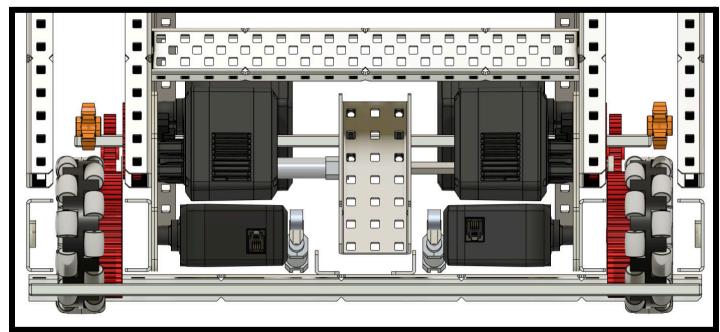
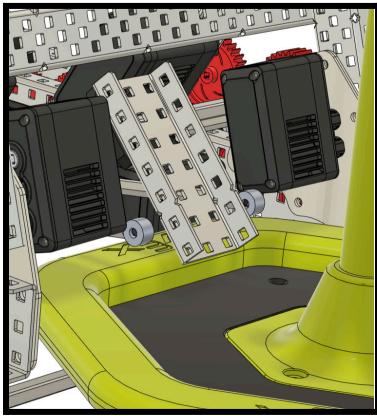
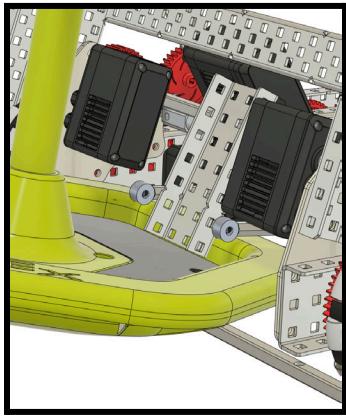
As mentioned in our [skills time distribution](#) (Pg. 249) and our [subsystem analysis for R.1.2.11](#) (Pg. 250-256), a clamp that can reliably and quickly grab goals at any orientation is extremely important. We think that we can achieve this characteristic by having a clamp that pulls the base of the goal towards the robot.



By changing the location of the pivot point, we can achieve more of a “pulling” motion as opposed to a “tilting” one, allowing the clamp to pull the goal into the robot which will hopefully increase the clamp's grabbing range to around an inch. This also almost entirely eliminates the need for an alignment mechanism as the pulling motion of the clamp should force the inside edge of the goal to flatten against the surface of the 3x1 clamp arm. In turn, eliminating the alignment mechanism will help prevent the goal from bouncing out of the clamp, further enhancing the clamp's grabbing ability.

To maximize mechanical advantage, the pistons are mounted as close to perpendicular to the clamp arm as possible. For the new clamp, this requires positioning the pistons behind the 3x1 C-channel. This placement also protects the pistons from impacts with other robots and contributes to a lower CG.



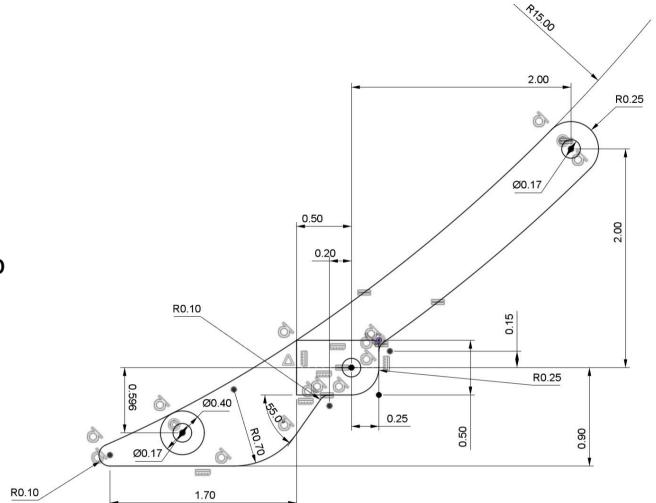
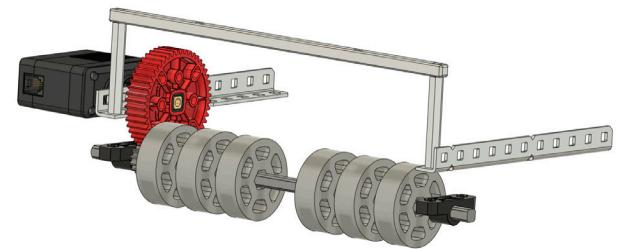


To reduce the weight and complexity of the clamp as much as possible, we integrated it into other mechanisms and pre-existing structures on the robot. Specifically, we extend the pistons so that they are attached to the front crossbar and pivoted the back clamp on the PTO shaft using HS pillow blocks (these will allow for fine tuning later). Additionally, for maximum air retention and weight savings, we only use two single acting 25mm stroke pistons.

Bottom Stage Intake:

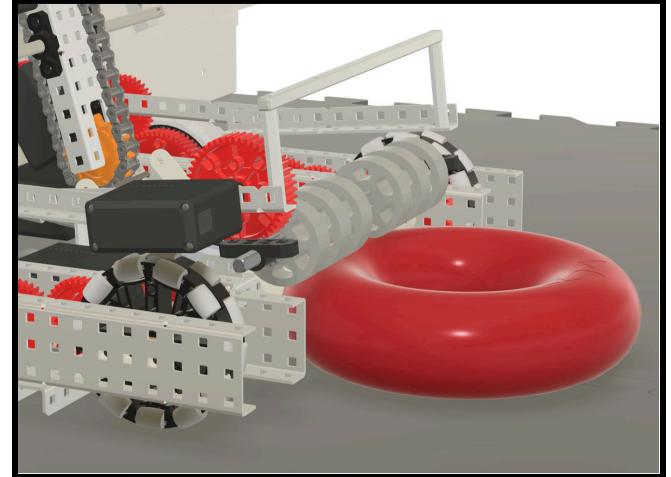
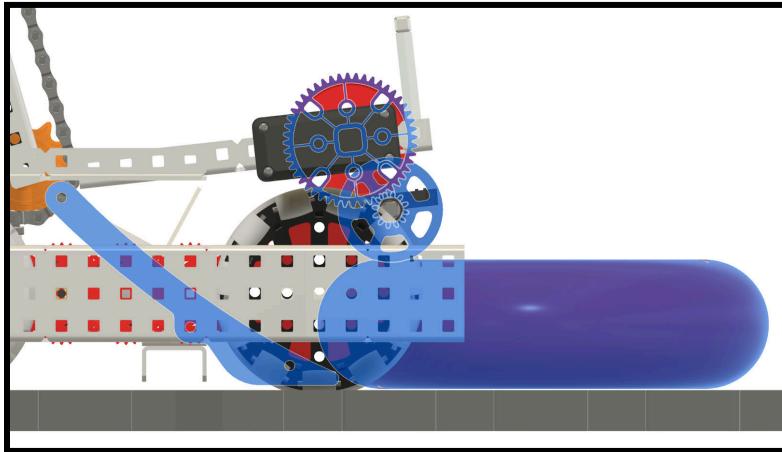
There are three big changes we wanted on this intake: Shallower intake ramp angle, intaking corner stacks, and lower power.

In order to intake effectively out of the corner, the intake needs to be small enough to get between the first (bottom) and second rings. To accomplish this, we mounted the motor on the side of the intake (above the drivetrain) to allow rings to go over the roller. The roller itself is made up of 1.625" flex wheels as they are the smallest size of flex wheel which should help the intake squeeze between rings. To account for the drop in linear speed created by the smaller wheel diameter, we increased the speed of the roller from 600 to 800 RPM with a 48:12 gear ratio. The roller is mounted on pillow blocks to allow for the correct spacing of this gear ratio. The HS shaft crossbar is mounted 2.1" above the roller, allowing the second ring to go over the roller as the robot drives into the corner stack.



The shallower intake ramps are made up of two doubled up polycarbonate profiles (idea credit to 229V ACE) that help to smoothly direct the rings upwards. The angle that the ring changes is also significantly reduced from the last robot, made possible by the [longer top stage](#) (Pg. 259-262).

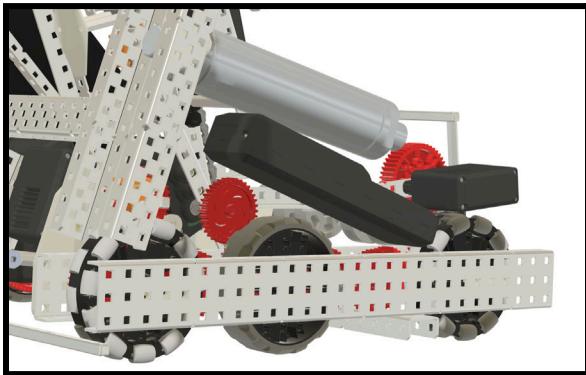
All of the structure is designed to be as light as possible and 1x1 L-channels are used whenever possible. High strength shafts are used at the high impact areas on the front and the 5.5W motor is contained fully within the footprint of the robot.



Brain, Battery, and Pneumatic Reservoir Mounts:

To lower the CG, all of these components need to be mounted as low as possible. Due to the nature of our intake/wall stake mechanism, all of these components needed to be mounted on the side of the robot above the drivetrain gearing. The brain is mounted on the left side of the robot using a 1x1 L-channel between the inner upright and the intake structure. This allows the brain to be fairly low and with minimal structure, while also allowing a side panel (will be added later) to cover most of the brain.

The pneumatics reservoir is mounted on top of the battery; both components are positioned as low as possible.



Note: The mounting positions for these two components are not finalized, and are just meant to show that we can mount them in a reasonable position.

11/07/24

Design: CAD Day 5 (C.3.2)

Designed by: Carl, Alex

Witnessed by: Alex

Witnessed on: 11/08/23

Goal: Design and complete the hang for this robot.**Hang Arm:****Bottom Hang Arm:**

The bottom hang arm is constructed from a 1x2x14 C-channel mounted sideways to reduce sideways movement as much as possible by allowing all of the pivots to be supported in two places. This will be extremely important on the pivot with the top arm as there is only around 0.25" between the top arm and the bar hooks.

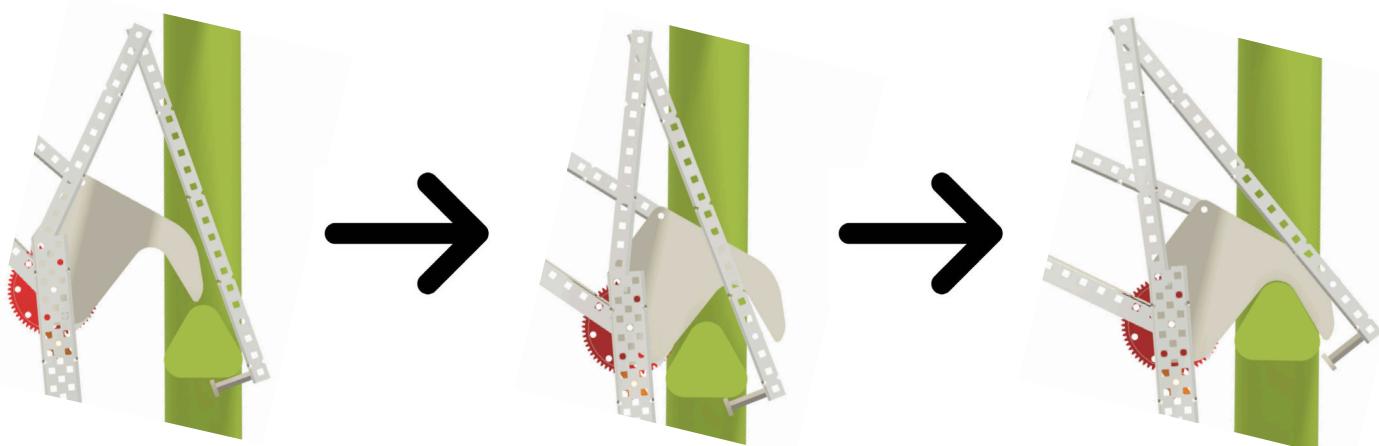
**Top Hang Arm:**

Because of the limited space available for the hang arm, we only had space for a 1x1 L-channel, however, it would be possible to switch this out for a cut in half 3x1 C-channel if necessary. At the end of the arm is a 0.75" standoff to grab the rung.

Geometry Considerations:

The length and pivot location of the lift arms were carefully tuned and tweaked so that the following are true:

- Hang arms are slightly above the next rung (the arms are as short as possible, increasing strength and lowering weight)
- When the bottom arm is lowered, the hang pulls the rung into the polycarbonate hooks
- The hang stays within the confines of <SG2> and <SG3>
- The hang does NOT interfere with any other subsystems



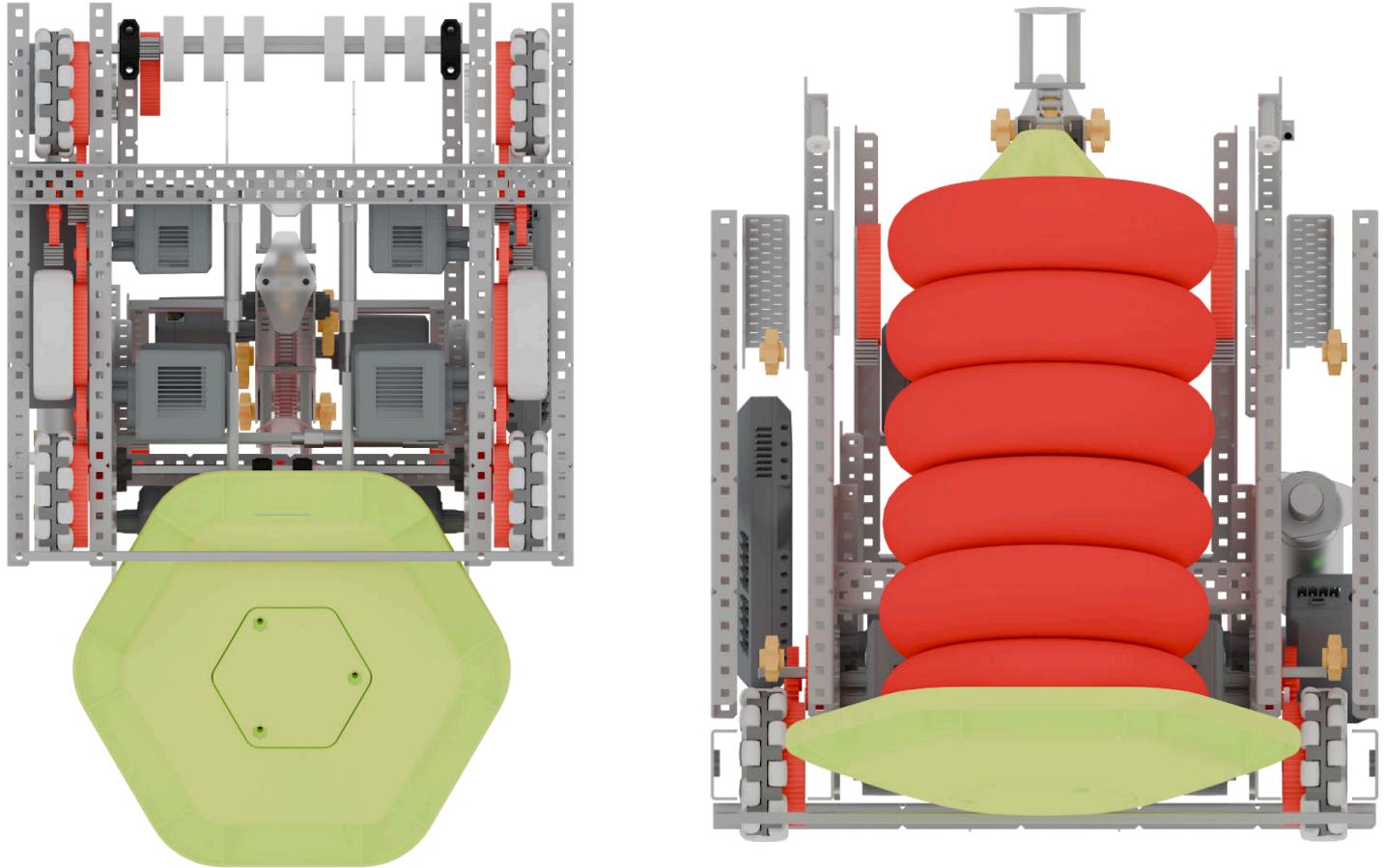
Lift Integration:

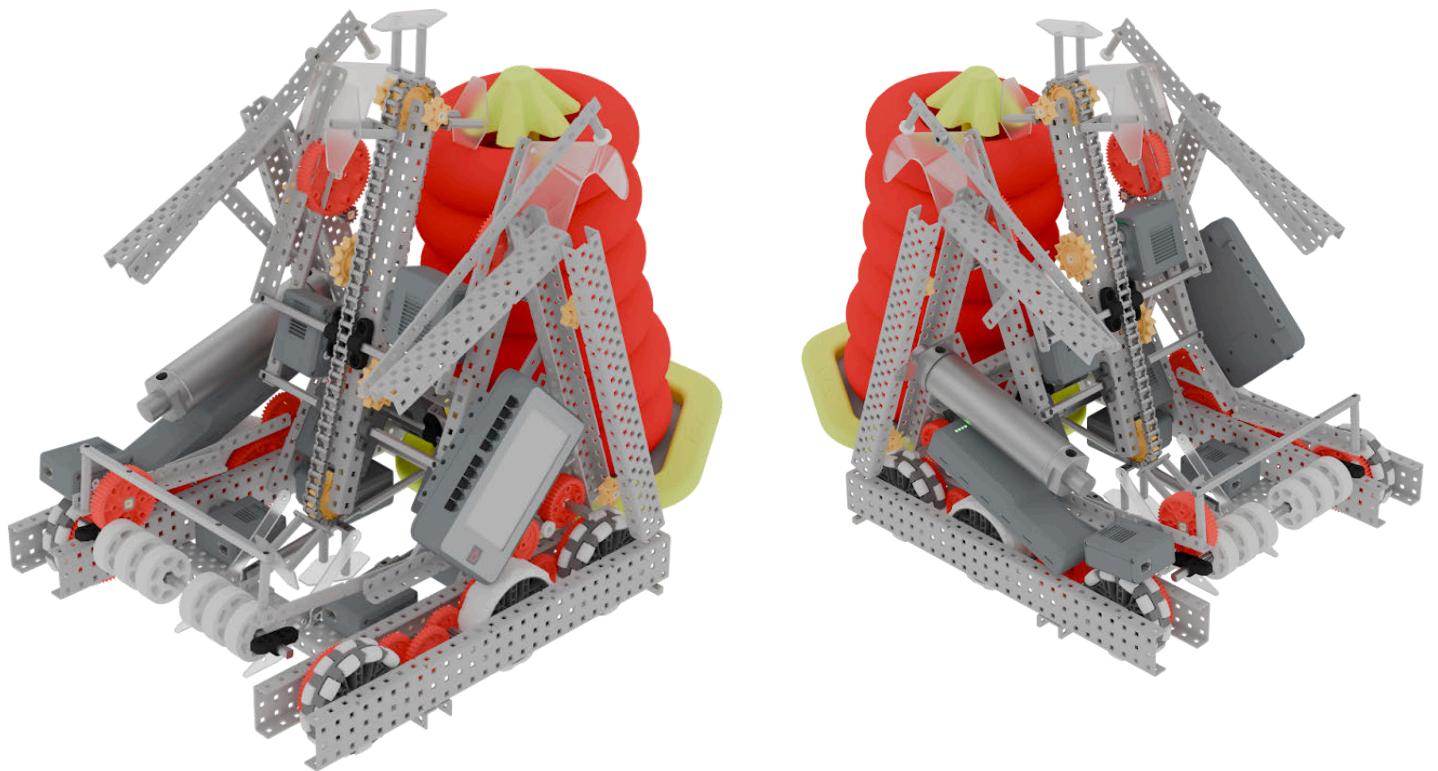
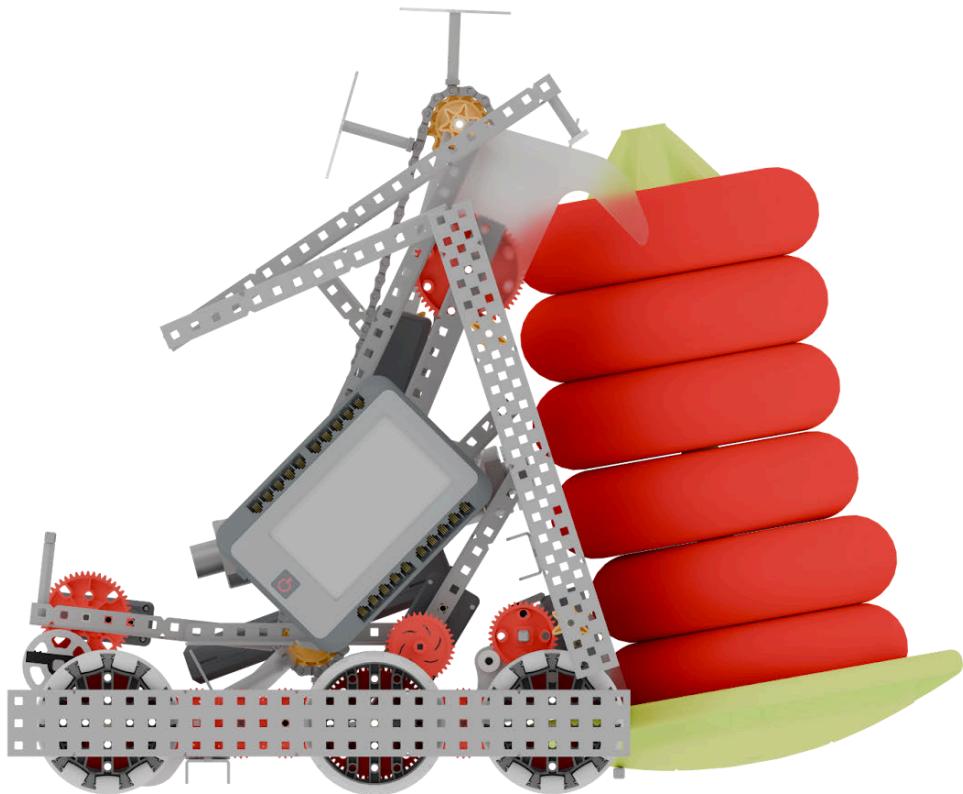
The lift system is not used during hanging and the hang is not used during matches. With this in mind, we will be using the lift to control the bottom hang arm, providing the following advantages:

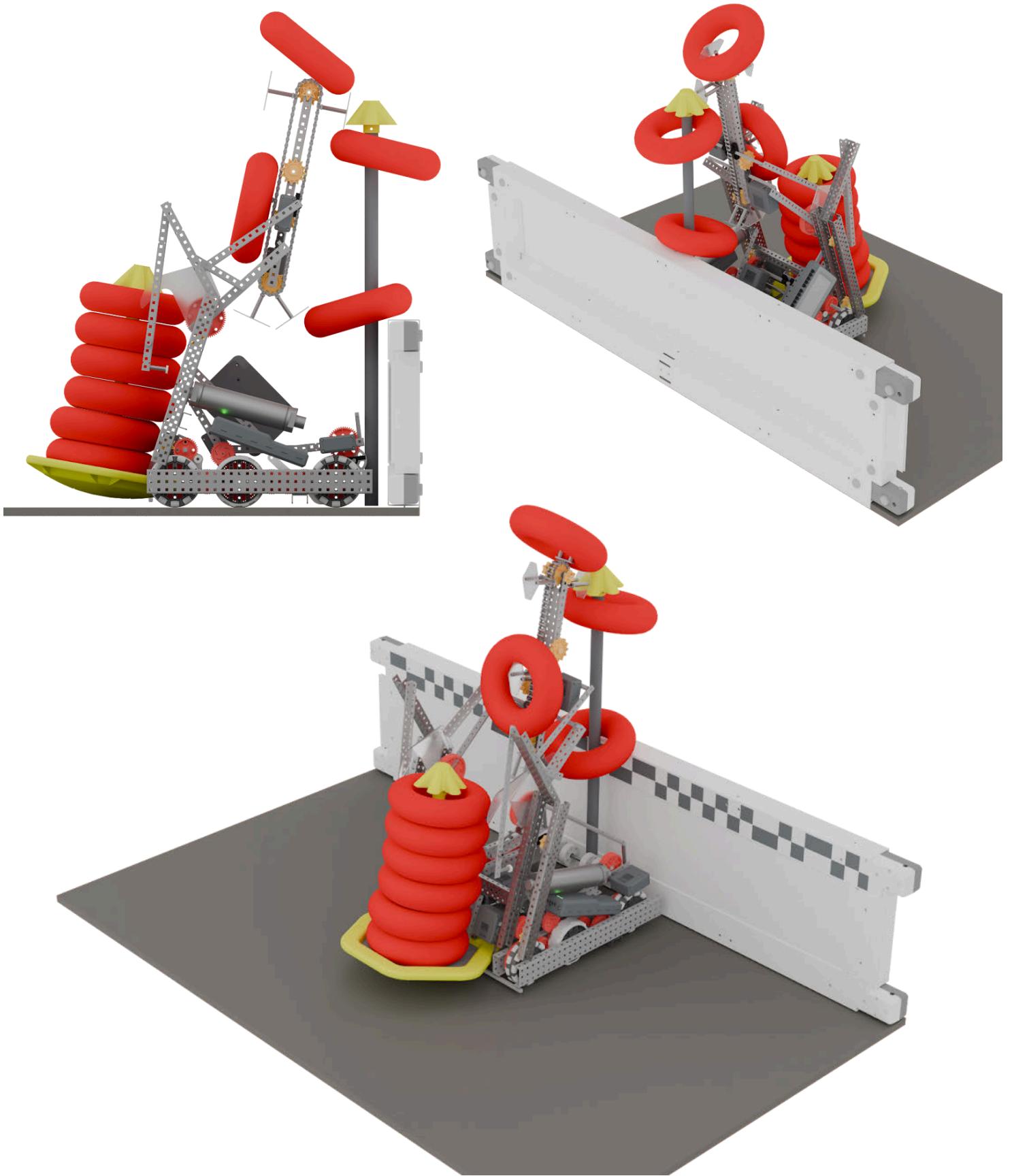
- Less Weight
 - A small linkage is much lighter than 2 or more additional pistons and an additional air reservoir
 - No additional structure for pistons
- Fine Control & Easy Tuning
 - The lift is almost infinitely adjustable, allowing for precise adjustments to the position of the bottom arm (pistons only have two positions that can be achieved)
- Speed
 - With the ability to achieve multiple lift positions, the arm should be able to gradually adjust to the right position as the robot climbs instead of after the robot is on a rung

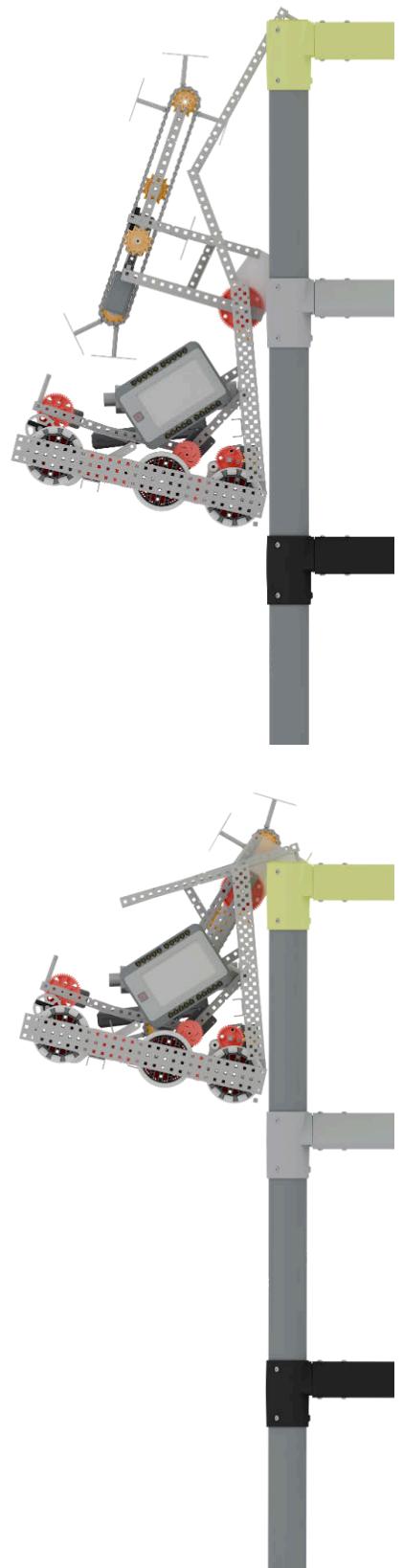
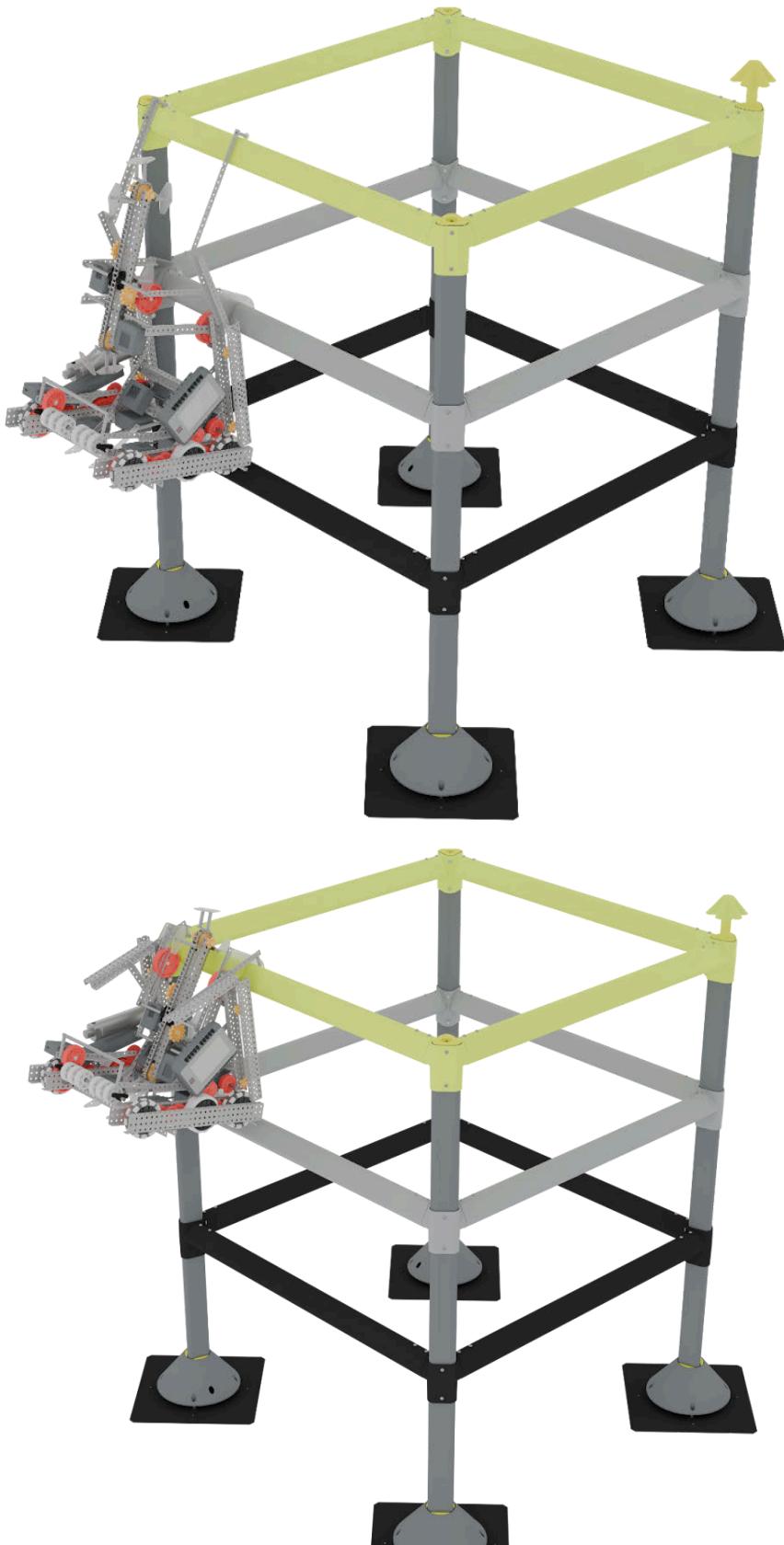
The bottom lift arm is connected to the bottom hang bar via a 1x1 L-channel serving as the linkage.

Full Robot Renders (Reference for Building):









11/08/24 Background Research: System Identification

Designed by: Alex

Witnessed by: Carl

Witnessed on: 11/08/24

Goal: Research what system identification is, how it works, and where we would want to use it.

What is System Identification?

According to [wpilib](#), system identification is “the process of determining a mathematical model for the behavior of a system through statistical analysis of its inputs and outputs.” This effectively means that you can record actual data from a system and use system identification to calculate the feedforward constants without manual tuning.



How Does System Identification Work?

We did our research by reading the [Controls engineering in the First Robotics Competition Book](#) by Tyler Veness. Essentially, System Identification is an optimization algorithm. This means that it is taking the testing data and system model input into the model and finding the best feedforward constants to fit the testing data it was given. Not only is this method to determine the feedforward constants more efficient for us time-wise, but it also means that it will likely calculate better feedforward values to fit the system.

Going into the technical aspects here is the math for the “System Model”. This is very similar to the models we discussed in our previous entry on [feedforward constants](#) (Pg. 65-66) where there is a more in-depth description of how this works.

$$a_k = \frac{v_k - v_{k-1}}{T}$$

$$u = K_s \text{sgn}(v_k) + K_v v_k + K_a a_k$$

where u is the motor input voltage, v is desired velocity, and a is the desired acceleration. We will focus on the optimization of the K_s , K_v , and K_a values for this system.

Next, we record data of the system's reaction (v) to various control inputs (u) and run an Ordinary Least Squares (OLS) algorithm to find the optimal feedforward values to match the feedforward data to real recorded data on the robot. Recording real data from the robot allows a better understanding of the underlying robot's dynamics and better account for the effects in future movements.

$$c = Bx$$

Where B is a matrix of $\text{sgn}(v)$, v , and a over time and c is the control inputs (u) for all times. Formatting the recorded data like this allows for the use of OLS algorithms to find theoretically optimal values of x (The feedforward dynamics underlying the system). We then use the OLS algorithm to find the most optimal x (feedforward) values for the given data

$$\begin{bmatrix} u_0 \\ \vdots \\ u_{k-1} \end{bmatrix} \approx \begin{bmatrix} v_1 & a_1 & \text{sgn}(v_1) \\ \vdots & \vdots & \vdots \\ v_k & a_k & \text{sgn}(v_k) \end{bmatrix} \begin{bmatrix} K_v \\ K_a \\ K_s \end{bmatrix}$$

We then run the OLS regression on this equation to find the best feedforward ($\begin{bmatrix} K_v \\ K_a \\ K_s \end{bmatrix}$) vector given the inputs from the drivetrain in the B matrix and c vector.

```
void characterize(const std::vector<double>& x, const std::vector<double>& u) {

    // Allocate large enough matrices for linear regression
    Eigen::MatrixXd A = Eigen::MatrixXd::Zero(x.size() - 1, 3);
    Eigen::VectorXd c = Eigen::VectorXd::Zero(x.size() - 1);

    // Fill the matrix with recorded inputs and states
    for (int i = 1; i < x.size(); i++) {
        A(i-1, 0) = x[i];
        A(i-1, 1) = (x[i] - x[i-1]) * 100.0;
        A(i-1, 2) = signnum(x[i]);
        c(i-1) = u[i - 1];
    }

    // Compute the least-squares solution with SVD, which provides the most stability and
    // accuracy of the least squares methods
    Eigen::VectorXd b = A.bdcSvd<Eigen::ComputeThinU | Eigen::ComputeThinV>().solve(b);

    // Set the feedforward to the linear system solution
    ff = b;
}
```

Utilization

Below is an example of how we utilize the *OneDofVelocitySystem* in code characterizing a single motor group:

```
OneDofVelocitySystem sys;

std::vector<double> xRecorded;
std::vector<double> uRecorded;

while (master.get_digital(DIGITAL_A)) {
    double u = master.get_analog(ANALOG_RIGHT_X) / 127.0;
    motorGroup.move_voltage(12000.0 * u);
    double x_k = (left_mg.get_actual_velocity() - right_mg.get_actual_velocity()) / 200.0;

    uRecorded.emplace_back(u);
    xRecorded.emplace_back(x_k);
    pros::delay(10); // Run for 20 ms then update
}

sys.characterize(xRecorded, uRecorded);
auto ff = sys.getFF();

while (true) {
    const double velocity = master.get_analog(ANALOG_RIGHT_X) * maxVelocity / 127.0;
    acceleration = (velocity - lastInput) / 0.01;
    double voltage = sys.evaluate(Eigen::Vector3d(velocity, acceleration, signnum(velocity))) * 12000;
    motorGroup.move_voltage(voltage);

    lastInput = velocity;
    pros::delay(10); // Run for 10 ms then update
}
```

Conclusion

- System Identification allows the calculation of feedforward values
 - Uses real data from the robot so it can account for issues such as increased friction under certain field materials
- Uses OLS
 - Finds the most optimal solution given the data
- Machine learning
 - Because this approach uses recorded data to “learn” better feedforward values this approach can be described as a machine learning technique
- Next steps
 - Design an algorithm to do system identification on specific subsystem such as the drivetrain or lift

11/09/24 Build: Day 1 (R.2.1.1)

Designed by: Carl

Witnessed by: Alex

Witnessed on: 11/10/24

Goal: Begin assembly of drivetrain structure and add the inner uprights.

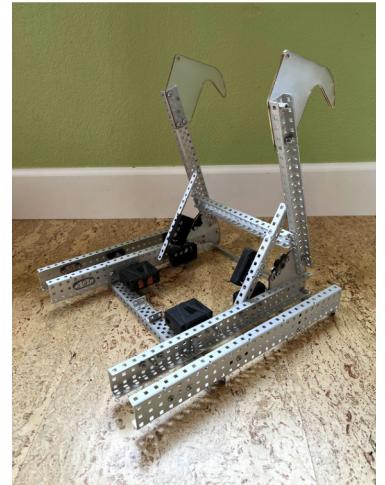
Build Techniques:

- Boxing
 - This can be seen on the inside drive rail above the front crossbar. This standoff will allow us to box the intake support into the structure, helping distribute the force of impacts
- Shoulder Screws
 - Makes squaring the robot easier
 - Allows us to not use bearing flats on screw joints
 - Saves weight, easier and faster to build
- Squaring the Frame
 - This goes without saying, but taking time to thoroughly square the frame of the robot using several different methods such as using squaring tools and measuring diagonals is critical

Overview:

Overall, our first day of building was very straightforward with no deviations from the CAD with the exception of hardware installation. We were able to reuse the HS shaft crossbar from the previous robot along with numerous other aluminum parts, saving time and materials.

Progress Images:



11/10/24 Build: Day 2 (R.2.1.2)

Designed by: Carl

Witnessed by: Alex

Witnessed on: 11/11/24

Goal: Assemble the top stage of the intake and begin work on the back clamp.

Top Stage Intake:

As we designed the top stage intake with a 3x1 C-channel, we needed to cut out slots in each end. This was done using a bandsaw to remove the bulk of the material and a file with a square edge to finish the part.

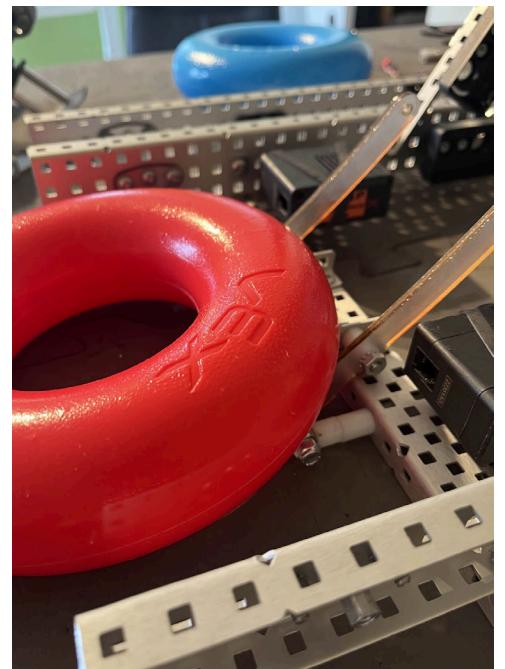
The bottom sprocket pivots on a screw joint and the top sprocket is rigidly connected to the top shaft, allowing the flex wheels to rotate in sync with the intake.

The amount of links in the main chain was not divisible by four, so two of the gaps between hooks are slightly smaller than the others. We only used the older variant of HS chain links as we found them to be noticeably stronger than the newer links.



Bottom Intake Spikes:

These polycarbonate parts were added in and spaced out from the 1x1 L-channels to be in the correct positions. It was immediately apparent that they were significantly better than the previous intakes ramp.



Back Clamp:

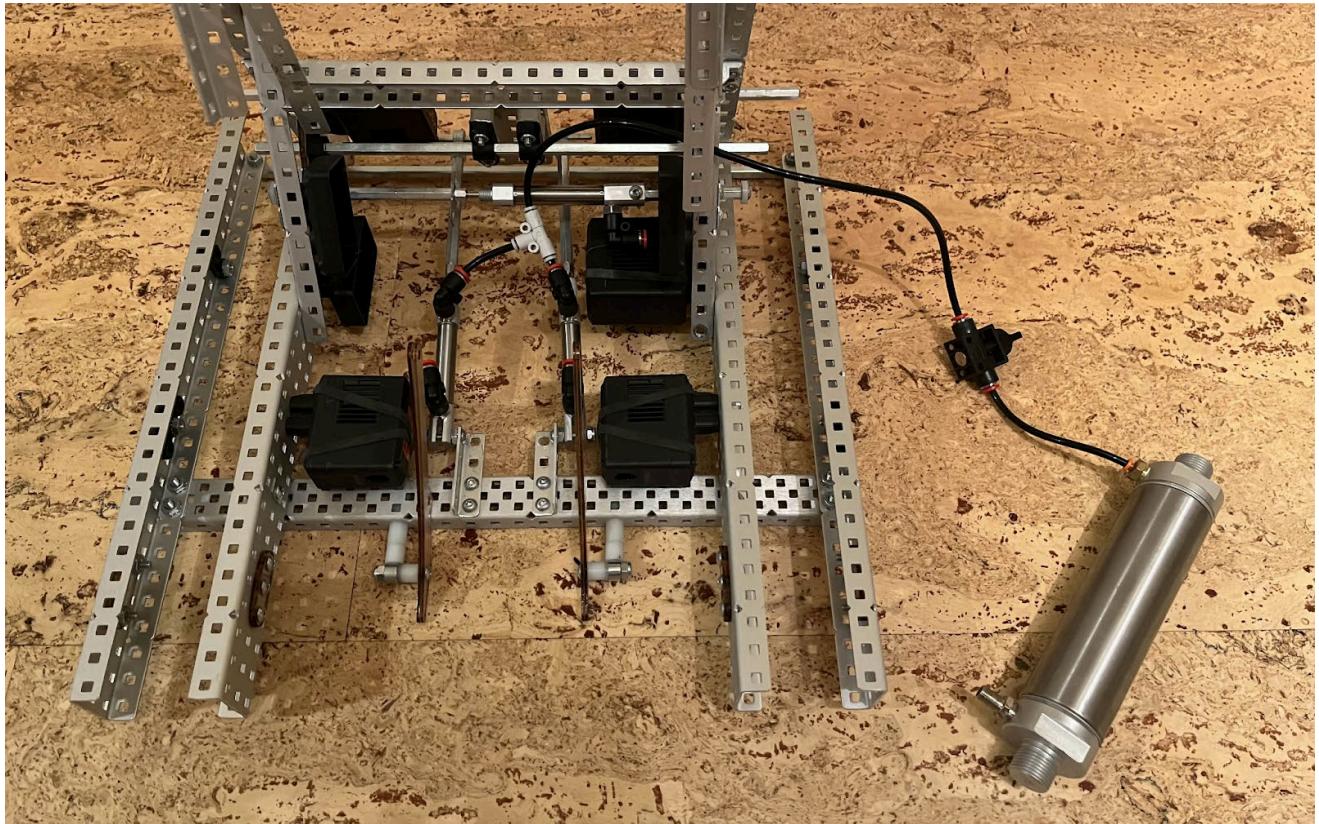
The back clamp was built exactly to the specifications of the CAD, however, even at 100 PSI, the pistons did not have enough force to hold the goal. We think this is because the clamp pushing on the inside edge of the goal has much less leverage than the old clamp that pushed on the outside edge. Keeping the clamp at its current pivot point is very important for the reasons discussed in [CAD Day 4](#) (Pg. 270-272). Here are some of the prototypes of a clamp that pushes on the outside edge of the goal that we created, followed by the final version:

*left: prototype 1
middle: prototype 2
right: final version*



After 2 prototypes, we found that spacing out a HS shaft collar with a 0.5" spacer was a very simple addition that allowed the clamp to grab a goal properly, even at around 35 PSI.

