VOL 1 ENGINEERING JOURNAL

502W

SD41 robotics

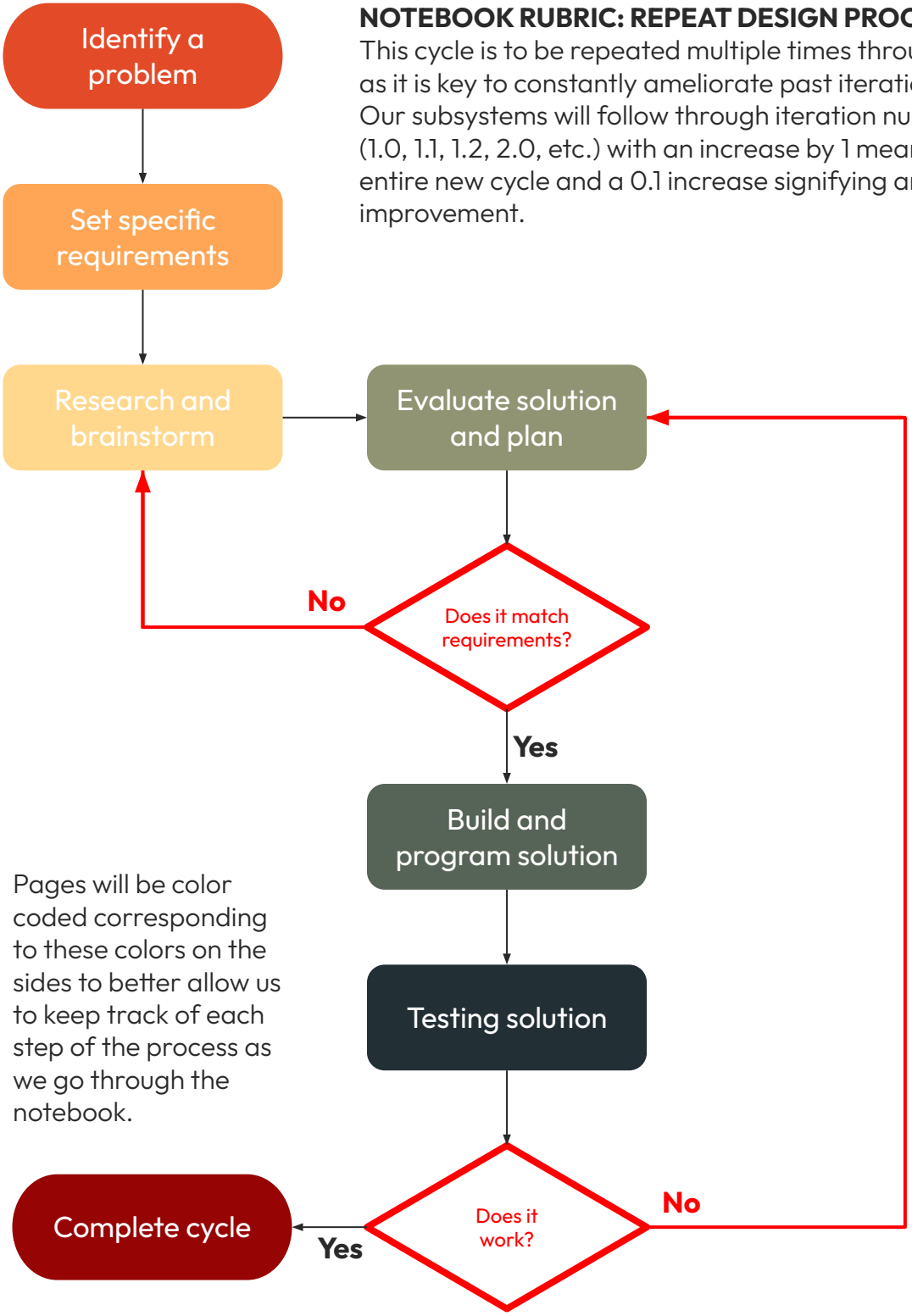
THE SWARM

502W

AUTHOR; RACINE LIU

DATE; 10.14.2024

DESIGN PROCESS

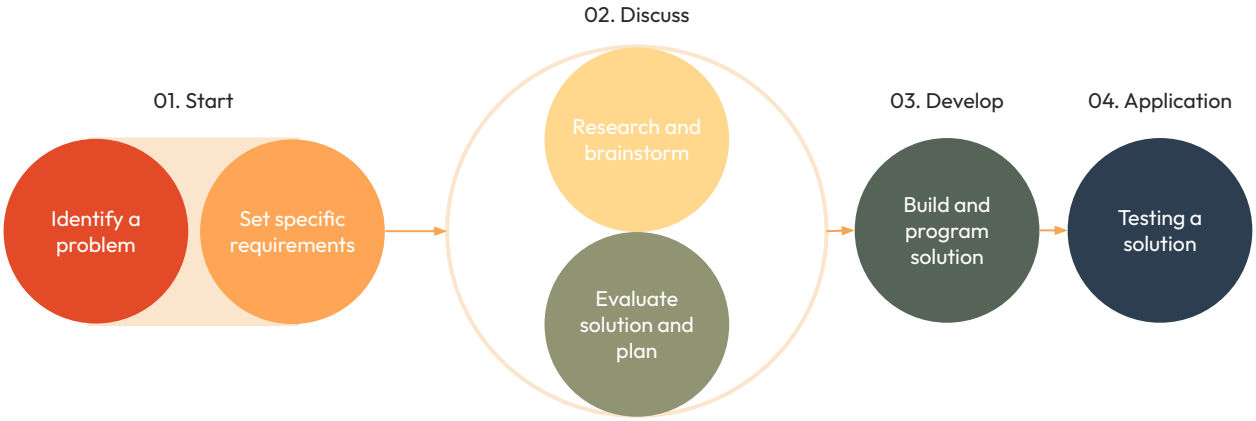


NOTEBOOK RUBRIC: REPEAT DESIGN PROCESS

This cycle is to be repeated multiple times throughout, as it is key to constantly ameliorate past iterations. Our subsystems will follow through iteration numbers (1.0, 1.1, 1.2, 2.0, etc.) with an increase by 1 meaning an entire new cycle and a 0.1 increase signifying an improvement.

Pages will be color coded corresponding to these colors on the sides to better allow us to keep track of each step of the process as we go through the notebook.

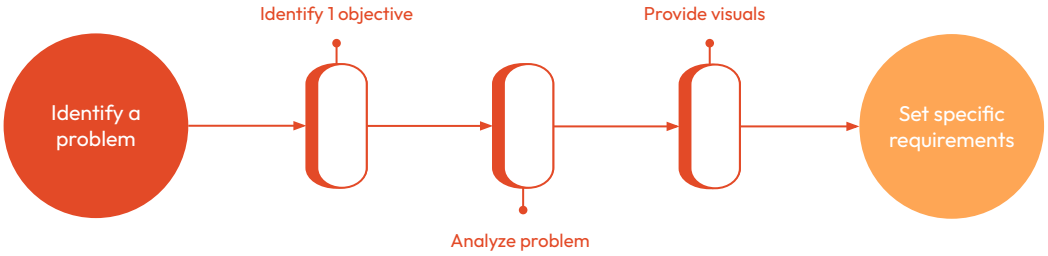
DESIGN PROCESS



Identify (define) a problem

NOTEBOOK RUBRIC: IDENTIFY THE PROBLEM

We first identify 1 specific objective, where we analyze each problem the objective is able to solve. We provide visuals to express the problem as well as an initial analysis of the objective.



Set specific requirements

NOTEBOOK RUBRIC: IDENTIFY THE PROBLEM (extended)

We then set specific goals and requirements that we want our solution to address, with emphasis on game strategy. Constraints are listed as well to better allow us to be aware of rules, materials, and time in order to stay organized.

Research and brainstorm

NOTEBOOK RUBRIC: BRAINSTORM, DIAGRAM OR PROTOTYPE SOLUTIONS

We research domains related to our objective in order to fully understand and build on it. We will make sure to follow the notebook rubric, of providing citations (in form of links or by team numbers) when needed, while we list at least 3 three solutions backed up by research.

502W

AUTHOR; RACINE LIU

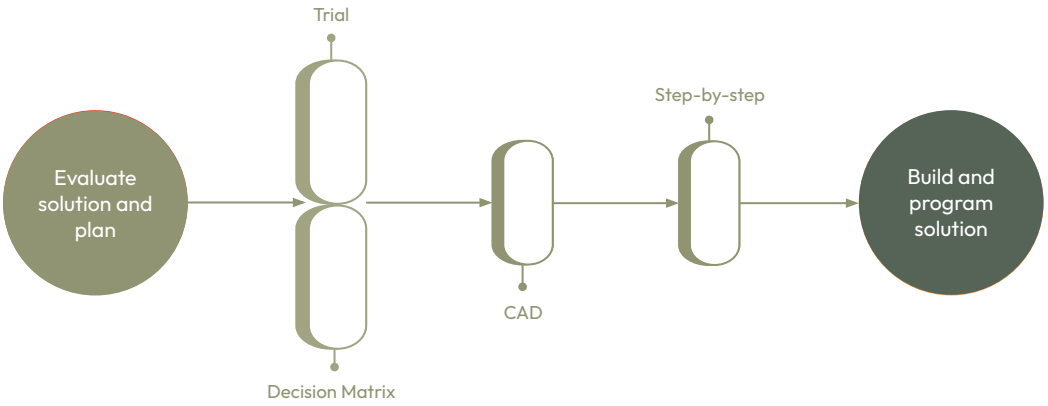
DATE; 10.14.2024

DESIGN PROCESS

Evaluate solution and plan

NOTEBOOK RUBRIC: SELECT BEST SOLUTION AND PLAN

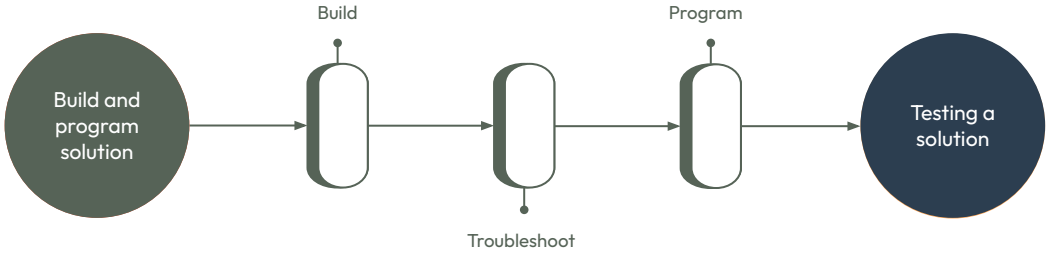
We proceed with a weighted decision matrix to select the best solution, breaking each one down and analyzing them. As well, in addition, reach a verdict through logic and experience. Once a solution has been determined, we create a CAD via Onshape along with a step-by-step guide to better aid us while building. We seek to add enough detail to our plans so that anybody on our team, even those who aren't strong builders, can build the solution. This will allow us maximum efficiency against the time constraints.



Build and program solution

NOTEBOOK RUBRIC: BUILD AND PROGRAM THE SOLUTION

We use the step-by-step CAD as a guide when building. As we build, we document the process in detail with photos, along with any other adjustments we make as we tune. We always strive to improve each mechanism from the initial plan, as we are aware that mechanisms need to be adapted to the real world's physics.



DESIGN PROCESS

Testing solution

NOTEBOOK RUBRIC: TEST SOLUTION

Finally, we test our solutions to see if they follow the initial goals as well as complement set game strategies. The more detailed and thorough each test is, the more likely we will achieve future success with each objective.

GAME ANALYSIS

AUTHOR; RACINE LIU

DATE; 10.14.2024

THE GAME

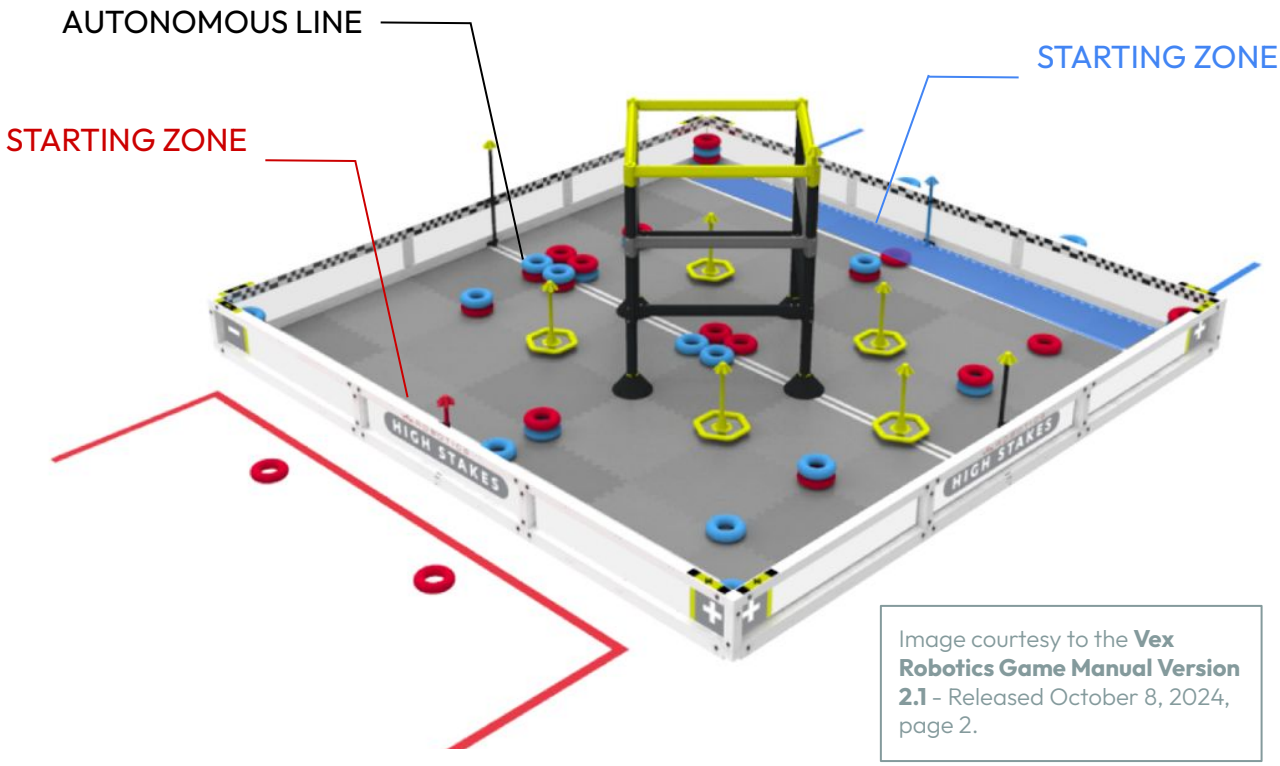


Image courtesy to the **Vex Robotics Game Manual Version 2.1** - Released October 8, 2024, page 2.

High Stakes is played on a 12'x12' square field configured as seen above. Each *Alliance* aims to score colored *Rings* onto *Stakes*, placing *Mobile Goals*, and *Climbing* at the end of the Match. The game is split into two periods, a 15-second autonomous period and a 1 minute 45 second driver control period. Just like the previous season, this game includes an endgame climb as well, where teams score points by ascending the *Ladder* at the center of the field.

This season's game bears a similar resemblance to the game Round Up from 2010-2011. The game uses the exact same game elements. Where instead, the stakes had no barbs up top, making game elements easier to remove. Descoring from mobile goals will prove to be a difficult challenge this year.

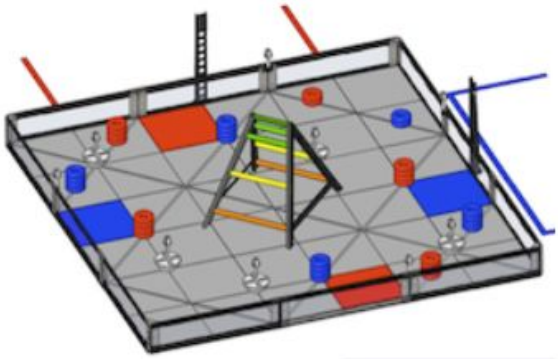
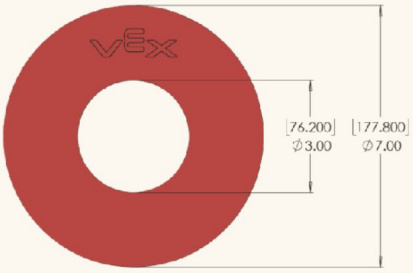


Image courtesy to **Team 254 2010 – 2011 VEX Season** – Round Up - published in 2011 via their website.

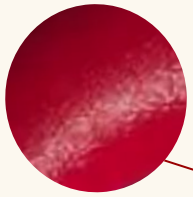
RINGS

Rings, resembling donuts, are the main scoring objects of **High Stakes**. There are **48 rings total** per game split between red and blue. Similar to Triballs from the previous season, each Ring is made of thin, hard plastic and is hollow on the inside, making them light components.

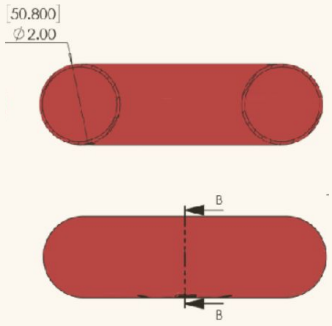
There are **2 colored Rings per alliance** that are to be given before a match. A preload must be placed in a way that it is contacting one Robot of the same Alliance color as well as not contacting or encircling a Stake or any other **Scoring Objects**.



Rings are slightly rough in texture to better help them stay stacked on top of one another.



The more irregularities on the surface create more contact points between the two objects. This provides an added benefit of increased friction in robot to ring interaction making them easier to manipulate.



As per <SG6>
Robots are limited to possessing **2 Rings** at a time. As per the game manual, scored rings on a **Stake** are not included in a **Robot's Possession** count. If by any case, excess rings are unable to be removed they must return to a legal starting position (as described by <SG1>). They will not be eligible to receive points for climb. Any offensive or defensive interactions with game elements will then be included in Match Affecting calculations.

Image courtesy to the **Vex Robotics Game Manual Version 2.1** - Released October 8, 2024, page A-20.

GAME ANALYSIS

ELEMENTS

AUTHOR; RACINE LIU

DATE; 10.14.2024

Image courtesy to the **Vex Robotics Game Manual Version 2.1** - Released October 8, 2024, page A-21.

STAKES

Stakes are vertical PVC pipes with a compliant barb at the top used for scoring rings. There are 4 types of stakes in the game: Mobile goals, Alliance stakes, Neutral stakes, and the high stake.

The barbs on the stakes are made of flexible plastic similar to silicone allowing rings to be scored with minimal extra force but makes for difficult descoring.

Mobile Goals are hexagonal, with a maximal diameter of 10” (254 mm) and an overall height of 14.5” (368.3 mm). There are 5 goals placed on the field for robots to freely move around. A filled mobile goal is composed of 6 rings.

They are heavy due to the weighted plates, making them harder to tip over as well as making it difficult to recover a tilted goal. The weight is both beneficial as well as hindering in other design aspects. More resources will be needed to ensure reliable manipulation of the goal, a potential strategy would be tipping over filled mobile goals to prevent other Alliances from stealing it.

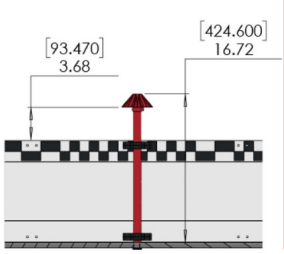


Image courtesy to the **Vex Robotics Game Manual Version 2.1** - Released October 8, 2024, page A-22.

ALLIANCE STAKES

The only colored stakes by alliance, located on the field walls **parallel** to Alliance stations. They can hold 2 rings each. Typically scored during the **autonomous period**.

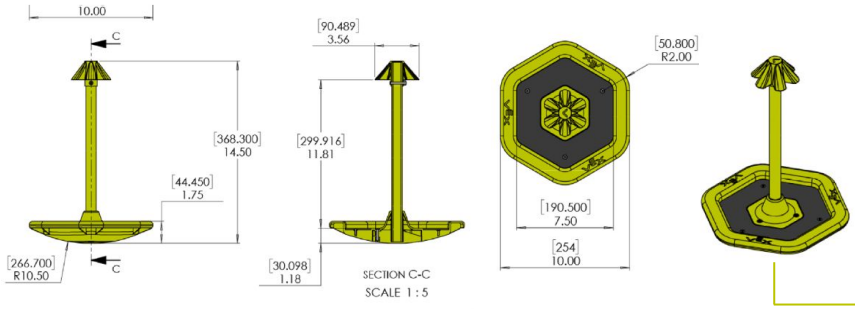


Image courtesy to the **Vex Robotics Game Manual Version 2.1** - Released October 8, 2024, page A-21.

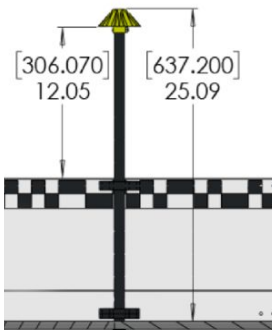


Image courtesy to the **Vex Robotics Game Manual Version 2.1** - Released October 8, 2024, page A-22.

Neutral stakes are the tallest stakes located on the field walls **perpendicular to alliance stations**. They can hold the same amount of rings as a mobile goal when full.

Note that the robot must **expand vertically** when scoring onto the neutral stake due to its height. A predicted issue with scoring on said rings is alignment especially during head to head games where heavy offense and defense is being played.

LADDER

The High Stake only allows for 1 ring to be scored as well as being located on the ladder.

It's predicted that teams will only go after the high stake during late season due to time constraints.

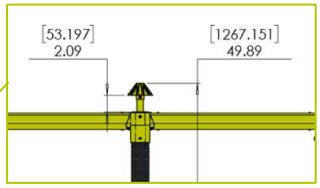


Image courtesy to the Vex Robotics Game Manual Version 2.1 - Released October 8, 2024, page A-23.

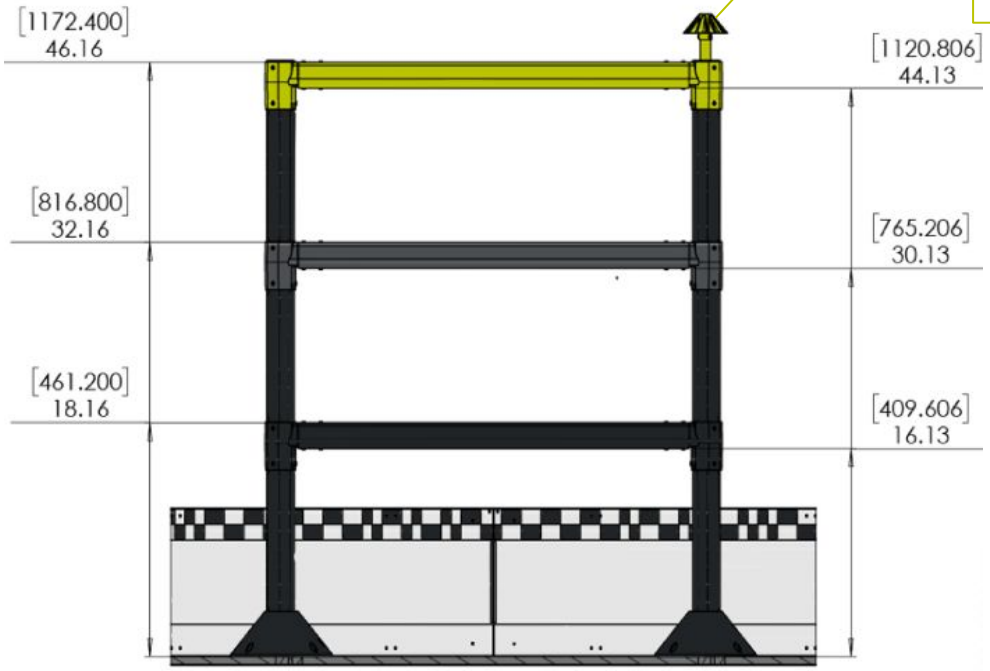


Image courtesy to the **Vex Robotics Game Manual Version 2.1** - Released October 8, 2024, page A-23.

The Ladder is by far the most notable component in the game, **located at the center of the field.** Similarly to the previous season, climbs are needed yet again. The ladder is separated into **3 triangular bars**, unlike the singular round elevation bar from over under. This allows the robot to better grip onto the bar without the risk of it slipping off.

The ladder itself covers **4 tiles** in the centre of the field which is around 1/9 of the entire field. Robots should aim to go under the ladder(staying under 16 inches) as it is a significant amount of space for maneuvering during the fast paced games.

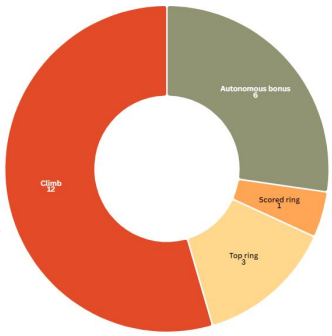
GAME ANALYSIS

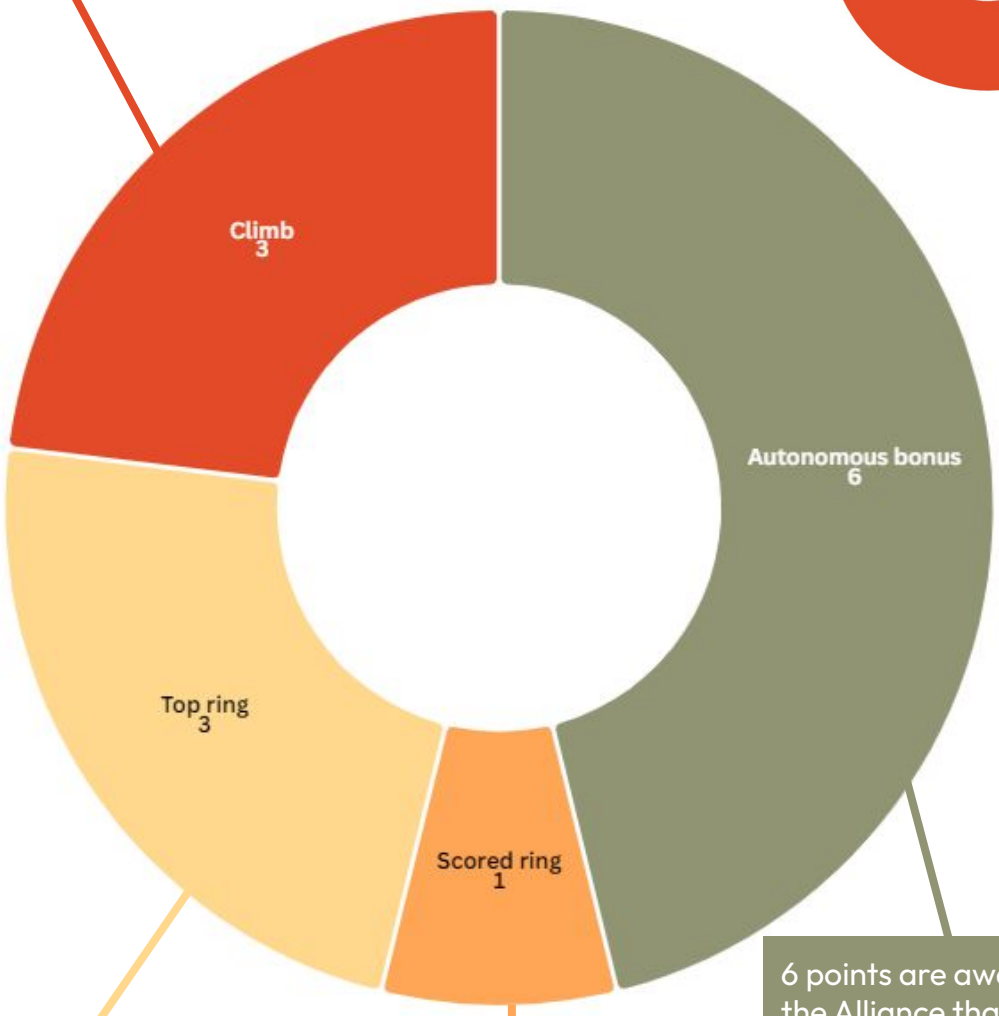
SCORING

AUTHOR; RACINE LIU

DATE; 10.14.2024

Alliances can be rewarded 3-12 points for an elevation. This makes it so that a climb may be a final contributing factor to winning or losing a match.

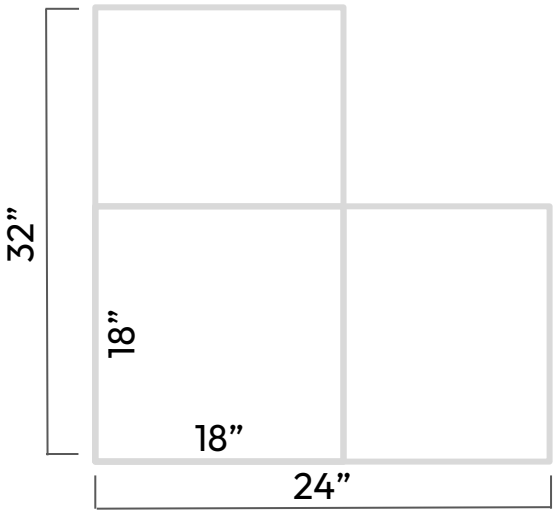




As per <SC4> the top ring is a Ring that is the highest Scored Ring on a Stake. It must be properly Scored, and if only one Ring is on the Stake, it counts as the Top Ring.

A ring is considered scored when fully under the barb as per <SC3>. Teams are awarded a single point for each rings that fit the criteria

6 points are awarded to the Alliance that scores the most points during the 15-second autonomous period. The Autonomous Bonus will allow teams to gain a head start in matches, which may contribute to the end result.



EXPANSION

This season, there are more extensive rules that apply to expansion in terms of direction.

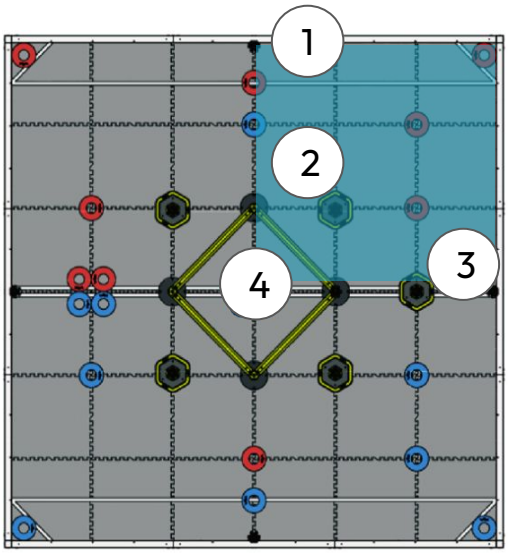
Horizontal expansion is limited. As per <SG 2>. Once the match begins, robots are only able to expand horizontally to 24” in one direction.

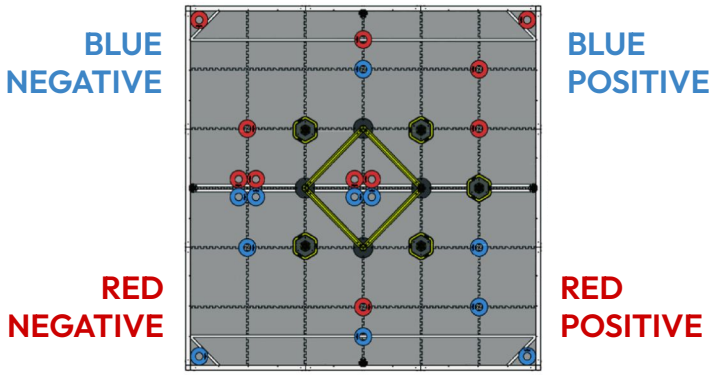
Per <SG4>, robots may expand vertically but they cannot expand over 2 planes of the ladder at anytime. This means that from the ground they height limit is 32”

AUTONOMOUS

Each game commences with a 15-second autonomous period. Each alliance must be contacting the starting lines located right next to the alliance station expanding horizontally along the sides.

Note that the rings are configured differently by color on each individual corner of the field hence 4 different autonomous programs are needed unlike the 2 required for the previous season. We will be addressing each autonomous route by their alliance color as well as which corner is in that section of the field as labeled below





AWP(Autonomous Win Point) is awarded to Alliances when they complete a set of specific tasks. As per <SC8> teams must score as least 4 rings on an alliance stake or mobile goal, rings scored on a minimum of 3 stakes on an alliances side, at least 1 ring on an alliance wall stake, not contacting the starting line when autonomous period ends and at least 1 robot contacting the ladder.

CORNERS

Mobile goals placed in different corners have significant effects on scoring. If a mobile goal is placed in a **Positive Corner**, all rings on that goal receive a score boost. Regular rings are worth **two points** each, while rings placed on the top of the goal (Top Rings) are worth **six points**. This doubling effect makes Positive Corners highly valuable for teams aiming to maximize their score from ring placements.

The Positive Corner presents a clear opportunity to **maximize ring scores**, making it a priority zone for alliances during gameplay. Teams should design robots capable of quickly delivering mobile goals to these zones to take advantage of the score-doubling effect. Additionally, a strategy could involve placing Top Rings on mobile goals positioned in the Positive Corner, as these rings offer the highest potential return.

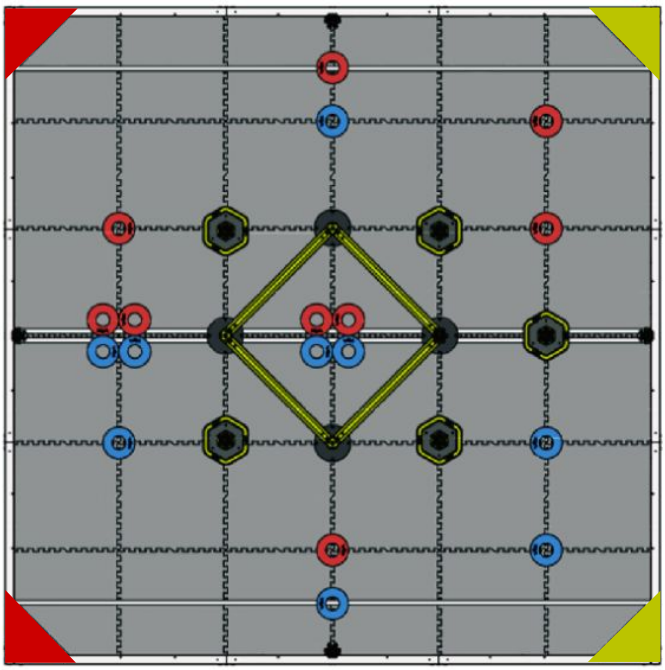


Image courtesy to the **Vex Robotics Game Manual Version 2.1** - Released October 8, 2024, page A-7.

As per <SG11> positive corners are protected in the last 15 seconds of each match.

As per SC5, a Mobile Goal is considered **Placed in a Corner** if it meets specific conditions: its base must contact the floor or a white tape line, some part of the flexible top of the Mobile Goal's Stake must be higher than the field's top edge, and its base must break the plane of the Corner.

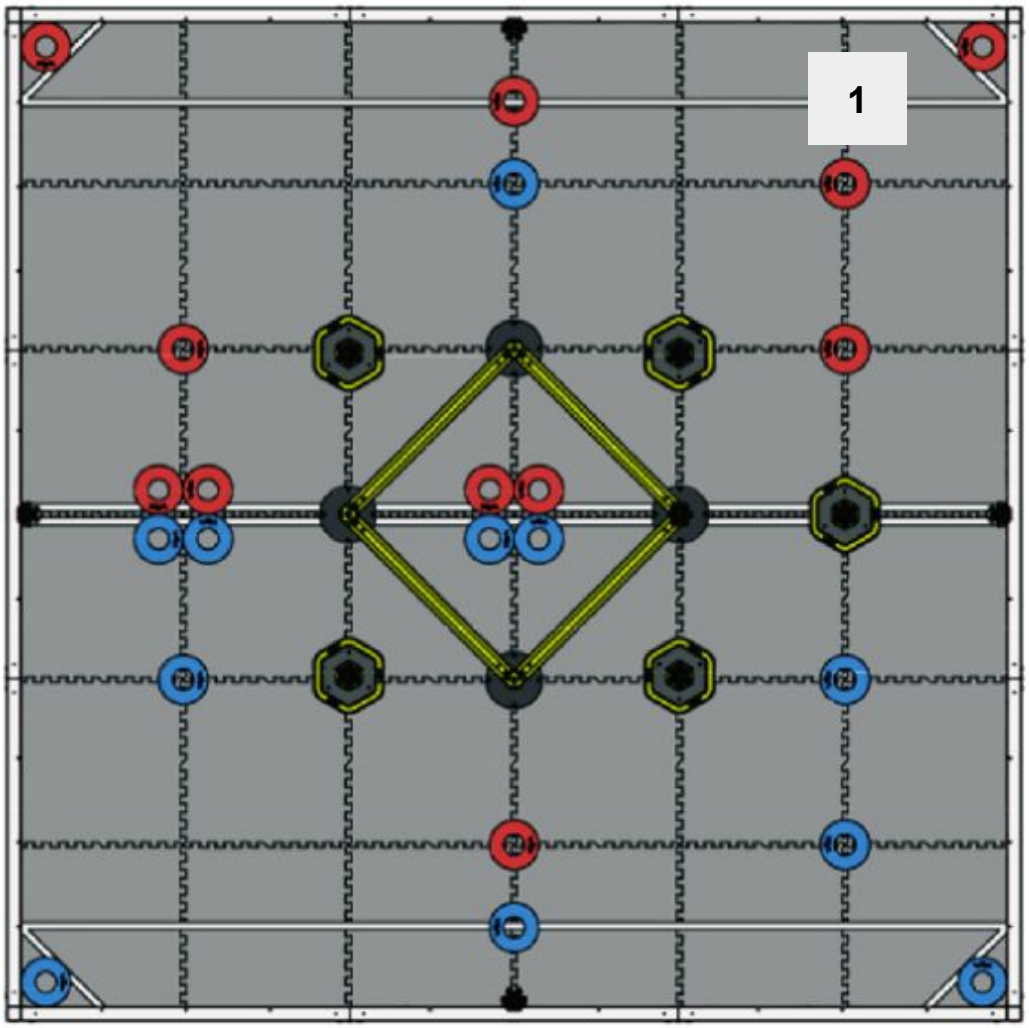
Only one Mobile Goal can be Placed in each Corner. If multiple Mobile Goals meet these criteria, tiebreakers such as how far the base extends into the Corner, the verticality of the Stake, and how far the flexible top extends will be used to determine which goal is Placed. If still tied after these tiebreakers, no goal will be considered Placed in that Corner.

On the other hand, placing a mobile goal in a **Negative Corner** has a much different impact. All rings on that mobile goal will be worth **zero points**. Furthermore, an equivalent number of points will be **subtracted** from the alliance's total ring score. For each regular ring, **one point** is deducted, and for each Top Ring, **three points** are removed from the alliance's total. It's important to note that this penalty only affects ring points; points from climbing or the Autonomous Bonus remain untouched

GAME ANALYSIS

GAME STRATEGY

Unlike the previous season of Over Under where multiple strategies can be applied, its obvious what to aim for this season: the positive corners. The general aim for all teams would be to fill up the mobile goals as fast as possible, then to place them into the positive corners. Opponent teams would then try to nullify the points by stealing the already filled mobile goals either descoring them or placing them into the negative corners. Top rings being 2 points higher than normal rings means that they should be prioritized.



It is essential to ensure at least 1 positive corner with 1 alliance actively filling up mobile goals while the other guards the corner, taking low risk cycles to score wall stakes or defense

Image courtesy to the **Vex Robotics Game Manual Version 2.1** - Released October 8, 2024, page A-7.

If an Alliance doesn't manage to secure any positive corner, then they should still prioritize filling up their own mobile goals in order to take the opportunity to remove the opposing Alliance's mobile go and replacing it with their own in the corner.

Top rings should be prioritized at all times and there should be a goal to have the fastest climb time in order to take advantage of the protected positive corners, either pushing opponent filled goals into the negative corner or removing their respective mobile goal from the corner.

TEAM MEETING

#1

AUTHOR; RACINE LIU

DATE; 10.14.2024

ABOUT TEAM MEETINGS

Team meetings are crucial in fostering effective communication and collaboration for the success of a team, they serve to always keep us on track. Major team meetings will occur after every competition in order to better prepare us for the next, as well as allowing for group reflection. We will also be scheduling weekly meetings to check in with our plans. All team members are expected to be present for the meetings.

AGENDA

- Initial setup of programs and cad
- Develop a team management system as well as assigning roles
- Long term and short term goal setting
- Deciding first tournament

ATTENDANCE

Full team: Vaughan Sandquist, Racine Liu, Aaron Lew, Aiden Tam

EVENT DATES

Tournament	Date	Level
WPRA Season Opener Qualifier	Sep 30	Blended
WPRA Halloween Qualifier	Oct 19	Blended
PYRS Seaquam Season Opener	Nov 2	Blended
WPRA Fall Qualifier	Nov 10	Blended
PYRS Salish Fall Qualifier	Nov 16	Blended
PYRS Heritage Woods	Dec 7	Blended
Gord Trousdell Ten Ton Robotics	Dec 15	HS
WPRA New Year Qualifier	Jan 4	Blended
PYRS Southridge Winter Qualifier	Jan 18	Blended
PYRS Surrey Christian Last Chance	Feb 8	Blended
Ten Ton Robotics Blended Last Chance	Feb 15	Blended
WPRA Last Chance Qualifier	Feb 17	Blended
PYRS BC Mainland Regionals	Mar 1	HS

To the left, is a list of future competitions that we are eligible for. We will be trying our best to attend all of them, as competitions are the fastest way to practice and refine our skills. Competing against a variety of teams exposes us to different strategies and approaches, allowing the opportunity to learn from others and improve our own skills. Therefore, our first competition will be the PYRS qualifier on November 2nd. This leaves us around 3 weeks to create a robot.

TEAM MEETING CONT

SEASON TIMELINE

TENTATIVE TIMELINE OF SEASON

OCTOBER 15, 2024
Robot 1.0

DECEMBER 22, 2023 - JANUARY 7, 2024
Winter Break

JANUARY 8, 2024
Robot 2.0

MARCH 1, 2024
Provincials

MARCH 4, 2024
Robot 3.0

MAY, 2024
Worlds

TEAM MEETING

GOALS

AUTHOR; RACINE LIU

DATE; 10.14.2024

GOAL SETTING

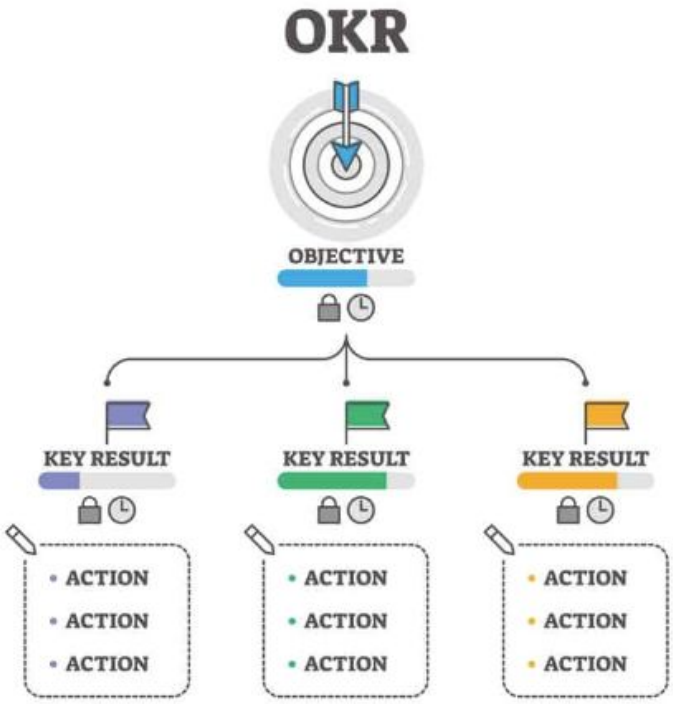
The previous season, we used smart goals to plan our schedules leading up to competitions. Although we met most of our deadlines, we found that smart goals for long term plans did not fit the most. Hence, we plan on merging smart goals with OKR style goals.

OKR GOALS

OKRs have two important parts: **The objective you want to achieve and the key results**, which are the way you measure achieving the objective.

Objectives: are memorable, qualitative descriptions of what you want to achieve. Objectives should be short, inspirational, and engaging. An objective should motivate and challenge the team.

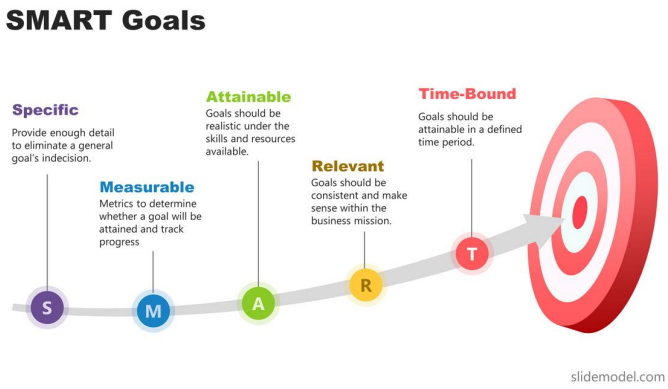
Key results: a set of metrics that measure your progress towards the objective. For each objective, you should have a set of two to five key results. More than that and no one will remember them.



There are a couple of key points in this definition. First, the objective should be concise and engaging, so a team can easily remember it. Next, there should be a small number of metrics to track the key results. These metrics should be something you can measure on a timely basis.

INTEGRATION

With OKR goals we are able to set key results for the design, build and code process instead of one goal encapsulating them all. SMART goals, on the other hand, are **Specific, Measurable, Achievable, Relevant, and Time-bound** objectives that ensure progress is well-defined and attainable. Combining the two by using smart goals to measure key results allows use to ensure that we can follow through with deadlines having a clear sense of progress



Objective: Finish Robot 1.0 by Nov 1

Key result 1: Design completion by Oct 18

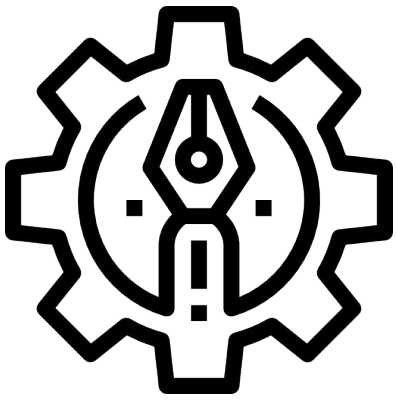
S Complete a functional 3D CAD model of the robot, including all critical components (drive train, intake, clamp and sensors).

M The design is finished when all major components are modeled and placed within the CAD.

A Given the team's current knowledge of CAD software and past experience, this can be completed within the given timeframe.

R A solid design is the foundation for a smooth build process and minimizes troubleshooting later on.

T The CAD model must be completed, reviewed, and approved by the team by October 4.



TEAM MEETING

GOALS

AUTHOR; RACINE LIU

DATE; 10.14.2024

Key result 2: Build completion by Oct 25

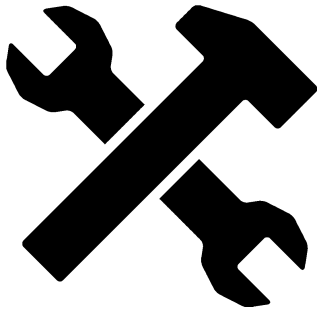
S — Assemble the entire robot, ensuring all mechanical components (wheels, motors, and manipulator) are mounted and functional.

M — The build is considered complete when the robot's drive train is functional, and basic mechanical movements are tested with no major issues.

A — With a clear design and access to necessary parts, the build can be completed on time if tasks are divided efficiently among the builders.

R — Without a complete build, the robot can't be programmed or tested, so the build is essential for the robot's success.

T — The entire robot assembly must be completed by October 11, allowing time for testing and programming.



Key result 3: Finalize programming by Nov 1

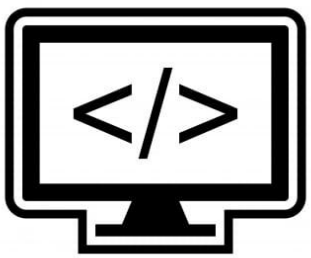
S — Program both manual control and autonomous routines to execute key competition tasks (movement, object manipulation, etc.).

M — The programming phase will be complete when the robot successfully navigates the autonomous challenge during practice and responds to all controller inputs in manual mode.

A — The team has the coding knowledge and past experience with Vex robotics to meet this goal within the timeframe.

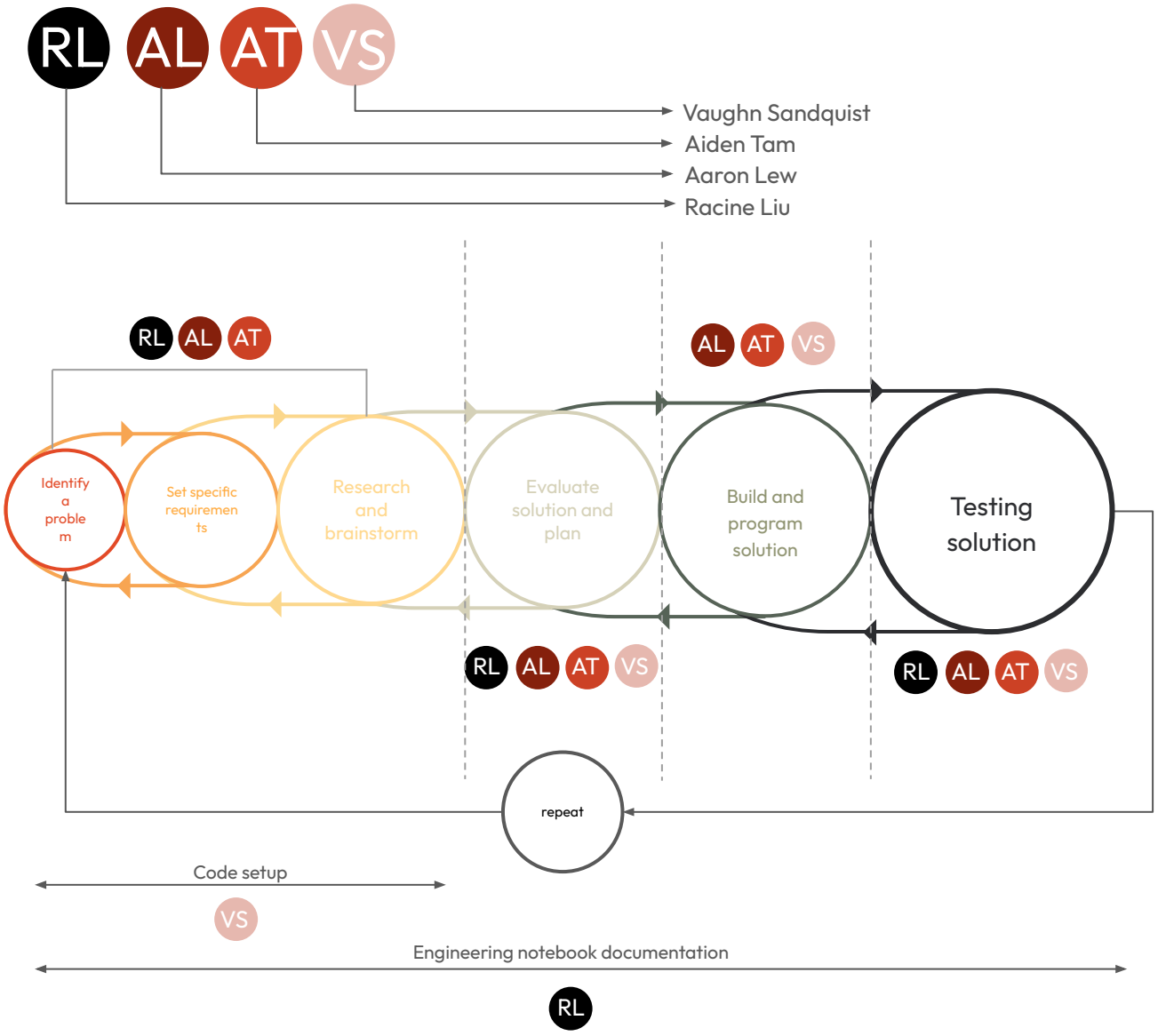
R — A well-programmed robot is crucial for both autonomous and driver-controlled segments of the competition.

T — Programming must be completed and fully tested by October 18.



TEAM MEETING

ROLES



Each robot subsystem has a design process with team roles divided as shown on the image to the left.

Aaron and Aiden: Work through all the steps of the design process.

Racine: Aid in decision matrix and document all steps taken, along with research and tests.

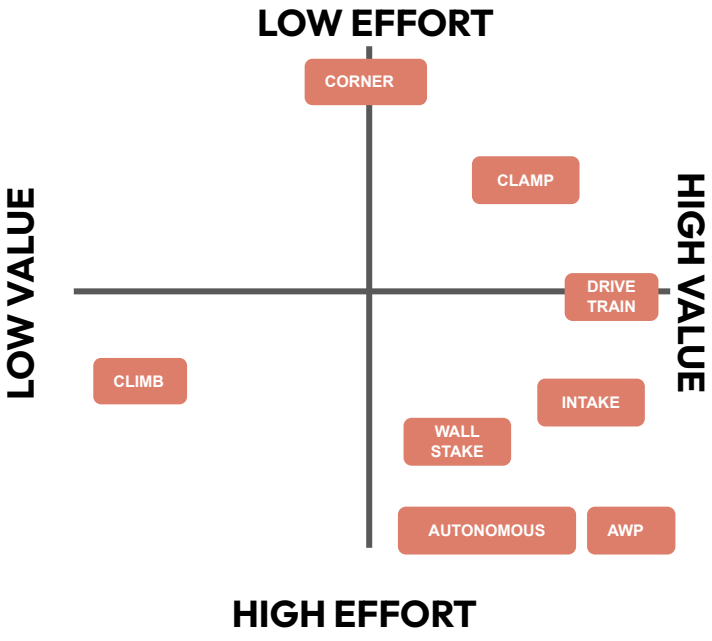
Vaughn: Creates code for all set solutions as well as autonomous programs

TEAM MEETING

AUTHOR; RACINE LIU

DATE; 10.14.2024

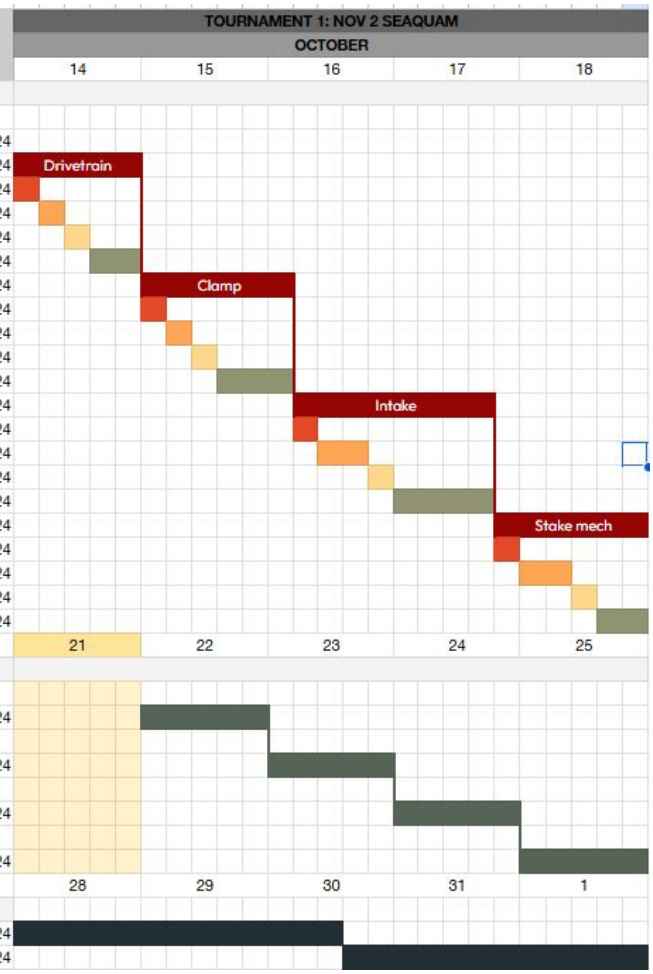
SCHEDULE



We've decided on pursuing the most important mechanisms for the upcoming competition due to constraints. We'll aim to have a:

- Wall stake mechanism
- Intake
- Drivetrain
- Clamp
- Autonomous programs

	TITLE	ASSIGNED TO	START DATE	END DATE
1	**Design**			
	Programming			
	Initial setup	Vaughn	10/14/2024	10/18/2024
	Drivetrain 1.0		10/14/2024	10/14/2024
	Identify a problem	Racine, Aaron, Aiden	10/14/2024	10/14/2024
	Set specific requirements	Racine, Aaron, Aiden	10/14/2024	10/14/2024
	Research and brainstorm	Racine, Aaron, Aiden	10/14/2024	10/14/2024
	Evaluate solution and plan	ALL	10/14/2024	10/14/2024
	Clamp 1.0		10/15/2024	10/16/2024
	Identify a problem	Racine, Aaron, Aiden	10/15/2024	10/15/2024
	Set specific requirements	Racine, Aaron, Aiden	10/15/2024	10/15/2024
	Research and brainstorm	Racine, Aaron, Aiden	10/15/2024	10/15/2024
	Evaluate solution and plan	ALL	10/15/2024	10/16/2024
	Intake 1.0		10/16/2024	10/17/2024
	Identify a problem	Racine, Aaron, Aiden	10/16/2024	10/16/2024
	Set specific requirements	Racine, Aaron, Aiden	10/16/2024	10/16/2024
	Research and brainstorm	Racine, Aaron, Aiden	10/16/2024	10/17/2024
	Evaluate solution and plan	ALL	10/17/2024	10/17/2024
	Wall stake mechanism 1.0		10/17/2024	10/18/2024
	Identify a problem	Racine, Aaron, Aiden	10/17/2024	10/18/2024
	Set specific requirements	Racine, Aaron, Aiden	10/18/2024	10/18/2024
	Research and brainstorm	Racine, Aaron, Aiden	10/18/2024	10/18/2024
	Evaluate solution and plan	ALL	10/18/2024	10/18/2024
2	**Build**			
	Drivetrain 1.0			
	Build and program solution	ALL	10/22/2024	10/22/2024
	Clamp 1.0			
	Build and program solution	ALL	10/23/2024	10/23/2024
	Intake 1.0			
	Build and program solution	ALL	10/24/2024	10/24/2024
	Wall stake mechanism 1.0			
	Build and program solution	ALL	10/25/2024	10/25/2024
3	**Programming**			
	Robot main program + Test	Aaron, Aiden, Vaughn	10/28/2024	10/30/2024
	Game autos	Aaron, Aiden, Vaughn	10/30/2024	10/1/2024



ROBOT 1.0

DRIVETRAIN 1.0

AUTHOR; RACINE LIU

DATE; 10.14.2024

IDENTIFY A PROBLEM

IMPORTANCE

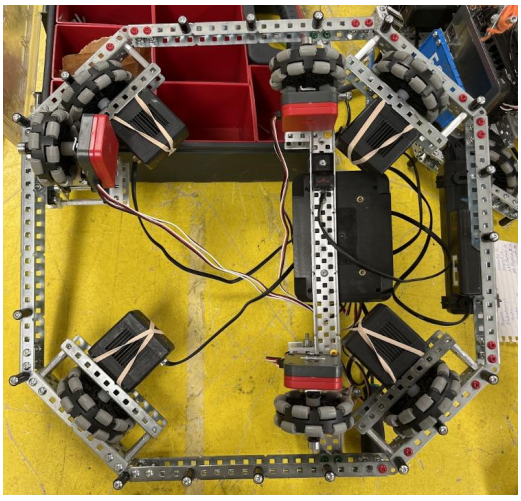
Drivetrains are assemblies, generally made of wheels, used to traverse the VRC field. A drivetrain is undeniably the most important subsystem of the robot as it allows for maneuvering around the field, maximizing a robot's competitive edge, especially in games that focus on offense and defense like Over Under. A drivetrain is a foundation for all other subsystems to be built on, hence why it is key for success.

