Robotics Engineering Notebook

REC Foundation

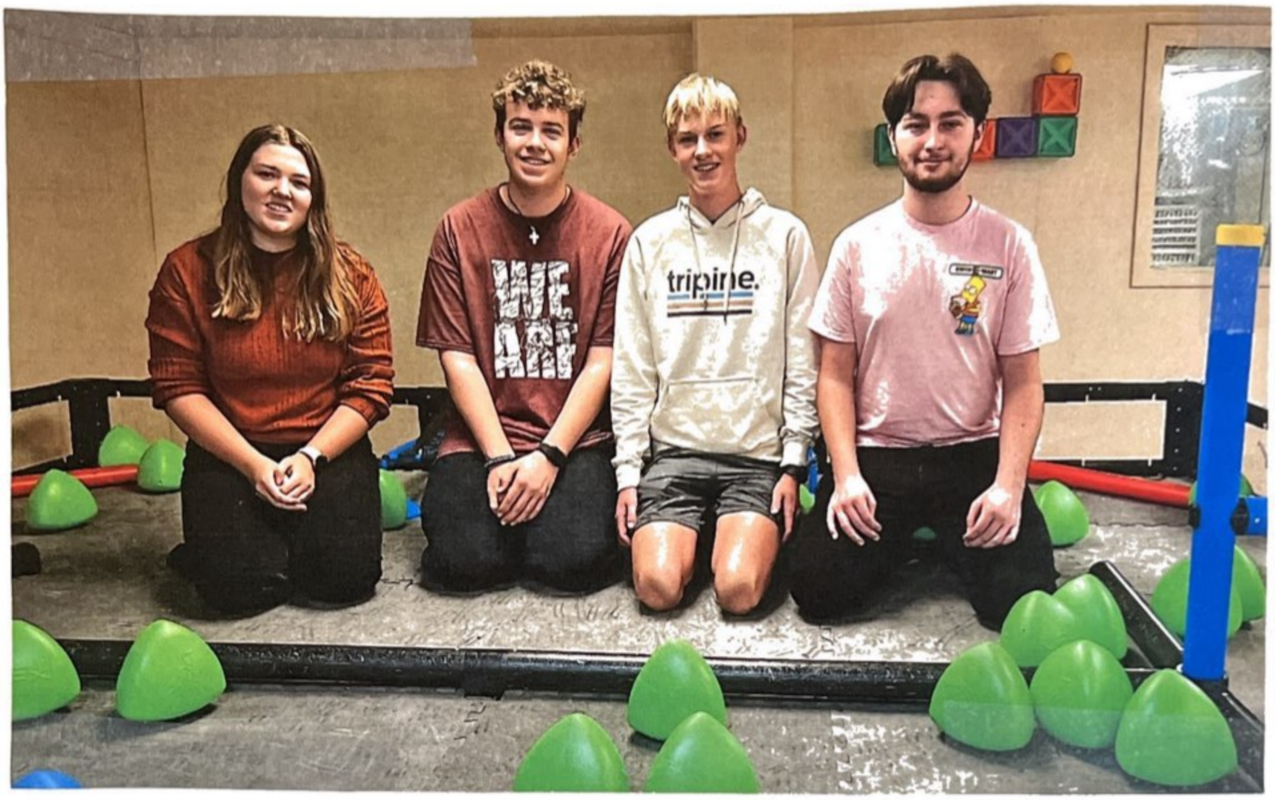
VEX

team name: Validation
team number: 63303V
season: VRC Over Under
start date: 5/16/2023
end date: 9/11/2023
book number: 1 of:

10
9
8
7
6
5
4
3
2
1 inch

Iteration 1
Iteration 2
Programming
Building
Problem

Team Photo



Team Profile

In order from Left to Right:
Vanessa Perkins - Journalist, 12th Grade
Eli Fritts - Builder, 12th Grade
Harris Perkins - Programmer, 10th Grade
Owen Taylor - Driver, 10th Grade

Team Validation

My Projects

Iteration 1

Iteration 2

Programming

Building

Problem

My Projects

Iteration 1 | Iteration 2 | Programming | Building | Problem

Design Process and Award Rubric:

Identify the Problem: Notebook displays the game and all design challenges associated with the rules at the start of every design cycle. Includes goals, words, and pictures.

Brainstorm, Diagram, or Prototype Solutions: Provide at least three labeled diagrams of possible designs including citations from outside sources.

Select Best Solution and Plan: Use testing/decision matrices to explain why designs were or were not chosen with a thorough plan on implementation.

Build and Program Solution: Record in detail steps of building and programming.

Test Solution: All testing steps and results are recorded.

Repeat Design Process: Can be seen that design process has been repeated multiple times to improve performance.

Useability and Completeness: Entire design and development process is recorded in enough clarity and detail to recreate the robot and project's history.

Record of Team and Project Management: Represents complete record of team and project assignments during team meetings including goals and decisions. Design Cycles are easy to identify. Management of time, resources, and budget is mentioned*

☆ Above is a summary of the design cycle and the criteria the notebook will be judged on.
We plan to include every element of the design cycle and rubric in our notebook to recieve the highest score possible.

☆ This page serves as a reminder of the type of notebook our team is striving to achieve this year.