Progress Image/Clamp Testing Setup:



11/11/24 Build: Day 3 (R.2.1.2)

Designed by: Carl

Witnessed by: Alex

Witnessed on: 11/12/24

Goal: Build the lift, mount the top stage intake, and build/install the bottom stage intake. Test both stages together.

4-Bar/Top Stage Mounting:

The 4-bar was built to the CAD with only the following changes:

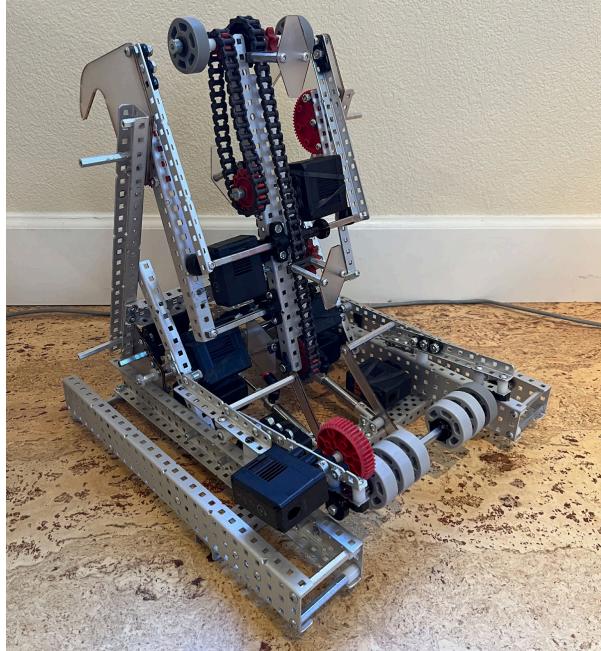
- Addition of hardware such as screws and nuts
- Addition of hardstops to prevent the lift from going too low and allowing the lift motor to rest
 - The hardstops are rubber to absorb impact from fast lift movements
- Cutting the top off of the 60T gears to allow them to be mounted to the bottom lift arms

The bottom arms are made up of a 3x1 C-channel cut in half using a bandsaw and the HS shafts for the intake pivots were drilled out on a drill press.

Bottom Stage Intake:

Again, no major CAD deviations except the addition of multi directional hard stops—limiting up and down movement—on the back of the 1x1's.

Progress Photo:



Preliminary Testing Data:

Trial #	Rings Scored (on goal)
1	6/6
2	6/6
3	6/6
4	6/6
5	6/6
<i>Average Accuracy</i>	100%

Not a single miss after filling up five mobile stakes!

11/12/24 Build: Day 4 (R.2.1.2) / CAD (C.3.2)

Designed by: Carl

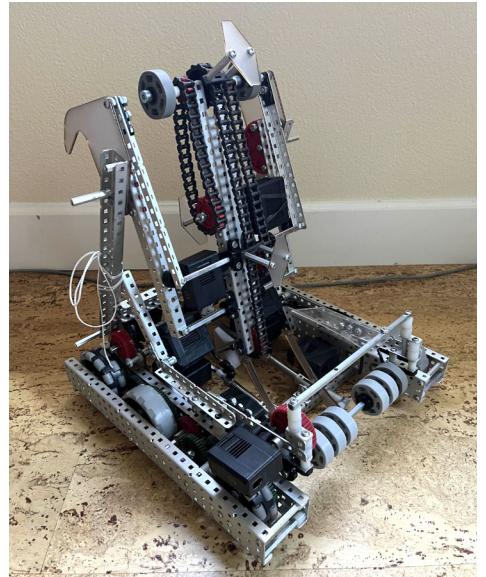
Witnessed by: Alex

Witnessed on: 11/13/24

Goal: Gear the drivetrain, CAD polycarbonate side panels to allow us to mount the outer uprights, and mount the electronics and pneumatic reservoir.

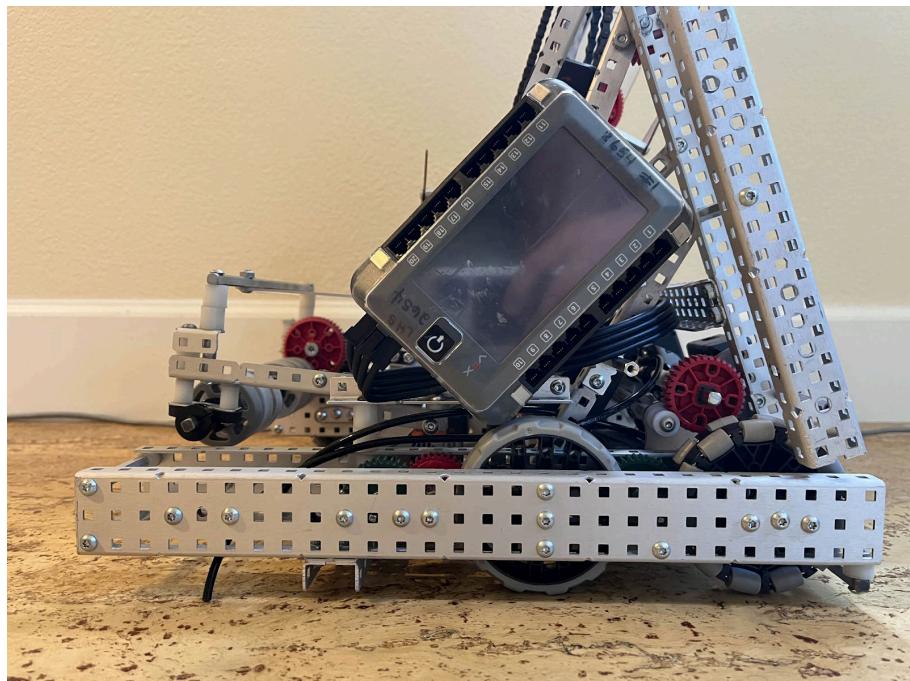
Drivetrain Gearing:

- No considerable changes from the CAD
- Beveled 60T gears to allow the drive gears to fit with the wheels
- Screw joints & screwing gears directly to wheels
 - This helps reduce slop, friction, and strengthens the chassis
 - Less motor burnout, higher accuracy, less deterioration over time
- Independent friction testing on each pivot/axle to increase performance
- “Hot Swap” motors (motor screws replaced with a rubber band) allow for easy switching of motors in the event that one fails



Mounting the Brain, Battery, and Pneumatic Reservoir:

Again, almost exactly the same as in the CAD. The only minor change was the addition of some very light structure to hold the front of the battery and reservoir:

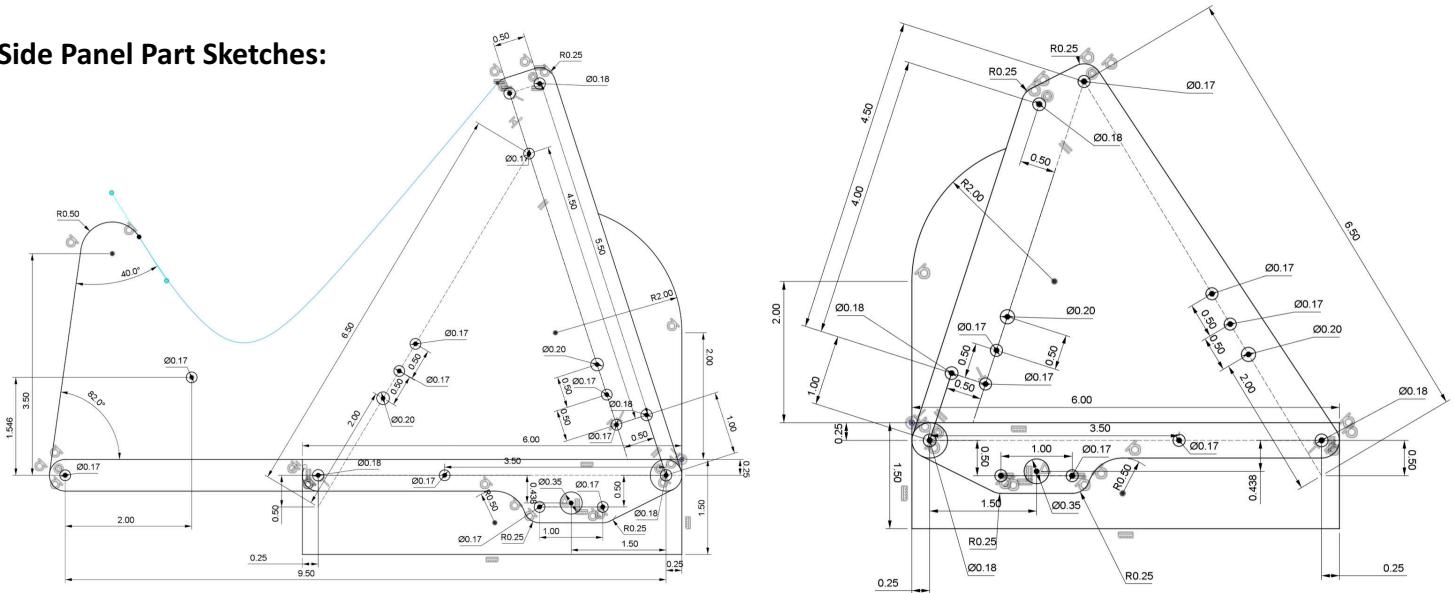


Side Panel Design/CAD:

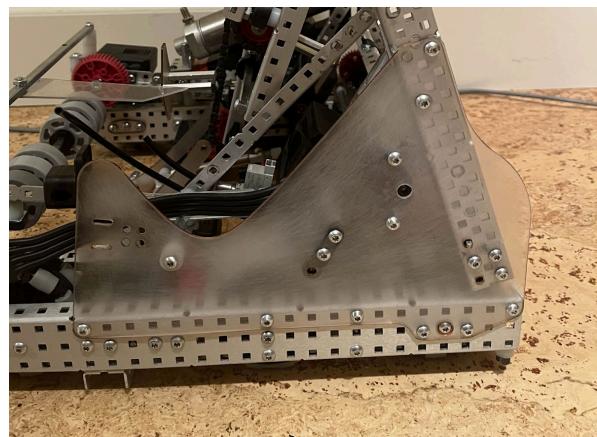
Functions of the Side Panels:

- Mount Outside Uprights
 - Attach the outside uprights to the drivetrain rails in a strong manner
 - Shaft Support
 - Support the second side of the center drivetrain motors shaft
 - Integrating into the side panel saves significant weight and simplifies the design
 - Protect Key Components
 - Having the polycarbonate on the side allows us to protect the brains wires and battery cable
 - The poly also protects the PTO and drive gearing from other robots
 - The side of our robot is much smoother with these parts allowing us to slide past other robots
 - Mount Sensors
 - Having the distance sensors for localization embedded in the side panels is a very light way to mount them while also protecting them from damage

Side Panel Part Sketches:



Implementation on CAD Model and Robot:



11/12/24 Build: Hang String Selection

Designed by: Alex

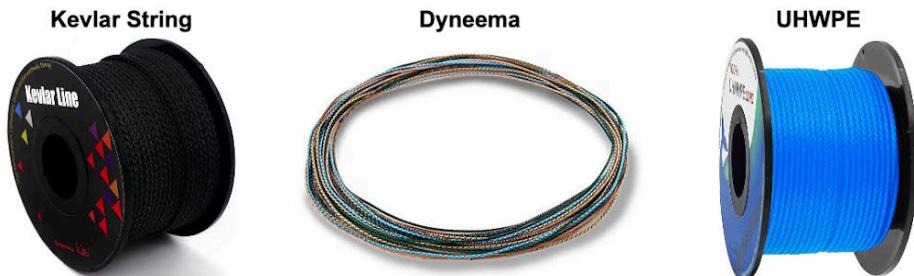
Witnessed by: Carl

Witnessed on: 11/12/24

Goal: Analyze various different options for string with a design matrix.

In order to effectively utilize the resources we have we think that it is important to carefully consider the selection of our hang string to work best with our design while still staying within our team budget.

The string we use for the hang must be carefully chosen so that it would never break, but still be as small and durable as possible. We found several high-performance strings from fiber-based Kevlar, or plastic based Dyneema or UHMWPE. To find the best solution to order we used a decision matrix to find the most optimal solution for our use case:



Criteria:	Weight	Score	Weighted Score	Score	Weighted Score	Score	Weighted Score
Strength	2	2	4	3	6	4	8
Size (Smaller is Better)	3	10	30	6	18	3	9
Flexibility	3	8	24	6	18	3	9
Abrasion Resistance	3	5	15	9	27	10	30
Price (Lower is Better)	2	5	10	4	8	3	6
Total Score:			83		77		62

Decision

With the kevlar string scoring the highest, we ordered that and it should be ready for us to use by 11/14/24. This goes along with our timeline to have the robot done perfectly on time.

11/13/24 Build: Day 5 (R.2.1.2)

Designed by: Carl, Alex

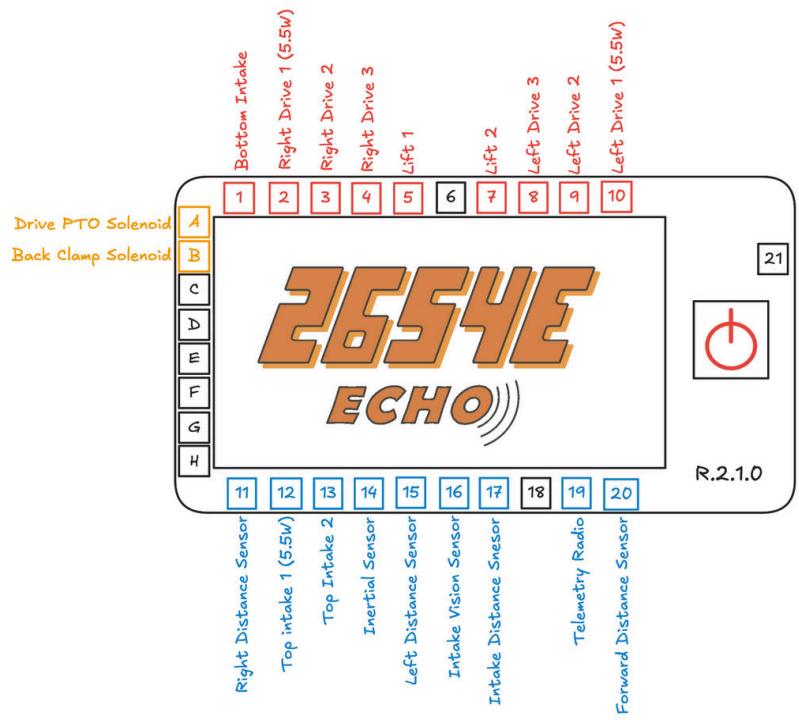
Witnessed by: Matt

Witnessed on: 11/14/24

Goal: Prepare the robot for testing for when Carl is out of town.

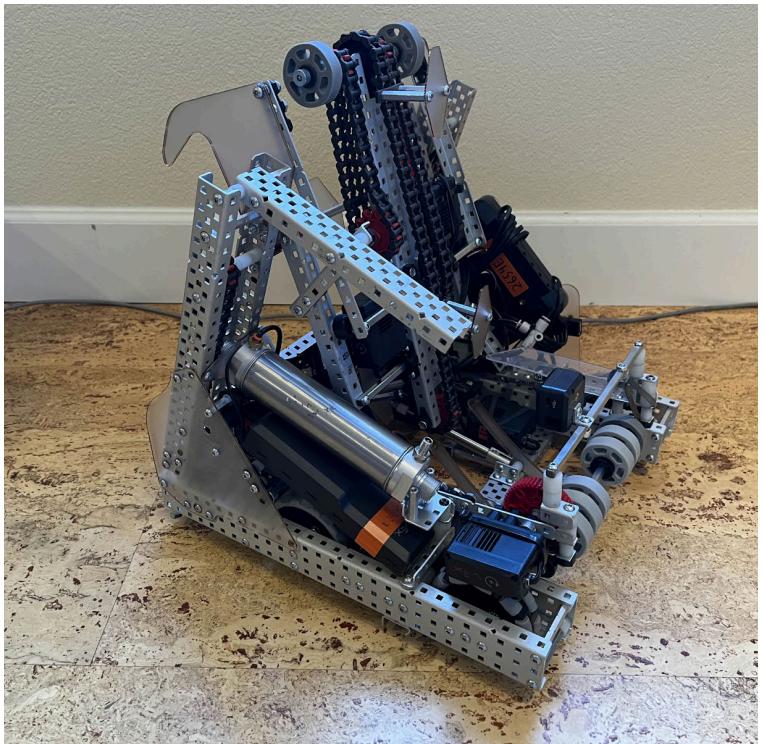
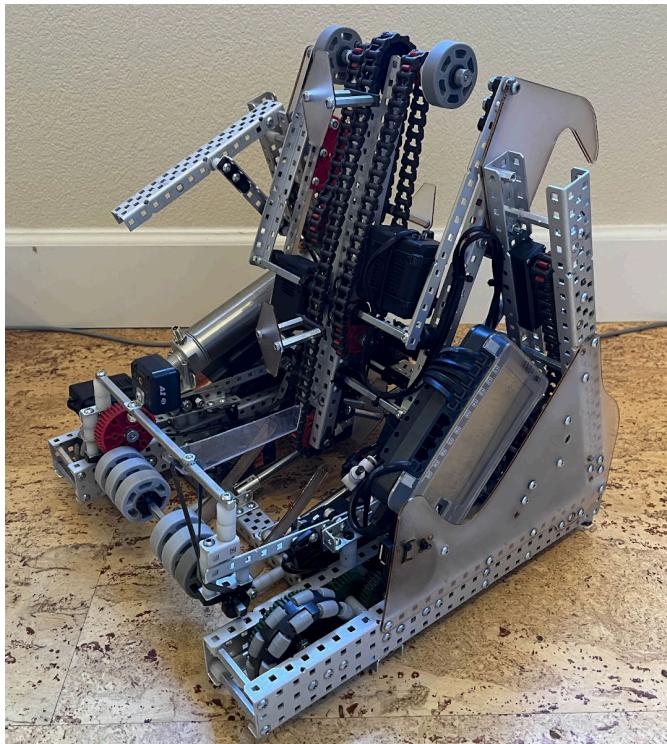
Wiring Diagram:

All of our wires are fully contained and pinned back away from moving parts; we have mitigated the risk of wires getting snagged or coming unplugged as much as possible. To keep track of the brain ports in an organized fashion, we created the diagram to the right.



Other Progress:

Just final touches such as finishing the drive gearing, mounting the inertial sensor (on rubber links), intake sensors, and beginning to build the hang, again, not deviating from the CAD.



11/13/24 Analysis/Design: Skills Pathing

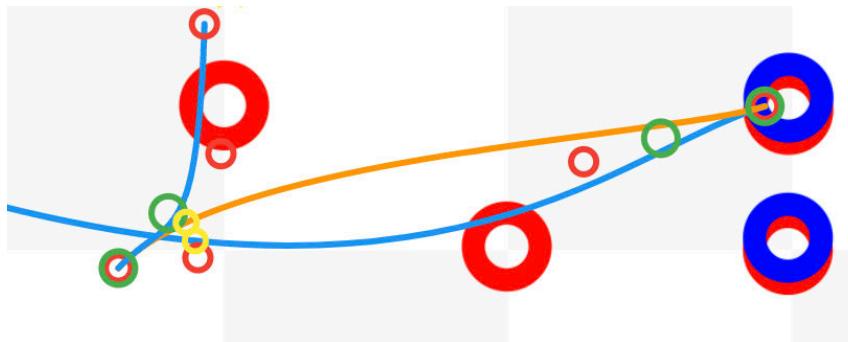
Designed by: Alex

Witnessed by: Carl

Witnessed on: 11/13/24

Goal: Explore ways to increase the efficiency of our skills routine to allow for enough time to hang

Currently, our path is very close to time (within 2 seconds), however to have enough time to get our tier 3 hang, we estimate we will have to have 12 seconds extra. We started this time cutting by improving the speed of our motions with [faster 2d motion profiling](#) (Pg. 236). This gave us 5 seconds to hang, which is still below our goal of 12 seconds. Overall, we thought the end of our current path is almost perfectly optimal as it gets all the rings and is almost never stationary. However, we think there is significant potential for optimization right here where the robot doubles back on itself.



This motion currently adds 2.8 seconds to the run by doing this double back movement. This double-back is avoidable if we instead get the non-stacked red ring in the middle of the picture when filling the first mobile stake. After going for the middle ring we would have intaked 6 rings on the goal and one for wall stakes, allowing us to score one of those on the wall stake immediately after dropping the first mobile stake in the corner.



These changes alone ended up saving **4 seconds** from the run. However, over the rest of the run, we haven't found more single significant repathing changes we can make to improve time for the rest of the path, but we can anticipate with our new intake improvements that we will be able to intake at a 25% higher speed, which gives us just enough time to complete the route with hang (12 seconds for hang in path simulation).

11/18/24 Build: Day 6 (R.2.1.6)

Designed by: Carl, Matt

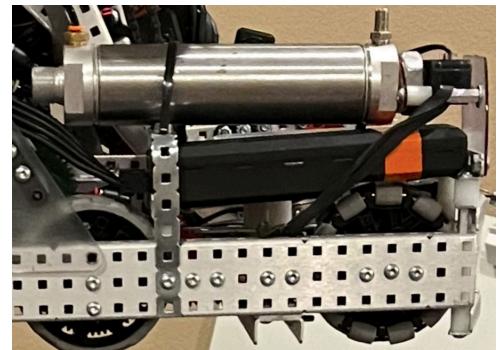
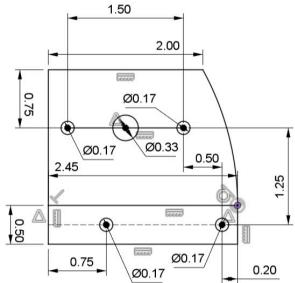
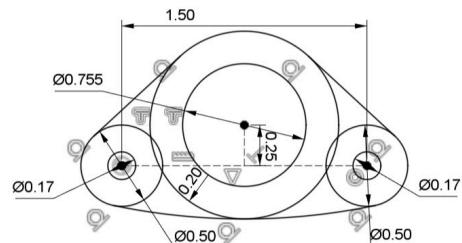
Witnessed by: Alex

Witnessed on: 11/18/24

Goal: Finish the hang, move the robot's CG further forward, and add a wall stake aligner.

Battery and Reservoir Re-mounting:

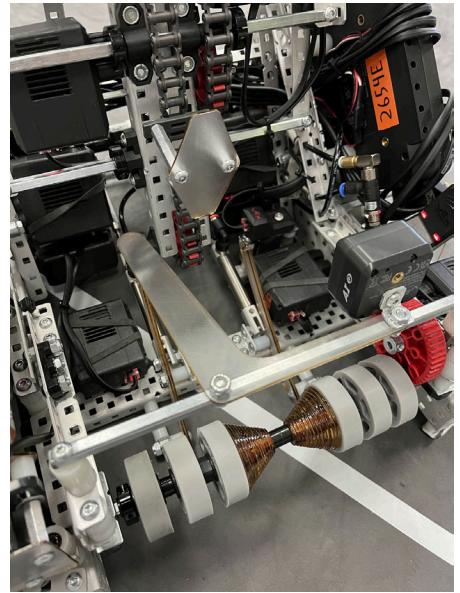
After very briefly driving the robot on a field, especially while holding a mobile stake, it was apparent that the robot's center of gravity was too far back, causing the robot to tip backwards when accelerating. To fix this, we can try to move the center of the gravity forward, and as the battery and reservoir are both relatively heavy parts, we chose to move those. In order to move these forwards, we also needed to move the intake motor to the left side of the robot to open up space. These are the polycarbonate mounts we used to attach the reservoir and distance sensor along with images of the completed mounting.



By doing this, we also lowered our center of gravity, created a more rigid battery holder, and added a mount for our front distance sensor.

Ring Compression:

Similarly to the [last intake](#) (Pg. 162), the rings occasionally get kicked out of the front by the hooks. This was solved by adding a compliant arm to push the rings towards the top stage, however, we made this one out of polycarbonate to reduce the weight and complexity, while also being much more consistent with its pushing force.

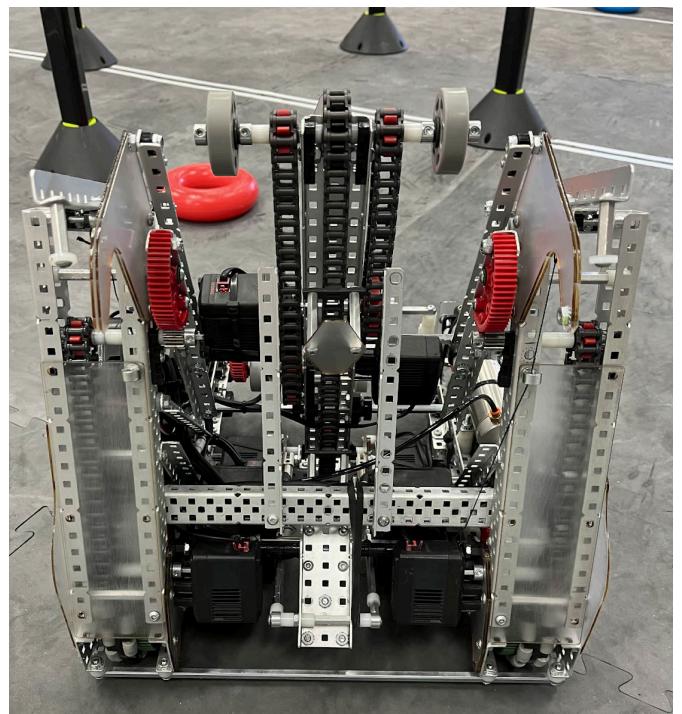


Wall Stake Aligner:

To easily score on the wall stake, we needed to add an aligner. To achieve this, we added a series of disks onto the intake's shaft. This was necessary because our current design doesn't allow us to place it on the existing HS shaft above as it would shift backward when attempting to score. To save time, we first 3D-printed the funnels to test their functionality and ensure they would work. We then laser cut the discs out of polycarbonate. One small caveat of this design is the amount of polycarbonate it requires.

Finishing the Hang:

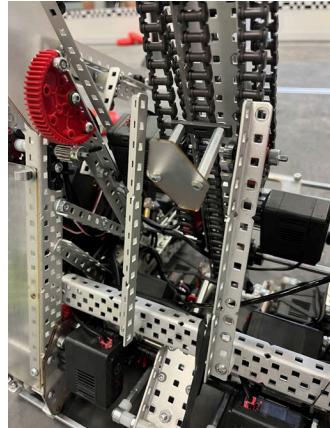
To complete the hang, we started by adding the second hang arm routing the string from the winch to each arm. This was achieved using a combination of standoffs and shaft collars, as shown in the picture. The setup helped guide the string from the winch at the center of the robot to power each side of the hang. We used a figure-eight knot between the HS shaft collar and the shaft to connect the string at the bottom and a bowline at the top to attach it to a shaft collar on each arm, allowing us to easily adjust the position when needed. We chose these knots because they remain secure even under stress.



Other Changes:

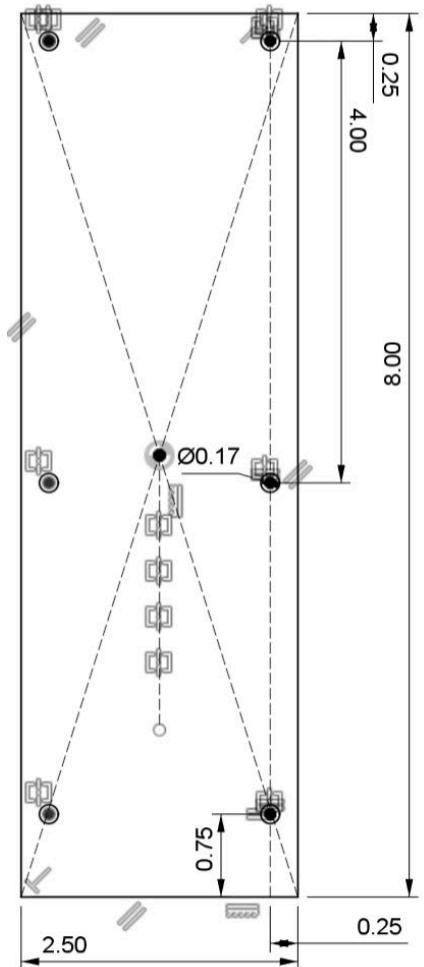
Polycarbonate Upright Backing:

Seen on the back of all the uprights, these rectangular polycarbonate parts will help the robot slide up the vertical post when hanging, but more importantly they protect the lift's chain.



Ring Offset Bars:

These 1x1 L-channels push rings away from the hooks so the hooks don't catch on rings that are already scored.



Inverted Funnels:

Originally, the front of the drive rails were connected to each other with 2.5" standoffs. Although being light and protecting the intake, the "extra" hole on the outside drive rail prevented the robot from going into the corner far enough to intake rings. This was improved by shortening the outer 3x1 rails by one hole and replacing the standoffs with angled 1x1x5 L-channels.

11/19/24 Testing/Identify: Hang (R.2.1.6)

Designed by: Alex

Witnessed by: Carl

Witnessed on: 11/19/24

Goal: Test and analyze the effectiveness of the hang prototype; identify desired changes

Method

We will attempt to hang manually, using 2 controllers for the lift and drivetrain control. While we are attempting to hang we will have someone ready to catch the robot as it falls until we get comfortable with the hanging action. This method will allow us to test the hang in a way that mitigates the potential risks of falling while still being able to improve the hanging mechanism.

1. Start the hang on the ground
2. Slowly go from bar to bar ready to catch the robot
3. Record the current capabilities of the hang
 - a. Identify the potential ways for the robot to fall and where assistance is needed

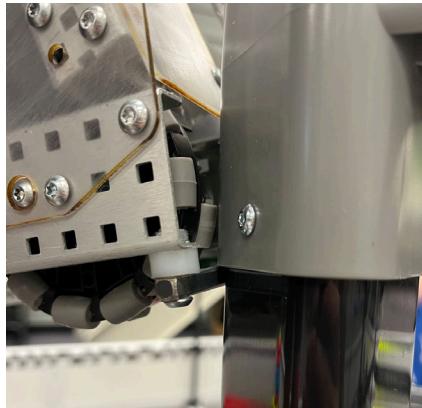
Results

Trial #	Floor -> Tier 1	Tier 1 -> Tier 2	Tier 2 -> Tier 3	
1	Assistance Neccesary	Assistance Neccesary	Assistance Neccesary	<i>Needed Assistance in 78% of transfers</i>
2	Assistance Neccesary	No Assistance Neccesary	Assistance Neccesary	
3	Assistance Neccesary	No Assistance Neccesary	Assistance Neccesary	

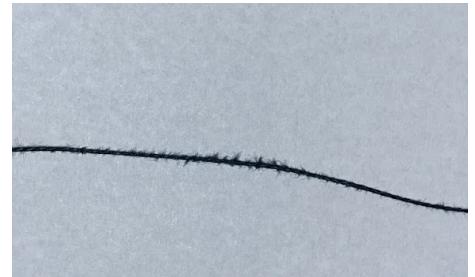
The data above supports that our hang is possible with it completing 2 transfers properly, however there is still a lot of room for improvement. The 3 biggest failure points are below:



Polycarbonate hooks yielding when the robot is hanging. We plan to fix this by adding metal supports to the polycarbonate hooks.



Back high-strength shaft getting stuck on the support for the hang structure when going from rung to rung. A polycarbonate back plate can prevent this getting stuck.



Kevlar string fraying after 2-3 hangs. This leads us to believe that the durability of this string is much less than originally thought in our [string selection](#) (Pg. 287).

11/19/24

Reevaluate: Hang String Selection (R.2.1.6)

Designed by: Alex

Witnessed by: Carl

Witnessed on: 11/19/24

Goal: Evaluate our selection of string and put in order for new hang string

In our [hang string selection](#) (Pg. 287), we decided to order the 0.8mm kevlar string, however, once we tested it we found that its abrasion resistance was much lower than that stated by the manufacturer and both strings ended up breaking after 2-3 hang attempts. It's for this reason that we will reconsider our string choices for hang to have higher abrasion resistance to deal with rubbing on metal edges throughout our robot. Additionally, we added a monofilament fishing line to the matrix as well to replace the kevlar with another option.

Monofilament Fishing Line**Dyneema****UHWPE**

Criteria:	Weight	Score	Weighted Score	Score	Weighted Score	Score	Weighted Score
Strength	2	3	4	3	6	5	10
Size (Smaller is Better)	2	8	16	6	12	5	10
Flexibility	3	3	9	6	18	7	21
Abrasion Resistance	4	5	20	9	36	10	40
Price (Lower is Better)	2	5	10	4	8	3	6
Total Score:			59		80		87

Decision

In this revised design matrix, the UHMWPE now scores the highest. We are going to go through and order 100ft of the 1mm thick UHMWPE that we found on amazon that has a breaking force of 440lbs, and it should arrive by 11/21, giving us adequate on field time to test before our next event.

11/19/24 Testing: Intake, Drivetrain, Wall Stakes (R.2.1.6)

Designed by: Carl

Witnessed by: Alex

Witnessed on: 11/20/24

Goal: Test all the subsystems of our robot (excluding hang) and identify any problems.

Overall Performance:

First, we wanted to test the overall performance of the robot with all the subsystems working together to uncover any big underlying problems. To do this, we drove the robot around the field in a manner comparable to skills runs/matches until the subsystems stopped working.

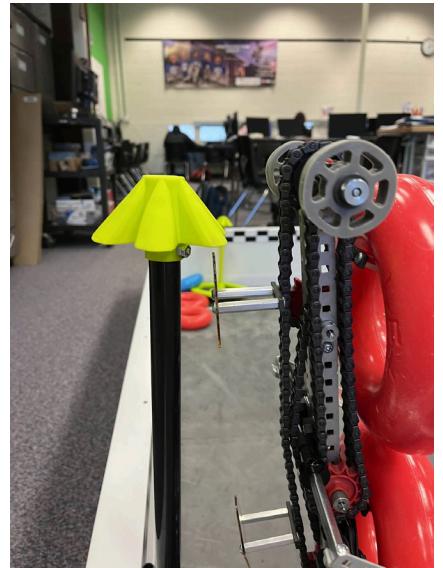
Subsystem:	Time Till Preformance Drop	Time Till Total Burnout
Drivetrain	3 min	6 min
Top Intake	4 min	5 min
Bottom Intake	3 min	6 min
Back Clamp	14 actuations	17 actuations
Lift	5 min	well over 6 min

We were quite satisfied with the fact that all of our mechanisms had the capability to last for well over a 2 minute match, even in extreme circumstances. This test did, however, highlight some smaller issues with the robot that need to be addressed.

Scoring on Wall Stakes:

Scoring on wall stakes during this testing period was very difficult for the following reasons:

- Bad Alignment Mechanism
 - The bottom stage intake was not rigid enough side to side
 - This caused the alignment mechanism to push the intake side to side, not turn the robot
 - The alignment mechanism was too contained within the drivetrain
 - The robot required too much precise driving before the aligner to be used
- Hooks Getting Stuck on Scored Rings
 - The hooks on the top stage consistently got stuck on previously scored rings when there were more than three rings on the wall stake



(Intake too close to wall stake)

Loading for Wall Stakes:

Similar to the problem we had with [R.1.2.6](#) (Pg. 161) with the rings falling off the hooks while loading them on the back of the intake. This will likely be able to be resolved in a similar way as before.

11/20/24 Design: New Negative Eliminations Autonomous

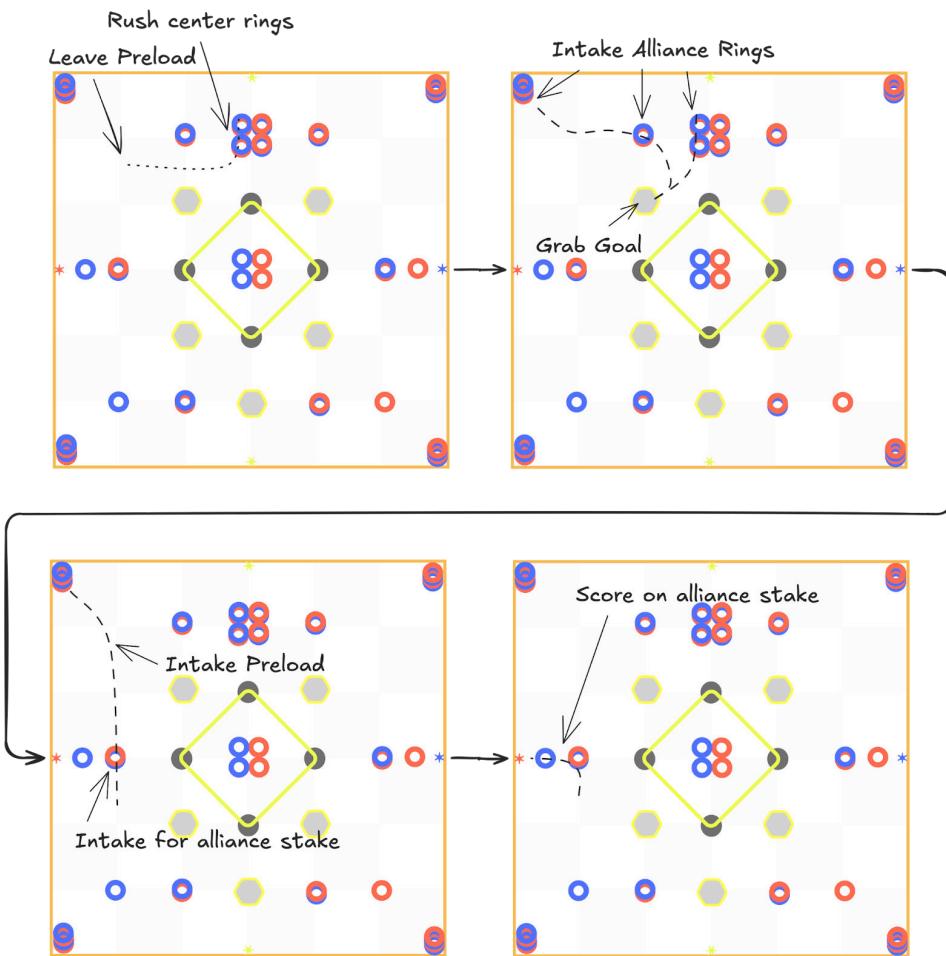
Designed by: Alex

Witnessed by: Carl

Witnessed on: 11/20/24

Goal: Develop a new higher-scoring negative side eliminations autonomous routine.

After collaborating with our alliance at the Butter Nexus league event, we have decided to run our autonomous on the negative side autonomous. We wanted to use this as an opportunity to build upon our previous [negative side autonomous](#) (Pg. 214) using our new corner intake mechanism. Additionally, in this autonomous routine we wanted to “rush” the middle rings to ensure that we would be able to control these crucial rings at the start of the autonomous period.



This autonomous routine will allow us to score 7 rings and essentially guarantee that we will get one of our alliance's rings from the center line. Additionally this autonomous routine will allow us to start the match with an entirely full goal and a ring on the alliance stake. In total this autonomous routine could score us up to 11 points and prepare us very well for the match.

11/20/24 Design: Drivetrain System Identification

Designed by: Alex	Witnessed by: Carl	Witnessed on: 11/20/24
-------------------	--------------------	------------------------

Goal: Design an implementation of system identification for the drivetrain.

In [background research](#) (Pg. 278-280) we discussed the working mechanism behind system identification. In this entry, we will go through the implementation details needed for having system identification on the drivetrain for the angular and linear velocity controllers, along with a static friction constant.

Requirements

- Collect control inputs from the drivetrain
 - Interpret them as an angular velocity input (turning) and a linear velocity input
 - Get most up to date control inputs
 - Take into account the possibility that multiple control inputs/no control inputs were given for the same frame and only account for the most recent control input
- Collect current speed data from drivetrain
 - Get the speed from the motors on the drivetrain
 - Interpret as angular/linear velocity
 - (Possibly) Filter data coming from the sensors for secant line approximation of derivative
- Data analysis
 - Utilize system identification
 - Send feedforward values over controller
- Movement
 - Create a movement that allows for a consistent output to give more expected results

Design

Control Input Collection

To record the control inputs there are 2 primary parts: ensuring latest data and recording the data. To ensure the latest data we will add an extra variable to our drivetrain class that records the last control input given to the drivetrain in the `setPct(left, right)` function. Additionally, because the inputs to this function are listed as left/right, we have to have a translation step to record them as linear/angular inputs. We use the following algorithm to calculate the linear and angular velocities the inverse of the calculation for arcade control.

```
void setPct(const double left, const double right) {
    this->left11W.move_voltage(left * 12000.0);
    this->right11W.move_voltage(right * 12000.0);

    lastULinear = (left + right) / 2.0;
    lastUAngular = (right - left) / 2.0;
}
```

Next, we use a vector member variable in the drivetrain subsystem class to record the drivetrain control inputs over time in a standard library vector. Additionally, we use a recording variable to ensure that data points are not recorded unnecessarily.

```
void periodic() override {

    // Rest of function removed for brevity

    if (recording) {
        uLinear.emplace_back(lastULinear);
        uAngular.emplace_back(lastUAngular);
    }
}
```

Speed Data Collection

We record the speed in a slightly different way, as it doesn't need to be recorded with the last value the same way. We utilize the motor encoders and perform a secant line approximation of the current velocity of the drive given the change since the last frame. Additionally, because of the double secant line approximation of the drivetrain angular velocity, which we thought could be inaccurate, we opted to use the IMU's internal readings of the robot angular velocity as it does not go through this secant line approximation process.

```
if (recording) {
    uLinear.emplace_back(lastULinear);
    uAngular.emplace_back(lastUAngular);

    xLinear.emplace_back(((leftChange + rightChange) / (2.0 * 0.01)).getValue());
    xAngular.emplace_back((-imu.get_gyro_rate().z) * (degree / second).getValue());
}
```

Data Analysis

For the analysis of the data we will use two *OneDofVelocitySystem* objects for the linear and angular velocity of the robot. We will then run the characterization step given the system data and print out the estimated ideal feedforward variables.

```

void analyzeSysIdData() const {
    OneDofVelocitySystem linear;
    OneDofVelocitySystem angular;

    linear.characterize(xLinear, uLinear);
    angular.characterize(xAngular, uAngular);

    std::cout << "Linear: " << linear.getFF() << std::endl;
    std::cout << "Angular: " << angular.getFF() << std::endl;
}

```

This function sends the feedforward values directly to the terminal, which then can be read using the controller.