CONSIDERATIONS

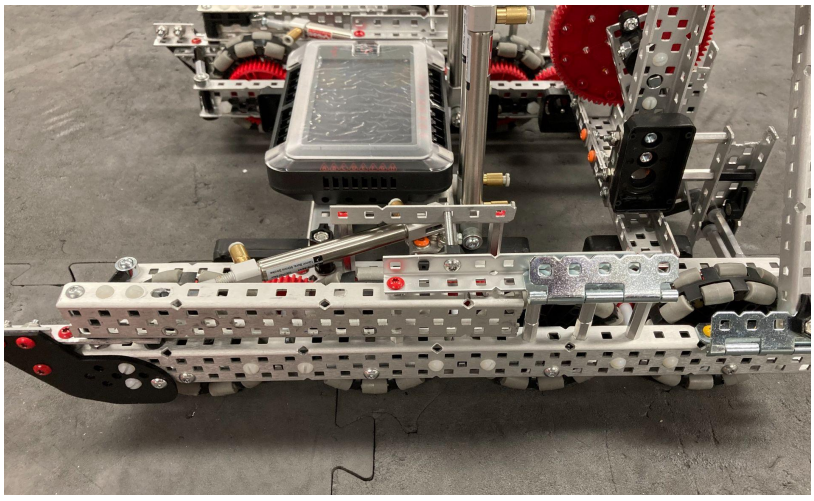
- Are there obstacles on the playing field which need to be driven over or climbed up on?
- How much defense will the drivetrain be exposed to?
- How much of an advantage is there for the drivetrain to be omni-directional?
- Is the drivetrain going to be pushing multiple/heavy game pieces, or does it need to be fast?
- How many motors are going to be needed for functions other than the drivetrain?

PREVIOUS EXPERIENCE

In the previous season of Over Under, we stuck to one type of drivetrain the entire comp as well as running the same gear ratio 3 to 4 at 450 rpm. We alternated wheel sizes and types to better suit our needs. In the season of Spin up, we had used 2 different types of drivetrains. We started out the season with an X-drive then switched to a tank drive for Worlds in favor of speed and pushing power.



Our provincials drivetrain for Spin up was an X-drive, it allowed us to strafe and gain an edge on maneuverability around the field, though it did not allow for greater pushing power.



Our provincials drivetrain for Over Under was a Tank drive. We focused on speed, a 4 to 3 gear ratio. As well as enough torque to push robots around when rigorously scoring or defending.

AUTHOR; RACINE LIU

DATE; 10.14.2024

DRIVETRAIN 1.0

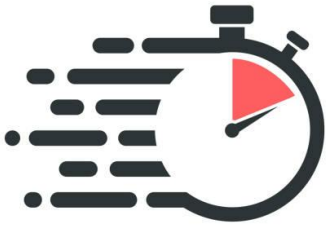
CONSTRAINTS

SETTING REQUIREMENTS

Speed

"Is the drivetrain going to be pushing multiple/heavy game pieces, or does it need to be fast?"

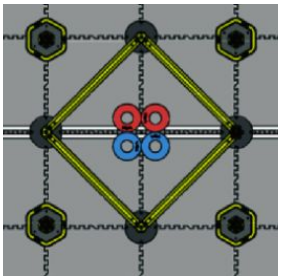
Speed will be to our advantage as we will be able to quickly go across the field traveling from corner to corner within minimal time as well as beating the other robots to our decided destination. With Speed on our side, we will be able to maximize Top stakes owned scored in both skills and during games. Higher speed ties into our second goal of maneuverability.



Maneuverability

"Are there obstacles on the playing field which need to be driven over or climbed up on?"

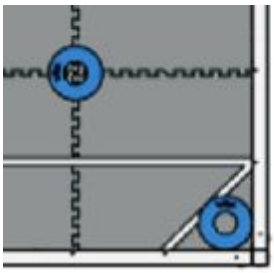
One of the game's most notable elements is the ladder. Being able to smoothly traverse under the ladder will up our advantage when playing offense or defense, as then we will be able to cross the field at any given moment and place. With a wider path of movement, more strategies will be open to implementation.



Versatility

"How much defense will the drivetrain be exposed to?"

We predict that High Stakes similar to Over Under, will be a very contact heavy game. Robots will most definitely focus on protecting corners and blocking neutral stakes from being scored. In order to be able to score points and even defend, we must have a balance of speed and torque



CONSTRAINTS

RULE CONSTRAINTS:

As per <R4> the drivetrain must fit within an 18" x 18" x 18" volume

MATERIAL CONSTRAINTS:

We have full access to all potential materials as it is the start of the season

TIME CONSTRAINTS:

This subsystem should be completed by October 22nd

DRIVETRAIN 1.0

AUTHOR; RACINE LIU

DATE; 10.14.2024

IDEATE

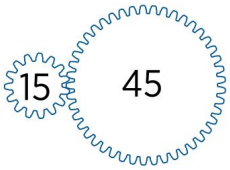
SPEED

Controlling the speed of a VEX Robotics system involves a nuanced approach, leveraging various techniques to ensure precise and effective movement. The primary method is voltage control, where the power supplied to the motors determines their speed. By utilizing motor commands in programming, one can set power levels to achieve the desired velocity. Additionally, considerations of gear ratios further contribute to achieving optimal speed control, ensuring both accuracy and efficiency in movement

CALCULATIONS

Gear ratios

Gears play a crucial role in a drivetrain by transmitting power from the motor to the wheels. By using different gear ratios, you can trade off speed for torque and vice versa. A higher gear ratio results in higher speed but lower torque, while a lower gear ratio provides more torque at the expense of speed.



Gear ratios can be calculated with this formula:

$$\text{gear ratio} = \frac{\text{rotations of } \textit{driver} \text{ gear}}{\text{rotations of } \textit{driven} \text{ gear}} = \frac{\text{teeth of } \textit{driven} \text{ gear}}{\text{teeth of } \textit{driver} \text{ gear}}$$

$$\text{For this gear pair, the gear ratio} = \frac{\text{rotations of gear A}}{\text{rotations of gear B}} = \frac{3}{1}$$

RPM

RPM stands for revolutions per minute (or revs per minute) and is a measure of how fast the engine is spinning.

RPM can be calculated by the formula below:

$$\text{RPM} = \text{cartridge RPM} \times \text{gear ratio}$$

Linear speed

The linear speed of a wheel is also influenced by its diameter. A larger wheel diameter will result in higher linear speed. Conversely, a smaller wheel diameter will yield lower linear speed.

The formula for linear speed is given by:

$$\text{Linear Speed} = \pi \times \text{wheel diameter} \times \text{RPM}$$

SPEED AND POWER

Linear distance is given by
Linear distance = wheel radius × angular velocity × time

$$\text{Linear distance} = r \times \omega \times t$$

Torque is given by
Torque = force x wheel radius

$$\tau = f \times r$$

Thus, we can derive that

$$f = \tau / r$$

Power = Force × Linear distance / Time

$$P = (\tau / r) \times r \times \omega \times t / t$$

$$P = \tau \times \omega$$

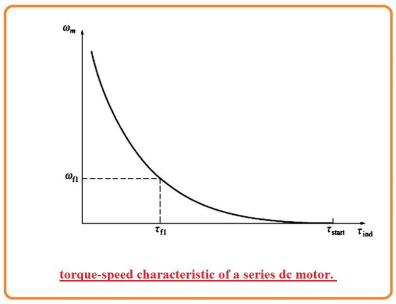
$$\tau = P / \omega$$

Torque = Power / Angular Velocity

Thus, Torque is inversely proportional to the angular velocity.

In other words, this inverse relationship arises from the fundamental physics of rotational motion. In a rotating system, the power (P) is given by the product of the torque (T) and the angular velocity (ω), i.e., P = T x ω. The angular velocity is the reciprocal of the period of rotation (T), so the formula becomes P = T x (2π/T) = 2πT^2.

If the power is held constant, then as the period of rotation (and hence the speed) decreases, the torque must increase to keep the power constant. Conversely, if the speed increases, the torque decreases to keep the power constant. Hence, speed and torque are inversely proportional.



torque-speed characteristic of a series dc motor.

The figure on the left graphically shows the relationship between speed (y-axis) and power (x-axis). We can view that it is a reciprocal graph
(Inversely proportional). As the speed/angular velocity decreases, torque increases.

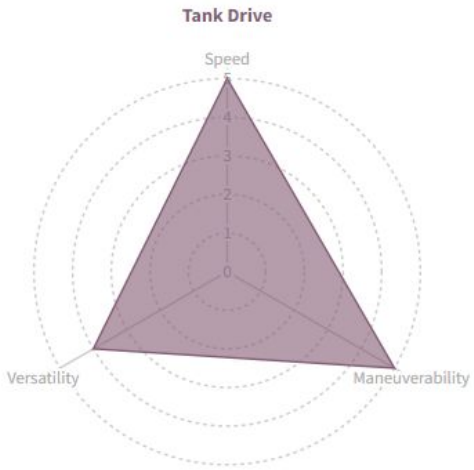
$$\tau \propto 1/\omega$$

AUTHOR; RACINE LIU

DATE; 10.14.2024

DRIVETRAIN 1.0

TANK DRIVE



STATS

Speed

With the customizability, we are able to achieve a very fast drivetrain

Maneuverability

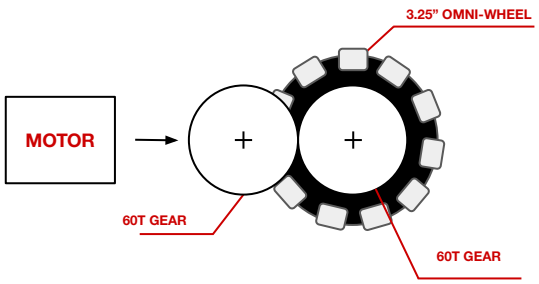
Tank Drives can easily traverse across the field and through being able to quickly navigate the field

Versatility

Very good pushing power, but has a disadvantage when being pushed from the side. Unable to strafe.

SOLUTIONS

Solution 1

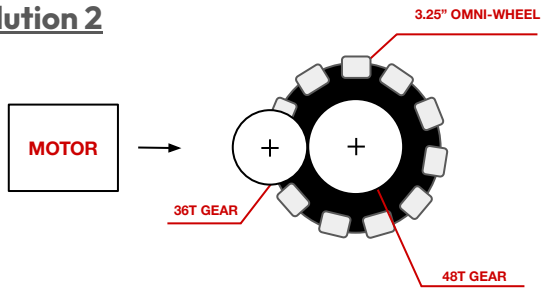


Gear ratio: **1:1**

This gearing means that we get the direct motor power, making it a very fast drivetrain with equal speed and torque.

Engine RPM	600 rpm
Drivetrain transmission ratio	60/60
Wheel RPM	600 rpm
Tire diameter	3.25 in
Vehicle speed	8.5 ft/s

Solution 2

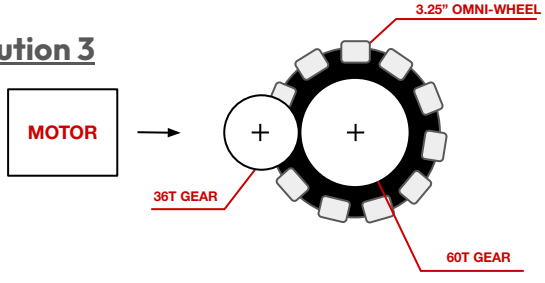


Gear ratio: **4:3**

Output torque is greater than input torque and reduces max speed. We can carry load as well as being capable of pushing more at fast speeds.

Engine RPM	600 rpm
Drivetrain transmission ratio	48/36
Wheel RPM	450 rpm
Tire diameter	2.75 in
Vehicle speed	5.4 ft/s

Solution 3



Gear ratio: **5:3**

High torque, lower speed, and great pushing power. We have experience with this gearing already as we used it for worlds.

Engine RPM	600 rpm
Drivetrain transmission ratio	60/36
Wheel RPM	360 rpm
Tire diameter	3.25 in
Vehicle speed	5.1 ft/s

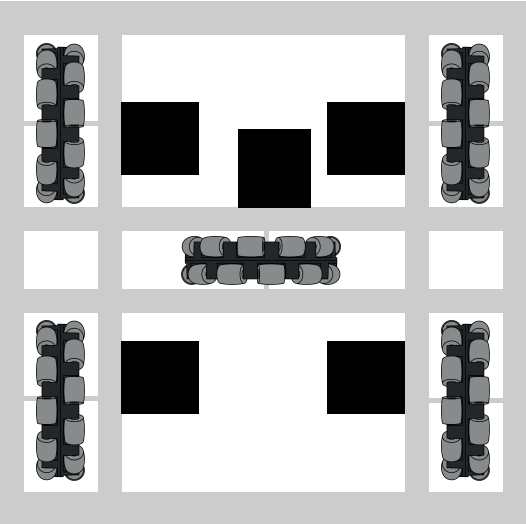
DRIVETRAIN 1.0

AUTHOR; RACINE LIU

DATE; 10.14.2024

H-DRIVE

FEATURES



The H-Drive, similar to the Tank Drive, characterized by its 'H' shape configuration, typically incorporates a combination of omni-directional wheels. The sides of the drivetrain facilitate forward and backward movement, while the omni-directional wheel positioned perpendicular to the drive direction enables smooth strafing or lateral motion.

The central wheel's ability to rotate independently allows the robot to change orientation without altering its position, enhancing overall agility. It uses 2-5 motors (2 or 4 for the drive and 1 for the controlling wheel).

STRENGTHS

With an H-Drive, there is an advantage in scoring objects that require precision as well as moving into defensive positions. Similar to a Tank Drive, they are easier to build as well as customizable. Gearing for torque or speed is simple as well.

The ability to strafe allows another way for movement, helping with positioning and overall maneuverability around the field. The H-Drive's ability to rotate in place facilitates efficient alignment during game scenarios, providing teams with a strategic edge in both offensive and defensive maneuvers.

SHORTCOMINGS

Having an extra wheel and motor in the middle takes up a large amount of space and creates more design constraints. As well as this, using up an extra motor for it, taking away a motor from other possible design features.

Using a single motor for strafing means it's under a lot of strain from having to push the weight of the robot by itself even though all the other wheels are omni-wheels. This means it can't strafe with very much torque, making it quite weak and slow while combating the amount of friction it must push against.

Difficulty may arise when attempting to go over game elements, there is a chance that the perpendicularly placed omni-wheel may get caught on pieces, preventing a robot from crossing.

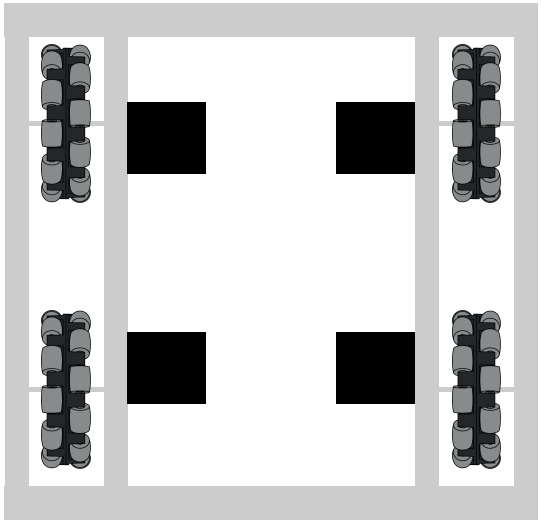
DRIVETRAIN 1.0

TANK DRIVE

AUTHOR; RACINE LIU

DATE; 10.14.2024

FEATURES



Tank drives are a very popular type of drivetrain, as well as the simplest. These drives consist of two separately controlled sides of parallel wheels, which allows for precise control of the robot. This allows for greater maneuverability and control over the robot's movement, which can be important in competitions where speed and agility are key factors.

Additionally, tank drives are relatively easy to build and can be modified and adjusted to suit a variety of different competition scenarios.

STRENGTHS

Tank Drives are easily customizable to fit one's requirements. The amount of wheels can be adjusted, ranging from 4 to 8 with either omni-wheels or traction wheels in various sizes. It's favorable for motor allocation as well, where a minimum of 4 motors to a maximum of all 8 motors can be used. Gearing adjustments allow for increased agility and speed, or for strength and pushing power. Hence, various combinations allow for a wide range of options.

In performance, this type of drivetrain is very simple to build and consistently performs well. Many former world champions have used the Tank Drive for its simplicity. Again reiterating its adjustability, a Tank Drive is a very strong option for offense where it's able to have an advantage in greater pushing power.

SHORTCOMINGS

Tank Drives suffer from the inability to strafe, which may impact versatility around the field. Strafing refers to a driving technique that allows a vehicle to move laterally without changing its direction of travel. In the context of VEX Robotics, strafing enables a robot to move sideways or sidestep obstacles without changing its orientation.

This type of Drivetrain is weak to getting pushed from the side, especially with all omni -wheels. They have reduced traction when forces are applied laterally. The wheels will not provide as much grip in the lateral direction, making it easier for an opponent to push the robot sideways.

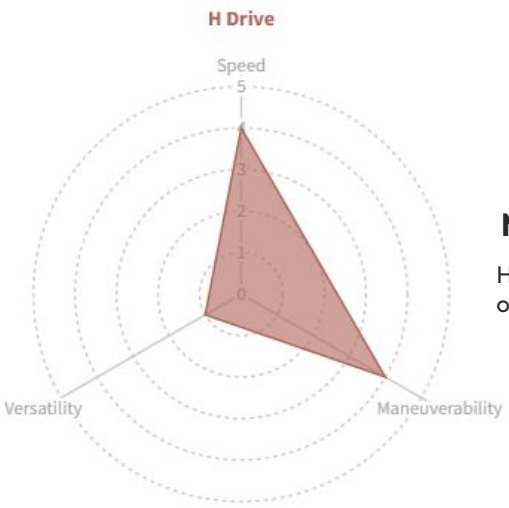
AUTHOR; RACINE LIU

DATE; 10.14.2024

DRIVETRAIN 1.0

H-DRIVE

STATS



Speed

An H-Drive has the potential to be very fast. The center omni-wheel drags on the ground, lowering speed

Maneuverability

H drives can navigate as well as a tank drive. May be a hinder if caught on a ring

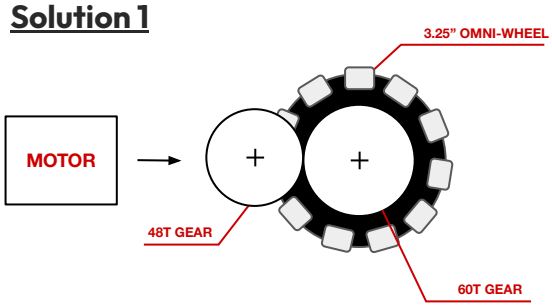
Versatility

It has a disadvantage when being pushed from the side as well as putting strain onto the motor to the central omni wheel. Risk of motor damage.

SOLUTIONS

We can keep the 3 solutions previously given for the tank drive for the sides of the drivetrain. Below are solutions for the central perpendicularly lined wheel.

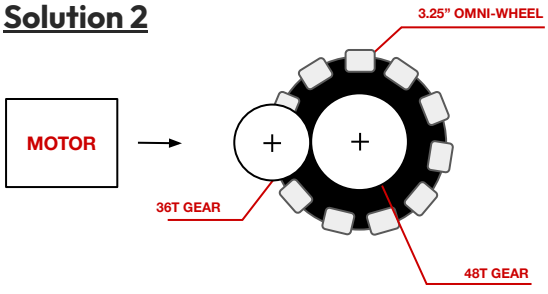
Solution 1



Gear ratio: **5:4**
This gearing means that Speed is traded off for greater pushing power, which is needed as we have only 1 wheel controlling movement in the X direction.

Engine RPM	600 rpm
Drivetrain transmission ratio	60/48
Wheel RPM	160 rpm
Tire diameter	3.25 in
Vehicle speed	2.27 ft/s

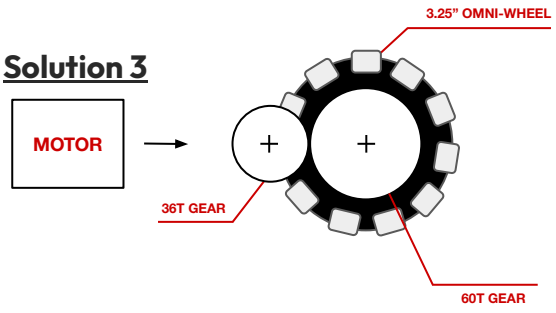
Solution 2



Gear ratio: **4:3**
Output torque is greater than input torque and reduces max speed. We can carry load as well as being able to travel at minimal speed. Less strain on the wheel.

Engine RPM	600 rpm
Drivetrain transmission ratio	48/36
Wheel RPM	150 rpm
Tire diameter	3.25 in
Vehicle speed	6.38 ft/s

Solution 3



Gear ratio: **5:3**
High torque, lower speed, and great pushing power. Fast speed with some torque for center wheel.

Engine RPM	600 rpm
Drivetrain transmission ratio	60/36
Wheel RPM	360 rpm
Tire diameter	3.25 in
Vehicle speed	5.1 ft/s

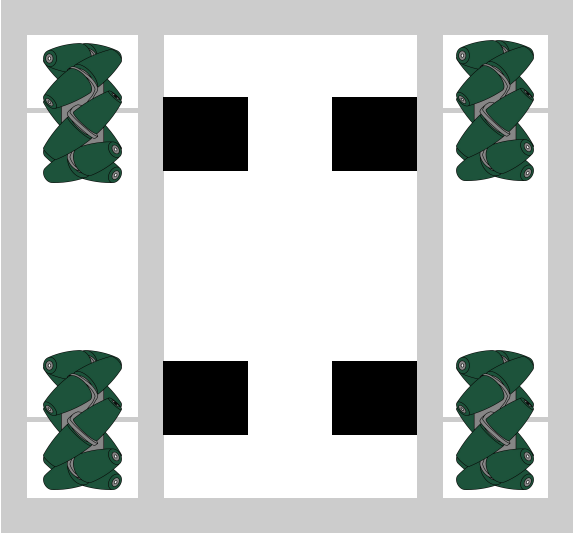
DRIVETRAIN 1.0

MECANUM DRIVE

AUTHOR; RACINE LIU

DATE; 10.14.2024

FEATURES



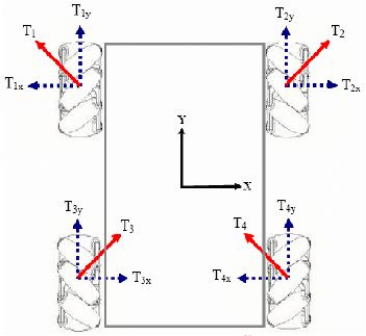
The Mecanum Drive features four wheels, each with rollers set at a 45-degree angle to the wheel's axis. This unique wheel arrangement enables a Mecanum robot to move in any direction by varying the speed and direction of individual wheels.

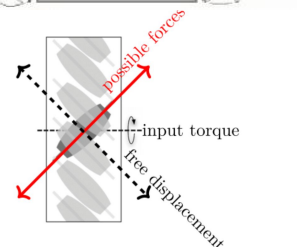
Mecanum robots can seamlessly transition between forward, backward, sideways, and diagonal movements, offering adaptability for diverse game challenges. The intuitive control and maneuverability of Mecanum Drive make it a popular choice for teams aiming for a combination of simplicity and advanced functionality.

STRENGTHS

One significant advantage lies in its omnidirectional movement capability, allowing robots to effortlessly glide in any direction with individual wheel control. This agility is particularly beneficial in navigating complex terrains and executing intricate maneuvers required in competitive scenarios.

Although they are one of the more complex drivetrains involving specific wheel angling and positioning, they are the only drivetrain other than the X-Drive capable of diagonal movement. Mecanum robots can seamlessly translate lateral, diagonal, and rotational movements, providing teams with a versatile platform for strategic gameplay. The ability to strafe with precision enhances a team's ability to outmaneuver opponents and strategically position themselves on the field, contributing to a competitive edge.





Credit: **ResearchGate**
https://www.researchgate.net/publication/268326364_A_Design_Of_Omni-Directional_For_Mobile_Robot

SHORTCOMINGS

One limitation is reduced traction, especially when compared to other drive systems like Tank or X-Drive. The Mecanum wheels' ability to move in any direction relies on the rollers, which can lead to less grip on the playing surface. This can be a concern when navigating over uneven terrain or when faced with aggressive defensive maneuvers from opponents. Additionally, the complexity of the wheel design and the need for precise control during operation may pose challenges for less experienced teams in terms of programming and fine-tuning. Teams often need to invest time in mastering the control nuances of Mecanum Drive to optimize its full potential.

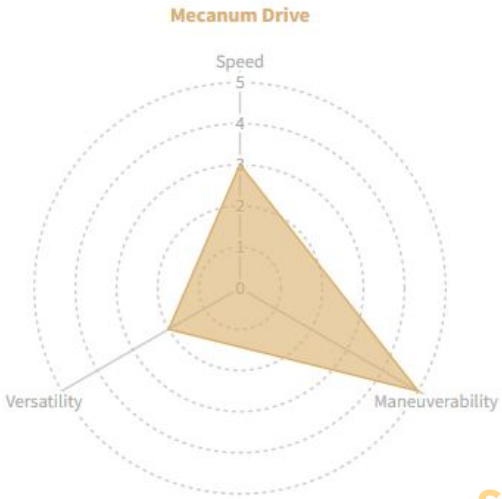
AUTHOR; RACINE LIU

DATE; 10.14.2024

DRIVETRAIN 1.0

MECANUM DRIVE

STATS



Speed

With extra friction, speed may be slower.

Maneuverability

They can traverse across the field and with size adjustments are able to easily navigate through complex code

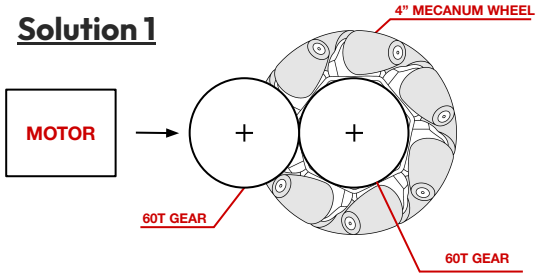
Versatility

Able to withstand some pushes from the side, able to maneuver away. But reduced traction make it less than ideal for both offense and defense.

SOLUTIONS

As the mecanum drive is similar to the tank drive but with slightly lower stats, we are able to use the same solutions. If we gear for too much speed, then the drivetrain becomes unable to control.

Solution 1

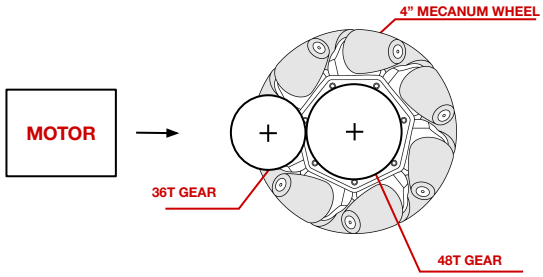


Gear ratio: **1:1**

This gearing means that we get the direct motor power, making it a very fast drivetrain with equal speed and torque.

Engine RPM	600 rpm
Drivetrain transmission ratio	60/60
Wheel RPM	600 rpm
Tire diameter	4 in
Vehicle speed	10.47 ft/s

Solution 2

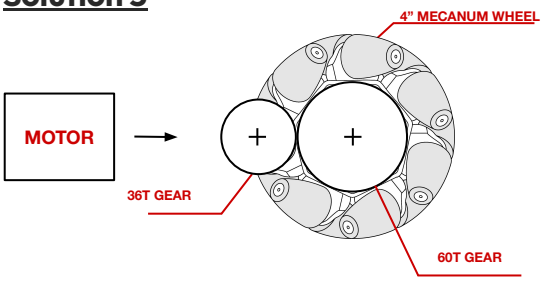


Gear ratio: **4:3**

Output torque is greater than input torque and reduces max speed. We can carry load as well as being capable of pushing more at fast speeds.

Engine RPM	600 rpm
Drivetrain transmission ratio	48/36
Wheel RPM	450 rpm
Tire diameter	4 in
Vehicle speed	7.85 38 ft/s

Solution 3



Gear ratio: **5:3**

High torque, lower speed, and great pushing power. We have experience with this gearing already as we used it for worlds.

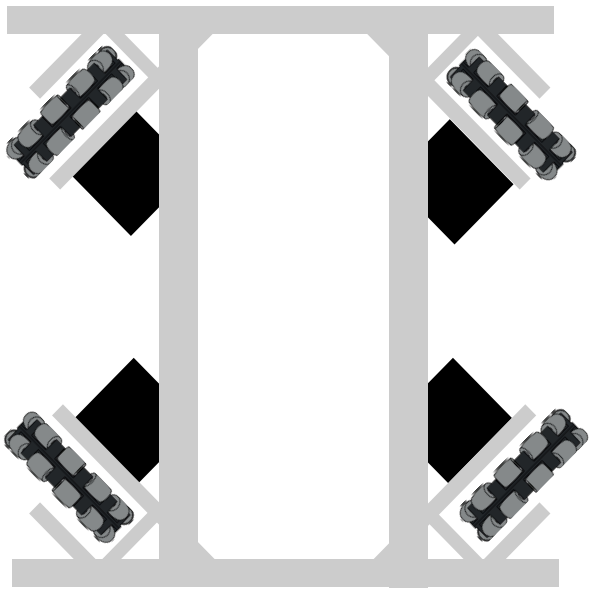
Engine RPM	600 rpm
Drivetrain transmission ratio	60/36
Wheel RPM	4 rpm
Tire diameter	3.25 in
Vehicle speed	6.28 ft/s

DRIVETRAIN 1.0

AUTHOR; RACINE LIU

DATE; 10.14.2024

X-DRIVE



Features

The X drive configuration is characterized by four omni-directional arranged in an "X" pattern, allowing for unparalleled maneuverability and agility. Its unique design enables robots to move seamlessly in any direction without the need for orientation adjustments, facilitating precise lateral and diagonal movement.

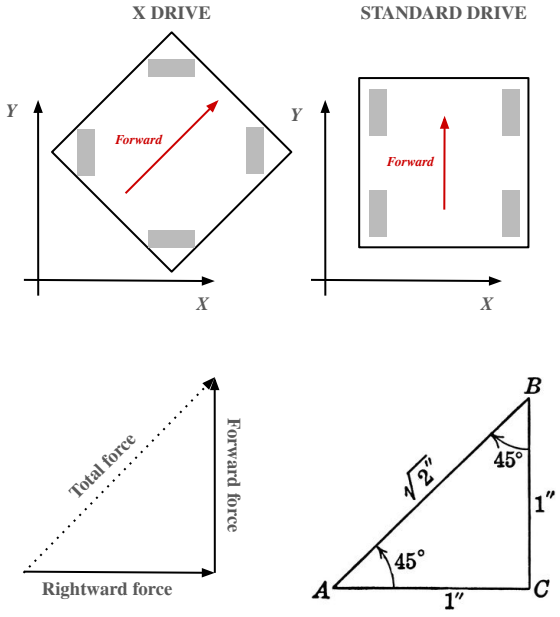
The X-Drive is a unique drivetrain that uses 4 omni wheels in each of its corners all at a 45° angle from the horizontal sides of the robot. The layout of the wheels allows the drivetrain to move omnidirectionally by changing which motors are driving and with what speed.

STRENGTHS

Due to the omnidirectional capabilities of the X drive, maneuverability increases, allowing for more offensive advantages.

When all drivetrains have the same gearing, the X drive will still be faster. This is due to the motors pushing at a 90° angle to each other, adding their force together allowing it to go √2 times faster than a standard drivetrain because the force is going along the hypotenuse. This can be visualized through the diagram to the right. Because the X-Drive has the speed of the tank drive but in both x and y directions, through vector addition when calculating the hypotenuse we form two 45 degree angles hence creating a special triangle with the hypotenuse being:

$1^2 + 1^2 = \sqrt{2}$.



SHORTCOMINGS

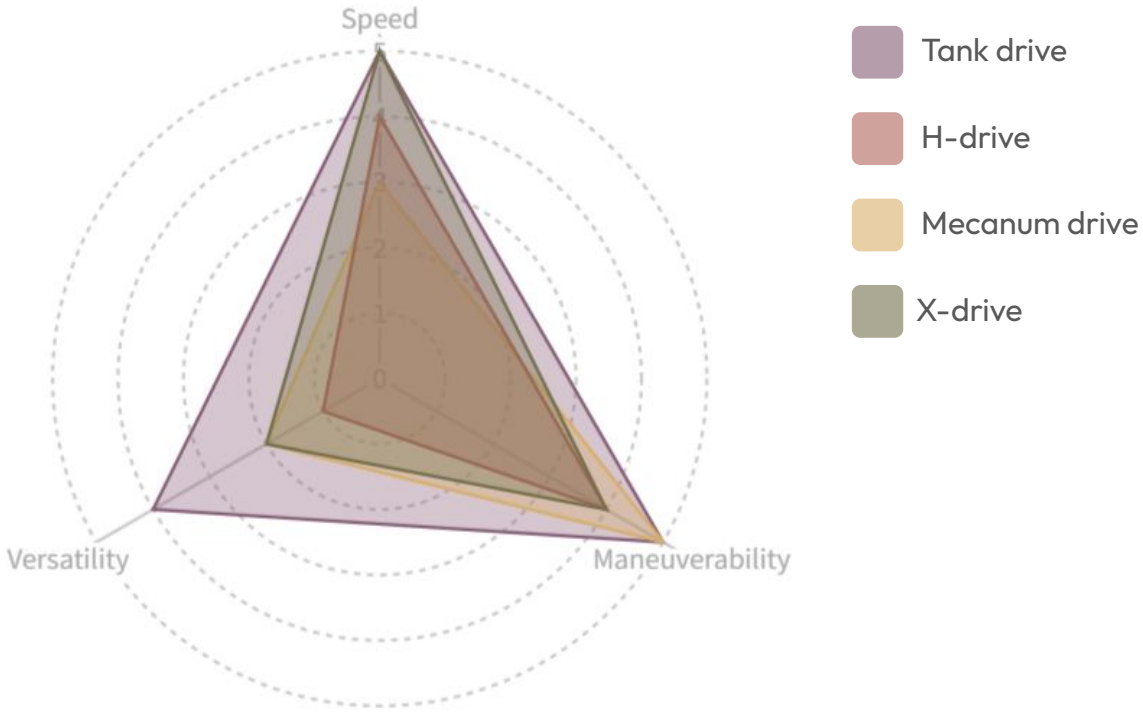
The X-Drive is complex to build with quite a few trade-offs. You trade off torque for more speed and maneuverability. In terms of defensive abilities, it doesn't have the greatest pushing power and can be pushed around easily. It also takes up a lot of room in the drivetrain which may be a detrimental to things like intakes which need to be build close to the field.

DRIVETRAIN 1.0

EVALUATE

RAW RANKING VISUALIZATION

When viewing the raw visualization, it is evident that the capabilities of a **tank drive** encompasses all the other drivetrain capabilities for our requirements.



RANKING

We rank each type of drivetrain on a scale of 1-5 with 5 being the highest and 1, the lowest. Weighting will also be given in terms of what we think is most important by increments of 2 (Speed, Versatility, then Maneuverability). We then total the ratings to get an idea of the highest score.

Criteria	Weight	Tank Drive		H-Drive		Mecanum Drive		X-Drive	
		Rating	Total	Rating	Total	Rating	Total	Rating	Total
Speed	6	5	30	3	18	4	24	5	30
Versatility	4	4	16	2	8	1	4	2	8
Maneuverability	2	5	10	5	10	4	8	4	8
		56		36		36		46	

DRIVETRAIN 1.0

EVALUATE

AUTHOR; RACINE LIU

DATE; 10.14.2024

GEARING SOLUTIONS - PROS/CONS

Solution 1 - 1:1 gear ratio
Pros: Very fast and maneuverable. Will allow us to drive across the field in minimal amounts of time, this will allow us to gain an advantage in games by beating opposing alliances to desired positions.
Cons: May be too hard to handle while driving at 600 rpm on 3.25" omni wheels. Although this gear ratio is equal in speed and torque, there may not be enough pushing power at the raw motor output.

Solution 2 - 4:3 gear ratio
Pros: Still very fast, but easier to control. We still will have a speed advantage, but not in all cases. More pushing power, strong in both offense and defense.
Cons: May not be fastest in all situations.

Solution 3 - 5:3 gear ratio
Pros: We have more experience, allowing for more control. Very strong pushing power.
Cons: No speed advantage

Once we consider our initial 3 goals of speed (for fast scoring efficiency), maneuverability (the ability to easily be controlled and navigate through difficult scenarios), and versatility (in offense and defense). It is clear that **Solution 2** is a balance of all these qualities.

ADDRESSING REMAINING CONSIDERATIONS

"How much of an advantage is there for the drivetrain to be omni-directional?"

There is only a slight advantage of omnidirectional movement. Only in skills will the advantage truly be shown. In games, it is unlikely that omnidirectional movement will be too great of an advantage through heavy offense and defense(robot to robot contact). They will only benefit programming in terms of saving time.

"How many motors are going to be needed for functions other than the drivetrain?"

There are 2 options on the amount of motors that can be used: 4 or 6. As this game relies heavily on robot to robot contact, pushing power and speed is essential, hence we will implement a 6 motor drive. For our remaining two 11W motors or 5.5W motors, we will use for the arm.

VERDICT

We will be creating an 8 wheel, 6 motor Tank drive geared 4:3 at 450rpm on 2.75" omni wheels.

AUTHOR; RACINE LIU

DATE; 10.14.2024

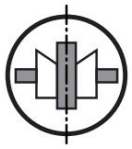
DRIVETRAIN 1.0

PLAN

SPECIAL SYMBOLS


Make sure parts can rotate freely


Make sure gears are properly engaged

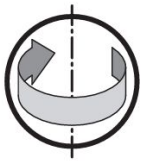

Assemble symmetrically


Special attention in assembly

Complete 2 pieces

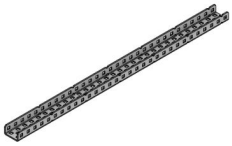
Assemble step 3 according to step 1


Assemble mirrored

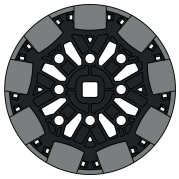
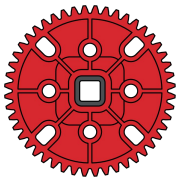
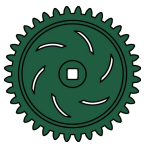

Flip

PARTS LIST

Metal

Item	1x2x1x31 C-channel	1x2x1x27 C-channel	1x2x1x19 C-channel	1x2x1x6 C-channel	1x1x1x5 Angles
Part ID	276-2289	276-2289	276-2289	276-2289	276-6484
Illustration					
Quantity	2	3	1	4	2

Motion

Item	2.75” Omni Wheels	48 Teeth Gears	36 Teeth LS Gears	3” LS Shaft	11W Motors
Part ID	276-8106	276-7573	276-2169-002	276-1149	276-4840
Illustration					
Quantity	8	6	8	8	6

DRIVETRAIN 1.0

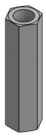
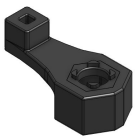
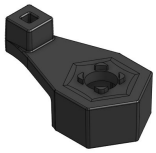
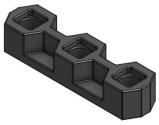
PLAN

AUTHOR; RACINE LIU

DATE; 10.14.2024

PARTS LIST CONT.

Assembly

Item	1" Spacer	½" Spacer	⅜" Spacer	¼" Spacer	1/16" Spacer	1" Standoff
Part ID	Local	276-6343	276-6342	276-6341	Robosource	276-2013
Illustration						
Quantity	8	8	4	12	22	16
Item	Standoff Retainer	Nut Retainer	Low Profile Bearing	Nylock nut	Shaft Collar	Round Bore Insert
Part ID	276-8020	276-6482	276-8023	275-1027	276-2010	276-8034
Illustration						
Quantity	4	12	16	56	10	8

Screws

Item	2 ½" Screw	1 ¼" Screw	⅞" Screw	½" Screw	⅜" Screw	¼" Screw
Part ID	276-8016	276-4997	276-4995	276-5007	276-4991	276-5006
Illustration						
Quantity	8	16	8	20	4	16

AUTHOR; RACINE LIU

DATE; 10.14.2024

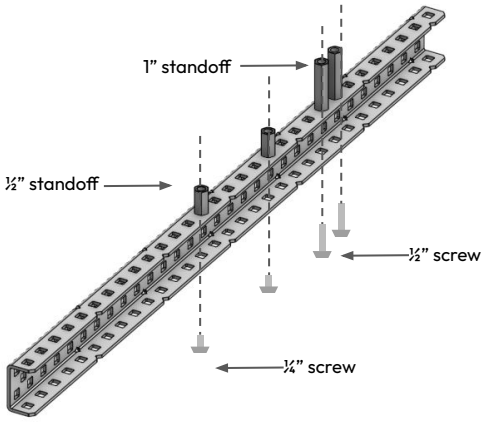
DRIVETRAIN 1.0

STEP BY STEP

STEP 1:

x2

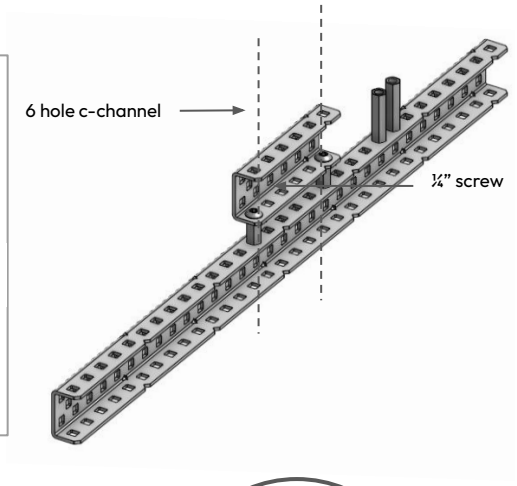
x2
x2
x2
x2
x1



STEP 2:

x2

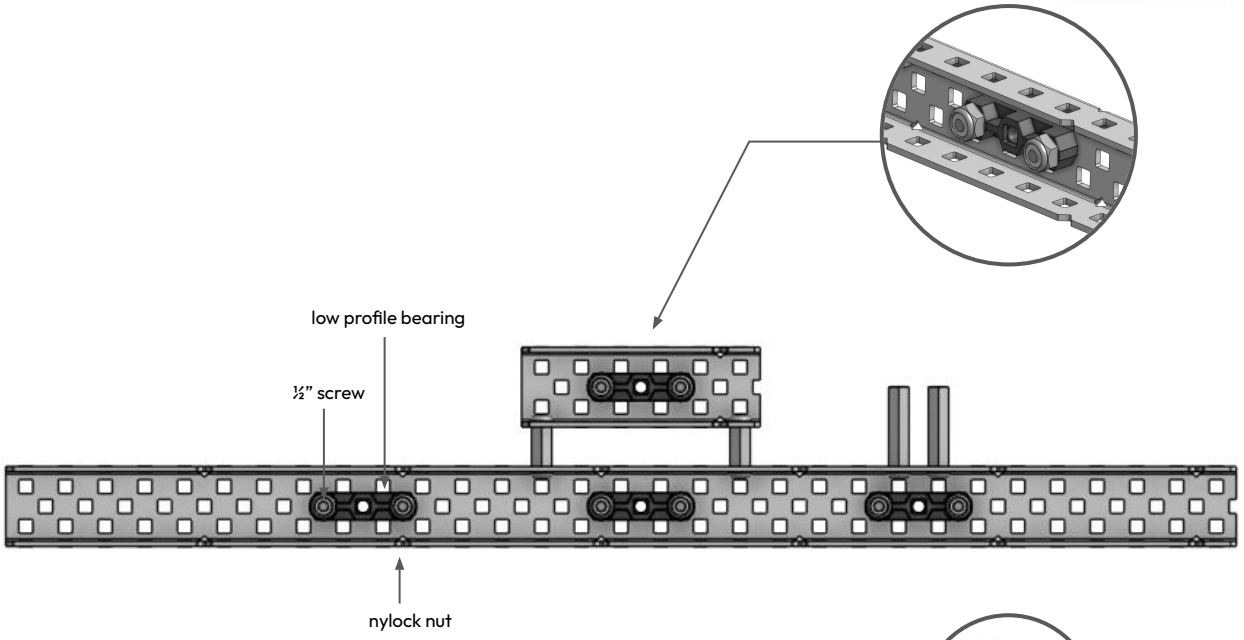
x2
x1



STEP 3:

x2

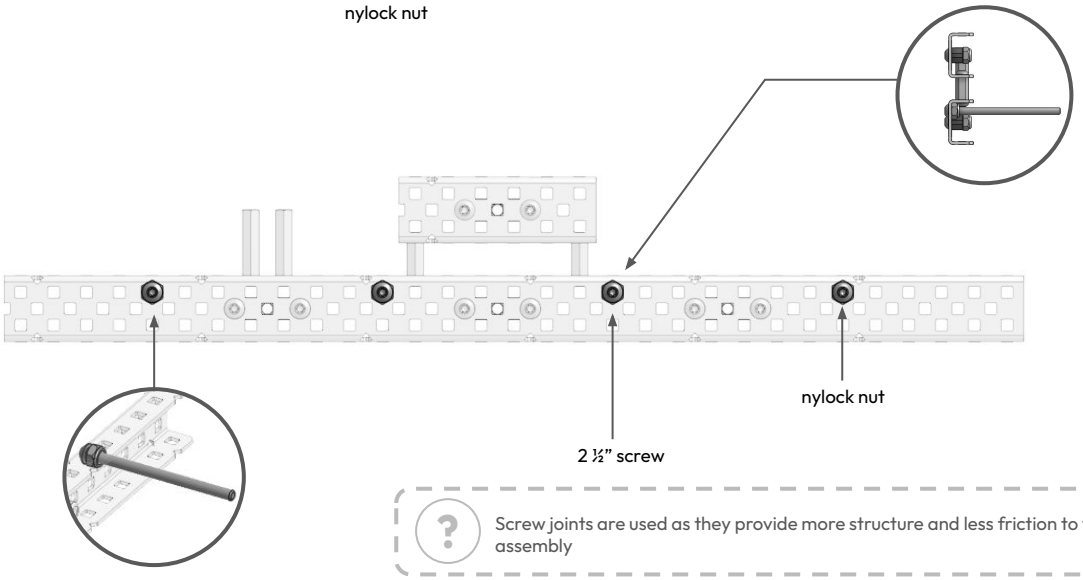
x8
x8
x4



STEP 4:

x2

x4
x4



? Screw joints are used as they provide more structure and less friction to the assembly

DRIVETRAIN 1.0

STEP BY STEP

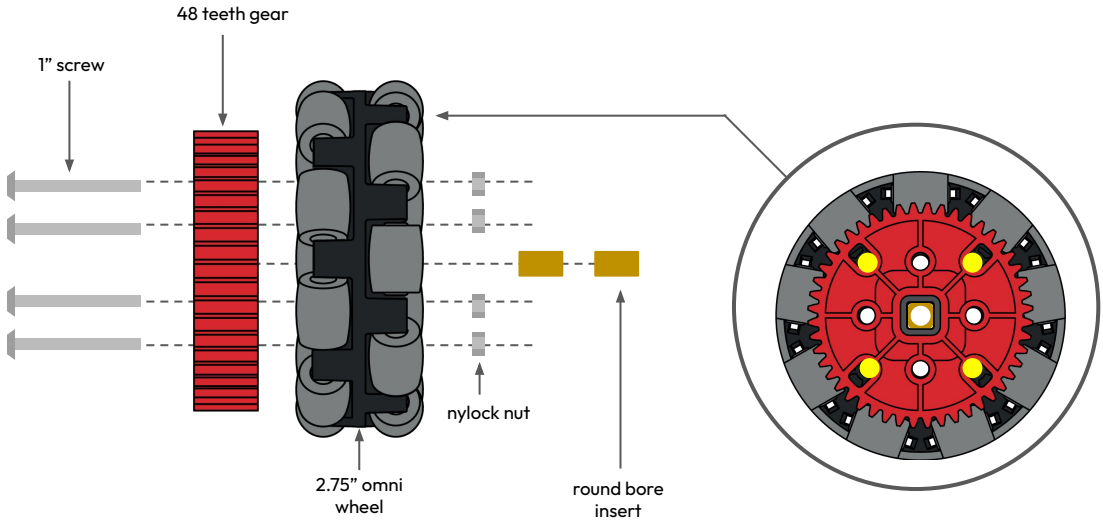
AUTHOR; RACINE LIU

DATE; 10.14.2024

STEP 5:

x8

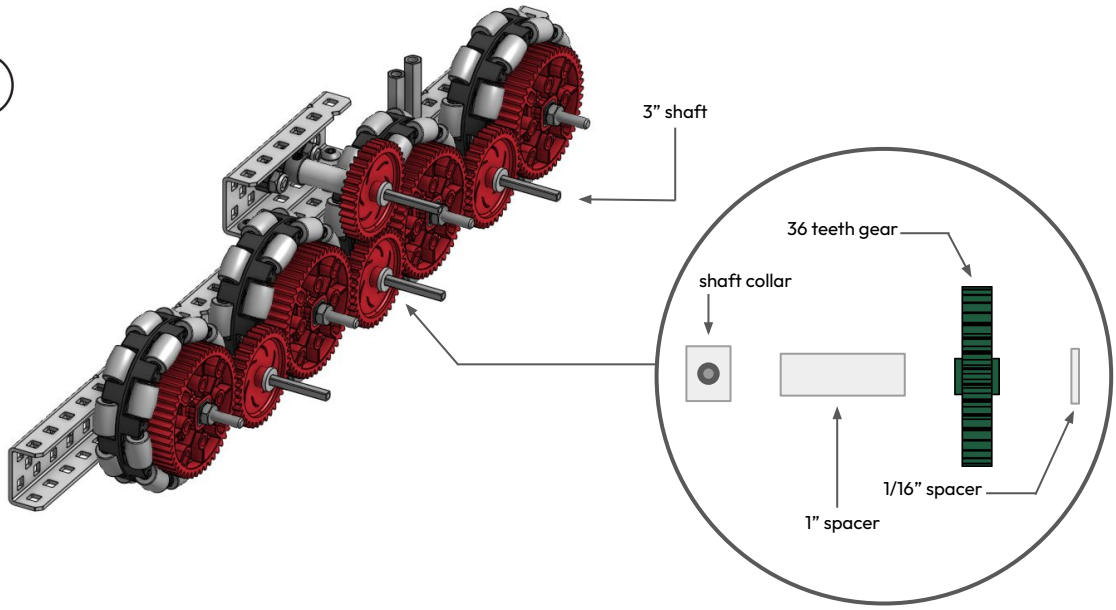
x4
x4
x2
x1
x1



STEP 6:

x2

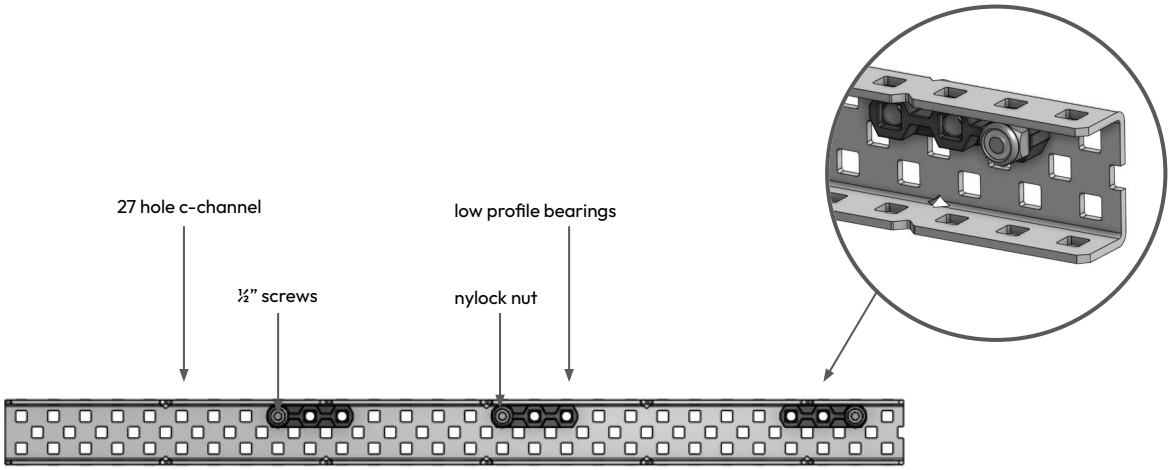
x4
x4
x4
x4
x4



STEP 7:

x2

x3
x4
x3
x1



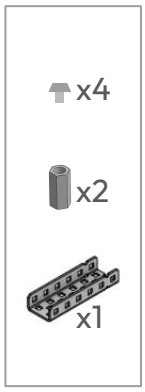
AUTHOR; RACINE LIU

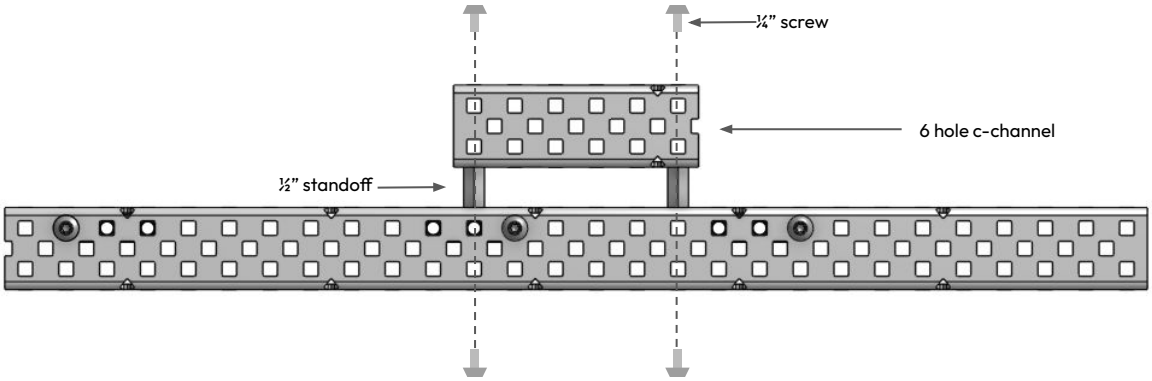
DATE; 10.14.2024

DRIVETRAIN 1.0

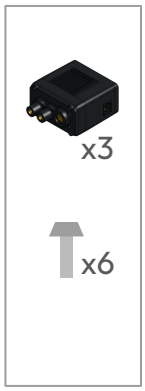
STEP BY STEP

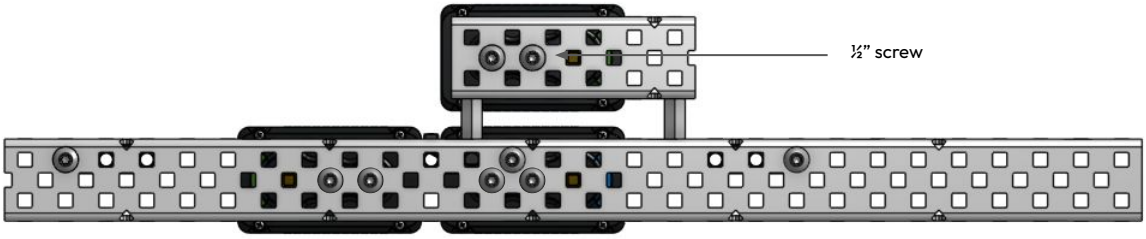
STEP 8:





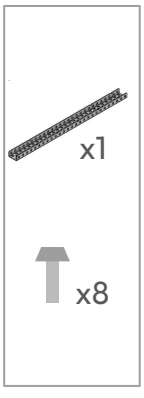
STEP 9:

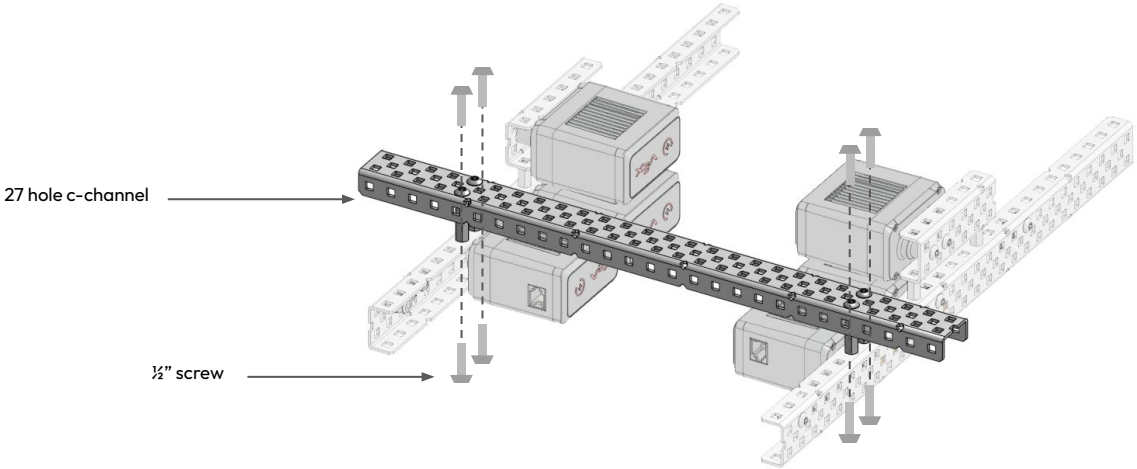




Stacking motors help increase torque and add more power for faster movements

STEP 10:





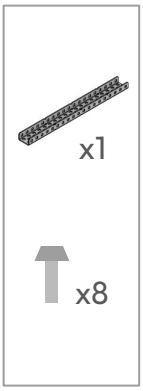
DRIVETRAIN 1.0

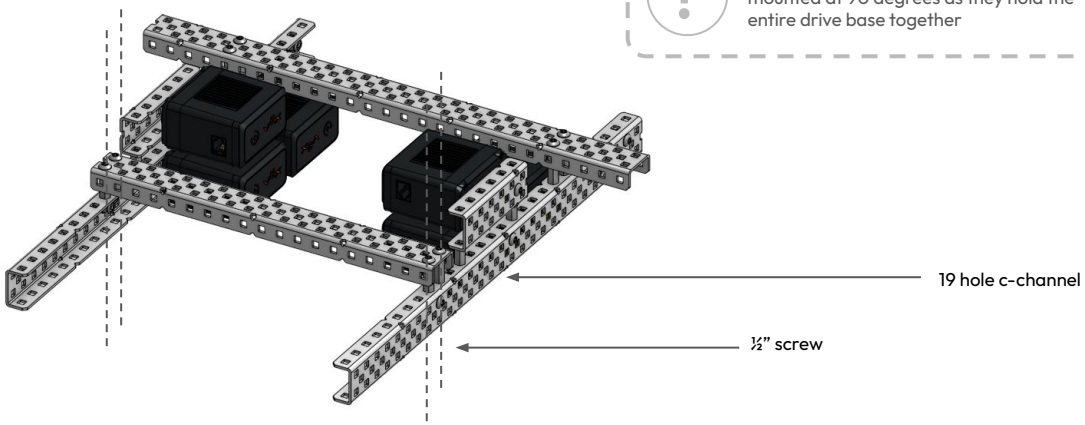
STEP BY STEP

AUTHOR; RACINE LIU

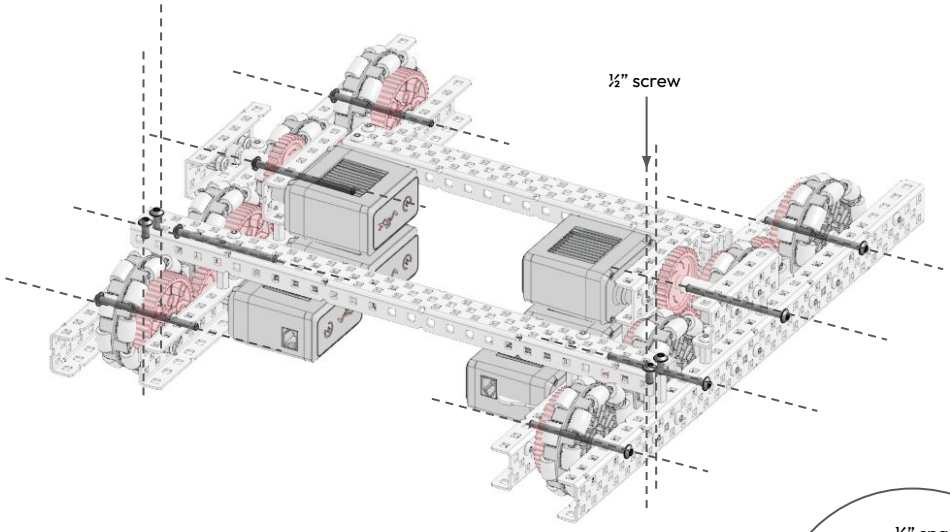
DATE; 10.14.2024

STEP 11:

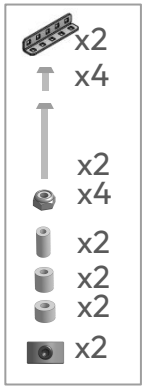


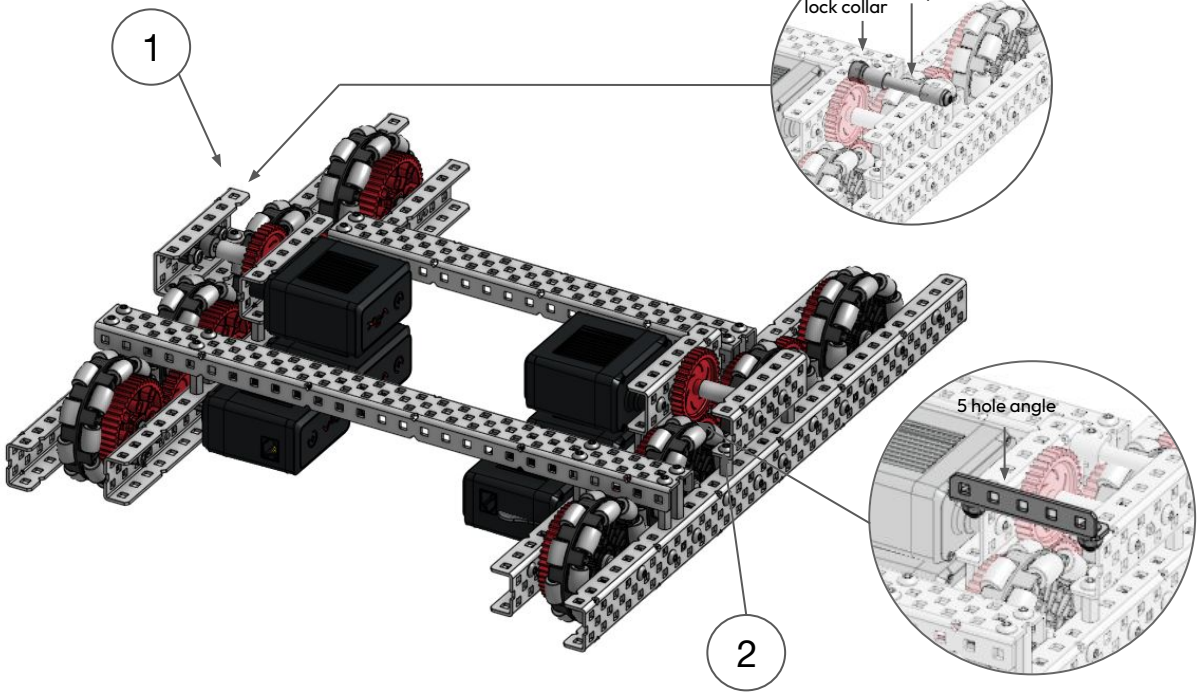


STEP 12:



STEP 13:





AUTHOR; RACINE LIU

DATE; 10.14.2024

CLAMP 1.0

DEFINE/CONSTRAINT

IMPORTANCE

A clamp is essential when handling mobile goals because it provides a secure and stable grip, preventing the mobile goals from slipping or falling during movement. This stability is particularly important when the robot needs to move quickly or navigate obstacles, as a firm hold reduces the risk of errors. Additionally, using a clamp offers greater precision when manipulating mobile goals, allowing the robot to place them accurately in designated scoring zones. With a secure hold scoring rings on the mobile goals will be a much more reliable process

CONSIDERATIONS

- How will the clamp perform under different driving speeds and conditions?
- How much force is required for the clamp to securely grip the mobile goals?
- What materials should the clamp be made from for durability and weight optimization?
- How will the clamp's design affect the robot's stability and mobility?
- Will other teams try and steal the mobile goal when possessed?