* Information taken from VEX Robotics "Engineering Notebook Rubric".

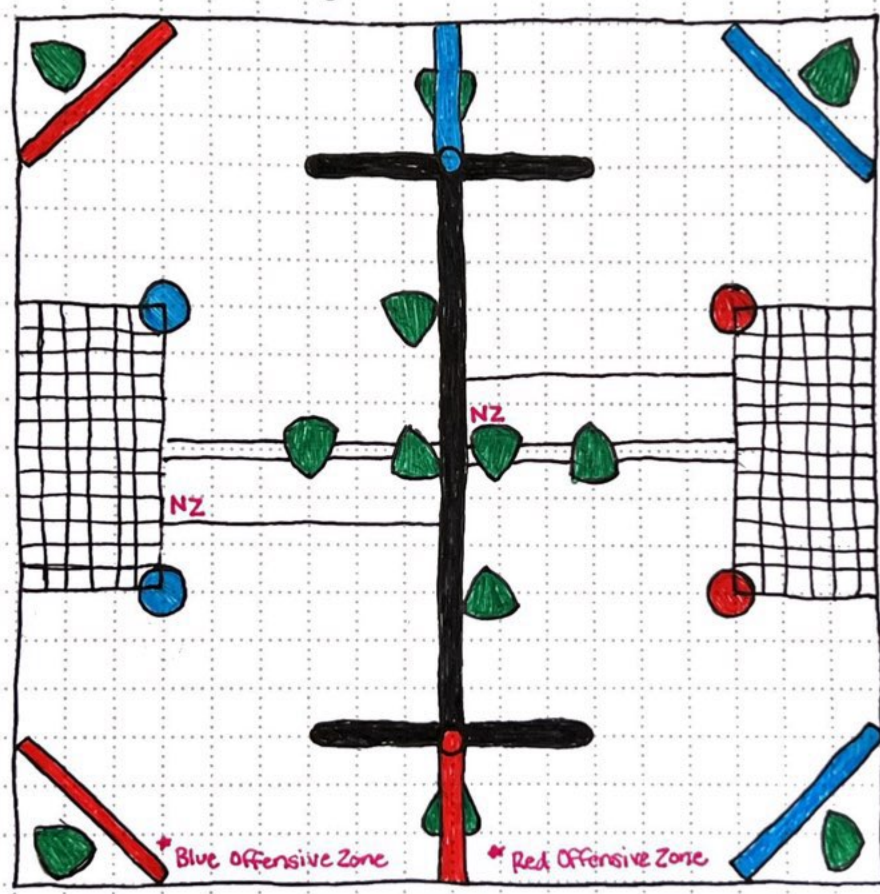
project: Design Process and Award Rubric designed by: Vanessa Perkins witnessed by: Eli [illegible]
date: 5/3/2023

May or June ??

The Challenge-Overview:

Over Under is played on a 12 foot by 12 foot square field with two (2) alliances, red and blue, with two (2) robots per alliance. Each match consists of a 15 second autonomous period and a one (1) minute forty-five (45) second driver period. The overarching object of this game is to score the highest amount of points in a match by scoring triballs in goals or offensives zones and elevating your robot.

Below is a drawing of the field layout.



* Robots that in the first row of tiles parallel to the start barriers excluding the match load zones. Blue on the top with matching elevation bar. Red on bottom.

NZ- Neutral Zone

=- Autonomous Line

* Teams stand on the opposite side of the field as their alliance's offensive zone and goal.

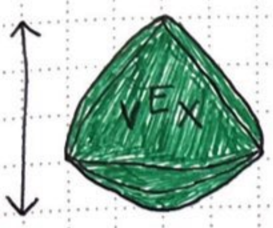
All Field Pieces:

- 60 Triballs
  - 4 Alliance/Preloads, 1 per robot
  - 44 Match Loads, 22 per alliance
  - 12 Begin on field
- 2 Elevation Bars, 1 per alliance
- 2 goals, 1 per alliance
- 4 Match Load Bars/Zones, 2 per alliance

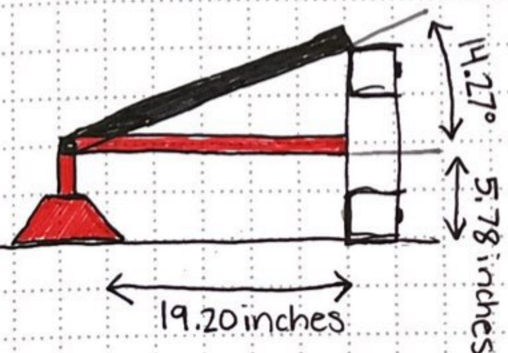
project The Challenge-Overview designed by: Vanessa Perkins witnessed by: Eli Eritts