Movement/Command

To have the most consistency in our system identification we want to have a repeated set of motions that we can test our robot to move through during this process making the consistency (ideally) perfect.

```

Sequence *characterizeAngular() {
    return new Sequence({
        new InstantCommand(
            [this]() mutable {
                this->recording = true;
                uAngular.clear();
                uLinear.clear();
                xLinear.clear();
                xAngular.clear();
            },
            {}),
        this->pct(0.5, -0.5)->withTimeout(500_ms),
        this->pct(1.0, -1.0)->withTimeout(400_ms),
        this->pct(-0.5, 0.5)->withTimeout(800_ms),
        this->pct(-0.2, 0.2)->withTimeout(800_ms),
        this->pct(1.0, -1.0)->withTimeout(400_ms),
        this->pct(-0.2, 0.2)->withTimeout(300_ms),
        new InstantCommand(
            [this]() mutable {
                this->recording = false;
                analyzeSysIdData();
            },
            {}),
        }));
}

```

This command allows us to easily, quickly and reliably run system identification on angular velocity. We do an almost identical procedure for the linear velocity, but that code is omitted for brevity.

Method

To keep consistency when doing drivetrain identification, we created the following method for our system identification:

1. Clear out $\frac{1}{4}$ of the field for the test
2. Upload Sys-ID controller mode
3. Connect V5 controller
4. Run Linear System Identification
 - a. Run system identification
 - i. Record data
5. Run Angular System Identification
 - a. Run system identification
 - i. Record data
6. Repeat steps 5 and 6 with a goal in the clamp
7. Test skills route with new values

Ideally, this strict test regime will ensure that data is robust against a variety of fields and can be tuned properly at a moment's notice.

Summary

- Designed an approach for system identification on the drivetrain
 - Utilizes the fact that linear and angular velocities are independent for easier and more consistent identification
 - Records data from the real robot and uses OLS to find the best FF values for the system
- Created robust method for the tuning of new robots
 - Consistency is KEY for these values so having a robust method is a necessity

11/22/24 Design: Hang Macro

Designed by: Alex

Witnessed by: Carl

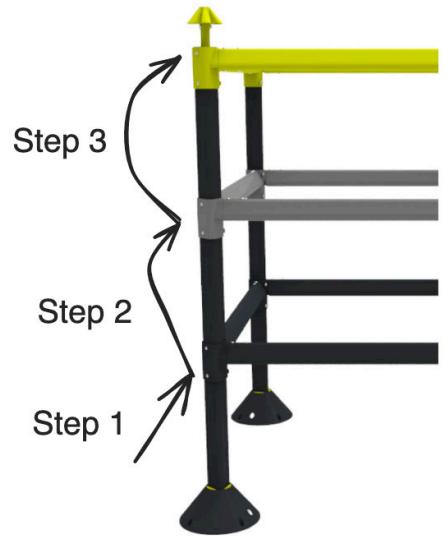
Witnessed on: 11/23/24

Goal: Design a macro for the robot to hang automatically.

Because of the complexity of our [hang mechanism](#) (Pg. 273-277), we will need a macro that allows the hang to be completed fully autonomously for it to be viable. Our plan is to lay out the hang macro into a couple easier steps and then integrate that into a single macro that we can use to fully hang.

Hang Key Steps:

1. Get off the ground
2. Move from bar to bar
3. Repeat step #2



Drivetrain Subsystem Modifications

Currently, the drivetrain code is not designed to work with the PTO for the hang mechanism, nor is it compatible with tracking the current length of the string, which is important to being able to put the hooks in the right position to release the goal. As we want the hang macro to ideally be an irreversible process when started (ensuring the drivetrain motors on each side don't fight).

Releasing PTO/strings

```
InstantCommand *releaseHang() {
    return new InstantCommand(
        [this]() {
            this->pto.set_value(true);
            ptoActive = true;
            this->lastStringLength = this->getStringDistance();
            this->stringRelease.set_value(true);
        },
        {}
    );
}
```

Keeping track of string position

```
void periodic() override {
    // ...

    if (ptoActive == true) {
        // Get the current length of the string (Also measures drivetrain change in
        // other parts of the match
        const QLength currentLength = this->getStringDistance();

        // Find the change
        const auto change = currentLength - lastStringLength;

        // Add the change to the actual string length
        this->stringLength += change;

        // Update last string length to better keep track of the string positioning
        lastStringLength = currentLength;
    }
}
```

Making sure motors don't fight

```
RunCommand *hangPct(double pct) {
    return new RunCommand([this, pct]() { this->setPct(pct, pct); }, {this});
}
```

Getting Off the Ground

For step 1 the lift has to move up to grab the bar, and then the drivetrain has to move backwards to hook onto and lift up from the bar. We used the following code implementation to create this motion:

```
hang = new Sequence({drivetrain->releaseHang(),
lift->positionCommand(75_deg)->race(drivetrain->hangPctCommand(0.0))->withTimeout(0.2_s),
lift->moveToPosition(125_deg)->race(drivetrain->hangPctCommand(0.0))->withTimeout(0.4_s),
            lift->positionCommand(125_deg)->race(drivetrain->hangDown(-1.0, -2.20_in)),
lift->positionCommand(75_deg)->race(drivetrain->hangPctCommand(-0.18))->withTimeout(0.2_s),
lift->positionCommand(110_deg)->race(drivetrain->hangPctCommand(1.0))->withTimeout(0.1_s)});
```

This code controls the movement of the lift to allow it to lift up to the second bar and set the robot onto the passive hooks. Additionally, at the end it lets out the drive for a tenth of a second to fully set the robot into the passive hooks.

Bar to Bar Transition

We will have to utilize the lift and the drivetrain together in synchronization to have the robot hang effectively as the winch and lift positions need to constantly be adjusted to achieve the desired result. This complex sequence for changing bars is reflected in the code below.

```
barToBarHang =
new Sequence({lift->positionCommand(55_deg)->race(drivetrain->hangUp(1.0, 7.5_in)),
  lift->positionCommand(95_deg)->race(drivetrain->hangPctCommand(0.0))->withTimeout(0.5_s),
  lift->positionCommand(95_deg)->race(drivetrain->hangDown(-1.0, 2_in)),
  lift->positionCommand(115_deg)->race(drivetrain->hangDown(-1.0, -2.20_in)),
  lift->positionCommand(75_deg)->race(drivetrain->hangPctCommand(-0.18))->withTimeout(0.2_s),
  lift->positionCommand(120_deg)->race(drivetrain->hangPctCommand(1.0))->withTimeout(0.1_s)});
```

Complete Sequence

We then combine these steps using a nested sequence to autonomously control the hang throughout the entire motion.

```
hang = new Sequence({drivetrain->releaseHang(),
  lift->positionCommand(75_deg)->race(drivetrain->hangPctCommand(0.0))->withTimeout(0.2_s),
  lift->moveToPosition(125_deg)->race(drivetrain->hangPctCommand(0.0))->withTimeout(0.4_s),
  lift->positionCommand(125_deg)->race(drivetrain->hangDown(-1.0, -2.20_in)),
  lift->positionCommand(75_deg)->race(drivetrain->hangPctCommand(-0.18))->withTimeout(0.2_s),
  lift->positionCommand(110_deg)->race(drivetrain->hangPctCommand(1.0))->withTimeout(0.1_s),
  barToBarHang, barToBarHang});
```

Summary

- A macro is a necessity to effectively make our hang work in skills or matches
 - Reduces points of failure
 - Makes the hang faster
 - Makes it possible in programming skills
 - Eliminates driver error
- Designed a sequence to hang using nested *Sequence* objects
 - Use repetitive nature to our advantage
 - Easy to modify values based on hardware modifications

11/22/24 Build: Hang Improvements (R.2.1.7)/(C.3.2)

Designed by: Alex

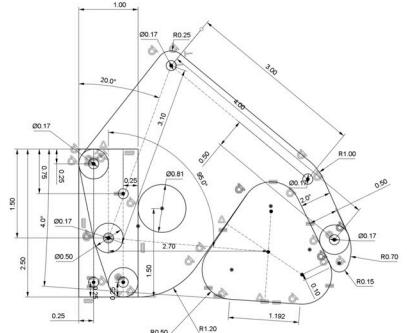
Witnessed by: Alex

Witnessed on: 11/23/24

Goal: Fix the hang issues from [R.2.1.6 \(Pg. 292\)](#) to allow for testing of the macro.

String Replacement:

The kevlar string was replaced with the [UHMWPE](#) string (Pg. 293). The new string was much stronger and had better abrasion resistance than the kevlar and integrated into the robot nicely.

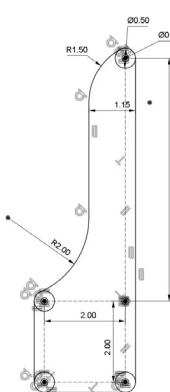


Hook Improvements:

The polycarbonate hook parts were redesigned to accommodate bracing from an aluminum plate, while also being optimized to take up less polycarbonate and nest more efficiently.

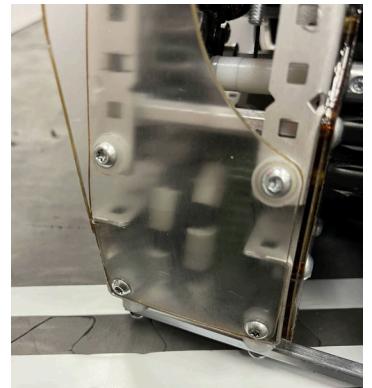


The hooks were integrated into the physical robot without problem using a bent piece of aluminum for strength in multiple directions (image on the far right)



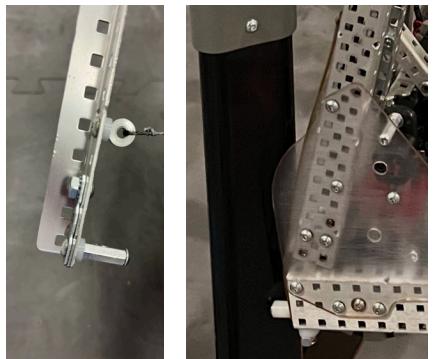
Upright Backing Polycarbonate:

The polycarbonate on the back of the uprights was modified to cover the back of the HS shaft crossbar. The design of this part was also changed to save polycarbonate.



Hang Arm Pivot Reinforcement:

Even with doubled up bearing flats, the intersection between the top and bottom hang arms was far too weak. The pivots were rebuilt using aluminum plates and standoffs with 2.5" screws to maximize their stability.



Other Changes:

Several other changes were made to the hang including:

- Reinforced standoffs at the top of the hang arm (close right)
- Added 0.5" spacer to outside drive rail to prevent the robot from sliding off of the vertical pipe (shown on far right)
- Slight changes to string routing and lift arms

11/22/24 Testing: Hang Macro

Designed by: Alex	Witnessed by: Carl	Witnessed on: 11/23/24
-------------------	--------------------	------------------------

Goal: Test and revise the hang macro.

Developed in our [previous entry](#) (Pg. 300-302), we need to robustly test our hang macro to ensure safety and speed while our robot is hanging.

Method

1. Tune the hang to a reasonable point to test without manual assistance
 - a. Note all major deviations from the initial design
2. Test the hang 5 times without code changes
 - a. Note all times the hang fell/needed assistance
 - b. Record the time the hang took from pressing the button to being secure on the top bar

Results

Changes needed

- Winch mechanism needed to move farther to pull bar into hooks
 - 0 in -> -2 in
- Lift down farther and sooner in between bars
 - 5 in -> 3 in and 75 deg -> 65 deg
- Lift down when we initially lift to not break multiple hang planes at the same time (110 deg -> 60 deg)

Trial	Needed Assistance?	Time (s)
1	No	12.4
2	No	11.8
3	No	12.2
4	No	11.4
5	No	12.0

Average time: 11.96s

Conclusion

The hang is currently in a viable state as analyzed in our [skills time analysis](#) (Pg. 242) and appears to be as fast as we can get it with the current winch system. We went through a process of tuning and measurement to find that the robot hung in an average of 11.96s. This time is fast enough, however there is a large potential to go faster by decreasing the friction on the string throughout the system, as that is currently very high.

11/23/24 Build/Design: Wall Stake Mechanism Improvements (R.2.1.10)/(C.3.2)

Designed by: Carl

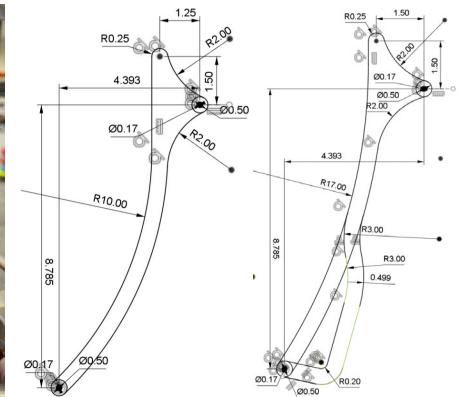
Witnessed by: Alex

Witnessed on: 11/23/24

Goal: Change the robot's wall stake mech to fix the [problems identified earlier](#) (Pg. 294).

Loading the Top Intake:

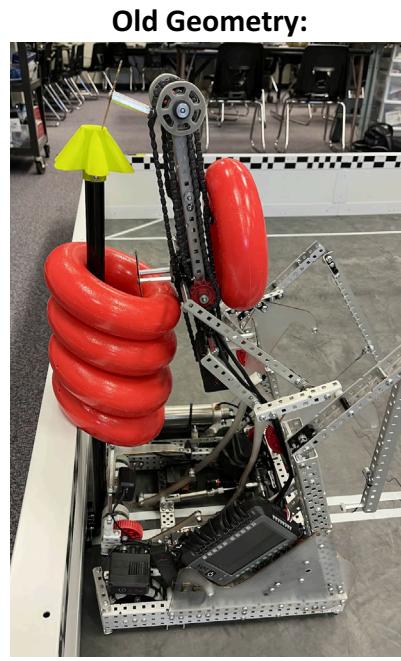
The top intake had very similar issues as the [old one](#) (Pg. 161) when loading, so we decided to solve them in a similar way. In order to implement polycarbonate strips behind the top intake, we needed to remove the 11W motor to allow the polycarbonate to be in the right position. This motor was relocated to the bottom, lowering the robot's CG and allowing the polycarbonate to support the rings correctly. The polycarbonate strips were made perpendicular to the old ones, increasing customizability (allowed for the motor recess), strength, and simplifying the mounting of them.



Scoring:

Our previous top stage scored adequately when aligned correctly with the exception of the hooks getting stuck on scored rings. This problem could only really be fixed by tilting the intake more, causing it to stick out further forward. This was not an issue, however, because we were rebuilding the aligner anyway and a change in alignment needs could be easily accommodated. The intake was tilted by taking advantage of the adjustable pillow blocks on the front and by raising the intake set point in code.

Images:



Old Geometry:

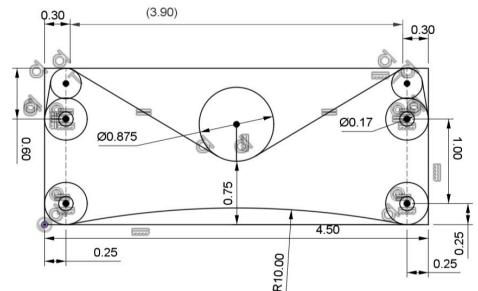
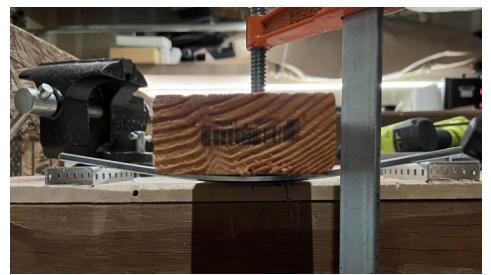


New Geometry:

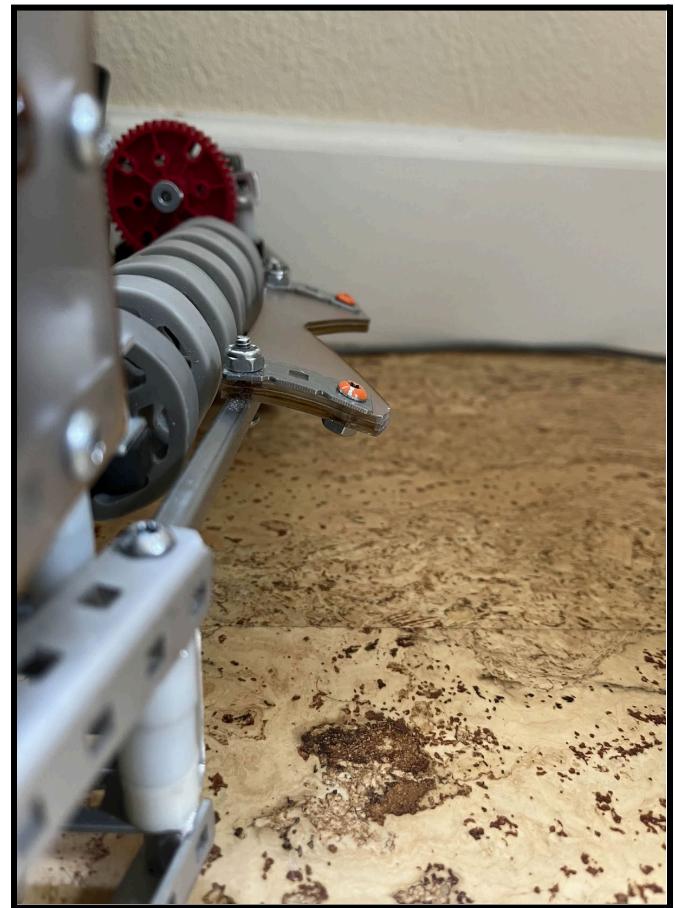
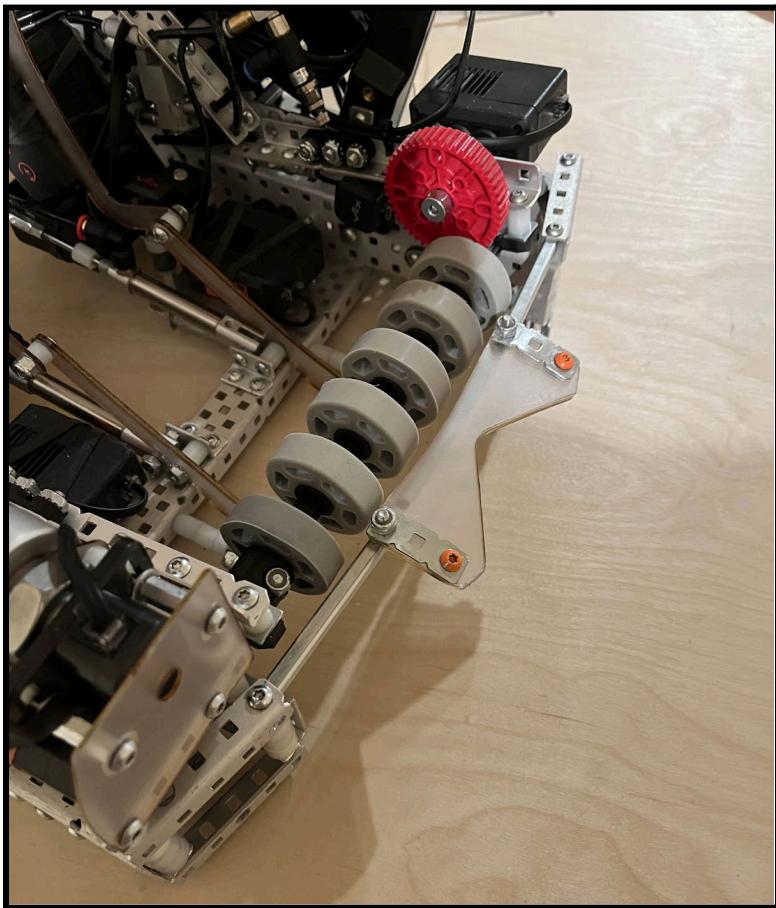
Aligner Mechanism:

Due to the structural limitations of an intake that clears the corner, we could not add any more support to the intake meaning that putting the wall stake aligner on the moving part of the intake was not viable. Additionally, as mentioned in the previous paragraph, the aligner needed to be further forward to accommodate the increased tilt of the top stage.

The new aligner was constructed out of a HS Shaft that was bent via a clamping setup and a triple stacked polycarbonate pole guide. This aligner is much stronger than the original and works significantly better.



Completed Bottom Intake Images:



11/24/24 Build: Hang String Release (R.2.1.11)

Designed by: Carl, Matt

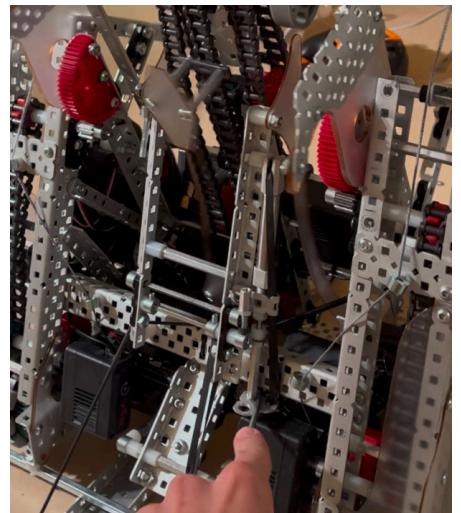
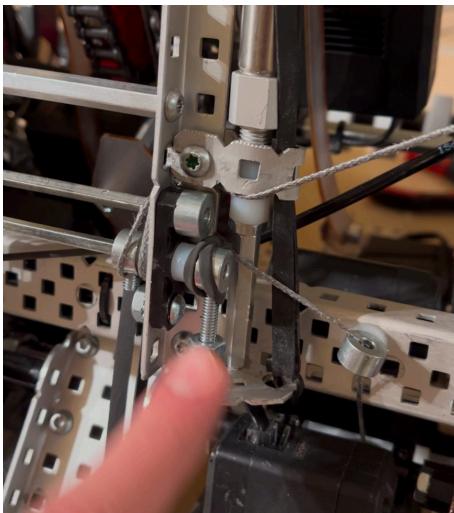
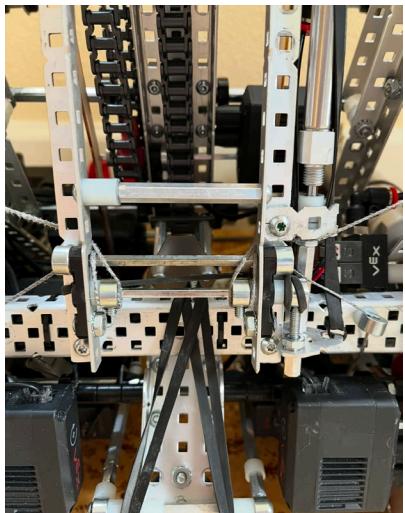
Witnessed by: Alex

Witnessed on: 11/25/24

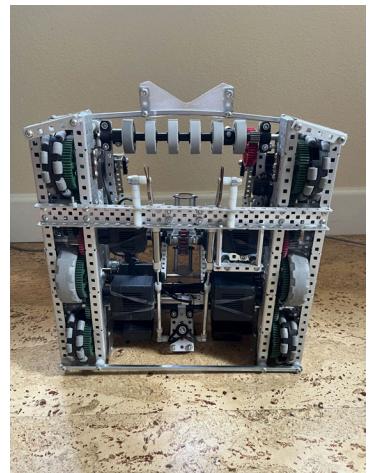
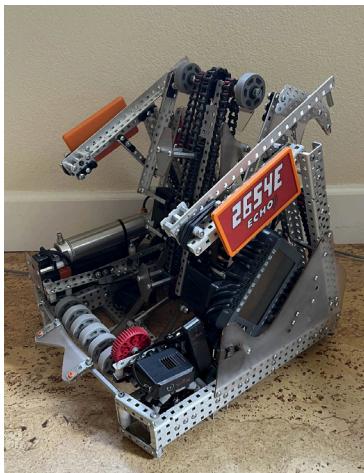
Goal: Create a pneumatic mechanism that can release the hang's string.

Release Mechanism:

To allow the robot to go under the hang structure both forward and backward, we added a mechanism to hold the string down until it was needed for hanging. To minimize weight and air usage, we limited the design to one 25 mm piston and minimal additional structure. We centralized the release mechanism by connecting both strings to a single central shaft, using shaft collars and threaded rods to hold the loops of string. The shaft uses a rubber band to create rotational force, enabling the string to release quickly. When the piston is extended, it allows the shaft to rotate, releasing the string loops from the threaded rods. The string release process is challenging to capture fully due to its speed.



Progress Photos:



12/01/24 Testing: Drivetrain System Identification

Designed by: Alex	Witnessed by: Carl	Witnessed on: 12/02/24
-------------------	--------------------	------------------------

Goal: Test the system identification and tune RAMSETE path following constants.

To test the drivetrain system identification we will create a new command using the control as the control input and record the actual movements of the drivetrain.

Method

1. Utilize the method created in [drivetrain system identification](#) (Pg. 296-299) to get tuning variables for the current robot while holding a goal and not holding a goal
2. Run step 1 three times and compare the results for the tuning values
 - a. This step allows us to test the consistency of the tuning process over multiple times
3. Apply one of the sets of tuning constants in the code
4. Test the code over the programming skills routine without feedback (Entirely open loop)
 - a. Check for accuracy

Results

Consistency test:

In the consistency tests the values appeared very reasonable and were very consistent from run to run, showing promise in this method.

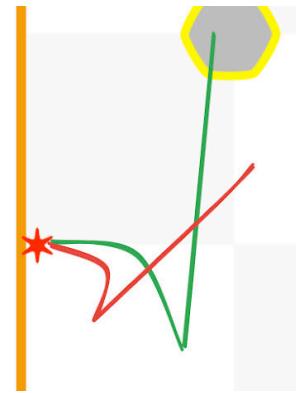
Trial	Linear FF (No Goal)	Angular FF (No Goal)	Linear FF (Goal)	Angular FF (Goal)
1	0.538/-0.058	1.04/0.32	0.552/-0.068	1.12/0.42
2	0.498/-0.062	1.00/0.30	0.572/-0.062	1.09/0.37
3	0.502/-0.059	1.08/0.27	0.565/-0.070	1.13/0.34

All items in table listed as (velocity FF)/(acceleration FF)

However, this data does also have issues, for example, the linear acceleration FF was always negative, indicating a possible issue with the data coming from the drivetrain.

Accuracy test:

During the accuracy test we had severe inaccuracies compared to the expected results. Before we graphed the data it was more than obvious that the robot was not moving accurately as it was not able to get past the hang structure, showing very severe inaccuracies. This test proved that the current implementation of system identification has some extreme errors and must be corrected.



Conclusion

Our testing showed that there is a severe error with the system identification that we currently have. We got to this conclusion after running several tests that showed multiple potential errors, and then severe accuracy errors when testing on the real robot with realistic paths. This is why we do not think it is worth sinking time into this change while we already have this working.

Next Steps

Currently, we have little time until our next event (Dec. 3) so we have decided to continue to use our last implementation of RAMSETE without the angular velocity control, just using our old left/right drivetrain control for the league night on December 3rd. After this event we will reconsider system identification for the event on December 14th.

12/02/24 Testing: New Negative Eliminations Auton

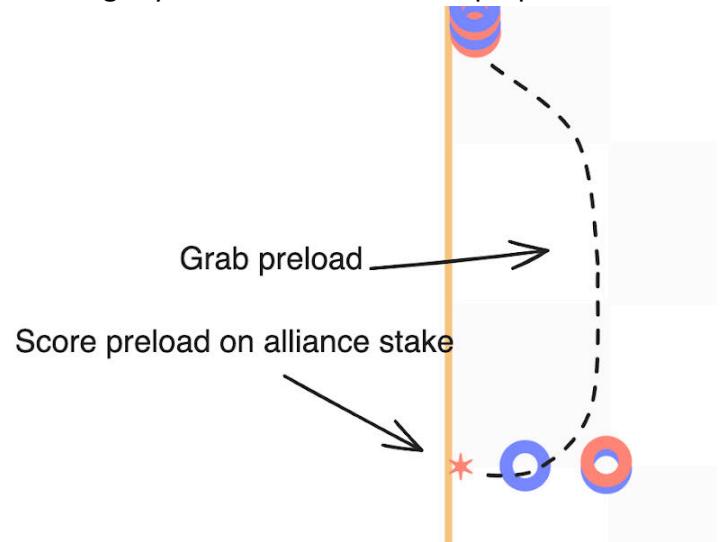
Designed by: Alex	Witnessed by: Carl	Witnessed on: 12/02/24
-------------------	--------------------	------------------------

Goal: Rigorously test the new negative side autonomous routine (Pg. 295).

For the negative side autonomous routine we are going for 3 main objectives, rush the rings near the centerline (without crossing the autonomous line), score a ring on the alliance stake, and grab the goal on our side of the field. Secondary tasks are to intake the rings from the corner to slightly increase our score and prepare us batter for the match, however this is not a primary objective.

Changes

We found from other testing that we can score with the wall stake mech forwards with the robot. We changed the autonomous routine to only get 5 rings because we thought that we could make it more reliable that way. Additionally, scoring from the front of the robot allows us to score without dropping the goal giving us a second advantage in the match.



Results

Post-Tuning Success Rate:

Trial	Scored Rings Total	Top Rings Scored	Crossed Line?	Scored Alliance Stake?	Grabbed Goal?
1	5	2	No	Yes	Yes
2	6	2	No	Yes	Yes
3	5	2	No	Yes	Yes
4	4	2	No	Yes	Yes
5	6	2	No	Yes	Yes

Main component success rate: 100%

Full ring success rate: 33%

Our results are very promising for this auton, scoring all the necessary objectives. Even though it misses some rings, they are all with the corner which was a hardware issue that we will fix with future robot iterations.

12/04/24 Analysis: Butter Nexus League Finals (12/03/24)

Designed by: Matt	Witnessed by: Alex	Witnessed on: 12/05/24
-------------------	--------------------	------------------------

Goal: Record match results and analyze matches.

Match	Final Score	Win Point	Auton Win	Notes
QF 1-1	38-6	N/A ▾	Yes ▾	Our autonomous worked perfectly. We gained 3 goals in autonomous and secured both positive corners once the match began. In the last 45 seconds, we scored on wall stakes and mobile stakes. Our opponents lost a battery at the 1-minute mark, allowing us to leave the corner unprotected.
SF 1-1	47-18	N/A ▾	Yes ▾	Our autonomous worked perfectly. We gained 3 goals in autonomous and secured both positive corners once the match began. In the last 15 seconds, we scored on wall stakes and mobile stakes.
F 1-1	26-32	N/A ▾	No ▾	Our autonomous got all critical objectives but missed one ring. However, our alliance's auton did not work. Once the match started, we were able to get the 3rd goal and secured one positive corner. We switched with our alliance to score on wall stakes and mobile stakes, but unfortunately, our alliance lost the positive corner they were holding. During the last 15 seconds, we sat in front of the wall stake while our alliance tried to hang.

Matches Summary:

Strengths:

- Mobile Goal Control - We were able to get 3 goals every match in autonomous or once it started.
- Top Rings on Wall Stakes - In our matches we were able to get the top ring on at least one of the wall stakes.
- Positive Corner Control - We were able to get 2 positive corners in our first two matches.
- Defensive Wall Stakes - In our matches when we had the top ring we protected the wall stakes protecting the top ring

Points of Failure:

- Our alliance didn't score any rings in the autonomous period of finals.
- Our alliance lost 2 positive corners to the opposing alliance in finals.
- Our alliance went to hang instead of protecting the top ring on the wall stake.

Conclusion:

Our robot, autonomous routine, and strategy through the elimination matches proved to be very effective. In finals, however, the three separate blunders by our alliance partner made losing the match out of our control. We believe that we had very thorough strategic discussions with our alliance prior to the event, and the shortfalls in finals were a result of our alliance's poor execution of said strategy.

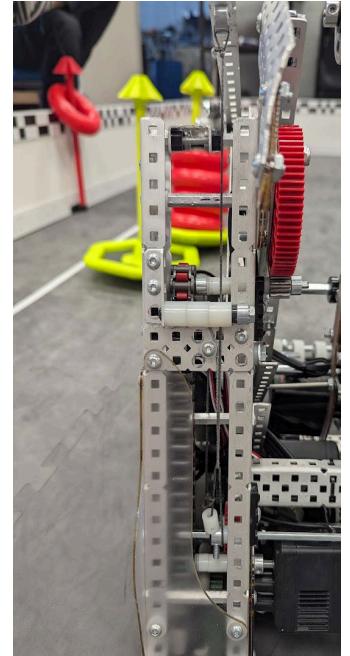
12/05/24 Build: Hang Changes (R.2.1.14)

Designed by: Carl, Matt	Witnessed by: Alex	Witnessed on: 12/06/24
-------------------------	--------------------	------------------------

Goal: Reduce friction and string wear. Add an alignment mechanism into the string release.

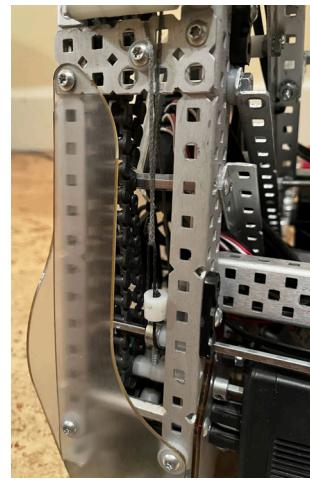
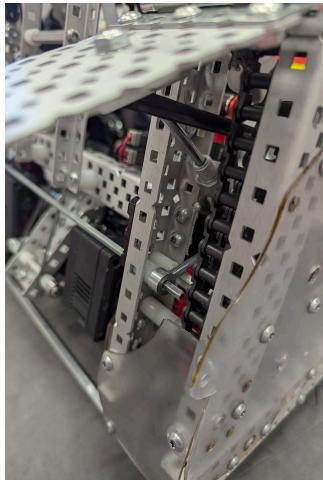
String Rerouting:

To speed up our hang, we needed to reduce friction by improving the routing. The old method effectively brought the string from the center to each side, but caused significant friction and wear. We solved this problem by changing the winch location to directly above the drivetrain, eliminating the need for a complex routing. This adjustment simplified the design, allowing a single shaft collar to guide the string to the top where 0.5" OD spacers to provide a smooth angle change in the path of the string.



Alignment Mechanism and String Release:

In order for the hang to work properly, the positioning prior to hanging needed to be exact. Using an aligner mechanism allows for some variance in the program or driving, while still allowing for perfect alignment. To maintain the use of one piston and to keep minimal additional structure, we developed a solution that combines the alignment mechanism and string release into a single system. The alignment mechanism needed to be very thin in order to fit on the back of the uprights, and is deployed via a 25mm piston attached just above the offset drivetrain gear. When the aligner is extended, the threaded rod is no longer captive, allowing the shaft to spin and deploy both sides of the hang quickly. Additionally, we integrated adjustable string loops into the release system to allow for easy resetting and tuning.



Furthermore, we extended the linkage between the lift and hang by 2 holes to better comply with <SG3>.

12/09/24 Testing: Hang Macro Revised

Designed by: Alex	Witnessed by: Carl	Witnessed on: 12/09/24
-------------------	--------------------	------------------------

Goal: Test and further develop the hang macro with the changes from [string rerouting](#) (Pg. 312).

Use a similar approach to our [last hang macro testing](#) (Pg. 304) to further develop a new hang macro with the changes from string rerouting.

Method

1. Tune the hang to a reasonable point to test without manual assistance
 - a. Note all significant deviations from the initial design
2. Test the hang 5 times without code changes
 - a. Note all times the hang fell/needed assistance
 - b. Record the time the hang took from pressing the button to being secure on the top bar

Results

Changes

- Remove arm lift step to de-jam hooks from bar
- Change all lift set points by 20 degrees to account for new lift changes
 - The arm now has to move less to reach the same setpoints taking less time.

Trial	Needed Assistance?	Time (s)
1	No	8.5
2	No	8.1
3	No	8.7
4	No	8.3
5	No	8.4

Average time: 8.4s (31% improvement)

Hang Step Through

Step 1

Before hanging, the robot is roughly aligned with the hanging structure vertical bar on the right side of the robot.



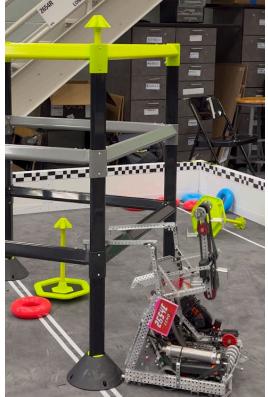
Step 2

Hang alignment mech/winch string releases, robot drives to bar.



Step 3

Robot grabs the bar using the lift and begins to pull in the winch. The lift says up to ensure the robot's passive hooks can clear the bar.



Step 4

Robot pulls the passive hooks around the bar using the lift.



Step 5

The robot uses the lift bringing the passive hooks down onto the horizontal bar of the hanging structure.



Step 6

Winch is let out to prepare to grab the next bar. The robot's lift moves dynamically as the winch unspools to allow it to miss the next horizontal bar and avoid getting stuck.



Step 7

The robot utilizes the lift and the winch to put the hooks directly on the bar above it.



Step 8

Lift the robot using the winch and arm together to get the passive hooks over the bar.



Step 9

Utilize the arm and winch together to drop the passive hooks onto the second bar, start lifting the arm again.



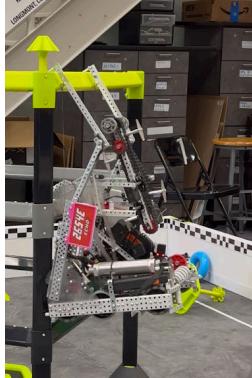
Step 10

Repeat step 6 to despool the winch.



Step 11

Repeat 7 and grab the tier 3 bar.



Step 12

Repeat steps 8 and 9 to winch the robot fully up to tier 3



Conclusion

The new hang macro with robot changes has an overall hang speed increase of ~31% from the old hang macro and string routing. By modifying the hang macro to remove unnecessary movements with the reduced friction causing a higher speed, the robot was able to hang in an average time of 8.4 seconds, a huge improvement over the old hang design which was hanging in about 12 seconds.

12/09/24 Testing: RAMSETE/Feedforward Constant Improvement/System Identification

Designed by: Alex	Witnessed by: Carl	Witnessed on: 12/09/24
-------------------	--------------------	------------------------

Goal: Test and further develop RAMSETE for higher accuracy.

Particle Filtering Issue

After testing the RAMSETE, we noticed that it occasionally seems sluggish and slower to react than we thought it would, generally indicating either a lag in measurements or outputs. Initially, because we were not familiar with RAMSETE, we thought this was normal, but after further investigation, we think it has something to do with our [particle filter implementation](#) (Pg. 131-135). Upon inspection we found an issue with how the motion algorithms are supplied with updates from the Monte Carlo Localization. The current implementation has the following code at the END of the Monte Carlo Localization Update to update the average position of the particles:

```
float xSum = 0.0, ySum = 0.0;

for (size_t i = 0; i < L; i++) {
    xSum += particles[i][0];
    ySum += particles[i][1];
}

prediction = Eigen::Vector3f(xSum / static_cast<float>(L), ySum / static_cast<float>(L),
angle.getValue());
```

This code, while it works, is placed in a poor spot to ensure that it is run every frame, as earlier in the program we utilized an algorithm to reduce unnecessary updates. If this algorithm (in the following code) pops the execution out of the function, this leads to the program skipping the rest of the instructions in this function, which is helpful because it stops unnecessary convergence, however, it does have the unintended side effect of causing the average particle prediction to not update on certain frames when it still can, leading to delayed data from the sensors.

```
if (distanceSinceUpdate < maxDistanceSinceUpdate && maxUpdateInterval > pros::millis() *
millisecond) {
    return;
}
```

The fix for this problem is very simple; if we do the averaging for the particle prediction every frame when it branches out, then we will no longer have this issue where the belief is behind the position of the robot.

Feedforward Constant/System Identification Improvement

In our last entry on system identification we identified several problems with our current approach that caused severe inaccuracies with the data coming from the robot, however, these numbers can still give us a ballpark for correct values to use in our path following and we can tune them further manually to get better values.

Additionally, we thought there were potentially issues in our new RAMSETE implementation, that we weren't able to robustly test before the Butter Nexus League, which we then needed to test with manually tuned values. After this tuning process we ended up with the following feedforward values.

	Goal	No Goal
Linear Velocity FF	0.66	0.63
Angular Velocity FF	0.1377	0.1302
Linear Acceleration FF	0.042	0.037
Angular Acceleration FF	0.0184	0.0155

As we tested these values without feedback throughout the skills path, they proved to provide very high accuracy, within about 4 inches at the end of the path. This is very close to the accuracy that we need from the pure FF to account for it with feedback.

RAMSETE Tuning

In addition to tuning the feedforward constants, we also have to adjust the RAMSETE path following constants to better fit this robot. For this we have created the following method to more consistently tune the constants:

1. Increase the b value until there is consistent oscillations in the path
2. Decrease the b by 15%
3. Slowly increase the ζ value by 0.1 until the oscillations in the path are damped
4. Test skills for accuracy

b : 38

ζ : 0.6

When testing these values throughout the skills path we noted very high accuracy in key points such as the wall stakes. These values also reduced oscillations of the robot to a minimum and prove to be very good tuning variables for RAMSETE. Currently, we do not have the time to do a full numerical comparison of the accuracy, however, we believe that these changes will make the RAMSETE more than adequately accurate to move effectively.

12/11/24 Build: Bottom Intake Modifications (R.2.1.15)

Designed by: Carl	Witnessed by: Alex	Witnessed on: 12/11/24
-------------------	--------------------	------------------------

Goal: Analyze the interaction between the intake stages to find out why rings occasionally get flung out. Fix the problem.

Problem Summary:

When testing skills, we noticed that rings would fly out of the front/side of the robot when transitioning to the top stage intake. After reviewing videos of our skills runs, we noticed that the rings would mostly fall out if the robot was making a sharp turn. We also noticed that the rings always fell out the left side of the robot. From these observations, we concluded that this was due to the uneven pressure from the polycarbonate strip that was pushing rings into the hooks combined with the centripetal force generated from the sharp turns.