SETTING REQUIREMENTS

Security - "Will other teams try and steal the mobile goal when possessed?"

Consistency – "How will the clamp perform under different driving speeds and conditions?"
The clamp's performance under varying driving speeds and conditions is essential for consistent operation. The mobile goal's herxagonal shape may affect how the clamp engages with it, particularly during high-speed maneuvers or rapid direction changes. Ensuring that the clamp can grip and release the goal reliably, regardless of speed, is critical for successful operation during the competition.

Speed – "How quickly can the clamp grip and release a mobile goal?"
The clamp should be able to quickly engage and disengage from the goal to maximize efficiency during gameplay. This responsiveness is essential for executing strategies that require rapid movement between tasks.

CONSTRAINTS

RULE CONSTRAINTS:

As per <R4> the clamp must fit within an 18" x 18" x 18" volume

MATERIAL CONSTRAINTS:

We have full access to all potential materials as it is the start of the season

TIME CONSTRAINTS:

This subsystem should be completed by October 23rd

CLAMP 1.0

RESEARCH

AUTHOR; RACINE LIU

DATE; 10.15.2024

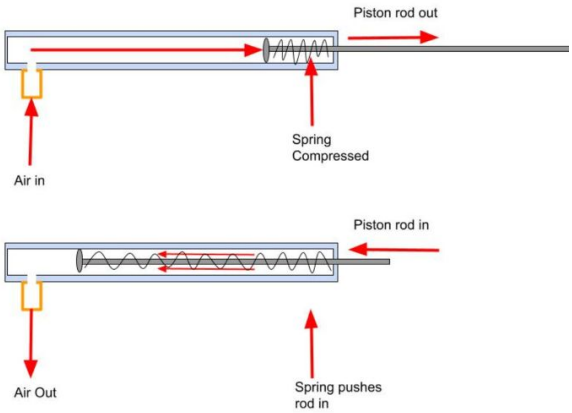
HOW THEY WORK

Pneumatics work by using air pressure. This can be created with something as simple as a bicycle tire pump.

The basic pneumatic system uses a storage tank in which the air pressure can be pumped up with the bike pump, pneumatic tubing to connect the devices, a valve to control the release of pressure, and a pneumatic cylinder. The compressed air can then actuate pneumatic cylinders by either allowing air in or pulling air out. This means that pneumatics are binary; they can only be in one of two states, fully extended and fully retracted.

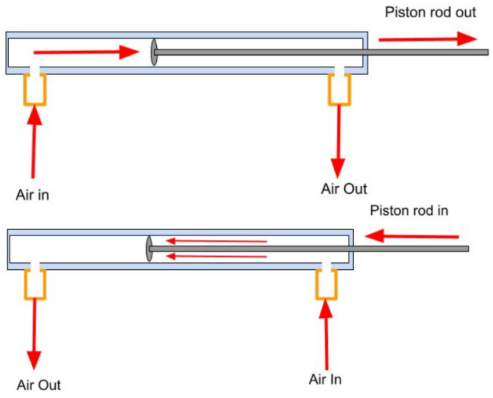
PISTON OVERVIEW

SINGLE ACTING



The Single Acting Cylinders only have one valve that acts both as an intake and exhaust for air. When air is released into the pneumatic it pushes the rod out with the resistance of the spring. Then when air is released, the spring's potential energy is transformed into kinetic energy and the rod gets sprung back. Because of this, with Single Acting Cylinders they retract much faster than they can extend.

DOUBLE ACTING

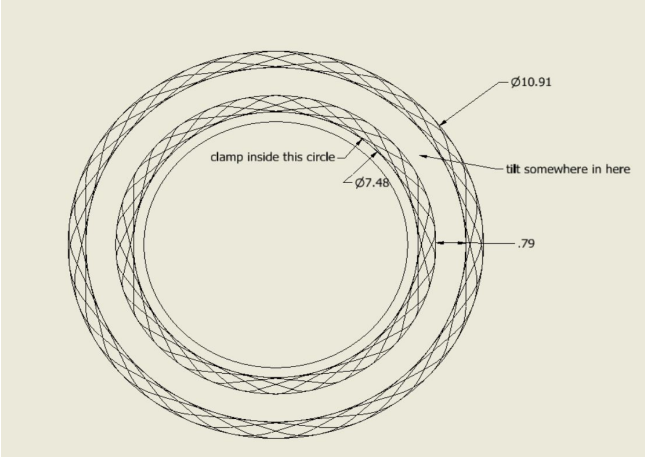


The Double Acting Cylinders have two valves, one as an intake and the other as exhaust for air. A valve releases air pressure into the bottom of the cylinder. The air pressure pushes on the surface area of an internal piston which forces the piston and piston rod out of the cylinder. As the piston/piston rod moves out, exhaust air flows out the top of the cylinder. The valve can also be set to release air pressure into the top of the cylinder. When this happens, the air pressure pushes the piston and piston rod back into the cylinder. As the piston/piston rod moves in, exhaust air flows out the bottom of the cylinder.

CLAMP 1.0

RESEARCH

CLAMPING AREA



To the left is an image made by circumscribing the hexagonal shape of the mobile goal.

We want to find the optimal clamping placement in order to be able to consistently clamp onto the goal no matter the orientation. As indicated in the image.

PHYSICS

A clamp works by applying force over an area to hold an object in place. The force is applied through mechanical advantage, like a screw or lever mechanism, which amplifies the input force.

To justify the relationship between force, pressure, and the contact area in the context of a clamp, we need to look at the basic physics equations. The key relationship here is:

$$P = \frac{F}{A}$$

Where:

- P is the pressure (in Pascals, Pa)
- F is the applied force (in Newtons, N)
- A is the area over which the force is applied (in square meters, m^2)

When tightening a clamp, the force F is applied to an object via the clamp's contact area A. The pressure P exerted by the clamp on the object is dependent on this force and the contact area. F is typically determined by how tightly the clamp is pressed. More force = more pressure where A is the surface of the clamp in contact with object.

CONCLUSION

If A decreases (smaller contact area): The pressure increases, since pressure is inversely proportional to the area. We should aim to have smaller areas of contact when designing our clamp

CLAMP 1.0

BRAINSTORM

AUTHOR; RACINE LIU

DATE; 10.15.2024

SOLUTIONS

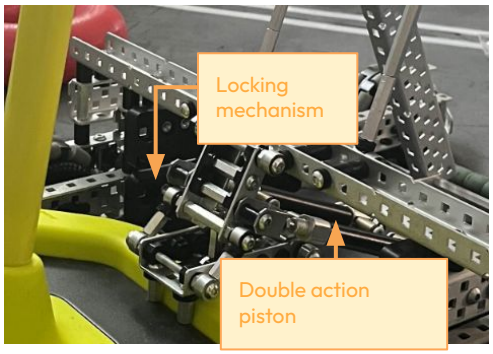
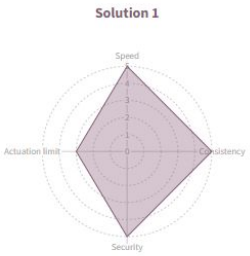
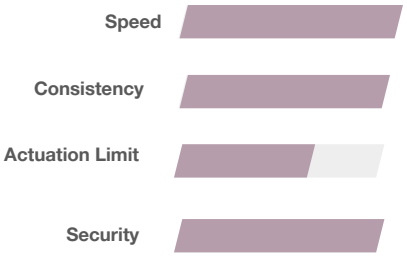


Image courtesy to Team 538S- Released September 19 2024 via messages

LOCKING PISTON

Pros: anti steal, fast pickup and release
Cons: more air with double acting piston





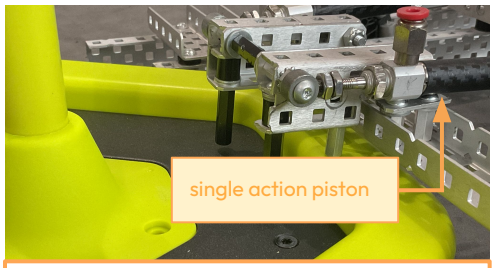
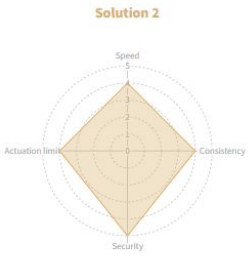
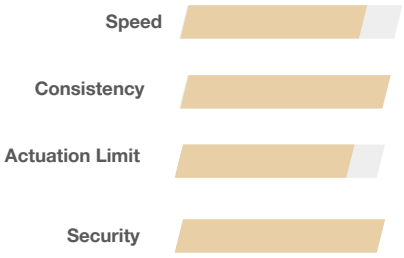


Image courtesy to Team 727R- Released September 25 2024 via messages

PISTON

Single action Piston
Pros: less air in order to deploy
Cons: slower release





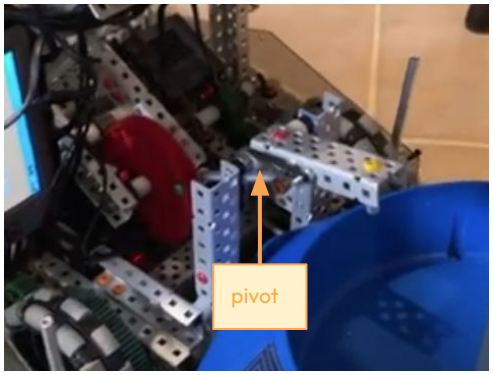
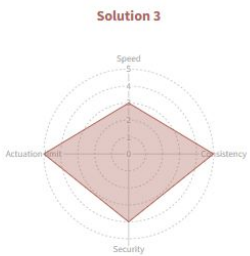
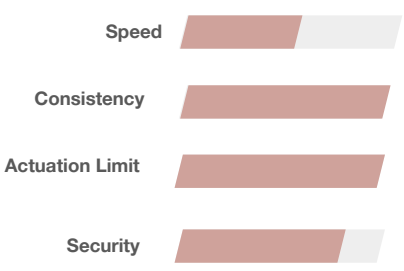


Image courtesy to Team 24118E 0 Piston MogoMech - Released April 8 2024 on YouTube, Timestamp: 0.19

MOTOR

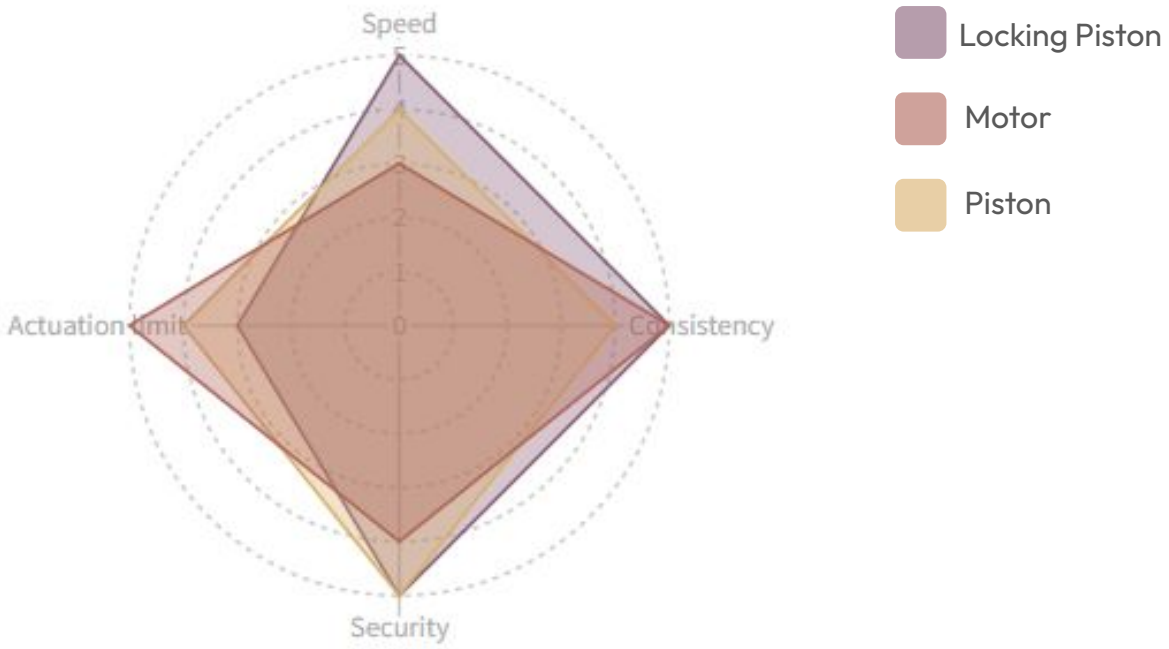
Pros: unlimited uses per match
Cons: other mechanisms could be using motors, big and not space efficient





RAW RANKING VISUALIZATION

When viewing the raw visualization, it is unclear on which mechanism has a significant advantage over another.



RANKING

We rank each type of intake on a scale of 1-5 with 5 being the highest and 1, the lowest. Weighting will also be given in terms of what we think is most important by increments of 2. We then total the ratings to get an idea of the highest score.

Criteria	Weight	Locking Piston		Piston		Motor	
		Rating	Total	Rating	Total	Rating	Total
Actuation	7	3	21	4	28	5	35
Speed	5	5	25	4.5	22.5	3	15
Security	3	5	15	5	15	4	12
Consistency	1	5	5	4	4	4	4
Total		66		69.5		66	

We will be creating a **Piston Non-Locking clamp**

CLAMP 1.0

PLAN

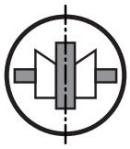
AUTHOR; RACINE LIU

DATE; 10.15.2024

SPECIAL SYMBOLS


Make sure parts can rotate freely


Make sure gears are properly engaged

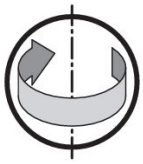

Assemble symmetrically


Special attention in assembly

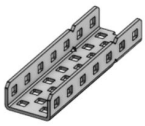
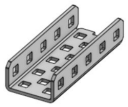
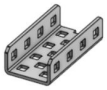
Complete 2 pieces

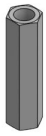
Assemble step 3 according to step 1


Assemble mirrored


Flip

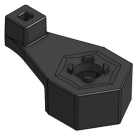
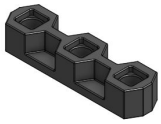
PARTS LIST

Item	1x1x1x19 Angle	1x1x1x5 Angle	1x1x1x4 Angle	1x2x1x7 C-channel	1x2x1x5 C-channel	1x2x1x4 C-channel
Part ID	276-6484	276-6484	276-6484	276-2289	276-2289	276-2289
Illustration						
Quantity	1	2	1	2	2	2

Item	2" Standoff	1" Standoff	½" Standoff	2" Screw	½" Screw	⅜" Screw
Part ID	276-2013	276-2013	276-2013	276-5004	276-5007	276-4991
Illustration						
Quantity	9	6	6	6	24	4

CLAMP 1.0

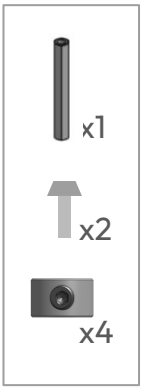
PLAN

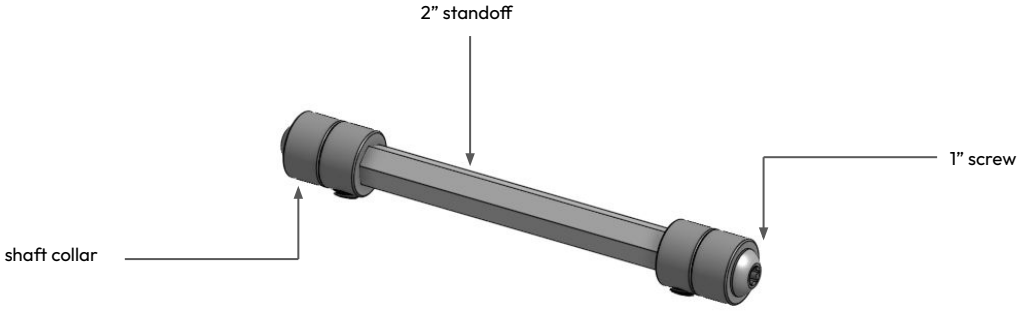
Item	13.5" High Strength Shaft	Coupler	Nylock nut	Nut retainer	Low profile bearings	Shaft collar
Part ID	276-7465	276-4989	275-1027	276-6482	276-8023	276-2010
Illustration						
Quantity	1	4	28	4	14	12

Item	½" Spacer	¼" Spacer	1/16" Spacer	Washer	25mm piston	Pressure Gauge
Part ID	276-6343	276-6341	Robosource	275-1025	276-8642	276-8750
Illustration						
Quantity	2	3	6	10	2	1

STEP BY STEP

STEP 1:





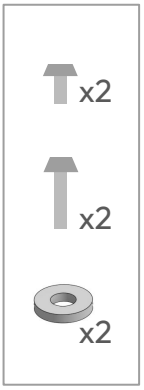
CLAMP 1.0

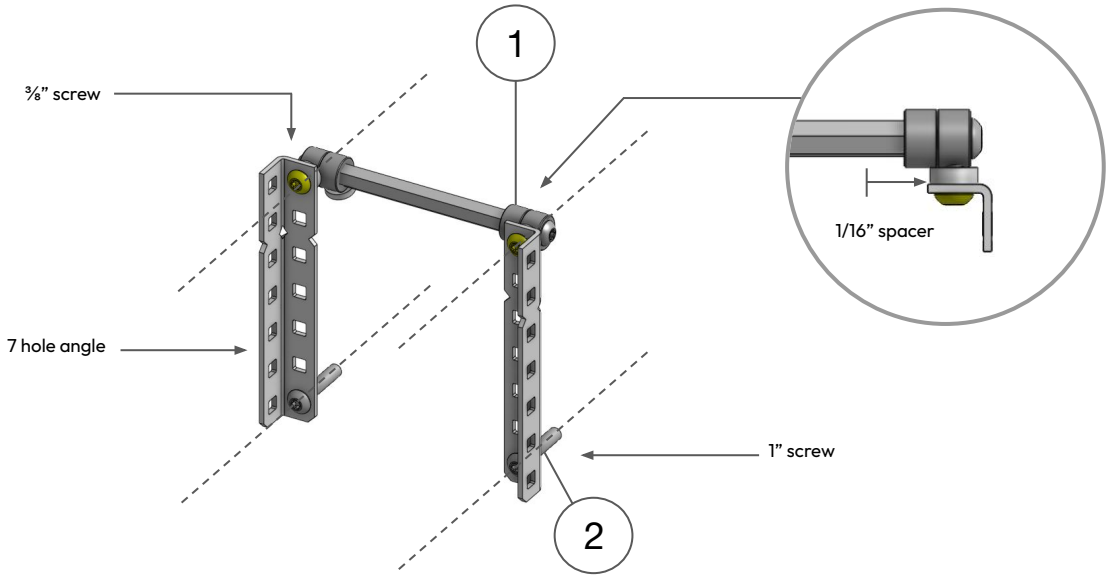
STEP BY STEP

AUTHOR; RACINE LIU

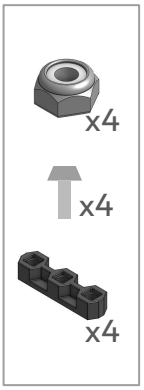
DATE; 10.15.2024

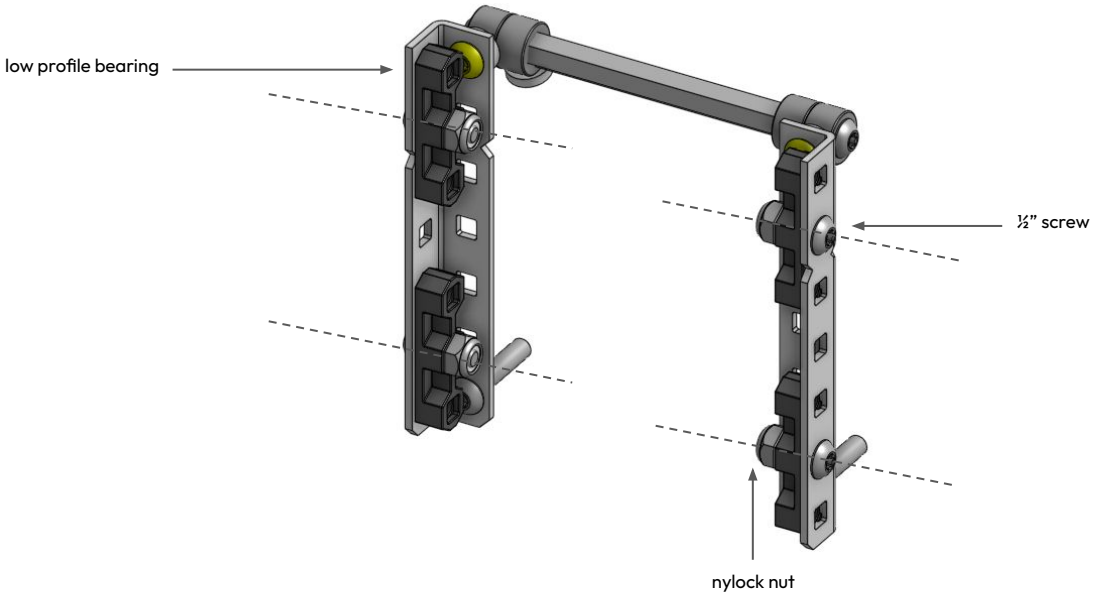
STEP 2:



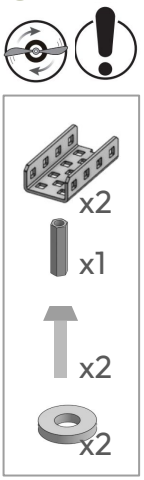


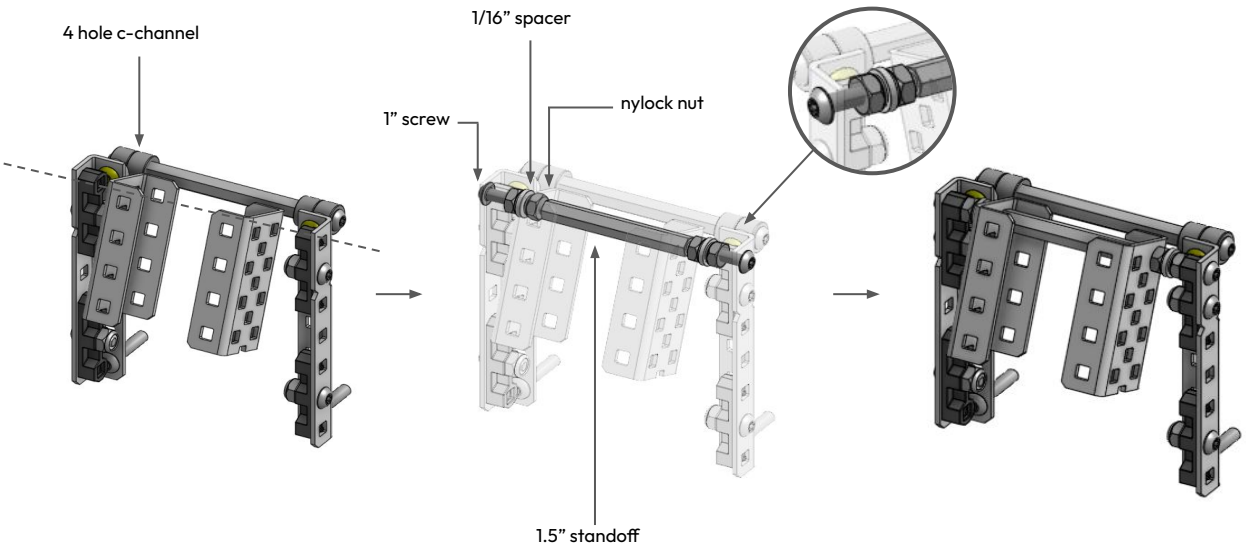
STEP 3:





STEP 4:





AUTHOR; RACINE LIU

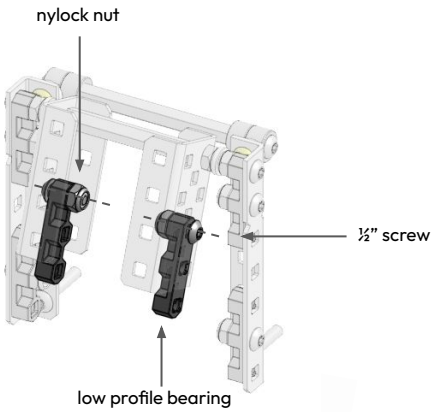
DATE; 10.15.2024

CLAMP 1.0

STEP BY STEP

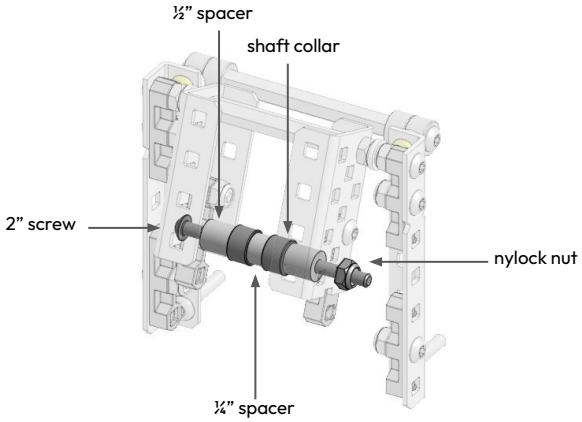
STEP 5:

x2
x2
x2



STEP 6:

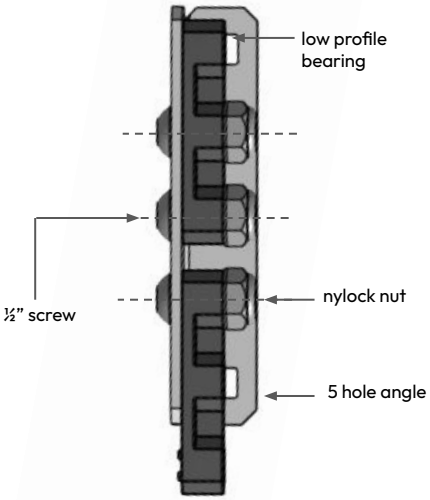
x1
x1
x2
x2
x1



STEP 7:

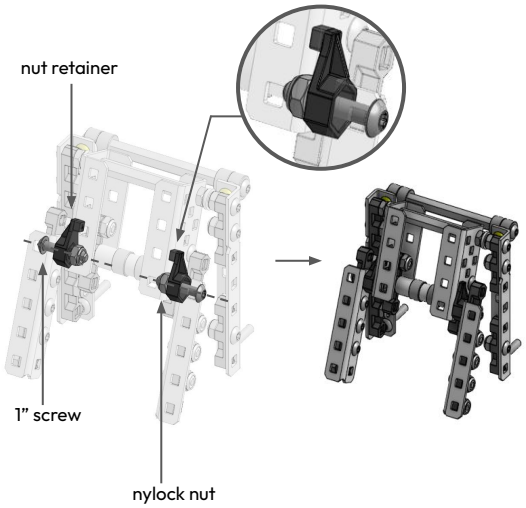
x2

x1
x3
x3
x2



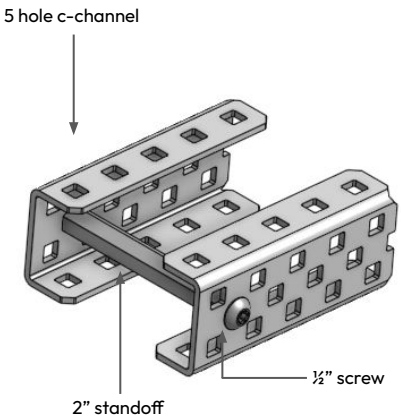
STEP 8:

x2
x2
x2



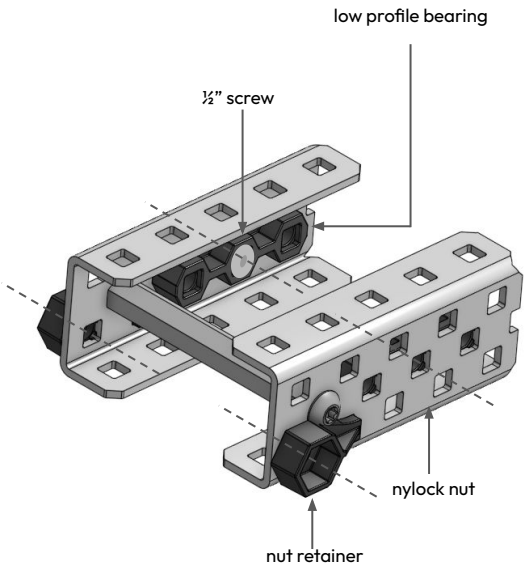
STEP 9:

x2
x2
x1



STEP 10:

x2
x2
x2
x2



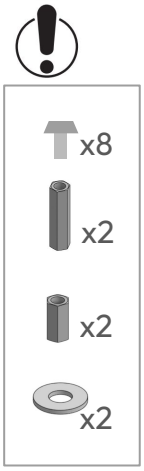
CLAMP 1.0

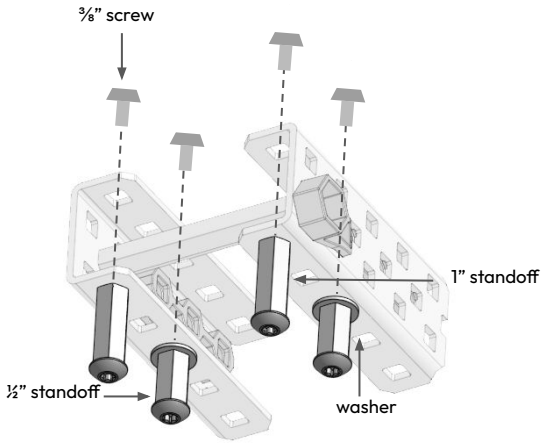
AUTHOR; RACINE LIU

DATE; 10.15.2024

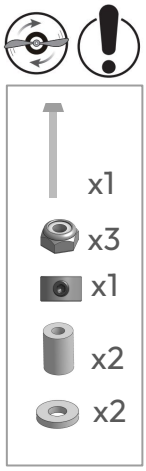
STEP BY STEP

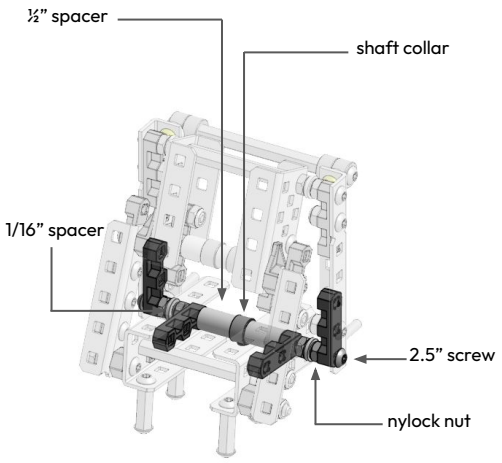
STEP 11:



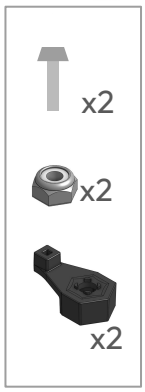


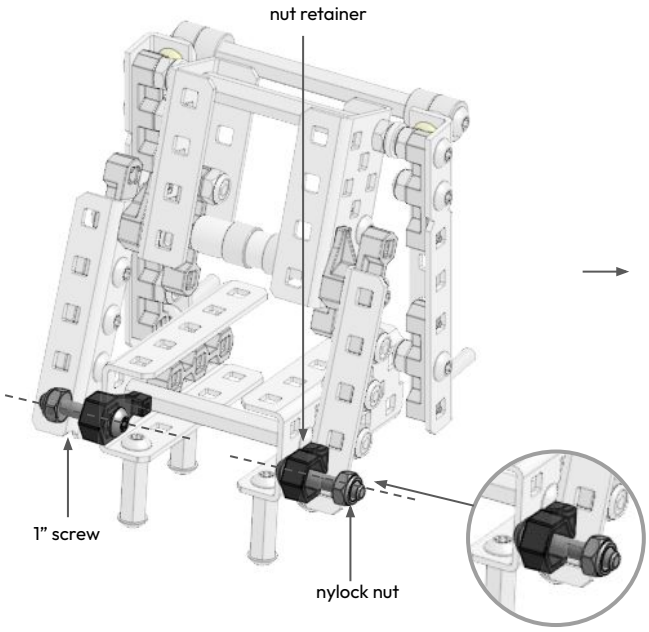
STEP 12:

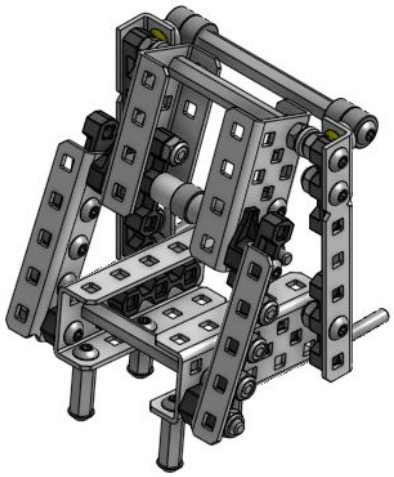




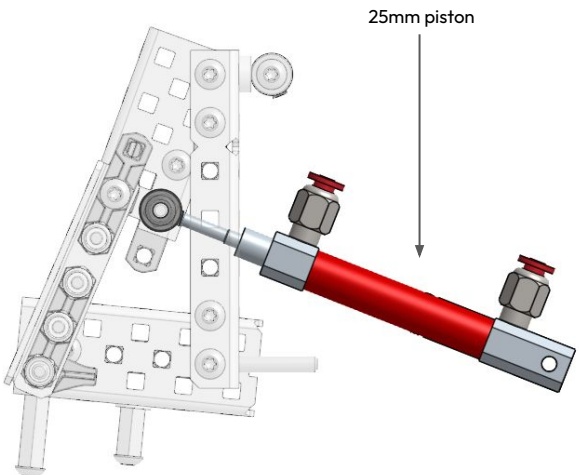
STEP 13:

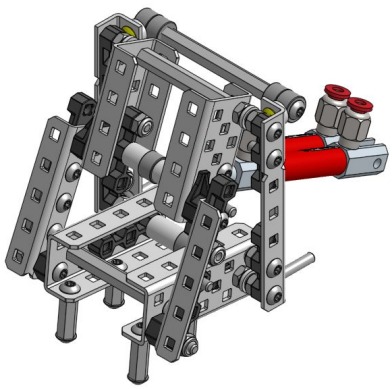






STEP 14:





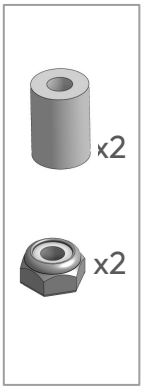
AUTHOR; RACINE LIU

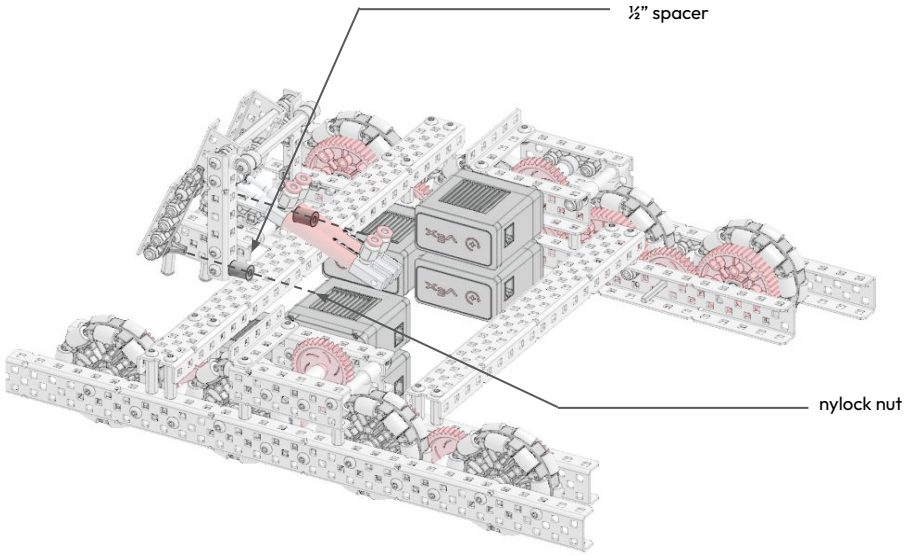
DATE; 10.16.2024

CLAMP 1.0

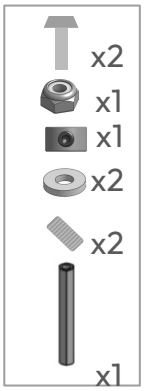
STEP BY STEP

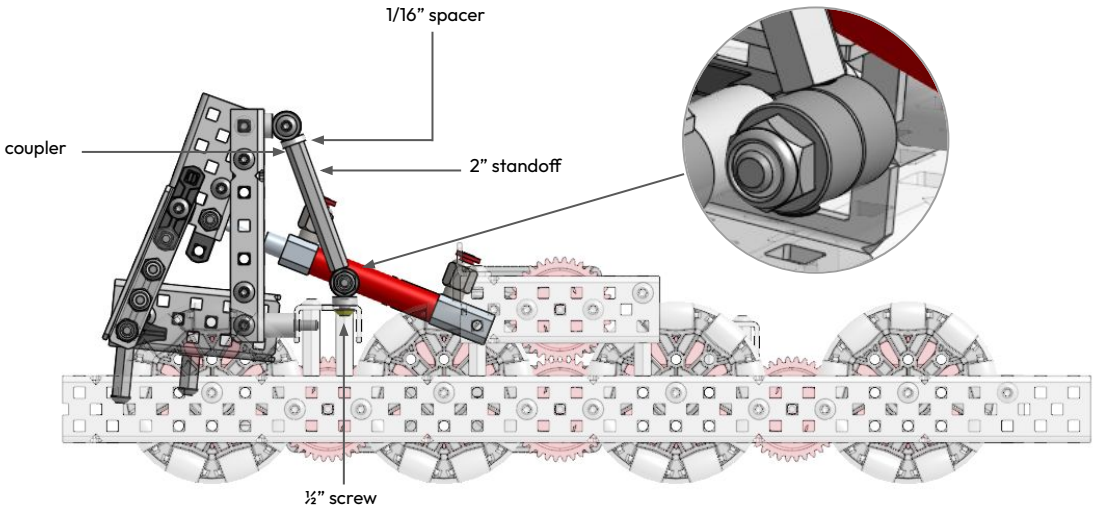
STEP 15:



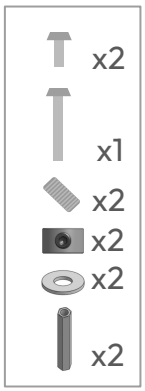


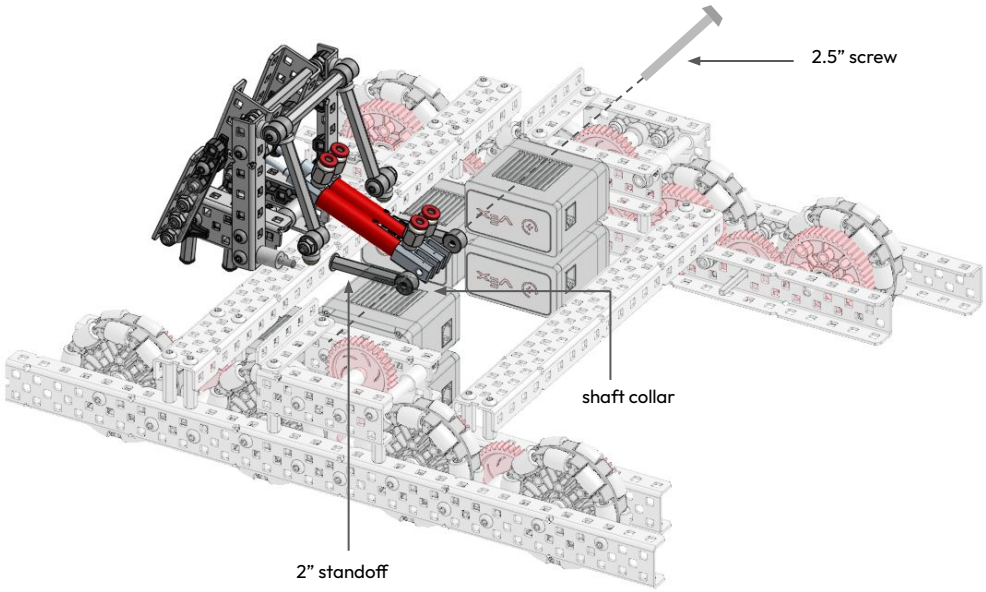
STEP 16:





STEP 17:





INTAKE 1.0

AUTHOR; RACINE LIU

DATE; 10.16.2024

IDENTIFY

IMPORTANCE

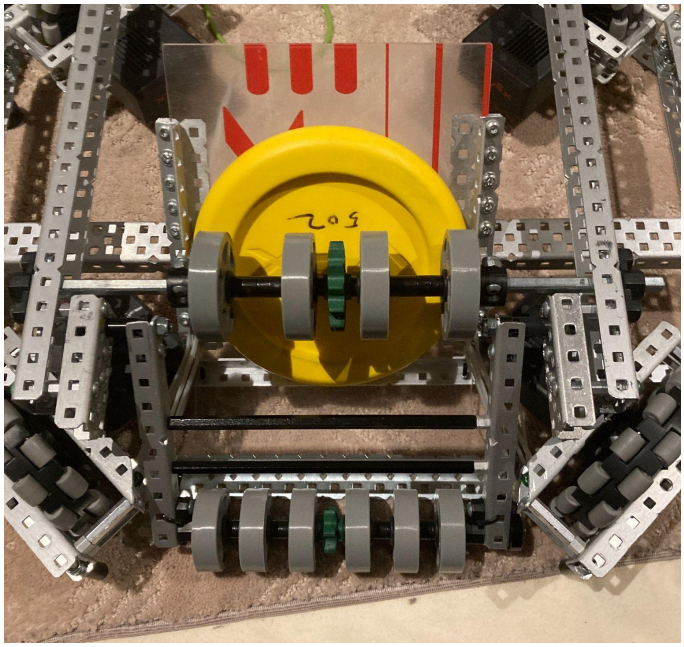
Intakes are one of the most common, versatile, and useful mechanisms in VEX Robotics.They serve as the initial point of contact between the robot and the game elements, allowing the robot to quickly collect, manipulate, or control them as needed during gameplay. Intakes enable robots to interact with game elements in a controlled and precise manner, facilitating tasks such as scoring points, building structures, or completing objectives within the game. Intakes may be especially useful this season when scoring the rings onto the mobile goals.

CONSIDERATIONS

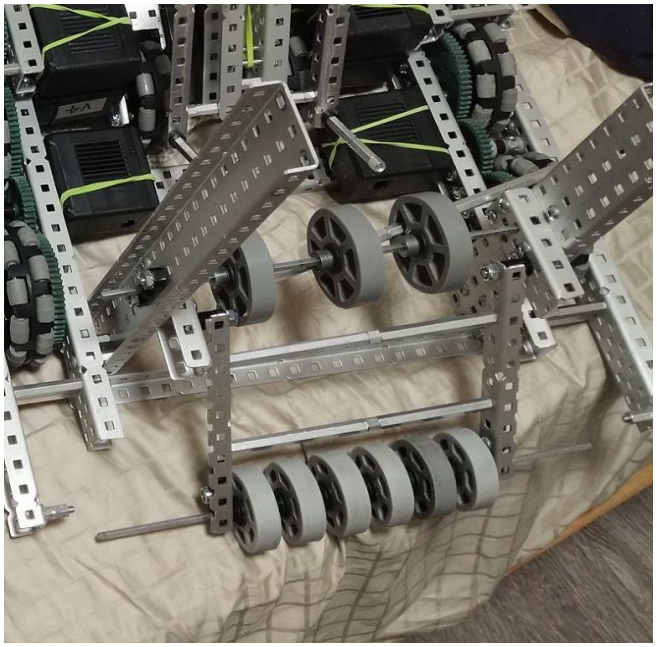
- How much defense will the intake be exposed to while possessing the Triball?
- Are there any power or weight considerations that need to be addressed in the design of the intake mechanism?
- What speed and efficiency are required to collect and manipulate game elements effectively?

PREVIOUS EXPERIENCE

In the season of Spin up, we started out the season as a push bot without an intake. We quickly learned how essential intakes are to possessing game objects and scoring them. We later added a flex wheel intake, which remained with us throughout the rest of the season. This experience allowed us to gain insight on the importance or grip and funneling of an intake.



Our first intake version using flex wheels on our provincial's robot. We found that the intake tended to get deformed from wear as our main issue.



Our second intake version, still using flex wheels. Here we added more bracing so that performance would remain optimal.

AUTHOR; RACINE LIU

DATE; 10.16.2024

INTAKE 1.0

CONSTRAINT

SETTING REQUIREMENTS

Grip

"How much defense will the intake be exposed to while possessing the Triball"

As discussed on previous pages, we predict that High Stakes will be a very contact heavy game. This means that rigorous defense should be expected, making grip essential for keeping control of the Rings.
More grip also allows for more precision when scoring the Rings onto the goal as well as being able to quickly possess it.

Lightweight

"Are there any power or weight considerations that need to be addressed in the design of the intake mechanism?"

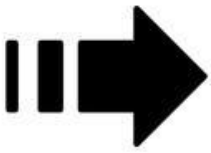
Generally speaking, a robot balanced in weight will be less prone to unexpected events in games. The lighter the intake the lighter the robot as it is a major component.

Being lighter allows for more speed.

Speed

"What speed and efficiency are required to collect and manipulate game elements effectively?"

High speed and efficiency in intaking and redirecting rings will be preferred. This allows for maximization of time allowing us to score more and smoothly as well.



CONSTRAINTS

RULE CONSTRAINTS:

As per <R4> the intake must fit within an 18" x 18" x 18" volume when mounted onto the drivetrain
As per <SG7> Possession is limited to 2 rings at a time

MATERIAL CONSTRAINTS:

We have full access to all potential materials as it is the start of the season

TIME CONSTRAINTS:

This subsystem should be completed by October 24th

54

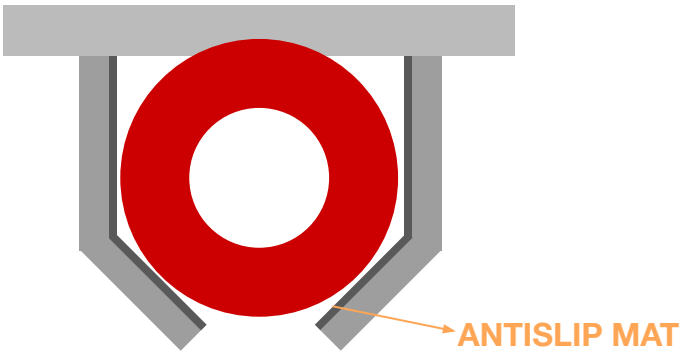
INTAKE 1.0

CLAW

AUTHOR; RACINE LIU

DATE; 10.16.2024

FEATURES



Claws serve as versatile and efficient mechanisms for manipulating game elements. These claws typically feature a gripping mechanism with movable "arms" that can open and close to grasp objects. Claws often offer precise control over gripping force, allowing robots to securely hold game elements while navigating the field or completing scoring objectives.

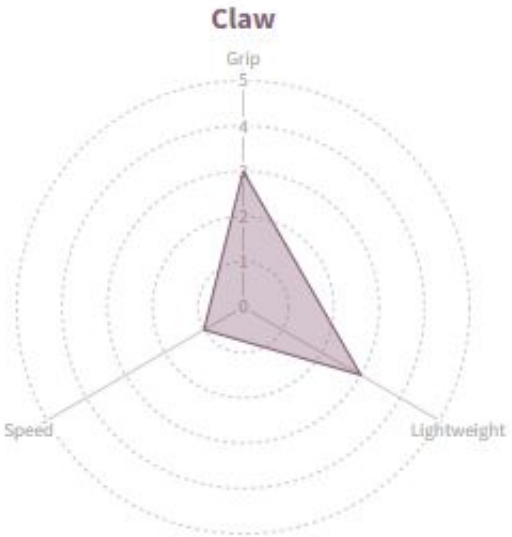
STRENGTHS

One of the primary strengths of claws lies in their versatility and adaptability, allowing robots to effectively handle a wide range of game elements during competitions. The adjustable features of these claws enable quick customization to match specific game requirements, facilitating efficient scoring and task completion. Furthermore, the precise control over gripping force enhances the robot's ability to securely hold and maneuver game elements on the field, contributing to overall performance and competitiveness.

SHORTCOMINGS

One common challenge is achieving consistent and reliable gripping performance, particularly when dealing with irregularly shaped or slippery game elements. Additionally, the size and weight of the claws may impact the overall maneuverability and agility of the robot, especially in dynamic and fast-paced competitions. Claws may take more time to open and close, which may cause lost in time.

STATS



Grip

Has sufficient grip, yet Ring may get stolen during rigorous robot contact.

Lightweight

Made of C-Channels, in order to be strong against defense more bracing is needed hence making it heavier.

Speed

Slow, opening and closing the arm may prove to be low in efficiency.

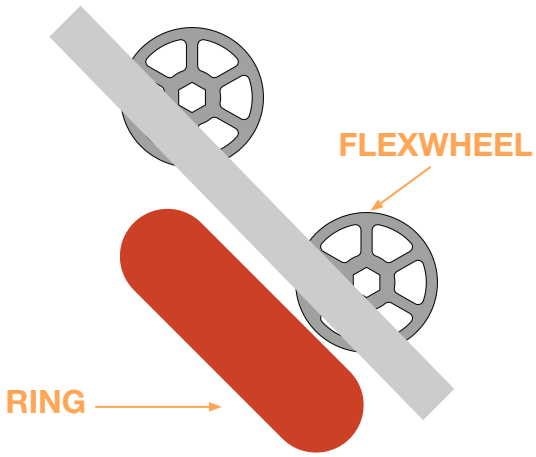
AUTHOR; RACINE LIU

DATE; 10.16.2024

55

INTAKE 1.0

FLEX WHEEL



FEATURES

Consists of flex wheel assemblies that conform to the shape of game elements, allowing for efficient collection and manipulation. These wheel assemblies typically consist of multiple rows of wheels mounted on a shaft, providing a wide contact surface for gripping objects. The flexible nature of the wheels enables them to adapt to various shapes and sizes of game elements encountered on the field, facilitating versatile intake capabilities.

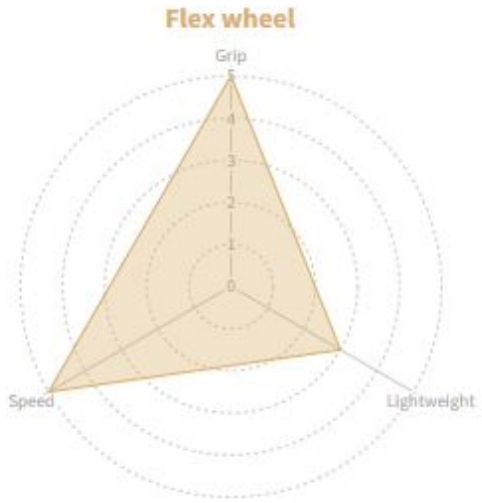
STRENGTHS

They can quickly and reliably collect game elements with minimal manual adjustment. The flexible wheels provide a gentle yet secure grip on objects, reducing the risk of damage or slippage during intake. This allows robots to efficiently gather game elements from different orientations and positions on the field, contributing to faster cycle times and improved scoring. Furthermore, the adjustable features of these intakes enable teams to fine-tune performance according to specific game requirements, enhancing versatility and adaptability in competition.

SHORTCOMINGS

Flex wheel intakes may be heavier than other intakes, as well as having some level of difficulty with funneling. Though the flex wheels have malleable shapes they will still have difficulty maintaining consistent gripping performance across various game elements, particularly those with irregular shapes or surface textures.

STATS



Grip

Malleable flex wheels, great friction against Rings, allowing for lots of grip.

Lightweight

May be heavy with sufficient bracing.

Speed

Very fast as flex wheels spin to intake.

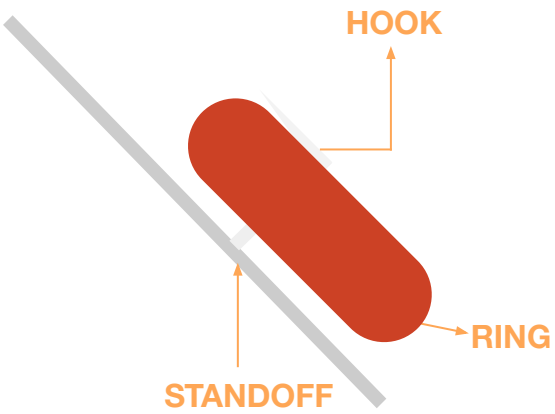
INTAKE 1.0

AUTHOR; RACINE LIU

DATE; 10.16.2024

HOOK

FEATURES



A hook conveyor is designed to pick up and transport objects using a series of hooks attached to a rotating or linear conveyor system. These hooks can grab objects from a field and move them into the robot's mechanism for scoring or storage. The system is typically powered by motors and can be programmed for both manual and autonomous control, offering flexibility in design and function.

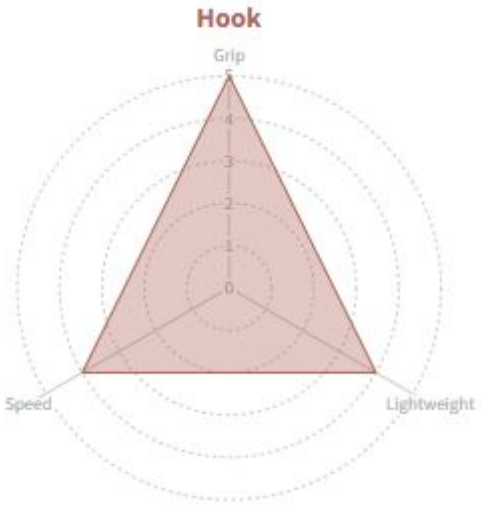
STRENGTHS

One of the key strengths of a hook conveyor intake is its ability to handle irregularly shaped objects, making it versatile in games with varied object designs. It's relatively simple to build and maintain, with fewer moving parts compared to more complex intake systems. This simplicity not only increases reliability during matches but also allows for faster troubleshooting if something goes wrong. The hook system also excels at precision handling, as the hooks can be designed to securely grip and transport specific game objects, reducing the chances of objects slipping or being dropped during movement.

SHORTCOMINGS

It can struggle with object alignment, especially if the objects are scattered unpredictably on the field, leading to missed grabs and wasted time. Additionally, the hooks may not be strong enough to reliably lift heavier or larger objects, making it less effective in games that require handling bulkier materials. The system can also be sensitive to jamming if objects get stuck between hooks or if they're misaligned, potentially causing mechanical failure during crucial moments in a match.

STATS



Grip

Has sufficient grip, as it can hook onto the center of the ring

Lightweight

Conveyer is light as well as hooks

Speed

Moderate speed. At fast speeds tuning may be extensive.

AUTHOR; RACINE LIU

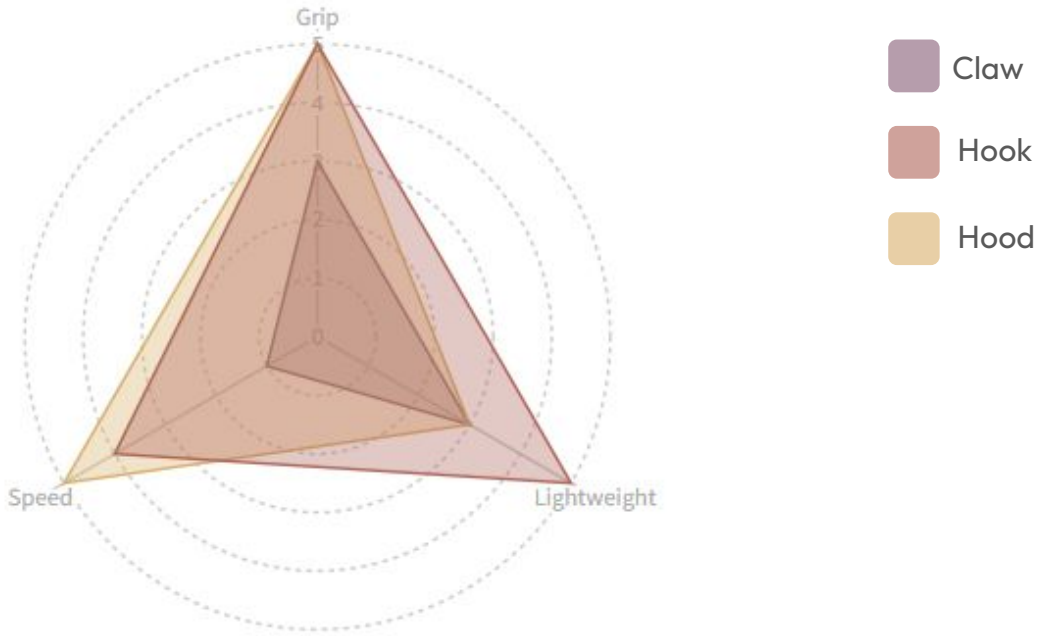
DATE; 10.16.2024

INTAKE 1.0

EVALUATE

RAW RANKING VISUALIZATION

When viewing the raw visualization, it seems hood and hook are the clear majority.