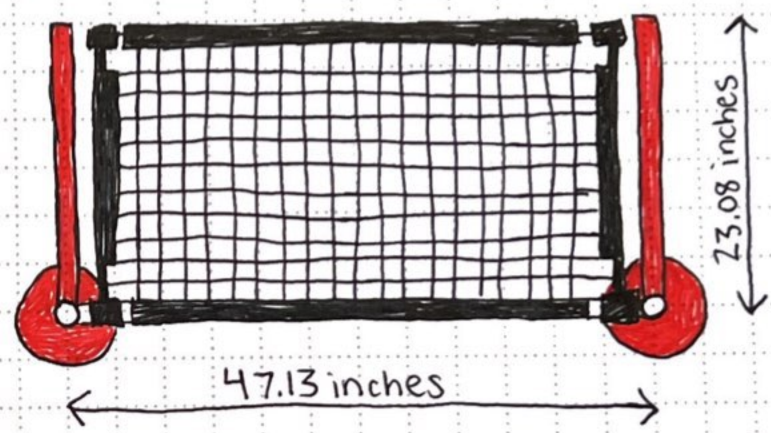
date: 5/3/2023

Field Elements:

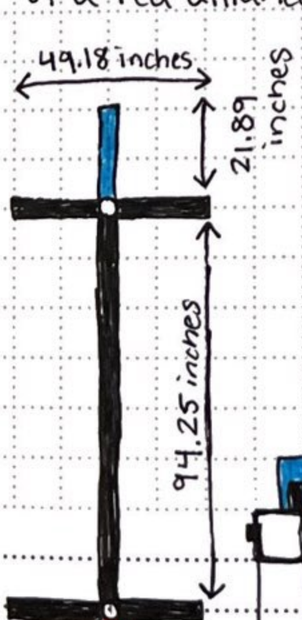


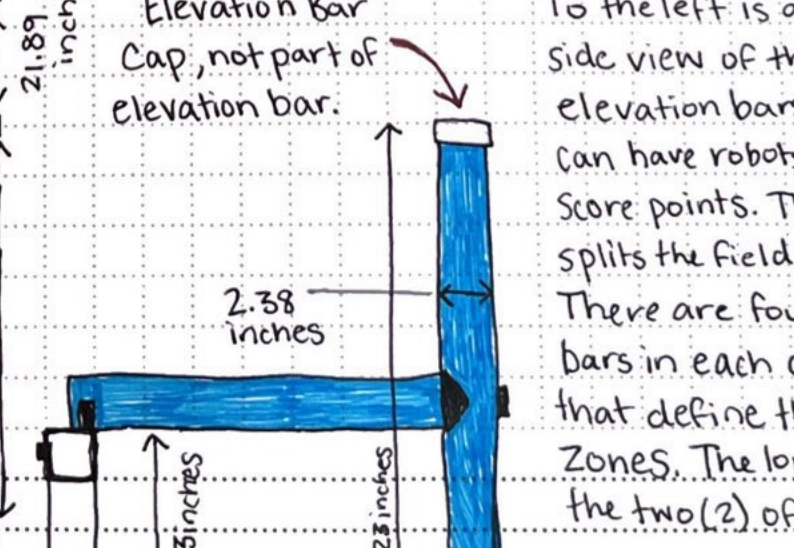
Triballs are red, blue, or green plastic scoring objects shaped like rounded pyramids/Reuleaux triangle. There are sixty (60) total between the field, match loads, and preloads.





Goals are either red or blue to match the alliance, two (2) total, one (1) of each located on either side of the field where triballs can be scored underneath. The goal refers to the net and supporting structures. Above are sketches of a red alliance goal with some dimensions (Side and above view).





To the left is an above view and side view of the barriers and elevation bars. Elevation bars can have robots hang on them to score points. The long barrier splits the field in two halves. There are four (4) match load bars in each corner of the field that define the match load zones. The long barrier defines the two (2) offensive zones.

Rules and Specifics:

Below is a summary of some of the most important rules and specifics this season.

G1 Treat Everyone with Respect.

SG2 Horizontal expansion is limited to thirty-six (36) inches at any point during matches.

SG5 Stay away from nets of the Goals to avoid violation.

SG6 Match load Triballs must enter the match either in the match load zone or placed gently on a robot that is contacting the match load zone or bar.

SG7 Triball possession is limited to one (1).

SG8 Stay out of opponent's Goal unless they are Double-Zoned.

SG11 Elevated Robots are protected. During the last thirty (30) seconds robots may not contact:

- the opposing alliance's elevation bar
- opponent robots contacting their elevation bar
- opponent robots who meet the definition of elevated

* If triballs are intentionally launched out of the field the robot will be Disqualified.
If a triball is unintentionally launched out of the field, the robot will recieve warnings.

* A triball is considered possessed by a robot if it is being held by the robot or in a concave area.
This allows robots to have wings that flap out to push triballs across the field and into goals, multiple at a time. Wings must remain within the horizontal expansion limit.

project Rules and Specifics designed by: Vanessa Perkins witnessed by: Harris Perkins
date: 5/3/2023

Scoring:

Autonomous Bonus	8 points
Each Triball Scored in a Goal	5 points
Each Triball Scored in an Offensive Zone	2 points
Elevation - Top Tier	20 points
Elevation - 2nd Tier	15 points
Elevation - 3rd Tier	10 points
Elevation - 4th Tier	5 points

A Triball is considered "Scored in a Goal" if it:
- is not contacting a Robot of same alliance as the goal color
- atleast 2 corners of the Triball are within the goal

A Triball is considered "Scored in an Offensive Zone" if it:
- is not contacting a Robot of the same color alliance as the zone
- the triball is contacting the gray foam tiles within zone

Alliance Triballs can be scored anywhere and still count for that colored alliance.