Solution:

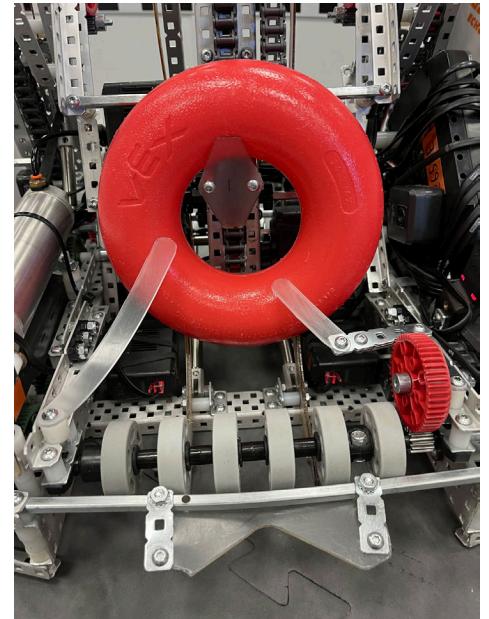
In order to get rid of the uneven force put on each side of the ring, we had two options:

1. Remove the polycarbonate strip entirely
 - a. This would have been an easy option
 - b. This option would not be viable as the intake ejected far more rings before the strip was added
2. Add another poly strip on the other side
 - a. This would balance out the force and could solve our problem
 - b. The intake's 48T gear and the 11W motor on the top stage prevent mirroring the current design
 - i. A similar alternative may be feasible
3. Timing the hooks to grab the rings perfectly using advanced software control
 - a. Generally overcomplicated
 - b. This would be compromising performance because of a potentially solvable build issue

We chose **option #2** as we thought it would have the highest performance if implemented properly.

Implementation:

We first started by experimenting with different mounting positions and lengths for the polycarbonate strip. We eventually found that by adding a 5x1 plate with a slight twist, we could get around the 48T gear and achieve a similar level of compression to the other side. Once the positioning of the scrap and length of the poly strip were finalized, we cut a second piece of polycarbonate to hold the scrap in the desired location.



12/13/24 Testing: Skills Repath Testing (R.2.1.15)

Designed by: Alex, Carl

Witnessed by: Matt

Witnessed on: 12/14/24

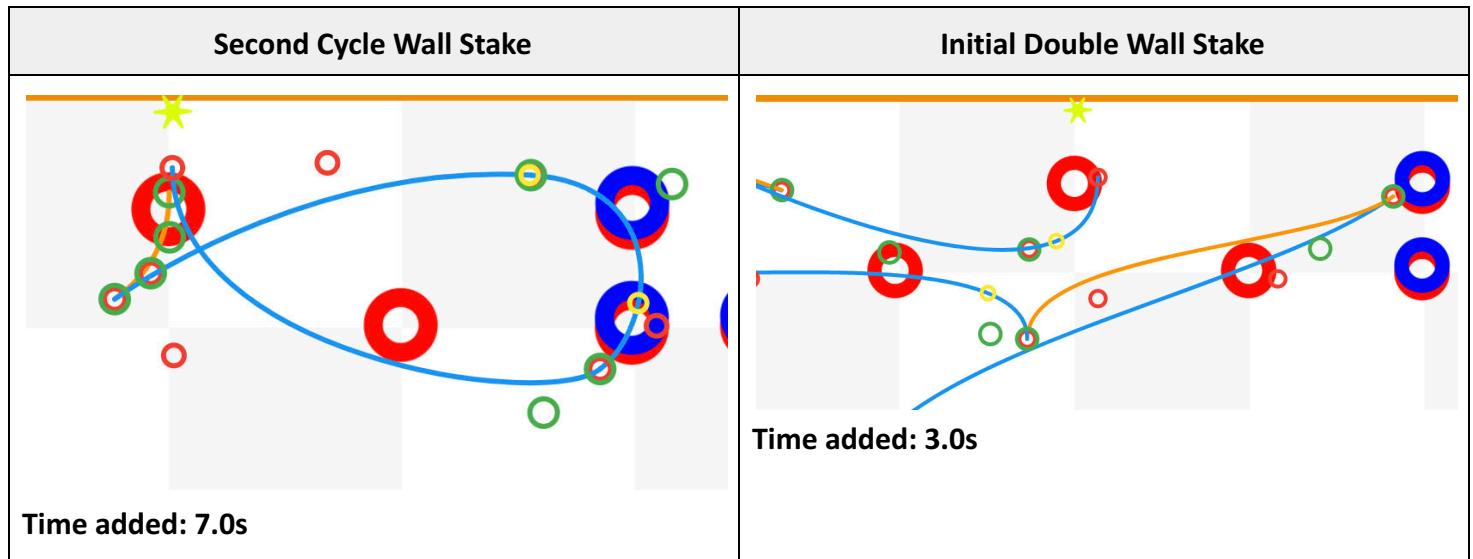
Goal: Test and adjust the new skills re-path.

Initial Testing

In our initial testing, the path did not need any major path adjustments, only minor accuracy adjustments to align with the wall stakes. We found that the new Monte Carlo Localization changes make our initial estimates more consistent.

Adjustments: Increased Score

Currently, the path misses 2 rings around the field, which we think would be fairly easy to re-path and get to slightly increase our score. We considered the 2 following ways to score more rings, adding a new path for a second wall stake cycle, scoring both of the remaining rings, or a minor re-path allowing for us to score 2 on the first wall stake. Drawings of the paths are the following:



Based on the associated risk and the time added, we decided to do the initial double wall stake path. We do not think this path change will increase the risk for the rest of the path, actually mitigating the problems we were having with blue rings getting into the way for pushing the second goal into the corner. Additionally, this route gets us a score of **69**.

Results/Data:

#	Trial	Alliance 1	Mobile Goal 1		Wall 1		Goal 2		Alliance 2		Mobile Goal 3		Wall 2		Mobile Goal 4		Hang			
			Scored	Good Grab	All Rings	In Corner	Missed	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored	Good Grab	In Corner	Some Rings	Aligned + Scored	Good Grab	In Corner	Some Rings
1	Scored	Good Grab	All Rings	In Corner	Missed	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored	Good Grab	In Corner	Some Rings	Aligned + Scored	Good Grab	In Corner	Some Rings	Aligned	
2	Scored	Good Grab	In Corner	All Rings	Missed	In Corner	Aligned + Scored	Good Grab	In Corner	Some Rings	Missed	Good Grab	In Corner	Some Rings	Aligned + Scored	Good Grab	In Corner	Some Rings	Misaligned	
3	Scored	Good Grab	In Corner	All Rings	Missed	In Corner	Aligned + Scored	Good Grab	In Corner	Some Rings	Missed	Good Grab	In Corner	Some Rings	Aligned + Scored	Good Grab	In Corner	Some Rings	Ended Early	
4	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	Some Rings	Aligned + Scored	Good Grab	In Corner	Some Rings	Aligned + Scored	Good Grab	In Corner	Some Rings	Ended Early	
5	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	Some Rings	Aligned + Scored	Good Grab	No Rings	Not in Corner	Aligned + Scored	Good Grab	Some Rings	Not in Corner	Ended Early	
6	Scored	Good Grab	In Corner	All Rings	Missed	In Corner	Aligned + Scored	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Ended Early	
7	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Missed	Good Grab	Some Rings	In Corner	Missed	Good Grab	Some Rings	Aligned + Scored	Good Grab	Some Rings	Aligned + Scored	Good Grab	Some Rings	Ended Early
8	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Missed	Good Grab	All Rings	In Corner	Missed	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Ended Early	
9	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Missed	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Misaligned	
10	Scored	Good Grab	In Corner	All Rings	Missed	Ended Early	Ended Early	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Ended Early	
11	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Missed	Good Grab	Some Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Ended Early	
12	Scored	Good Grab	In Corner	All Rings	Missed	Ended Early	Ended Early	Good Grab	Some Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Ended Early	
13	Scored	Good Grab	In Corner	All Rings	Missed	In Corner	Missed	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned	
14	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned	
15	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned	
16	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	Not in Corner	Aligned	
17	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Missed	Good Grab	Some Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned	
18	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Misaligned	
19	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Ended Early	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned	
20	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	Aligned + Scored	Good Grab	Aligned + Scored	Good Grab	Aligned + Scored	Good Grab	Aligned + Scored	Good Grab	Aligned + Scored	Good Grab	Aligned	
21	Scored	Good Grab	In Corner	All Rings	Missed	Ended Early	Ended Early	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned	
22	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Missed	Good Grab	Some Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned	
23	Scored	Good Grab	In Corner	All Rings	Ended Early	Ended Early	Ended Early	Good Grab	Some Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Ended Early	
24	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Misaligned	
25	Scored	Good Grab	In Corner	All Rings	Missed	Ended Early	Ended Early	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Ended Early	
26	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Missed	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	All Rings	In Corner	Aligned + Scored	Good Grab	Some Rings	In Corner	Aligned	

Analysis:

Overall, we were able to achieve an extremely high scoring skills routine, with one run of 68 points, one below our theoretical max. Despite this, we did encounter several reliability issues, particularly when aligning with wall and alliance stakes that caused the robot to score significantly lower than desired.

If our robot misses a wall stake, the missed ring sometimes falls onto the robot. This normally results in the lift/intake jamming as it comes down, meaning that the robot loses intaking abilities for the remainder of the run. This is shown in the data where a missed wall or alliance stake results in a catastrophic failure (Ended Early designation). In most cases, this jam is undetectable or unable to be rectified without making any significant changes to the design.

We have identified two main factors that we believe contribute to these inconsistencies:

- Smaller margin for error
 - The aligner offers a correction when the robot is off by $\pm 1"$, which is a smaller window than required for other tasks
- Location of the wall and alliance stakes
 - These stakes are located in the middle of the field walls, meaning that the readings from the distance sensors are less accurate than they would be if we were closer to either wall

Because of our competition tomorrow, we do not have the time to address any of these issues before then, however, we will ensure that we spend adequate time resolving these issues before Kalahari.

12/17/24 Analysis: Kent Denver Matches (12/14/24)

Designed by: Carl	Witnessed by: Alex	Witnessed on: 12/18/24
-------------------	--------------------	------------------------

Goal: Record results for matches and key points.

Qualifications:

Because of the lower level of competition at this tournament, analyzing each individual match would be redundant. Here are some key takeaways from our qualification matches:

- Consistent win point in autonomous
 - Our solo AWP routine allowed us to score 5/6 win points, even in matches when our teammates did not have autos
 - Missed alliance stake once due to setup issue
- Collaborative match strategy
 - Allowed alliance partners with slow or non existent intakes to protect the positive corner
 - One drive team member always communicating with our alliance partners
- High hang
 - In Q7, we got pushed into the hang structure resulting in our intake chain snapping
 - We had an autonomous win and two goals—one in the positive corner—but the other alliance controlled both wall stakes and three goals, including one in the positive corner, putting us at a points disadvantage
 - We were able to leave the positive corner at 15 sec and get a T3 hang, winning us the match
- Bad wall stakes
 - The tournament was using metal fields which caused the top of the wall stakes to lean in
 - This resulted in our hooks being in the wrong spot to score, so aligning was more difficult

Qualification					
Q 1 9:05 AM	2654E 1166N	42	0	45539A 3946N	
Q 7 9:35 AM	3946R 13358C	37	45	30809S 2654E	
Q 16 10:18 AM	55745F 2654G	11	29	2654E 20969A	
Q 20 10:36 AM	2481A 3946E	21	39	55745F 2654E	
Q 25 10:56 AM	2654E 3946A	42	24	2481B 13358A	
Q 29 11:16 AM	2654E 2654A	41	23	2481D 3946W	
Quarterfinals					
QF 1-1 1:34 PM	2654E 13358C	43	4	2481B 2481C	
Semifinals					
SF 1-1 2:06 PM	2654E 13358C	47	23	2654G 3946A	
Finals					
F 1 2:21 PM	2654E 13358C	48	0	1166B 3946S	

Eliminations:

- After placing first in qualifications, we picked 2nd seed due to their reliable wall stakes and high scoring
- We used a strategy of getting one positive corner, three mobile goals, and at least one wall stake
 - This strategy was easy to execute and low risk
 - Would've also worked IF we had lost auto
- Effective communication allowed us to quickly swap full goals for empty goals with our partner
- Consistency issues with rush auton
 - Ran AWP instead for last two matches

12/17/24 Analysis: Kent Denver Skills (12/14/24)

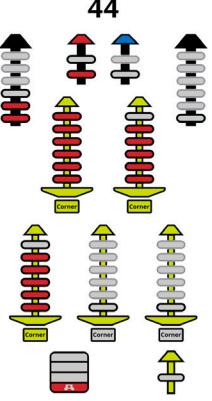
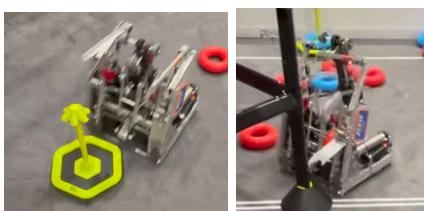
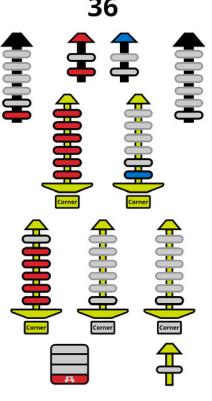
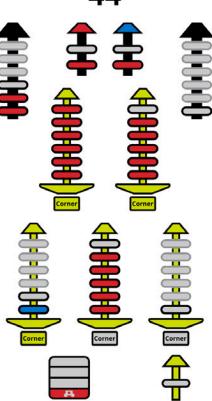
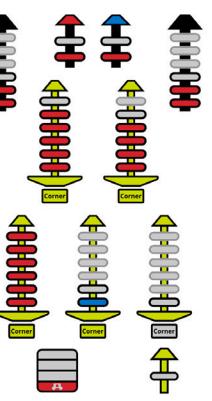
Designed by: Alex

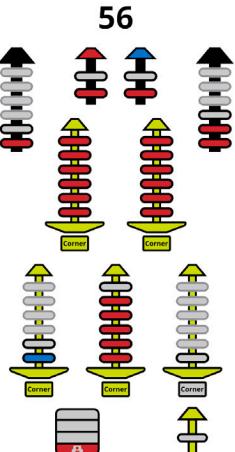
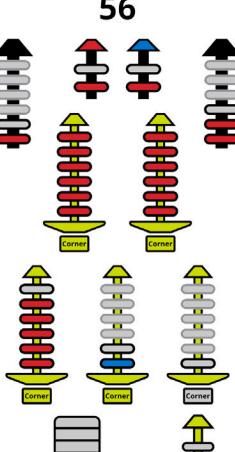
Witnessed by: Carl

Witnessed on: 12/18/24

Goal: Record skills results and analysis of failures.

In skills at this competition, we won the Skills Champion Award with a score of 55 in programming and 56 in driver, for a total score of 111. This was well below our expectations, and our analysis of the failures is below.

Driver Skills 1	Autonomous Coding Skills 1
<p>In this run there were 4 major failures: (No override used)</p> <ul style="list-style-type: none"> • Missed blue corner goal (Rings in the way) • Missed blue alliance stake (Misaligned) • Missed second neutral stake (Misaligned) • Missed hang (Misaligned) <p>This resulted in a very low-scoring run, primarily because of missing hang</p>  <p style="text-align: center;">44</p> 	<p>Major failures:</p> <ul style="list-style-type: none"> • Misaligned goal grab caused missed wall stake • Missed blue alliance stake (Misaligned) • Missed hang (Misaligned)  <p style="text-align: center;">36</p> 
Driver Skills 2	Autonomous Coding Skills 2
<p>Major failures: (No override again)</p> <ul style="list-style-type: none"> • Hang did not release • Final goal didn't get placed in the corner (Rings rolled into the way) • Missed second neutral stake (Misaligned) <p>This again resulted in a very low-scoring run</p>  <p style="text-align: center;">44</p> 	<p>This run was relatively successful, only missing the hang because of the misalignment.</p>  <p style="text-align: center;">55</p> 

Driver Skills 3	Autonomous Coding Skills 3
<p>Critical issues: (Override needed)</p> <ul style="list-style-type: none"> • Hang didn't release  <p style="text-align: center;">56</p> 	<p>Another very successful run, other than an unreleased hang this run had zero major issues:</p>  <p style="text-align: center;">56</p> 

Summary

Hang		Wall Stakes	
Failure Reason	Count	Failure Reason	Count
Bad Release	3	Hardware	3 (Missed one ring)
Misalignment	3	Misalignment	3 (Missed both rings)

Conclusion

Because of the low reliability of the algorithms at this competition we believe that a significant investigation is required to determine the root cause of these inconsistencies. To do this testing we first need to fully implement the [ground truth localization method described earlier](#) (Pg. 172), and do rigorous tests to figure out the root of inaccuracies in the robot's movement throughout the skills run.

Additionally, we need to look into what caused the hang release errors as they did not happen at all when testing, and we were unable to recreate them. In the future, we will record our competition skills in 4K at 60FPS to allow for easier diagnosis of any other problems that we might have.

Finally, communicating about stop time with the skills ref allowed us to lower our risk of severely damaging our robot, especially when hanging, by introducing a legal way to terminate the run. This was used in all of our runs at the end to successfully prevent any damage to the hang.

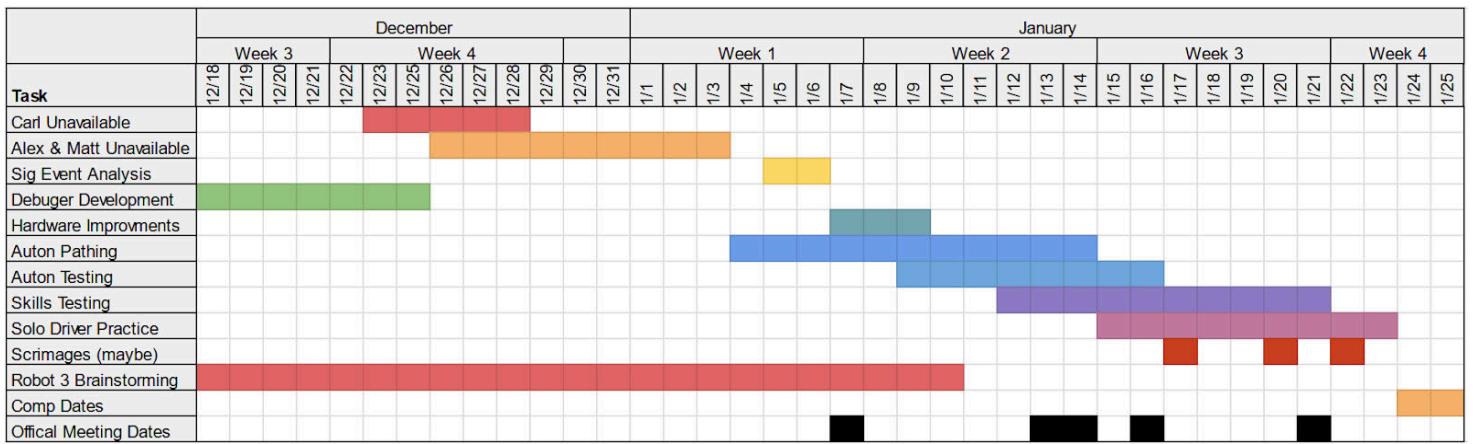
12/18/24 Time Management: Kalahari Timeline

Designed by: Carl	Witnessed by: Alex	Witnessed on: 12/19/24
-------------------	--------------------	------------------------

Goal: Create a rough outline of what tasks we need to complete before our next tournament.

Timeline:

The timeline we created is between now (12/18) and the Kalahari Classic Signature Event (01/24).



Rationale:

- We elected to start development of the debugger board as early as possible to hopefully have a completed version by the 7th (end of winter break).
 - Over break we are mostly unavailable and we won't have access to a field, so we can only do limited tasks.
 - Tasks after the 7th are condensed to allow plenty of driver practice
 - No big design changes are needed
 - Most of our skills and autos only require fine tuning
 - Scrimmages with other competitive teams are roughly planned out to allow for high level strategy refinement and practice driving against other robots

12/27/24 Identify/Design: Wireless Debugger Tool

Designed by: Alex

Witnessed by: Carl

Witnessed on: 12/27/24

Goal: Develop the system requirements for a wireless debugger tool and discuss the schematic and layout design.

This project's goal is to create a wireless debugging tool that will allow us to more easily look into what is happening on the robot during skills runs. This will improve debugging and the quality of life while trying to fix code issues on the robot.

Requirements

- Gets power from the V5 brain
 - Can be supplied through smart ports or ADI, but MUST not use an external power supply
 - This will allow easy use with the robot and a smaller form factor
- Utilize a connection protocol with the VEX brain
 - Fast connection (>200 Kb/s)
 - Allows us to get enough useful data from the robot to send over wireless communication
- Wireless capabilities
 - Must be able to send data at a rate of about 1MB/s
 - Enough bandwidth for some overhead for wireless connection

Ideate

In the VEX system there are 2 primary ways to communicate with the VEX brain, over the USB connection or with a smart port. Using either of these systems will meet the >200Kb/s requirement, however in the following analysis we will determine the best solution given our needs.

USB Serial Connection		Smartport Serial Connection	
Pros: <ul style="list-style-type: none"> ● High speed (912600 baud) ● Better documentation ● More difficult uC side computation 	Cons: <ul style="list-style-type: none"> ● Would need power from another source ● Would interfere with uploading 	Pros: <ul style="list-style-type: none"> ● Built in power ● High speed (921600 baud) ● Already have experience developing boards with smart ports ● Simple software 	Cons: <ul style="list-style-type: none"> ● More complicated electronically to figure out ● Little to no documentation on smartport communication

For wireless communication there are many options out there (thousands) but we have experience with the [ESP32-C6 MINI-1 from espressif](#) and this board will suit our needs very well, while still being very affordable. This board supports WIFI or bluetooth at well above the data rates we need, and has an accessible Arduino programming interface.



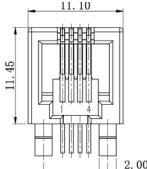
ESP32-C6-MINI-1

Primary Selection

For the communication protocol we think that the Smartport serial connection is the better choice for this board because of the built in power connection in the Smartport protocol. We believe that this will certainly outweigh the extra development time we will spend to develop a board with a smartport as no other power solution would have to be explored and implemented on the robot.

Component Selection

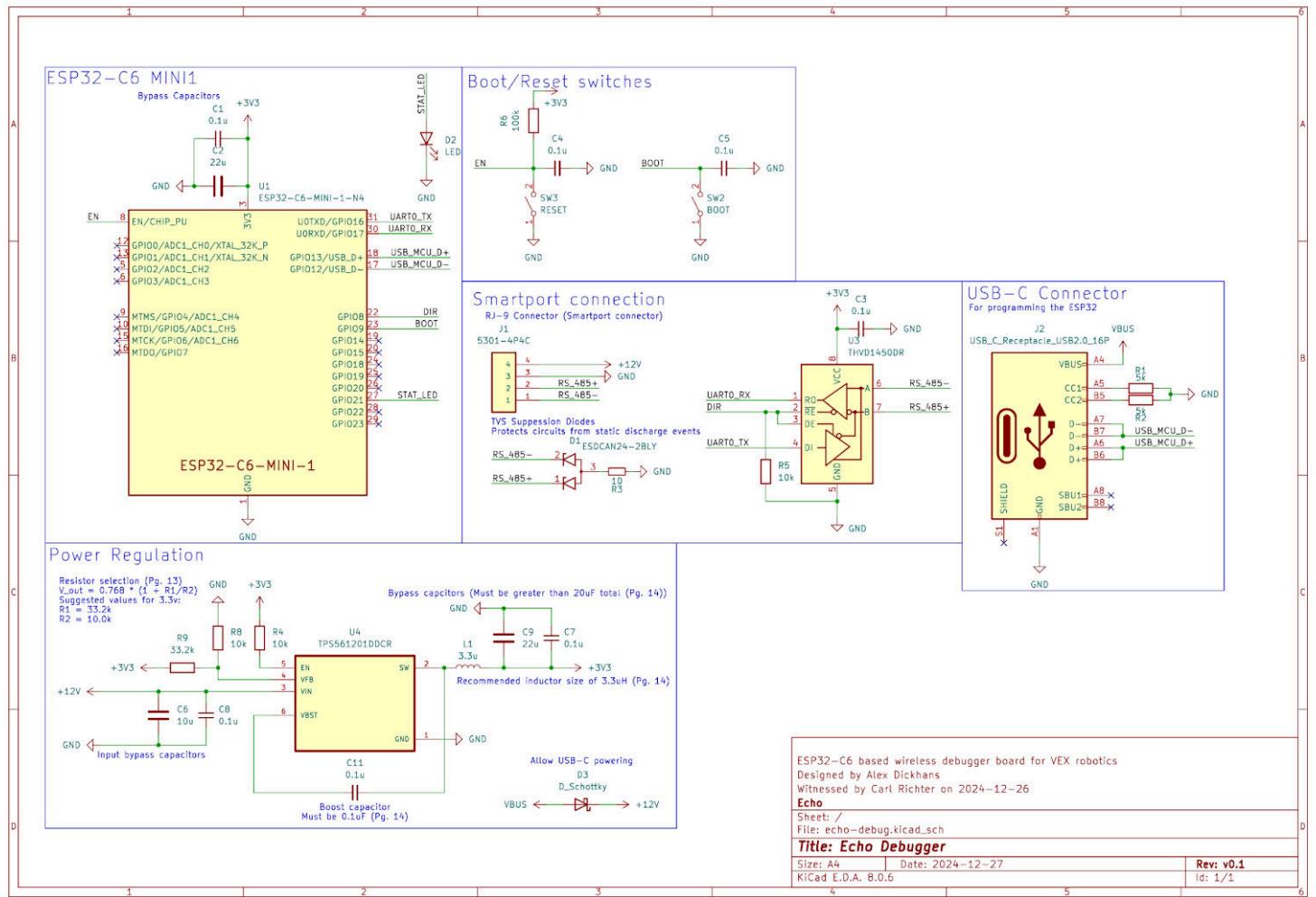
The critical first step in PCB design is the selection of components. This process is where the PCB designer is able to specify how they want the PCB to work with general terms in the parts chosen. There are 27 individual parts on this PCB including passive components so we won't list every part, but the larger components will be listed and explained.

Part	Selected Part	Justification
Voltage Regulation	 TPS561201DDCR	This step-down voltage regulator will allow us to step down the 12V input voltage from the smartport connection down to the 3.3V that is required for the ESP32 and the RS-485 transceiver. It is over 90% efficient at all reasonable current requirements for our board and will not dissipate too much heat.
Smartport RJ-9 Connector	 5301-4P4C	This port will allow for the board to connect physically to the smartport wire from the brain. It has the capabilities to support the current from the brain (about 200mA max) without overheating.
RS-485 Transceiver	 THVD1450DR	This component allows for the ESP32 to read and send data over the RS-485 smart port bus. This takes the differential higher-voltage signal from the RS-485 connection and makes it into a serial UART connection that the ESP32 can easily read.

Transient Voltage Suppression (RS-485)		Transient voltage suppression utilizes diodes to dissipate high-voltage spikes common with static electricity. The selected component is designed for CAN bus and RS-485 connections and has a small package, which fits well in many designs.
	ESDCAN24-2BLY	

Schematic Design

We will be developing the Printed Circuit Board (PCB) for this project in [KiCAD](#). This is an Electronic Design Automation (EDA) program that allows for the development of electrical designs with PCBs. The first step in any PCB design process is to specify the schematic. The schematic details which electrical connections are wired between components.



PCB Layout

The next step in PCB design is to actually lay out the placement of the components and the physical wires between them. The first step in this process is choosing a PCB stackup or specifying the actual inner layers in a PCB. In our design we chose to use a 4 layer design with an internal power and ground plane with external signal layers. This design maintains cost-effectiveness while allowing for more flexibility for the connections to power and ground with lower signal emittance. There is not enough room here to discuss all the choices made in the PCB layout process, for brevity, we will show the most important parts here:

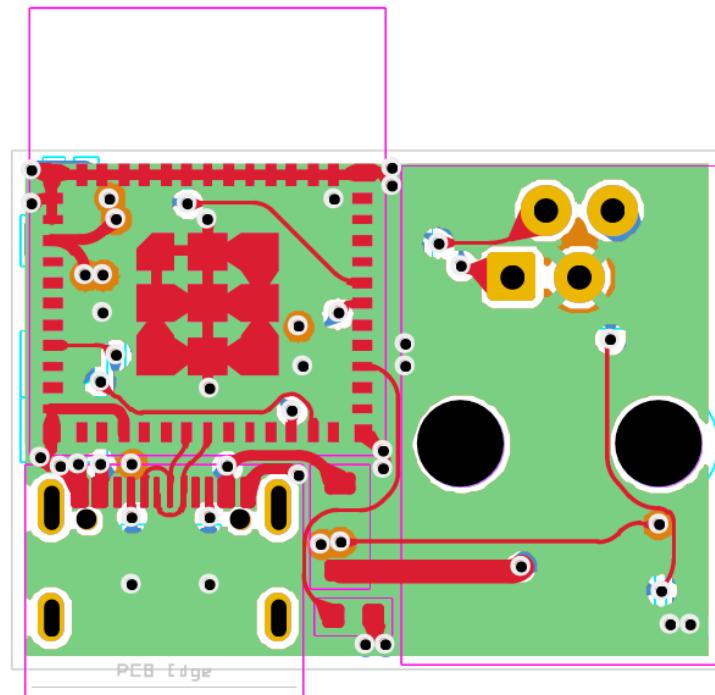


Figure: PCB Design with All Layers

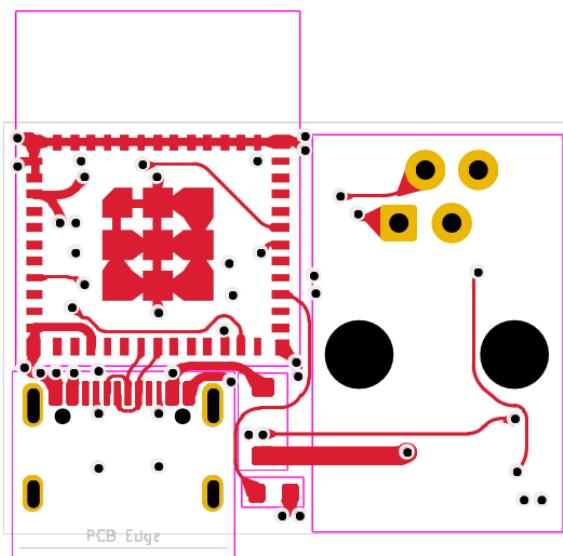


Figure: Only Top Layer Copper

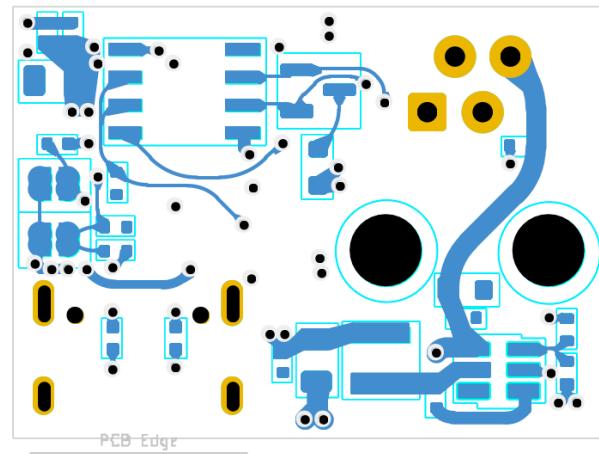


Figure: Only Bottom Layer Copper

Manufacturing Renderings:

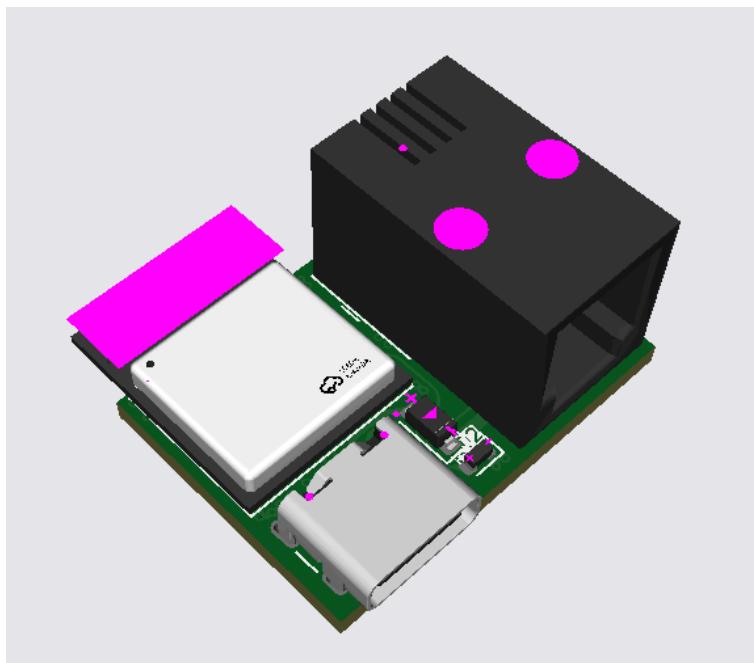


Figure: Isometric View

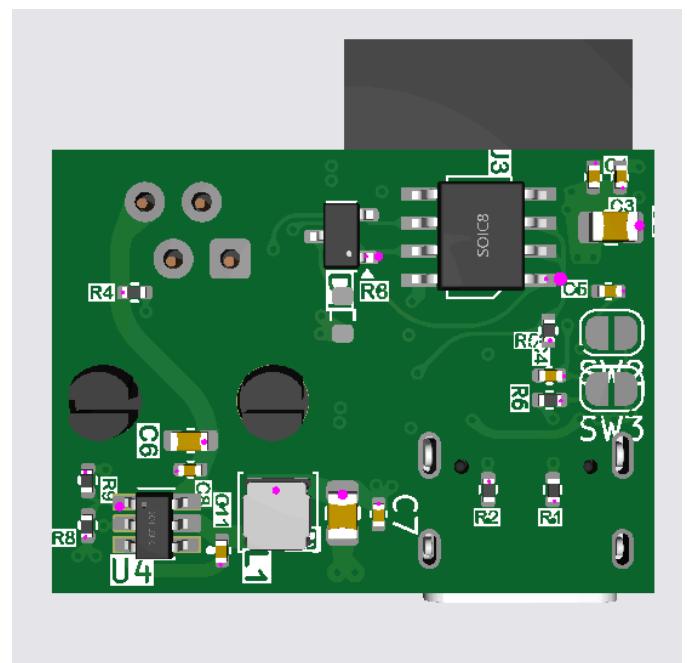


Figure: Bottom View

Manufacturing

Manufacturing PCBs yourself is not feasible, especially for this size and layer count would be near impossible to manufacture yourself. For this reason we sent out this PCB to the [JLCPCB](#) PCB manufacturing house to have assembled. We expect to have the new iteration of the board by the end of January.

01/05/25 Analysis: Sugar Rush Signature Event

Designed by: Carl	Witnessed by: Matt	Witnessed on: 01/06/25
-------------------	--------------------	------------------------

Goal: Watch Sugar Rush elimination matches to determine what good high level strategies might be at Kalahari. Make note of unique designs or autonomous routines.

Notable Designs:

The “Lady Brown” mechanism (Pg. [203/256](#)) was the most prominent design at the competition by far, with around 90% of the teams that made it to eliminations utilizing it. It proved to be consistent at scoring on wall stakes and had some descoring abilities, however it still took around 3-5 seconds to score 2 rings. There were also some teams that shortened their arm to be able to rotate 180°, allowing them to use it to score on alliance stakes and remove goals from the corner easier. It is also worth noting that this modification slightly reduced the speed at which they could score.

No teams at the event had higher than a tier one hang, meaning that if we were able to use our high hang in a match, it would introduce another way to win, also known as an additional “win condition”. We would likely only need to hang in a match if we did not have at least two of the other win conditions: 3 goals, top ring on both wall stakes, and auto win.

Notable Strategies:

- Screening
 - This was a common strategy that was used at the beginning of competitive matches
 - To get control of the third goal, one alliance robot would block the small gap between the wall stake and hang structure, restricting both opposing robots
 - This allowed the other alliance robot to put a goal in the corner and secure the third goal
- Protecting 2 goals in the corner
 - This was a very effective way to keep control of three goals, helping to guarantee the win
 - This strategy was also used when an alliance had only two goals, allowing one robot to score wall stakes (without the risk of losing a goal) or attempt to steal a goal
- Last second descore
 - Putting the opponents goal in the negative corner in the last 3-5 sec

Overall, we will not need to make many significant changes to our strategy as long as we are careful and “play safe”. Consistent autons will also be extremely important due to the high correlation between auton and match wins and the fact that we can play more conservative if we have auton win. High hanging is also something that can be used, but it should not be necessary in most conditions.

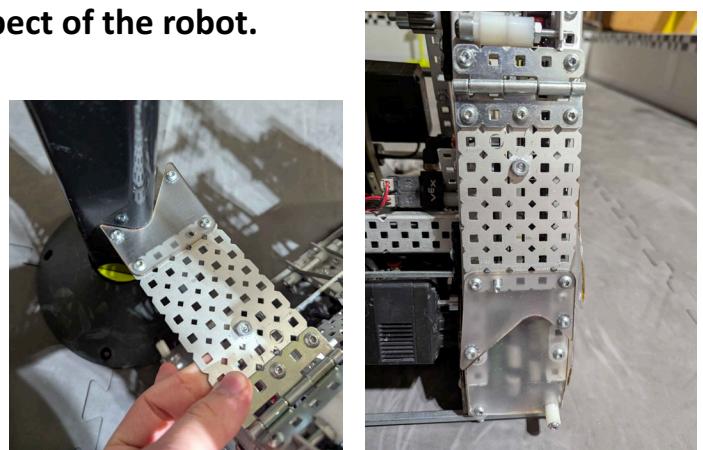
01/07/25 Build: Hardware Improvements (R.2.1.17)

Designed by: Carl	Witnessed by: Alex	Witnessed on: 01/08/25
-------------------	--------------------	------------------------

Goal: Address minor issues with the hardware aspect of the robot.

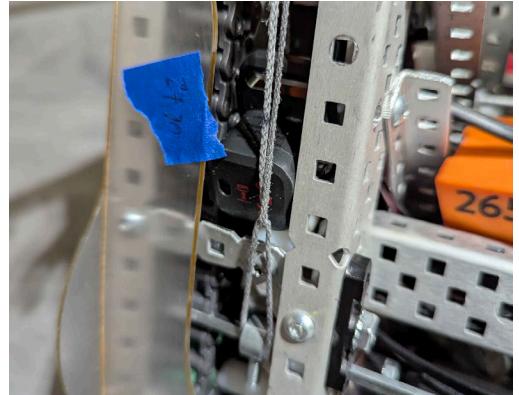
Modified Pole Aligner:

At our last tournament and during testing, we observed that the accuracy demanded from the driver or program was too high to achieve a quick or reliable hang. To solve this, we re-designed the plastic part on the aligner to have a wider angle, allowing for a position variance of $\pm 1"$. This modification was tested thoroughly and did not negatively interfere with any other subsystems.



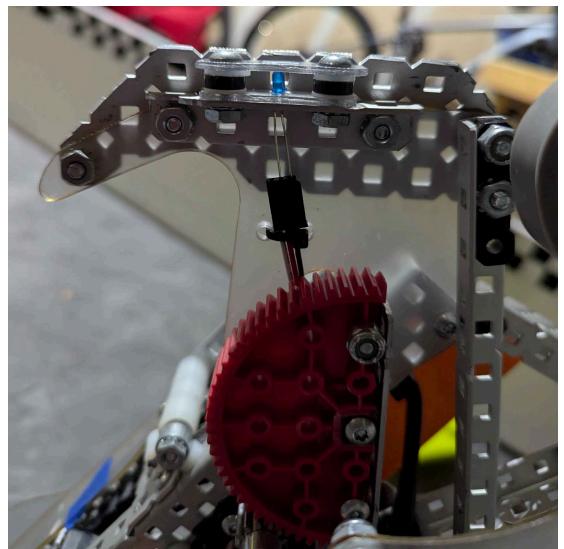
Lowering Rear Distance Sensor:

Our original mount for the rear distance sensor was around 11.5" above the ground, just 0.5" below the top of the wall. We believe the distance sensor may have failed to read the wall correctly, possibly due to the robot tilting, and relocating it to a lower position would eliminate the chance of those errors. There was very limited space for this sensor and we had to mount it between the upright and intake chain.



LED for Overhead Camera Tracking:

As described [here](#) (Pg. 172), we want to use an LED in combination with an overhead camera to record the exact position of the robot for easier tuning of skills. The LED was mounted with a custom polycarbonate part and secured on the inside of the hang hook. This LED is technically a nonfunctional decoration under <R9>, so we will not need to remove it during matches.



01/09/25 Testing: Autonomous Error Analysis

Designed by: Alex

Witnessed by: Carl

Witnessed on: 01/10/25

Goal: Test and diagnose errors in the autonomous routine identified in our [Kent Denver Post-Competition Analysis](#) (Pg. 322-323).

Initial Testing

Before going through the lengthy process of testing against the ground truth localization, we wanted to first test skills by itself to attempt to find and diagnose specific issues. We have clues from previous runs that there are serious issues with our current localization because of large jumps in the particle belief over time. To accomplish this testing we will run with a wired connection to the robot and collect the desired position of the robot from motion profiling and the localization prediction of the robot. This will allow us to analyze the localization and additionally later tune RAMSETE.

Localization Tuning

For our initial runs we were looking for major localization hiccups and inaccuracies. In the following graph we have only the localization readings live from the robot graphed on the field.

In this run we saw several major discrepancies in the localization. First, after the robot grabs the goal, highlighted below, there are a few significant wobbles in the robot's state belief.

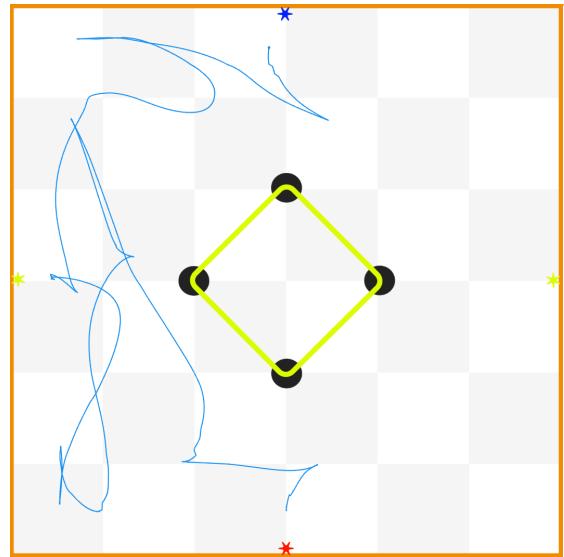


Figure: Initial Localization Data Run #1

Figures: Illustrates the “wobbles” in the state belief

Additionally, not illustrated in the testing were key points that were in inaccurate positions. While scoring the wall stakes, the robot appeared to be perfectly aligned, however, in reality, the robot was fairly far off. These severe issues caused us to reevaluate where the sensors were in the code. Upon inspection, we found the back distance sensor was specified to be on the right side of the robot, when in reality it was on the left side. After testing with this change we found a significant reduction of deviations.

This run was significantly better and most of the major jumps we saw were effectively eliminated. However, we did notice a few smaller jumps that weren't obvious in the previous run. These issues seemed to stem from the robot not converging on the correct belief fast enough. To correct this we believe that having the robot update the state belief every frame, instead of after a specified distance and trusting the distance sensors on the robot more strongly (Standard deviation multiplier decreased from 2.0->1.4) would improve the errors we saw in this run.