RANKING

We rank each type of intake on a scale of 1-5 with 5 being the highest and 1, the lowest. Weighting will also be given in terms of what we think is most important by increments of 2. We then total the ratings to get an idea of the highest score.

Criteria	Weight	Claw		Hood		Hook	
		Rating	**Total**	Rating	**Total**	Rating	**Total**
Grip	5	3	15	5	25	5	25
Lightweight	3	3	9	3	9	5	25
Speed	1	1	1	5	5	4	4
Total		25		39		54	

After observing the raw ranking and the decision matrix, we have concluded to create a **Hook intake**

INTAKE 1.0

PLAN

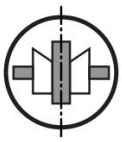
AUTHOR; RACINE LIU

DATE; 10.17.2024

SPECIAL SYMBOLS


Make sure parts can rotate freely


Make sure gears are properly engaged

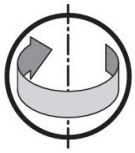

Assemble symmetrically


Special attention in assembly

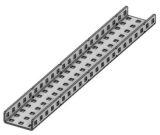
Complete 2 pieces

Assemble step 3 according to step 1


Assemble mirrored


Flip

PARTS LIST

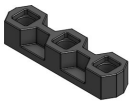
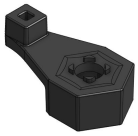
Item	1x3x1x22 C-channel	1x3x1x3 C-channel	1x2x1x33 C-channel	1x2x1x20C -channel	1x2x1x12 C-channel	1x2x1x2 C-channel
Part ID	276-4359	276-4359	276-2289	276-2289	276-2289	276-2289
Illustration						
Quantity	1	1	2	1	2	4

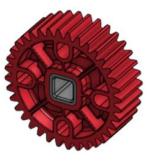
Item	1x1x1x26 Angle	1x1x1x19 Angle	1x1x1x16 Angle	1x1x1x15 Angle	1x1x1x12 Angle	1x1x1x3 Angle
Part ID	276-6484	276-6484	276-6484	276-6484	276-6484	276-6484
Illustration						
Quantity	2	1	2	2	2	1

INTAKE 1.0

PLAN

Item	11” HS Shaft	9.5” HS Shaft	9.5” LS Shaft	11W Motor	45A 2” Flex wheel	30A 2” Flex wheel
Part ID	276-7465	276-7465	276-1149	276-4840	217-6353	217-6353
Illustration						
Quantity	1	1	2	1	4	5

Item	Adapters	Coupler	Pillow block bearing	Low profile bearing	Shaft collar	Nut retainer
Part ID	217-8004	276-4989	276-8383	276-8023	276-2010	276-6482
Illustration						
Quantity	18	4	6	14	16	4

Item	12T Spur gears	Round bore insert	8T Sprocket	12T Sprocket	36T HS Gear	36T LS Gear
Part ID	276-2251	276-8034	276-8030	276-3877	276-7747	276-2169-002
Illustration						
Quantity	2	6	10	2	4	4

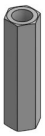
INTAKE 1.0

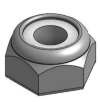
PLAN

AUTHOR; RACINE LIU

DATE; 10.17.2024

Item	½" HS Spacer	¼" HS Spacer	1" Spacer	½" Spacer	⅜" Spacer	¼" Spacer
Part ID	276-3441	276-3441	Local	276-6343	276-6342	276-6341
Illustration						
Quantity	8	4	2	4	6	12

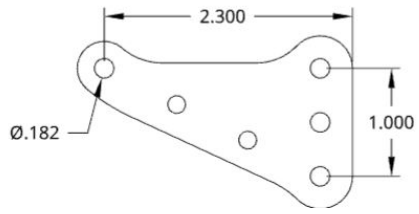
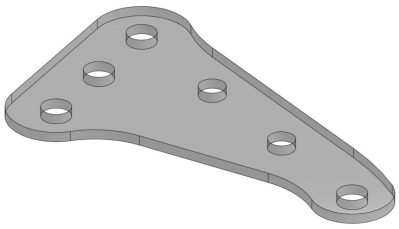
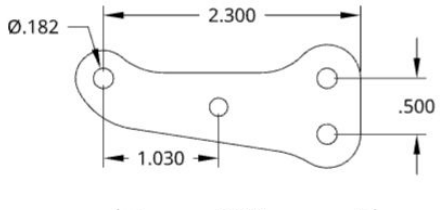
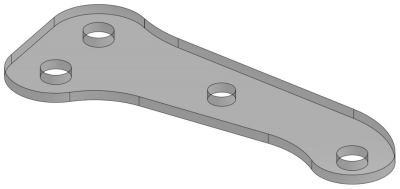
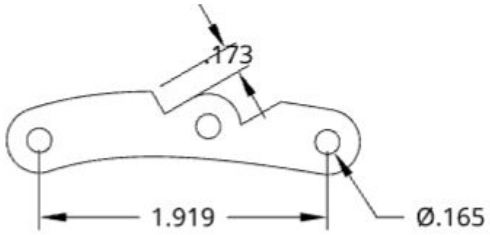
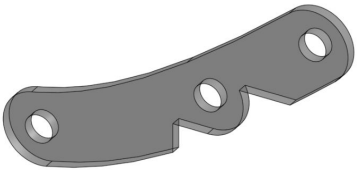
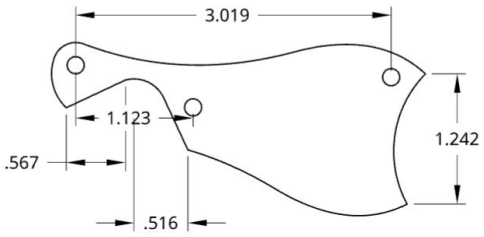
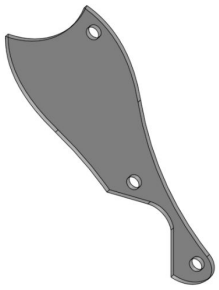
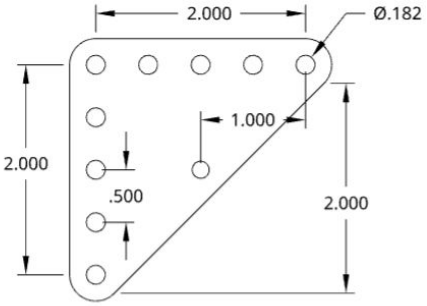
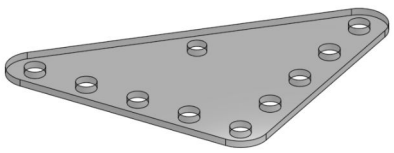
Item	LS Pillow block	1/16" Spacer	Washer	2" Standoff	1 ½" Standoff	⅞" Standoff
Part ID	276-2016	Robosource	275-1025	276-2013	276-2013	276-2013
Illustration						
Quantity	2	12	2	8	8	10

Item	3" Standoff	1 ½" Screw	1" Screw	½" Screw	⅜" Screw	Nylock nut
Part ID	276-2013	276-4998	276-4996	276-5007	276-4991	275-1027
Illustration						
Quantity	2	8	8	68	24	40

INTAKE 1.0

PLAN

CUSTOM PLASTIC

Item	Drawing	Illustration	Quantity
P1			4
P2			4
P3			4
P4			4
P5			2

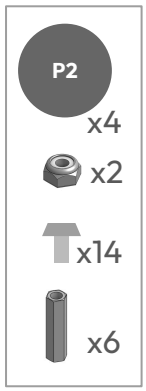
INTAKE 1.0

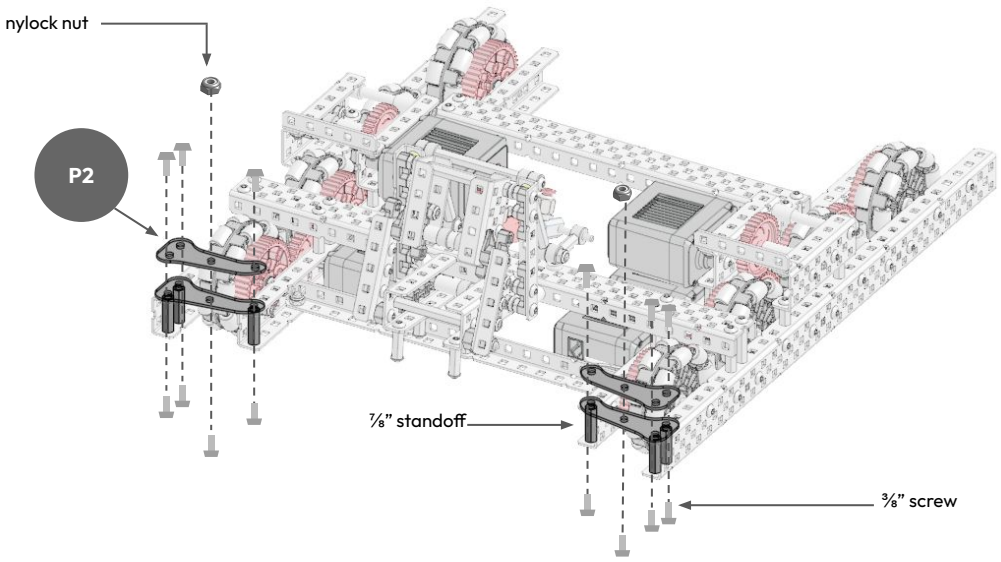
AUTHOR; RACINE LIU

DATE; 10.17.2024

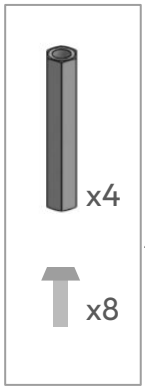
STEP BY STEP

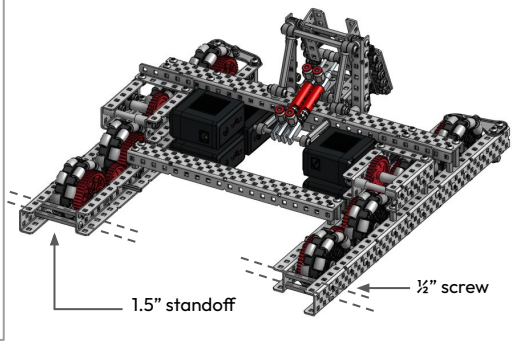
STEP 1:



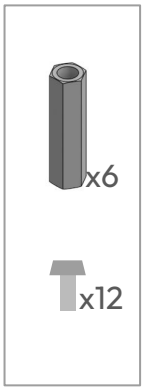


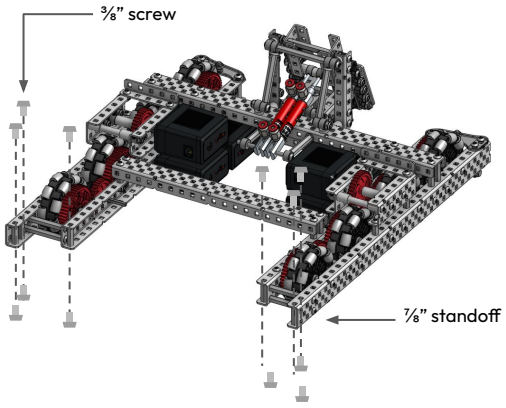
STEP 2:



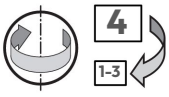


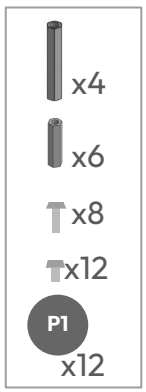
STEP 3:

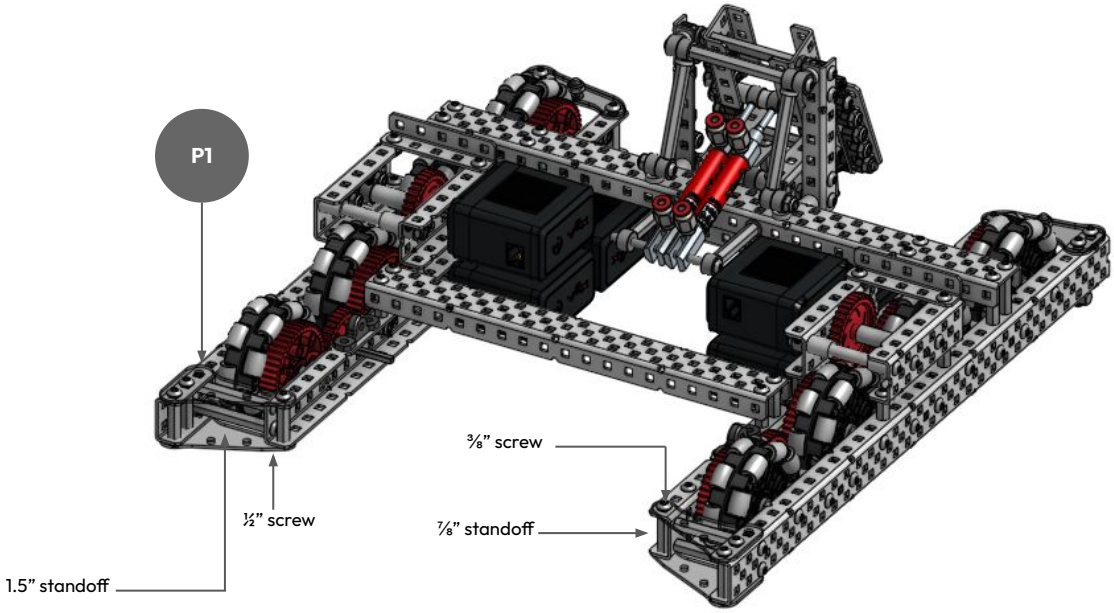




STEP 4:







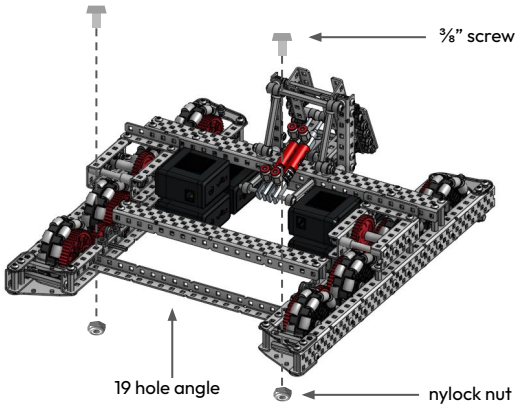
AUTHOR; RACINE LIU

DATE; 10.17.2024

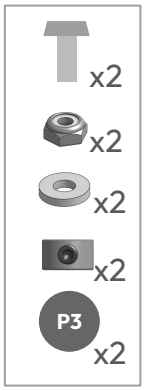
INTAKE 1.0

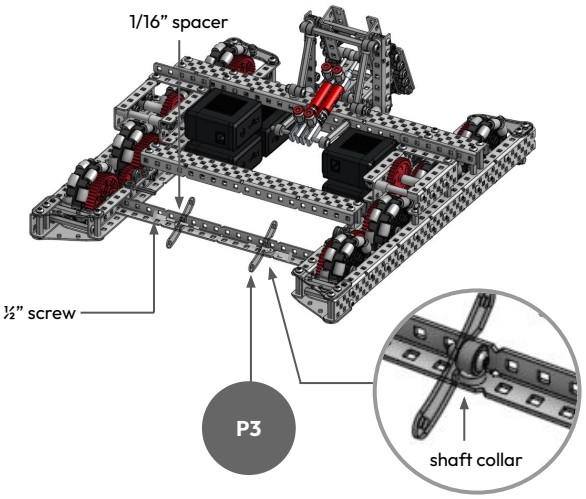
STEP BY STEP

STEP 5:

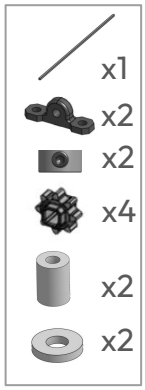


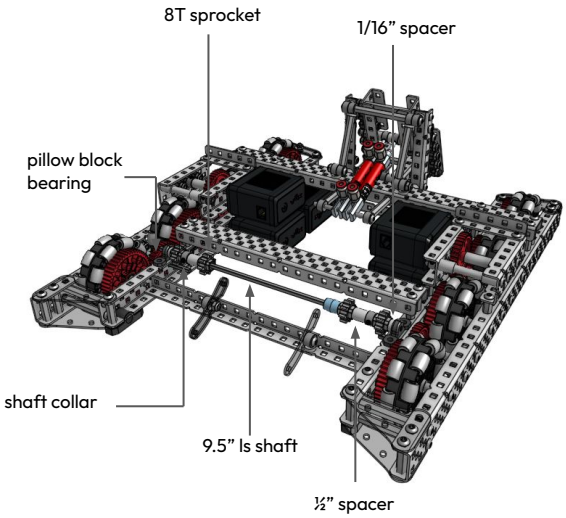
STEP 6:



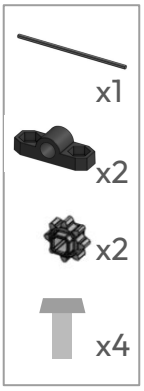


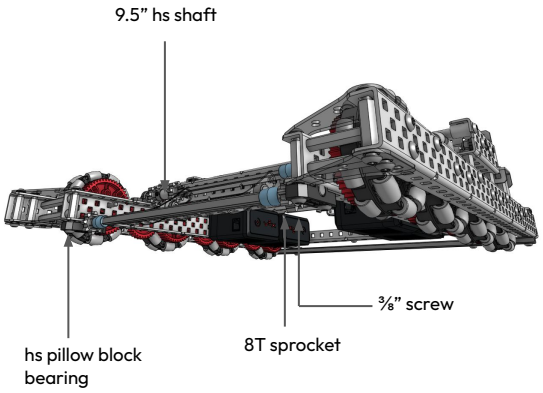
STEP 7:



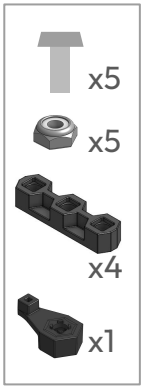


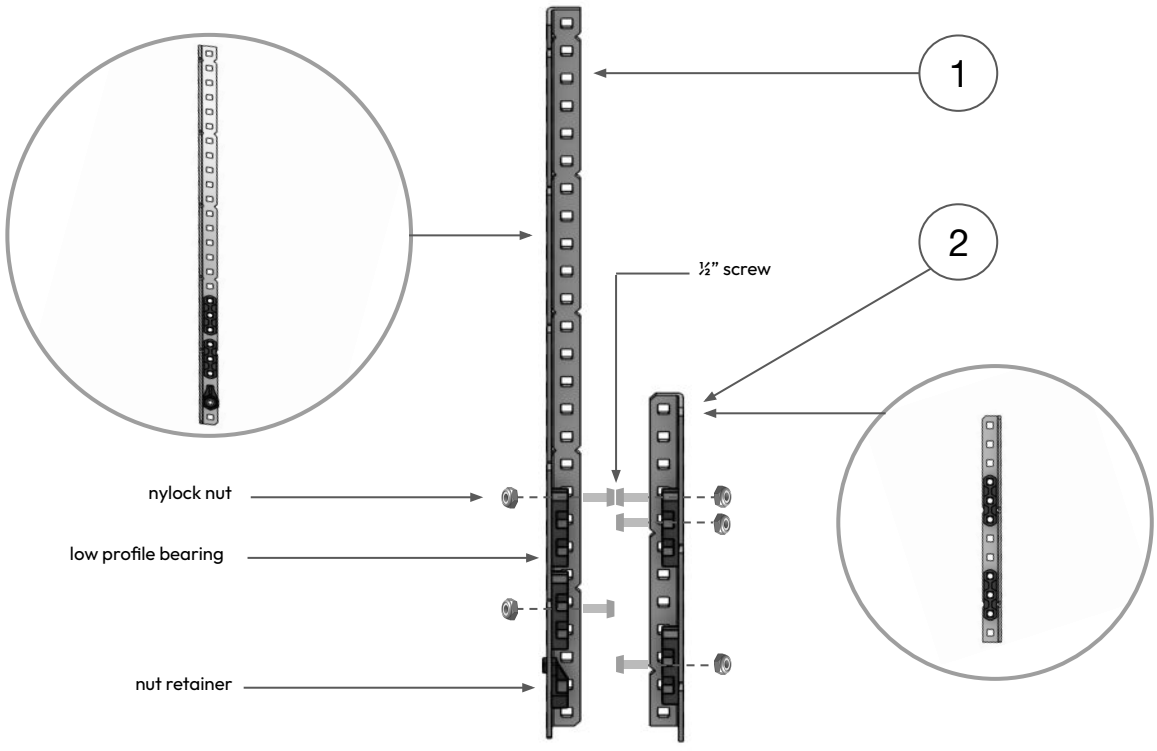
STEP 8:





STEP 9:





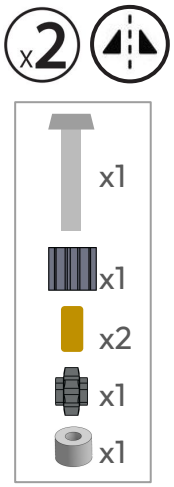
INTAKE 1.0

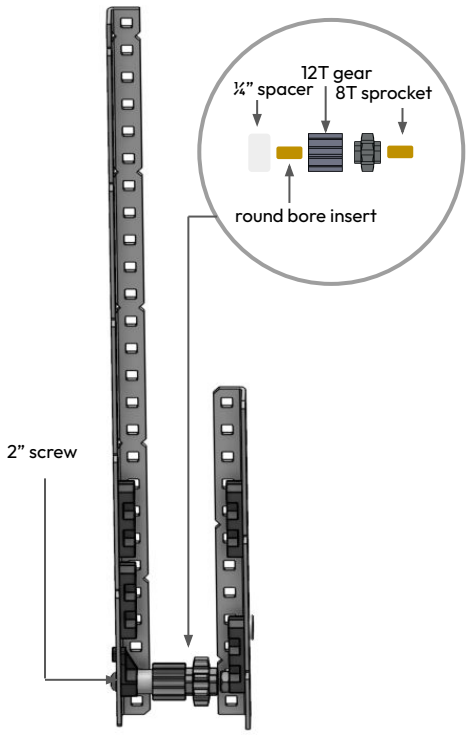
AUTHOR; RACINE LIU

DATE; 10.17.2024

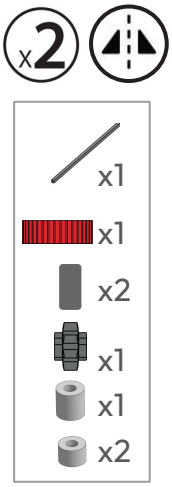
STEP BY STEP

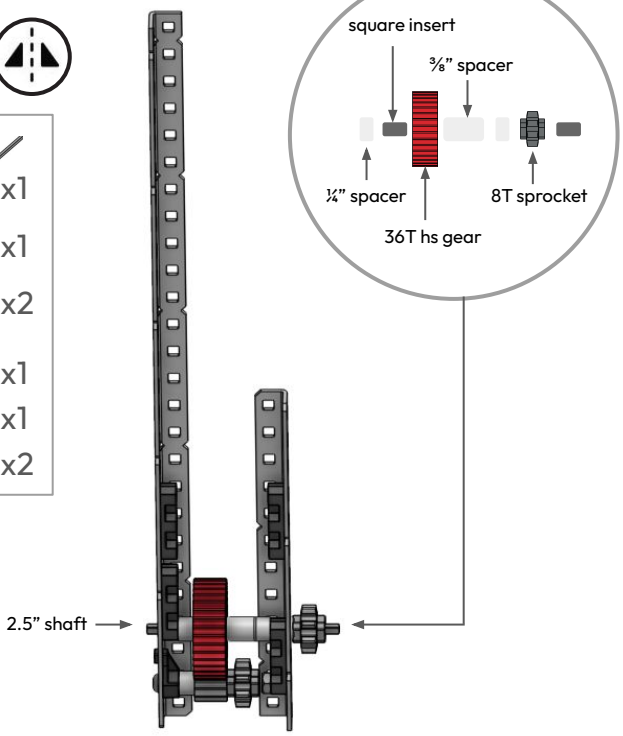
STEP 9:



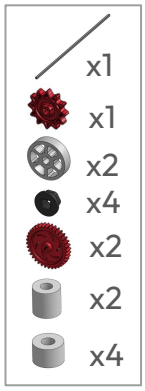


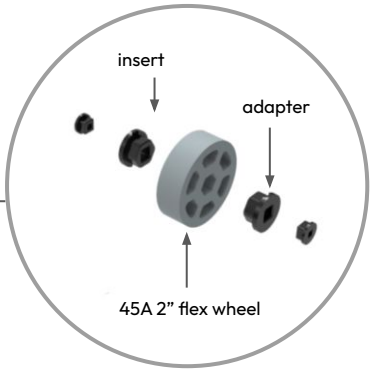
STEP 10:





STEP 11:





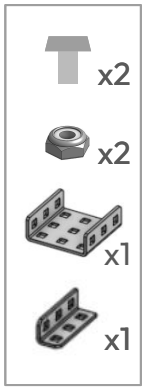
AUTHOR; RACINE LIU

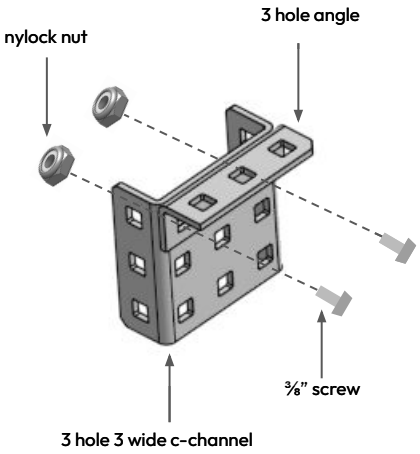
DATE; 10.17.2024

INTAKE 1.0

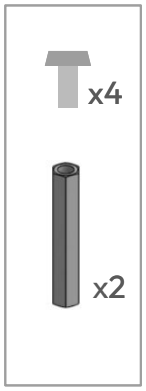
STEP BY STEP

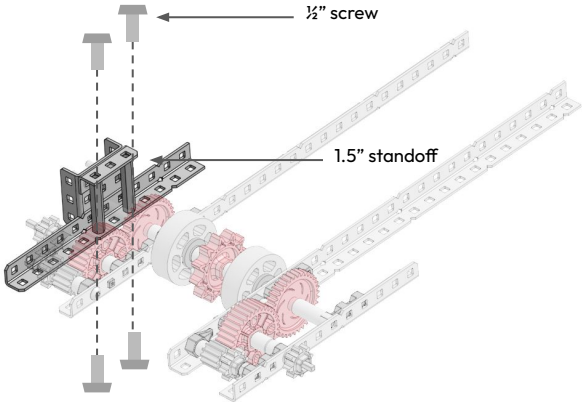
STEP 12:



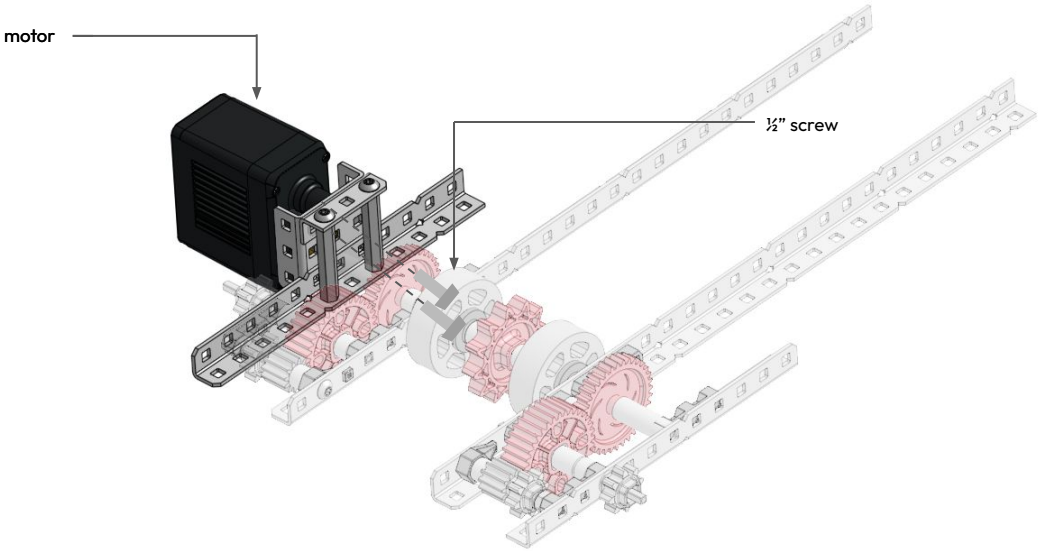


STEP 13:

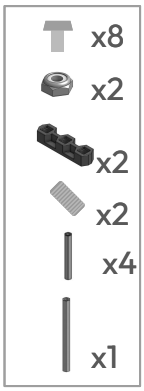


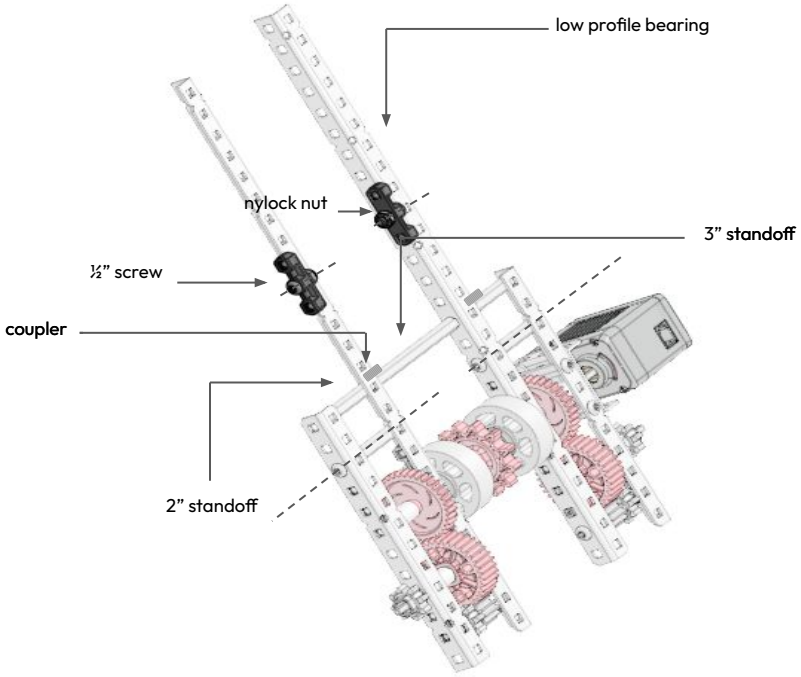


STEP 14:



STEP 15:





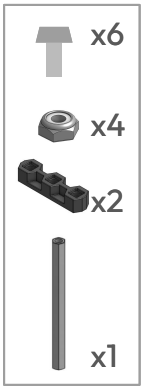
INTAKE 1.0

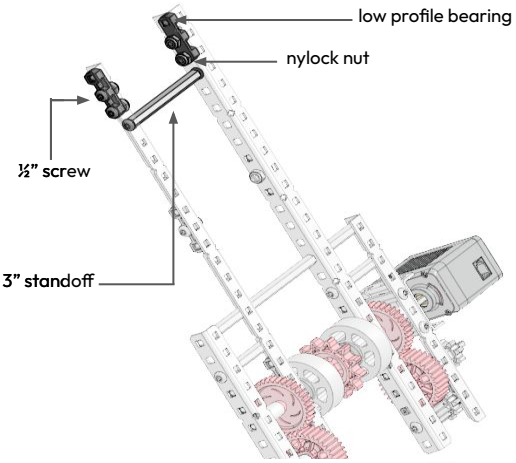
AUTHOR; RACINE LIU

DATE; 10.17.2024

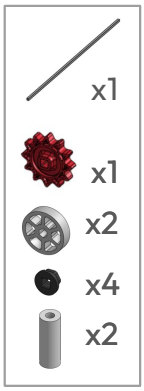
STEP BY STEP

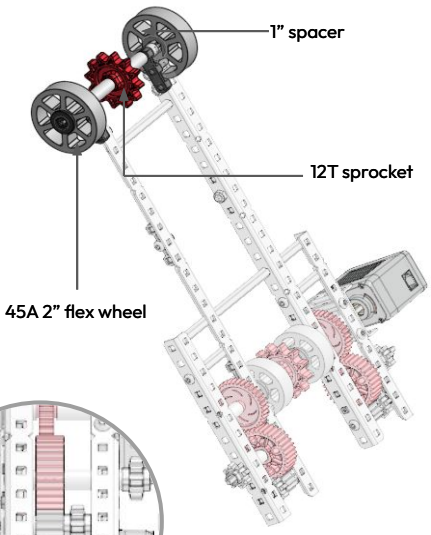
STEP 16:



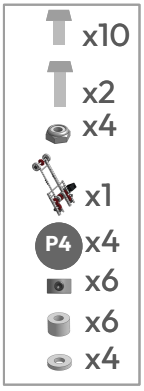


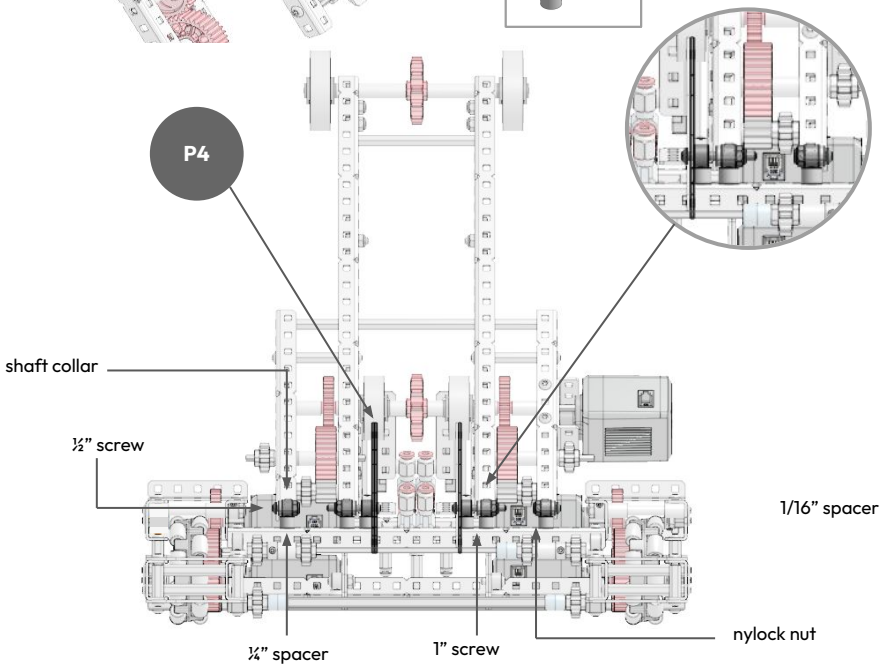
STEP 17:



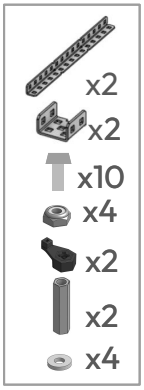


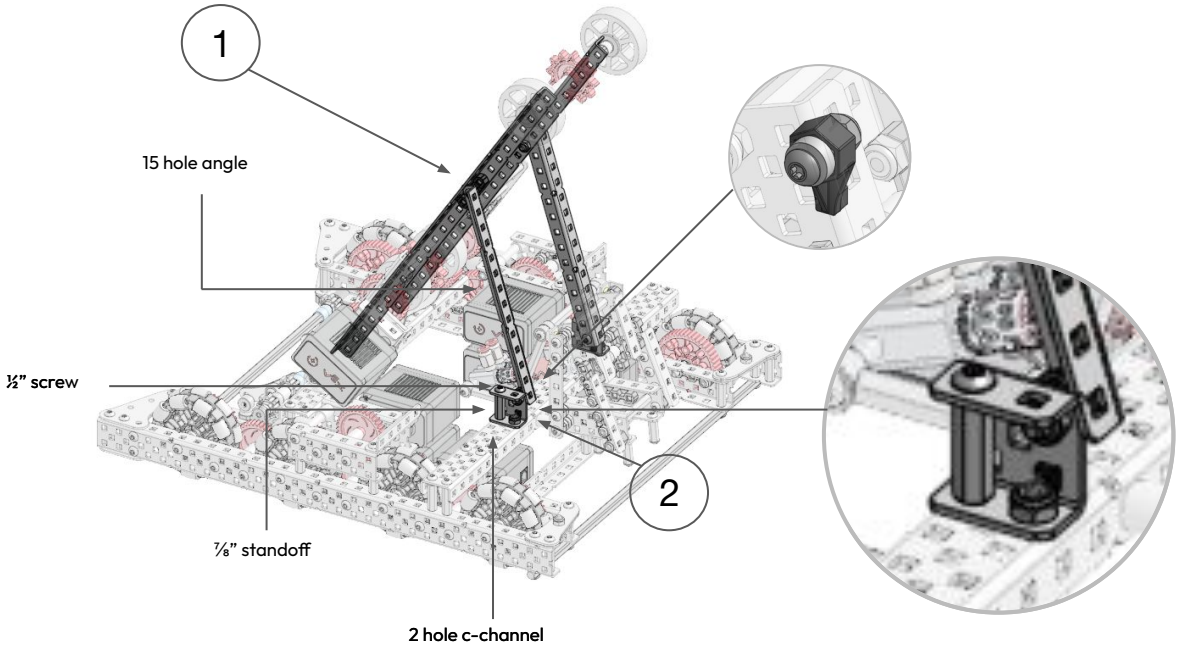
STEP 18:





STEP 19:





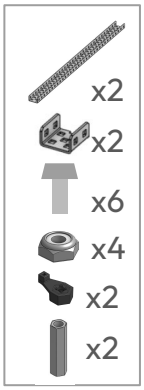
AUTHOR; RACINE LIU

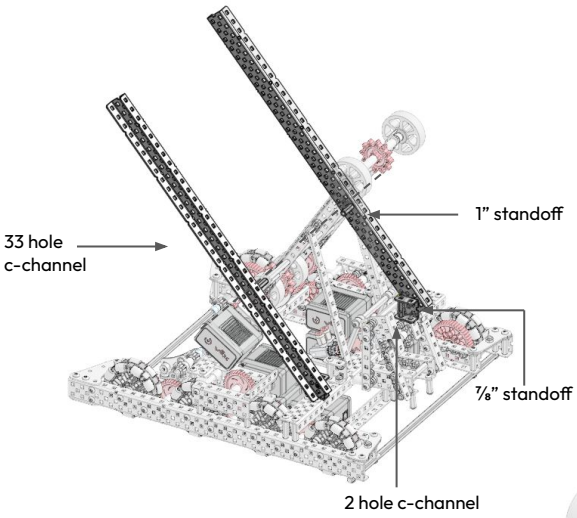
DATE; 10.17.2024

INTAKE 1.0

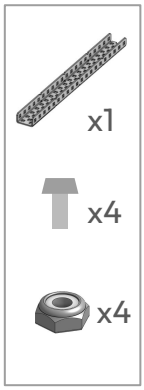
STEP BY STEP

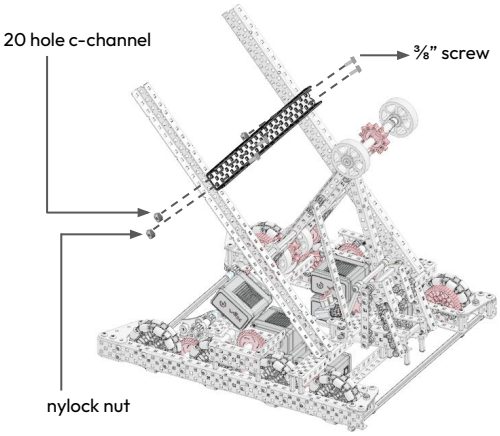
STEP 20:



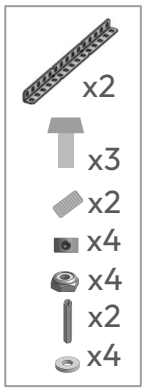


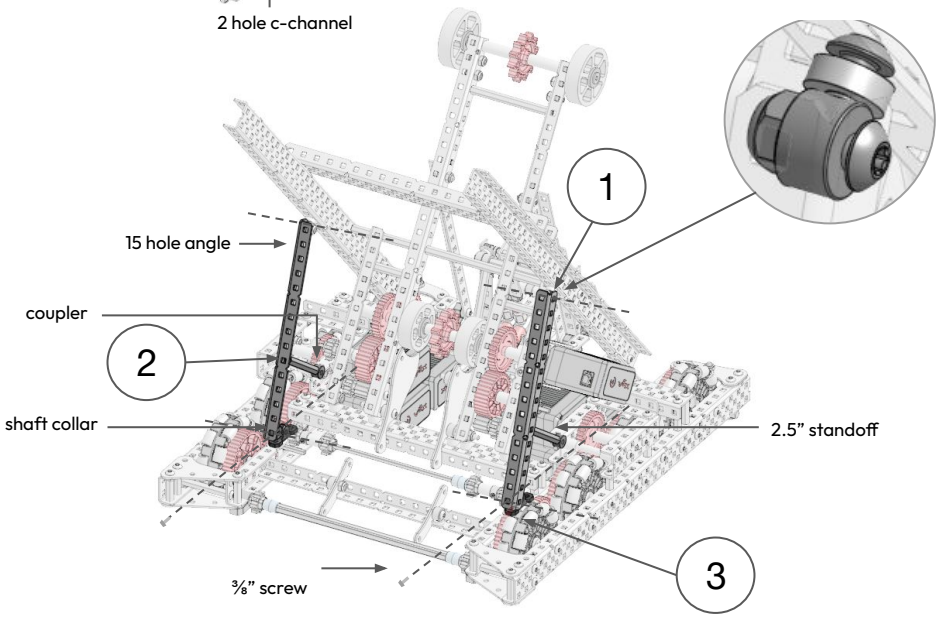
STEP 21:



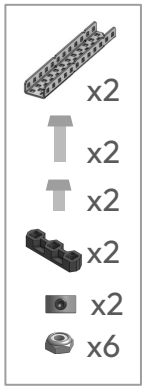


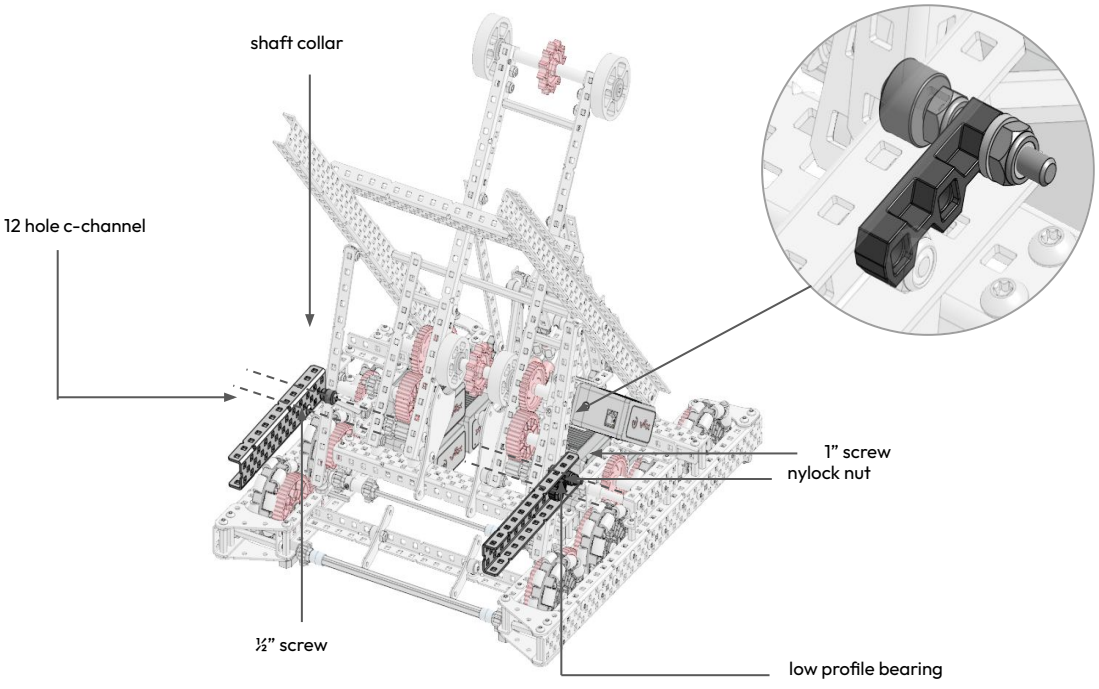
STEP 21:





STEP 22:





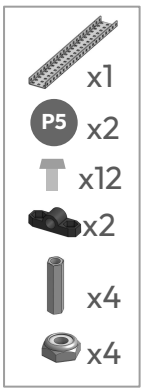
INTAKE 1.0

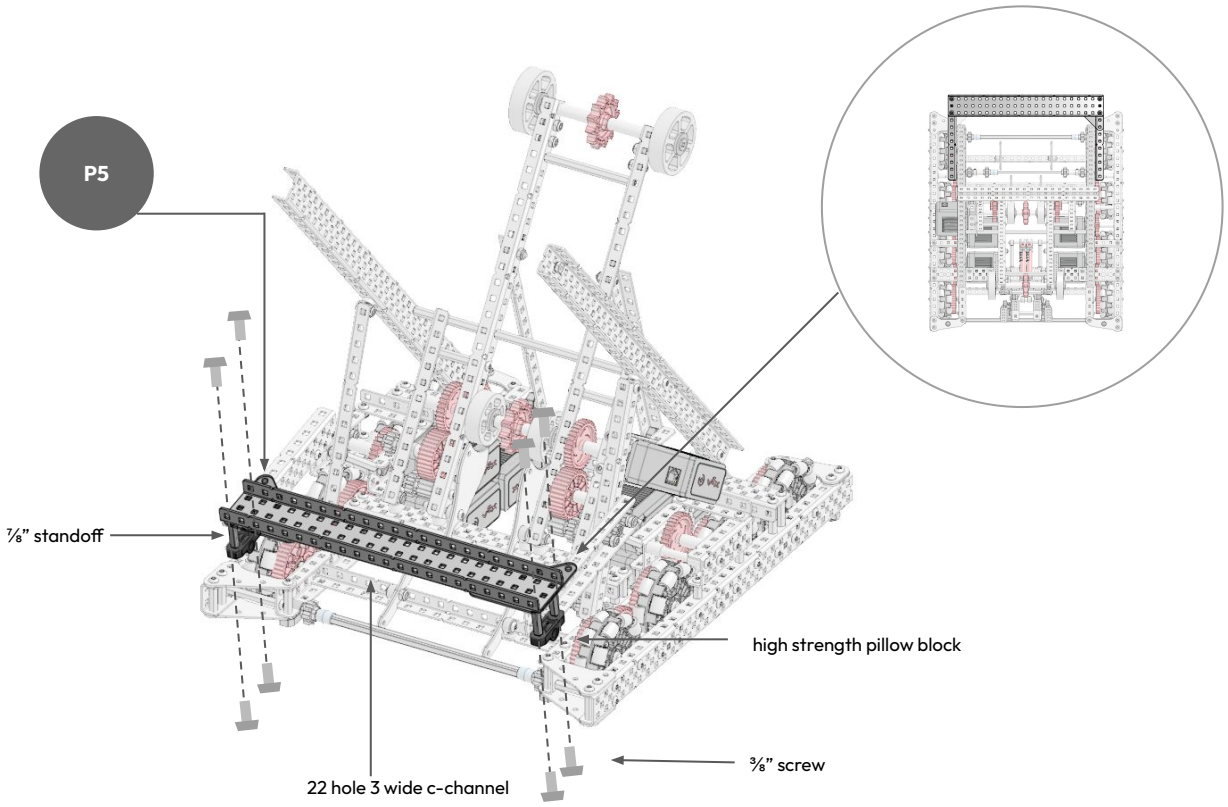
AUTHOR; RACINE LIU

DATE; 10.17.2024

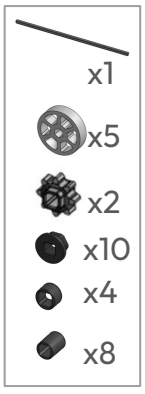
STEP BY STEP

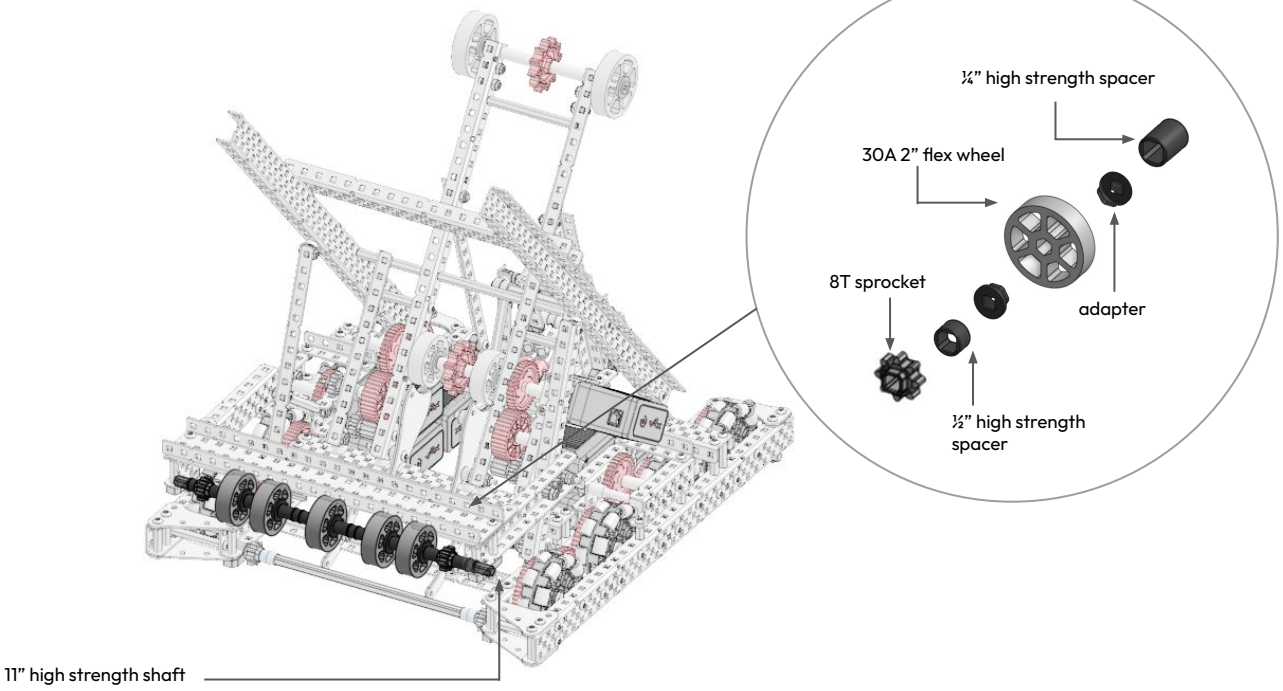
STEP 23:





STEP 24:





AUTHOR; RACINE LIU

DATE; 10.18.2024

WALL MECHANISM 1.0

DEFINE AND CONSTRAINT

IMPORTANCE

Wall stake mechanisms are essential in games where securing top rings are a priority. Being able to score them gives us a significant advantage over other teams that cannot, as it is 6 points if we dominate both neutral stakes. It gives us a significant advantage if we are unable to secure the positive corner and helps further

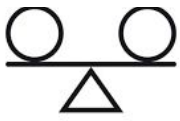
CONSIDERATIONS

- How will the mechanism ensure consistent alignment of objects with the pole to prevent misplacement or missed scoring opportunities?
- What design elements will prevent objects from slipping or dislodging while placing on the field
- Is your mechanism designed to handle high-speed object intake and placement without compromising accuracy and stability?

SETTING REQUIREMENTS

Precision - "How will the mechanism ensure consistent alignment of objects with the pole to prevent misplacement or missed scoring opportunities?"
Ensuring precise alignment with the pole is critical to avoid missed scoring opportunities and wasted time. A mechanism that reliably positions objects will maximize your score and minimize errors during fast-paced gameplay.

Stability – "What design elements will prevent objects from slipping or dislodging?"
Stablility is crucial when arm is raised, a well-designed system prevents the rings from wobbling or falling, maintaining your progress and securing points.



Speed – "How quickly can the clamp grip and release a mobile goal?"
Speed is essential in competitive robotics, but rushing can lead to misaligned or unstable stacks. Balancing speed with control allows you to maximize scoring without sacrificing accuracy or risking mechanical errors.

CONSTRAINTS

RULE CONSTRAINTS:

As per <R4> the clamp must fit within an 18" x 18" x 18" volume and must not expand over 24" in one direction

MATERIAL CONSTRAINTS:

We have full access to all potential materials as it is the start of the season

TIME CONSTRAINTS:

This subsystem should be completed by October 25th

WALL MECHANISM 1.0

AUTHOR; RACINE LIU

DATE; 10.18.2024

BRAINSTORM

SOLUTIONS

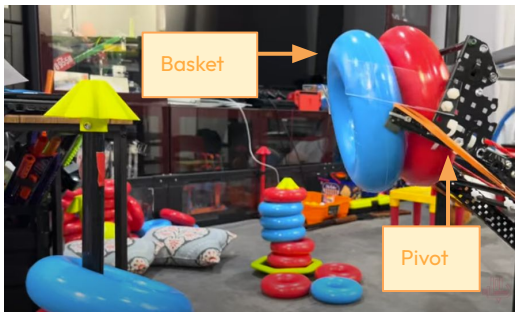
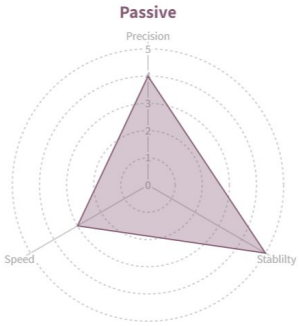
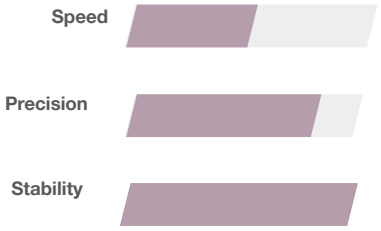


Image courtesy to Team 10955M - VEX High Stakes 10955M MJS Robotics First V-Load - Released September 11 2024 on youtube. Timestamp: 0s

PASSIVE

Pros: Good angling, simply drive into the pole
Cons: Time consuming to engineer, redirect from hook is slow





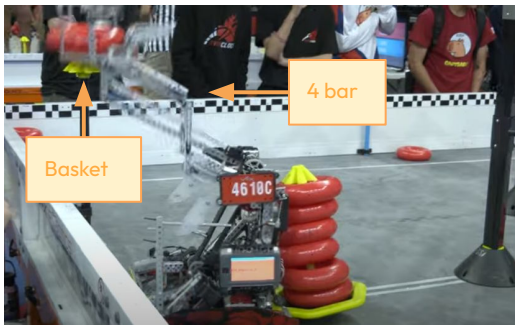
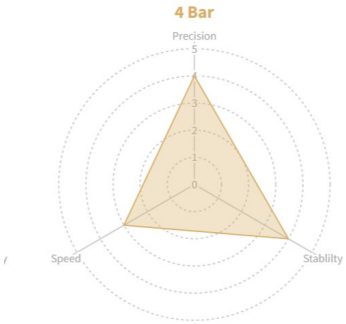
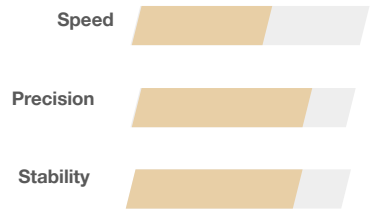


Image courtesy to Team 4618C pits and parts - Released September 24, 2024 on YouTube, Timestamp: 2.59s

4 BAR ARM

Pros:Sturdy, good alignment
Cons: redirect from hook slow, heavy





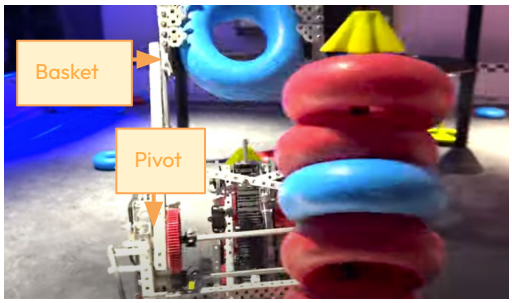
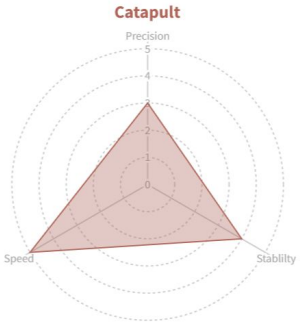
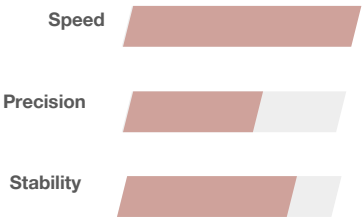


Image courtesy to Team18522R | Vex High Stake reveal【New Start Robotics】 Released August 14 2024 on youtube. Timestamp 30s

CATAPULT ARM

Pros: Fast, customizable. Simple mechanism
Cons: Difficulty aligning



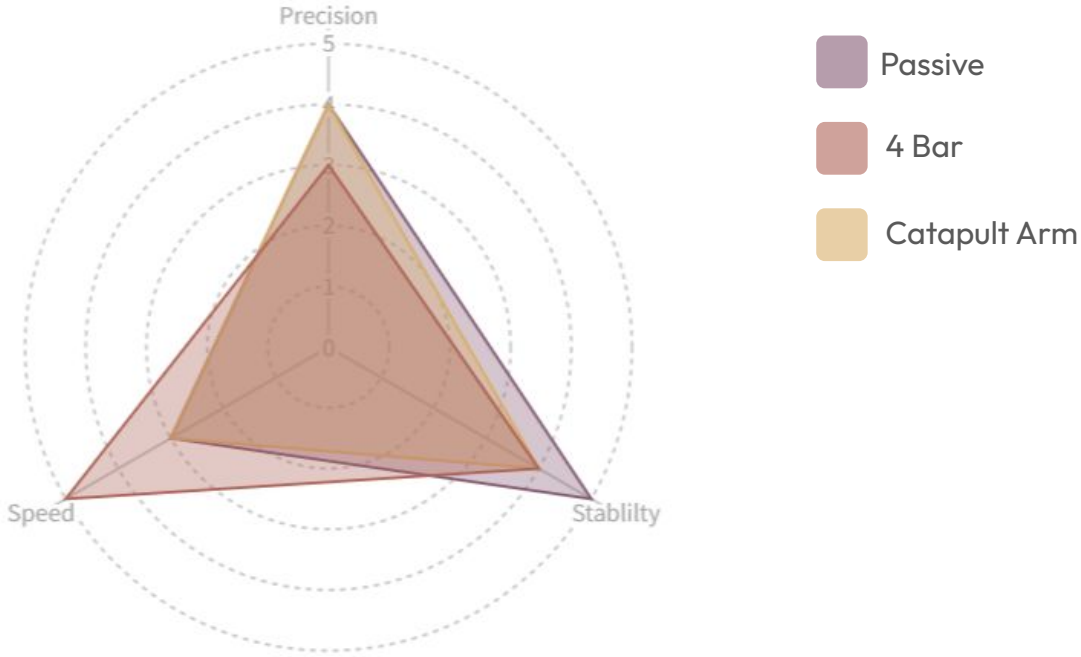


WALL MECHANISM 1.0

EVALUATE

RAW RANKING VISUALIZATION

When viewing the raw visualization, it seems unclear of the majority.



RANKING

We rank each type of intake on a scale of 1-5 with 5 being the highest and 1, the lowest. Weighting will also be given in terms of what we think is most important by increments of 2. We then total the ratings to get an idea of the highest score.