A Robot is considered "Elevated" at the end of a match if it:
- is contacting either one or more of their alliance's elevation bars, the barrier, or an alliance partner robot that meets these requirements
- is not contacting any Field Elements other than those in point 1.
- is not contacting the yellow Elevation Bar cap

A Height Guide is used by referees to measure elevation tiers at the end of the match. Points are awarded based on tiers. The highest elevated robot receives top tier and 20 points. The lowest elevated robot receives 4th tier and 5 points.

project: Scoring designed by: Vanessa Perkins witnessed by: Eli Kraitts
date: 5/3/2023

Game Analysis:

Below is a quick summary of our team's analysis of this year's game. We discussed which aspects of the game our robot should focus on and which will be less pertinent to our overall performance.

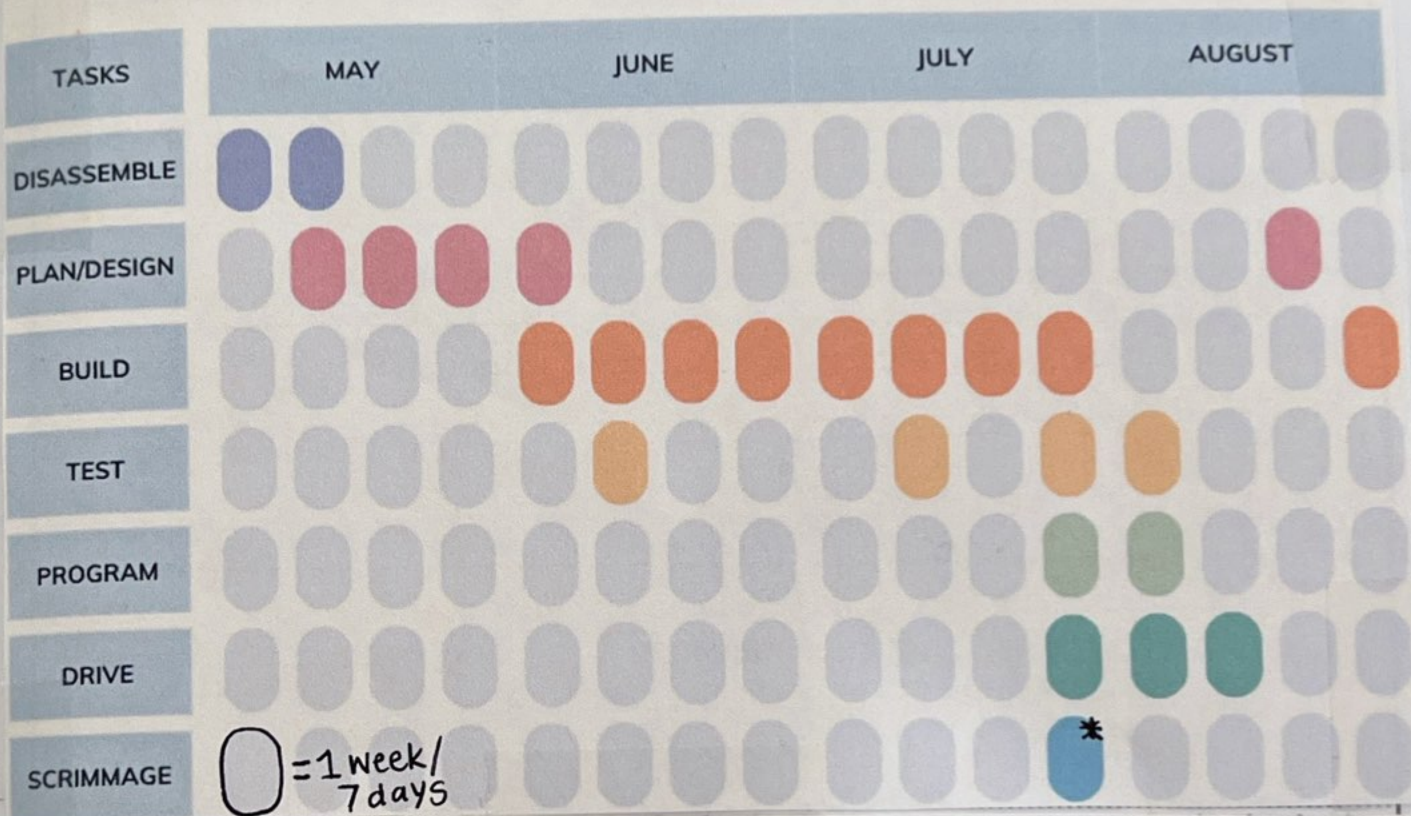
Elevating is good. We think all early season robots will have low level hanging ability so we want our robot to be able to tie or exceed the opposing alliances elevation to maintain field and scoring control.
Utilize Match Load Zones. With only 12 Triballs beginning on the field, our robot needs to be able to get matchloads to the offensive side. Our robot needs to do cross court shots to accomplish this.
Climb barrier. Having a robot that can climb the middle barrier is crucial to be able to save time during matches.
Offensive side of the field is not our primary destination. Our robot will need to throw triballs over to the offensive side from the defensive side before we drive over to score them all.
Don't Double Zone. Having both the robots in the alliance on the defensive side will not be worth the risk. To be able to descore the other team's triballs will not be as important as maintaining our alliance's currently scored triballs.
Scoring about 75% of Triballs will help us more closely gaurntee a win. We think that our robot will have to be extremely fast and efficient to keep up with the opposing team's progress.