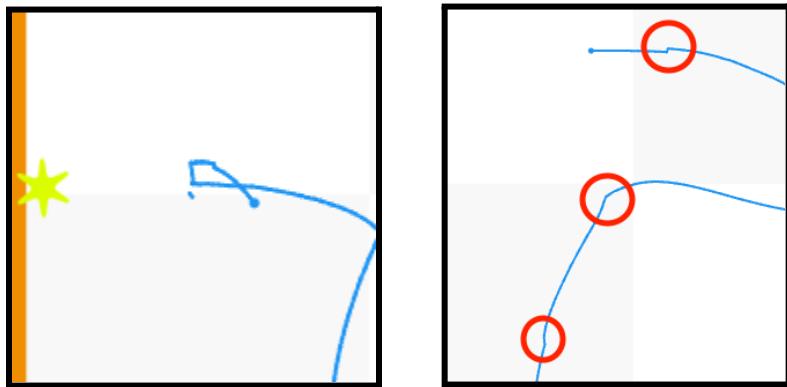


Figure: Showing subtle jumps in the run #2

In the post testing (shown left) there were a few minor issues, however, none amounted to more than an inch throughout the entire run, and all key points on the path are accurate.

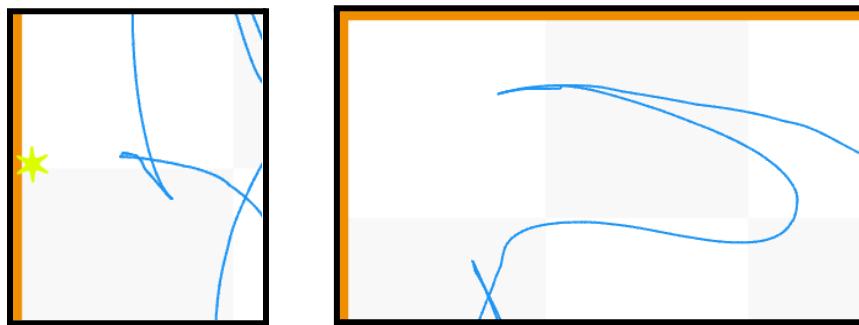


Figure: Illustrates elimination of jumps present in last run

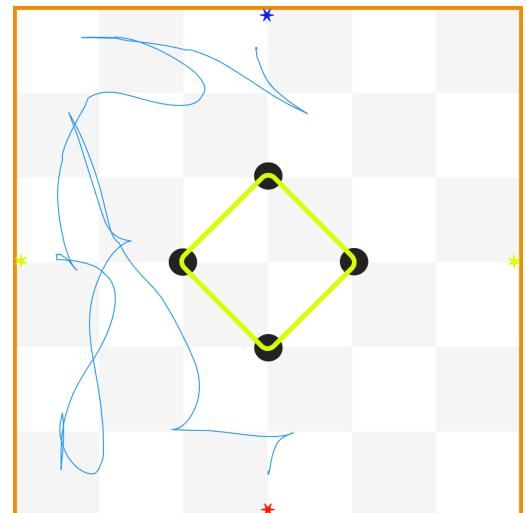


Figure: Localization Data Run #2

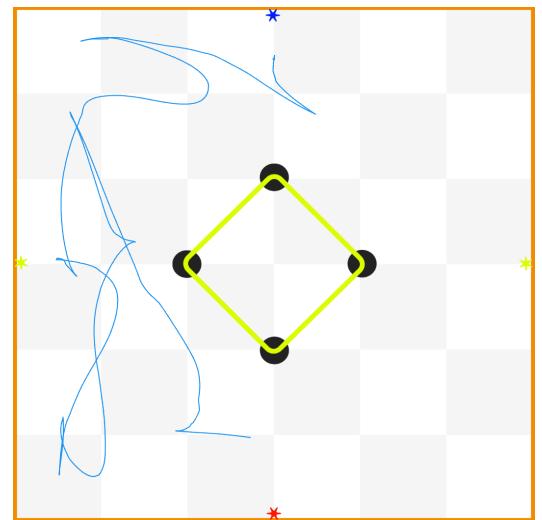
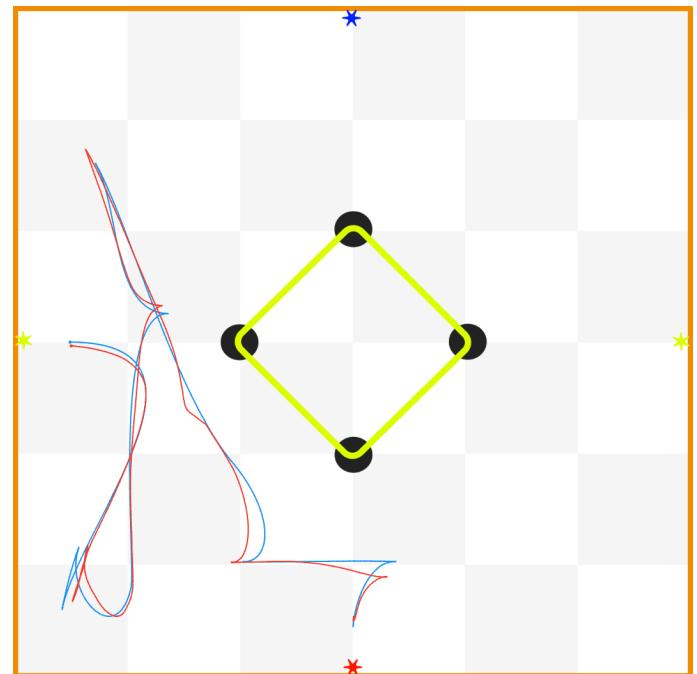
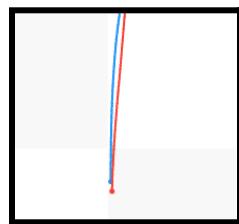


Figure: Localization Post-Testing

RAMSETE Tuning

Initial Run:

In our initial testing run RAMSETE had significant tracking difficulties, shown in the debugger utility graph. We believe that these issues were caused by an overly high kV tuning variable. This caused the robot to always be slightly ahead of the target point for RAMSETE, which leads to high tracking errors on paths such as the high-curvature movements in this path. To fix this, we lowered the kV tuning parameter from 0.6 to 0.5, causing the robot to match, or very slightly trail the target point



*Figure: Initial RAMSETE Testing
Target illustrated in blue, localization in red*

Final Run:

In our final run we saw better path tracking over the entirety of the path and especially at key points as the robot was more closely tracking the target point, not leading it.

Conclusion

Initial testing with the localization demonstrated obvious errors in localization (large jumps and obvious errors). Through diligent tuning we were able to limit these issues significantly, and after RAMSETE tuning are now comfortable with the reliability that this solution provides.

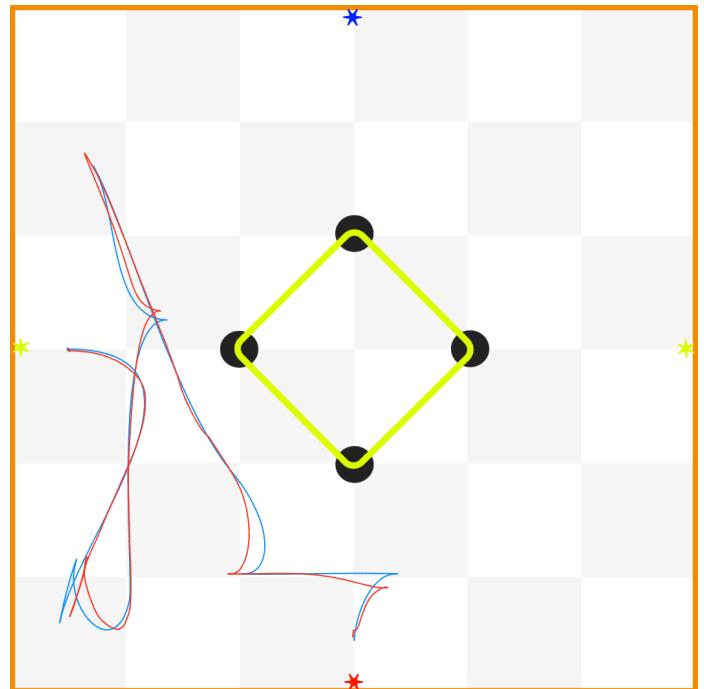


Figure: Post-Adjustment RAMSETE Testing

01/10/25 Testing: Skills Tuning Day 1 (R.2.1.17)

Designed by: Alex

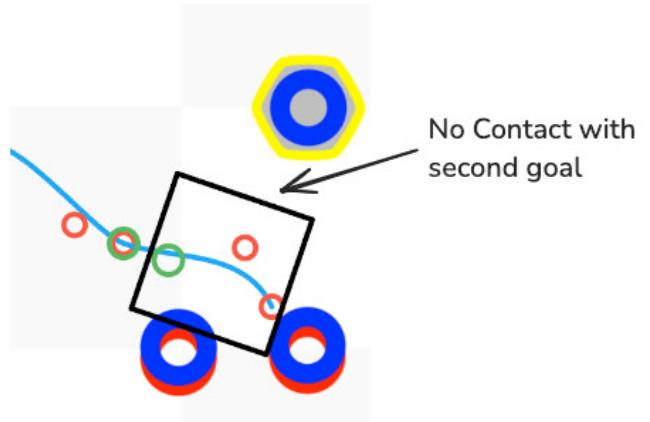
Witnessed by: Carl

Witnessed on: 1/10/25

Goal: Test and diagnose errors in the autonomous coding skills routine.

Pathing

After fixing the inaccuracy [issues we saw at the Kent Denver Competition](#) (Pg. 332-334), the skills testing went smoothly with only very minor path adjustments. The largest path adjustment we had to make was at the end of the path where the robot had issues where it would hit the goal, causing a slight double possession violation. We adjusted the path to steer well clear of the goal.



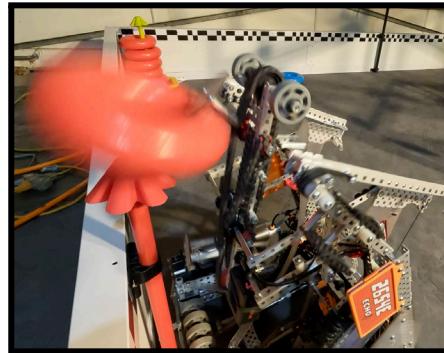
Alliance Stake

In our runs we had many times where the robot was well aligned for the alliance stake but the wall stake scoring method we used caused the ring to miss up to 50% of the time, without a pattern to when it missed. After exploring different ways to score on the alliance stake we would like to continue to score from the front of the robot.

Trial	New Scored	Old Scored
1	Yes	No
2	Yes	No
3	Yes	Yes
4	Yes	Yes
5	Yes	No



Old



New

Summary

- Changed path ending to avoid hitting and double possessing a goal
- Modified the alliance stake to increase consistency of scoring
- Ended the day with a scoring several **69** point runs, our current goal for this run ([video](#))

01/12/25 Strategy: Kalahari Autonomous Planning

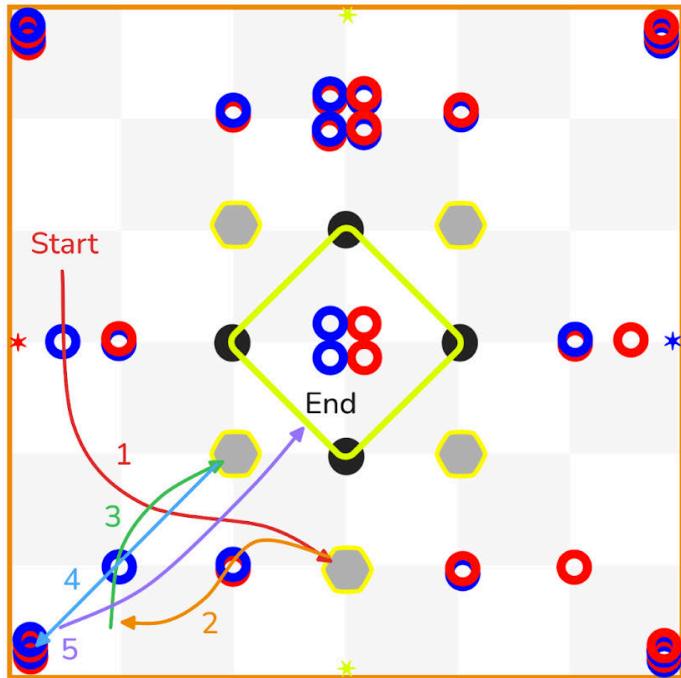
Designed by: Carl, Alex

Witnessed by: Matt

Witnessed on: 1/13/24

Goal: Develop autonomous routines specifically tailored to the high level match play we expect at Kalahari.

3 Stake Solo AWP Idea One (Not Used):



Path 1 (Reverse): Passively scores ring on the alliance stake, proceeds to grab mobile stake on the center line

Path 2 (Forward): Intakes and scores red ring, drops goal at the end

Path 3 (Reverse): Begins by turning 90°, pushes blue ring away and grabs goal

Path 4 (Forwards): Drives towards the corner

Path 5: Cycles rings out of the corner and then touches the hang structure

Total Score: 10 points

- 4 rings
- 3 top rings

Pros:

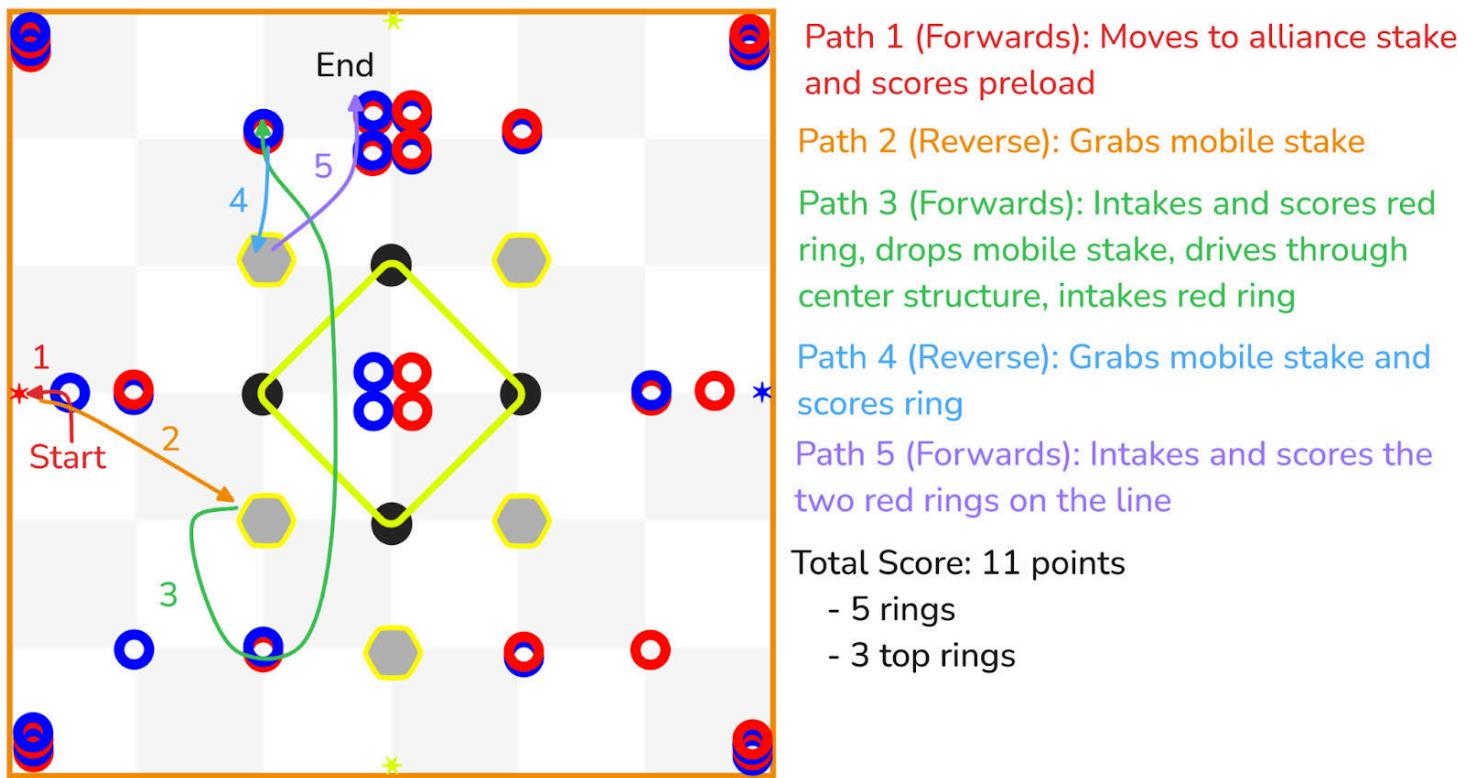
- Allows our alliance partner to run an autonomous
- Suitable for elimination matches
- High scoring

Cons:

- Robot would be extremely close to crossing the centerline when grabbing the first goal
- Bad setup for matches
 - Only 1 or 2 rings on the each goal
 - Getting more rings would require a significant amount of driving
- To score the alliance stake, we would need to add a passive mechanism to the side of our robot, similar to this one (pictured above on the right) by 3658D



3 Stake Solo AWP Idea Two (Not Used):



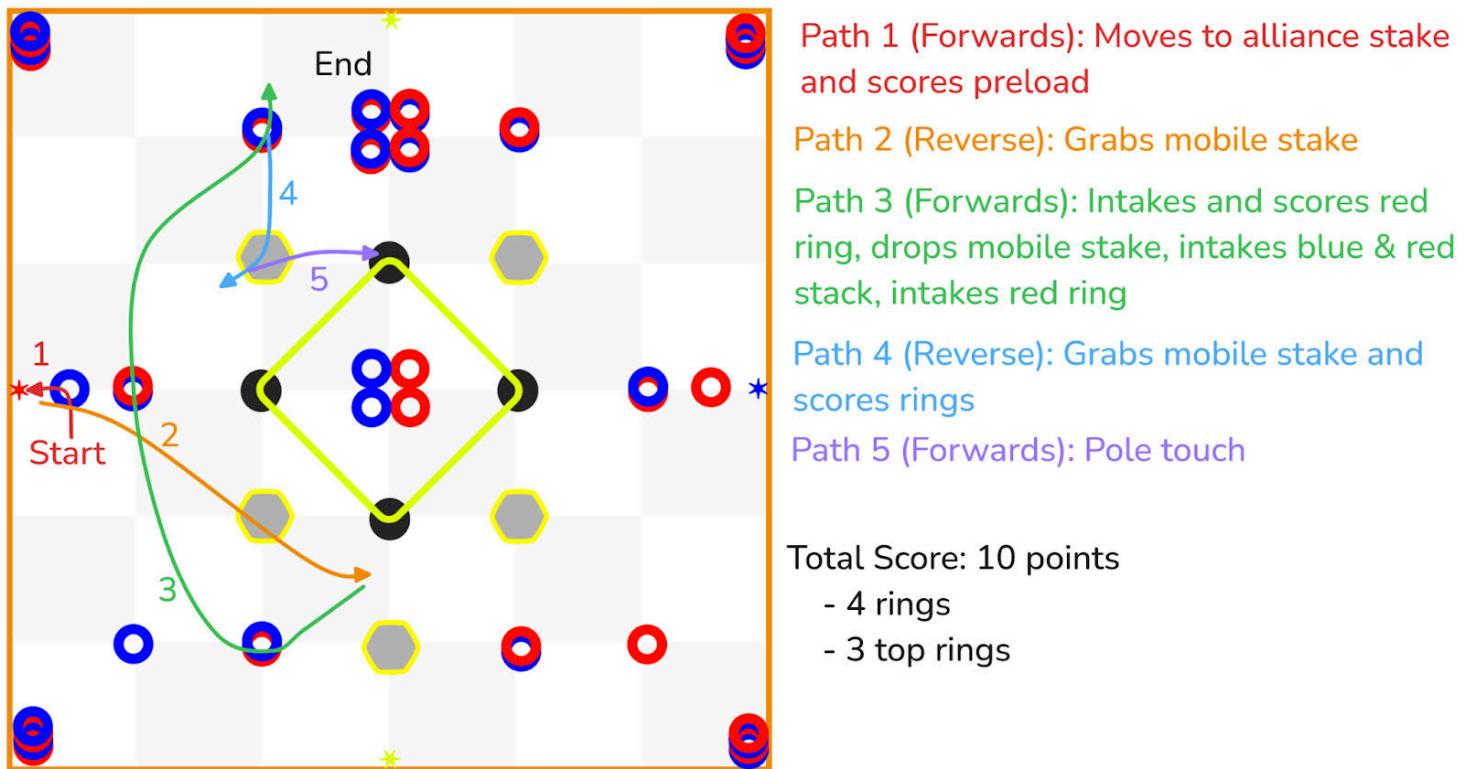
Pros:

- Very high scoring
- Good setup for matches
 - Three rings on the goal that we end with
 - Three more rings (plus alliances preload) on the way to or in the nearest positive corner

Cons:

- Would likely miss AWP if the center rings got knocked over
 - This situation is likely as these rings are the last thing we get
- Relatively inefficient route
 - Movements would need to be extremely fast to avoid running out of time
 - This might cause inconsistencies
- Only can be used if our alliance partner doesn't have an auton

3 Stake Solo AWP Idea Three (Used):



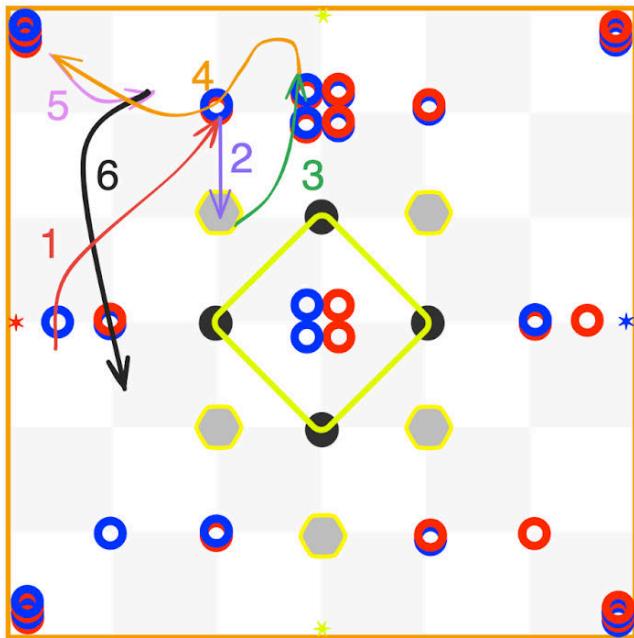
Pros:

- High scoring
- Good setup for matches
 - Two rings on the goal that we end with
 - Close to several other rings
- Very efficient path
 - We will likely have time to also get at least one ring from the center line
 - This will NOT effect if we get AWP or not
- Very safe for AWP tasks

Cons:

- Only can be used if our alliance partner doesn't have an auton
- Slightly lower score than Idea Two
 - If we also grabbed a ring (or two) from the centerline we could score the same or maybe even higher

Negative Side Auton (Used):



- Path 1 (Forward): Score on Alliance Stake
- Path 2 (Reverse): Grab Mobile Stake
- Path 3 (Forward): Score on Mobile Stake
- Path 4 (Forward): Drive to Corner
- Path 5 (Reverse): Leave Corner
- Path 6 (Forward): Score Mobile Stake

Total Score: 11

- 7 Rings
- 2 Top Rings

Pros:

- High scoring
- Good setup for matches
 - Six rings on the goal that we end with
- Very efficient path
 - We will likely have time to also get at least one ring from the center line
 - This will NOT effect if we get AWP or not
- Very safe for AWP tasks AND alliance stake

Cons:

- Doesn't get to the rings on the line immediately
 - Those could get messed up by the opponent
- Starting position may conflict with teammates

Summary:

- Two new autonomous routines: Full AWP (for signature events) and a new negative side
- The new negative auton will replace our old one (Pg. 295)
 - We will need to design and build a mechanism to score the alliance stake on the robot's side
 - The path is more efficient meaning we can drive slower to increase reliability
- The full AWP will score four rings over three stakes, achieving a fully solo AWP
 - This will be extremely important in qualis if our alliance partners do not have fully reliable autos
- For positive side, we plan to keep with our current path as it already gets all available rings reliably

01/14/25 Build & Test: Passive Ring Mechanism (R.2.1.18)

Designed by: Carl

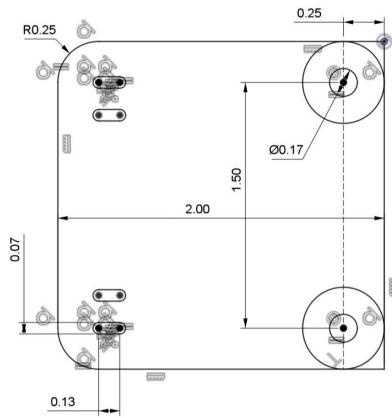
Witnessed by: Alex

Witnessed on: 01/14/25

Goal: Design, build, and test a passive ring scorer.

Design and Build:

The inspiration for this design comes from 3658D who used polycarbonate with a grippy surface (rubber bands) to hold the ring. Our adaptation is designed to use a similar polycarbonate part (pictured left) with cutouts for attaching the grip. We found that mesh was very effective at holding the ring given enough compression, so we chose to use it as the grip.



The passive holder was mounted directly to the bottom hang arm as that allowed for adjustable positioning of the mechanism through the motion of the lift. After some brief testing , the assembly was mirrored to the other side of the robot to accommodate use on both alliance stakes.

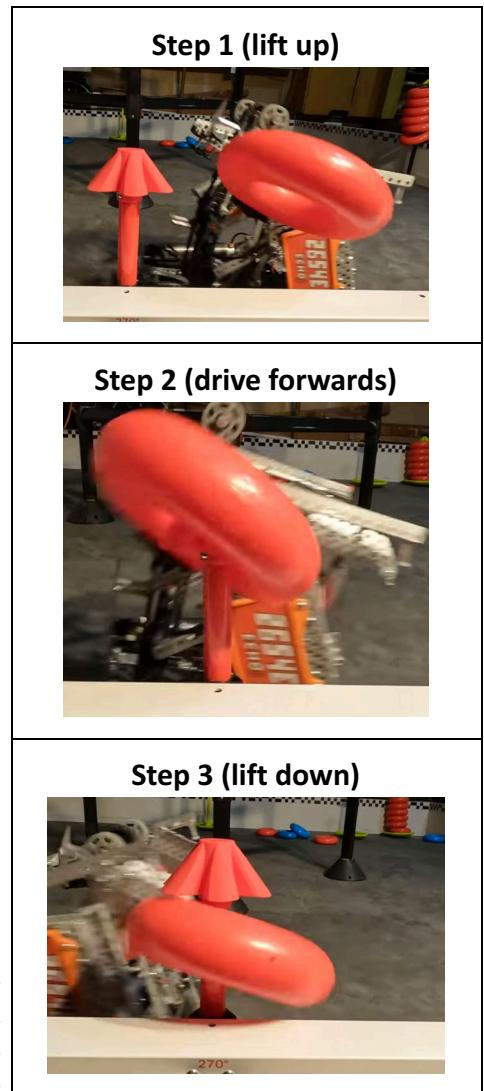
Testing:

Method:

1. Run the code at the beginning of autonomous
2. Record if the ring was successfully scored

The entire point of this mechanism was to increase the reliability and speed of alliance stakes, so, after optimizing the time of the scoring sequence as much as possible (~0.75 sec), we conducted 15 trials to ensure consistency.

Trial #	Scored?
1	Yes
2	Yes
3	Yes
4	Yes
5	Yes
6	Yes
7	Yes
8	Yes
9	Yes
10	Yes
11	Yes
12	Yes
13	Yes
14	Yes
15	Yes



01/20/25 Testing: Kalahari Skills Tuning

Designed by: Alex, Carl

Witnessed by: Matt

Witnessed on: 01/21/25

Goal: Test and diagnose errors in the autonomous coding skills routine.

To prepare for Kalahari, we spent extensive time tuning out issues (primarily those listed below) to give ourselves a higher chance of hitting a world record at the tournament. We also collected data that demonstrates a relatively high level of reliability compared to our [pre Kent Denver skills tuning](#) (Pg. 320).

- Intaking in the center structure
 - Because of bad intake timing the ring would stop at a position that would cause the robot to get stuck in the center structure
 - Going through the center slower eliminated the issue
- Wall stake scoring issues
 - First ring would get stuck around the stake caused the second ring to get deflected back into the robot
 - Ring occasionally got misindexed
 - Fixed by correcting bottom distance sensor position



Figure: Robot getting stuck in the center due to rings

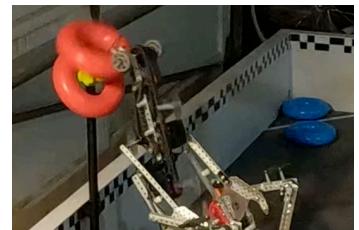


Figure: Rings getting deflected from scoring on the wall stake

Reliability Data

This data was collected over the course of 22 skills runs that took place from January 19-20. It provides a holistic view of weaknesses and strengths in the program while also gives us a baseline that we can compare other runs to.

Skills Testing 01.19.2024		Notes: In the event of a catastrophic hardware or miscellaneous software failure, the run was terminated immediately and "Ended Early" was selected for those runs. If a catastrophic failure resulted from missing one of the tasks listed below, the failed task received the "Missed" distinction and all subsequent tasks received "Ended Early".									
#	Trial	Alliance 1	Mobile Goal 1	Wall 1	Goal 2	Alliance 2	Mobile Goal 3	Wall 2	Mobile Goal 4	Hang	
1	Scored	Good Grab	All Rings	In Corner	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Missed
2	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored
3	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored
4	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored
5	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored
6	Scored	Good Grab	In Corner	All Rings	Missed	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored
7	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored
8	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored
9	Scored	Good Grab	In Corner	All Rings	Missed	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored
10	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored
11	Scored	Good Grab	In Corner	Some Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	Some Rings	Aligned + Scored
12	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	Some Rings	Aligned + Scored
13	Scored	Good Grab	In Corner	All Rings	Missed	Not in Corner	Ended Early	Ended Early	Ended Early	Ended Early	Ended Early
14	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored
15	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	Some Rings	Missed
16	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	Not in Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored
17	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	Some Rings	Aligned + Scored
18	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored
19	Scored	Good Grab	In Corner	Some Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored
20	Scored	Good Grab	In Corner	All Rings	Missed	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Missed
21	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored
22	Scored	Good Grab	In Corner	All Rings	Aligned + Scored	In Corner	Aligned + Scored	Good Grab	In Corner	All Rings	Aligned + Scored

01/22/25 Testing: Kalahari Autonomous Tuning

Designed by: Alex, Carl

Witnessed by: Matt

Witnessed on: 01/23/25

Goal: Diagnose errors and evaluate consistency in our [Kalahari autonomous routines](#) (Pg. 336-339).

Alliance Stake Changes

After adding the [passive alliance stake mechanism](#) (Pg. 340), for our negative side auton, we elected to adapt the rest of our routes to utilize this mechanism to increase reliability and time utilization.

Testing Method (For each path)

1. Tune path until the path would be ready for comp
2. Test the path 10 times for AWP paths and 5 for elimination paths
3. Record data on consistency
 - a. AWP objectives - does the run get the AWP objectives it's supposed to

Solo Signature AWP (SAWP)

After some initial testing, we found that we had around 3 seconds remaining after completing the [original path](#) (Pg. 338)

which allowed us to add the two rings on the center line to our path, and even if the other alliance were to mess these up, we would still have enough rings for an AWP.

SAWP	Scored Rings	Top Rings	AWP Objectives?	Tr	Notes
1	5	3	YES		
2	4	3	YES		Missed ring near the autonomous line
3	5	3	YES		
4	5	3	YES		
5	5	3	YES		
6	4	3	YES		Missed the same ring (maintaining safe distance from line)
7	2	2	NO		Missed second goal grab because of rolling ring
8	5	3	YES		
9	5	3	YES		
10	4	3	YES		Missed same ring on autonomous line

Positive Safe

For the positive routine there were no deviations from the plan besides the alliance stake to implement with the hardware changes. Although this autonomous routine doesn't achieve all of the objectives, it proves to be effective at the ones it does achieve.

Positive	Scored Rings	Top Rings	AWP Objectives?	Tr	Notes
1	4	2	YES		
2	3	2	YES		Corner rings fell down
3	4	2	YES		
4	4	2	YES		

Negative	Scored Rings	Top Rings	AWP Objectives?	Tr	Notes
1	6	2	YES		Corner rings fell down
2	7	2	YES		
3	6	2	YES		Corner rings fell
4	6	2	YES		Corner rings fell
5	7	2	YES		

Negative Elim

For the negative elimination path there were no deviations as this autonomous already was designed to utilize the new mechanism. This auton proved to be very consistent with the autonomous objectives that it achieved.

01/28/25 Analysis: Kalahari Matches (01/24-25/25)

Designed by: Matt, Carl	Witnessed by: Alex	Witnessed on: 02/01/25
-------------------------	--------------------	------------------------

Goal: Review our Kalahari matches to identify strengths and weaknesses.

Qualification Matches:

Match	Final Score	Win Point	Auton Win	Notes
Q3	19-45	Yes	Yes	Won auton and got third goal, alliance stayed in the positive corner, we had both top rings on wall stakes. Messed up grabbing third goal initially.
Q25	38-19	No	Yes	Alliance missed AWP task but still won auton, got 2 goals while alliance camped the corner. Played wall stakes and got top ring on one of them. Alliance put their goal in the negative in the last 5 seconds.
Q63	38-35	Yes	Yes	Ran solo AWP and won auto, Lost third goal after having it, played wall stakes while alliance camped corner, Got the top ring on one wall stake.
Q77	23-44	Yes	Yes	Got third goal after autonomous, alliance camped corner while we played wall stakes and got both top rings.
Q89	39-43	No	No	Teammate missed AWP, lost autonomous, got third goal after, deployed hang early accidentally so we couldn't do wall stakes. Hung in the last 20 second and won the match due to it.
Q122	29-30	Yes	Yes	Didn't have third goal, our alliance blocked the other positive corner so we couldn't do wall stakes. Didn't have any top rings so we lost the match.
Q138	38-25	No	Yes	Teammate missed AWP. Messed up opposing alliance when they were trying to manage third goal, the goal tipped so no one got it, alliance camped corner while we did wall stakes, ring fell in robot so we lost control of both top rings.
Q159	29-35	Yes	Yes	Lost third goal, alliance camped corner with 2 goal, we played wall stakes and got both top rings.

Elimination Matches (R16 1-1):

After winning quals, we were able to alliance with 2145Z to form the first seeded alliance. Despite this, we lost our first elimination match due to a combination of strategic, communication, and autonomous errors.

Original Strategy (created after alliance selection):

- Achieve early control of two goals and at least one positive corner
- Play safe: 2145Z holds positive corner, we do wall stakes
- T3 hang at around 20 seconds, 2145Z switches to wall stakes

Major Failure Points:

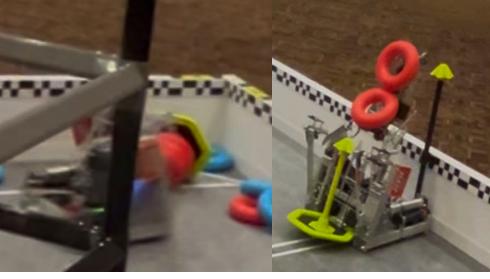
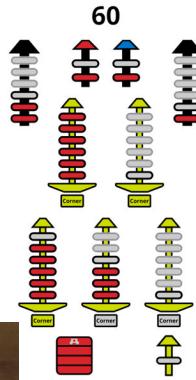
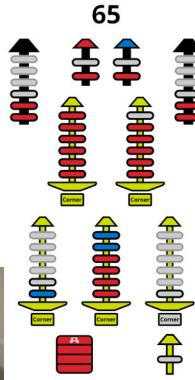
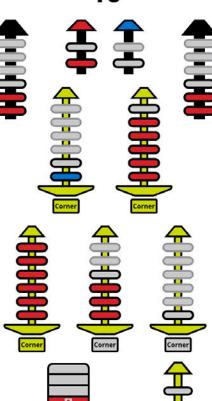
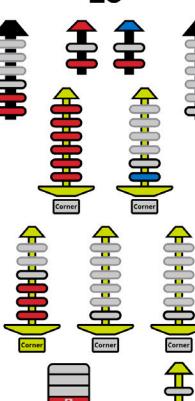
- 2145Z scored top ring for opponents in auto resulting in an autonomous loss
- With this point deficit, our alliance abandoned the positive to attempt a risky play, however, we failed to cover the positive corner after they left resulting in a descore by the blue alliance
 - After the match we discovered that this was an unnecessary play considering the extra 12 points provided by our T3
- Overall, a few small mistakes by both us and our alliance spiraled out of control quickly resulting in a big point deficit that we could not recover
 - We need more practice in more competitive matches where ALL teams have very capable robots in order to prevent these situations in the future

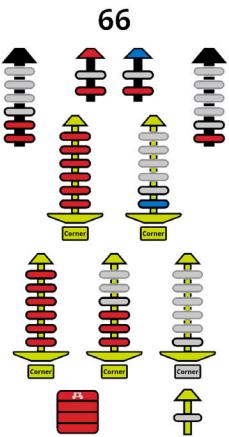
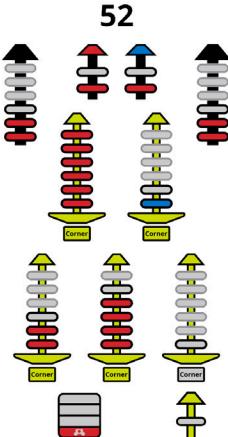
01/29/25 Analysis: Kalahari Skills (01/24-25/25)

Designed by: Alex	Witnessed by: Carl	Witnessed on: 02/01/25
-------------------	--------------------	------------------------

Goal: Examine our skills from Kalahari and determine our strengths and weaknesses.

At Kalahari we had our second time doing skills on this robot, and the first one with renewed localization to better account for distance sensor “size” variable variations on metal and plastic fields.

Driver Skills 1	Autonomous Coding Skills 1 (World Record)
<p>In this run there was 1 major failure in programming: (Override used)</p> <ul style="list-style-type: none"> Missed second wall stake due to misalignment <p>Driver:</p> <ul style="list-style-type: none"> Hooked goal in the corner, losing 5 points  	<p>Major failures:</p> <ul style="list-style-type: none"> Missorted ring (Caused missed top ring) <p>Minor issues:</p> <ul style="list-style-type: none"> Missed ring on the second goal fill  
Driver Skills 2	Autonomous Coding Skills 2
<p>Major failures: (No override)</p> <ul style="list-style-type: none"> Missed ring for second alliance stake Ring jammed at end of path <ul style="list-style-type: none"> Caused missed rings Led to missed hang Hang alignment way off  	<p>Single point of failure caused low score:</p> <ul style="list-style-type: none"> Blue goal got blocked from the corner by an extra ring Caused wheel slip, which resulted in missing goal, corner, and alliance stake Hang missed  

Driver Skills 3 (World Record)	Autonomous Coding Skills 3
<p>Critical issues: (Override needed)</p> <ul style="list-style-type: none"> • Missed a ring on last wall stake • Lead to an intake jam <p>Driver:</p> <ul style="list-style-type: none"> • Jam quickly fixed but missed a couple rings to maintain time in path  <p style="text-align: center;">66</p> 	<p>Critical Issues:</p> <ul style="list-style-type: none"> • Missed hang • Messed up rings at the end of the path  <p style="text-align: center;">52</p>  

Analysis:

- Beginning of the path
 - Absurdly consistent
 - Minor issues with the intaking of the red ring before the goal push, but quickly resolved
 - Occasionally had wheel slip on blue ring goal while putting it in the corner
- Middle of path
 - Very good
 - One issue area where rings commonly missed
- End of path
 - Seemed to be localization issues in 2 runs but otherwise absurdly consistent

Problem diagnosis:

- Wheel Slippage
 - In multiple runs the tracking accuracy of localization suffered a serious drop
 - This was often after hitting an object such as a neutral stake or pushing a mobile stake where wheels would slip
 - This causes the robot to predict movement that didn't happen
 - It appears to be possible to detect this and account for it in monte carlo localization
- Neutral Stake Scoring
 - This issue is likely caused by the intake scoring too fast after it finishes putting the lift up
 - Causes the rings to maintain vertical momentum while scoring, causing rings to get stuck on the tip of the stake

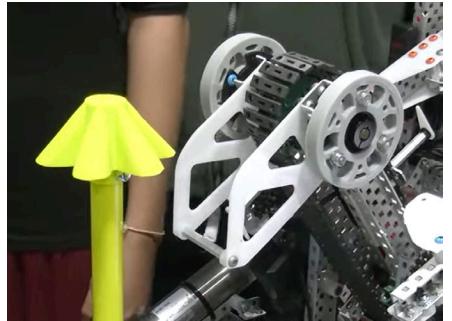
02/04/25 Build: Intake Optimizations (R.2.1.19)

Designed by: Carl	Witnessed by: Alex	Witnessed on: 02/04/25
-------------------	--------------------	------------------------

Goal: Create a structure to allow the robot to intake while in possession of a full goal.

Problem:

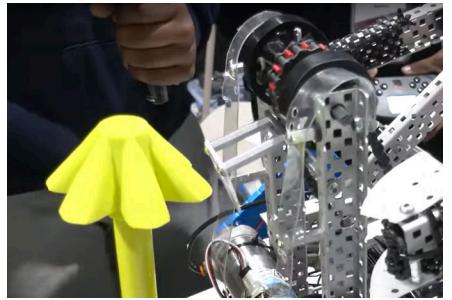
Currently, our intake hooks will snag on the top ring on a full mobile stake. This negatively impacts the robots ability to successfully release the goal. As the intake is on a lift, we can move the hooks away from the goal which allows the goal to drop normally. Throughout the course of programing/driving at competitions and during practice, we have had multiple instances where this extra motion caused inconsistencies in addition to the added cognitive load on the driver.



(Image from [39V Pits and Parts](#))

Possible Solutions:

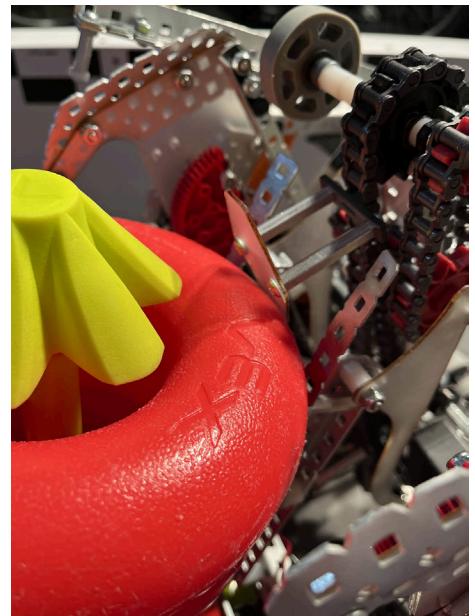
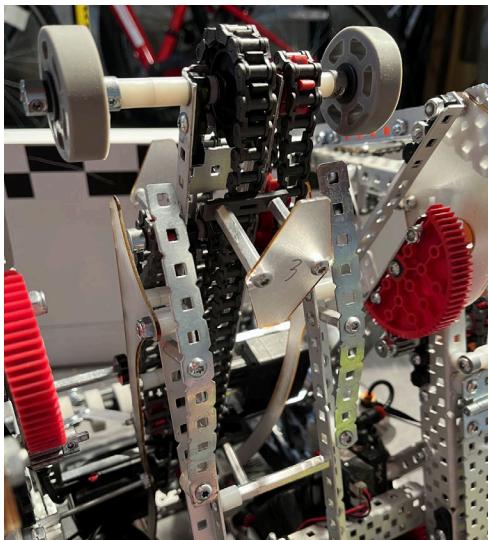
Many other teams, specifically those with a “Lady Brown” design, have already developed solutions to this problem. Here are some of the most effective designs we have found:



(Image from [1687C Pits and Parts](#))

Our Implementation:

Most other teams use polycarbonate mounted to the intake to push the rings away from their hooks. Both of these things weren't possible for us because of our unique intake mechanics and lack of available plastic, so our solution uses 1 hole wide plates mounted to the already existing 1x1 L-channels. We found that steel was the most effective material for the redirectors as it was stronger than aluminum and had much less friction with the rings. Adding these guides prevented the hooks from catching and did not have any noticeable impact on intaking performance.