Criteria	Weight	Passive		4 Bar		Catapult arm	
		Rating	**Total**	Rating	**Total**	Rating	**Total**
Speed	5	3	15	3	15	5	25
Precision	3	4	12	4	12	3	9
Stability	1	5	5	4	4	4	4
Total		32		31		38	

After observing the raw ranking and the decision matrix, we have concluded to create a **catapult arm**

ARM 1.0

PLAN

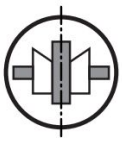
AUTHOR; RACINE LIU

DATE; 10.18.2024

SPECIAL SYMBOLS


Make sure parts can rotate freely


Make sure gears are properly engaged

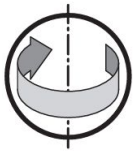

Assemble symmetrically


Special attention in assembly

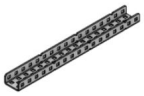
Complete 2 pieces

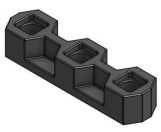
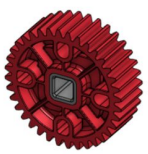
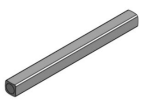
Assemble step 3 according to step 1


Assemble mirrored


Flip

PARTS LIST

Item	5.5W Motor	1x2x1x26 C-channel	1x2x1x17 C-channel	C-channel coupler	¼" Spacer	1/16" Spacer
Part ID	276-4842	276-2289	276-2289	276-2575	276-6341	Robosource
Illustration						
Quantity	2	2	1	2	2	4

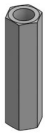
Item	Low profile bearing	12T Gear	36T HS Gear	1 ½" Shaft	Nylock nut	Round bore insert
Part ID	276-8023	276-2169-001	276-7747	276-1149	275-1027	276-8034
Illustration						
Quantity	2	2	2	2	4	2

AUTHOR; RACINE LIU

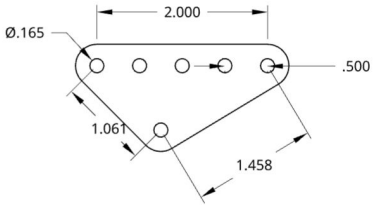
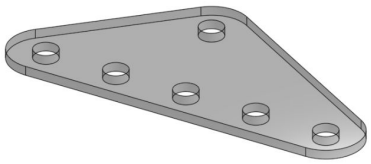
DATE; 10.18.2024

ARM 1.0

PLAN

Item	6” Standoff	1” Standoff	1 ½” Screw	½” Screw	⅜” Screw
Part ID	276-2013	276-2013	276-4998	276-5007	276-4991
Illustration					
Quantity	1	5	2	6	12

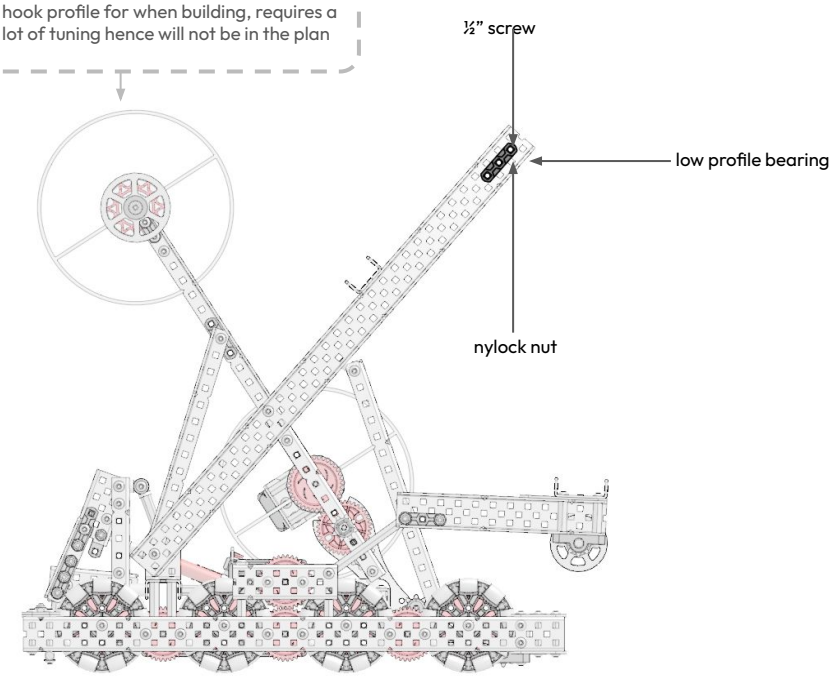
CUSTOM PLASTIC

Item	Drawing	Illustration	Quantity
P1			4

STEP BY STEP

STEP 1:

This plastic piece is a visualization for the hook profile for when building, requires a lot of tuning hence will not be in the plan



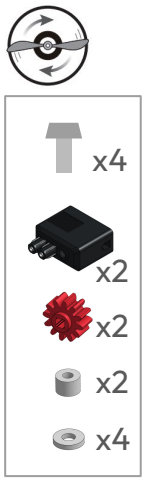
ARM 1.0

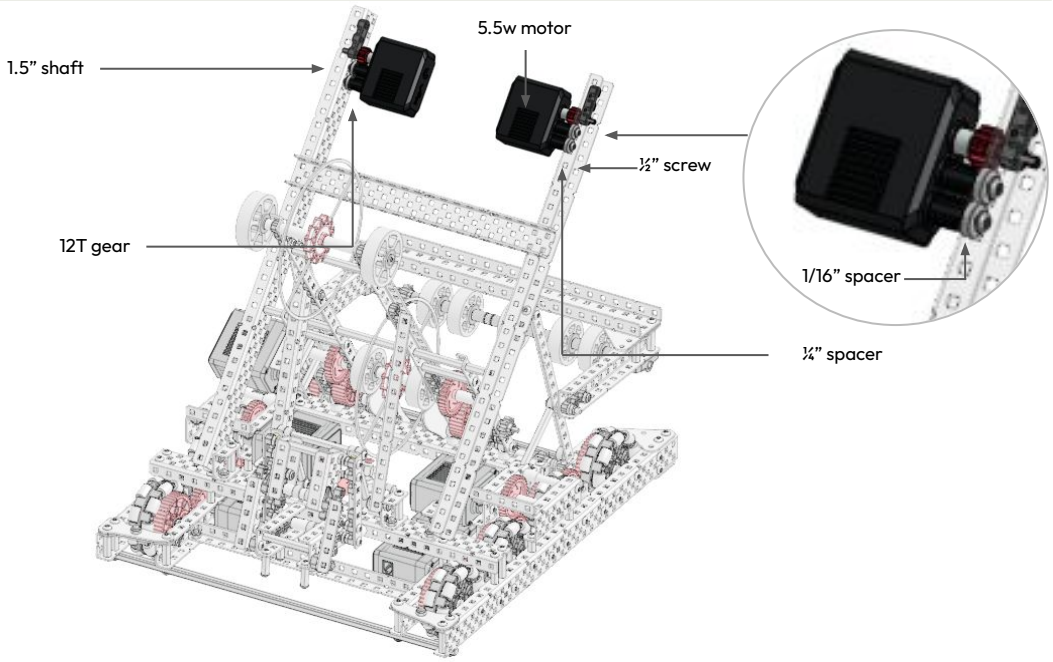
AUTHOR; RACINE LIU

DATE; 10.18.2024

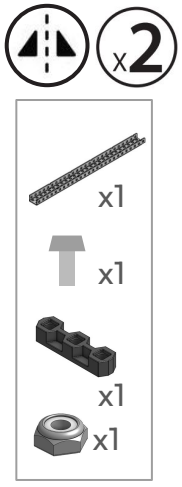
STEP BY STEP

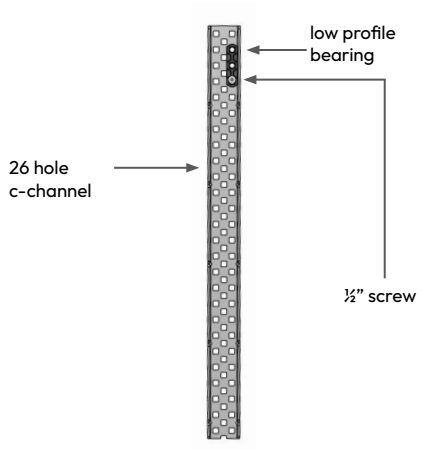
STEP 2:



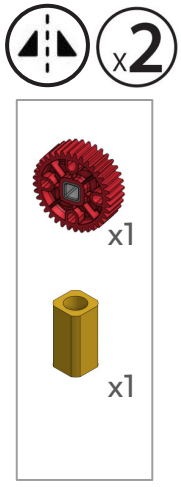


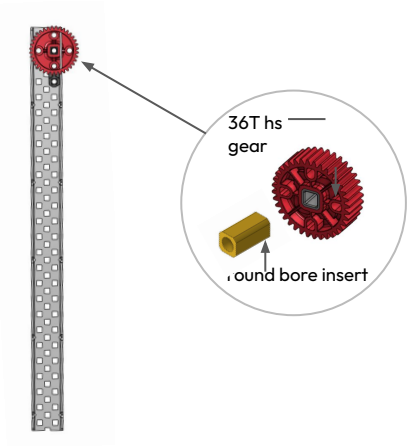
STEP 3:



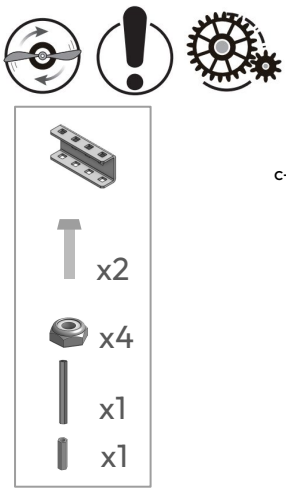


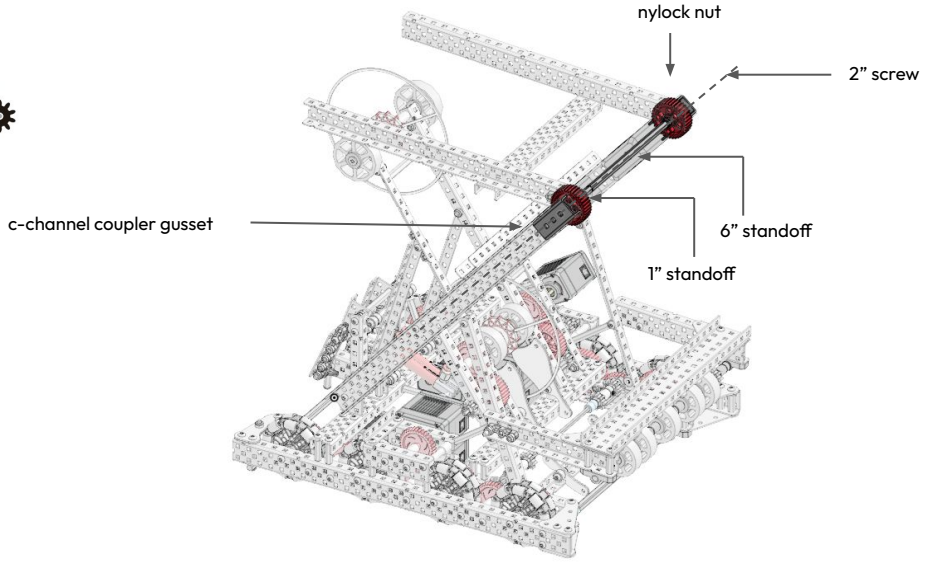
STEP 4:



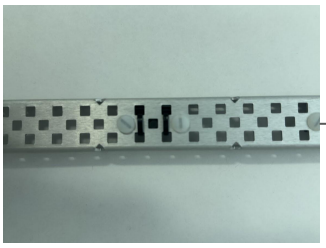


STEP 4:

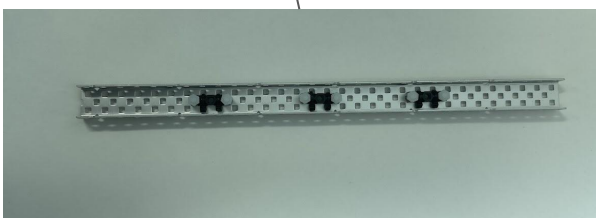


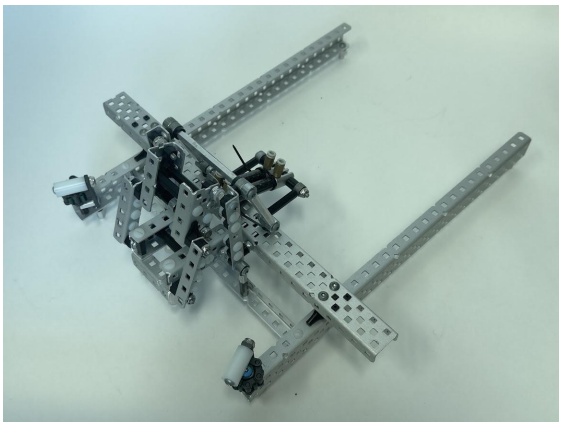


CLAMP + INNER FRAME



We've chosen to use zip ties as well as plastic screws in order to save on weight. Compared to metal screws, plastic screws have a drawback as they can get loosened easily as well as being more susceptible to breaking. Having zip ties there area is an added measure of security

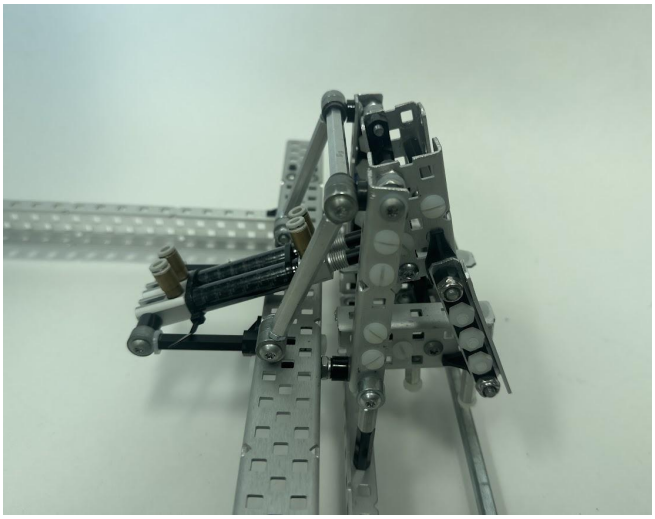




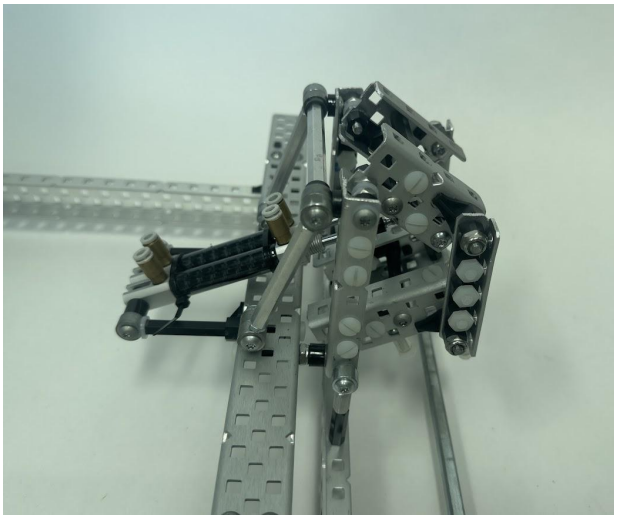
To the right is an image of the inner frame completely assembled along with the clamp. Careful measures are taken to ensure the inner frame is square as it will imact our entire robot builf

A High Strength shaft is drilled so that it can be used as the bottom brace of the frame, allowing the mobile goal to tilt up. First, the holes are marked then drilled. Ensure a slow speed in drilling so that the shaft does not break

Closed clamp



Open clamp



CLAMP 1.0

TEST

AUTHOR; RACINE LIU

DATE; 10.24.2024

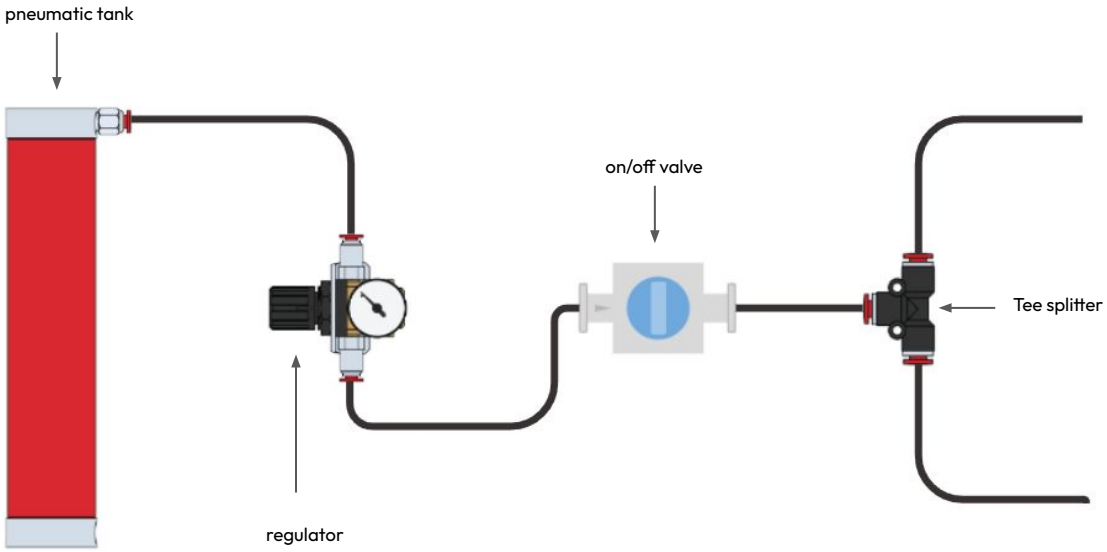
ACTUATION TEST

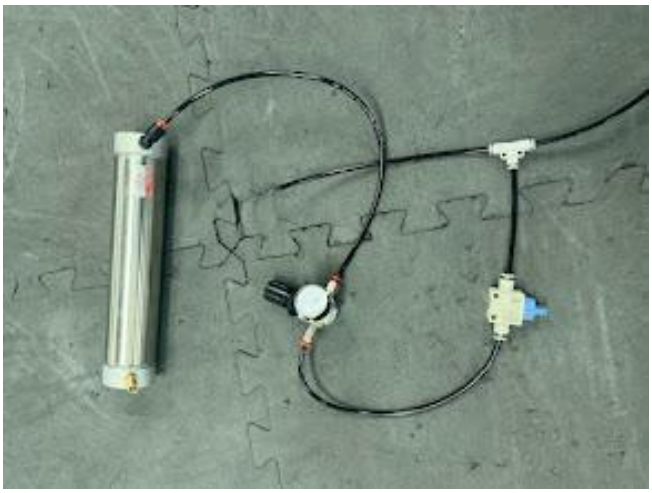
Requirements:
We want to see how many actuations the clamp can handle in a game

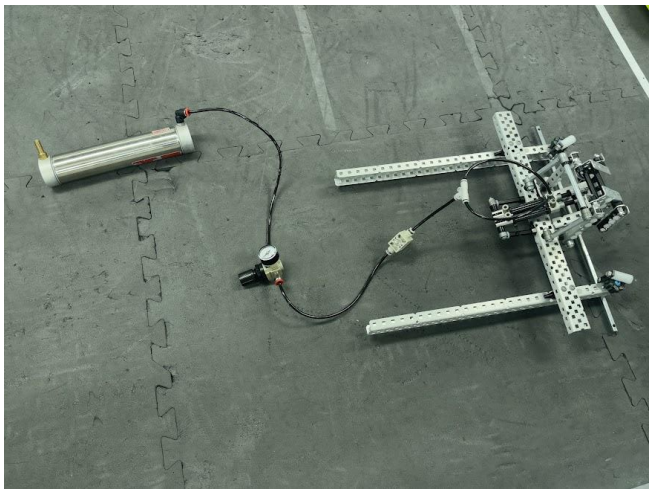
Materials:
Clamp, air tank, tubing, regulator, on/off valve and an air pump

SETUP

Setup pneumatic system:







The 2 remaining open ends of the tubing are to be plugged into the 2 open ends of the single acting pistons.

AUTHOR; RACINE LIU

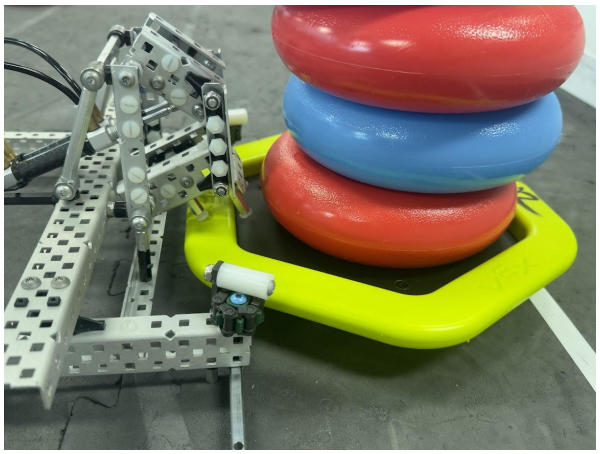
DATE; 10.24.2024

CLAMP 1.0

TEST

PROCEDURE

1. Fill up a mobile goal with 6 rings, as it'll be the max weight possible achieved in the game
2. Place the mobile goal in clamping position
3. Fill up the air tank with 50psi
4. Turn the on and off valve repeatedly so that the clamp repeatedly clamps onto the goal
5. Count the amount of full actuations achieved



RESULTS

Quantitative result: The clamp can be actuated 25 times

Qualitative result:
After the 25 consistent pickups, the clamp slows down. Then slight movement is required to hoold it up. Once held up, the mobile goal remains lock. Hence the clamp can be actuated a further 5 more times successfully.

As the pistons are single acting, dropping the mobile goal remains the same throughout the test.

CONCLUSION

The clamp can be actuated 30 times in game

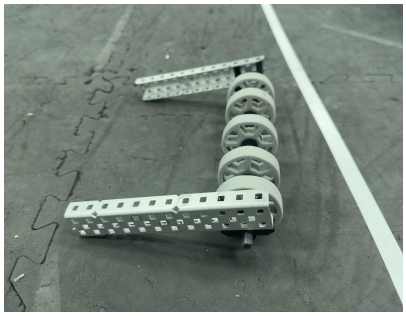
It's likely that it will not be used that many times within the 1 minute 15 second game which will allow us to expend more air from this tank to other future mechanisms.

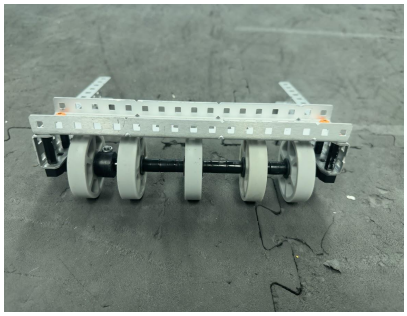
INTAKE 1.0

BUILD

AUTHOR; RACINE LIU

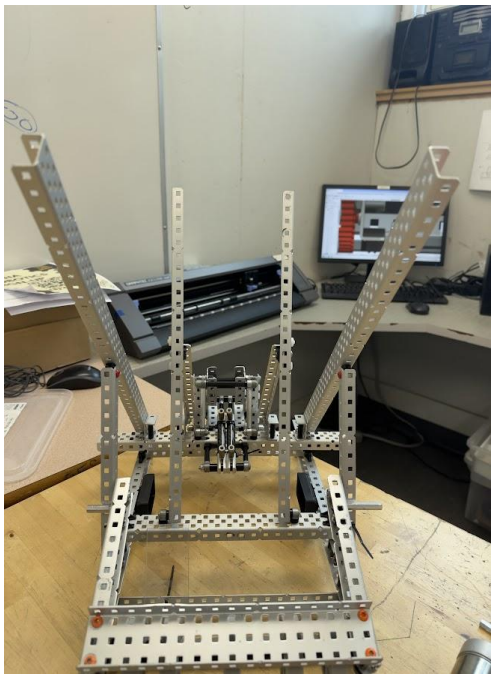
DATE; 10.24.2024



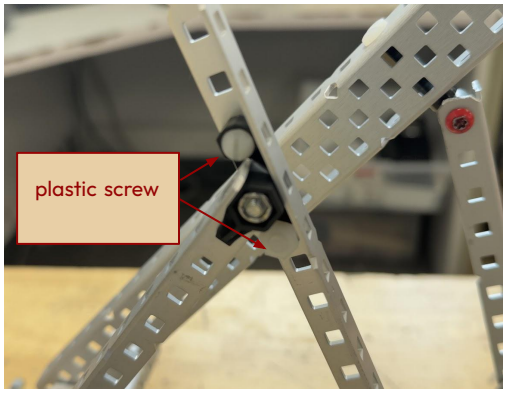


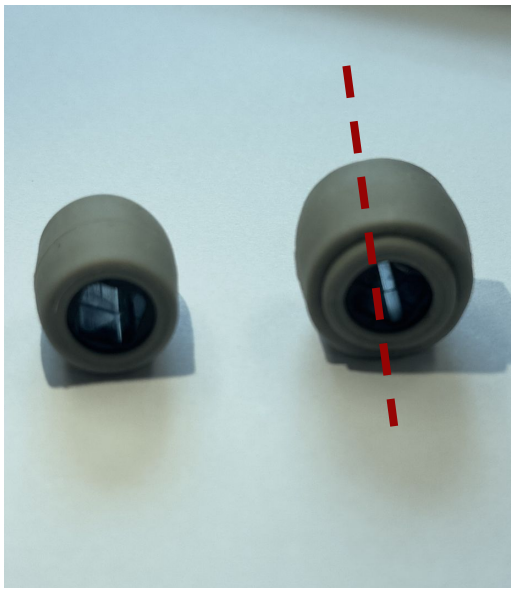
1st floating stage of the intake. 30a 1" flex wheels were chosen as they are the most flexible hence the most grip, as well as the smallest size allowing tor easier manipulation og the rings.

Robot structure mounting, plastic screws are preferred to be used anywhere. For mounting the structure the main screws holding them attached need to be metal due to their strength.
Plastic screws are used for the other ends of the bearings in this instance



When assembling the structure, every component needs to be as tightened as possible as other mechanisms will be relying on them to remain secure in order to be consistent in action





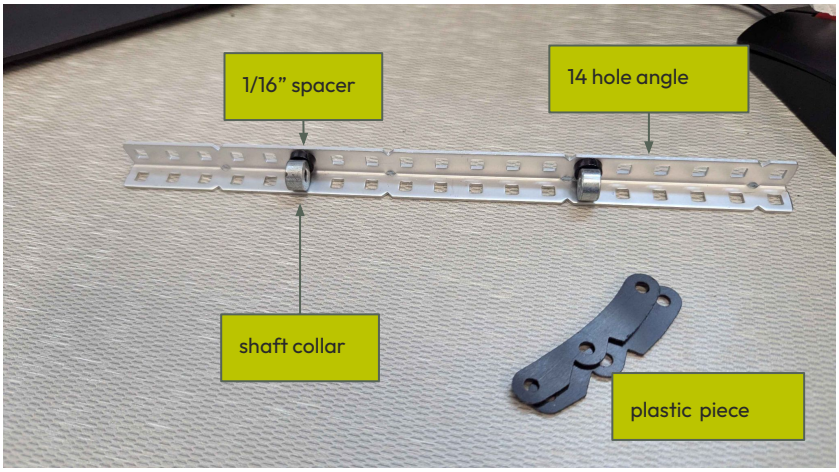
Counterrollers help intake the rings from the bottom and with the guidance of the floating stage, the process of intaking the rings will be fast.

We took apart an old 4" omni wheel for it's rollers. Newer omni wheels have a thinner hole hence are unable to be used. After taking the rollers, high strength spacers are then placed inside so that they can fit onto a shaft.

INTAKE 1.0

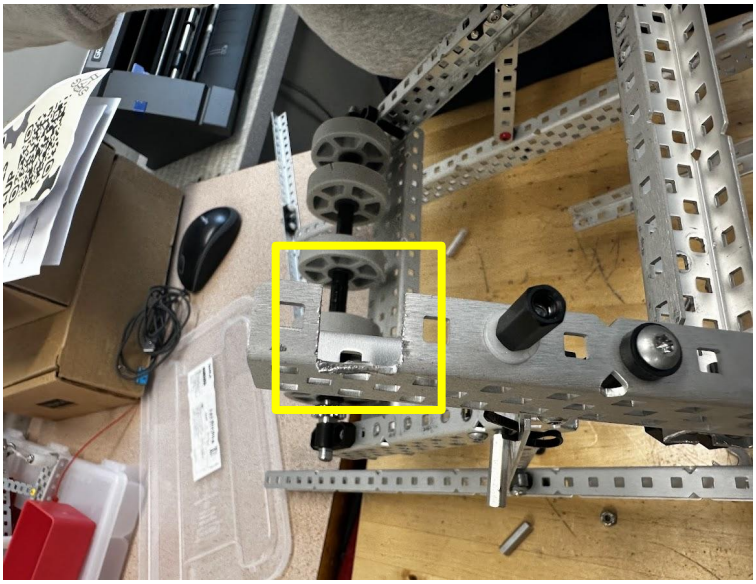
BUILD

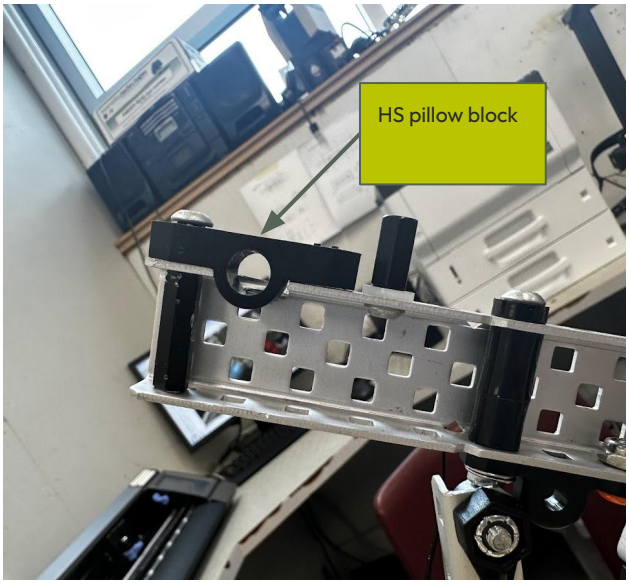
2 omni rollers are double stacked on the end so that the rings will have no chance of contacting the sprocket potentially disturbing the chain, note the square on the image to the right



When assembling the intake ramp, plastic pieces are printed from delrin via a laser cutter.

Differently from the cad, we decided to double stack the custom plastic parts from both sides allowing for more strength in the components. This means less chance of breakage, again reiterating the need for reliability in parts.





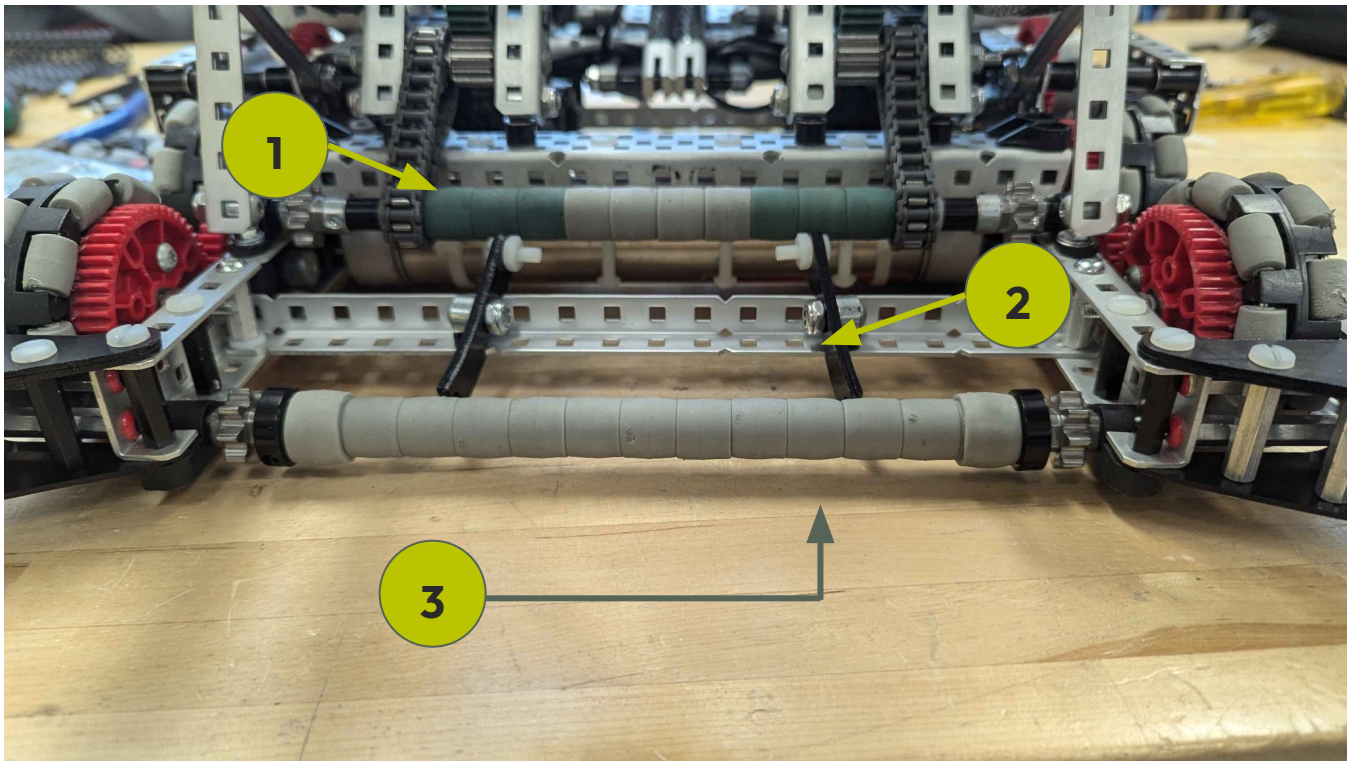
As we have low clearance, mounting the pillow block right way down will result in the counterrollers making contact with the ground. To the upper left is an image of the structure flipped and the 2nd hole of the end of the inner brace cut out, this allows for the mounting of the pillow blocks in a way where the counterrollers will be able to spin freely when mounted

INTAKE 1.0

BUILD

AUTHOR; RACINE LIU

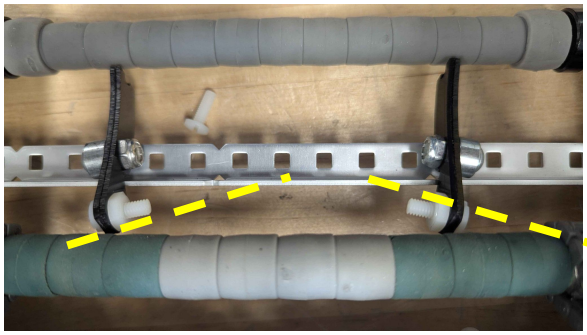
DATE; 10.24.2024



The 2 sets of counterrollers and the ramp are mounted to the robot. Note that there are 2 chains on both sides of each counterroller. We've chosen to add both to prevent the likelihood of the intake chain snapping during games, rendering the intake non-functional.

In this case, when 1 chain snaps the other one will still allow the intake to spin.

Additional plastic screws are used to secure the ramp
unlike in the cad.



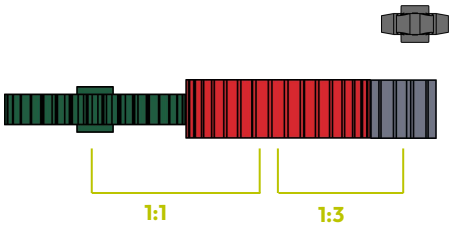
AUTHOR; RACINE LIU

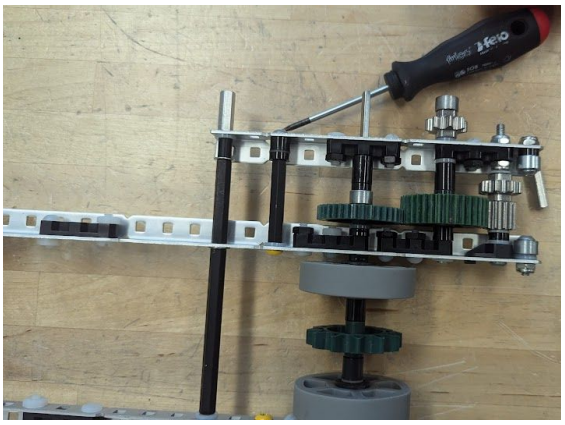
DATE; 10.24.2024

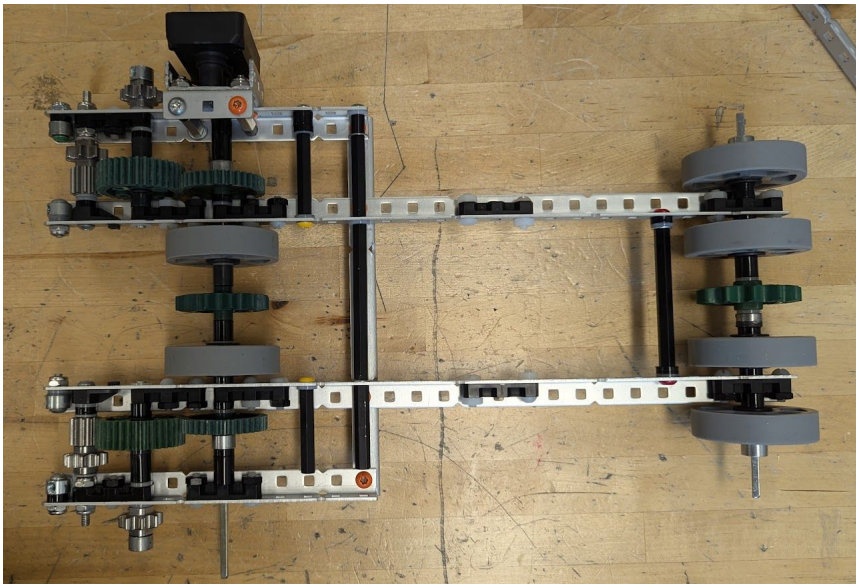
INTAKE 1.0

BUILD

The two 36T gears connected gear the intake to spin at 600RPM. The bottom 12T gear connected to the 36T gears spin the counterollers at a ratio of 1:3





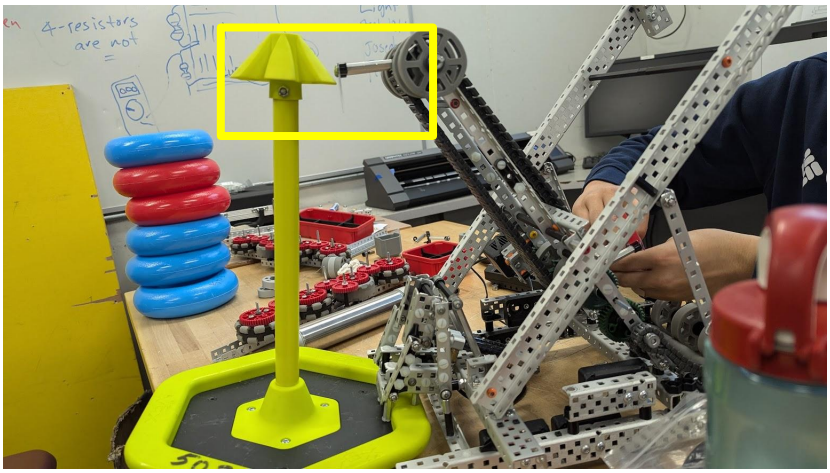


When building, we always check if the components can freely spin, the longer they can spin the less friction there is. It's a good measure of ensuring low friction, leading to more consistency.

Note that we have motor caps on with the motor unscrewed, all our motors are like this so they are able to be hot swapped if needed to quickly deal with overheating.

As the ring is 1.5" thick, the hooks are needed to be longer in order to allow pickup. We decided on 2" hooks to allow plenty of space for pickup.

Determining the amount of hooks is relatively straight forward, due to the ring size to the length of our conveyor running 2-3 hooks is a possibility. Hence we ran the max possible: 3

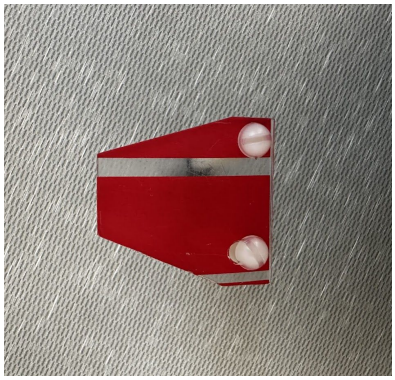


INTAKE 1.0

ITERATE

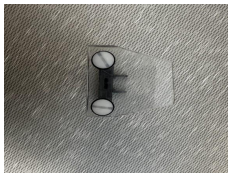
AUTHOR; RACINE LIU

DATE; 10.24.2024



Upon a quick test of running the intake at full speed, we realized there was a significant issue with friction as the hooks swung around.

Below are images of our initial hooks:


We realize when an object moves in a circular path, it requires a centripetal force *Fc* to keep it moving. This force depends on the mass *m* of the object (in this case, the hook), the radius r of the circular path, and the speed *v* at which it travels. The formula for centripetal force is:

$$F_c = \frac{mv^2}{r}$$

There are 2 issues in need of being addressed in order to combat the issue:

1. Since Fc is directly proportional to mass, a heavier hook means greater centripetal force. This increased force pulls the hook outward, intensifying contact (and thus friction) between the hook and the conveyor structure.

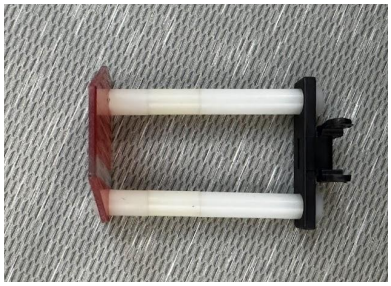
2. The hook being an unevenly distributed mass, which results in a variable moment of inertia as it rotates. This inconsistency in the weight distribution causes the hook to shake and vibrate as it rotates, leading to even more friction due to uneven contact points.

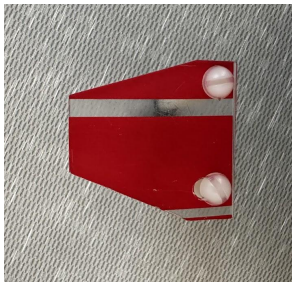
Our solution instead of combatting uneven distribution is simply just lowering *Fc* which reduces the outward pull and, therefore, the friction between the hook and the conveyor. To so is to simply decrease any variable in the numerator.

$$F_c = \frac{mv^2}{r}$$

Our speed for the intake is already set, lowering it in any way would be harmful towards game performance, hence the only variable we can make lighter is the mass. Hence the solution is to create lighter hooks

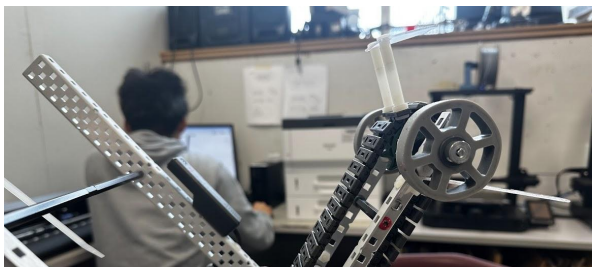
HOOK ITERATION 2

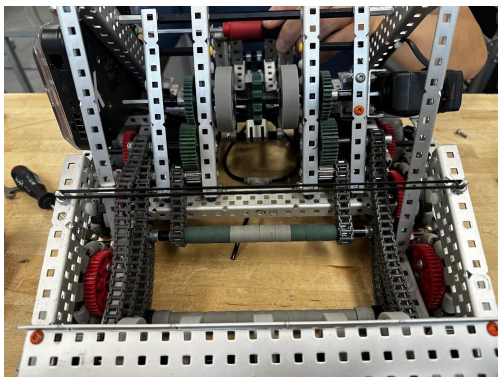




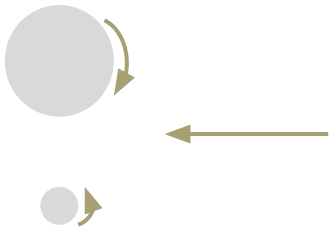
For our second version of the hook, we opted to swap out the heaviest component, standoffs, for plastic spacers resulting in significant changes in weight. Lowering friction, allowing for more consistent scoring.

We note that there may be issues in the hooks getting caught in the rings when the mobile goal is full. In order to combat this issue in the future, we cut out 2 polycarbonate strips the same length as the hooks when being spin around to push the rings away.





Then, we add all the chains required. Note the floating stage spins the opposite direction to the hook and counterrollers in order to intake the rings.



We decided to mount the battery to one side of the structure, using the longest possible battery cable to extend across the robot where the brain will be mounted on the other side. This way both the battery and the brain will be easily accessibly from both sides.

For the battery mount itself, we decided to used pre-provided battery clips as they allow for a tight hold as well as easy changes.

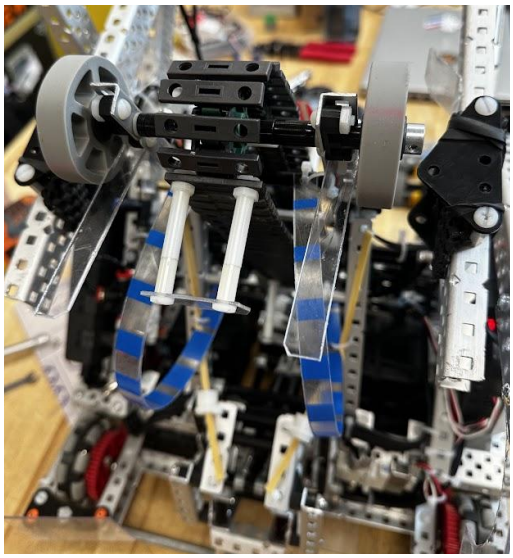
ARM 1.0

BUILD

AUTHOR; RACINE LIU

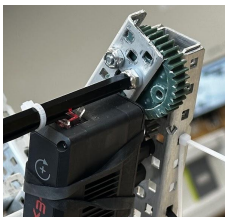
DATE; 10.24.2024

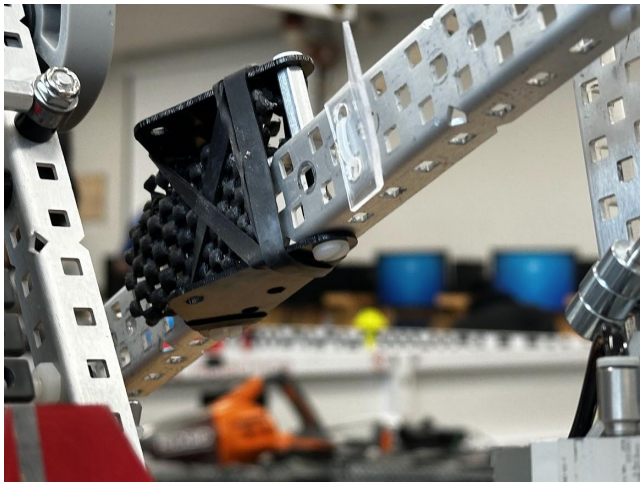
SCORING CONSISTENCY

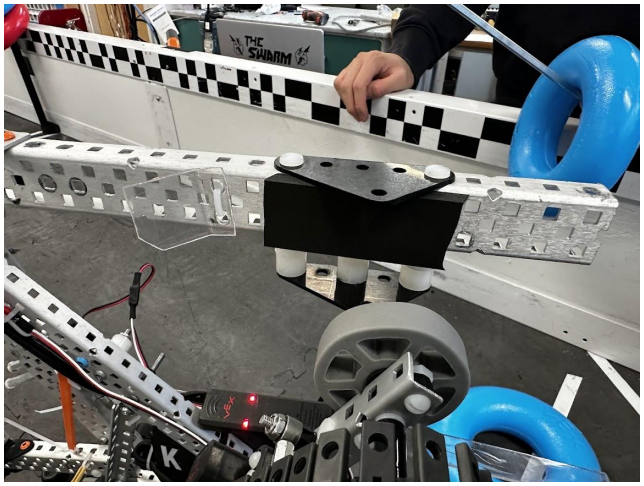


As a secondary precaution, we used another 2 polycarbonate stripes attached in a curved matter to further prevent rings obstructing the intake.

Measures like these taken are meant to ensure consistently and reliability throughout matches, optimizing our time to focus on strategy.







The upper 2 images were processes testing the grip onto the ring, previously on the CAD we were only able to express the general structure. We tried out 2 kinds of materials: Antistlipmat as well as adhesive foam.

With the anti slipmat, we secured the material through using elastic bands hoping that it would add more grip in the friction hold of the ring. We found that when raising the arm by had the rings had difficulty staying put.

The second solution of using the foam on the image to the upper right, proved to be more consistent as we found the material compressed more while keeping the ring wedged in

CODING THE DRIVE

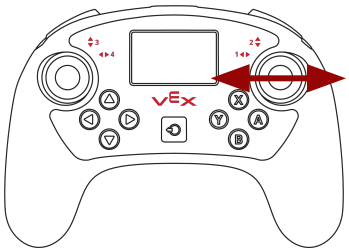
First, following our plugged in brain ports, we group 2 separate sides of the drivetrain together. This allows each side of the drivetrain to act as a single unit, receiving the same commands and improving synchronization of movement

Indicates the motor cartridge

```
// motor groups
pros::MotorGroup leftMotors({-12, 13, -14}, pros::MotorGearset::blue); // left motor group
pros::MotorGroup rightMotors({15, -16, 17}, pros::MotorGearset::blue); // right motor group
```

Type of control

We've commonly used a split arcade drive throughout the past seasons hence we will be sticking to it. As opposed to a regular arcade drive where the robot movement is controlled with 1 joystick, we have the movement controlled by two



1. Left Joystick: Controls forward and backward movement (**Throttle**).
2. Right Joystick: Controls turning left and right (**Turn**).

Throttle definition:

Throttle refers to how much power the robot is given to move forward or backward. It determines the speed of the robot's movement.

How the Power is Calculated

Throttle: For forwards and backwards movement, pushing the joystick in in the desired direction gives the robot the power to move. The further the joystick is pulled, the more power is applied.

Turn: Pushing the right joystick to the right adds power to the left motor while reducing power to the right motor. Pushing the right joystick to the left adds power to the right motor while reducing power to the left motor, allowing the robot to turn left.

DRIVETRAIN 1.0

CODE

AUTHOR; RACINE LIU

DATE; 10.25.2024

Power sent to the left and right drivetrains can be calculated as such

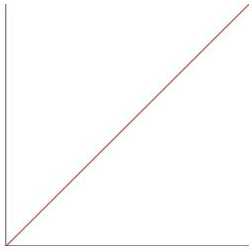
Left motor power = Throttle + turn

The left motor power is a combination of the throttle input and the turning input.
When the robot needs to turn to the right, the **turn** value will be positive, causing the left motor to receive more power than the right motor. This results in a right turn.
Conversely, if the **turn** value is negative (indicating a left turn), the left motor will get less power.

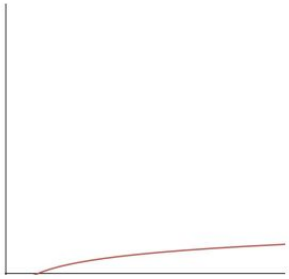
Right Motor Power = Throttle - turn

The right motor power is calculated similarly but in a way that reduces its power when turning.
When the robot turns right (positive **turn**), the right motor receives less power (because **turn** is subtracted from **throttle**), allowing the robot to pivot to the right.
When the **turn** value is negative (indicating a left turn), the right motor will get more power than the left.

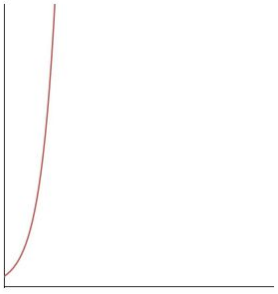
As we decided to use arcade drive, we now need to define drive curves because they play a crucial role in fine-tuning the robot's response to joystick inputs. Drive curves transform the raw input values from the joysticks into smoother and more manageable outputs for the motors, allowing for better control and maneuverability. Below are some commonly used curves



Linear (y = x)
A direct relationship between input and output; for every unit increase in input, there is a corresponding unit increase in output. This results in a consistent response but may lack sensitivity at low speeds.

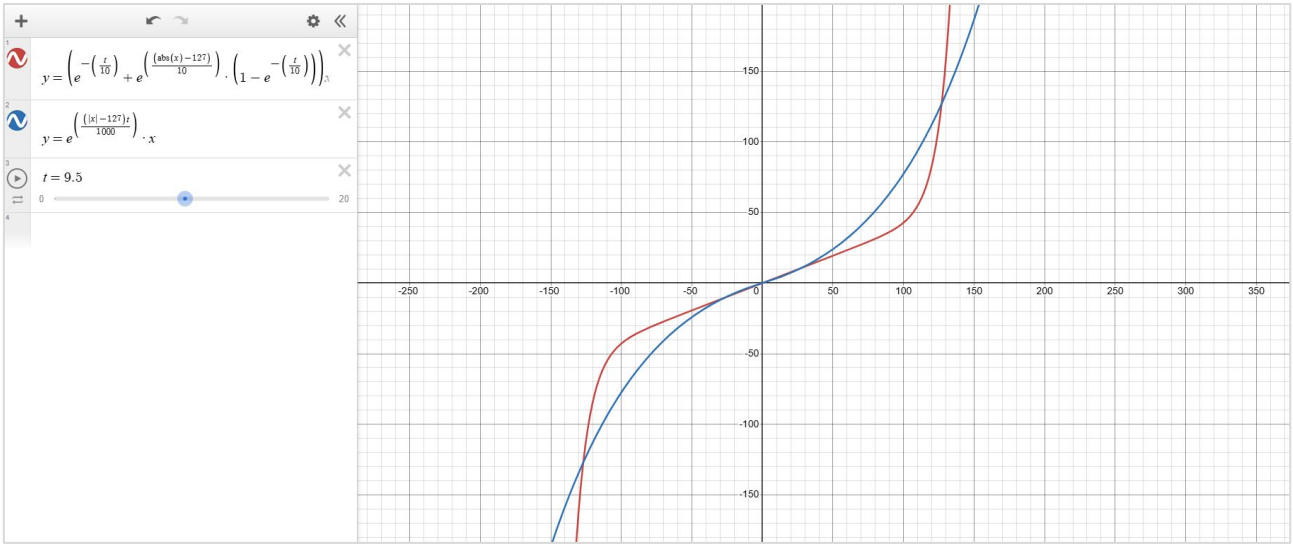


Logarithmic (y = log(x))
This graph begins with a slightly larger increase, followed by flattening. This means that once the controller position reaches a certain position, the velocity would become constant.



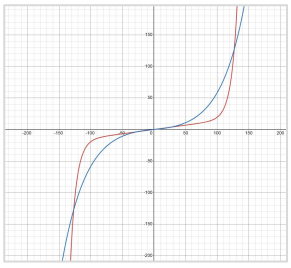
Exponential (y = 2^x)
This curve has a gradual start and a steep increase, allowing for more sensitive control at lower speeds and greater responsiveness at higher speeds.

A flatter curve indicates a more graduate increase, whereas a steeper curve means increased sensitivity. We've decided to use the Eztemplate drive curve as it effectively combines the strengths of linear, logarithmic, and exponential drive curves to enhance the overall driving experience, providing both sensitivity and control. This blend allows drivers to execute a wide range of maneuvers effectively.

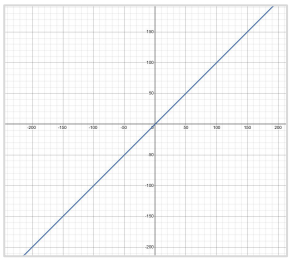


T value

1. **Scaling Factor:** The variable t is often used as a scaling factor that adjusts the overall shape of the curve. It affects how quickly or slowly the output changes in response to the input.
2. **Influence on Sensitivity:**



   - When t is larger, the curve becomes steeper, meaning that small changes in joystick input result in larger changes in motor output. This increases sensitivity and responsiveness, allowing for rapid acceleration and fast responses.



   - Conversely, when t is smaller, the curve flattens out, resulting in a more gradual change in output relative to input. This decreases sensitivity, providing smoother control and making it easier to manage delicate movements.

DRIVETRAIN 1.0

CODE

AUTHOR; RACINE LIU

DATE; 10.25.2024

parameters

```
EZDriveCurve::EZDriveCurve(float deadband, float curve)
    : deadband(deadband),
      curveGain(curve) {}
```

The deadband is a threshold value. If the joystick input is within this threshold (either positive or negative), the output will be zero. Creates a "neutral zone" where no motion occurs, providing better control and preventing jittery movements when the joystick is at rest.

This value influences the shape of the drive curve. It determines how sensitive the output is to changes in joystick input.

Ex. Larger curveGain = steeper curve = more sensitivity

If the joystick input is within this range, the function will return 0, meaning no power will be sent to the motors.

divisor

```
float EZDriveCurve::curve(float input) {
    if (fabs(input) < deadband) return 0;
    return (powf(2.718, -(curveGain / 10)) + powf(2.718, (fabs(input) - 127) / 10) * (1 -
powf(2.718, -(curveGain / 10)))) * input;
}
```

This mathematical constant e (Euler's number) is used to create exponential curves, which help in achieving smooth acceleration and deceleration based on joystick input.

This is the maximum value for joystick input, representing full forward motion. It acts as a reference point for scaling the response of the drive curve.

The math in the curve function creates a smooth exponential drive curve that adjusts motor output based on joystick input, enhancing control sensitivity

```
void EZDriveCurve::changeCurve(float change) {
    // Allow the user to change the curve gain
    curveGain += change;
}

float EZDriveCurve::getCurveGain() { return curveGain; }
```

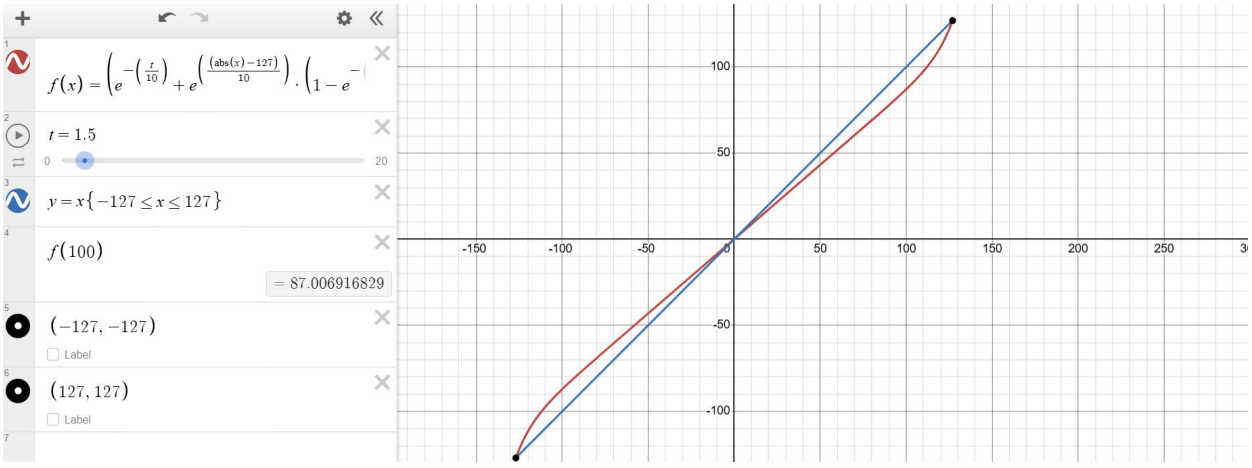
- changeCurve(float change): This function lets users modify the sensitivity by adding a specified change value
- getCurveGain(): returns the current value of curveGain, allowing other parts of the code to access the curve.

Together, these functions allow for fine tuning tuning and monitoring of the drive curve's behavior based on the driver's preferences or testing needs.

Tuning setup

```
void change_drive_curve() {
    // Temporary: change the drive curve
    // Up and down arrows increase and decrease the throttle gain
    // Left and right arrows increase and decrease the turn gain
    while (true) {
        if (controller.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_UP)) {
            throttleCurve.changeCurve(0.1);
        }
        if (controller.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_DOWN) &&
throttleCurve.getCurveGain() > 0) {
            throttleCurve.changeCurve(-0.1);
        }
        if (controller.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_LEFT) &&
steerCurve.getCurveGain() > 0) {
            steerCurve.changeCurve(-0.1);
        }
        if (controller.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_RIGHT)) {
            steerCurve.changeCurve(0.1);
        }
        pros::delay(10);
    }
}

void display_curve_gains() {
    // Temporary: display the drive curve gains on the controller
    while (true) {
        float driveGain = throttleCurve.getCurveGain();
        float turnGain = steerCurve.getCurveGain();
        controller.print(0, 0, "D: %.1f  T: %.1f", driveGain, turnGain);
        pros::delay(100);
    }
}
```

Below is a visualization of our drive curve (red) graph after being tuned to the driver's preferences in comparison to a linear curve (blue)



The code takes the user's throttle (forward/backward input) and turn (left/right input), applies custom drive curves to each to make the robot's movements smoother, and then calculates how much power to send to each side of the drivetrain.