OVERALL: Our robot needs to be able to:
- elevate
- utilize match loads
- climb the barrier
- score triballs fast and efficiently

to full optimize this year's game, Over Under.

project Game Analysis designed by: Vanessa Perkins witnessed by: Eli Hawk
date: 5/3/2023

GANTT CHART
EARLY SEASON TIMELINE



Disassemble- Take apart last season's robot to reuse parts and save some budget

Plan/Design- Read and analyze the rules, research ideas, select a robot design

Build- Begin assembling the pieces

Test- Intermittently test what we have built and change as necessary

Program- Set up controls and autonomies

Drive- Practice learning controls and strategies

Scrimmage- Final test for the robot when it is complete

*After our scrimmage scheduled for July 29th, we plan to re-evalute the robot, redesign, and rebuild as necessary.

project Early Season Timeline designed by: Vanessa Perkins witnessed by: Eli [illegible]

date: 5/4/2023

Goals:

Below is a list of our team's overall goals for the season, for the first tournament, and for the robot to help keep us on target throughout our progress.

Long Term Goals:

- ☐ Qualify for WORLDS ●
  - End in Top 5 of Division
- ☐ Win States Tournament Champion, Excellence, and/or Skills ●
- ☐ Have 5 Autonomouses ●
  - 1 full win point
  - 1 skills
  - 3 normal
- ☐ CAD a majority of the robot ● ●

First Competition Goals:

- ☐ Qualify for STATES ●
- ☐ Win Tournament Champion ● ●
- ☐ Focus on Skills after first competition ●
- ☐ Improve Scouting skills ●

VP

Robot Goals:

- ☐ Can Score from Match Load Zone ●
- ☐ Can climb barrier ●
- ☐ Can hang ●

Primary Responsibilities:

- ● - Whole Team
- ● - Vanessa (Journalist)
- ● - Harris (~~[illegible]~~ Programmer)
- ● - Owen (Driver)
- ● - Eli (Builder)

The dots next to each of our goals correspond to the matching team member who is ultimately responsible for completing the goal. Several Dots displays teamwork.

VP

project: Goals designed by: Vanessa Perkins witnessed by: Owen taylor date: 5/4/2023

Team Budget:

Replacement Parts	
Single Pneumatic	150
Double Pneumatic	250
11W Motor (2)	90
Screws and Nuts (200)	50
Steel Bar (8)	15
Travel	
States	1500
Registrations	
Season	150
Sullivan Tournaments (2)	150
Signature Average	300
States	300
Total Cost	2955

Income/Funding	
Start up	1500
Fundraiser 1	1000
Fundraiser 2	500
Total Funding	3000

Above is a spreadsheet of our team's budget.
On the left side is all the costs we will need covered this season to the closest estimate possible.
On the right side is the money that goes into our budget. The School received two grants as specified below that was divided between our four teams to give us a starting budget of $1500.00.

Johnson County Community Foundation Grant $4,000
Wachs Leadership Fund $1,000
Previous Balance $1,000

The majority of our parts on the robot this season will be reused from previous years to save resources and manage our budget more efficiently.
Our goal is to raise $1500.00 between two fundraisers to help pay for expenses.

project: Team Budget   designed by: Vanessa Perkins   witnessed by: Eli Fruits
date: 5/4/2023

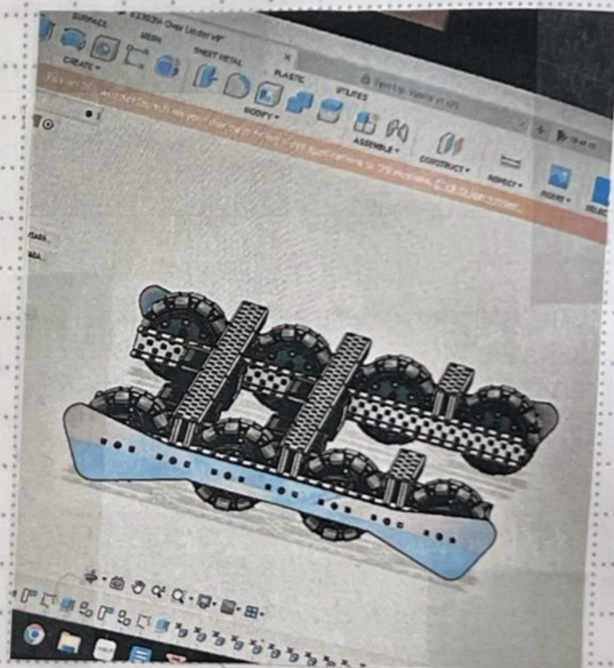
Date: 5/6

Brainstorming Base Designs:

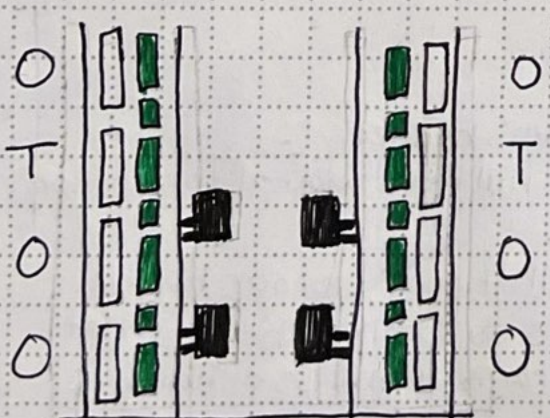
Goals for today → Research and come up with atleast 3 base design options to select between
→ Respect and Listen to every team member's ideas

Design Option One: Basic Drive

Specifics:
- 4 motors
- 6 omni wheels (3.25 inch)
- 2 traction wheels (3.25 inch)
- 360 rpm
- 36:60 gear ratio
- 30 long hole base
- Sleds



CAD of this design option by Eli Fritts



The gearing alternates between 60 tooth gears and 36 tooth gears.

* This base will need Lexan Sleds to help it climb the barrier.

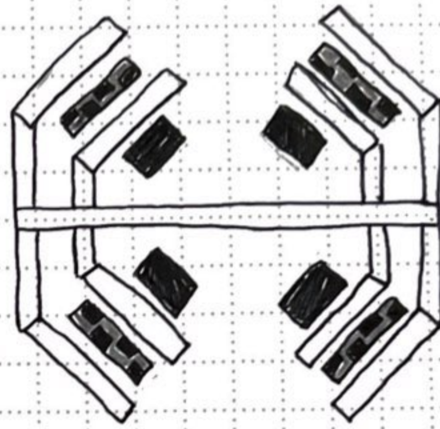
The O's and T's on the sides of the base sketch correspond to whether they are omni or traction wheels.

project: Brainstorming Base Designs
designed by: Vanessa Perkins
witnessed by: Owen taylor
date: 5/6/2023

(Continued)

Design Option Two: X-Drive

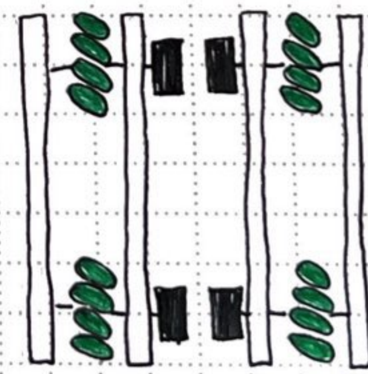
Specifics:
- 4 motors
- 4 omni wheels (3.25 inch)
- Green Cartridge
- 200 rpm
- Good manueverability



* Not designed for gearing room.

Design Option Three: Mecanum

Specifics:
- 4 motors
- 4 mecanum wheels
- Green Cartridge
- 200 rpm
- Sleds
- Good manueverability



* Too wide of wheels to have sufficient room to add gearing.

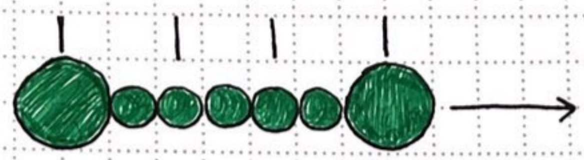
project: Brainstorming Base Designs (continued) designed by: Vanessa Perkins witnessed by: Owen taylor date: 5/6/2023

(Continued)

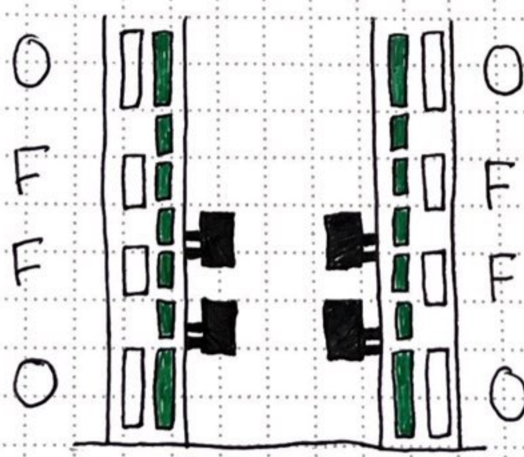
Design Option Four: 4 Inch Omni Drivetrain

Specifics:

- 4 motors
- 4 omni wheels (4inch)
- 4 two inch flex wheels
  or 4 two point seventyfive (2.75) inch traction wheels
- 600 rpm geared down
- 36:72 gear ratio



This image shows the gears from the side. The larger are 72 teeth gears and the smaller are 36 tooth. The ticks above represent where the wheel axels will be attached to gears.



The purpose of this base is to have two small levitating wheels in the center that have the function of helping the robot climb the barrier more efficiently.

The O's and F's represent which wheels are Omni or Flex / 2.75 inch traction.

Date: 5/11

Select Approach:

Goals for today → Choose a base design that best meets the requirements of the game so we can begin building.

We created a decision matrix of what aspects mattered the most on our base. Below is that decision matrix with the winning base.