02/04/25 Design: Skills Path End Repath (R.2.1.19)

Designed by: Carl

Witnessed by: Alex

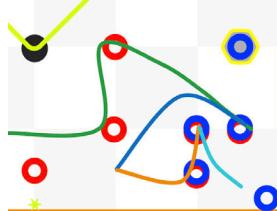
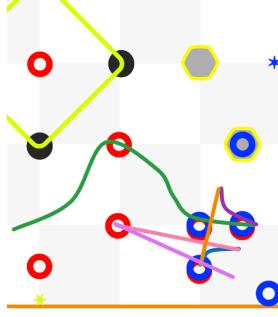
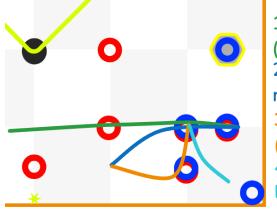
Witnessed on: 02/04/25

Goal: Repath the end of skills to increase consistency while maintaining the same point scoring capabilities.

Identify

Currently, the **end of the path for skills can rarely** result in a blue ring getting hit into the corner in a way that would **block a goal** from getting in, **miss a ring** on the goal, or both. We saw all of these situations at the [Kalahari signature event](#) (Pg. 345-346), and it reduced the score of several of our runs by a significant amount. These situations **reduce** the score of skills by **1-7 points** from the potential score cap. For this reason we want to consider **new endings** for the skills path that avoid these potential issues, especially blocking a goal from the corner.

Brainstorming

Path 1 (Current)	Path 2	Path 3
 <p>Pros:</p> <ul style="list-style-type: none"> • Already Programmed • Blue rings are very unlikely to get into the way <p>Cons:</p> <ul style="list-style-type: none"> • Multiple high curvature movements • Can occasionally miss the last red ring <p>1. Intake rings (Forward) 2. Reposition for next rings (Reverse) 3. Intake rings (Forward) 4. Reposition for last ring (Reverse)</p>	 <p>Pros:</p> <ul style="list-style-type: none"> • Already Programmed • More consistent blue ring • Less movements <p>Cons:</p> <ul style="list-style-type: none"> • Higher speed movements • Blue ring is in a risker spot <p>1. Intake rings (Forward) 2. Reposition for next rings (Reverse) 3. Intake red ring (Forward) 4. Reposition for last ring (Reverse) 5. Intake last ring (Forward) 6. Place goal (Reverse)</p>	 <p>Pros:</p> <ul style="list-style-type: none"> • Already Programmed • Very simple • Cuts lots of time so the robot has more time to intake <p>Cons:</p> <ul style="list-style-type: none"> • Misses 1 ring regardless <p>1. Intake rings (Forward) 2. Reposition for next rings (Reverse) 3. Intake rings (Forward) 4. Reposition for last ring (Reverse)</p>

Selection

We chose to select **path 2** because of the **reduction in high curvature motions and total movements** in the path, along with a more consistent blue ring position. We will test this route in practice and at the competition this weekend.

02/11/25 Analysis: Silver Creek Matches (02/08/25)

Designed by: Matt	Witnessed by: Alex	Witnessed on: 02/12/25
-------------------	--------------------	------------------------

Goal: Examine our matches from the Silver Creek Tournament and determine our strengths and weaknesses.

Matches:

In qualification matches, we went 4-2-0 and got a few win points ranking us 5th. We got chosen by 13358C as first seed and went on to win the tournament.

Match	Final Score	Win Point	Auton Win	Notes
Q6	36-7	Yes	Yes	Got 3 goals, goals not fully filled, we scored wall stakes while our alliance sat in the corner, got 1 wall stake, other was empty.
Q12	34-23	No	Yes	Got 3 goals, alliance sat in corner while we did wall stakes, got no rings on wall stakes, lost a goal due to not being pumped up.
Q15	18-39	Yes	Yes	Got 3 goals, alliance sat in corner while we did wall stakes, got both top rings on wall stakes.
Q28	34-3	No	Yes	Got 3 goals, alliance sat in corner while we did wall stakes, got both top rings on wall stakes.
Q32	32-32	Yes	Yes	Got 3 goals, alliance held corner while we did wall stakes, got 1 top rings on wall stake.
Q41	19-42	Yes	Tie	Got 3 goals, alliance held corner while we did wall stakes, got 1 top ring on wall stake, our alliance lost third goal to negative.
QF	40-18	N/A	Yes	Got 3 goals, switched around with alliance with wall stakes, got both top rings on wall stakes.
SF	40-20	N/A	Yes	Got 3 goals, opposing alliance got both positive corners, our alliance filled up negative wall stake, got both top rings on wall stakes.
F	37-17	N/A	Yes	Got 3 goals, opposing alliance got both positive corners, our alliance filled up negative wall stake, we baited opponents out and our alliance got the positive corner, got both top rings on wall stakes.

Matches Summary

Strengths:

- Mobile Goal Control - In every match we were able to get the third goal
- Autonomous - Our autonomous was able to hit in almost every match winning us auton every time except for one tie.
- AWP - We were able to do our part in AWP and got it in almost every match to bring our ranking up even with our losses.
- Strategy - We had good strategy when different scenarios happened such as when we lost both positive corners.

Weaknesses:

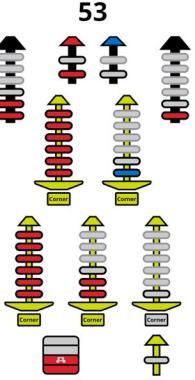
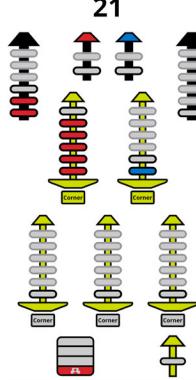
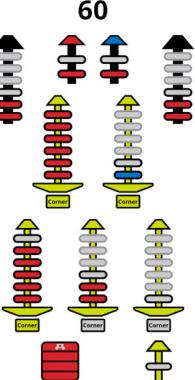
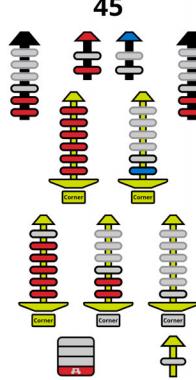
- Pre-match routine - In one of our matches we were not pumped up which made us lose the match.
- Positive corner control- In our elimination matches we lost both positive corners in our SF and F matches.

02/11/25 Analysis: Silver Creek Skills (02/08/25)

Designed by: Alex	Witnessed by: Matt	Witnessed on: 02/12/25
-------------------	--------------------	------------------------

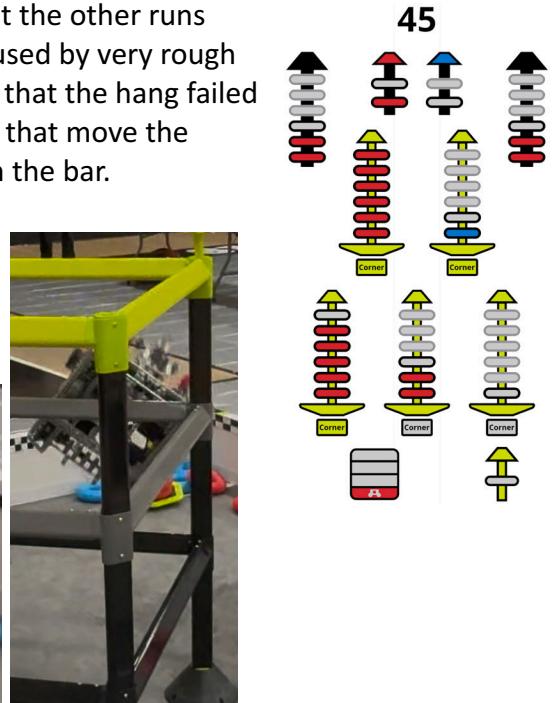
Goal: Examine our skills from the Silver Creek Tournament and determine our strengths and weaknesses.

Silver creek was our third time competing with this robot, and the first one with the [new path changes at the end](#) (Pg. 348).

Driver Skills 1	Autonomous Coding Skills 1
<p>In this run there was an override because of a missed wall stake at the end. We got a lower score in the end because of a panicked reaction to the missed wall stake.</p> 	 <p>This run was canceled early because of a misloaded intake ring, causing a jam. If we ran into the hang structure with our lift up the robot would have suffered severe damage.</p>  
Driver Skills 2	Autonomous Coding Skills 2
<p>This run missed a corner goal at the end of the skills run because of poorly placed blue rings.</p> 	 <p>This run had several issues, first the far alliance stake missed, then the corner missed due to the same corner rings and then our hang failed, which we assume is due to a setup issue.</p>   

Autonomous Coding Skills 3

This run had the same issues with alliance stake and corner miss that the other runs had, but also ended up falling from tier 3. We think that this was caused by very rough play against us in the last match with the lift up. In this run we think that the hang failed because the passive hooks on the robot hit the active hooks that lift that move the robot up causing a significant vibration that removed the robot from the bar.



Conclusion

- Low score of 105 points
 - Due to fall from tier 3 and [poor routing](#) (Pg. 348)
 - Fall was likely unavoidable code and build wise but led to significant issues at this competition
- We would like to revert to the code for [skills code we used at Kalahari](#) (Pg. 345)
 - However, after the fall from tier 3 we would like to spend the time working on the robot instead of working on quick repath

02/11/25 Analysis: Pikes Peak Matches (02/09-10/25)

Designed by: Matt	Witnessed by: Alex	Witnessed on: 02/12/25
-------------------	--------------------	------------------------

Goal: Examine our matches from the Colorado Signature Event and determine our strengths and weaknesses.

Matches:

In qualification matches we went 7-1-0 and got 7/8 auto win points ranking us 2nd. In alliance selection we first chose 652A who declined, we then chose 13358C who accepted and got knocked out in SF. We lost primarily because our alliance was unable to get the third goal in or after autonomous.

Match	Final Score	Win Point	Auton Win	Notes
Q8	33-30	Yes ▾	Yes ▾	Got 2 goals, alliance held corner while we did wall stakes, our alliance discontinued and got pushed out of the positive corner and our mobile goal got tipped, got one wall stake, lost other one because ring fell of intake.
Q20	42-22	No ▾	Yes ▾	Got 3 goals, alliance held corner while we did wall stakes, got both wall stakes.
Q36	29-39	Yes ▾	Yes ▾	Got 2 goals, alliance held corner while we did wall stakes, got both wall stakes.
Q54	29-25	Yes ▾	Yes ▾	Got 3 goals, switched around with alliance doing wall stakes, while our alliance camped positive they got pulled out, we helped come back and protect it, we lost a goal but our alliance protected it from the negative corner, got both wall stakes.
Q73	36-22	Yes ▾	Yes ▾	Got 3 goals, alliance held corner while we played defense and played wall stakes, our intake jammed while trying to do wall stakes so we got none, but played defense so they didn't get any wall stakes.
Q96	25-42	Yes ▾	Yes ▾	Got 3 goals, alliance held corner while we did wall stakes, the dropped goal but we were able to retrieve it, got 1 wall stake.
Q104	36-25	Yes ▾	Yes ▾	Got 3 goals, alliance held wall stakes while we did wall stakes, didn't get any top rings on wall stakes.
Q118	33-40	Yes ▾	Yes ▾	Got 2 goals, switched around with our alliance with wall stakes, got 1 wall stake top ring and alliance hung.
R16	42-29	N/A ▾	Yes ▾	Got 3 goals, switched around with our alliance with wall stakes, got 1 wall stake top ring.
QF	41-33	N/A ▾	Yes ▾	Got 3 goals, switched around with our alliance with wall stakes, got both wall stakes top rings.
SF	38-40	N/A ▾	Yes ▾	Got 2 goals, switched around with our alliance with wall stakes, got 1 wall stake top ring.

Matches Summary

Strengths:

- Mobile Goal Control - In almost every match we were able to get a third goal and we were able to effectively control the goals we had. Additionally we were able to steal some goals in matches.
- Autonomous - Our autonomous was very consistent and we won autonomous in every single match.
- AWP - We were able to get AWP in all but one match ranking us highly.

Weaknesses:

- High hang - Our hang broke the day before this competition so we weren't able to **safely** hang in matches.
- Recovering from a deficit - As noted earlier, a tier 3 hang significantly improves our recovering ability, however, we could not hang in semifinals as our alliance was occupied with a wall stake and could not protect our goal. We could not hang with the goal or drop it without risking a negative play.

02/11/25

Analysis: Pikes Peak Skills (02/09-10/25)

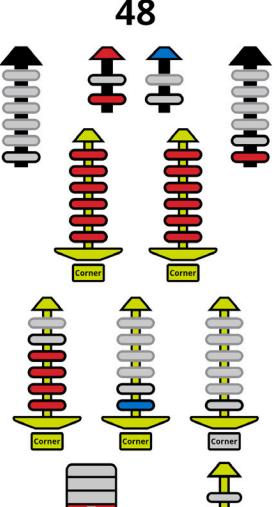
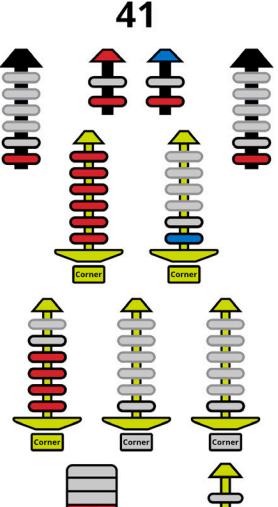
Designed by: Alex

Witnessed by: Matt

Witnessed on: 02/12/25

Goal: Examine our skills from the Colorado Signature Event and determine our strengths and weaknesses.

The Colorado Signature event was our 4th time doing skills on this robot, and the second one with the [new path changes at the end](#) (Pg. 348).

Driver Skills 1	Autonomous Coding Skills 1
<p>Issues:</p> <ul style="list-style-type: none"> Radio disconnected after about 10 seconds for about 10 seconds Led to a low score/need to override Brain log shows the disconnect at 16:11:30 For future runs we used bluetooth because of the radio interference on the VEXnet RF bands at the event  	<p>Issues:</p> <ul style="list-style-type: none"> Intake jammed at the end of the path Missed corner as well in a similar way Hang took longer than expected >9s and it fell from tier 2.  

Note: after our autonomous coding skills run we decided to not continue with skills at this tournament. We made this decision because of the perceived risk with climbing and the uncontrollability during programming skills that we ran into at the last competition.

Conclusion

At this competition we had a very low score due to **inconsistencies** in the **robot and the hang**. We think that prioritizing a **robust robot** that cannot fall is very important because falling destroyed our robot's performance in skills over multiple competitions and caused many issues with other subsystems.

ROBOT 3

02/13/24 Time Management: Robot Rebuild

Designed by: Alex

Witnessed by: Carl

Witnessed on: 02/13/24

Goal: Develop a timeline to rebuild the robot before our state competition on March 14th.

Justification

After the Colorado signature event [matches](#) (Pg. 352) and [skills](#) (Pg. 353) we found many major issues with the current robot that we would like to fix on our next bot. Additionally, because of the falls we had our robot would need major repairs for states, so we think it would be best to just go ahead and do the rebuild before states. Below is the timeline we have planned to complete the rebuild before states:

Task	Week 1				Week 2				Week 3				Week 4																	
	2/11	2/12	2/13	2/14	2/15	2/16	2/17	2/18	2/19	2/20	2/21	2/22	2/23	2/24	2/25	2/26	2/27	2/28	3/1	3/2	3/3	3/4	3/5	3/6	3/7	3/8	3/9	3/10	3/11	3/12
Strategy Reevaluation																														
Brainstorm																														
Design																														
Build																														
Test/ Improve																														
Program																														
Drive																														
Competition																														

Individual Rationale:

- Strategy Reevaluation
 - Fully reevaluate our approach to strategy and its relation to robot build
 - Ensure that our strategy for the rebuild keeps up with current strategy
- Brainstorm
 - Develop multiple solutions to meet strategy requirements
- Design
 - CAD selected design
- Build
 - Build the selected design from the CAD
- Test/Improve
 - Spend time to thoroughly develop the robot to test for robustness
- Program
 - Develop programming on the robot, tune skills and autonomous
- Drive
 - Spend time on driver practice to ensure we have skilled driving on this robot
- Competition
 - Attend the competition

02/13/25 Evaluate: Robot 2 Subsystem Analysis (R.2.1.19)

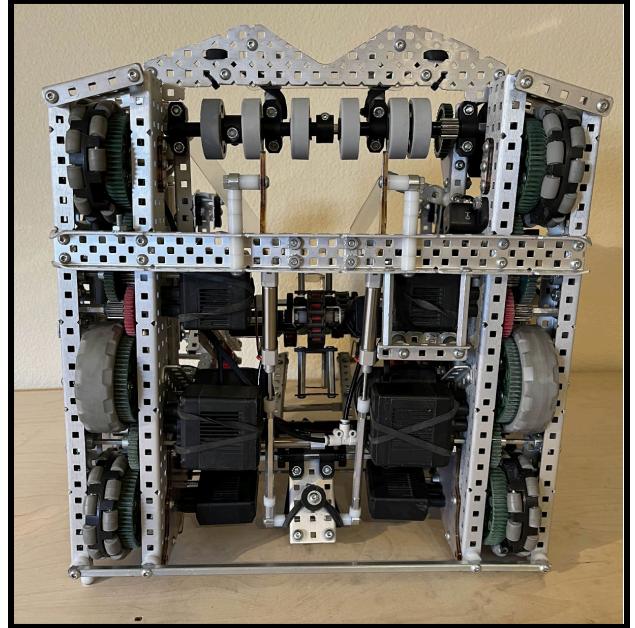
Designed by: Carl	Witnessed by: Alex	Witnessed on: 02/15/25
-------------------	--------------------	------------------------

Goal: Perform an analysis of each subsystem of Robot 2 for reference in Robot 3's design.

Drivetrain:

Our second robot utilized a 55W drive base with 360 RPM 3.25" wheels. Our robot (15 lbs) was very comparable to other teams with 66W drive bases in terms of acceleration and pushing power, however, we were lacking in top speed. This was not a huge issue in skills, but a faster robot would allow us to react to negative corner plays faster and play wall stakes more competitively.

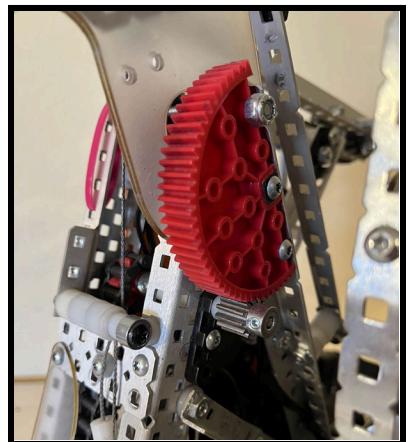
The drive friction was extremely low as a result of the screw joints and limited gearing which allowed us to drive for an extended amount of time (4-5) without burnout. The 1/16 dropped traction was effective for our IME odometry and allowed us to defend the corner and wall stakes extremely effectively.



The footprint of the robot was 28 holes wide and 27 holes long which was relatively bulky compared to other robots but that didn't significantly hinder our performance. Making the drive narrower would reduce robot structure (saving weight) and allow us to intake out of the corner easier. This would also improve turning speed and overall defence evasion.

Lift:

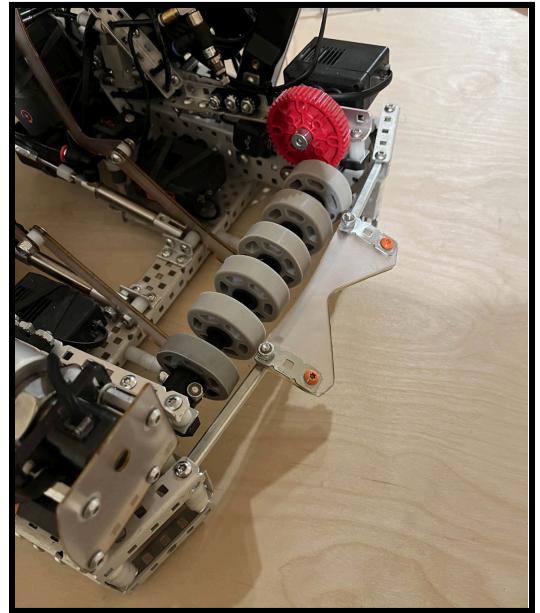
The lift was 33RPM and driven by two 5.5W motors with a 1:6 gear ratio. The lift is operating at close to the minimum amount of torque required to drive the bottom hang arms, although the linkage with the lift was likely not optimal. The lightweight construction of the lift arms resulted in them easily bending (primarily when the robot fell but also through normal usage) which meant the robot needed to be regularly tuned to score on wall stakes.



Intake:

Our bottom intake (5.5W, 800 RPM, 1.625" wheels) and top intake (16.5W, 400 RPM, 12T sprockets) were very successful and had no major issues (except those stemming from T3 falls). The intake had almost 100% consistency and when coupled with color sorting was dominant in skills and matches.

We could not have a crossbar on the bottom intake (for corner clear) so there was a fair amount of friction from misaligned bearings at some points. The top intake utilized a lengthy chain route which was ideal for motor placement but likely added friction. The bottom intake was a good speed, however, the top stage could likely be around 10% faster without a noticeable impact on consistency (assuming adjusted gearing).

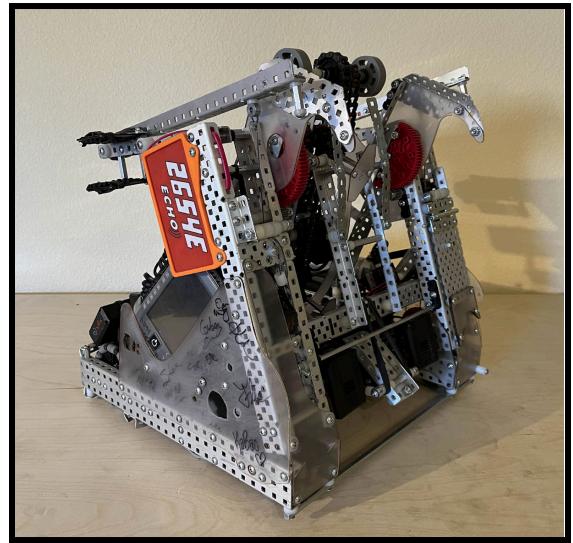


Hang:

The hang proved to be successful and we were able to repeatedly achieve a tier 3 elevation in less than 9 seconds. Having the hang allowed us to achieve world record skills runs and win several matches that we would've lost otherwise, however, there were a few critical issues:

- **Falling**

- The robot had several ways fall, most of which were due to the extremely precise motion and close tolerances when hanging including:
 - Less than a 0.25" gap between the passive bar hooks and top hang arms meaning that the hang arms could occasionally collide with the hooks
 - 0.1"- 0.2" precision was required from the winch to correctly set the passive hooks on a rung
 - Small hooks on the top arms were quite small and needed very precise lift position in order to correctly grab the next rung
- In practice, we always had a person to catch the robot in the event of failure but that was not a possibility in skills or matches so falls were very detrimental
- Failures from hanging increased the odds of falling again significantly and negatively affected other subsystems (primarily the intake and lift)



- **Design Limitations**

- Top and bottom hang arms could not be braced against each other due to our lifted intake
 - This lack of bracing made the hang quite flimsy
 - Building strong pivots for the hang joints resulted in extra weight
- The lift and hang move together
 - Although having the hang arms integrated into the motion of the lift was great for fine adjustments and dynamic hang control, there were problems with it
 - The hang arms would be exposed a lot more when the lift was up—when scoring wall stakes—so they sustained damage from defensive robots
 - The lift speed was determined by the slower gear ratio needed to produce enough torque when hanging
 - The hang exerted abnormal forces on the lift which likely caused some of the inconsistencies when scoring wall stakes
- We believe the geometry of the hang (lengths of the arms and positioning of the pivot points) could be improved in multiple aspects but these changes would not be possible to implement without a complete redesign of the robot
- It was very difficult to align the robot for a hang even with the aligner mechanism

Back Clamp:

The back clamp was very effective and reliable with the ability to grab goals from a variety of positions and hold them very securely. The two 25mm pistons were more than enough power and did not use an excessive amount of air. The two minor issues with this subsystem were that the goal could tilt into the intake hooks easily (jamming the intake) and it could not grab or align goals that were pointed directly into the robot. These issues were generally avoidable but solving them would be beneficial.

Because we pivoted the clamp on the PTO shaft and mounted the pistons to the front crossbar, we didn't need to add any additional structure making the system very light.



Overall Notes:

Pros:

- Very efficient wall stakes
- Proven concept for a viable T3 hang
- Robust construction
 - All electrical components, chains, and other fragile parts were very well protected from other robots with big polycarbonate panels
- Good sensor integration
- No burnout issues

Cons:

- Complex wall stake mechanism
 - Lots of moving parts and pivots
 - Geometry was sensitive to minor changes
 - Could jam from heavy defence or driver error
- T3 hang was not compliant/reliable enough
 - Falling off had catastrophic consequences
- No goal rush mechanism
- Could not hang with a goal
 - Limits use cases in matches
- Bad hang lineup



Robot Media on YouTube:

- High Hang: https://www.youtube.com/shorts/kFnde_ClqSw
- Robot Reveal: https://www.youtube.com/watch?v=X0uw_-r3_tI
- World Record Skills: <https://www.youtube.com/watch?v=PDdXTecw3OI>
- Robot Explanation: https://www.youtube.com/watch?v=Wj1L_OZ68vs&t=794s

02/13/25 Brainstorm/Identify: Robot 3 Requirements

Designed by: Carl	Witnessed by: Alex	Witnessed on: 2/14/25
-------------------	--------------------	-----------------------

Goal: Identify necessary robot functions and create some basic requirements to consider.

Desired Functions & Traits:

Note: Desired functions and traits based primarily on: [Sugar Rush Analysis](#) (Pg. 335), [Kalahari Analysis](#) (Pg. 348), [Silver Creek Analysis](#) (Pg. 353), [Pikes Peak Analysis](#) (Pg. 356), and [Robot 2 Subsystem Analysis](#) (Pg. 360).

1. Sub 10 second T3 hang
 - a. Maximizes time for other scoring objectives
2. Hang with a goal
 - a. Allows for much lower risk elevations
3. **ALL SUBSYSTEMS MUST BE VERY RELIABLE**
 - a. Key to world record skills scores, auto and match wins, and avoiding damage to the robot
 - b. The simpler the robot, the better
4. Fast & competitive wall stake mechanism
 - a. Our last robot lacked speed and reach to score while being defended. Better scoring would increase our winning margin
5. Goal rush mechanism
 - a. Getting control of the third goal during auton is beneficial for high scoring autonomous routines and securing the match win
6. Fast hang lineup
 - a. A necessity for hang to be viable in matches
7. Competitive drivetrain
 - a. Allows for overall high performance and allows us to react quicker to unexpected scenarios
 - b. Must be at least 6 motors
 - i. More power for hang
 - c. Dropped traction required for programming
8. High stake scoring
 - a. Effectively 2 points in skills (not very beneficial)
 - b. 8-10 extra points matches (generally match affecting)

Summary:

These 8 criteria provide rough requirements for our next robot, however, they do not describe any specific robot mechanisms. To determine what mechanisms we will use we will research different options for each subsystem and examine the limitations of integrating those subsystems onto the same robot.

02/14/25 Background Research: Subsystem Options

Designed by: Carl

Witnessed by: Alex

Witnessed on: 2/15/25

Goal: Identify and compare different options for each subsystem.

Goal Clamp:

As our old back clamp was very effective and the vast majority of teams utilize a similar design, doing an entire analysis of this subsystem would be redundant (especially because the specific design for the back clamp is driven by the robot's structure and layout).

Drivetrain:

As mentioned on the previous page, we will be using 6 motors on the drivetrain. This will simplify drivetrain gearing (all motors can use 600 RPM cartridges), allow for a faster OR more powerful hang (this will help if we are hanging with a goal), and accommodate faster gear ratios. Here are the gear ratios that we considered for this robot:

Drivetrain Stats:	Pros:	Cons:
<ul style="list-style-type: none"> • 36:60 360 RPM • 3.25" wheels • 61 in/sec • Used on Robot 2 (Pg. 242) 	<ul style="list-style-type: none"> • Great pushing power • 16 LB max weight • Easy to build • Compact gearing • Good crossbar options • Good drivetrain length 	<ul style="list-style-type: none"> • On the slower end • Wheels are fairly big
<ul style="list-style-type: none"> • 36:48 450 RPM • 2.75" wheels • 65 in/sec • Used on Robot 1 (Pg. 84) 	<ul style="list-style-type: none"> • Good pushing power • 15 LB max weight • Great motor locations • Most compact gearing • Small wheels • Good drivetrain length 	<ul style="list-style-type: none"> • Still slightly slow • Less space for crossbars
<ul style="list-style-type: none"> • 48:72 400 RPM • 3.25" wheels • 68 in/sec 	<ul style="list-style-type: none"> • 14 LB max weight • Ideal balance of speed and power • Good drive length 	<ul style="list-style-type: none"> • Very bulky gearing • Low space for crossbars • Bad motor mounting options

Drivetrain Stats:	Pros:	Cons:
<ul style="list-style-type: none"> • 48:60 480 RPM • 2.75" wheels • 69 in/sec 	<ul style="list-style-type: none"> • 14 LB max weight • Ideal balance of speed and power • Very good crossbar positions • Small wheels 	<ul style="list-style-type: none"> • Bad motor mounting options • Very short chassis length; other teams that have used this drive have had tipping problems
<ul style="list-style-type: none"> • 36:48 450 RPM • 3.25" wheels • 77 in/sec • Used last year (Pg. 52) 	<ul style="list-style-type: none"> • High top speed • Good crossbar positions and gearing • Good motor mounting 	<ul style="list-style-type: none"> • 12 LB max weight • Will likely overheat with added T3 hang and structure

Note: We only considered drivetrains utilizing 600 RPM motors for the reasons [discussed here](#) (Pg. 102)

Note: Max weights based on robots with similar characteristics from other seasons.

Wall Stake Mechanism:

Mechanism Name & Image:	Pros:	Cons:
Standard Lady Brown 	<ul style="list-style-type: none"> • Fast scoring • Good under defence • Ability to descore • Easy to line up • Extremely simple • Easy to build • Low weight 	<ul style="list-style-type: none"> • Can't do alliance stake with mobile goal • Can't flip goals without alternate mechanism
360 Lady Brown 	<ul style="list-style-type: none"> • Ability to do Alliance Stake • Flipping and un-flipping mobile goals • Ability to descore • Fast scoring • Good under defence • Easy to line-up • Extremely simple • Low weight 	<ul style="list-style-type: none"> • Less reach • Worse scoring <ul style="list-style-type: none"> ○ Still better than non-Lady Brown mechanisms • Requires good pre-planning for arm geometry

Mechanism Name & Image:	Pros:	Cons:
Echo Mech 	<ul style="list-style-type: none"> • 2-ring wall stakes • No horizontal expansion • Fast scoring • Lift motion good for hang integration 	<ul style="list-style-type: none"> • Hard to score under defense • Hard to line-up • Prone to jamming • Complex • Requires precise tuning and set points • No color sorting option for wall stakes • Lots of motors
Fish Mech 	<ul style="list-style-type: none"> • Fast scoring • Lightweight • Some adaptations can get alliance stake 	<ul style="list-style-type: none"> • Very space consuming • No descore • Hard to score under defense • Hard to line-up • Needs at least 11W

Top and Bottom Intake:

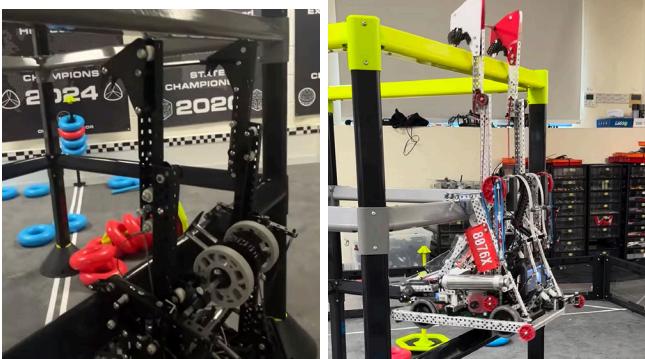
Similarly to the back clamp, we believe that our current implementation of a hook intake is close to ideal (with the exception of the issues listed [here](#) (Pg. 358)) so it is not necessary to conduct further background research and analysis especially considering the similarities between our design and other competitive intakes.

Despite this, there are still improvements that we can make such as lighter structuring and improved integration. Additionally, by eliminating some of the excessive intake chaining (and associated friction) a faster top stage would likely be possible even if we move down to 11W.

The bottom stage should be able to remain very similar to the old intake, however, we might be able to make several improvements to the structuring depending on horizontal expansion constraints based on the hang and wall stake mechanism.

Tier 3 Elevation:

Since our original attempt at a T3 climb, several other teams have developed and implemented high hangs. With the exception of a few alternative designs (none of which have worked) we have only seen two types of successful hangs (both of which we independently identified [earlier](#) (Pg. 244):

Linear Slide Based Hang	Two Arm Hang
<p>Image Credit: 5203G Gremlin + 8076X Recoil</p>  <p>Pros:</p> <ul style="list-style-type: none"> • Can hang anywhere; instant lineup • Forces are symmetrical and predictable • Simple control and only one power input <p>Cons:</p> <ul style="list-style-type: none"> • Swinging can cause falls • Complex mechanism with lots of custom and moving parts • Complicated string routing • Poor integration with most other subsystems • We have no experience with this design 	<p>Image Credit: 3004A Qwerty + 13176A Smilebots</p>  <p>Pros:</p> <ul style="list-style-type: none"> • Relatively simple • We have significant experience with this design • Lots of space for other mechanisms <p>Cons:</p> <ul style="list-style-type: none"> • High twisting forces • Geometry needs lots of refinements • Complex hang motions

We also considered developing another type of hang to hopefully eliminate some major issues of the other designs, but decided against it because of our limited time before states and the immense uncertainty of that solution.

02/15/25 Design: Ideal Robot vs. Physical Limitations

Designed by: Carl	Witnessed by: Alex	Witnessed on: 2/16/25
-------------------	--------------------	-----------------------

Goal: Determine the most optimal option for each subsystem and identify sacrifices needed to accommodate integration of these subsystems.

Realistically Possible Solutions:

Wall Stakes: 360 Lady Brown

High Hang: Two Arm Hang (front mounted)

Drivetrain: 36:48, 450 RPM, 2.75" wheels

Power Distribution:	
Drivetrain	66W
Wall Stakes	5.5W
Bottom Intake	5.5W
Top Intake	11W

Rationale:

A 360 Lady Brown was by far the most optimal wall stake mechanism offering extremely fast and reliable scoring, good alignment, and defence resistance. This design is also incredibly simple allowing it to be powered by a single 5.5W motor and meet all of the [relevant requirements](#) previously described (Pg. 362).

In a perfect world, we would want to use a slide based hang because of the incredibly easy alignment it offers, however, a 360 Lady Brown cannot be integrated into this design for a multitude of unresolvable structural constraints. The two are simply mutually exclusive.

In order to maintain enough time to hang in skills, we need at least 16.5W on the intake so we can score all of the red rings quickly. With a 66W drivetrain that leaves 5.5W for the wall stake mechanism and based on our research, there aren't any other competitive wall stake designs that use only 5.5W. As wall stakes are an integral part of our skills and match strategy, abandoning them for a faster hang lineup is not justifiable. Furthermore, a slide hang would likely not be able to hang without and with a goal as the balance would change significantly.

The two arm hang on our last robot was mounted on the back, however, **mounting it on the front** would offer several key advantages:

- Hanging with a goal is possible
- Hang geometry will be less restricted by the intake
- The hang can share the bulk of its structure with a Lady Brown if implemented correctly

For the drivetrain we needed a compact solution (meaning 2.75" wheels) with good gearing in order to nicely fit the hang. The only two drives that fit these criteria are 450 RPM (65in/sec) and 480 RPM (69in/sec). We ultimately decided to go with 450 RPM because its speed was very comparable to 480 RPM but it had better drivetrain length, gearing, and motor mounting options.

02/16/25

Design: CAD Day 1 (C.4.1)

Designed by: Carl

Witnessed by: Alex

Witnessed on: 2/16/25

Goal: Design the drivetrain and PTO for our new robot.

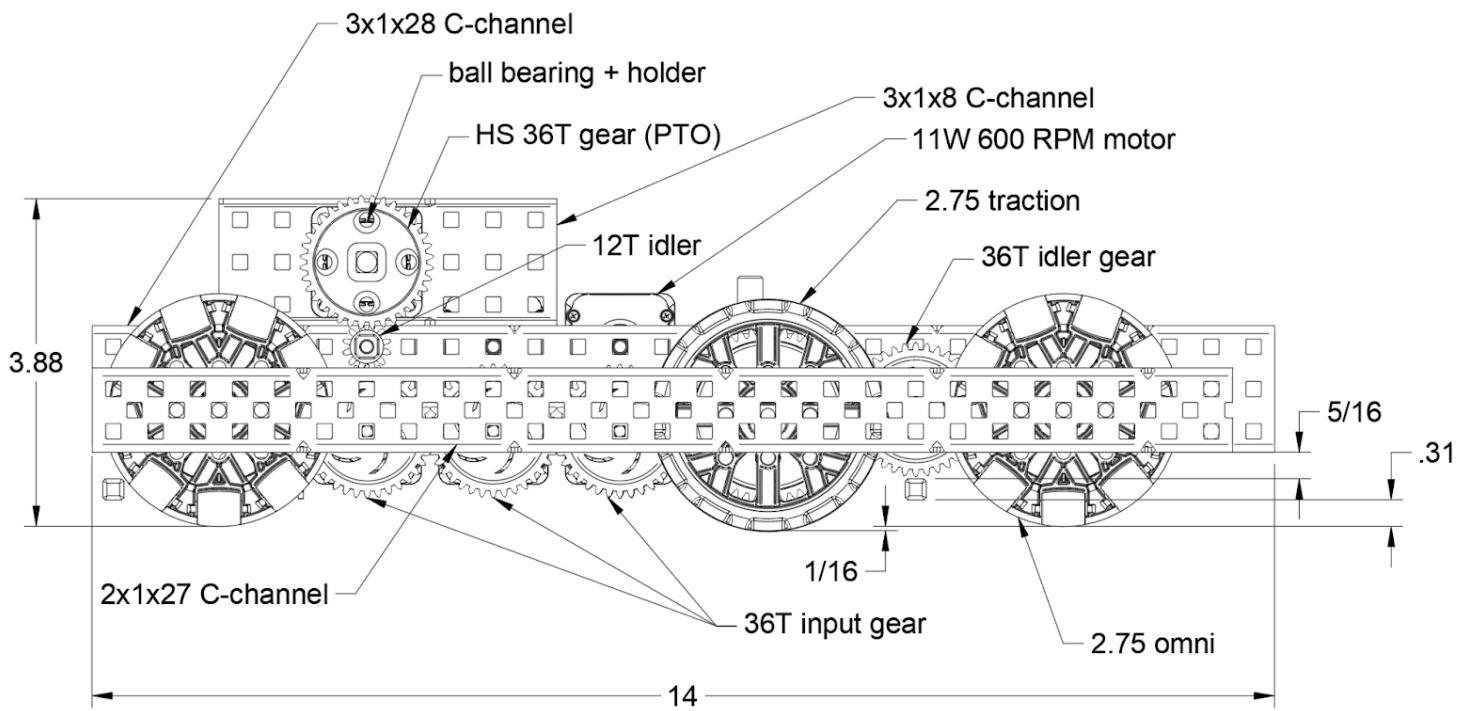
Drivetrain Design Constraints:

- Must be 450 RPM on 2.75" (Pg. 365)
- Must use a dropped traction wheel (Pg. 360)
- Must use six 11W motors (Pg. 360)
- Strong construction (2+ full width crossbars)



Image: complex drive gearing from Robot 1

Our [first robot's drivetrain](#) (Pg. 140) utilized a dropped traction wheel in the center while our [second robot's drivetrain](#) (Pg. 260) had one offset backwards. There was a noticeable difference between the handling and maneuvering of the two drivetrains as a result of the changed center of rotation. The wheel in the back made for higher control of the back of the robot but lowered the control of the front whereas the centered wheel was much more even. Due to the sizes of the gears we are using, a centered wheel would require more complicated gearing (similar to Robot 1) which makes it very difficult to achieve a compact design. This means that the traction wheel needs to be offset to the front or back, and as intaking and wall stake scoring would both benefit from a more controllable front, we went with a forward offset. Here is our layout plan for the drivetrain:



Drivetrain Layout Explanation:

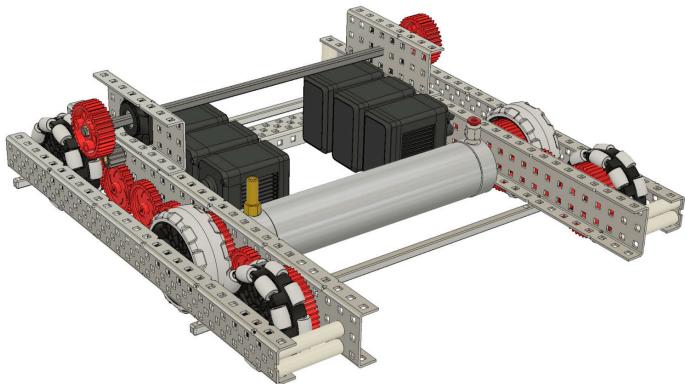
In order to fit 3 motors on each side, we decided to mount them vertically as they fit very well and provided enough space for a pneumatic reservoir in front (very good for weight distribution). We used two full width HS shafts as the crossbars to ensure a rigid construction while also taking up unused space under the drivetrain. The back shaft will provide a lever point for the goal and the front shaft can be used to mount the intake ramps. The inside drive rail is a 3 wide C-channel to accommodate mounting the vertical motors.

Our last robot had a very fast hang with 55W on the drive, however, the motors did struggle at some points in the hanging process. We believe that keeping the same ratio will be beneficial as it is a proven speed and with the 20% power increase the new robot should have more than enough torque to hang with a full mobile stake. Furthermore, we will be using ball bearings on the winch shaft to reduce friction from the extremely high perpendicular load on the shaft.