DRIVETRAIN 1.0

CODE

AUTHOR; RACINE LIU

DATE; 10.25.2024


```
void Chassis::ez_arcade(int throttle, int turn) {
    float curved_throttle = throttleCurve->curve(throttle);
    float curved_turn = steerCurve->curve(turn);

    float leftPower = curved_throttle + curved_turn;
    float rightPower = curved_throttle - curved_turn;
```

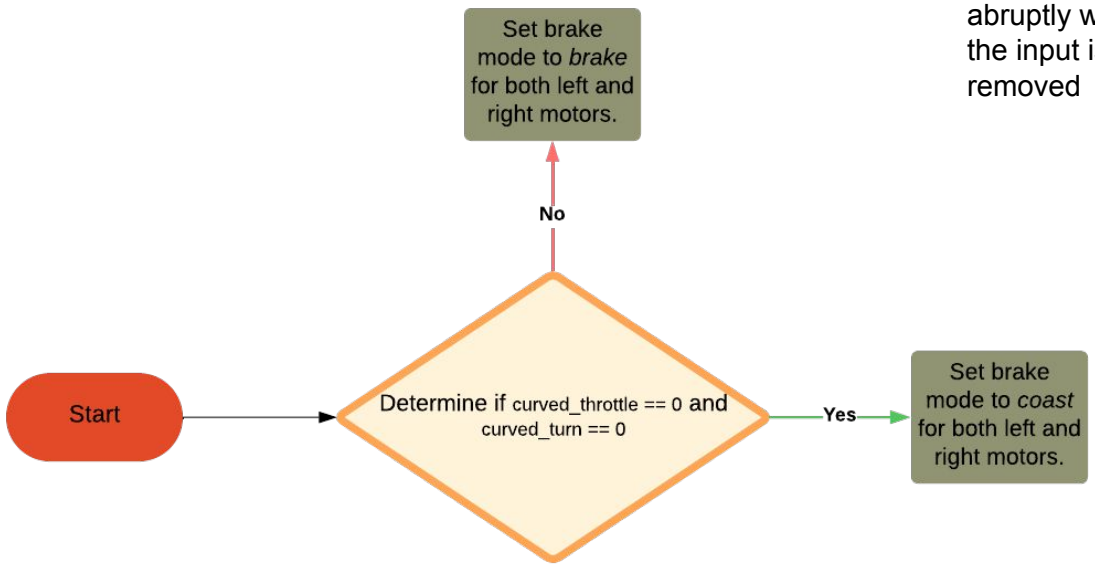
Smoothed right/left steering

Smoothed forward/backward speed

```
   // set motor to brake if any input is registered, else coast
   if (curved_throttle == 0 && curved_turn == 0) {
       drivetrain.leftMotors->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
       drivetrain.rightMotors->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
   } else {
       drivetrain.leftMotors->set_brake_mode(pros::E_MOTOR_BRAKE_BRAKE);
       drivetrain.rightMotors->set_brake_mode(pros::E_MOTOR_BRAKE_BRAKE);
   }
```

coast mode, allows the robot to gradually slow down.

brake mode, means the robot will stop more abruptly when the input is removed

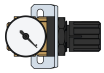


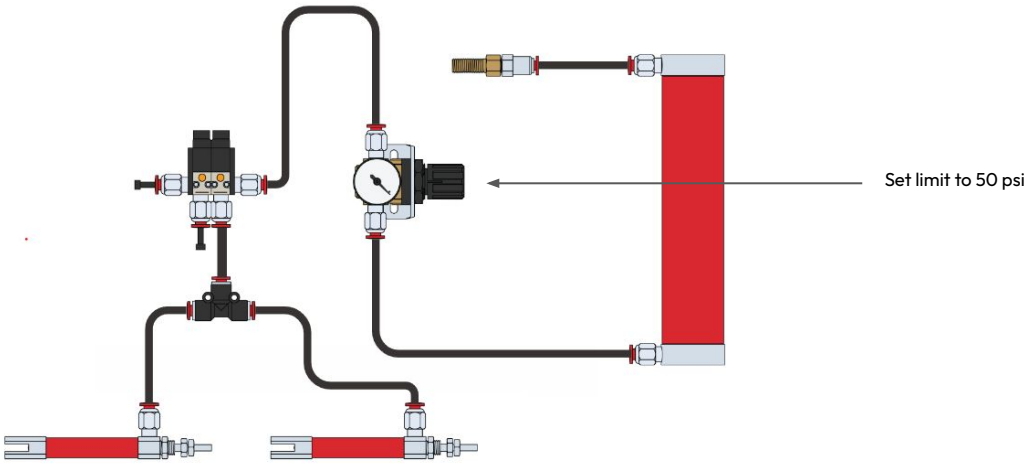

```
    drivetrain.leftMotors->move(leftPower);
    drivetrain.rightMotors->move(rightPower);
}
```

We first state which port is designated for each mechanism

```
//3 wire devices
pros::adi::DigitalOut mogo_grabber(8);
```

Tubing

Item	200mL Air Tank	Tee Fitting	Straight Male Fitting	Straight Female Fitting	Double Acting Solenoid	25mm Piston	Air Pressure Regulator	Valve Stem	4mm Plug
Illustration									
Quantity	1	1	10	1	1	2	1	1	2




```
void button_toggles() {

    while (true) {

        if (controller.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_L2)) {
            mogoGrabberState = !mogoGrabberState;
            mogo_grabber.set_value(mogoGrabberState);
        }

    pros::delay(10);
    }
}
```

The clamp actuates when the button L2 is pressed

10 millisecond delay before executing action

ARM + INTAKE 1.0

AUTHOR; RACINE LIU

DATE; 10.25.2024

CODE

As the intake loads into the arm, we've decided to code the arm an intake within one function,

This boolean variable tracks whether the arm is currently being controlled manually. It starts as false, indicating that the arm is not moving manually.

```
void intake_and_arm() {
    // X and B move the arm up and down manually
    //
    bool movingArmManually = false;
    ArmPosition queuedPosition = INACTIVE;

    while (true) {
```

This begins an infinite loop, the code inside will run continuously until the program is stopped

```
if (controller.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_X)) { // Manually move arm up
    armMotors.move/""(64); // move arm up
    movingArmManually = true; // disallow automatic arm movement
    pros::Task ([&] () {
        while (controller.get_digital(pros::E_CONTROLLER_DIGITAL_X)) { pros::delay(20); } // wait until released
        movingArmManually = false; // allow automatic arm movement
        // If not holding B, stop the arm
        if (!controller.get_digital(pros::E_CONTROLLER_DIGITAL_B)) armMotors.move(0);
    });
}
```

Enables manual control of the robot's arm to move upward when the X button is pressed. While the arm is being manually moved, automatic control is disabled. Once the X button is released, the code checks whether the B button is being pressed to determine whether to stop the arm or allow it to continue moving

```
if (controller.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_B)) { // Manually move arm down
    armMotors.move(-64); ——→ Speed, negative value indicates downwards motion
    movingArmManually = true;
    pros::Task ([&] () {
        while (controller.get_digital(pros::E_CONTROLLER_DIGITAL_B) || armMotors.get_position() > LOADING) { pros::delay(20); }
        movingArmManually = false;
        if (!controller.get_digital(pros::E_CONTROLLER_DIGITAL_X) || armMotors.get_position() <= LOADING) armMotors.move(0);
    });
}
```

Purpose: This checks two conditions:

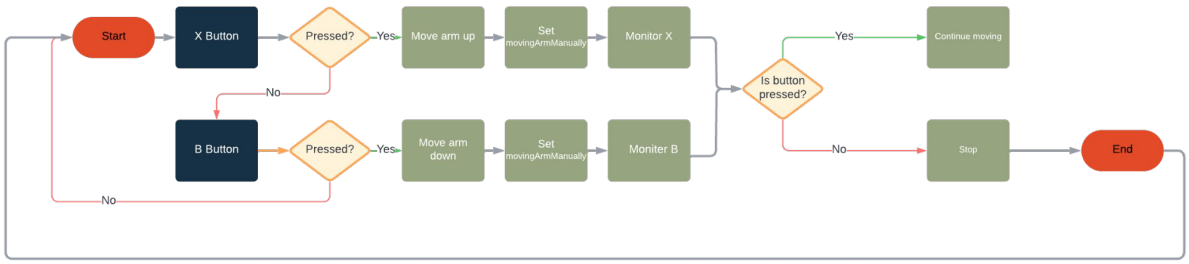
1. If the X button is not pressed.
2. If the arm's position is at or below the LOADING position.

If either condition is true, it stops the arm motors by setting their speed to 0. This ensures that when the manual control is released (with the B button), the arm stops moving if the user is not trying to move it upwards (with the X button) or if the arm has reached the designated loading position.

Purpose: Inside this task, the code continuously checks two conditions:

1. If the B button is still being pressed.
2. If the arm's current position is above a predefined LOADING position.

If either condition is true, it waits for 20 milliseconds before checking again. This loop ensures that the arm keeps moving down as long as the B button is held or until it reaches the LOADING position.



ARM + INTAKE 1.0

CODE

Arm and intake working in accordance

```
if (controller.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_R1)) { // Intake and move arm to loading
    intake.move(127); // Spin the intake inwards
    pros::Task ([&] () {
        queuedPosition = LOADING; // Queue the arm position to loading
        do (pros::delay(10)); while (movingArmManually);// Wait for any manual movements to stop
        // If the queued position is still loading, move the arm to the loading position
        if (queuedPosition == LOADING) armMotors.move_absolute(LOADING, 200);
    });
    pros::Task ([&] () {
        while (controller.get_digital(pros::E_CONTROLLER_DIGITAL_R1)) { pros::delay(20); } // Wait until released
        // If other intake control buttons aren't being held, stop the intake
        if (!(controller.get_digital(pros::E_CONTROLLER_DIGITAL_R2)
            || controller.get_digital(pros::E_CONTROLLER_DIGITAL_A))) intake.move(0);
    });
}
```

When the R1 button is pressed:

1. The intake motor starts spinning inward.
2. The arm's position is set to the "loading" position unless it is being manually moved.
3. The intake stops once the R1 button is released (unless other buttons are pressed to keep it running).

```
if (controller.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_L1)) {  // Move arm to scoring
    pros::Task ([&] () {
        queuedPosition = SCORING; // Queue the arm position to scoring
        do (pros::delay(10)); while (movingArmManually);// Wait for any manual movements to stop
        // If the queued position is still loading, move the arm to the loading position
        if (queuedPosition == LOADING) armMotors.move_absolute(LOADING, 200);
    });
}

if (controller.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_R2)) { // Intake
    intake.move(127);
    pros::Task ([&] () {
        while (controller.get_digital(pros::E_CONTROLLER_DIGITAL_R2)) { pros::delay(20); }
        if (!(controller.get_digital(pros::E_CONTROLLER_DIGITAL_A)
            || controller.get_digital(pros::E_CONTROLLER_DIGITAL_R1))) intake.move(0);
    });
}
```

1. When L1 is pressed, the arm moves to the scoring position.
2. When R2 is pressed, the intake motor starts spinning. Once R2 is released, the intake stops (unless other buttons are pressed to keep it running).

```
    if (controller.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_A)) { // Outtake
        intake.move(-127);
        pros::Task ([&] () {
            while (controller.get_digital(pros::E_CONTROLLER_DIGITAL_A)) { pros::delay(20); }
            if (!(controller.get_digital(pros::E_CONTROLLER_DIGITAL_R2)
                || controller.get_digital(pros::E_CONTROLLER_DIGITAL_R1))) intake.move(0);
        });
    }

    pros::delay(10);
  }
}
```

In case of rings getting stuck, we created an outtake, spinning the intake motors in reverse at full speed controlled by button A

DRIVETRAIN 1.0

AUTHOR; RACINE LIU

DATE; 10.28.2024

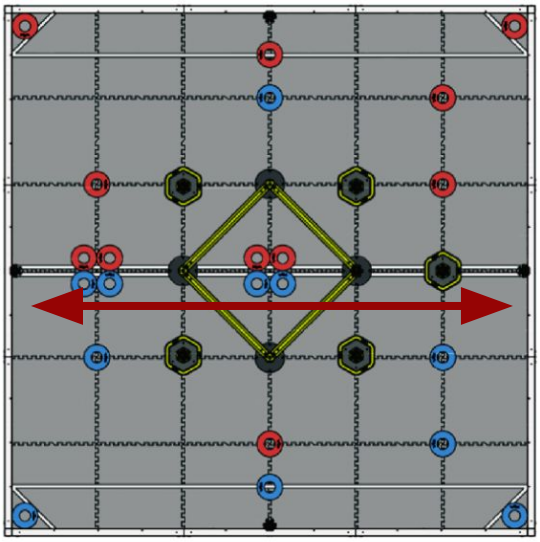
TEST

TEST NO.1 SPEED

Requirements:
We want to see how much time the robot takes to travel across the field

Materials:
Robot and field

IMPORTANCE



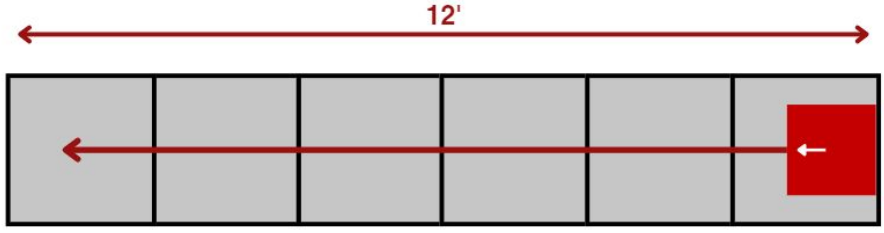
We will need to traverse across the field through multiple game scenarios, hence speed is essential.

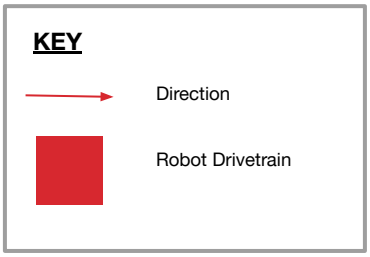
Top stakes need to be prioritized as they are worth the most points, but again as there are multiples there's bound to be scenarios where teams take turns trying to possess the most top rings. We will need to take all opportunities to rush to available stakes.

Corners are also prioritized, and they are found across the field from one another. Possibly rushing from corner to corner with a stolen mobile goal is a strategy.

PROCEDURE

1. Set up field tiles in the same way as the vex field (12x12)
2. Place the robot so that it's centered on the tiles at one end of the 6 tiles on the field. Make sure the front end is pointed to the far end, as it will drive around 12 feet forward soon.
3. Driver starts near the robot on the side of the field and the Timer starts across the driver.
4. Start stopwatch as the driver pushed the joysticks forward to the max
5. Drive forward until end of field, passing the 6th tile where the timer stops the stopwatch
6. Repeat steps 2-5 a total of 5 times





AUTHOR; RACINE LIU

DATE; 10.28.2024

DRIVETRAIN 1.0

TEST

RESULTS

QUANTITATIVE DATA: SUCCESS

Test	1	2	3	4	5
Time	1.98s	2.01s	1.95s	1.97s	2.00s

Average: 1.982s

QUALITATIVE DATA: SUCCESS

Upon observing, the robot stayed in position, driving in a straight line as well as appearing to travel very fast. Acceleration was smooth.

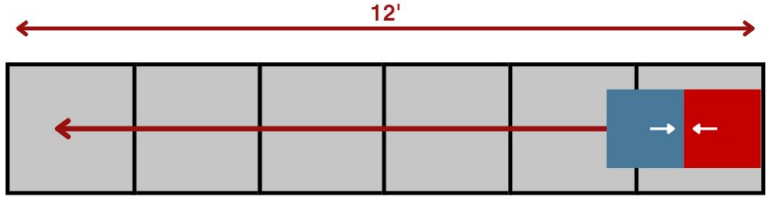
TEST NO 2 : PUSHING POWER

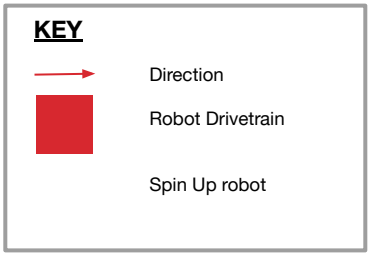
Requirements:
We want to see if the robot is capable of defense and offense when going head to head with another robot. And if the gears stall out.

Materials:
Drivetrain, Over Under Worlds robot, VRC field tiles, stopwatch, and 2 people(driver and timer)

PROCEDURE

1. Set up field tiles in the same way as the vex field (12x12)
2. Place the robot so that it's centered on the tiles at one end of the 6 tiles on the field. Make sure the front end is pointed to the far end, as it will drive around 12 feet forward soon.
3. Place the Worlds bot in front of the robot,
4. Driver starts near the robot on the side of the field, and the Timer starts across the driver.
5. Start stopwatch as the driver pushed the joysticks forward to the max
6. Drive forward until end of field, passing the 6th tile where the timer stops the stopwatch
7. Repeat steps 2-6 a total of 5 times





DRIVETRAIN 1.0

AUTHOR; RACINE LIU

DATE; 10.28.2024

TEST

RESULTS

QUANTITATIVE DATA: SUCCESS

Test	1	2	3	4	5
Time	2.98s	2.97s	3.00s	2.98s	3.02s

QUALITATIVE DATA: SUCCESS

Gears did not stall out, robot speed driving across the field was slower than test no. 1 as expected pushing a disabled robot with only 1 second more than regular time.

TEST NO 3: MOTOR LIFE

Requirements:

We want to see how long the robot can handle driving around without the motors overheating. Knowing this will allow us to know when to check on the robot to cool it down during games.

Materials:

Drivetrain, VRC field tiles, stopwatch, and 2 people(driver and timer)

PROCEDURE

1. Set up field tiles in the same way as the vex field (12x12)
2. Place the robot on the field
3. Start stopwatch
4. Drive the robot in circles around until speed decreases and maneuverability in turns becomes difficult.
5. Stop stopwatch
6. Wait 10 or more minutes for robot to completely cool down
7. Repeat steps 2-5 a total of 5 times

RESULTS

QUANTITATIVE DATA: SUCCESS

Test	1	2	3	4	5
Time	2.49 min	2.53 min	2.55 min	2.46 min	2.51 min

Average: 2.5s

QUALITATIVE DATA: SUCCESS

Upon observing, the robot didn't appear to be slower in turns at first. The driver was able to feel the difference, but the observers only are able to view the effects later.

AUTHOR; RACINE LIU

DATE; 10.28.2024

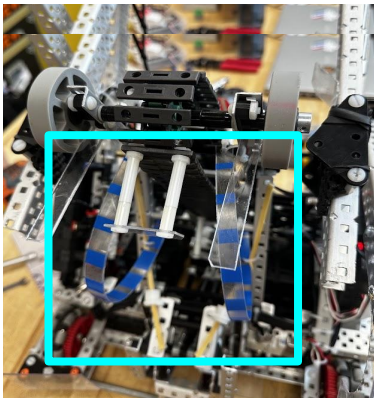
INTAKE 1.0

TEST

Requirements:
Testing plastic curve effect, we are unsure if the secondary 2 plastic pieces help keep the rings in place as we only approximated when building.

We'll be testing from no plastic to 3 curve of varying shapes to see how the rings score.

Materials:
Robot, mobile goal, and 6 rings



PROCEDURE

1. Pump robot with air and clamp onto mobile goal
2. Continuously run the intake
3. Intake 6 rings
4. See if there's a jam at the 6th ring preventing the intake from running
5. Repeat steps for the 4 options, 10 times

RESULTS

1	j	j	j	j	j	j	j	j	j	j
2	j	j	c	sc	j	sc	j	j	j	c
3	c	c	sc	c	sc	c	c	c	c	c
4	c	j	sc	c	c	j	j	c	c	j
5	j	j	j	j	j	j	j	j	j	j

LEGEND

1. **Clear(c):** Intake hook smoothly passing through 6th ring
2. **Semi Clear(sc):** Intake hook momentarily caught on 6th ring
3. **Jam(j):** Intake hook fully caught on the 6th ring

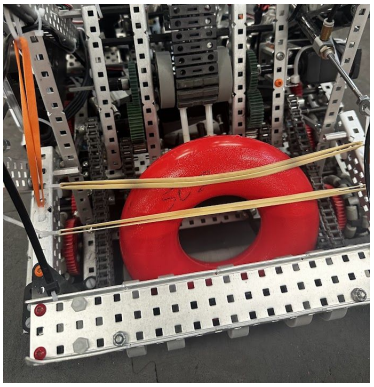
Conclusion: position 3 is the most ideal

INTAKE 1.0

ANTISTALL

AUTHOR; RACINE LIU

DATE; 10.28.2024



Upon trying to intake multiple rings while the robot is driving, we found that the intake would jam quite often, therefore being unable to score the rings onto the mobile goal.

This would be a significant issue in games or auto if the intake jams, preventing us from scoring rings. A solution to this issue is to code an antistall into the intake to minimize the time it takes to outtake

ANTISTALL CODE

We use a class, it has functions and members. members are variables specific to the class that can be either private (only changeable from inside) or public (can be changed from outside) it also has a constructor at the very top of the code, that's what gets called when instantiating (make a new instance of and initialize) an object of AntiStall type

```cpp
#include "anti_stall.hpp"

AntiStall::AntiStall(pros::MotorGroup& motors, int velocityThreshold)
    : motors_(motors), threshold(velocityThreshold) {
    stallTask = new pros::Task(preventStall, this, "AntiStallTask");
}
```

"this" is a pointer to the current instance of the AntiStall, it is an arrow that guides the code towards the memory dedicated towards whatever instance of AntiStall we're in, be that one for the intake or some other motor

```cpp
AntiStall::~AntiStall() {
    if (stallTask) {
        stallTask->remove();
        delete stallTask;
    }
}

pros::Task* AntiStall::getTask() {
    return stallTask;
}
```

Destructor (~AntiStall()):

- Cleans up when an AntiStall object is removed.
- Checks if stallTask exists (not null).
- Calls remove() on stallTask to stop it.
- Deletes stallTask to free up memory.

INTAKE 1.0

ANTISTALL


```cpp
pros::Task* AntiStall::getTask() {
    return stallTask;
}

void AntiStall::preventStall(void* param) {
    AntiStall* self = static_cast<AntiStall*>(param);
    int timeStalled = 0;
    while (true) {
        float targetVelocity = self->motors_.get_target_velocity();
        // printf("target velocity: %.2f\n", targetVelocity);
        if (std::abs(targetVelocity) < 10 || !self->enabled) {
            timeStalled = 0;
            pros::delay(10);
            continue;
        }

        float actualVelocity = self->motors_.get_actual_velocity();
        if (std::abs(actualVelocity) < (self->threshold / 100.0f) * targetVelocity) {
            timeStalled += 10;
        } else {
            timeStalled = 0;
        }

        if (timeStalled >= MAX_STALL_TIME) {
            timeStalled = 0;
            self->motors_.move_velocity(-targetVelocity);  // Reverse to clear stall
            pros::delay(REVERSE_TIME);
            if (self->motors_.get_target_velocity() == -targetVelocity) {
                self->motors_.move_velocity(targetVelocity);  // Restore original direction
            }
            pros::delay(BUFFER_TIME);  // Prevent immediate re-trigger
        }

        pros::delay(10);  // Task delay
    }
}

void AntiStall::enable() { enabled = true; }
void AntiStall::disable() { enabled = false; }
void AntiStall::toggle() { enabled = !enabled; }
bool AntiStall::isEnabled() const { return enabled; }

void AntiStall::setThreshold(int newThreshold) { threshold = newThreshold; }
int AntiStall::getThreshold() const { return threshold; }
```

Main detection function

Checks if the actual motor velocity (actualVelocity) is less than a certain percentage of the target velocity (targetVelocity), based on the threshold.

temporarily reverses the motor direction to help dislodge whatever might be causing the stall.

By reversing the motor briefly, this code gives the motor a chance to "shake off" a minor obstacle, effectively increasing motor longevity and system reliability.

INTAKE 1.0

TEST

AUTHOR; RACINE LIU

DATE; 10.28.2024

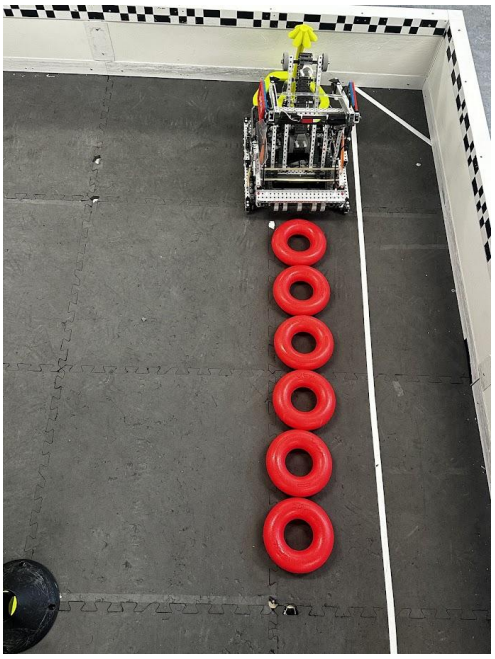
TEST NO.1 SPEED

Requirements:
We want to see how much time the robot takes to fill up a mobile goal

Materials:
Robot, field, intake, mogo, and 6 rings

PROCEDURE

1. Set up field tiles in the same way as the vex field (12x12)
2. Place the robot on the field
3. Clamp a mogo
4. Line up 6 rings in a row
5. Start stopwatch
6. Intake
7. Stop stopwatch
8. Repeat 10 times



RESULTS

QUANTITATIVE DATA: SUCCESS

Test	1	2	3	4	5
Time	7.65s	7.59s	7.58s	7.63s	7.63s

QUALITATIVE DATA:

Intaking is smooth with no jams both on the 6th ring and while intaking. The anti stall triggers and reverses allowing for smooth intake. The robot is slower intaking the rings lined up than expected but during games rings will not be lined up as suchmaking the current intake viable to use.

AUTHOR; RACINE LIU

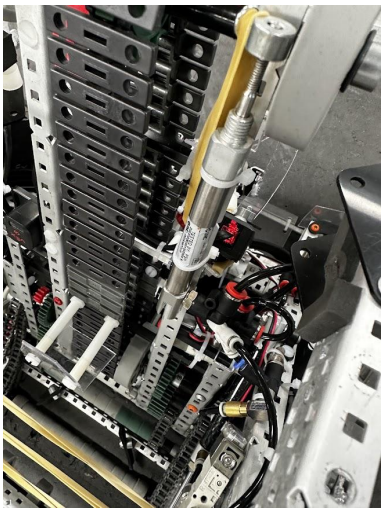
DATE; 10.29.2024

INTAKE 1.0

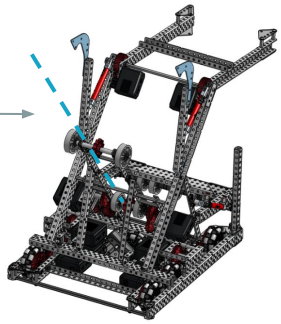
COLOR SORTER

We often encounter large piles of rings or rings we want to avoid, which can disrupt our path and slow us down. While we can usually rely on our driver to navigate around these rings, any mistakes could cost us the game. To minimize this risk, we developed a color sorter.

We mounted a single action piston along the intake, where when it triggers it shoots out to the profile of the hook allowing the ring to slide off of it, being ejected from possession.



Path of piston



The ColorSorter class here is responsible for automatically sorting rings based on their color, using an optical sensor, motor group, and a digital output that controls a mechanism to block or release rings.

```cpp
#include "color_sorter.hpp"

ColorSorter::ColorSorter(pros::Optical& optical, pros::MotorGroup& intake, pros::adi::DigitalOut& ring_blocker)
    : optical_(optical), intake_(intake), ring_blocker_(ring_blocker) {
    sortTask = new pros::Task(colorSort, this, TASK_PRIORITY_DEFAULT + 4, TASK_STACK_DEPTH_DEFAULT + 4, "ColorSorter");
}

ColorSorter::~ColorSorter() {
    if (sortTask) {
        sortTask->remove();
        delete sortTask;
    }
}

pros::Task* ColorSorter::getTask() {
    return sortTask;
}
```

Destructor (~ColorSorter): Stops and deletes the sorting task to free memory.

getTask(): Returns a pointer to the sorting task for external access

The code below continuously checks the color of rings detected by an optical sensor, along with proximity data, to identify whether a ring should be removed

```cpp
void ColorSorter::colorSort(void* param) {
    ColorSorter* self = static_cast<ColorSorter*>(param);
    float hue, proximity;
    while (true) {
        if (!self->isEnabled() || !self->teamAssigned()) {
            pros::delay(10);
            continue;
        }
```

If the sorter isn't enabled or the team color isn't set, it pauses briefly (10ms) and restarts the loop.

INTAKE 1.0

COLOR SORTER

AUTHOR; RACINE LIU

DATE; 10.29.2024


```
        hue = self->optical_.get_hue();
        proximity = self->optical_.get_proximity();
        // printf("hue: %.2f proximity: %.2f\n", hue, proximity);
        if (self->intake_.get_target_velocity() > 0 // If intake is moving
            && proximity > 50 // Something is close to the sensor
            && (((hue < 50 || hue > 310) && self->teamColor == AutonCategory::Blue) // Red disk when on blue team
                || (hue > 50 && hue < 310 && self->teamColor == AutonCategory::Red)) // Blue disk when on red team
            ) {
                self->remove_ring();
                pros::delay(BUFFER_TIME-200);
            }
        pros::delay(2); // Fast loop to increase polling rate
    }
}
```

Ensures that only close objects are processed, avoiding false detections.

If conditions above are met, the ring is ejected. The code delays to insure no back to back removals occur

This function(above) checks whether the intake is moving and whether a ring is close enough, and it identifies the ring color. If an unwanted ring is detected based on the team color, it removes the ring and waits briefly before continuing the loop.

```
void ColorSorter::remove_ring() {
    ring_blocker_.set_value(true);
    int oldSpeed = intake_.get_target_velocity();
    double retractPosition = intake_.get_position() + ANGLE_CHANGE + 240;
    double reversePosition = intake_.get_position() + ANGLE_CHANGE - 110;
    int start = pros::millis();
    while (intake_.get_position() < reversePosition && pros::millis() - start < TRIGGER_TIMEOUT) { pros::delay(2); }
    int reverseSpeed = 600;
    intake_.move_velocity(-reverseSpeed);
    pros::delay(200);
    if (intake_.get_target_velocity() == -reverseSpeed) intake_.move_velocity(oldSpeed);
    while (intake_.get_position() < retractPosition && pros::millis() - start < TRIGGER_TIMEOUT) { pros::delay(2); }
    ring_blocker_.set_value(false);
}

void ColorSorter::setTeam(AutonCategory category) {
    teamColor = category;
}

AutonCategory ColorSorter::getTeam() const {
    return teamColor;
}
```

piston

Moves intake forward to eject the ring.

Moves backward briefly to seperate ring

The ring_blocker piston is responsible for extending and retracting to control ring flow. When a ring of the wrong color is detected, the ring_blocker is immediately set to true, extending the piston to prevent any rings from falling onto the mobile goal (mogo). As the ring moves closer to the top, the intake motor briefly reverses to dislodge the ring from the hook, causing it to fall off. With the piston in the extended position, the ring can't land on the scoring stake, so it tumbles away, unscored.

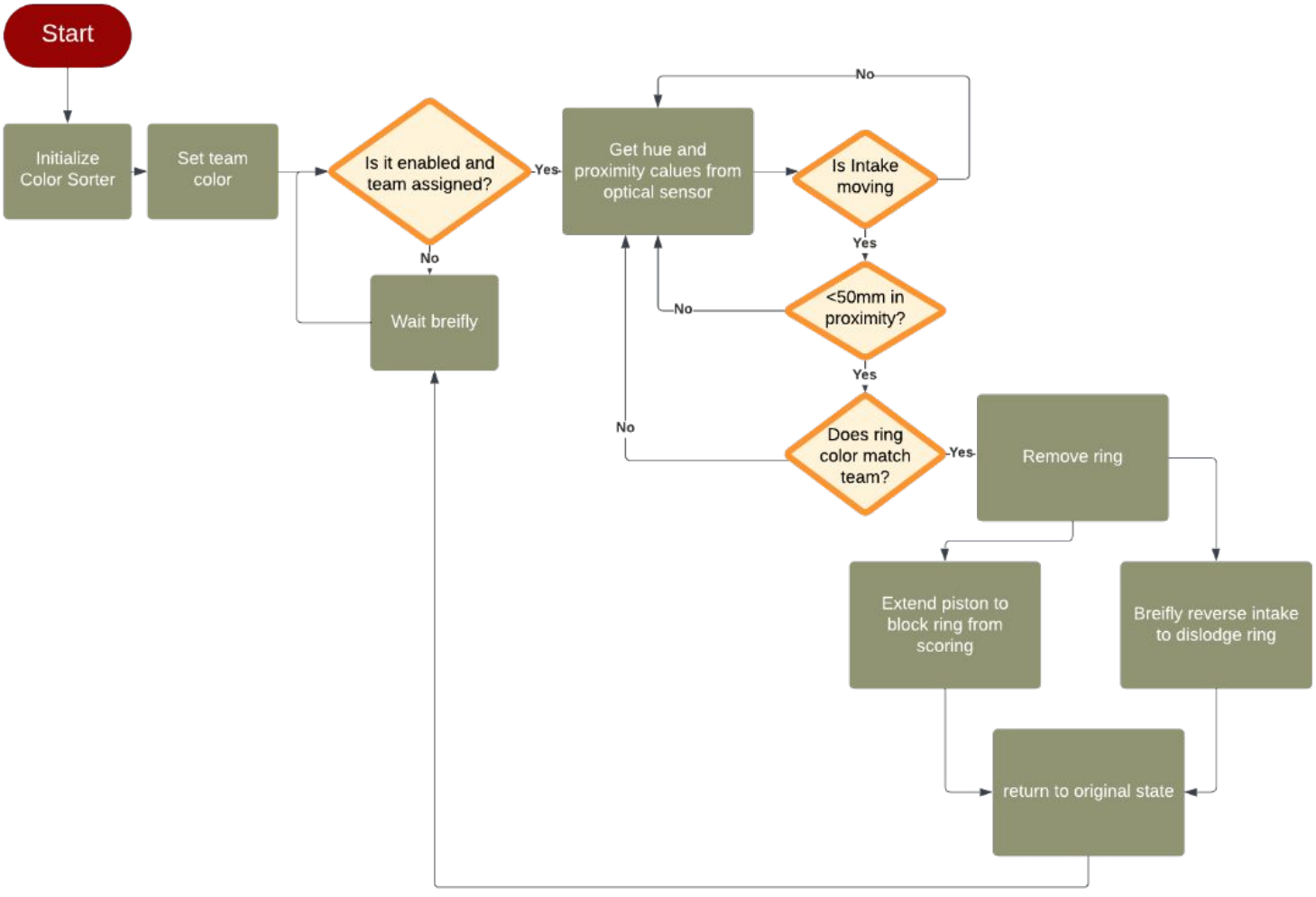
The remaining code in remove_ring simply manages the color sorting process, as the function name implies, controlling the sequence of actions needed to remove unwanted rings effectively.

INTAKE 1.0

COLOR SORTER


```cpp
bool ColorSorter::teamAssigned() const {
    return teamColor != AutonCategory::None;
}

void ColorSorter::enable() {
    optical_.set_led_pwm(100);
    enabled = true;
}

void ColorSorter::disable() {
    optical_.set_led_pwm(0);
    enabled = false;
}

void ColorSorter::toggle() {
    if (isEnabled()) { disable(); }
    else { enable(); }
}

bool ColorSorter::isEnabled() const {
    return enabled;
}
```

This code manages the ColorSorter's activation status, allowing it to be enabled, disabled, or toggled, and checks if a team color is set.



, but

INTAKE 1.0

AUTHOR; RACINE LIU

DATE; 10.29.2024

COLOR SORTER TEST

TEST NO.1 RELIABILITY

Requirements:
We want to see how consistent the color sort mechanism is
Materials:
Robot, field, intake, mogo, and 6 rings of each color

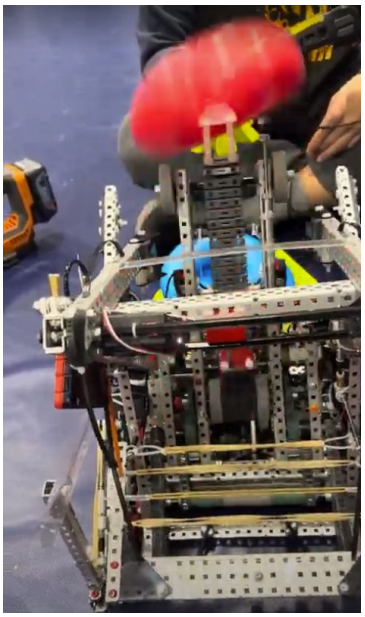
IMPORTANCE

Being able to rely on the color sorter makes it so that it's viable to use in games as well as having more confidence in our autonomous programs. This will allow us to save on time instead of trying to avoid opponent colored rings altogether.

PROCEDURE

1. Set the color to red
2. Intake 3 red rings in a row then 3 blue
3. Intake 6 blue
4. Alternate in taking 1 color each
5. Repeat steps 1-4 but with the color set to blue

RESULTS



QUANTITATIVE DATA: SUCCESS

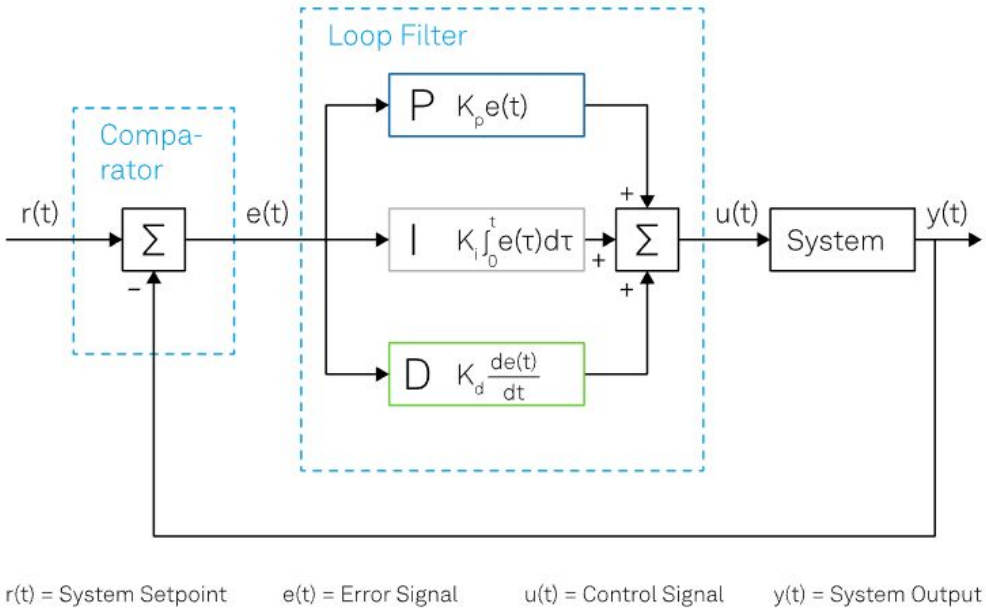
Test	1	2	3
RED	success	success	success
BLUE	success	success	success

QUALITATIVE DATA:

Color sorter worked every time. Although there seems to be issues with it stalling out and slightinconsitency in triggering times. So far it's still been able to sort the rings but later tuning is necessary to fix this slight issue.

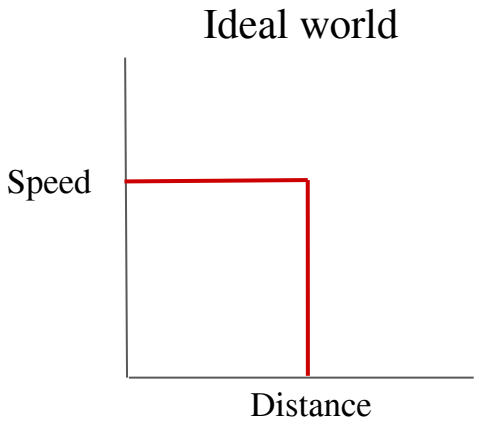
PID OVERVIEW

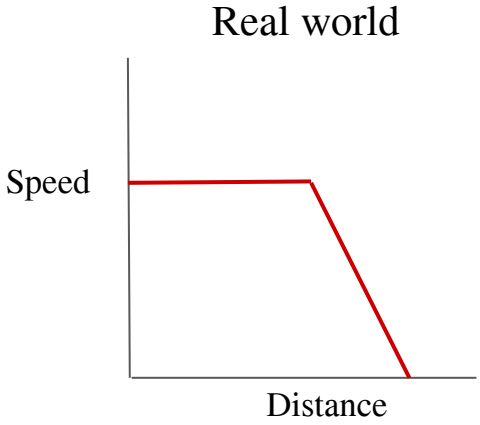
WHAT ARE PIDS?



A PID (Proportional, Integral, Derivative) controller is a feedback control mechanism to control systems such as motors. It works by taking input from sensors, processing it, and producing an output, which in the case of robotics, is usually the power of the motor.

The PID controller is designed to minimize the error between the desired set point (the target position or state) and the actual position or state of the system. In our case, the sensors would be encoders, and the output would be the power of the motor (-127 to 127). In an ideal world, we could tell the robot to move until it reaches a set distance (set point), followed by a stop and it would end up at the position we want it to. However, in our world, we have something called inertia. Inertia would cause the robot to exceed this set point. To solve this, we would have to slow down towards the end.





Each component, P, I and D are each multiplied by constants, kP, kI and kD, respectively in order to produce the best output and then added together. These constants are usually found based on manual tuning. Although there are many other ways to calculate them, they often do not really work for robots.

ARM 1.0

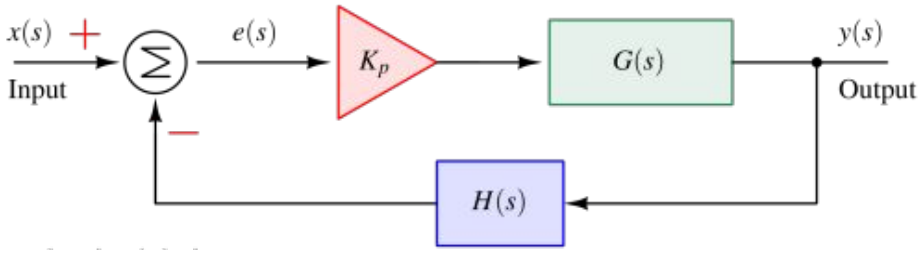
PID OVERVIEW

AUTHOR; RACINE LIU

DATE; 10.30.2024

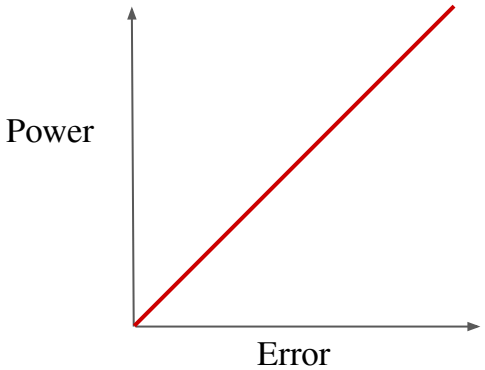
A frequently brought up thought is powering the motor in the opposite direction to stop it. Unfortunately, this will definitely cause the motor to break down. Immediately changing directions for a motor can cause it to wear out. This results in not only more effort to change motors, but also a degradation of performance over time.

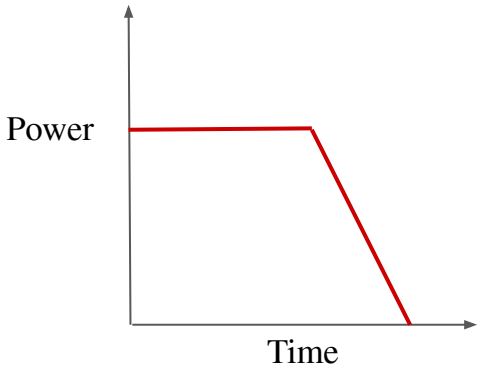
PROPORTIONAL CONTROLLER



This component is responsible for the immediate response to the error. It calculates the proportional error (the difference between the setpoint and the current position) and multiplies it by a constant (kP) to determine the control output. If the error is large, the output will be high, and if the error is small, the output will be low. The P controller helps the system to quickly react to changes in error.

It is simply kP*error. The error refers to the distance between the setpoint and our current position. This would form the bulk of our power. If the error is large, the power output would be high. When the errors reduce, the power delivered to the motor reduces as well.





Tuning kP for the P controller is relatively easy. Start with kP = kI = kD = 0.0, increase kP by 0.1 (Or small increments, depending on your system) until the robot can reach the setpoint. You may want to verify this by printing the encoder value or displaying it somewhere. If the robot starts braking (denoted by sharp “clicks”) or moving backwards, kP is too high - reduce it. If you are to choose between a higher or lower kP.

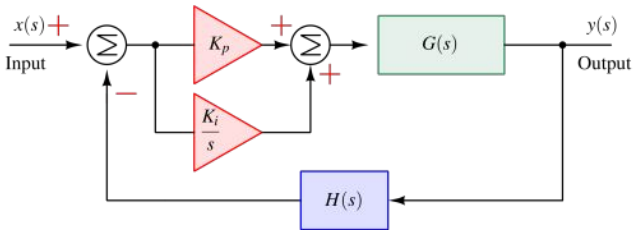
AUTHOR; RACINE LIU

DATE; 10.30.2024

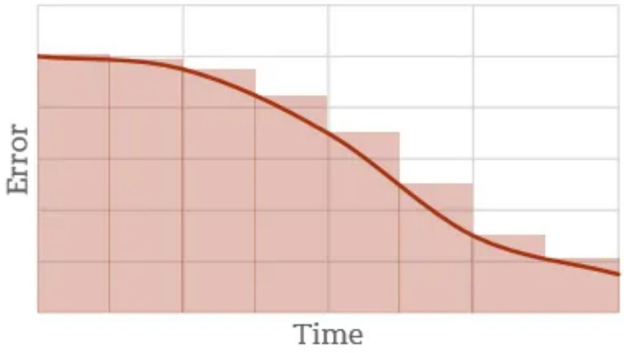
ARM 1.0

PID OVERVIEW

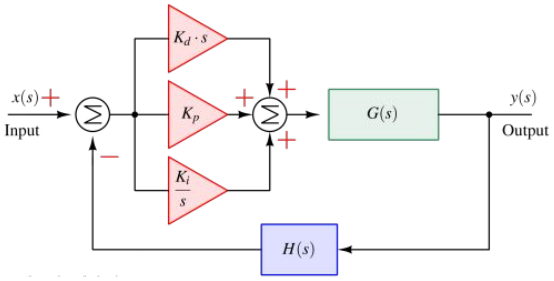
INTEGRAL CONTROLLER



The I controller is meant for systems that run for a long period of time. The I component in a PID loop is kI*sum of (errors*dT). For those who know integration, you would be wondering how it is linked. If we were to have the time between loops as dT, the area under the error-time graph (integral of the graph) at each point can be estimated by dT * error. Hence, our area under graph is the sum of errors. Some people prefer to remove dT from the equation and asume a constant sample rate, which works too - at least it does theoretically.



DERIVATIVE CONTROLLER



The D controller will basically act as a dampener. Mathematically, the derivative is the gradient at the point of the curve - in this case, the change in error/dT. Once again, you can remove dT, but it is preferred to keep it there just in case of other tasks interfering with the time between loops (the VEX Brain's processor is not multithreaded and could thus result in inconsistencies). So your D controller would be kD*(change in error/dT).

This can be thought of logically. Your change is error is negative because you are getting closer to a target. If your change in error is big, the magnitude of D would be big, slowing down the system. If your change is error is too small, the system will be slowed down by a larger amount. Either way, there's sort of negative progress, which is why I mentioned to choose a higher P value. Now, to tune kD. Increase kD by small increments (similar to small increments in P) until the robot reaches the setpoint consistently, without braking. If the robot tends to stop/slow down significantly halfway then continue, decrease kD.

ARM 1.0

PID CODING

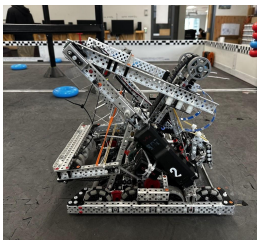
AUTHOR; RACINE LIU

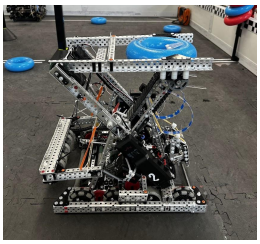
DATE; 10.30.2024

INITIAL AND LOADING STAGE

In order to facilitate scoring consistency, our goal is to have 4 arm stages which we will code with the use of PIDs.

```
void intake_and_arm() {
    while (true) {
        if (controller.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_X)) { // Manually move arm up
            while (calibratingArm) { pros::delay(10); }
            arm.move_velocity(100); // move arm up
            movingArmManually = true; // disallow automatic arm movement
            pros::Task ([&] () {
                while (controller.get_digital(pros::E_CONTROLLER_DIGITAL_X)) { pros::delay(20); } //
wait until released
                movingArmManually = false; // allow automatic arm movement
                // If not holding B, stop the arm
                if (!controller.get_digital(pros::E_CONTROLLER_DIGITAL_B)) { arm.move_velocity(0); }
            });
        }

        if (controller.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_B)) { // Manually move arm down
            while (calibratingArm) { pros::delay(10); }
            arm.move_velocity(-100);
            movingArmManually = true;
            pros::Task ([&] () {
                while (controller.get_digital(pros::E_CONTROLLER_DIGITAL_B)) { pros::delay(20); }  //
maximum arm position is loading
                movingArmManually = false;
                if (!controller.get_digital(pros::E_CONTROLLER_DIGITAL_X)) { arm.move_velocity(0); }
            });
        }
```

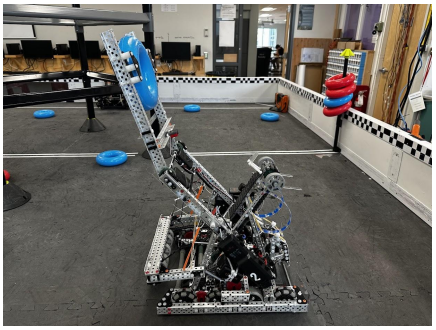




The code above states that when button X is is pressed the arm moves to the first position to load the ring. In order to return to it's initial state of being lowered, button B must be pressed.

RAISED STAGE

The next stage is to facilitate alignment to the wall stake in order to score. By using PIDs to hold it up, the ring will be more secure and less prone to falling off due to the jerk casued by manual control



AUTHOR; RACINE LIU

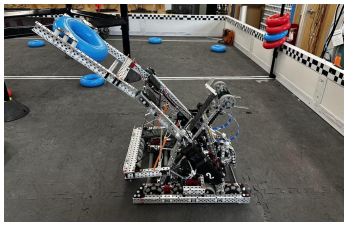
DATE; 10.30.2024

ARM 1.0

PID CODING

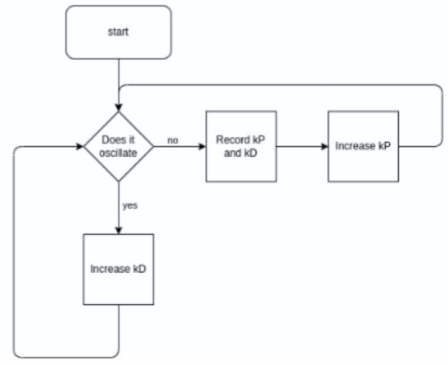

```
        if (controller.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_R1)) { // Intake and move arm
to loading
            intakeAntiStall.disable();
            intake.move_velocity(600); // Spin the intake inwards
            pros::Task ([&] () { auto_move_arm(LOADING); });
            pros::Task ([&] () {
                while (controller.get_digital(pros::E_CONTROLLER_DIGITAL_R1)) { pros::delay(20); } //
Wait until released
                // If other intake control buttons aren't being held, stop the intake
                if (!(controller.get_digital(pros::E_CONTROLLER_DIGITAL_R2)
                    || controller.get_digital(pros::E_CONTROLLER_DIGITAL_A))) {
intake.move_velocity(0); intakeAntiStall.enable(); }
            });
        }

        if (controller.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_L1)) {  // Move arm to scoring
            pros::Task ([&] () { handle_L1(); });
        }
        if (controller.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_R2)) { // Intake and move arm
to inactive
```

The arm is then rotated to the final stage which score the ring onto the wall stake so all the driver needs to do is to drive backwards and the ring will be securly scored on the stake.



After scoring the arm either goes straight to the load position or down in order to score rings onto teh mobile goals

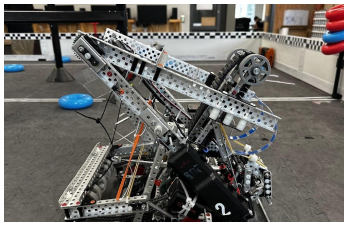
OVERVIEW AND TUNING



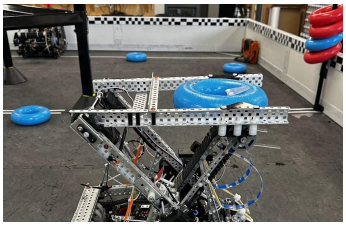
In order to tune PID values we first begin with all the values at 0 except P. After that we slowly increase either P I or D depending on where there is oscillation

```
double p = 2;
 double i = 0;
 double windupRange = 5;
 double d = 5;
```

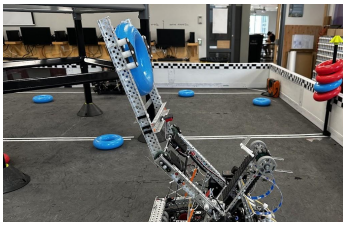
LOWERED



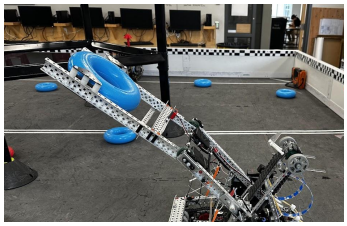
RECEIVE



RAISE



SCORE



ARM 1.0

TEST

AUTHOR; RACINE LIU

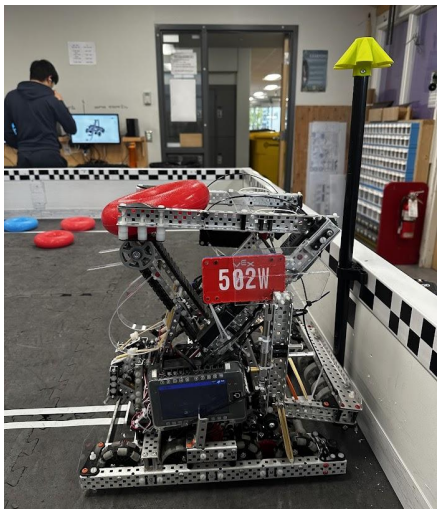
DATE; 10.31.2024

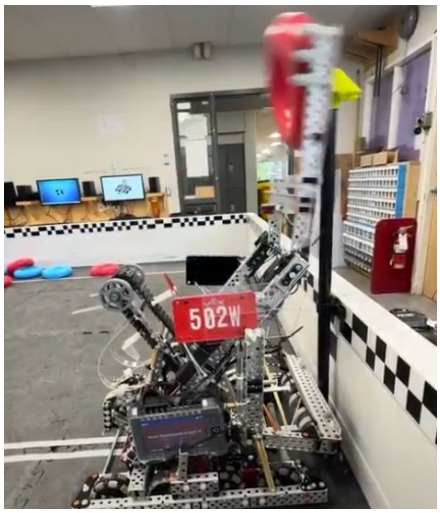
TEST NO.1 SPEED

Requirements:
We want to see how consistent and fast scoring is with the usage of PIDs implemented we want to ensure it scores in a second or less
Materials:
Stopwatch, robot, field, intake, and a ring

PROCEDURE

1. Load ring into arm
2. Align robot to the wall stake
3. Start stopwatch
4. Score
5. Stop stopwatch
6. Repeat process 10 times





RESULTS

QUANTITATIVE DATA: SUCCESS

Test	1	2	3	4	5
Time	0.73s	0.78s	0.76s	0.8s	0.75s

QUALITATIVE DATA:

Easy and fast scoring motion

AUTHOR; RACINE LIU

DATE; 11.02.2024

AUTONOMOUS

SOLO AWP

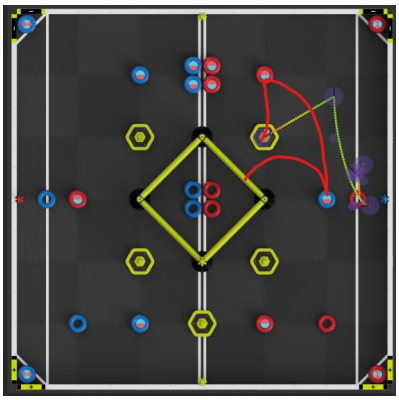
IMPORTANCE

AWPs are crucial in qualification matches as they significantly boost a team's rankings even if they lose the game. Unlike the previous season where 2 teams would need to work together to be awarded this season a solo AWP is possible.

The conditions of the AWP are:

1. Min of 2 stakes need to be scored
2. the ladder touched
3. no robot in contact with the starting line

ROUTE



1. Alliance stake is scored with preload via the intake
2. Robot intakes a ring partially
3. Grabs onto the mobile goal
4. Scores 1 ring from middle pile onto the mobile goal
5. Touches bar with arm raised

```
void blue_awp_solo(lemlib::Chassis& chassis) {
    arm_down();
    intakeColorSort.disable();
    chassis.setPose(61, 13, 270);
    intake_mover.set_value(true);
    chassis.turnToHeading(225, 750);
    chassis.moveToPoint(54, 8, 750, {.forwards = true, .minSpeed = 0}, false);
    intake.move_velocity(600);
    intake_mover.set_value(false);
    pros::delay(500);
    chassis.moveToPoint(56, 10, 750, {.forwards = false, .minSpeed = 0});
    pros::delay(50);
    intake.move_velocity(50);
    chassis.turnToHeading(360, 750, {.minSpeed = 70});
    chassis.waitUntilDone();
    chassis.moveToPose(63, -2.5, 270, 1250, {.forwards = false, .lead = 0.6, .maxSpeed = 80}, false);
    chassis.turnToHeading(270, 500, {}, false);
    intake.move_velocity(600);
    pros::delay(1250);
    intake.move_velocity(-600);
    pros::delay(250);
    chassis.moveToPose(50, 48, 0, 5000, {.forwards = true, .lead = 0.6, .minSpeed = 30});
    intakeColorSort.enable();
    chassis.waitUntil(12);
    intake.move_velocity(600);
    chassis.waitUntil(36);
    intake.move_velocity(0);
    chassis.turnToHeading(60, 750, {.minSpeed = 30});

    chassis.moveToPose(22, 27, 60, 5000, {.forwards = false, .lead = 0.55, .maxSpeed=80}); // Come back
and grab the mogo
    chassis.waitUntil(24);
    chassis.cancelMotion();
    chassis.moveToPose(22, 27, 60, 500, {.forwards = false, .lead = 0.1, .maxSpeed=20,
.earlyExitRange=3}); // Come back and grab the mogo
    chassis.waitUntil(8.5);
    mogo_grabber.set_value(true);
    intake.move_velocity(600);
    chassis.turnToHeading(0, 750);
    chassis.moveToPose(24, 44, 0, 5000, {.forwards = true, .lead = 0.3, .minSpeed = 60}); // Grab low
disk
    chassis.moveToPose(63, 63, 45, 2000, {.forwards = true, .lead = 0.3}); // Move into corner
    chassis.waitUntilDone();
    pros::delay(500);
    chassis.moveToPose(40, 40, 45, 2000, {.forwards = false, .lead = 0.1, .minSpeed = 60});
    chassis.turnToHeading(225, 750, {.minSpeed = 70});
    chassis.moveToPose(24, 24, 225, 2000, {.forwards = true, .lead = 0.1, .minSpeed = 60});
    arm.move_absolute(900, 200);
}
```

All the alliance needs to do is be able to drive off the line in auto

TOURNAMENT 1 OVERVIEW

AUTHOR; RACINE LIU

DATE; 11.04.2024

RANKING

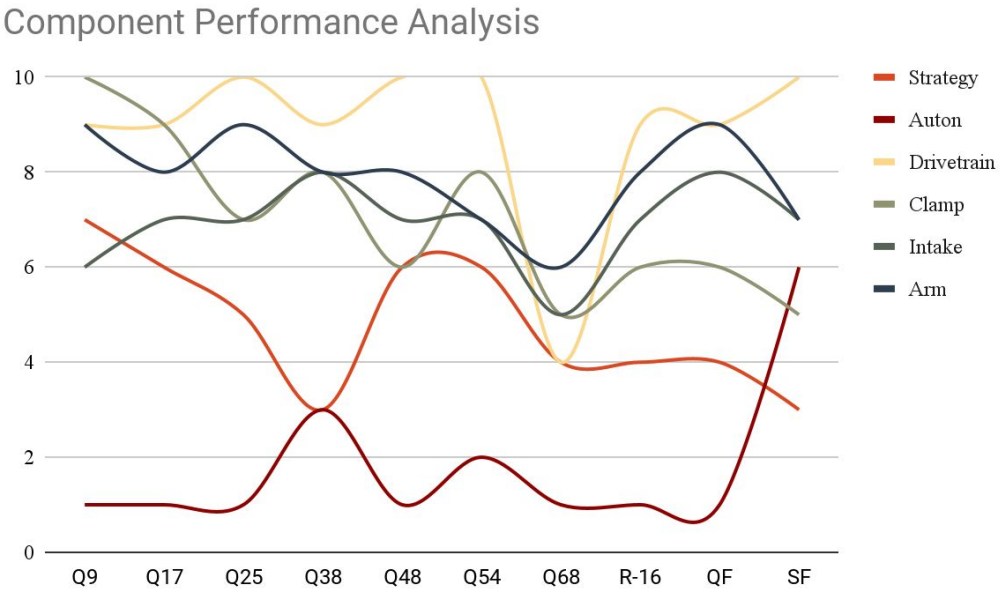
We were placed 11th in skills and qualifications. Winning the innovate award

We did not have the greatest performance this tournament in terms of qualification matches, but we are happy with our skills score especially since we went into the competition without a skills autonomous program coded.