Base Ideas	Structure	Programability	Transition Over Barrier	Speed	Traction	Torque	Total
Basic Drive	5	5	3	4	3	2	22
X-Drive	2	2	0	3	1	1	9
Mecanum	4	3	2	2	2	4	17
4 Inch Omni	2	3	5	4	1	3	13

* Structure: how sturdy the base is.
Programability: how easy and consistent the base is to program.
 - factors in the sink of the wheels into the field
Transition Over Barrier: how smoothly and efficiently the base scales the barrier.
Speed: how fast the base is.
Traction: the grip of the wheels.
Torque: the strength, ability to play defense, and withstand defense

Selected Design: Basic Drive

Explanation: The basic drive design is a happy medium of every aspect. The basic drive provides enough stability and structure to build all our other subsystems on top of. This base design is able to scale the barrier, gain speed, and decently keep up with offense and defense between the combined affects of traction and torque. Lastly, the high number of wheels help eliminate sink and improve programability.

* Breakdown of each criteria aspect we looked at. A rank of 0 (zero) means bad, and a rank of 5 (five) is good.

project: Select Approach designed by: Vanessa Perkins witnessed by: Eli Fritts
date: 5/11/2023

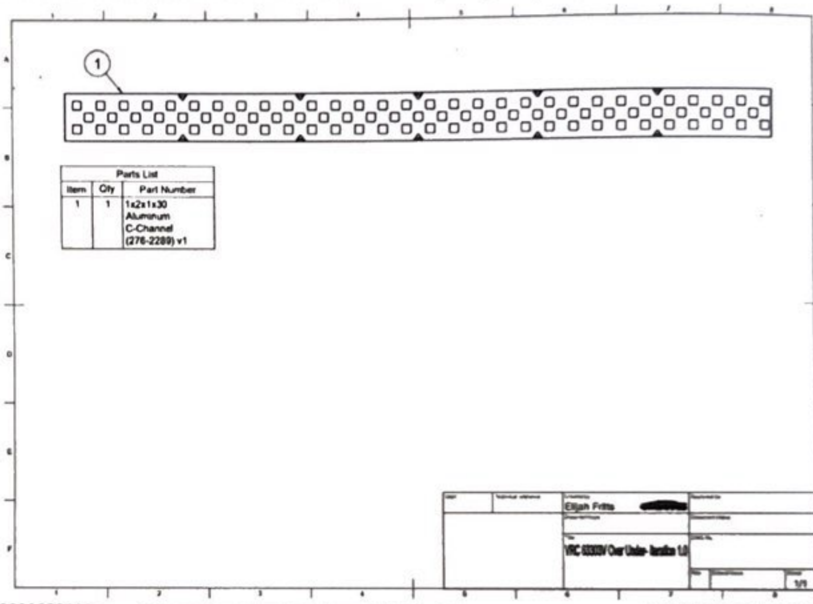
Iteration 2 | Programming | Building | Problem

Begin Building Base:

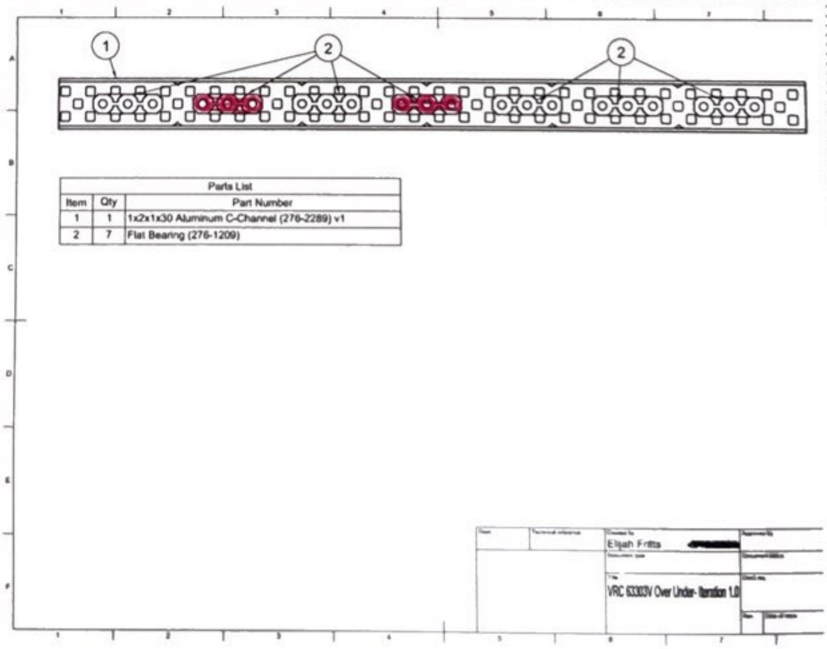
Date: 5/12

Goals for today → Gather parts for base
→ begin assembling
→ CAD along the way
→ get all the bearing flats attached to the c-channels

(1) Gather a 30 long aluminum c-channel. Repeat 3 times for a total of 4 c-channels.