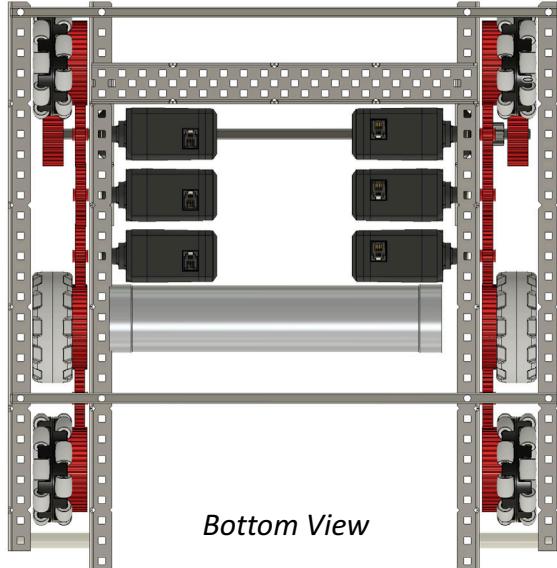
The length of the drive is 14 inches (28 holes), on par with our other robots. There is only one 36T idler in the main drive and one 12T idler to connect the PTO meaning the drive should be extremely low friction and relatively light. Additionally, there are no polycarbonate parts necessary for the drivetrain (with the exception of the wheel offset poly which we are re-using from the last robot).

For the back clamp, we have 0.5" OD spacers boxed through the drive rails to smoothly funnel the goal into position and a 2x1x19 C-channel going across the drive. This C-channel should help prevent the goal from tipping into the intake and also provides a place to mount other structures such as uprights.

Additional Chassis Images:



Isometric View



Bottom View

02/17/25 Design: CAD Day 2 (C.4.1)

Designed by: Carl	Witnessed by: Alex	Witnessed on: 2/18/25
-------------------	--------------------	-----------------------

Goal: Create structure for the wall stake mechanism and roughly design the hang.

Lady Brown Design Constraints:

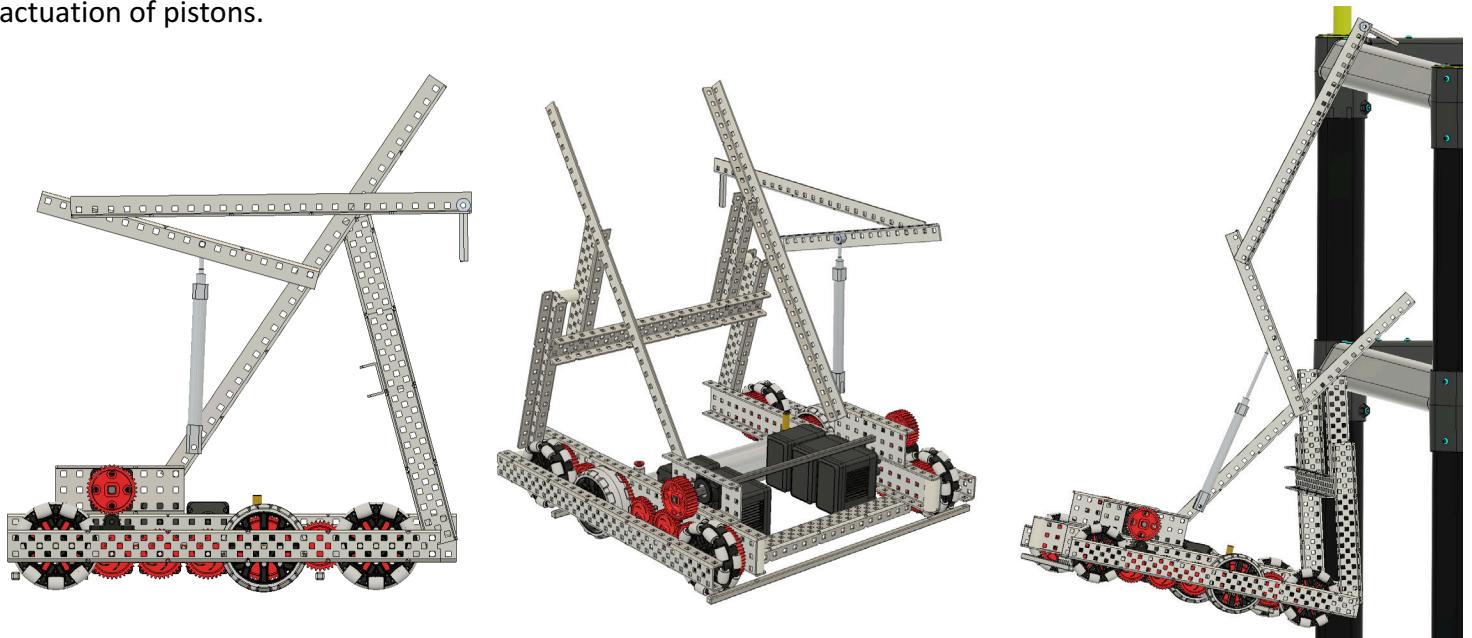
After further research and discussion with teams using a Lady Brown, we identified a few key aspects in building a successful one:

- The arm needs to reach as high as possible for better scoring
 - The arm should be as long as possible while also abiding by the horizontal expansion limits described in <SG2>
 - The pivot should be as high as possible but should also be low enough to fit under the bottom hang rung (16.1")
- The ring needs to be held in the arm securely
 - Hard to test in CAD, ensure accessible mounting holes for ring holder
- Correct gear ratio
 - Most competitive teams use a ratio of 1:3 with a 200 RPM 11W motor, however, we have seen teams successfully use 5.5W with the same gearing
 - We will also use a 1:3 gear ratio with a 5.5W motor to maintain competitive scoring speeds. In the event that this reduction does not provide enough torque we will look into other solutions such as band assistance
- Rigidity
 - The arms need to be force resistant from all directions to hold the ring securely and to prevent damage (primarily when doing wall stakes or tipping goals)
 - We will use a HS shaft between the gears to reduce twisting between the arms and add a brace to a higher point to join the two sides together
- Hang integration
 - The triangle bracing should be integrated into the hang mechanism to save weight and simplify the design

Based on our past experiences, the geometry required for hang is very specific so it needs to be designed before/with the wall stake mechanism. Our wall stake mechanism is relatively simple so we will create the hang structure first while keeping the LB in mind.

Hang Design Constraints:

As mentioned [earlier](#) (Pg. 368), we will continue to have a two arm hang meaning a majority of the [geometry concepts from the last robot](#) (Pg. 275) still apply. To keep the robot simple and wallstake scoring fast, we chose not to link the hang to the wall stake mechanism. The hang arms need to be powered both ways so linking them to the intake with a ratchet would not be possible leaving us only one option: pneumatics (similar to [3004A Qwerty](#)). This will not only simplify the robot significantly but might also decrease hang time due to the fast actuation of pistons.



Note: To save time when modifying the hang geometry, only one side is CADed for now.

The uprights are made up of C-channels which are staggered to allow for different spacing at the top and bottom. They are mounted at the front of the drive rails to help direct the robot around the bar in a similar fashion to our old hang. The uprights need to be wide at the bottom in order to fit the intake but narrower at the top so the passive bar hooks are further away from the hang arms (this should eliminate the [problem of the hooks and arms colliding](#) (Pg. 357)).

The triangle bracing is mounted flush with the 3 wide C-channel on the drive to ensure a strong connection at the bottom and will be connected at the top with a polycarbonate bracket (not designed yet). The piston and bottom hang arm are also both mounted to the triangle bracing to eliminate extra structure. The triangle bracing is also very close to the hang bar (so bar hooks will be easy to add) and in a good position to mount the LB.

The top and bottom hang arms are made from half-cut 3 wide C-channels as they are similar in size to 1x1 L-channels but significantly stronger due to the larger flange. This is necessary to deal with high forces from the hang without adding too much weight.

Lady Brown Addition:

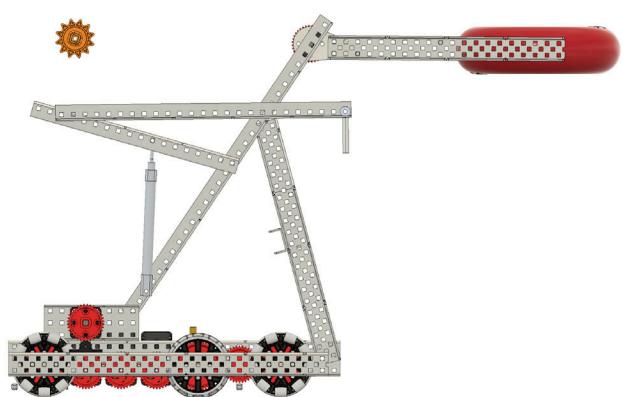
Implementation of this mechanism was very straightforward by just following the constraints on page 368. After finding a good balance of pivot height and arm length the arm appears to be able to hit desirable positions for scoring and loading as shown below.



Loading LB

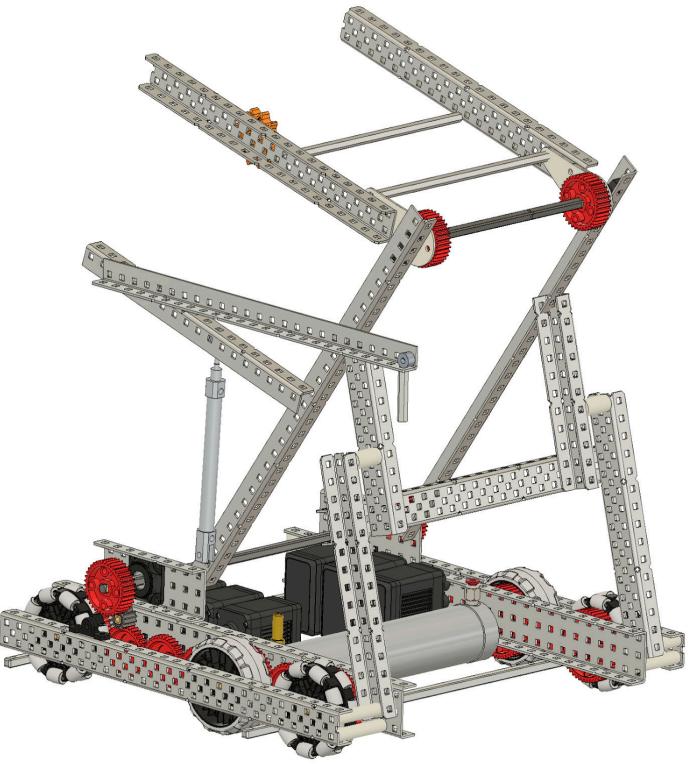
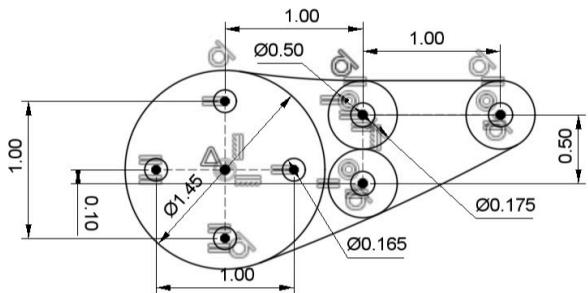
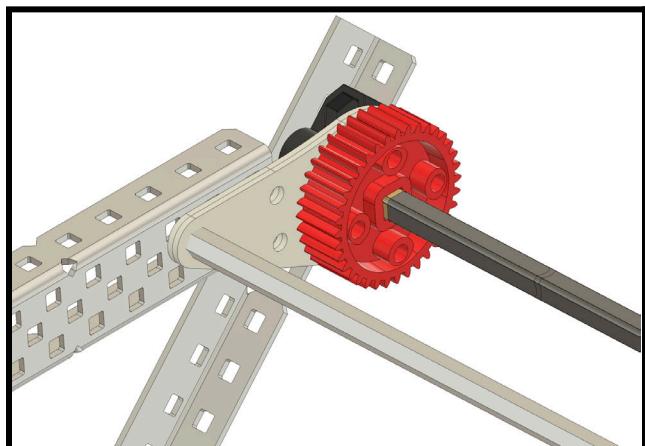


Scoring on Wall Stake



Scoring on Alliance Stake/max extension (23.5")

A high strength shaft connects to the 36T gear at the base of each arm and will hopefully prevent twisting between the two sides. Polycarbonate adaptors connect those gears to two 2x1x20 C-channels allowing the arms to be positioned above the main triangle bracing. This is necessary to achieve the correct gap for a ring between the arms. Standoffs are used to connect the two arms together and prevent them from bending outwards when holding a ring. Pillow blocks are used to mount the HS shaft to the triangle bracing making for a light and strong connection.



02/17/25 Design: Robot 3 Subsystems

Designed by: Alex

Witnessed by: Carl

Witnessed on: 2/18/25

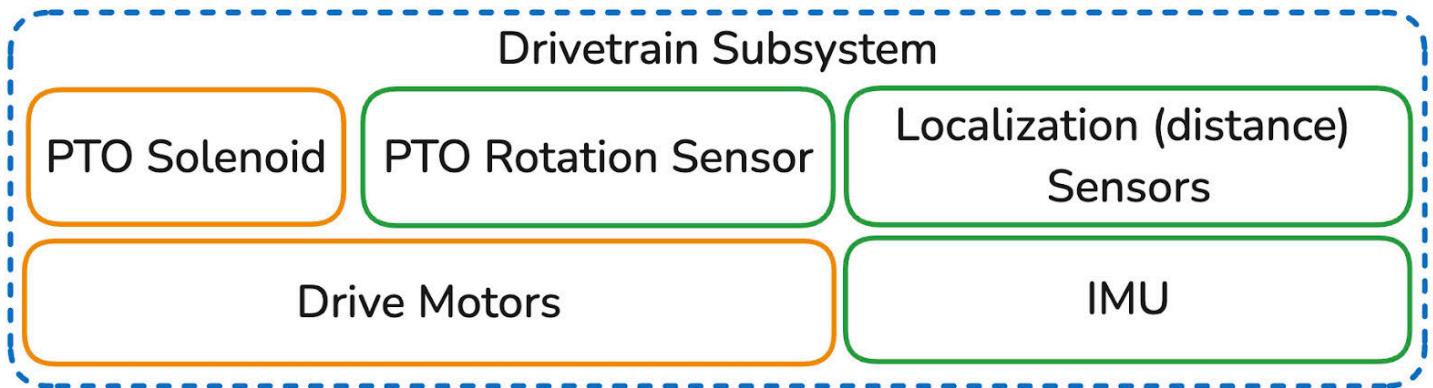
Goal: Create a structure of subsystems and commands for the new robot with the lady brown structure.

Subsystem Layout

With our [command based code](#) (Pg. 157) we have to meticulously lay out how our subsystems are organized with the hardware to ensure that later we have a straightforward process to program using the command based architecture. Because of the major changes we made on this robot we will have to go through the process of reevaluating which hardware components are assigned to different subsystems in code. For each of the subsystems we will put the sensors (**green box**) and actuators (motors and solenoids, **orange box**) together in the graphic.

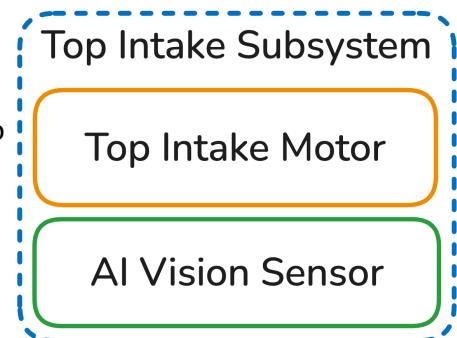
Drivetrain:

Very similar to the last robot, we will make a drivetrain object that consists of the drive motors, PTO solenoid, and PTO rotation sensor. We believe that this architecture for the drivetrain makes the most sense because all of these individual components either act directly on or get power directly from the drivetrain. Additionally, we will attach the



Top Intake:

While the top and bottom intakes largely act like one subsystem, because of the independent control that we have on this robot it makes significantly more sense to control each component independently as this allows us to do more complex indexing for the lady brown, or advanced control such as ejection code. For ejection to work properly, we utilized an AI vision sensor in this subsystem to allow the code to detect which color rings are currently in the intake.



Bottom Intake Subsystem:

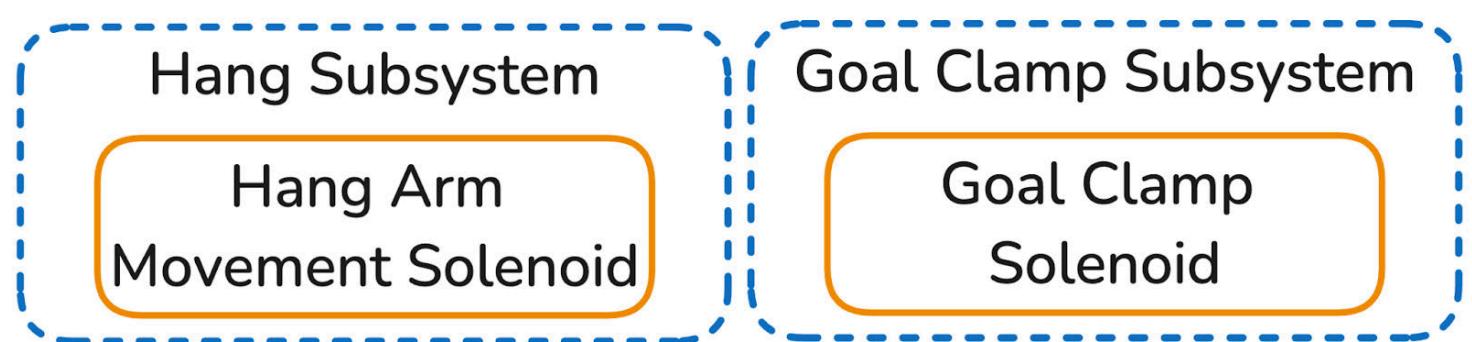
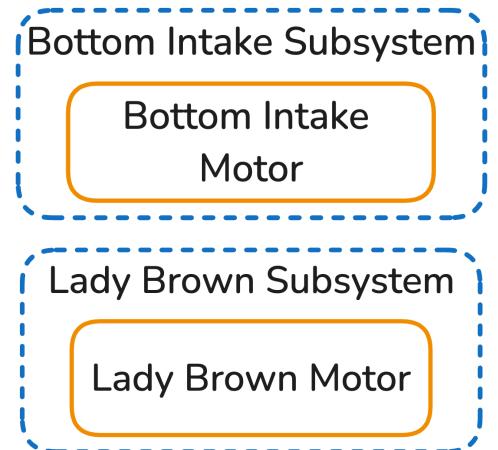
This code is very simple, just percentage motor control, but by compartmentalizing this part it allows for some more complex controls that we described in the Top Intake subsystem section.

Lady Brown Subsystem:

This subsystem is much the same as the previous ones, it's just a motor. But this subsystem will also have the capability to use PID control to move the lady brown arm very quickly and accurately.

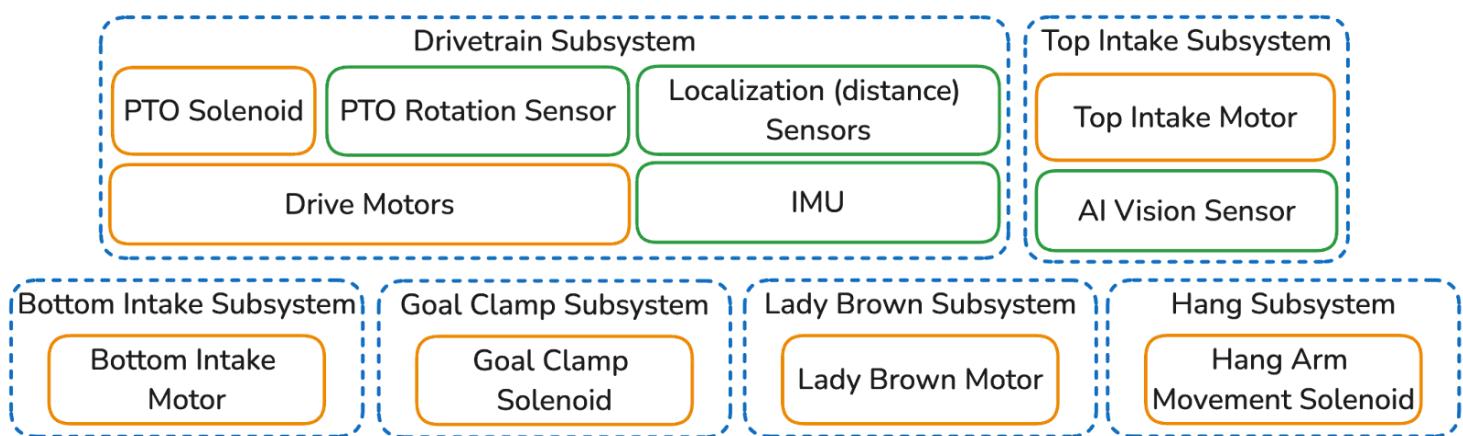
Goal Clamp and Hang Subsystem:

Both of these subsystems have a very simple solenoid only subsystem with control to set the current state of the value and get the last commanded value.

**Summary**

- Constructed layout of useful subsystems on the robot and their components
- Next steps
 - Implement subsystem code on robot (Completed: **03/03**)
 - Develop commands and macros for driver control (Completed: **03/04**)

Key: Sensors (Green Box) Actuators (Orange Box) Subsystems (Blue Dashed Box)



02/18/25

Design: CAD Day 3 (C.4.1)

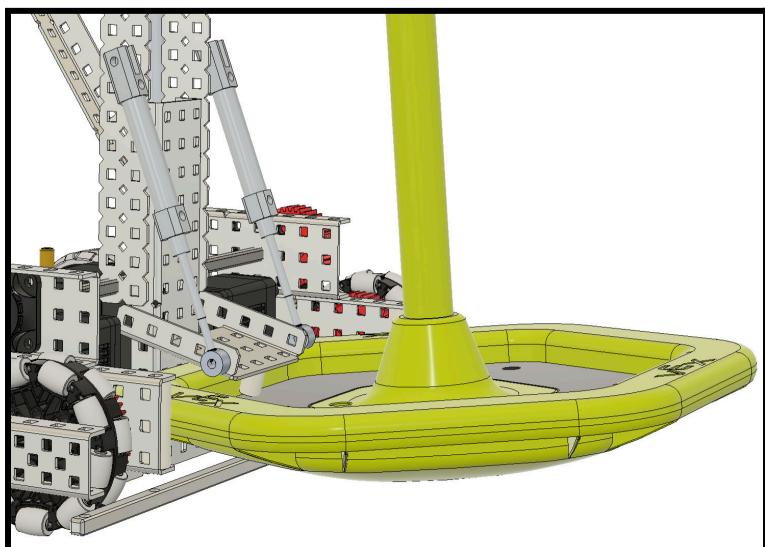
Designed by: Carl

Witnessed by: Alex

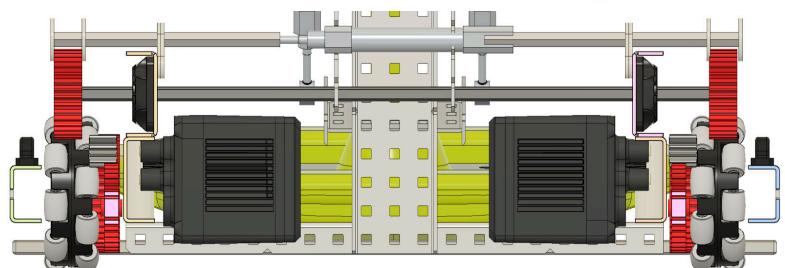
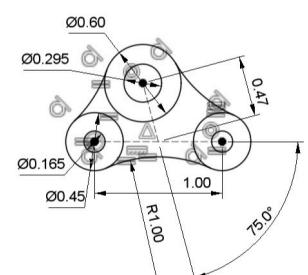
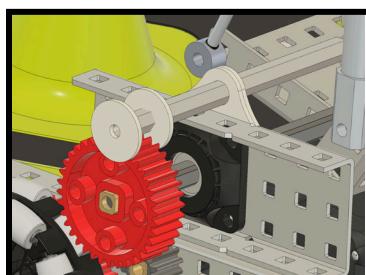
Witnessed on: 2/19/25

Goal: Implement the intake and back clamp into the CAD.**Back Clamp:**

As we identified [earlier](#) (Pg. 358) our back clamp on the last robot was very effective but lacked the ability to grab goals from every orientation. This will be addressed by only using a single 0.375 OD spacer to hold the goal into the robot while other spacers (not CADed) will push the lip of the goal down to tilt it. The pistons are mounted vertically on top of the clamp to maximize space within the robot for the inertial sensor and solenoids. The clamp itself is made up of a 5x1x5 C-channel and is pivoted off of a vertically mounted 3x1x12 C-channel. The pistons are mounted to aluminum plates that will also hold the top roller for the hook stage.

**PTO Piston Mount:**

The PTO on this robot functions nearly identical to our [previous one](#) (Pg. 264) with a piston moving the 36T gear on the winch into the drivetrain through the use of plastic discs. The piston was integrated above the winch using a double stacked polycarbonate bracket (pictured top right).



Top Intake (Hook Stage):

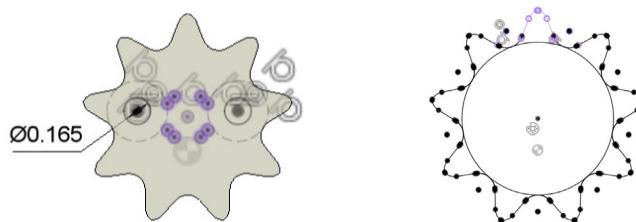
Most teams (including us) use a 400 RPM 12T sprocket to drive the hook stage of their intake. This speed is used because it is easy to make with a 2:1 gear and it provides a good balance of speed and torque. There are also some teams which utilize a 600 RPM 12T sprocket which is noticeably faster and lower friction due to the sprocket being directly driven by the motor. This speed works well when intaking single rings but often struggles to intake multiple at once.

We believe that the best speed for an intake is between these two speeds, however, we need a low friction way to change the motor's output. Based on our [gearing sheet](#) (Pg. 102), there are no reductions that can achieve an RPM from 400-600 with only using gears smaller than a 12T sprocket meaning gearing is not an option. Another option is changing the size of the sprocket to alter the linear speed of the chain. As we demonstrated with our old corner sweeper, it is possible to make custom sprockets with polycarbonate, so this solution is likely viable. Here are sprocket sizes that would yield viable intaking speeds along with the RPM equivalent on a 12T sprocket:

$$\textbf{9T: } 600\text{RPM} \cdot \frac{9}{12} = 450 \text{ RPM}$$

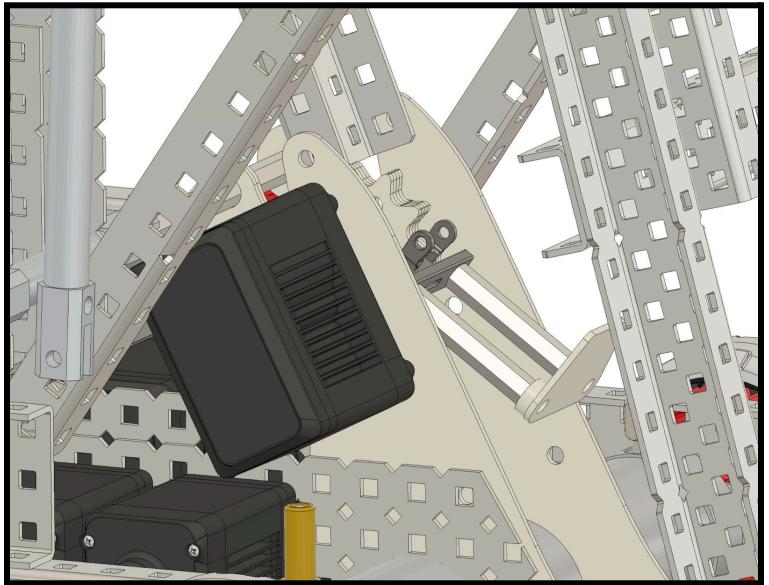
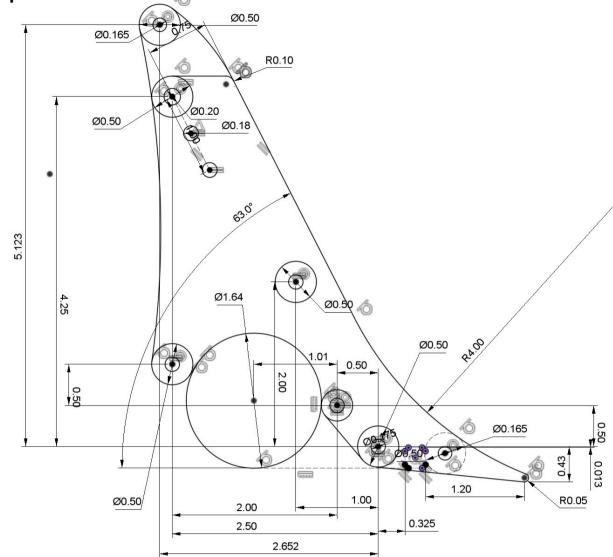
$$\textbf{10T: } 600\text{RPM} \cdot \frac{10}{12} = 500 \text{ RPM}$$

$$\textbf{11T: } 600\text{RPM} \cdot \frac{11}{12} = 550 \text{ RPM}$$



We selected to use a 9T sprocket as it offered the highest amount of torque while still achieving a 12.5% speed increase from our old intake. To design the sprocket, we projected a single tooth from an official 12T sprocket and patterned it around a center point ensuring proper meshing with the chain.

For the intake's structure, we chose to integrate the motor mount into the ramps which are also used to mount the reservoir. The ramps attach to the HS shaft crossbar with small cutouts and are secured to 5 wide aluminum plates creating a rigid and low profile connection with the 3x1x12 C-channel on the back clamp. A 3x1x19 C-channel attaches the top of the ramps to the top of the intake structure to reduce sideways bending in the plates.

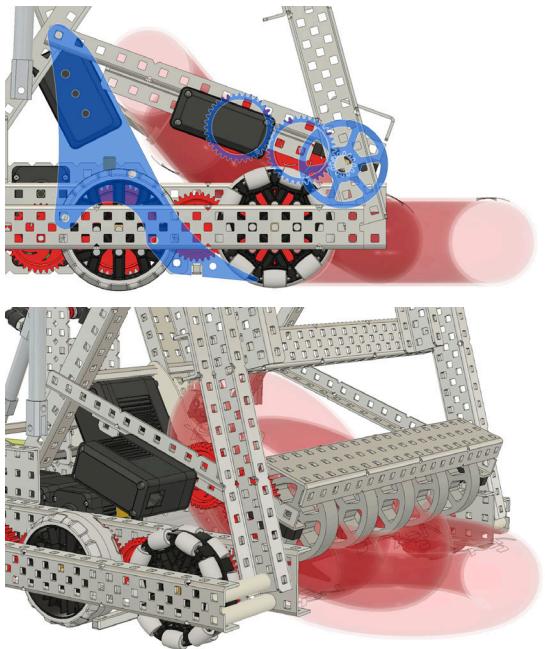


Bottom Intake (Flex Wheels):

In order for our robot to climb smoothly, the intake must be inside of the main uprights so that it doesn't snag on the rungs. Both sides of the intake are connected by a 3x1x19 C-channel to eliminate the twisting that our [last intake](#) (Pg. 357) had. This C-channel will also serve as a ram bar and might also be a good mount for the wall stake aligner.

We were forced to use 2" flex wheels in order to intake the rings further. This was necessary as the hook stage begins higher than on the last robot (to accommodate the air reservoir). In order to achieve a similar linear speed with the larger wheels, we will only be running a 600 RPM roller. The 5.5W motor is geared (36:12) to avoid any potential issues with chain snapping.

The main arms of the intake are made using 1x1x3 L-channels to save on weight and space. The arms are deliberately pivoted on the sixth hole of the triangle bracing so that a standoff can run between the two sides of the intake and the top of the intake ramps. Doing this allows for lightweight bracing of multiple components while also strengthening each individual connection.



Other Progress:

- Added other side of the hang for better visualization
- Rough positioning of the brain and battery
 - The battery is positioned opposite the intake motors to balance weight on each side of the robot
 - The brain is positioned close to the center of the robot to reduce the robot's moment of inertia (this should improve angular acceleration when turning)
 - Both are located as low possible to lower the robots CG
- Addition of small parts that were forgotten earlier



02/19/25

Design: CAD Day 4 (C.4.1)

Designed by: Carl

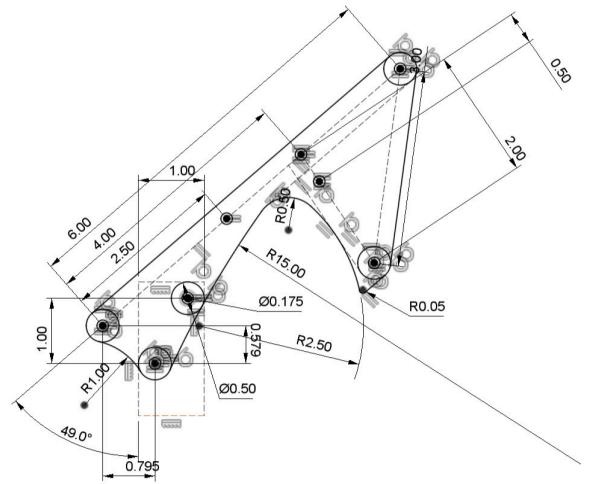
Witnessed by: Alex

Witnessed on: 2/20/25

Goal: Create polycarbonate hooks for the hang and prepare polycarbonate for laser cutting.
Finalize design to be ready for building by 02/20/25.

Bar Hook Design:

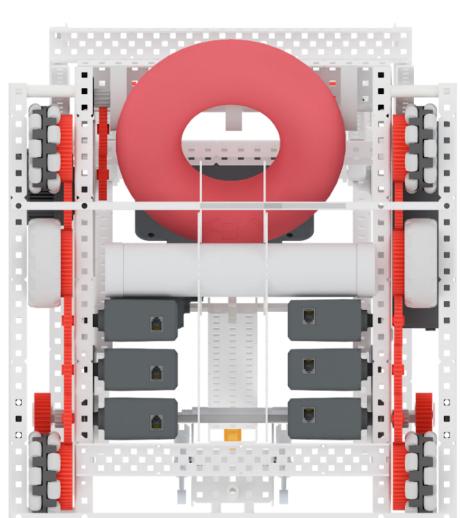
To hold the robot on the horizontal rung we will be using polycarbonate hooks similar to the last robot because they were very strong and customizable. We changed the design of the hooks on this robot to only interact with the rung as far down as possible to hopefully make the robot stay on better. We mentioned [earlier](#) (Pg. 369) that the uprights and triangle bracing needed to be attached with a custom polycarbonate bracket, and as the bar hooks are close to the same spot, we integrated the two parts together. The holes on the polycarbonate part are in 0.5" intervals to allow them to be reinforced by metal plates.

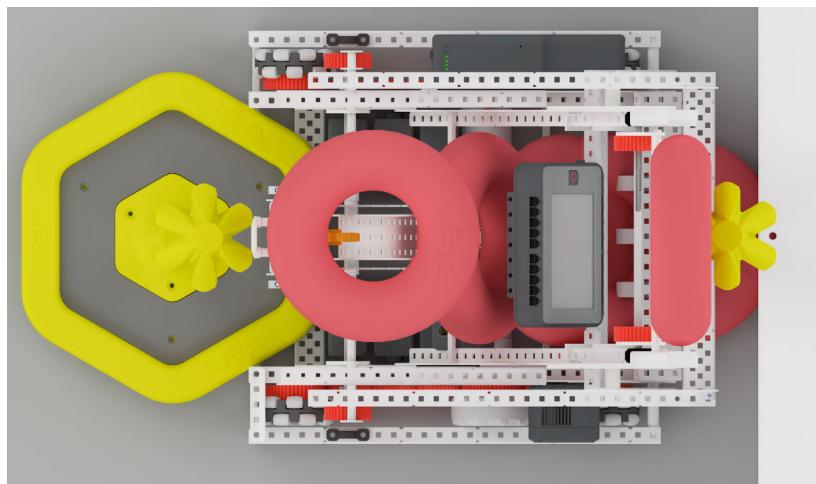
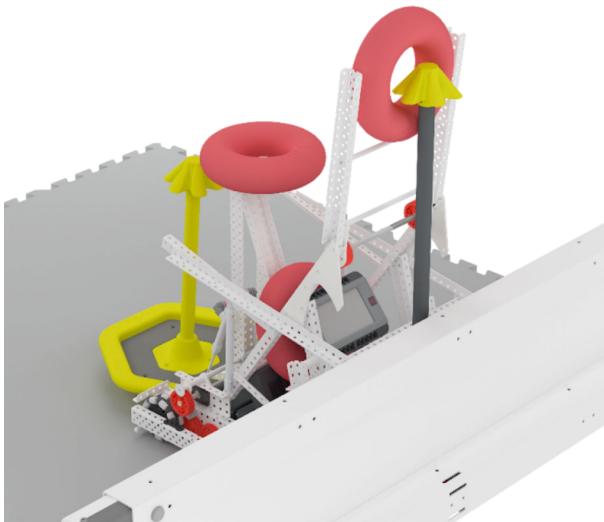


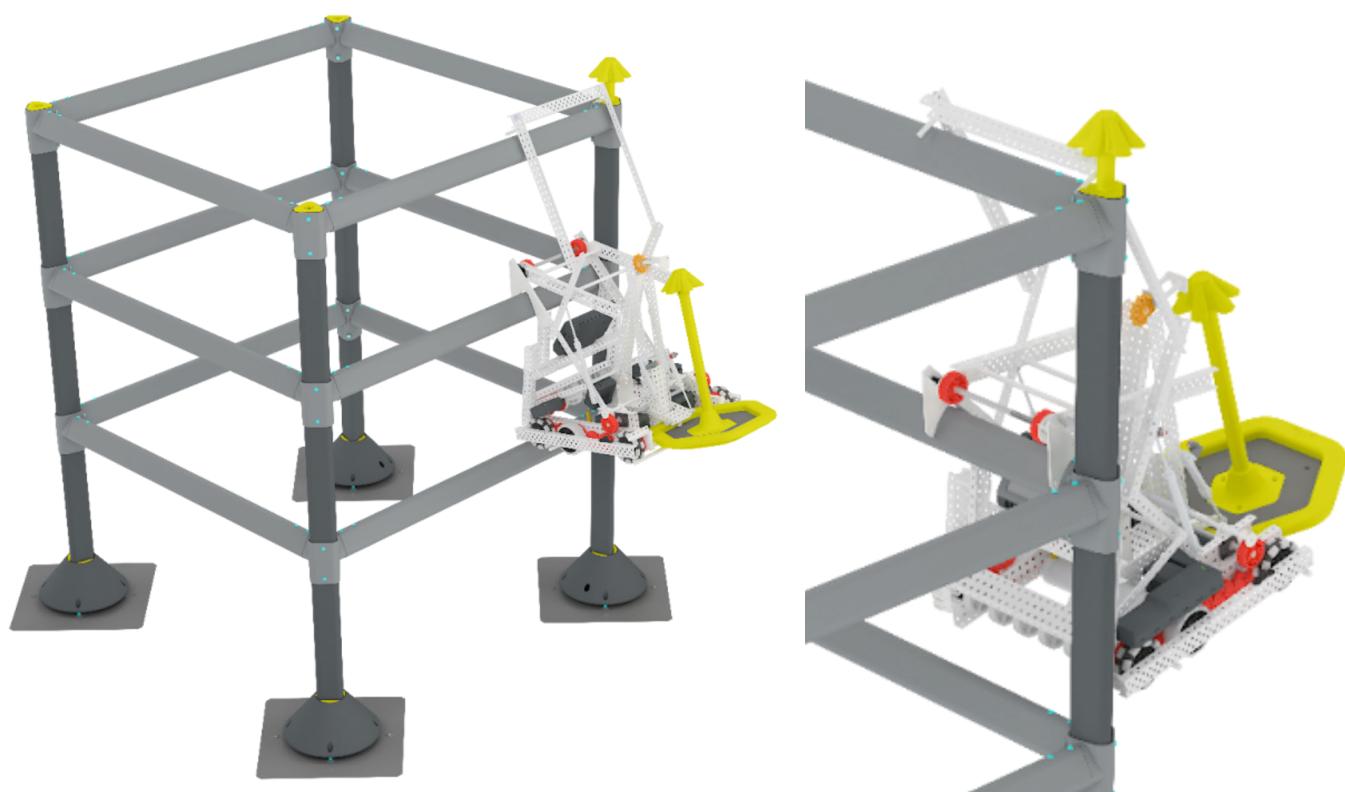
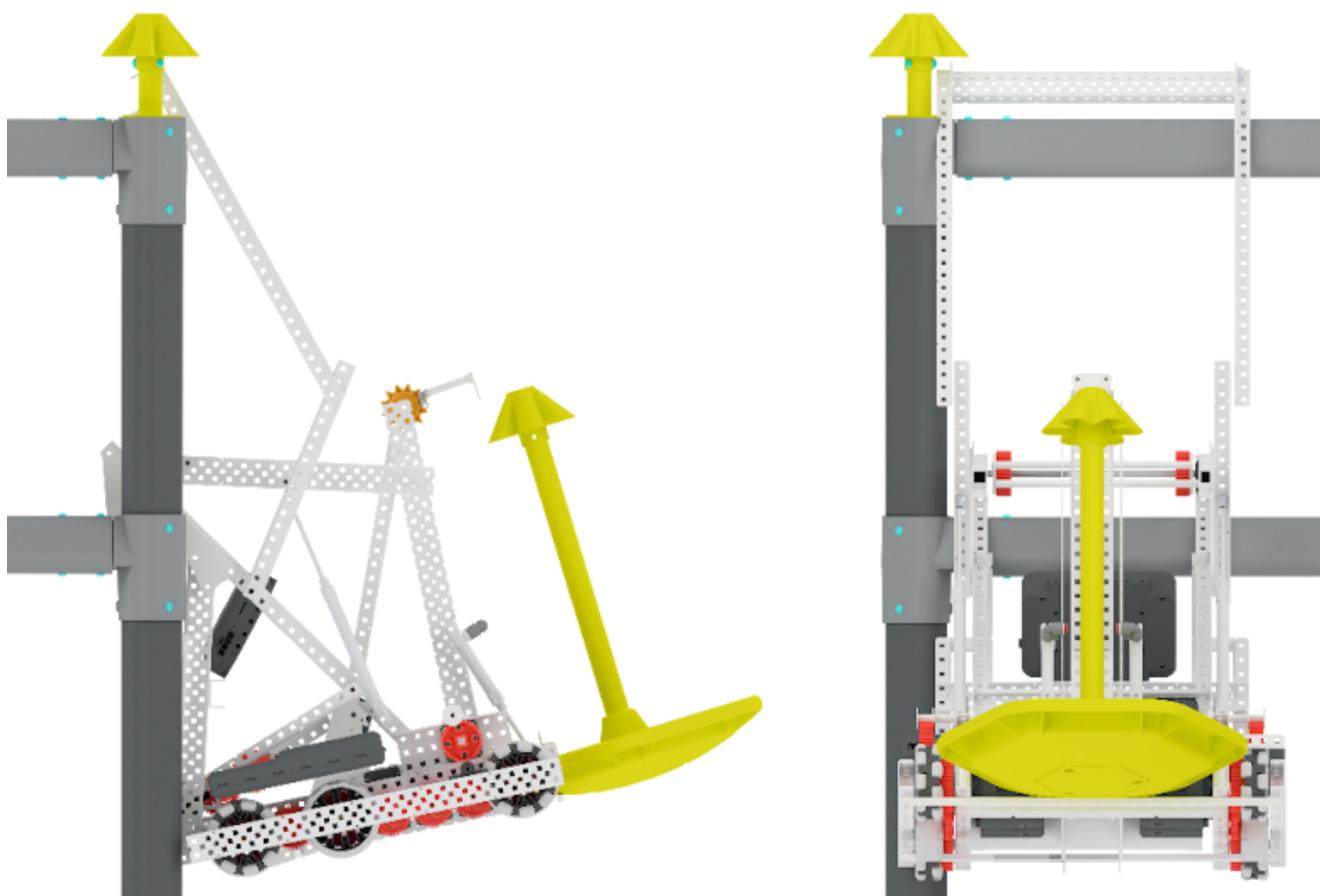
Hang Bracing:

In order to solve the [problem of the hang arms bending inwards](#) (Pg. 358), we added a 2x1x22 C-channel across the front of the arms. This will help link the motion of the arms together and prevent twisting allowing for a weaker (but also much lighter) pivot between the two arms.