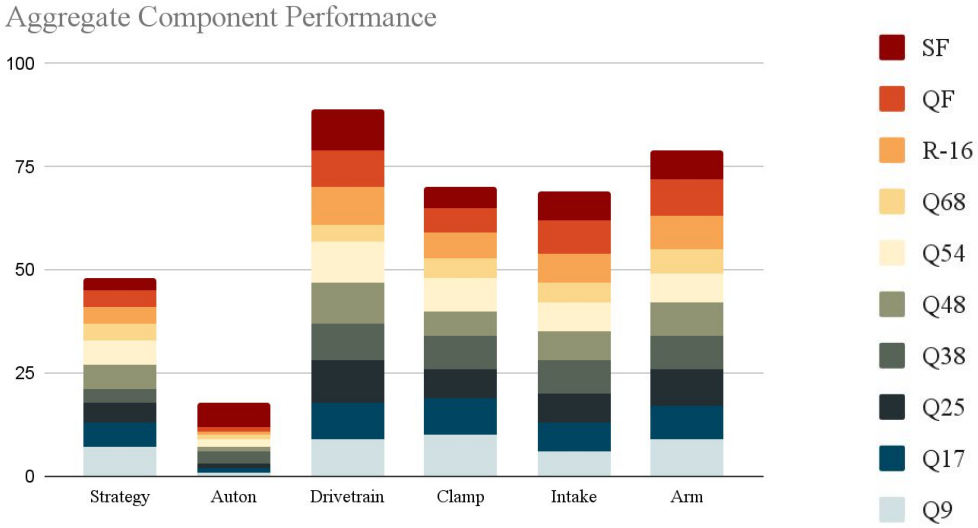
COMPONENT OVERVIEW

Below is a chart of our component performance (including strategy, autonomous, drivetrain, clamp, intake, and arm).

Q 9 1:10 PM	886S 6408W	3	47	502W 938X
Q 17 1:33 PM	98549Z 9181A	5	32	502W 10012Y
Q 25 1:49 PM	9181C 502W	36	30	9181T 10012W
Q 38 2:31 PM	502W 502E	29	31	886Y 28101A
Q 48 3:04 PM	502W 886U	43	28	6408H 502B
Q 54 3:28 PM	1011Y 502W	24	18	9568Z 9181Y
Q 68 4:02 PM	18670A 98549V	37	36	502W 18670C
R16 5-1 4:35 PM	9181C 502W	19	6	9181J 9181T
QF 3-1 4:51 PM	9181C 502W	39	20	98549Z 9181K
SF 2-1 5:14 PM	9181C 502W	26	35	886Y 886R



Overall, we had poorly run strategies as well as non-working autonomous programs as the robot would not be able to consistently score wall stakes. Our clamp performance decreased throughout the competition, and we found that it had difficulty clamping onto the goal in the autonomous period of games.



AUTHOR; RACINE LIU

DATE; 11.04.2024

TEAM MEETING #2

MAKING PLANS

TEAM REFLECTION

Overall we weren't the happiest with how the tournament went. We learned that game strategy is really important in terms of playing the game no matter how well built a robot is. There are many things to account for during the games as well as many elements scored.

Since auto is a 12 point swing and stakes are a 5 point swing. We will focus on bettering our game sense as well as making better autonomous programs.

AGENDA

- Determining next tournament
- Figure out team roles in the design process
- Long term and short term goal setting
- Game strategy development
- Making new schedule

ATTENDANCE

Full team: Vaughan Sandquist, Racine Liu, Aaron Lew, Aiden Tam

EVENT DATES

Tournament	Date	Level
WPRA Season Opener Qualifier	Sep 30	Blended
WPRA Halloween Qualifier	Oct 19	Blended
PYRS Seaquam Season Opener	Nov 2	Blended
WPRA Fall Qualifier	Nov 10	Blended
PYRS Salish Fall Qualifier	Nov 16	Blended
PYRS Heritage Woods	Dec 7	Blended
Gord Trousdell Ten Ton Robotics	Dec 15	HS
WPRA New Year Qualifier	Jan 4	Blended
PYRS Southridge Winter Qualifier	Jan 18	Blended
PYRS Surrey Christian Last Chance	Feb 8	Blended
Ten Ton Robotics Blended Last Chance	Feb 15	Blended
WPRA Last Chance Qualifier	Feb 17	Blended
PYRS BC Mainland Regionals	Mar 1	HS

To the left, is a list of future competitions that we are eligible for. We will be trying our best to attend all of them, as competitions are the fastest way to practice and refine our skills. Competing against a variety of teams exposes us to different strategies and approaches, allowing the opportunity to learn from others and improve our own skills. We decided to use 2 weeks to better prepare for the next competition. So we'll be attending will be the PYRS Salish qualifier. This leaves us around 2 weeks to create a robot.

TEAM MEETING NO.2

GOAL SETTING

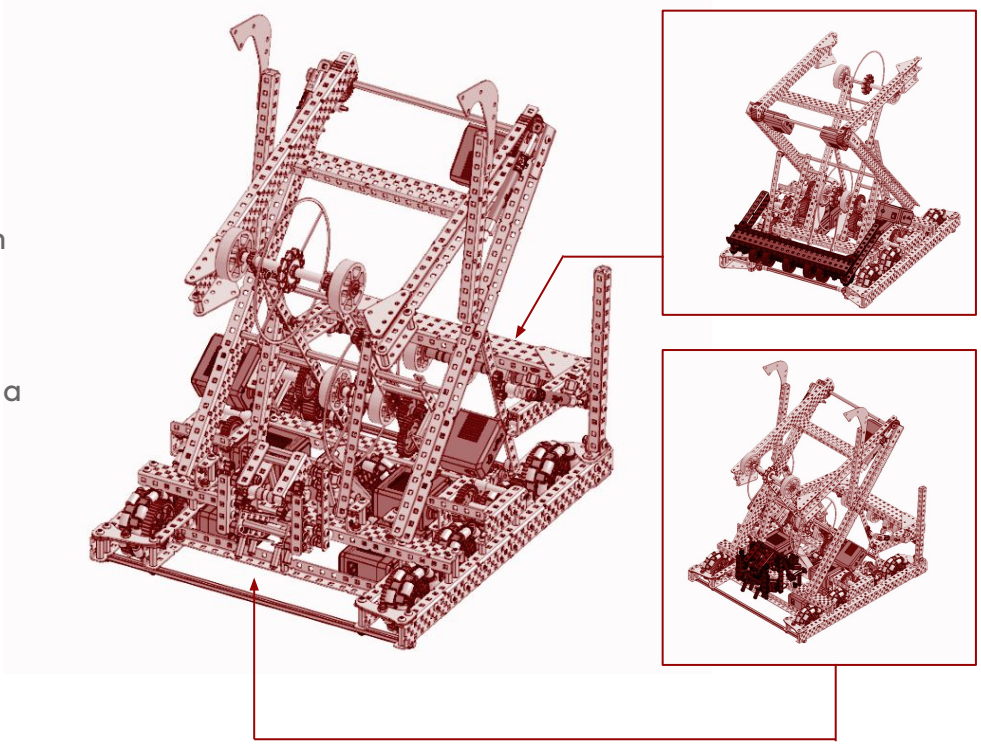
AUTHOR; RACINE LIU

DATE; 11.04.2024

BUILD AMELIORATION

Number 1
The first priority in terms of build is to fix issues with the intake. A better intake allows for a diff in scoring speed from the opponents to the alliance.

Number 2
The 2nd priority is figuring out a way for the clamp to be consistent in autonomous programs



OBJECTIVE : ROBOT BY NOV 15TH

Key result 1: Build ameliorations by Nov 8

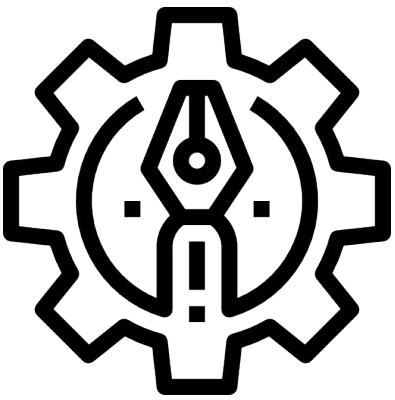
S — Finish robot ameliorations regarding the clamp and intake, having them completely tuned and ready to be coded.

M — The robot is done and ready to be coded. Cad is finished first prior to build.

A — Given the team's current knowledge of CAD software and past build experience, this can be completed within the given timeframe.

R — Having an ameliorated robot will better suit our gameplay after strategies are developed.

T — The robot build ameliorations must be completed (prior to that the designs fully reviewed, and approved by the team) by Nov 8.



AUTHOR; RACINE LIU

DATE; 11.04.2024

TEAM MEETING NO.2

INDIVIDUAL TASKS

Key result 2: Code completion by Nov 15

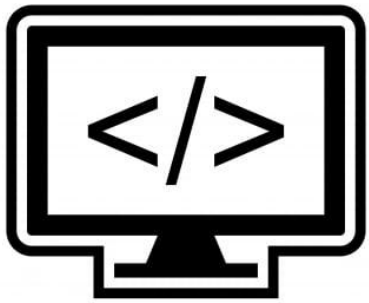
S Finish route and code for game autos as well as skills as well as having them run consistently

M Code uploaded and running successfully at least 3 times in a row

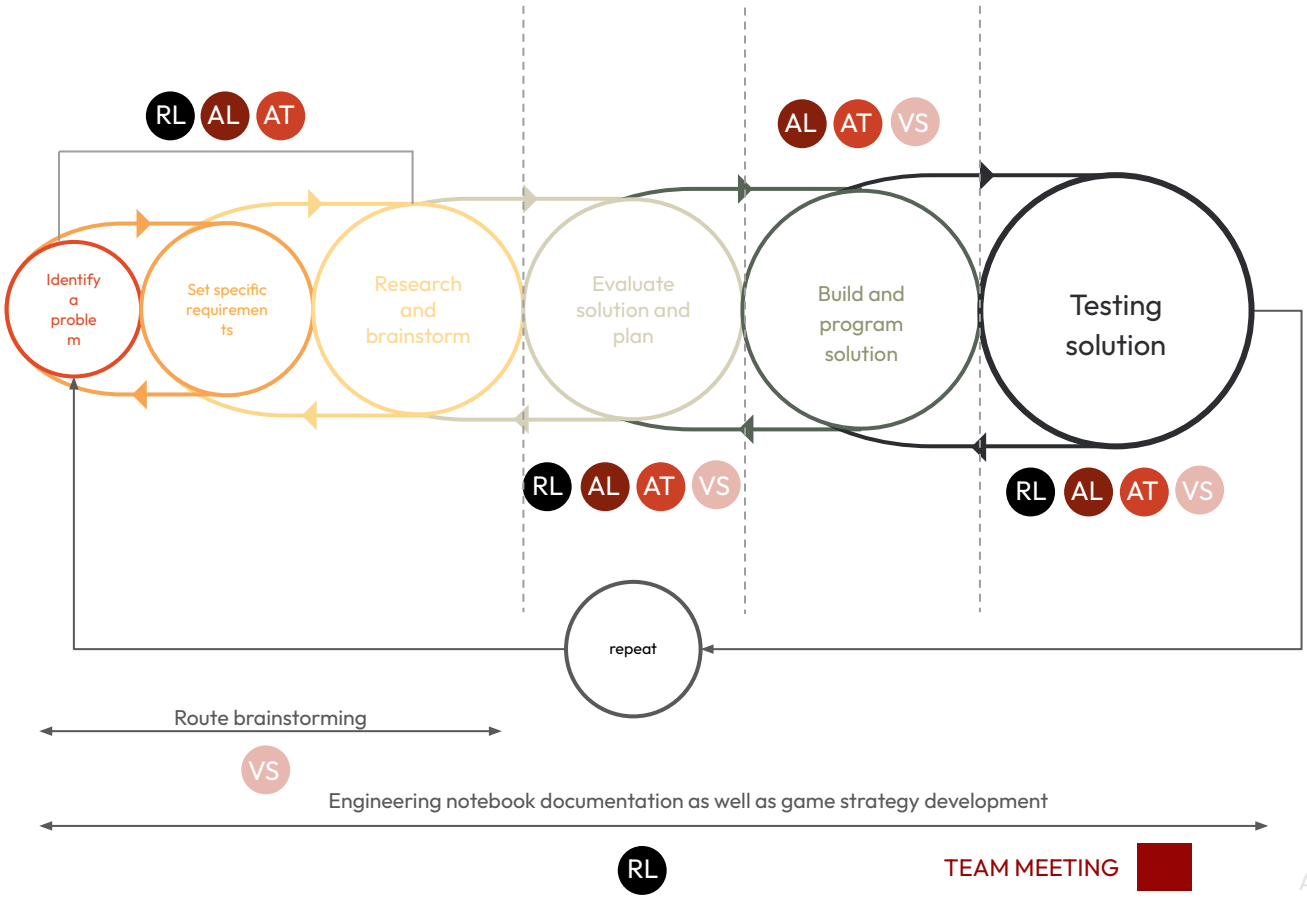
A Given the team's current knowledge in code as well as a the time allocation of a week, we are sure we can complete it.

R Consistent autonomous routes will help us win games as autos result in a 12pt swing. A good skills autonomous can put us into excellence contention.

T The robot code must be fully finished and tuned by Nov 15.



TEAM ROLES



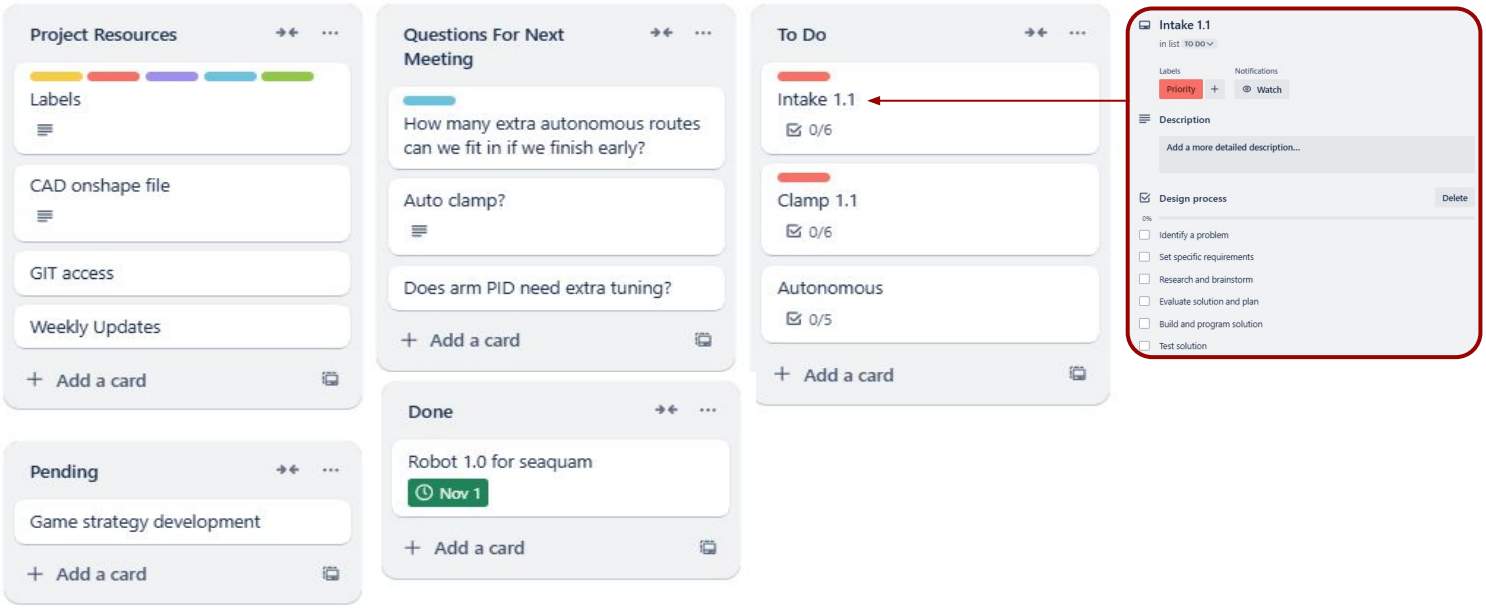
TEAM MEETING NO.2

AUTHOR; RACINE LIU

DATE; 11.04.2024

GANTT CHART

TRELLO



GANTT CHART

	TITLE	ASSIGNED TO	START DATE	END DATE
1	**Design + Build**			
	Intake 1.1		11/4/2024	11/6/2024
	Identify a problem	Racine, Aaron, Aiden	11/4/2024	11/4/2024
	Set specific requirements	Racine, Aaron, Aiden	11/4/2024	11/4/2024
	Research and brainstorm	Racine, Aaron, Aiden	11/4/2024	11/5/2024
	Evaluate solution and plan	ALL	11/5/2024	11/5/2024
	Build and program solution	Vaughn, Aaron, Aiden	11/5/2024	11/5/2024
	Test solution	ALL	11/6/2024	11/6/2024
	Claw 1.1		11/6/2024	11/8/2024
	Identify a problem	Racine, Aaron, Aiden	11/6/2024	11/6/2024
	Set specific requirements	Racine, Aaron, Aiden	11/7/2024	11/7/2024
	Research and brainstorm	Racine, Aaron, Aiden	11/7/2024	11/7/2024
	Evaluate solution and plan	ALL	11/7/2024	11/8/2024
	Build and program solution	Vaughn, Aaron, Aiden	11/8/2024	11/8/2024
	Test solution	ALL	11/8/2024	11/8/2024
2	**Programming**			
	Solo awp	Vaughn, Aaron	11/12/2024	11/13/2024
	Skills auto	Vaughn, Aaron	11/14/2024	11/15/2024
	Game strategy	Racine, Aiden	11/12/2024	11/15/2024

AUTHOR; RACINE LIU

DATE; 11.04.2024

INTAKE 1.1

IDENTIFY AND CONSTRAINT

PROBLEM

The intake seems weaker in game as it performed in practice. The antistall triggered many times during matches which meant that a lot of the time was spent out taking periodically instead of focusing on scoring which impacted the initial period of the game where teams rush to fill up their mobile goal on hand.

CONSIDERATIONS

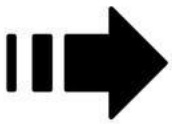
- How fast are we able to make the adjustment?
- How easy is the change?
- Will it affect other portions of the robot?
- Will scoring remain consistent?

SETTING REQUIREMENTS

Simplicity - "How easy is the change?"
Simplicity allows for easier tuning as well as quick troubleshooting. Not impacting other working components of the robot also saves us time like our 2nd requirement states.

Fast change – "How fast are we able to make the adjustment?"
The faster the adjustment, the more time we can spend on coding in order to have all programs completed before the upcoming tournament

Speed – "Will scoring remain consistent?"
High speed and efficiency in intaking and redirecting rings will be preferred. This allows for maximization of time allowing us to score more and smoothly as well.



CONSTRAINTS

RULE CONSTRAINTS:
As per <R4> the intake must fit within an 18" x 18" x 18" volume

MATERIAL CONSTRAINTS:
We have full access to all potential materials as it is the start of the season

TIME CONSTRAINTS:
This subsystem should be completed by November 8th

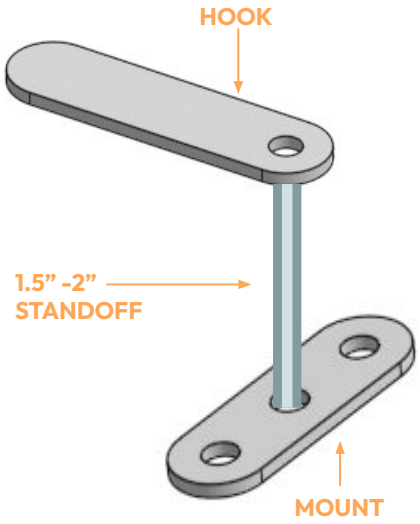
INTAKE 1.1

BRAINSTORM

AUTHOR; RACINE LIU

DATE; 11.04.2024

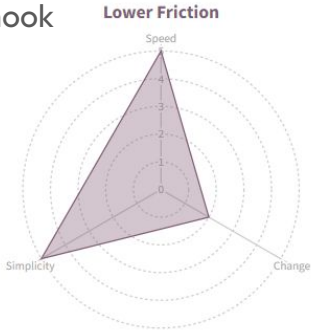
SOLUTIONS

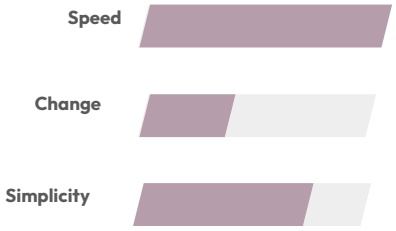


LOWERING INTAKE FRICTION

Pros: Better pickup from floating stage intake and optimal speed

Cons: Not guaranteed consistent scoring due to the shape of the hook

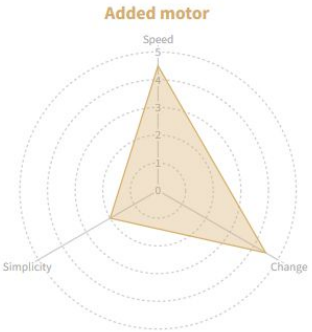


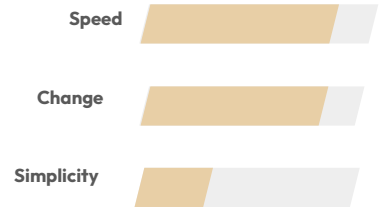


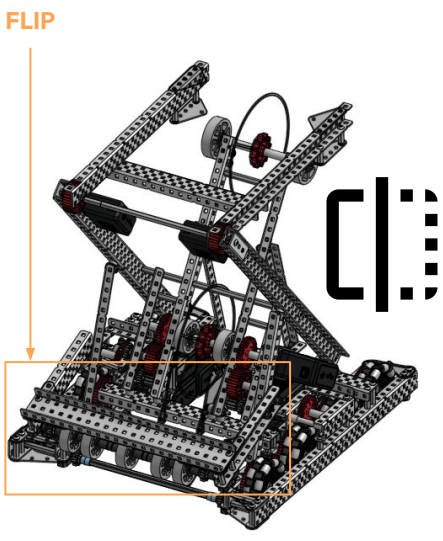
ADDING 1 MORE MOTOR

Pros: Strong intake easy fix

Cons: May impact the arm as we'd have to remove a motor in order to fulfill rules



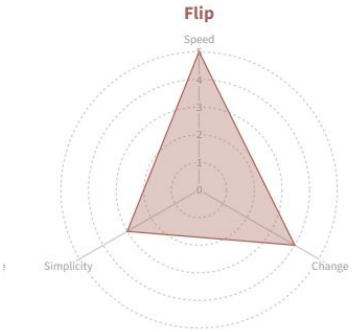


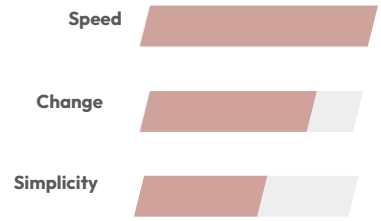


FLIPPING FLOATING STAGE

Pros: Faster and more consistent pickup

Cons: Time consuming





AUTHOR; RACINE LIU

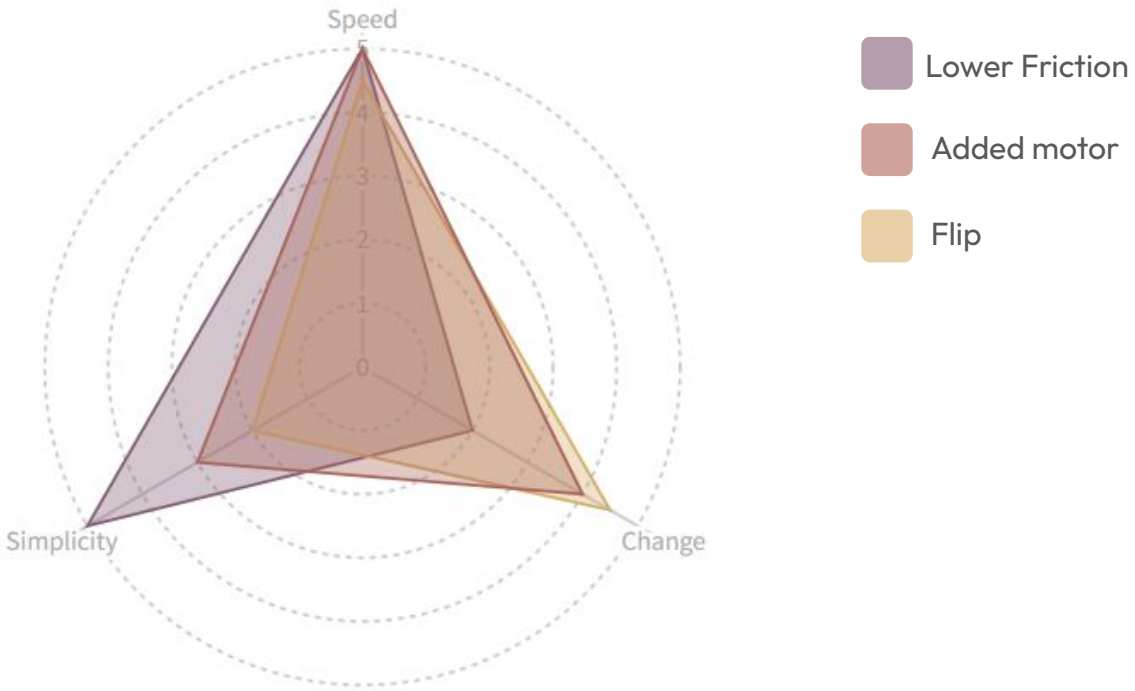
DATE; 11.05.2024

INTAKE 1.1

EVALUATE

RAW RANKING VISUALIZATION

When viewing the raw visualization, it seems unclear of the majority.



RANKING

We rank each type of intake on a scale of 1-5 with 5 being the highest and 1, the lowest. Weighting will also be given in terms of what we think is most important by increments of 2. We then total the ratings to get an idea of the highest score.

Criteria	Weight	Lower Friction		Added motor		Flip	
		Rating	**Total**	Rating	**Total**	Rating	**Total**
Speed	5	5	25	4.5	22.5	5	25
Change	3	2	6	4.5	13.5	4	12
Simplicity	1	5	5	2	2	3	3
Total		36		40		40	

After observing the raw ranking and the decision matrix, it seems that 2 options are tied. As they both don't take much time to complete, we've decided to go with both options.

INTAKE 1.1

PLAN

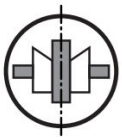
AUTHOR; RACINE LIU

DATE; 11.05.2024

SPECIAL SYMBOLS


Make sure parts can rotate freely


Make sure gears are properly engaged

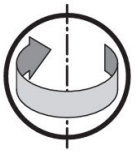

Assemble symmetrically


Special attention in assembly

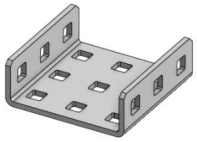
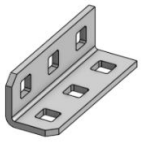
Complete 2 pieces

Assemble step 3 according to step 1


Assemble mirrored


Flip

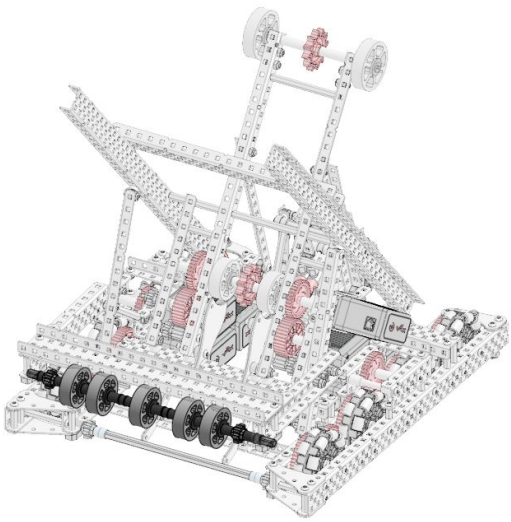
PARTS LIST

Item	5.5W motor	1x3x1x3 C-channel	1x1x1x3 Angle	⅜" Screw	1.5" Standoff
Part ID	276-4842	276-4359	276-6484	276-4991	276-2013
Illustration					
Quantity	1	1	1	6	1

STEP 1:

Remove floating stage structure from the robot in order to flip the parts.
Disassemble all leaving the HS shaft

Leave intake for next step →



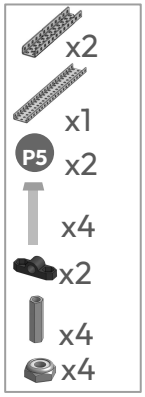
AUTHOR; RACINE LIU

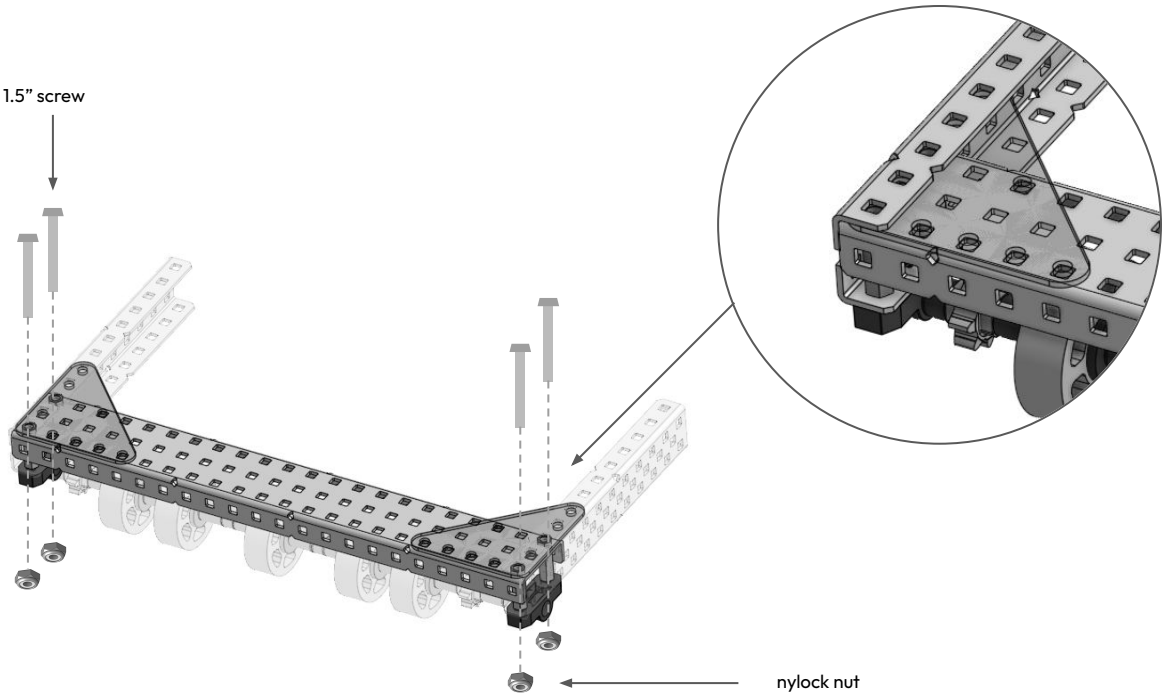
DATE; 11.06.2024

INTAKE 1.1

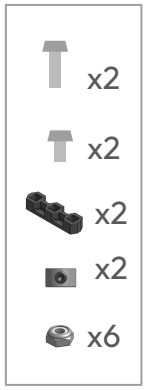
STEP BY STEP

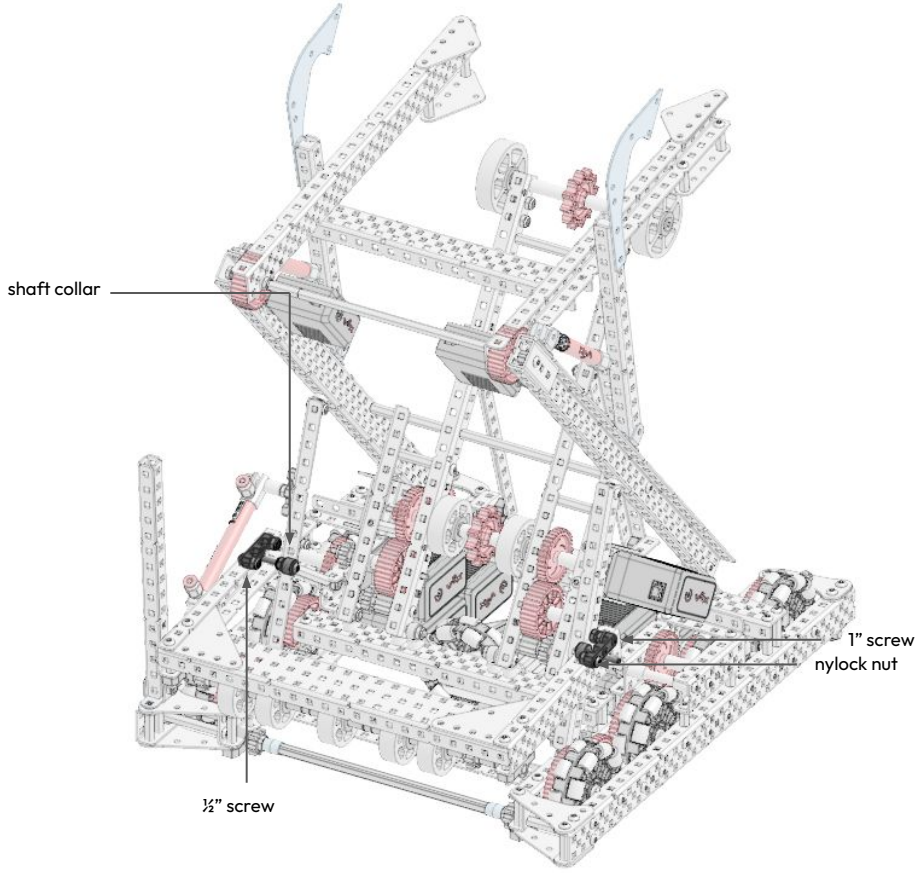
STEP 2:





STEP 3:





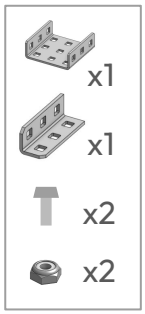
INTAKE 1.1

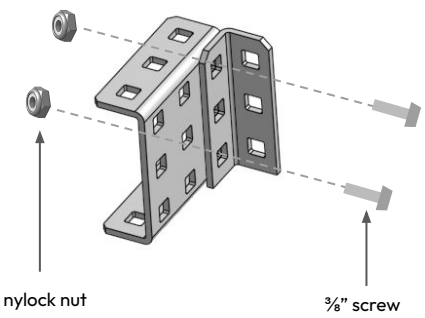
AUTHOR; RACINE LIU

DATE; 11.06.2024

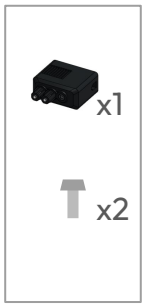
STEP BY STEP

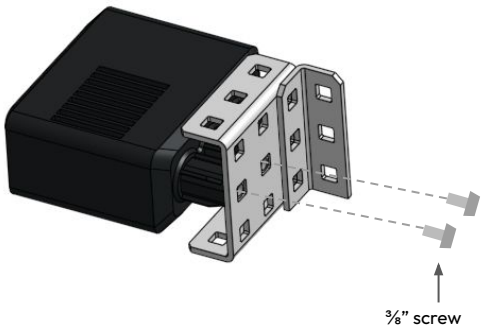
STEP 4:



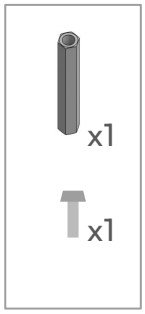


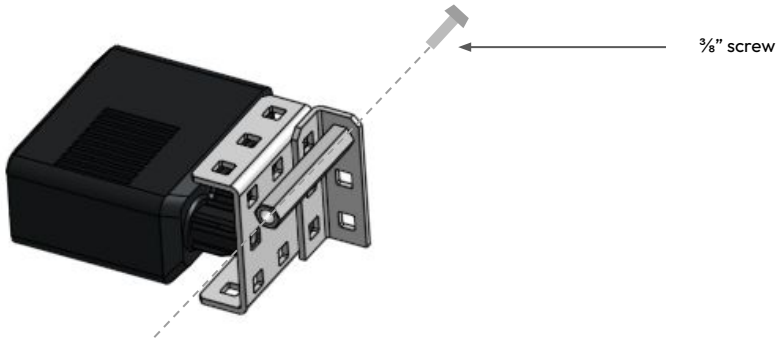
STEP 5:



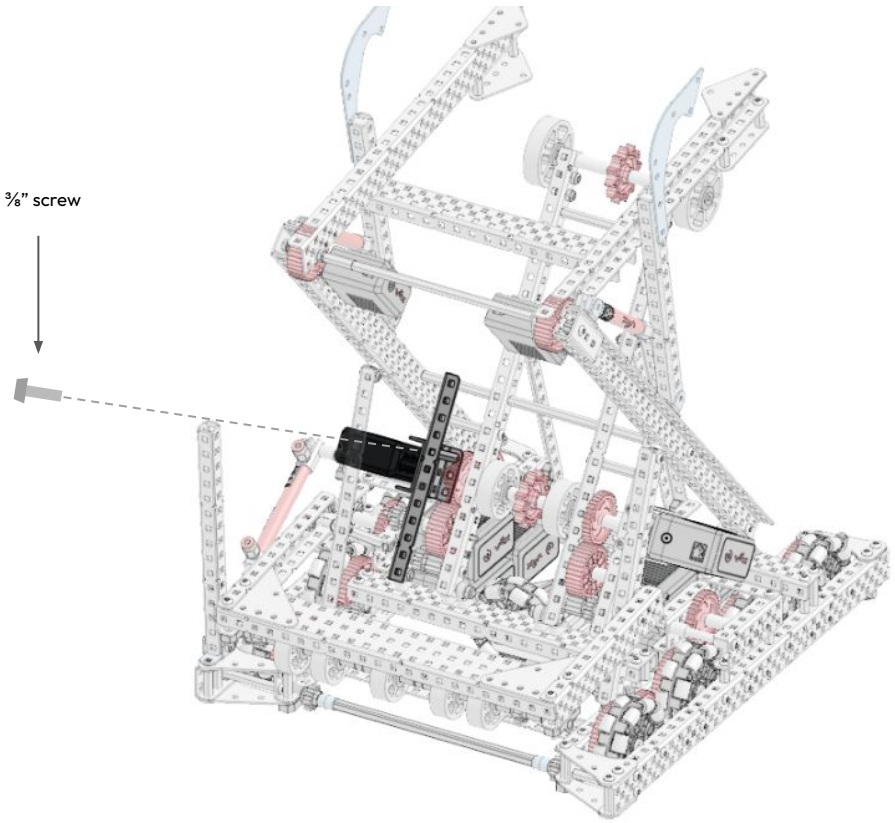


STEP 6:





STEP 7:

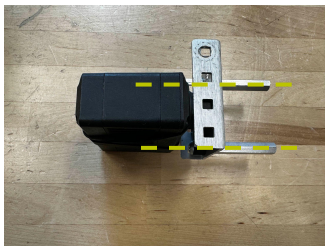


AUTHOR; RACINE LIU

DATE; 11.07.2024

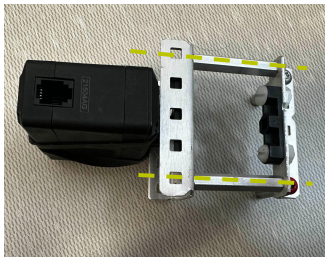
INTAKE 1.1

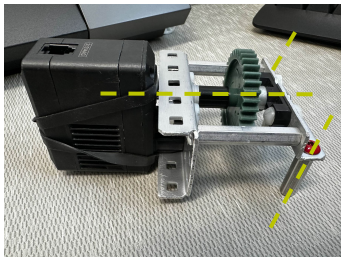
BUILD



When building we decided to add the 2nd standoff as it gives more support when mounted on top of the intake.

When the motor spins some shaking occurs so this way it'll ensure that they stay secure through wear over the extensive usage of the intake.

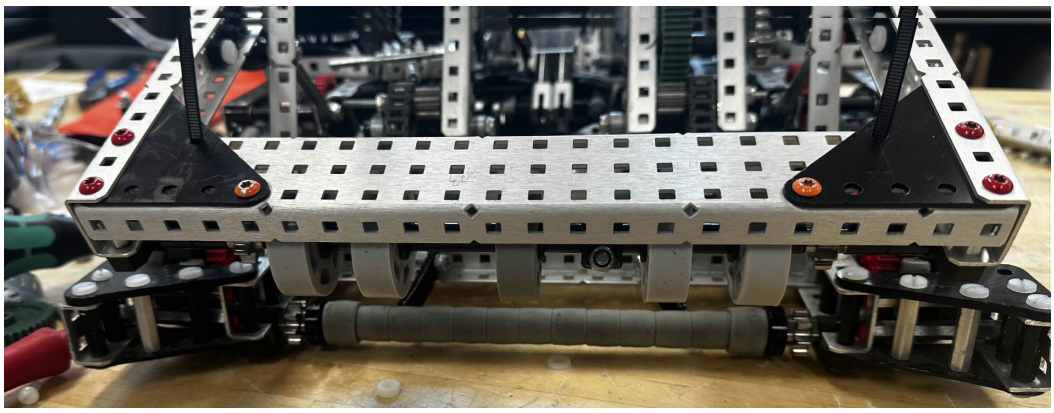




We also realized that since the 5.5w motor only supports low strength axles we needed to add a low strength 36T to attach onto the intake structure.

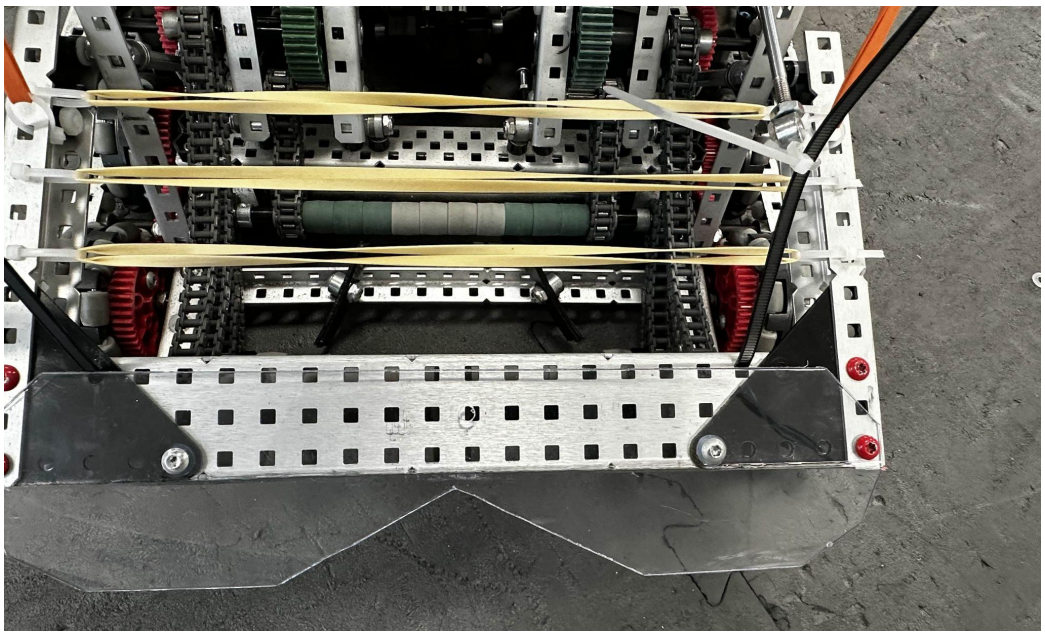
It will be mounted in a way with the 2 other 1" standoffs to be secure.

By flipping the intake position the flex wheels are closer to the ground making better pickup. They C-Channel facing inwards makes it harder to bend



We decided to add a wedge that doubles as an aligner. The wedge allows us to go for rings in the corner while the aligner sets us up to line up to the wall stake significantly decreasing the amount of time we need to allocate for scoring.

The 3 elastic bands help prevent the rings from falling out as the hook picks up the ring.



INTAKE 1.1

TEST

AUTHOR; RACINE LIU

DATE; 11.08.2024

TEST NO.1 SPEED

Requirements:
We want to see how much time the robot takes to fill up a mobile goal with a stronger intake

Materials:
Robot, field, intake, mogo, and 6 rings

PROCEDURE

1. Set up field tiles in the same way as the vex field (12x12)
2. Place the robot on the field
3. Clamp a mogo
4. Line up 6 rings in a row
5. Start stopwatch
6. Intake
7. Stop stopwatch
8. Repeat 10 times

RESULTS

QUANTITATIVE DATA: SUCCESS

Test	1	2	3	4	5
Time	6.16s	6.21s	6.20s	6.24s	6.18s

QUALITATIVE DATA:

Intaking is smooth with no jams both on the 6th ring and while intaking. The anti stall triggers and reverses allowing for smooth intake. The robot is a second faster than the previous iteration of the intake which means the change is a success

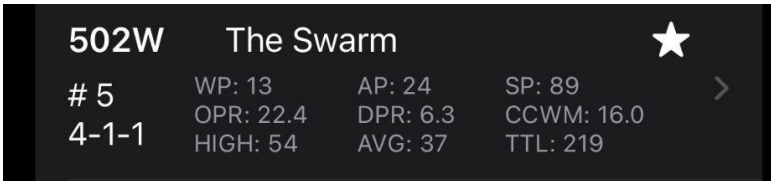
AUTHOR; RACINE LIU

DATE; 11.20.2024

TOURNAMENT 2 OVERVIEW

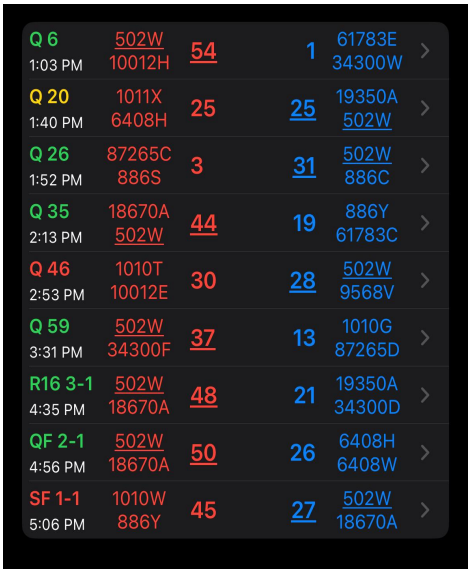
RANKING

We were placed 5th in qualifications. Winning the design award, but we are happy with our skills score especially since we went into the competition without a skills autonomous program coded.

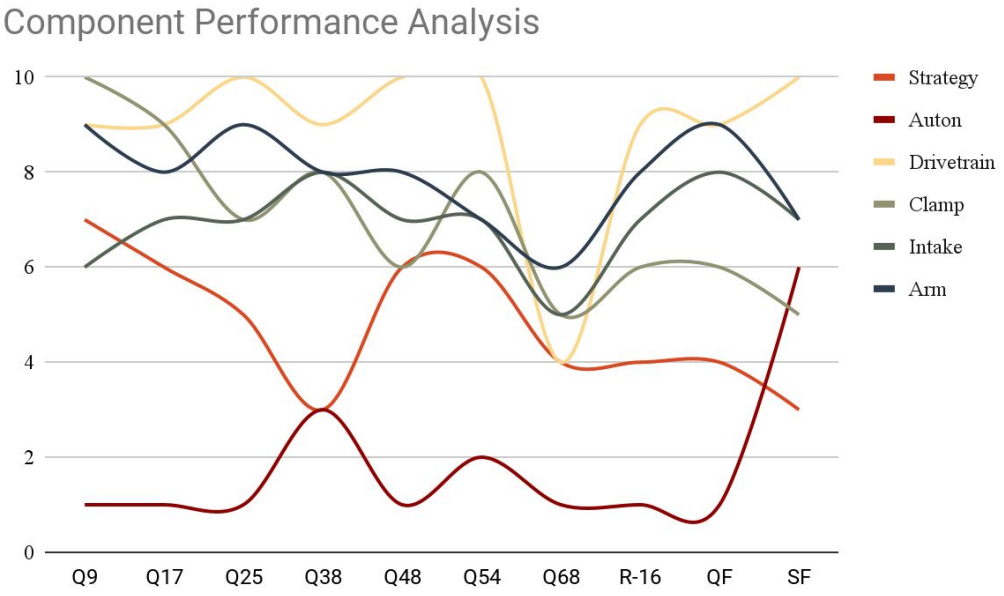


COMPONENT OVERVIEW

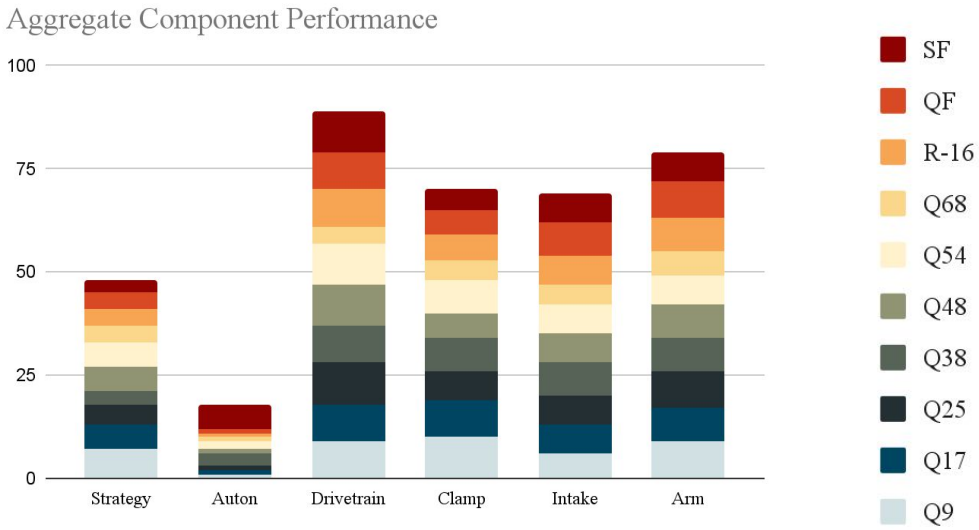
Below is a chart of our component performance (including strategy, autonomous, drivetrain, clamp, intake, and arm).



Q 6 1:03 PM	502W 10012H	54	1	61783E 34300W
Q 20 1:40 PM	1011X 6408H	25	25	19350A 502W
Q 26 1:52 PM	87265C 886S	3	31	502W 886C
Q 35 2:13 PM	18670A 502W	44	19	886Y 61783C
Q 46 2:53 PM	1010T 10012E	30	28	502W 9568V
Q 59 3:31 PM	502W 34300F	37	13	1010G 87265D
R16 3-1 4:35 PM	502W 18670A	48	21	19350A 34300D
QF 2-1 4:56 PM	502W 18670A	50	26	6408H 6408W
SF 1-1 5:06 PM	1010W 886Y	45	27	502W 18670A



Overall, we had poorly run strategies as well as non-working autonomous programs as the robot would not be able to consistently score wall stakes. Our clamp performance decreased throughout the competition, and we found that it had difficulty clamping onto the goal in the autonomous period of games.



AUTHOR; RACINE LIU

DATE; 11.21.2024

TEAM MEETING #3

MAKING PLANS

TEAM REFLECTION

We realized yet again that our game strategy was flaweds.

Since auto is a 12 point swing and stakes are a 5 point swing. We will focus on bettering our game sense as well as making better autonomous programs.

AGENDA

- Determining next tournament
- Figure out team roles in the design process
- Long term and short term goal setting
- Game strategy development
- Making new schedule

ATTENDANCE

Full team: Vaughan Sandquist, Racine Liu, Aaron Lew, Aiden Tam

EVENT DATES

Tournament	Date	Level
WPRA Season Opener Qualifier	Sep 30	Blended
WPRA Halloween Qualifier	Oct 19	Blended
PYRS Seaquam Season Opener	Nov 2	Blended
WPRA Fall Qualifier	Nov 10	Blended
PYRS Salish Fall Qualifier	Nov 16	Blended
PYRS Heritage Woods	Dec 7	Blended
Gord Trousdell Ten Ton Robotics	Dec 15	HS
WPRA New Year Qualifier	Jan 4	Blended
PYRS Southridge Winter Qualifier	Jan 18	Blended
PYRS Surrey Christian Last Chance	Feb 8	Blended
Ten Ton Robotics Blended Last Chance	Feb 15	Blended
WPRA Last Chance Qualifier	Feb 17	Blended
PYRS BC Mainland Regionals	Mar 1	HS

To the left, is a list of future competitions that we are eligible for. We will be trying our best to attend all of them, as competitions are the fastest way to practice and refine our skills. Competing against a variety of teams exposes us to different strategies and approaches, allowing the opportunity to learn from others and improve our own skills. We decided to use the remaining time to better prepare for the next competition. So we'll be attending will be the PYRS Heritage woods qualifier.

GAME STRATEGY

SWINGS

AUTHOR; RACINE LIU

DATE; 12.02.2024

Point swings

team gain + opponent loss

Point swings are how many points a team gains added with the amount of points the opponent loses. Swings are important because they show how well a team can control the game by scoring while also reducing the opponent's score. A big point swing can shift momentum, put pressure on the other team, and make a huge difference in close matches.

Components	Point gain	Opponent point loss	Swing
Autonomous period	6	6	12
Top ring	3	2	5

There are 2 major swings in High Stakes, the first being the autonomous period and the 2nd being top rings.

Winning or losing autonomous (auto) is highly significant in a game where the swing is 12, especially when the next highest possible swing from scoring top rings is only 5. This means the auto swing alone can have over double the impact of a single scoring action in other parts of the game. If a team consistently wins auto, they start with a major advantage that can set the tone for the rest of the match. Conversely, losing auto puts the team at an immediate disadvantage, forcing them to play catch-up for the remainder of the game.

Corners

Another significant aspect of the game that heavily changes the tone of the game are corners. Positive sides double a mobile goals points, while the negative side negates the points on a mogo if it is scored.

Match Priority

1. **Guarantee 1 positive corner and secure it for the rest of the match**
2. **Have control over as many top stakes as possible**

Qualification Rankings

Rankings are calculated based off of many factors. If teams tie in a ranking calculation, the next calculation is used to break that tie and determine their final rankings.

1. **Average Win Points** (WP divided by the number of matches already played by the team)
2. **Average Autonomous Points** (AP divided by the number of matches already played by the team)
3. **Average Strength of Schedule Points (**SP divided by the number of matches already played by the team)
4. **Highest match score**
5. **Second-highest match score**
6. **Random electronic draw**

Win Points

Win Points (WP) are the first basis of ranking teams. Each team receives 0-3 WPs for each Qualification match. Unless a team is DQd, both teams on an alliance always earn the same number of WPs for a match.

- 1 WP is awarded at the end of the autonomous period for each team in an alliance that earns an Autonomous Win Point (AWP) by completing all AWP tasks defined in the game manual. Both alliances can earn this WP in the same match if both alliances complete all tasks.
- 2 WPs are awarded for winning a Qualification match.
- 1 WP is awarded for tying a Qualification match.
- 0 WP are awarded for losing a Qualification match.

Autonomous Points

Autonomous Points (AP) are the second basis of ranking teams. If multiple teams have the same number of WPs, those teams are ranked according to APs. An alliance who wins the autonomous bonus by scoring the most points during the autonomous period of a Qualification match earns 6 APs. In the event of a tie, both alliances will receive three 3 APs.

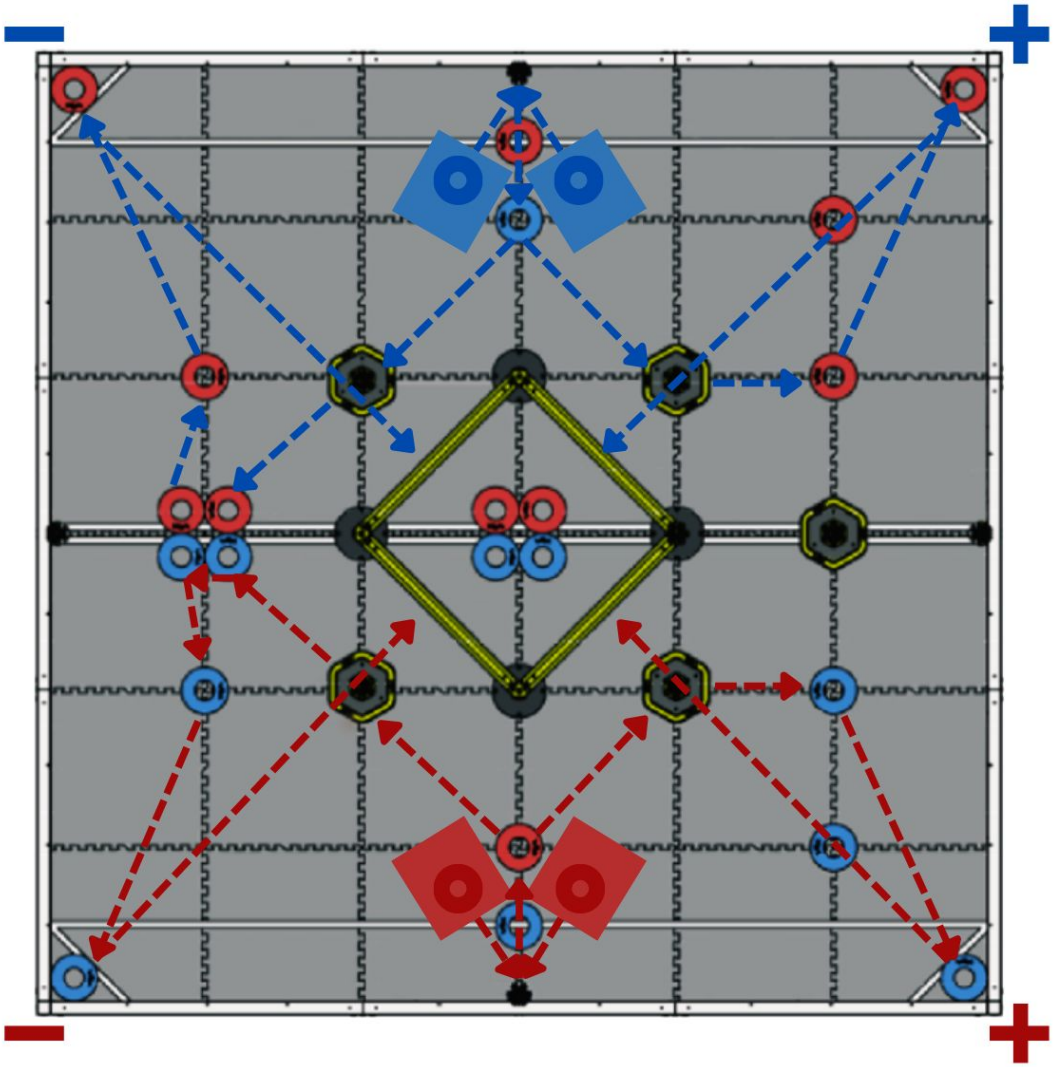
Strength Points

Strength of Schedule Points (SPs) are the third basis of ranking teams. If multiple teams have the same number of both WPs and APs, those teams are ranked according to their SPs. SPs are equivalent to the score of the losing alliance in a Qualification match. In the event of a tie, both alliances receive SPs equal to the tied score. If both teams on an alliance are Disqualified, the teams on the not-Disqualified alliance will receive their own score as SPs for that Match.

AUTHOR; RACINE LIU

DATE; 12.02.2024

Qualification Autos



Ring Side AWP:

= 7 + 3 = 10pts

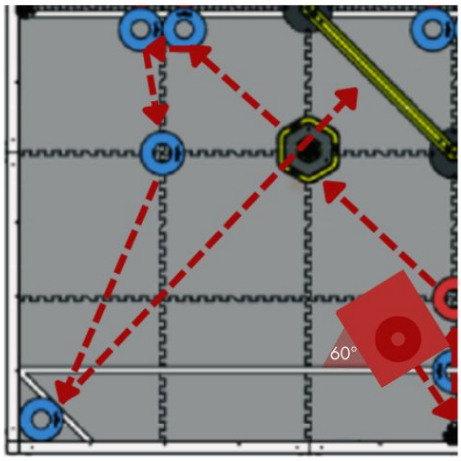
Goal Side AWP:

= 5 + 3 = 8pts

Qualification Autos Explained

Though there are no lines separating the two sides of the field, it is safer to stay within quadrants in order to ensure that no autos clash. AWPs are to be prioritized as they heavily impact rankings in qualifications. AWPs also obtainable in every match, not impacting if a match is won or loss.

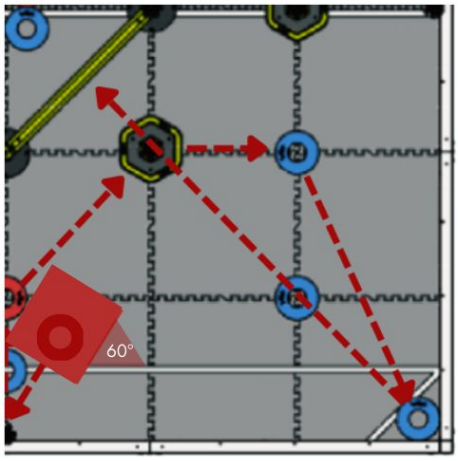
Ring Side Auto



Teams should aim for safe autonomous programs for AWP and to not run risks because guaranteeing the AWP is more important than winning the period. In short teams need programs that are safe yet maximizes points scored.

For instance, we go for the alliance stake first as if we set up the preload a specific way it guarantees more consistency over the intake. We go for the middle stack as opposed to doing a ring rush so little chance of fast disrupts will mess up our program. With this route we are able to score most the rings on the field, with 2 top stakes gaining us 10 points

Goal Side Auto

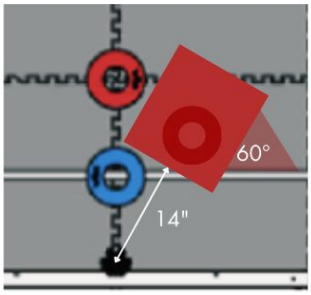


We've chosen not to interact with the 3rd goal at all in the autonomous period.

This is undoubtably the safest auto as it does not interact with anything near the autonomous line. We've chosen to score the alliance stake the same way we go about it on the ring side auto, then we grab the mobile goal and score all the rings available on the positive side (with one ring in the corner left)

Guaranteeing Consistency

When making guides teams should consider that not every field will always be the same, we've noticed alliance stakes are never consistently mounted the same especially due to the field walls tilting. In order to combat the issue we align our robot to the alliance stake a specific distance away every time in order for the ring to be consistently scored without fail.



GAME STRATEGY

ROBOT TYPES

AUTHOR; RACINE LIU

DATE; 12.04.2024

Types of Robots in Region

Throughout the season so far, we've come across 6 types of robots. So far, within BC there are no T3 robots and it's projected that they won't exist until post January

1. **Drivetrain**
2. **Drivetrain + Clamp**
3. **Drivetrain + Clamp + Intake**
4. **Drivetrain + Clamp + Intake + T1**
5. **Drivetrain + Clamp + Intake + Wall mechanism**
6. **Drivetrain + Clamp + Intake + Wall mechanism + T1**

In games, if alliance has a non functioning mechanism they are moved down a class.

When doing match strategy, opponents are always assumed to be Type 6

Selecting What Autos to run

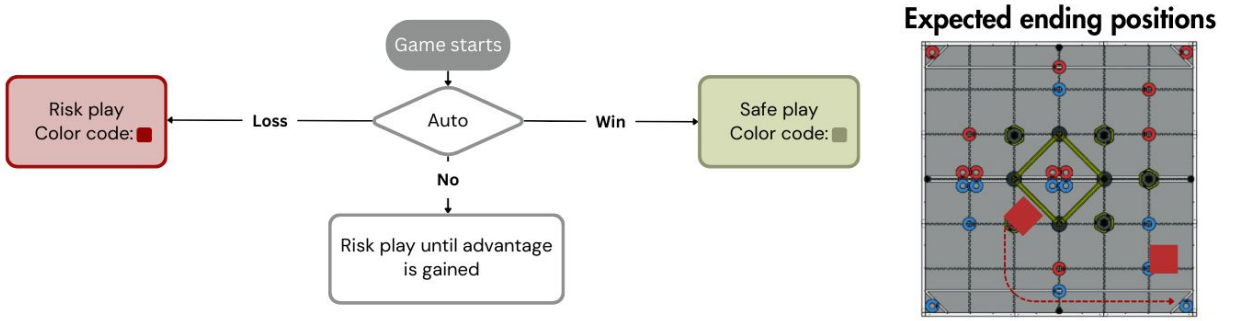
Alliances should focus on the consistency of the programs. Before games, contact the other alliance and discuss what autos they have in order to not clash. Always run the most consistent auto and if possible the ones that score the most points.

- If an alliance has a solo AWP program run the two simultaneously, it's never good to fully trust that it will go through.
- If an alliance does not have an auto, request them to code the robot to drive off the line
- If both of the alliance autos are the same, whoever scores the most points on ring sides will run their program
- If alliance no shows, run the autonomous program that scores the most points

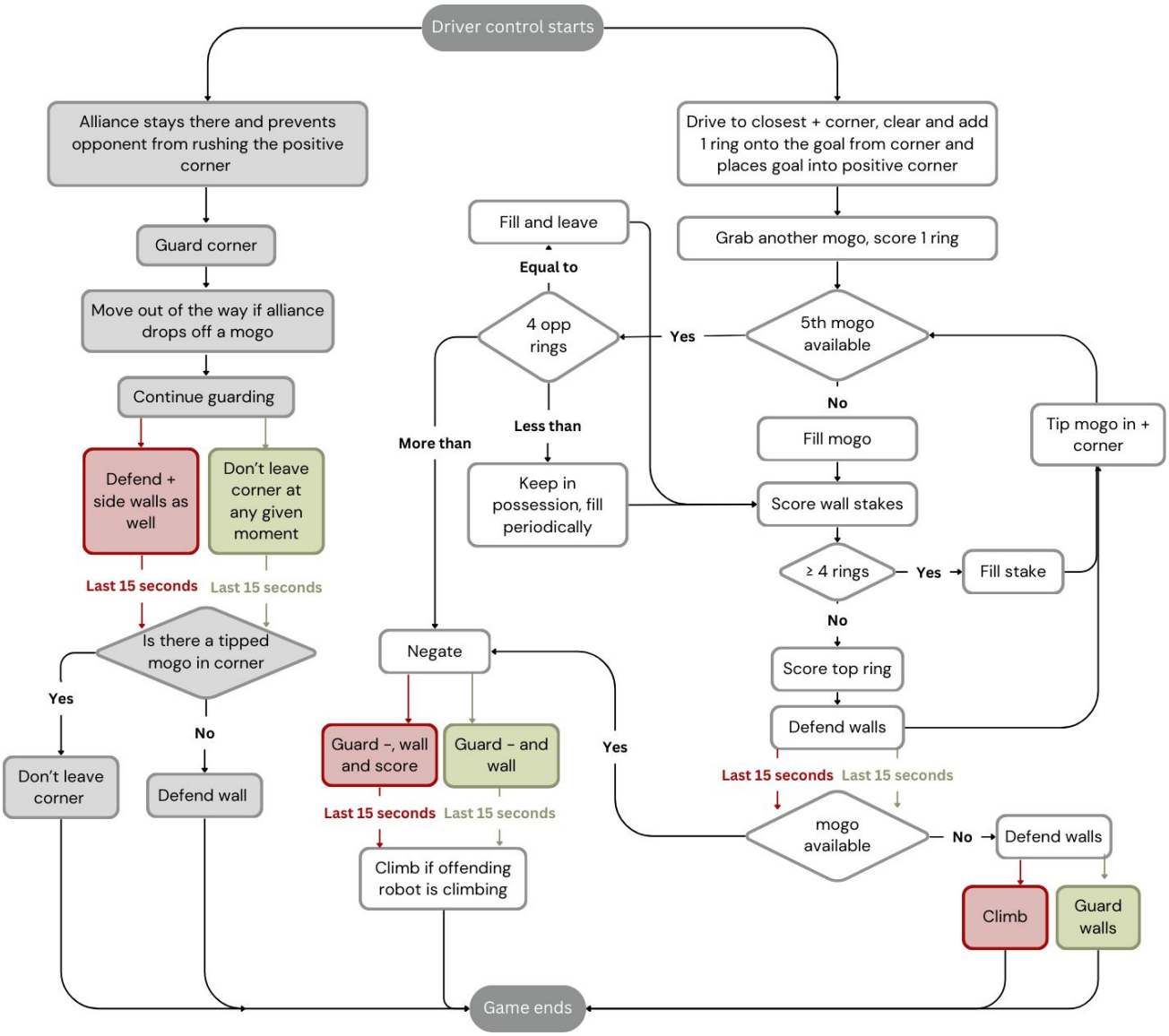
Type 1 Alliance: Drivetrain

☐ Ring side: Our team
☐ Goal side: Alliance

Have the alliance partner set up their robot where they are right beside the + corner and then driving off the line



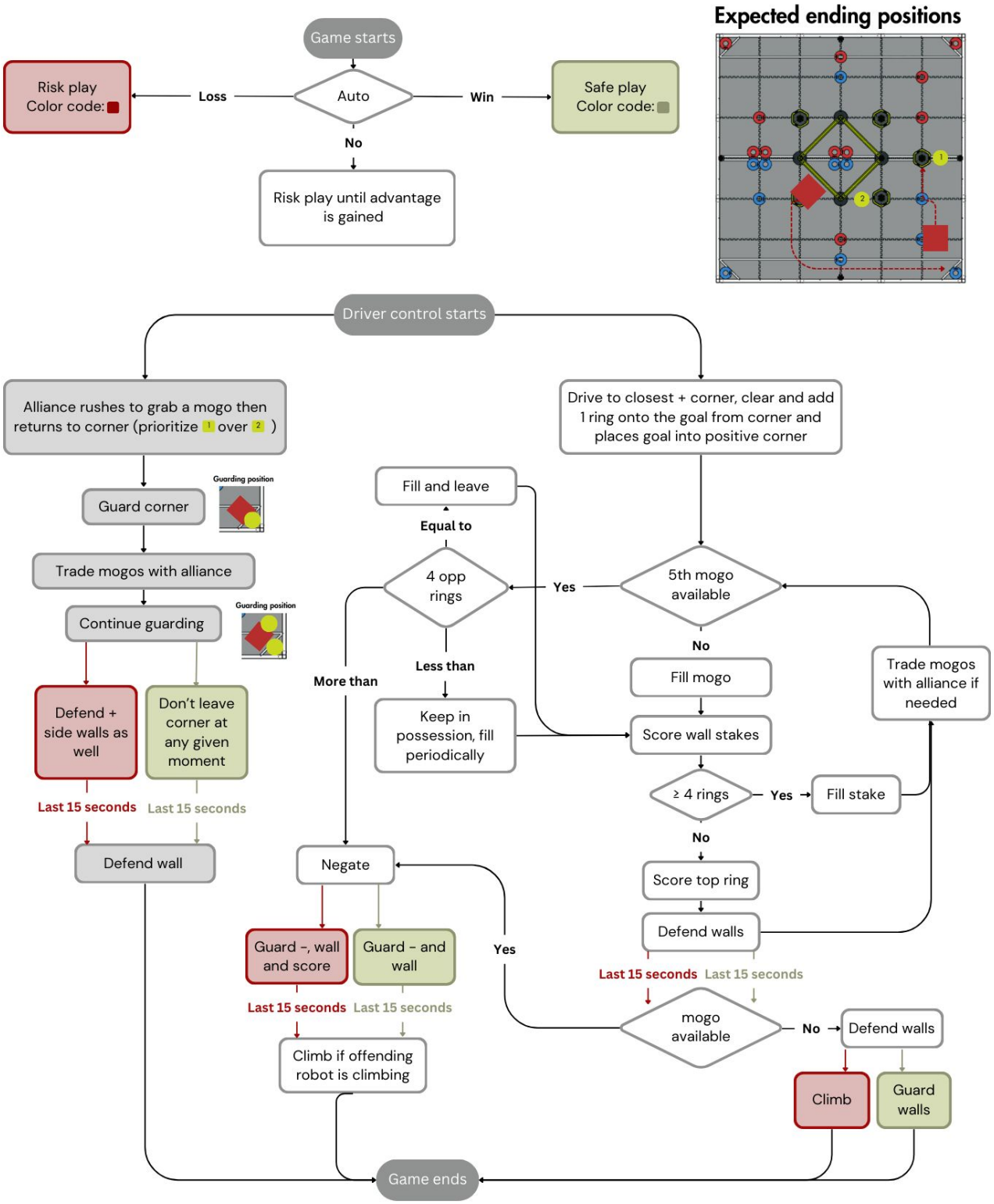
Expected ending positions



Type 2 Alliance: Drivetrain + Clamp

- ☐ Ring side: Our team
- ☐ Goal side: Alliance

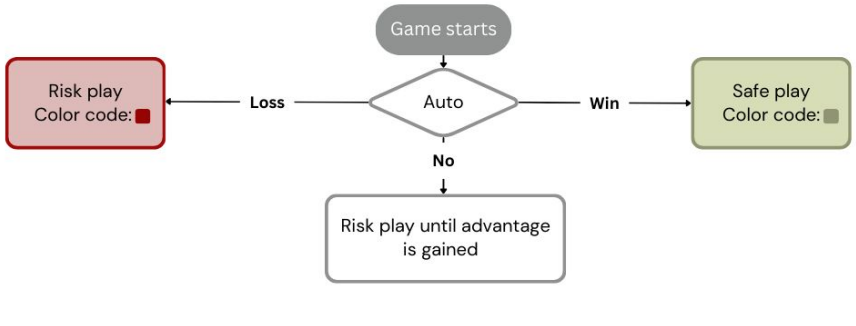
Have the alliance partner set up their robot where they are right beside the + corner and then driving off the line



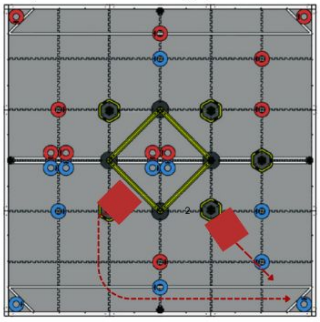
Type 3 Alliance: Drivetrain + Clamp + Intake

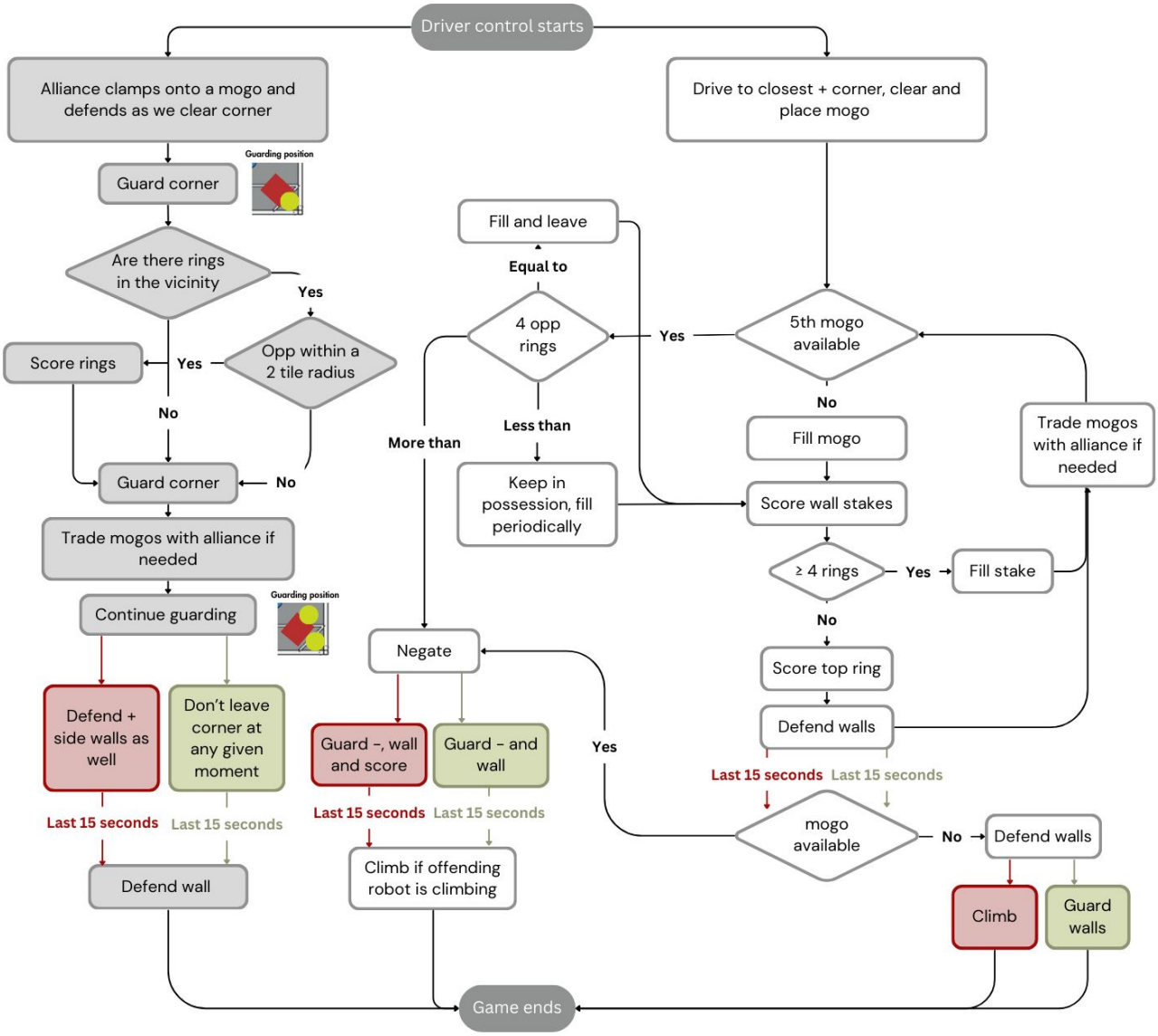
- ☐ Ring side: Our team
- ☐ Goal side: Alliance

If Alliance partner has the same autos on both sides they should be running the goal side for autonomous



Expected ending positions



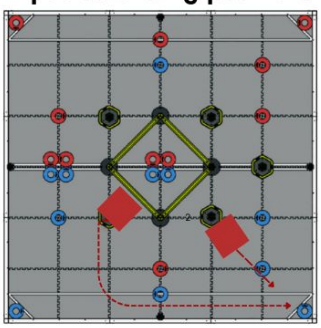


Type 3 Alliance: Drivetrain + Clamp + Intake

☐ Ring side: Alliance
☐ Goal side: Our team

If Alliance partner has a more consistent ring side autonomous they run that program

Expected ending positions



Game starts

Auto

Loss → Risk play Color code:

Win → Safe play Color code:

No → Risk play until advantage is gained

Driver control starts

Alliance clamps onto a mogo rushes to + corner

Guard corner

Guarding position

Are there rings in the vicinity

Yes → Opp within a 2 tile radius

Score rings

Yes

No → Guard corner

No

Trade mogos with alliance if needed

Continue guarding

Guarding position

Defend + side walls as well

Don't leave corner at any given moment

Last 15 seconds

Last 15 seconds

Defend wall

Drive to closest + corner, clear and add 1 ring onto the goal from corner and places goal into positive corner

5th mogo available

Yes

Should immediately go for mogo 1

No

Fill mogo

Score wall stakes

≥ 4 rings

Yes → Fill stake

No

Score top ring

Defend walls

Trade mogos with alliance if needed

Last 15 seconds

Last 15 seconds

mogo available

No → Defend walls

Climb

Guard walls

Yes

Fill and leave

Equal to

4 opp rings

More than → Negate

Less than → Keep in possession, fill periodically

Guard -, wall and score

Guard - and wall

Last 15 seconds

Last 15 seconds

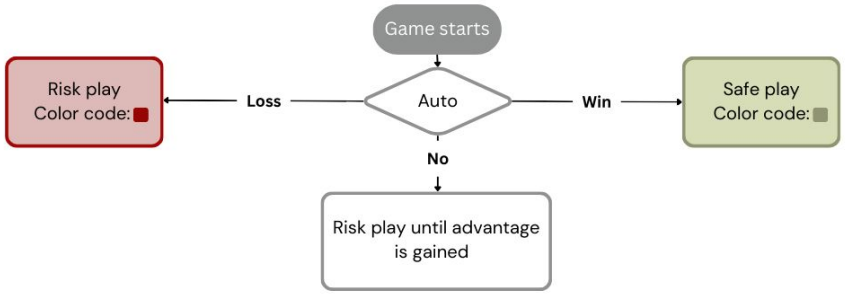
Climb if offending robot is climbing

Game ends

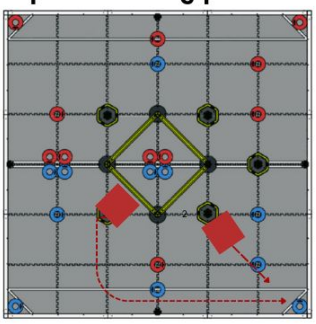
Type 4 Alliance: Drivetrain + Clamp + Intake + T1

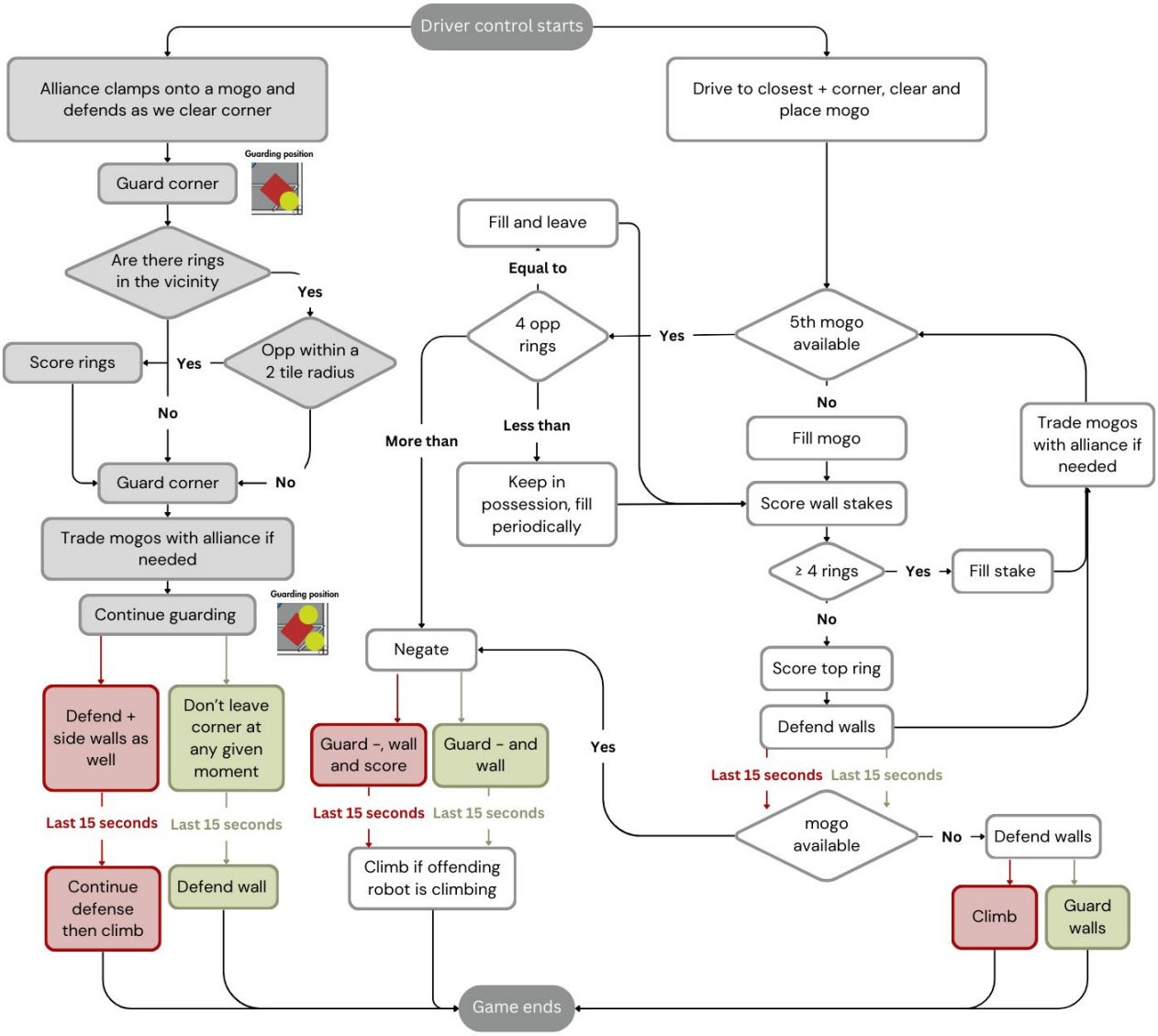
☐ Ring side: Our team
☐ Goal side: Alliance

If Alliance partner has the same autos on both sides they should be running the goal side for autonomous



Expected ending positions





GAME STRATEGY

QUALIFICATIONS

AUTHOR; RACINE LIU

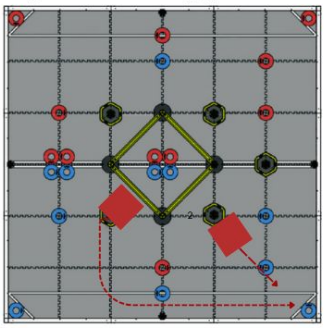
DATE; 12.08.2024

Type 4 Alliance: Drivetrain + Clamp + Intake + T1

☐ Ring side: Alliance
☐ Goal side: Our team

If Alliance partner has a more consistent ring side autonomous they run that program

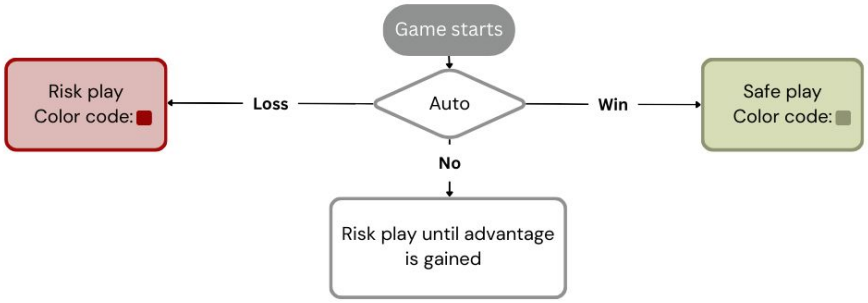
Expected ending positions



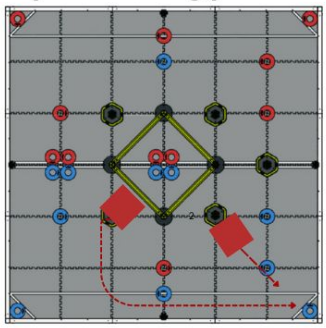
Type 5 Alliance: Drivetrain + Clamp + Intake + Arm

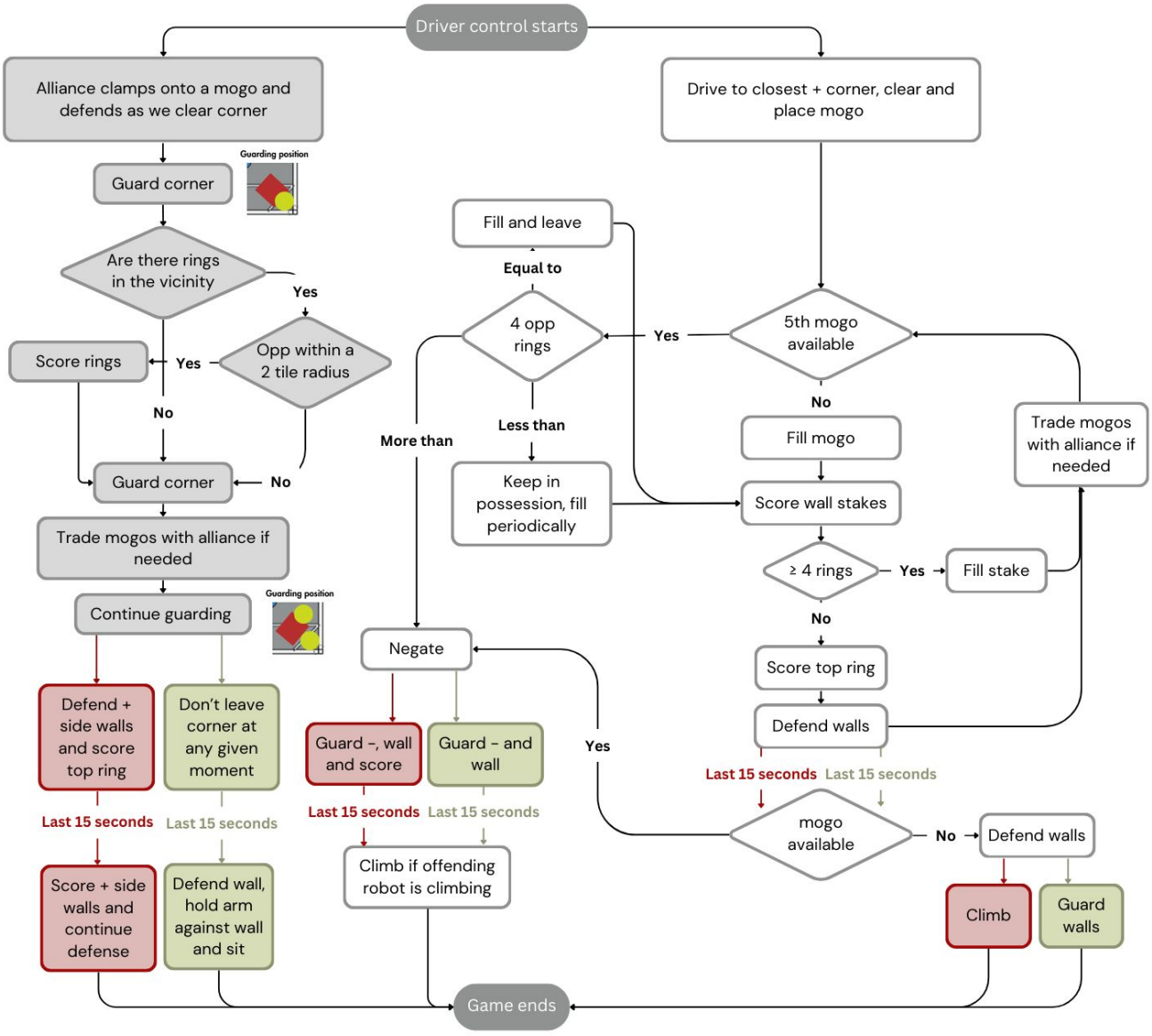
☐ Ring side: Our team
☐ Goal side: Alliance

If Alliance partner has the same autos on both sides they should be running the goal side for autonomous



Expected ending positions

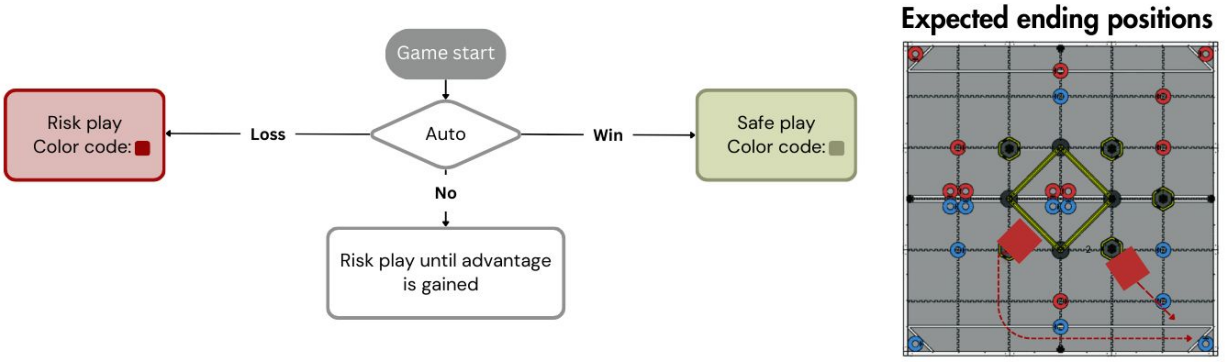


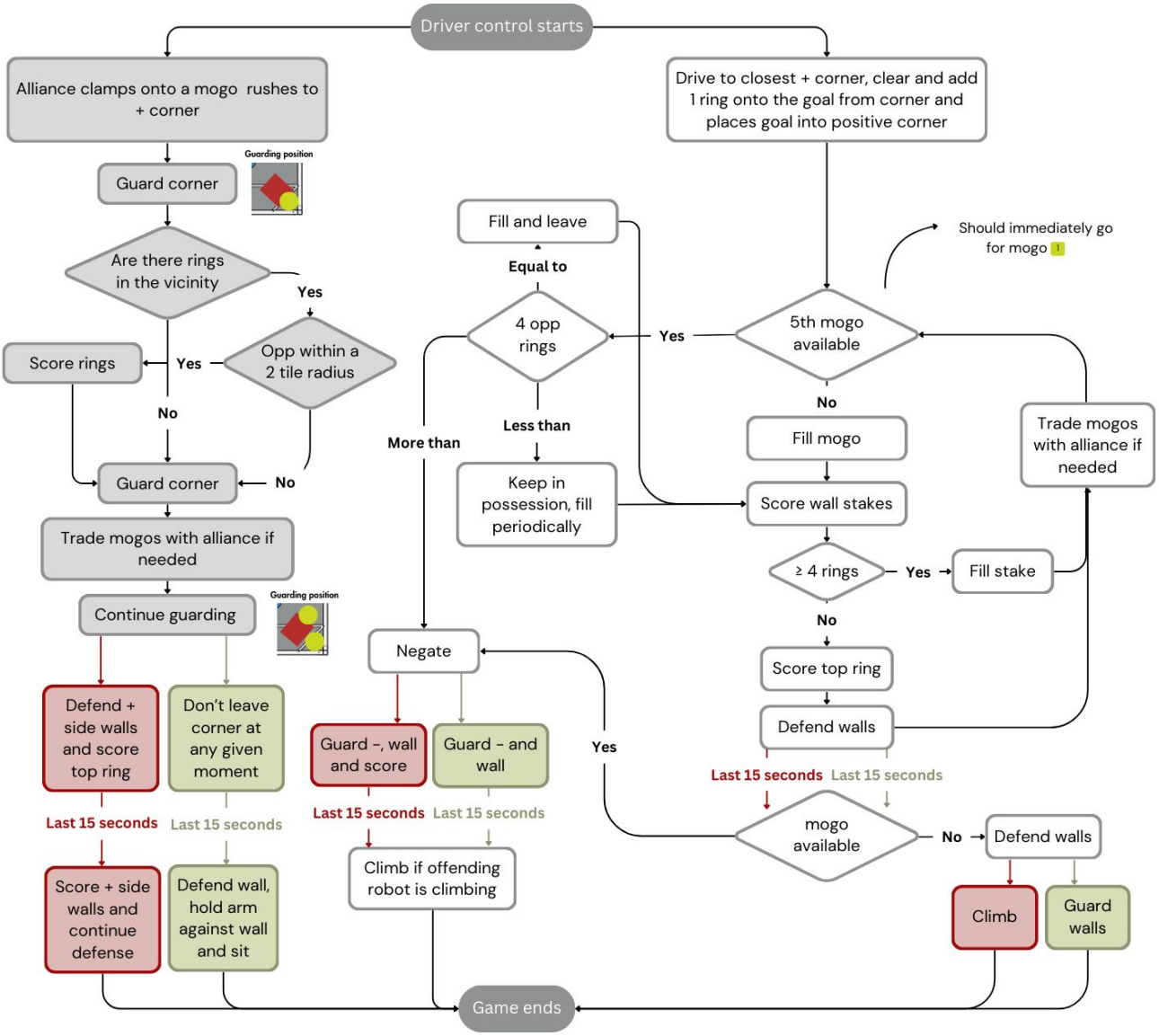


Type 5 Alliance: Drivetrain + Clamp + Intake + Wall

☐ Ring side: Alliance
☐ Goal side: Our team

If Alliance partner has a more consistent ring side autonomous they run that program



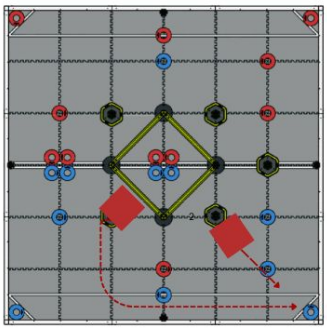


Type 6 Alliance: Drivetrain + Clamp + Intake + Arm + T1

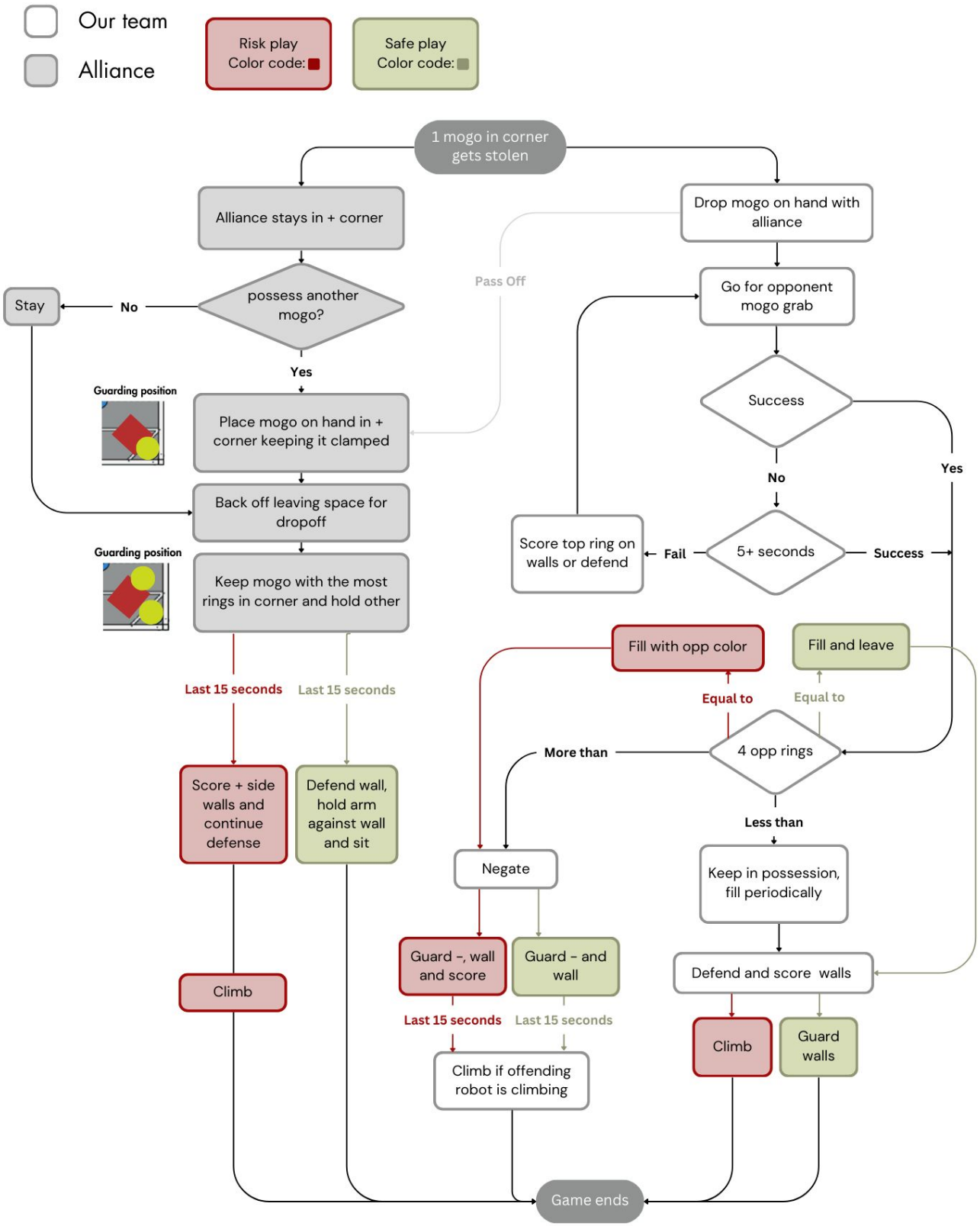
☐ Ring side: Our team
☐ Goal side: Alliance

If Alliance partner has the same autos on both sides they should be running the goal side for autonomous

Expected ending positions

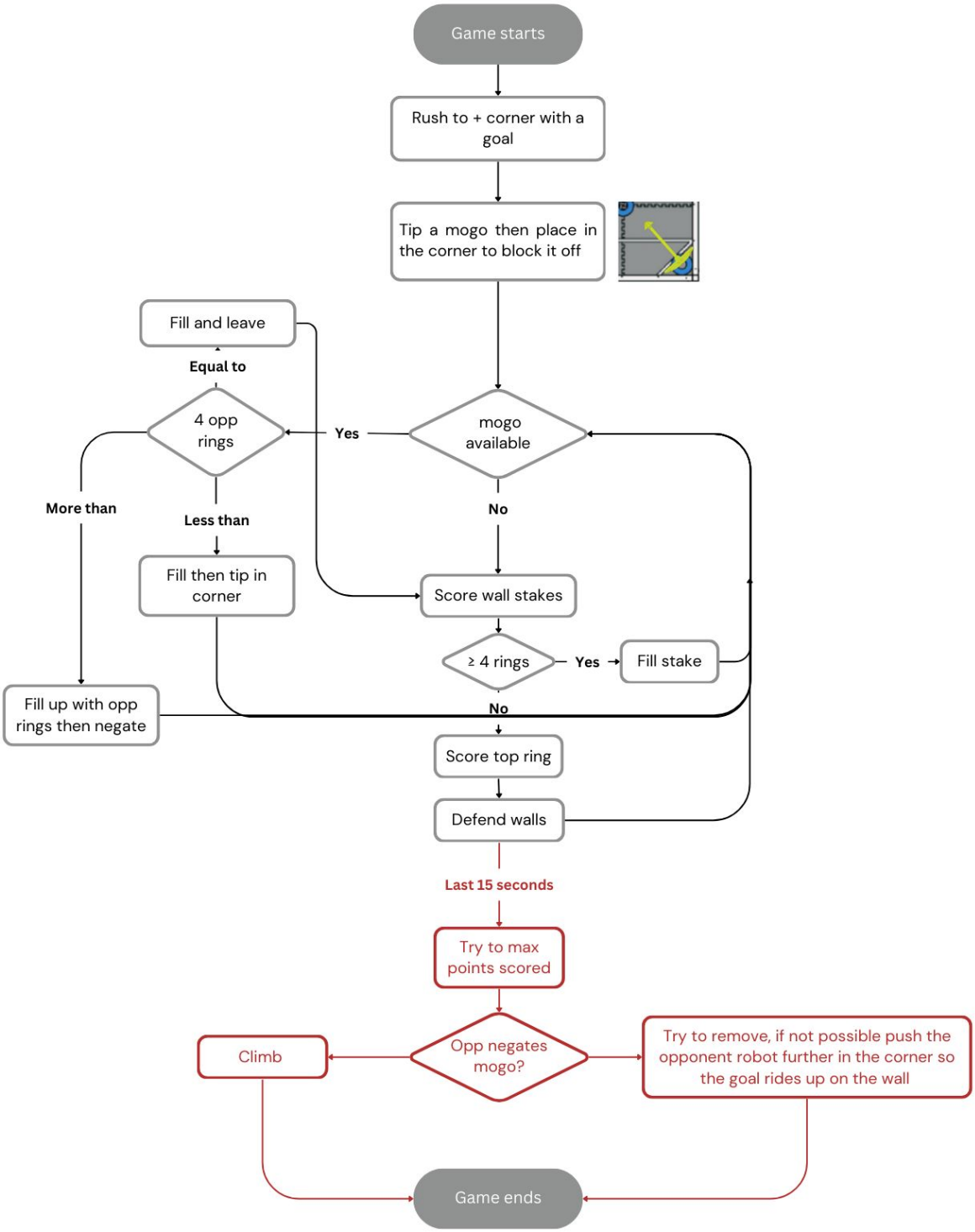


Alliance Owned Mogo Gets Negated



Alliance no shows

If an Alliance partner no shows, first examine the opposing robots when determining what autonomous program to run. Teams should run the side with the most consistent AWP. If both sides are the same, run ring side if deemed autonomous is winnable if not then run goal side at all times



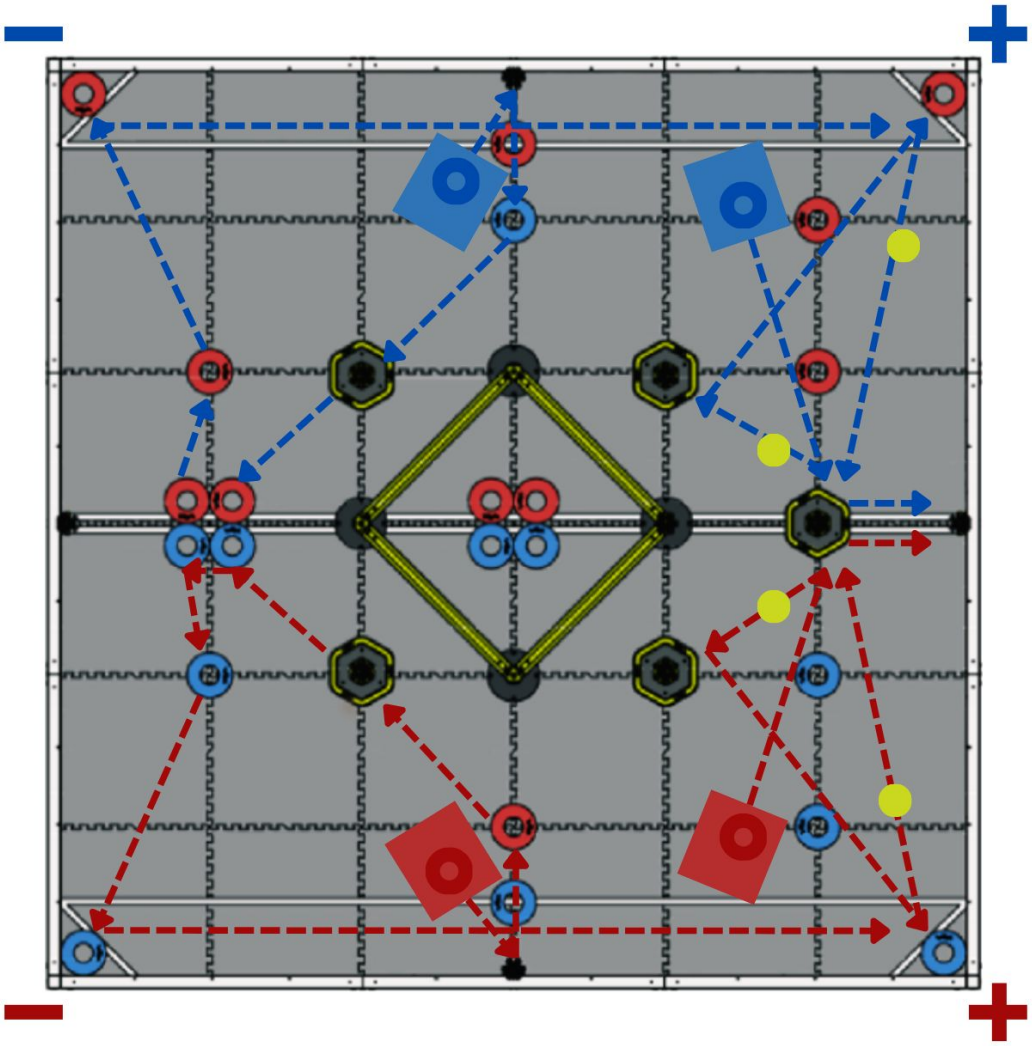
GAME STRATEGY

ELIMINATIONS

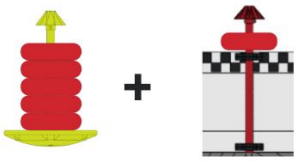
AUTHOR; RACINE LIU

DATE; 12.10.2024

Elimination Autos

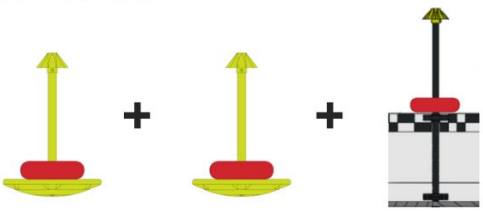


Ring Side Auto:



= 7 + 3 = 10pts

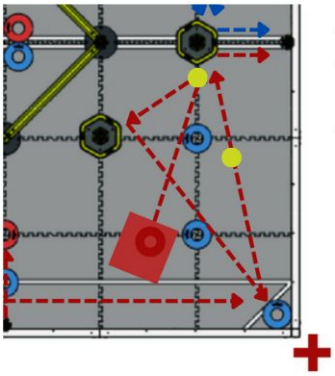
Goal Side Auto:



= 3 + 3 + 3 = 9pts

GAME STRATEGY

ELIMINATIONS



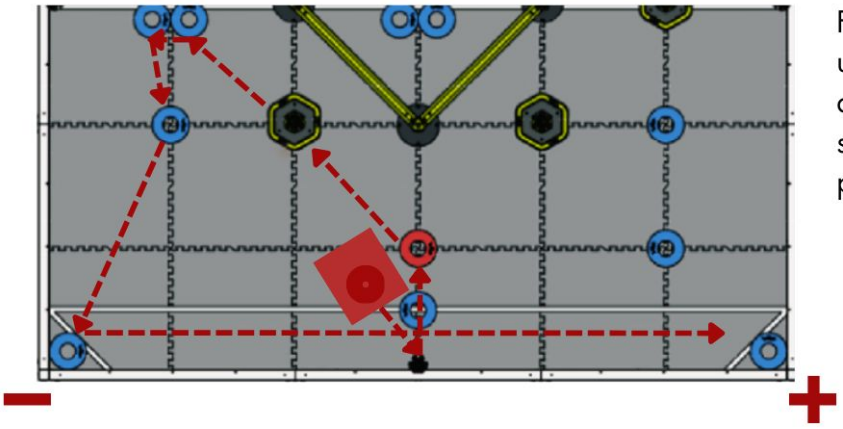
A goal rush is a priority as it is a means of scoring 1 more top ring in the autonomous period.

We've chosen to score all the top rings availible for scoring to max out points.

First we start with the preload in the wall stake arm in order to score it in the autonomous period. The position of the ring is further out in a way that it doesn't naturally load, so we can score without crossing the line.

When the period starts, we raise our arm as the robot moves to the center goal. We first rush to the mobile goal between the two sides, intaking the colored ring on the way. Then we drag the goal back, before dropping the goal and going for the 2nd goal scoring the intaked ring. We pick up the first mogo then score a wall ending the period.

This way we have full control of all 3 mobile goals by the end of the autonomous period, and we are able to start the game by allowing our alliance to place their goal into the positive corner and giving them time to grab the 3rd mogo to hold in possession



For our ring side auto we chose to fill up a mogo as much as possible as well as rushing to the positive corner so we score the most points as well as being prepped for the game

Alliance selection

There are 2 things teams should consider in terms of autos when selecting a teammate.

1. **How many points both autos score altogether**
2. **How it prepares an alliance for the incoming game**

GAME STRATEGY

ELIMINATIONS

AUTHOR; RACINE LIU

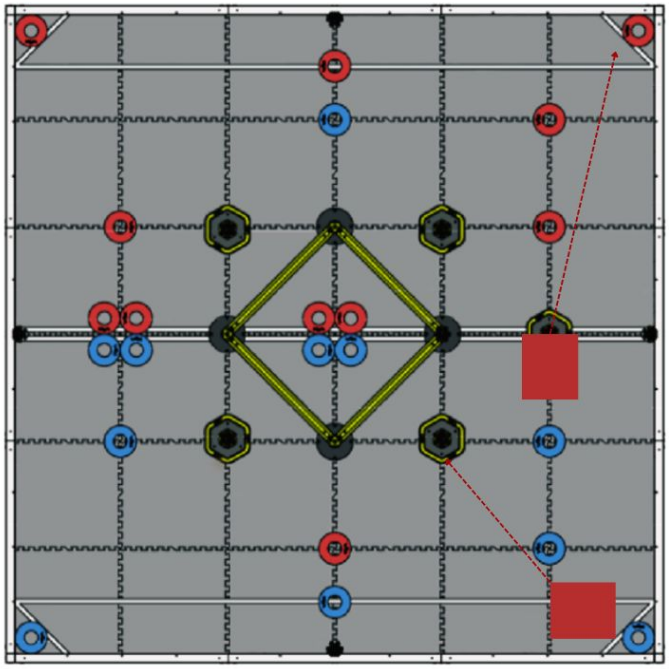
DATE; 12.10.2024

General Stategy

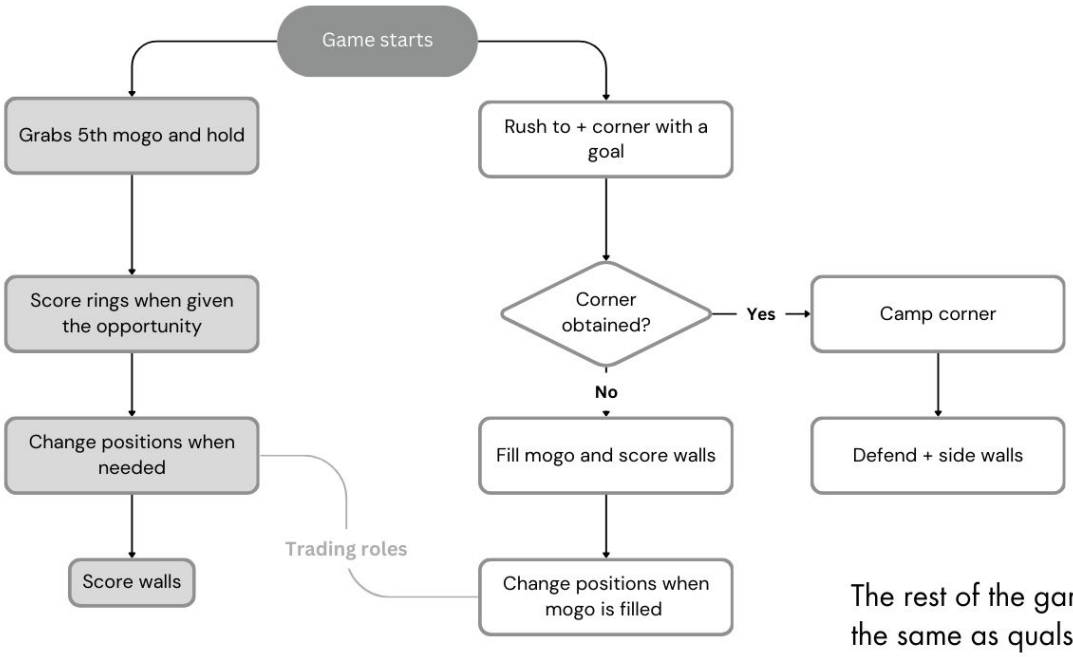
- [ ] Alliance 1
- [ ] Alliance 2

When autonomous period ends, if auto is won and everything hits, the alliance should be able to win the game given that winning the swing and gaining 5th mogo control typically guarantees success.

If auto is loss or even when both corners are lost teams must rely on quick thinking and rely on pressure and baiting. The easiest way to bait is to score walls on the + side and hope 1 team defends leaving corner. Once they leave and defend, one slip up on their end will allow for you to maneuver around to steal a corner.



Beginning of game



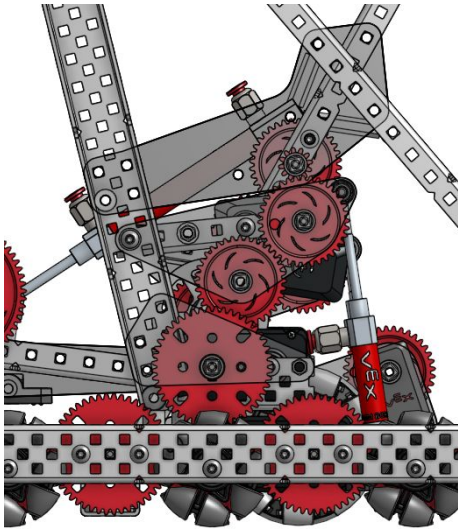
The rest of the game is played out the same as quals

INNOVATE SUBMISSION FORM

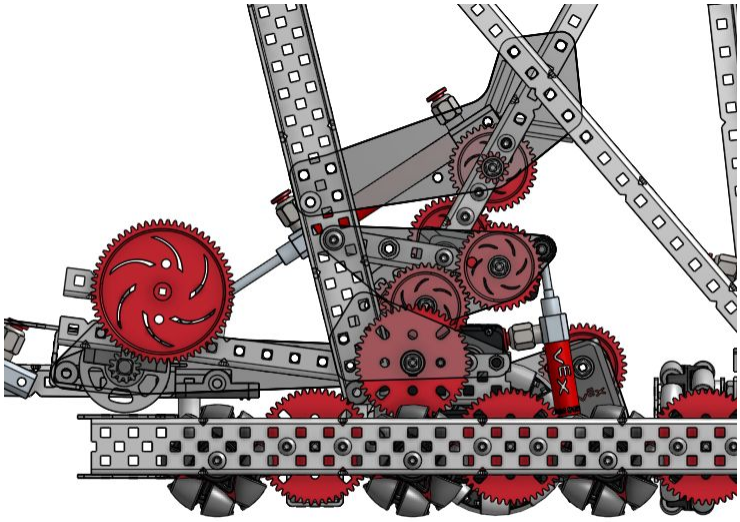
PTO

Driving and maneuverability are key to winning a match. When stealing from corners or even defending them. Being able to quickly block opponents and returning to corner allows for the greatest plays. We went for a faster drive, but a tradeoff for that is lack of power. In order to accommodate that, we added 1 extra motor to the drive.

We noted that we would be over the motor count if we followed with a 1.5 motor intake. Therefore, we are combining and transferring power when each are in the most use.



Motors powering intake



Motors powering drive

2 5.5 motors are mounted on the intake side allowing even power to the drive. We use 2 short pistons on both sides of the drive to transfer power.