Full Robot Renders (Reference for Building):







02/20/25 Strategy: Autonomous Reevaluation

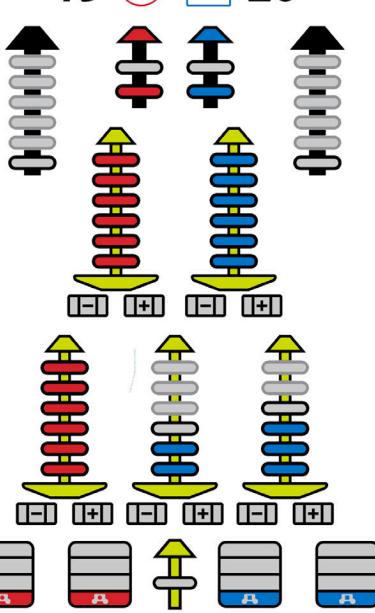
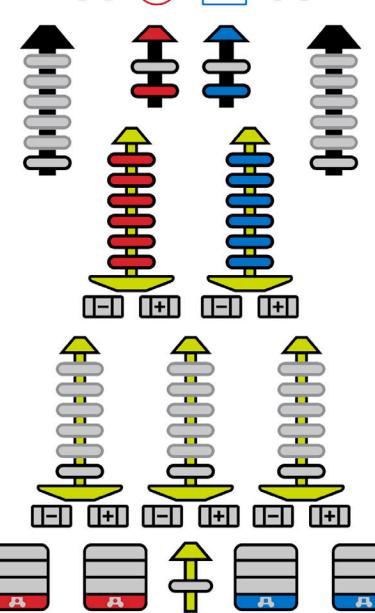
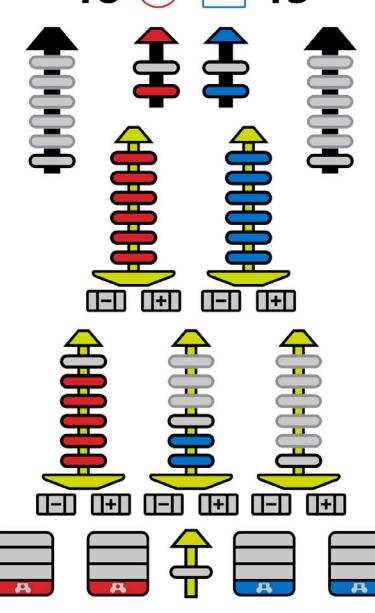
Designed by: Alex

Witnessed by: Carl

Witnessed on: 02/20/25

Goal: Develop a thorough data-driven strategy for the autonomous center goal rush.

Throughout this season, we have had the opinion that prioritizing the goal rush during the autonomous period should not be a priority of us or our alliance. Especially at events such as Kalahari and Colorado signature events we opted for a more consistent strategy where we filled the goal on our side for a better start to the match. However, while going to the score breakdown, we noticed this strategy relied on lower-scoring opponent autonomous routines, and at worlds we will have to assume that our opponents have the highest possible scoring autonomous routines. The score breakdowns in several potential matches are below:

Blue Rush Win	Tie Rush	Auton Hold
<p>19 (A) A 20</p>  <p>In this case we assume that both negative side autonomous routines are the same. As can be seen in this case, the blue team even though they score 1 less ring, they get 1 more point and would win autonomous.</p>	<p>11 (A) A 11</p>  <p>In this case we assume that both of the positive side alliances get stuck on the autonomous line for the entire autonomous period and don't score anything.</p>	<p>18 (A) A 15</p>  <p>In this case we assume that we hold our opponent at the line for as long as possible and then quickly fill our goal. In this case we would win autonomous but lose the third goal.</p>

Based on this analysis it shows that it is a necessity to have some way that your robot can win or at least tie every rush to avoid losing autonomous. However, getting more rings on one goal can be very advantageous to start the match in a better position by spending less time filling up the goals you already have.

Rush Options

Currently, many high-level teams are utilizing rush mechanisms to give them a time advantage when going to the center goal. The end goal of all these mechanisms is to pull the center goal back before the other team has the chance to get a grasp on it, winning the rush and allowing higher scoring autonomous routines with the top ring on that goal, and the third goal advantage in the match. To find the best options we will analyze rush mechanisms from this year and Tipping Point worlds where rush mechanisms were common

Rush Mechanism/Doinker:

Currently, many high-level teams are utilizing rush mechanisms that they add to their robot to get another couple inches of reach towards the goal.

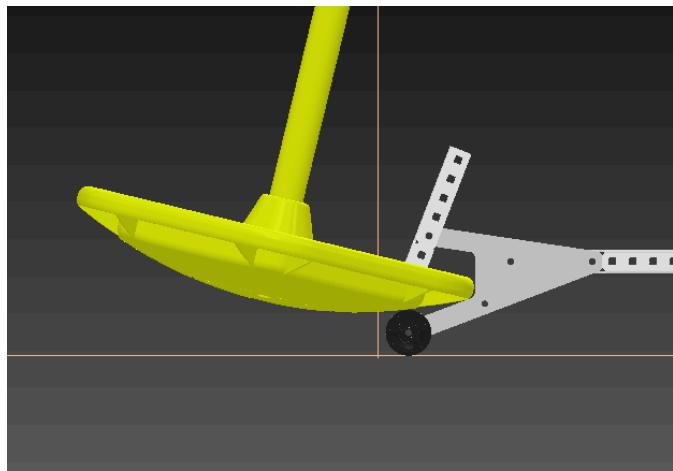


Figure: Rush arm in CAD grabbing a goal. Credit: 2775V



Figure: Goal rush being used in a match to grab a goal in autonomous. Credit: [Mecha Mayhem Finals 1](#), 2775V

VEXU Expansion:

Many VEXU teams are currently utilizing a string out the back of their robot taking advantage of the fact that only <SG1> states that only some of the robot has to be hanging over the line. This gives these teams an advantage by starting closer to the goal. These teams do this with their 24 in³ robots, but in VRC it can be done on the 18 in³ to get a couple more inches of reach in a similar way to the Rush Mechanisms.

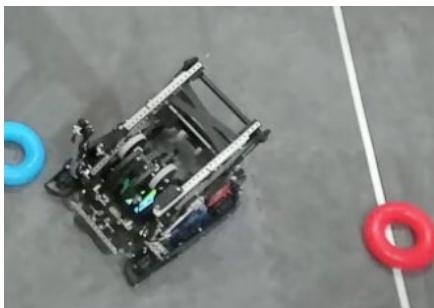


Figure: BLRS2 Autonomous Starting Position (Credit: [Purdue BLRS2, Illini Cornfield Clash F1](#))

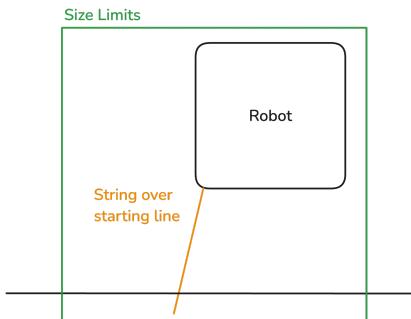
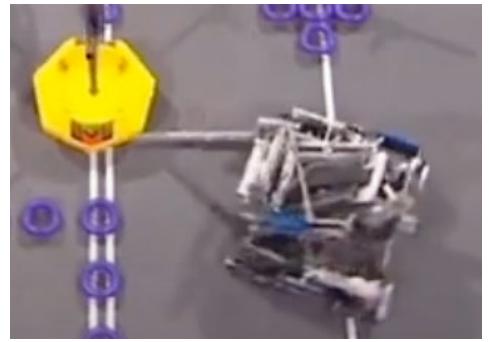


Figure: Diagram of String on BLRS2 Robot, Pulled in After the Start of Autonomous

Slapper mechanism:

In Tipping Point we saw several teams make a mechanism that went to the goal and slapped it out of the way. This approach was faster than the alternative of trying to pull it back, however, the same mechanism is not possible to replicate this year.

**No Rush Mechanism:**

Some teams even in Tipping Point would run without a rush mechanism and just go with their clamp on their lift or drivebase. We will assume that these mechanisms (back clamps) lose around 6 inches on any other rush mechanism because they have less reach. This will better inform the remaining calculations, and even though this might be an overestimation it will allow us to prove that a rush mechanism is needed without doubt.

Rush Analysis

There are countless videos on rush online that can give us much more information about rush and if any rush mechanisms have given a performance benefit to teams. To do this analysis we will be using the open-source [Tracker](#) video analysis tool to graph rush velocity profiles from various years.

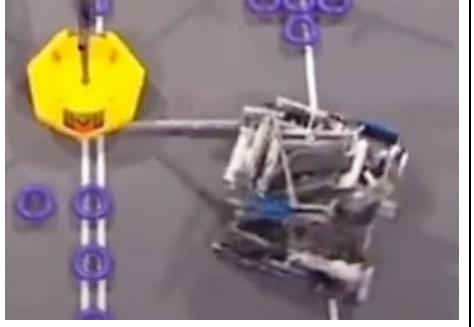
Method:

In the Tracker tool, we specifically use the point mass calculations with the autotracker feature to get more accurate velocity tracking than can be done manually. We will follow the following method to collect data from a variety of matches from Tipping Point and High Stakes to see if there have been any rush mechanisms that provide a significant advantage.

1. Open file in tracker
2. Create a point mass tracking object for each team in the match
3. Choose a point on the robot to track that doesn't move much relative to the drivetrain during the motion
4. Utilize the autotracker feature to track object throughout path with low evolve (~20%)
5. Ensure each point is in the same throughout the frames
6. Measure the time from when a robot gets to the goal to when that robot pulls the goal back far enough to not get taken back (pullback time)
7. Repeat steps 3-5 for all teams in a match
8. Graph the velocities over time for all of the robots

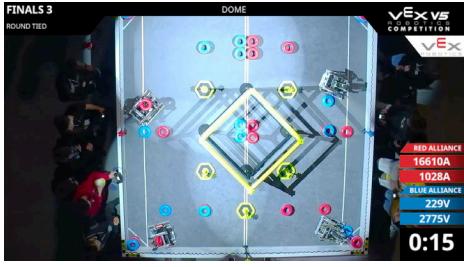
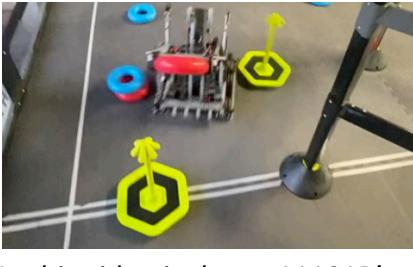
Tipping point:

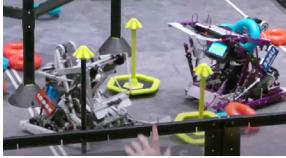
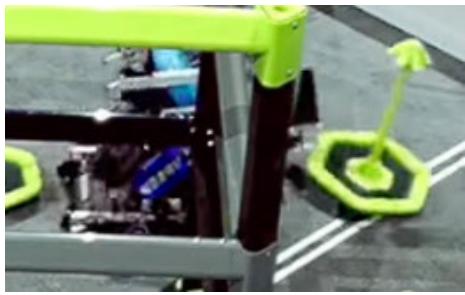
In Tipping Point there was a very heavy emphasis on the goal rush due to the amount of points that it was worth, often a make-or-break point for a match, even in qualifications, so teams spent a lot of time on the development of mechanisms and strategies to optimize rush, and this can be used as a starting point for the data collection.

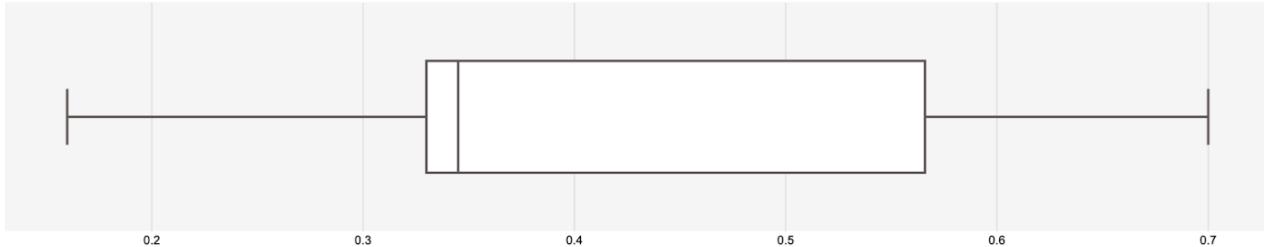
<u>Overall Finals 2</u>	<u>Overall Quarterfinal 3-1</u>	Notes: <ul style="list-style-type: none"> • 9257C was the only standout team with the pullback time, utilizing the slapper mechanism <ul style="list-style-type: none"> ○ This in practice moved the goal out of the way to grab much quicker (0.16s vs 0.33s) than the traditional grab and pull back technique
<p>Teams: 254F, 9257C, 38141B, 4154X</p> <p>Notes: Teams had very little deviation in velocity curves</p> <p>254F: 11 frames / 30 fps = 0.36s 9257C: 5 frames / 30 fps = 0.16s 38141B: 10 frames / 30 fps = 0.33s 4154X: N/A (in tugging battle)</p>	<p>Teams: 2114X, 8000A, 3796F, 4478E</p> <p>Notes: Same, very little variation in performance curves</p> <p>2114X: 21 frames / 30 fps = 0.7s 8000A: N/A (Tugging battle) 3796F: N/A (Tugging battle) 4478E: 12 frames = 30 fps = 0.39s</p>	

High Stakes:

This year many more teams have developed rush mechanisms at this point in the season. As we established in TiP matches the velocity curves are very consistent from robot to robot, so we will just analyse the time that teams are holding the goal at the center if there is no great data for position. We will go through and analyze 6 videos from this season:

<u>Mecha Mayhem F1-3</u>	<u>11101B Barcbots Rush (Private video)</u>	<u>Pikes Peak Signature Event F1-1</u>
<p>(Messed up frame rates caused video analysis to be useless)</p> <p>Pullback Time: 0.566s</p>  <p>In this match 2775V and 16610A had a rush where 2775V decisively won.</p>	<p>(Moving camera caused video analysis to not be possible)</p> <p>Pullback Time: 0.333s</p>  <p>In this video it shows 11101B's auton, one that has performed very well at multiple signature events. This one is fairly fast to pull back at 10 frames or 0.33s</p>	<p>(Camera was not in a good position for analysis)</p> <p>Pullback Time: 0.333s</p>  <p>In this match 11101B has a very similar time to pull back as they do in practice, beating out the opponents not because of speed, but because their rush arm stays in the goal better.</p>

Pikes Peak Signature R16 2-1	Pikes Peak Signature Event QF1-1	Finals 3 WPI
(Camera was not in a good position so video analysis could not be completed on rush)  In this match both teams had very effective goal clamps, and neither team won the rush. In this case, 1239E got pulled across the line. This shows that with good rush mechanisms there is almost no way to win the rush. 	(Camera was not in a good position so video analysis could not be completed on rush)  Pullback Time: 0.6s Even though 11101B got to the goal at the same time as 334U in this rush, because of the design of their doinker mechanism, they lost a grasp on the goal, losing it in auton.	(Poor angle for video analysis)  Pullback Time: 0.23s In this auton, 4024V has a very unique approach to rush that involves rotating the robot similar to 9257C did in Tipping Point. This brought the goal back very quickly, and is on the brink of being able to beat a team without a goal rush

Pullback Time (seconds):

In this chart it can be seen that the vast majority of rush mechanisms pull back the goal in the 0.35s area, but there are a couple outliers that are able to pull back the goal as fast as 0.16s (9257C from Tipping Point) and as slow as 0.7s (2114X, special case with a goal blocker mechanism)

Rush Simulation

While we can analyze previous rush mechanisms from previous years and even this year, another helpful tool to get a more analytical understanding of the differences in rush would be a simulator that uses the actual acceleration curves of the real VEX motors on a robot, which differs significantly from the ideal motor models, or just constant acceleration. Before going through the math we must first make a couple of assumptions, and those assumptions are justified in the following:

- Motors get up to full voltage instantaneously
 - Both teams motors should act the same
 - Rise time should be less than 1ms, so is largely inconsequential for comparison in this situation
- Motors stay at truly 12V
 - There is a bug where small changes in voltage decrease the motors output torque significantly, however this often isn't an issue during rush because the motors stay at the full 12V
- Motors follow very closely to the advertised performance across the curve
 - This should affect every team equally, so the effects should be small
- Robot drives in a perfectly straight line
 - Accounting for rotation would be a confounding variable in this study, additionally this is an assumption that is possible in this game
- Back EMF doesn't play a significant role in deceleration
 - Back electro-motive force is the force that motors have pushing back at all times and it makes it so the motor can't just spin freely without friction
- The robots all weigh the same amount
 - After careful analysis the amount of time that building a lighter robot takes is minuscule to other effects, such as the gear ratio, so this isn't considered as heavily

In our analysis we will only calculate the 1-dimensional position of the robot over time at the maximum possible acceleration of the robot at each individual point, given the torque curves on the VEX robotics website:

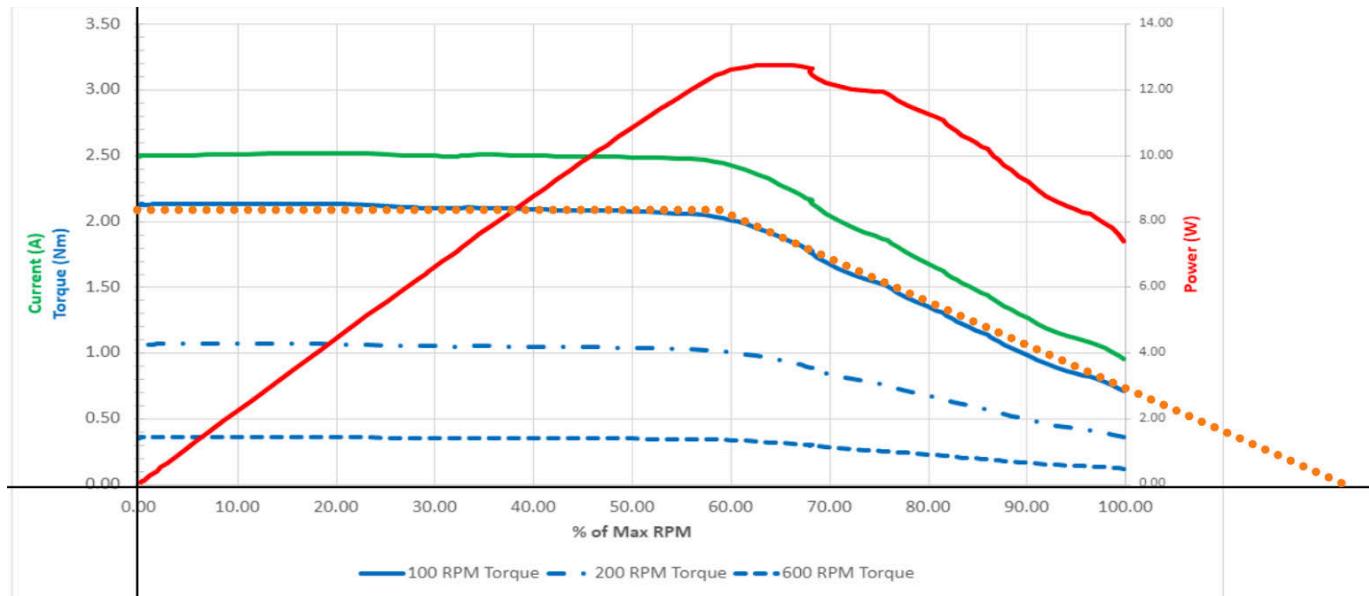


Figure: Illustration of torque for the V5 smart motors (100 rpm red cartridge shown in solid blue) (from [VEX Knowledge Base](#)) with an overlay of our torque model (dotted orange line) to predict motor torque at any speed.

Drivetrain Movement Equations

Where τ is torque, n is the number of motors, and r is the radius of the wheels:

$$a(t, s) = \tau(s) * n * r$$

Torque is calculated with the model shown in the last figure given the input of the current motor speed. This method of calculating torque allows the model to be more accurate to the true drivetrain model. To get the position of the robot we use a double [euler approximation](#) of the robot's acceleration to get the velocity and position of the robot over time.

```
for i in range(1, len(time)):
    dt = time[i] - time[i-1]
    direction = -1 if 0.91/2 < distance[i-1] or direction < 0 else 1 # Decel in time
    speed[i] = speed[i-1] + acceleration(speed[i-1]) * dt * direction
    distance[i] = distance[i-1] + speed[i-1] * dt
```

The results of the speed and distance are then graphed to get an idea of where the robot is at any given time.

```
import matplotlib.pyplot as plt
import numpy as np
import csv
import math
from utils import *

# Constants
ratio = 36.0 / 48.0 # input / output
gear = Gear.BLUE
wheel_radius = 0.035 # In meters
robot_mass = 6.8 # In kg
motor_count = 6
maxTime = 1.2

# Time array
time = np.linspace(0, maxTime, 500) # 10 seconds, 500 points

# Acceleration function
def acceleration(speed):
    return calculate_torque(math.fabs(speed), gear, ratio, wheel_radius) * motor_count / (wheel_radius * robot_mass)

# Calculate speed by integrating acceleration
speed = np.zeros_like(time)
distance = np.zeros_like(time)

direction = 1

for i in range(1, len(time)):
    dt = time[i] - time[i-1]
    direction = -2 if 1.25/2 < distance[i-1] or direction < 0 else 1
    speed[i] = speed[i-1] + acceleration(speed[i-1]) * dt * direction
    distance[i] = distance[i-1] + speed[i-1] * dt

# Plotting omitted for brevity
```

Results:

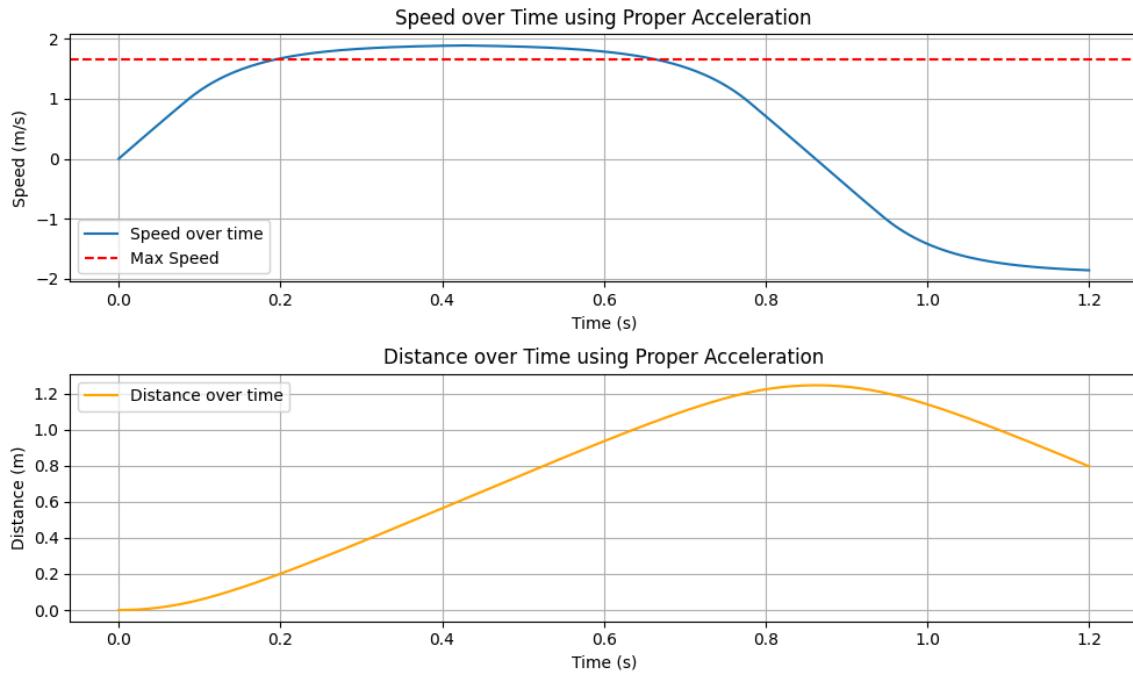


Figure: Illustration of velocity and position curves during a 1.22m rush movement on a 480 rpm, 6 motor, 2.75 inch drivetrain.

Notes:

- Drivetrain appears to spend a lot of time in the slowest sections of movement right at the very end and beginning.
 - At the end most teams will spend about .2-.3 seconds within 3 inches of the target, where a goal could potentially be stolen
 - In this range we could potentially not lose a rush, so if we can get there within this time there is almost no way to lose a rush.

Simulation vs. Real World

Upon graphing the simulation values to the real world data collected in the video analysis tool, we noticed that the deceleration at the end of the path was much higher than was predicted by the simulation. This is partially attributable to the opponents pulling on them right after in the match, but upon further research, we realized that this effect was due to the [Back EMF](#) present in the motors. This effect causes the motor to have a much greater braking effect (powered in opposite direction as the motor is moving) than pushing effect, leading to much higher acceleration than our move predicted.

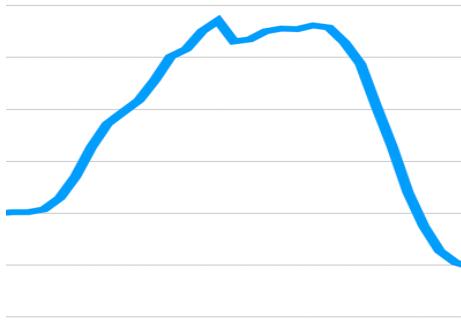


Figure: Velocity over time in the real world, gathered from video analysis from 254F in [F2 of Tipping Point World Championship](#)

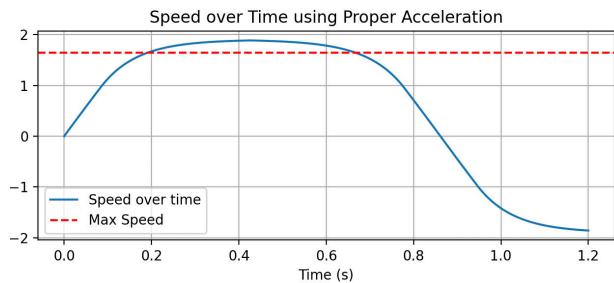


Figure: Predicted velocity profile during rush. Notes: the predicted speed decreases significantly slower with this model than we see in real life. (left)

Looking more into how back EMF works, and the graphs that were generated, we think it would be fair to assume back EMF effectively doubles the torque of the deceleration movement, and it can be calculated as such. With these adjustments the graph looks like the one on the right. Additionally, after looking at the differences in time given these new accelerations, the time differences between the teams with rush mechanisms and no rush mechanisms is only less because the robot will spend more time at full speed.

Conclusion

- Two teams that have well built goal rush arms will almost certainly tie
 - To win against another goal rush you would need to get to the goal in 0.3-0.35 sec
 - This would require a rush in around half of the current fastest rush time (basically impossible)
- Without a rush arm, we might lose to mechanisms that rotate, flip, or spin the goal and make it more difficult to grab
- Without a rush arm, we will **tie** against almost all currently competitive rush arms
 - It takes around 0.7 sec for a robot with a rush arm to get to the goal **and another 0.2-0.4 sec** to pull it out of reach
 - It would take about 0.75 sec for a robot **without** a rush to get to the goal **and another 0.2-0.4 sec** to pull it out of reach
 - A rush arm does not provide enough of an advantage to beat a robot without one
- Rushing with the back clamp increases chance of winning the pulling fight as the goal is centered so the drive can exert more force than if the goal was being held on the side of the robot

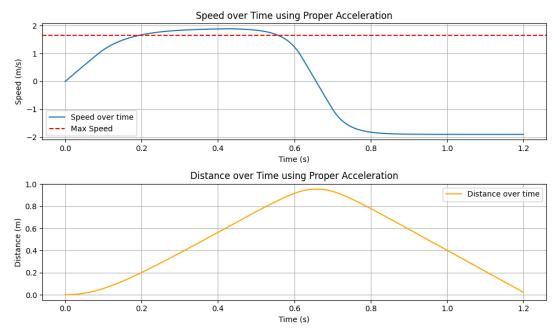


Figure: updated simulation with greater deceleration with assumed back EMF

(Notebook Submitted for Worlds on February 20th)

APPENDIX A: COMMUNITY CONTRIBUTIONS

Section Goals

In this section we wanted to show the most notable community outreach that we do as a team. This is by no means an exhaustive list, but these are a few especially notable things that we wanted to show. For this reason, none of these entries are attributed to a single person, and they aren't dated.

Mentoring and Leadership—Longmont High School Robotics

Overview:

Starting with two teams three seasons ago, we have helped our club evolve into a competitive 6 team organization. This year, all three of us have adopted leadership positions to help share our skills and expertise with as many students at our school as possible.

Leadership Positions:

President (Alex):

As President, Alex works with the Vice President and other leaders to meet club goals. He helps plan what tournaments teams go to, helps teams prepare for those tournaments, and manages club activities to keep everything running smoothly.

Vice President (Matt):

As Vice President, Matt collaborates with team leads to plan topics and dates for presentations and workshops. He ensures communication with school staff and coaches about changes and other items. Matt takes on leadership duties when needed, supports club initiatives, and handles any issues that come up.

Build Lead (Carl):

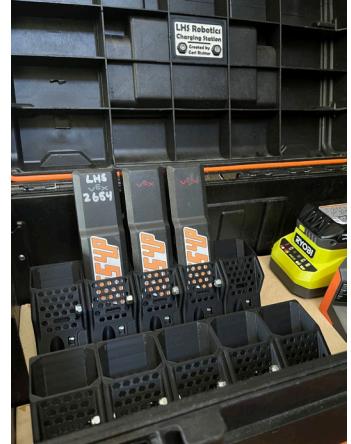
Carl is responsible for creating workshops and giving presentations to the entire club about robot construction and other build techniques. Carl also works with teams on an individual basis to help them troubleshoot and advise them with specific tips.

Parts Organization & Resource Management:

As there are five teams in our club, staying on top of part inventory is extremely important. Because we have been competing for several years, we can leverage our experience to predict what parts teams will need more of, less of, and what parts will need replacement. This helps all of the teams in our organization be successful while simultaneously helping us ensure that we have the necessary resources for our robot.

Portable Charging Station:

Over the summer, we fully developed a custom charging station that is able to charge controllers, tool batteries, and 10 V5 batteries simultaneously. This station helps keep the robotics room organized in addition to being incredibly helpful to all of our teams at tournaments, leagues, and scrimmages.



Competition Contributions

Overview:

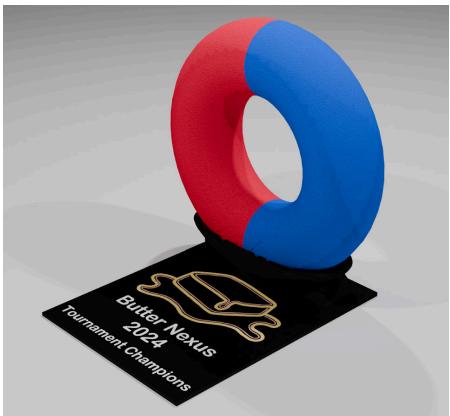
As competitors in VEX Robotics, we always appreciate the volunteers that make these events possible. This is why we contribute back to the community and give these same opportunities to other future engineers. In this section we will break down just a couple of the ways that we contribute to making competitions in Colorado possible.

Robotics Leadership Team:

As a part of our district's robotics leadership team we assist with the running of many VEX IQ and V5RC competitions throughout the district.

Award Design/Manufacturing:

For the Butter Nexus League and CECFC Winter Tournament we contributed to the design and manufacturing for all of the awards. Images are below:



Public Programming Contributions

Vexide Improvements

Contribution log:

All links refer to issues or pull requests on github.com/vexide/vexide.

Contribution Type	Description	Link
Pull request	Makes display widgets public for easier display use.	#81
Testing	Tested the AI vision sensor support on vexide	#58
Pull request	Text metrics getters (width/height)	#83
Pull request	Ensure safety in text screen printing, fixing malformed varargs	#84
Pull request	Add text alignment feature to Text widget API	#85
Issue	Reported issues with the Distance sensor API	#94, #115
Pull request	Improve panic hooks API	#118

Command Based PROS

We made a [Command Based](#) library in PROS inspired by [WPILib's state machine implementation](#). We created extensive documentation on this library we created and released it publicly. The documentation for the library is [here](#). We have heard from multiple teams online that they have used this library in their competition code.

Contribution Type	Description	Link
Commit	70 commits to the repository. Created the initial framework for this library	main
Documentation	Created Read the Docs site with 15 unique pages	docs

APPENDIX B:

EXTERNAL

SOURCES

Links

All links in the notebook with a page number next to it in this form link (Pg. #). For all external links they are here to be able to access them easily.

Page	Name	Link
N/A	VEX Robotics	https://www.vexrobotics.com/
55	LTC Robotics YouTube	https://www.youtube.com/@ltcrobotics6271
56	Vex Tipping Point Early Season Reveal	https://www.youtube.com/watch?v=35HMOQGIZHc
56	24C VEX Round Up Programming Skills	https://www.youtube.com/watch?v=uxV6C71yhm4
71	VEXCode C++	https://www.vexrobotics.com/vexcode
71	VEXCode Python	https://www.vexrobotics.com/vexcode
72	Pros C++	https://pros.cs.purdue.edu/v5/pros-4/index.html
72	vex-rt (Rust, QUEEN)	https://gitlab.com/qvex/vex-rt
72	Vexide(Rust)	https://pros.rs/
73	Visual Code Studio	https://code.visualstudio.com/
73	RustRover	https://www.jetbrains.com/rust/
73	CLion	https://www.jetbrains.com/clion/
74	Sublime Text	https://www.sublimetext.com/
74	ZED	https://zed.dev/
75	Git	https://git-scm.com/
75	Mercurial	https://www.mercurial-scm.org/
75	Subversion	https://subversion.apache.org/
76	vexide Official Example Repository	https://github.com/vexide/vexide-template
77	Fusion 360	https://www.autodesk.com/products/fusion-360/overview?term=1-YEAR&tab=subscription
77	Autodesk Inventor	https://www.autodesk.com/products/inventor/overview?term=1-YEAR&tab=subscription

77	Solidworks	https://www.solidworks.com/
77	OnShape	https://www.onshape.com/en/
78	VRC Fusion Library v2.0.2 Release	https://github.com/vindou/VEX-CAD-Fusion-360-Library/releases/tag/latest
99	Inverse Kinematics in 2D	https://www.alanzucconi.com/2018/05/02/ik-2d-1/
100	p5.js script for arm kinematics	https://editor.p5js.org/ad101-lab/full/oSOYs38kN
106	Official rust style guidelines	https://doc.rust-lang.org/nightly/style-guide/index.html
114	Minnesota Signature Event (Day 1)	https://vimeo.com/989615044
114	Minnesota Signature Event (Day 2)	https://vimeo.com/994119230
114	Minnesota Signature Event (Finals)	https://www.youtube.com/watch?v=Frelia6LLTI
117	2775V Jackson Area Robotics Pits & Parts	https://www.youtube.com/watch?v=wSc28QtjMv0
117	360X MOA Recap (youtube)	https://www.youtube.com/watch?v=JttAwTdMpHI
117	360X Full Circle Pits & Parts	https://www.youtube.com/watch?v=zfb0GDc2C_Y
117	11101B Barcbots Getting There Pits & Parts	https://www.youtube.com/watch?v=DbVMSuxRuag
117	81988Y High Stakes Reveal	https://www.youtube.com/watch?v=l0gzer3sFUo
117	1233H MOA Reveal	https://www.youtube.com/watch?v=0lbh8jtKHxw
117	8110W Whisper Pits & Parts	https://www.youtube.com/watch?v=G-EqAA0lfMU&t=58s
127	Monte Carlo Localization for Mobile Robots (Particle Filter)	https://www.ri.cmu.edu/pub_files/pub1/dellaert_frank_1999_2/dellaert_frank_1999_2.pdf
129	Markov Localization for Reliable Robot Navigation and People Detection	https://www.ri.cmu.edu/pub_files/pub1/fox_dieter_1999_3/fox_dieter_1999_3.pdf
130	Kalman Filter	https://www.uniteddhc.com/DSP/Kalman1960.pdf
157	FRC WPILib command based-programming	https://docs.wpilib.org/en/stable/docs/software/commandbased/command-scheduler.html
157	Command Based PROS GitHub Repository	https://github.com/alexDickhans/command-based-pros

157	Command Based PROS Documentation	https://command.alex.dickhans.net/latest/
177	2654E/P open-source path planner	https://github.com/alexDickhans/al_planner
191	Controls Engineering in FRC	https://file.tavsys.net/control/controls-engineering-in-frc.pdf
202	Highlander Summit Signature Event (Day 1)	https://www.youtube.com/watch?v=XO34Vi8LIQo
202	Highlander Summit Signature Event (Day 2)	https://www.youtube.com/watch?v=kjrg5qqv6y8
202	Howling Halloween At The Creek	https://www.youtube.com/live/HfrAYMHnWgg
212	Optical Sensor	https://www.vexrobotics.com/276-7043.html
226	Figma UI Design App	https://www.figma.com/
227	Flutter	https://flutter.dev/
229	Particle filter visualization skills 11/5	https://youtu.be/bH00zEN0BQI
244	FRC 95 The Grasshoppers Traversal Hang	https://www.youtube.com/watch?v=yF81_xCS6IY
256	Lady Brown (4042V)	https://www.youtube.com/watch?v=TKFtAeEy7SI&t=13s
256	Echo Mech (2654E)	https://www.youtube.com/watch?v=gpPzAisGwsY
256	Fish Mech (99904E)	https://www.youtube.com/watch?v=EeLUeFFbA_s
256	Redirect (5150Z)	https://www.youtube.com/watch?v=QPA_hxJhUs
278	WPILib System Identification Documentation	https://docs.wpilib.org/en/stable/docs/software/advanced-controls/system-identification/introduction.html
278	Controls Engineering in FRC	https://github.com/calcmogul/controls-engineering-in-frc
326	ESP32-C6 MINI-1 from espressif	https://www.espressif.com/sites/default/files/documentation/esp32-c6-mini-1_mini-1u_datasheet_en.pdf
326	TPS561201DDCR	https://www.ti.com/lit/ds/symlink/tps561201.pdf
326	5301-4P4C	https://wmsc.lcsc.com/wmsc/upload/file/pdf/v2/lcsc_2207051802_EVERCOM-5301-4P4C_C3097715.pdf
326	THVD1450DR	https://www.ti.com/general/docs/suppproductinfo.tsp?distId=10&gotoUrl=https%3A%2F%2Fwww.ti.com%2Flit%2Fgpn%2Fthvd1450

327	ESDCAN24-2BLY	https://www.st.com/en/protections-and-emi-filters/esdcancan24-2bly.html
328	KiCAD	https://www.kicad.org/
329	JLCPCB	https://jlpcpcb.com/
335	69 Point Skills Run (2654E)	https://youtu.be/IF8JOxUgz3k
346	1687C Genesis Pits & Parts High Stakes	https://www.youtube.com/watch?v=reU7aXCg8E8
346	39V Volt Pits & Parts High Stakes Robot	https://www.youtube.com/watch?v=_V_zLzCh04
359	2654E Echo Tier 3 Climb VEX High Stakes	https://www.youtube.com/shorts/kFnDe_ClqSw
359	2654E Echo Winter Reveal VEX High Stakes	https://www.youtube.com/watch?v=X0uw_-r3_tI
359	2654E Echo World Record Skills VEX High Stakes	https://www.youtube.com/watch?v=PDdXTecw3O1
359	2654E Echo Robot Explanation VEX High Stakes	https://www.youtube.com/watch?v=Wj1L_OZ68vs&t=794s
362	VRC High Stakes 2775V Sugar Rush Reveal	https://www.youtube.com/watch?v=VK8AK56XKaA
362	1082E & 1082M Texas R5 State Reveal	https://www.youtube.com/watch?v=cLF06QpuPtk
362	VEX HIGH STAKES fish mech scoring	https://www.youtube.com/shorts/6hi8UCelCnk
364	T3 Hang Explanation 5203G Gremlin	https://www.youtube.com/watch?v=aYHmwZHDbaU
364	8076X Tier 3 Climb V5RC High Stakes	https://www.youtube.com/watch?v=DbQt0K-1XOs
364	3004A qwerty Pits & Parts High Stakes Robot	https://www.youtube.com/watch?v=hQpB098lzuc
364	Tier 3 climb - Smilebots team 13176A. Vex high stakes	https://www.youtube.com/watch?v=BB9SXtIqDJs&list=PLSsRWcqG4qu6kWJiVxwfqdmW2Ehhs6QRC&index=8
380	Mecha Mayhem High Stakes F1-3	https://www.youtube.com/watch?v=1uGY_qF0tG8
380	Purdue BLRS2, Illini Cornfield Clash F1	https://www.youtube.com/live/pKC_5U7-tAk?si=WEzlQFnUseNSOFmu&t=20811
381	Tracker video analysis tool	https://opensourcephysics.github.io/tracker-website/

External Sources

382	Tipping Point Overall Finals 2	https://www.youtube.com/watch?v=AByb4KQGMGc
382	Tipping Point Overall Quarterfinals 3-1	https://www.youtube.com/watch?v=l7T-b0LxQDw&t=95s
382	Pike Peak Signature Event F1-1	https://www.youtube.com/live/hXAIHoWtMn8?si=wcn cYFeQAjk0lJKm&t=20763
383	Pikes Peak Signature R16 2-1	https://www.youtube.com/live/hXAIHoWtMn8?si=V-If yy8T4sRQVeXY&t=15655
383	Pikes Peak Signature Event QF1-1	https://www.youtube.com/live/hXAIHoWtMn8?si=q6jF Ev4mCMOSEDN6&t=18467
383	Wave at WPI Finals 3	https://www.youtube.com/watch?v=u1qGkq8g92I
384	Understanding V5 Smart Motor (11W) performance (VEX Knowledge Base)	https://kb.vex.com/hc/en-us/articles/360044325872-U nderstanding-V5-Smart-Motor-11W-Performance
385	Euler Method Approximation (Wiki)	https://en.wikipedia.org/wiki/Euler_method
386	Back EMF (Wiki)	https://en.wikipedia.org/wiki/Counter-electromotive_f orce
387	Tipping Point Overall Finals 2	https://www.youtube.com/watch?v=uYDjAl4q648

License

[2654E Engineering Notebook](#) © 2024-2025 by [Alex Dickhans](#), [Carl Richter](#), Matt Dickhans is licensed under
[Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International](#) 