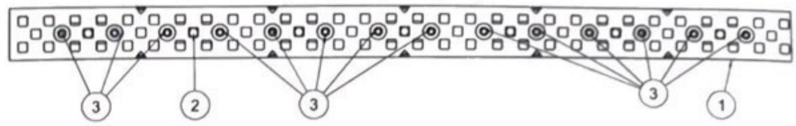
(2) Gather 7 bearing flats and space them out 1 hole in between each on the inside of the c-channel. Repeat a 2nd time. Repeat a 3rd and 4th time except exclude the 2nd* and 4th* bearing flats for a total of 5.



project: Begin Building Base designed by: Vanessa Perkins + Eli Fritts witnessed by: Harris Perkins date: 5/12/2023

(Continued)

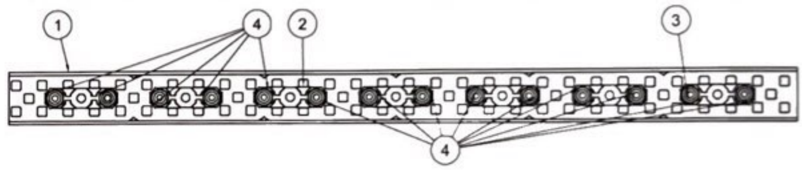
3. Place 0.5 inch (half inch) screws through each outside hole of each bearing flat for a total of 48 screws between all 4 c-channels.



Parts List		
Item	Qty	Part Number
1	1	1x2x1x30 Aluminum C-Channel (276-2289) v1
2	[illegible]	Flat Bearing (276-1209)
3	14	#8-32 x 1_2 Star Drive Screw (276-4992)

Elijah Fritts [redacted]
VRC 63303V Over Under- Iteration 1.0

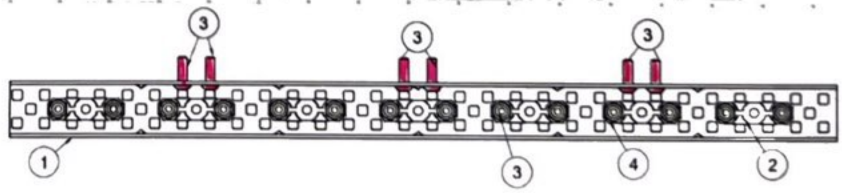
4. Place a nylock nut on each of the ~~step~~ screws from Step 3 for a total of 48 nylock nuts between all 4 c-channels. Tighten.



Parts List		
Item	Qty	Part Number
1	1	1x2x1x30 Aluminum C-Channel (276-2289) v1
2	7	Flat Bearing (276-1209)
3	14	#8-32 x 1_2 Star Drive Screw (276-4992)
4	14	#8-32 Nylock Nut (275-1027)

Elijah Fritts [redacted]
VRC 63303V Over Under- Iteration 1.0

5. Place 6 half inch screws on each 4 c-channels in the labeled positions* ⟶ for a total of 24 new screws added.



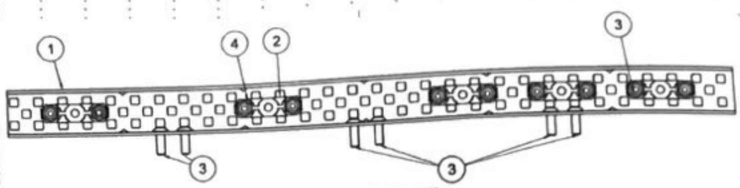
Parts List		
Item	Qty	Part Number
1	1	1x2x1x30 Aluminum C-Channel (276-2289) v1
2	7	Flat Bearing (276-1209)
3	20	#8-32 x 1_2 Star Drive Screw (276-4992)
4	14	#8-32 Nylock Nut (275-1027)

Elijah Fritts [redacted]
VRC 63303V Over Under- Iteration 1.0

project: Begin Building Base (continued) designed by: Vanessa Perkins & Eli Fritts witnessed by: Owen taylor date: 5/12/2023

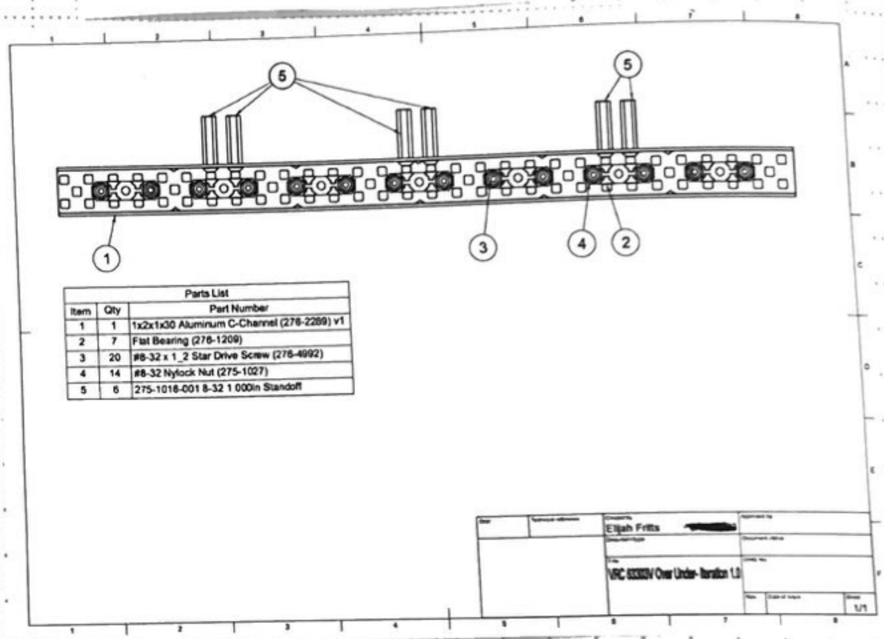
(Continued)

(5) Continued... To the right is the CAD for the 2 c-channels with only 5 bearing flats.

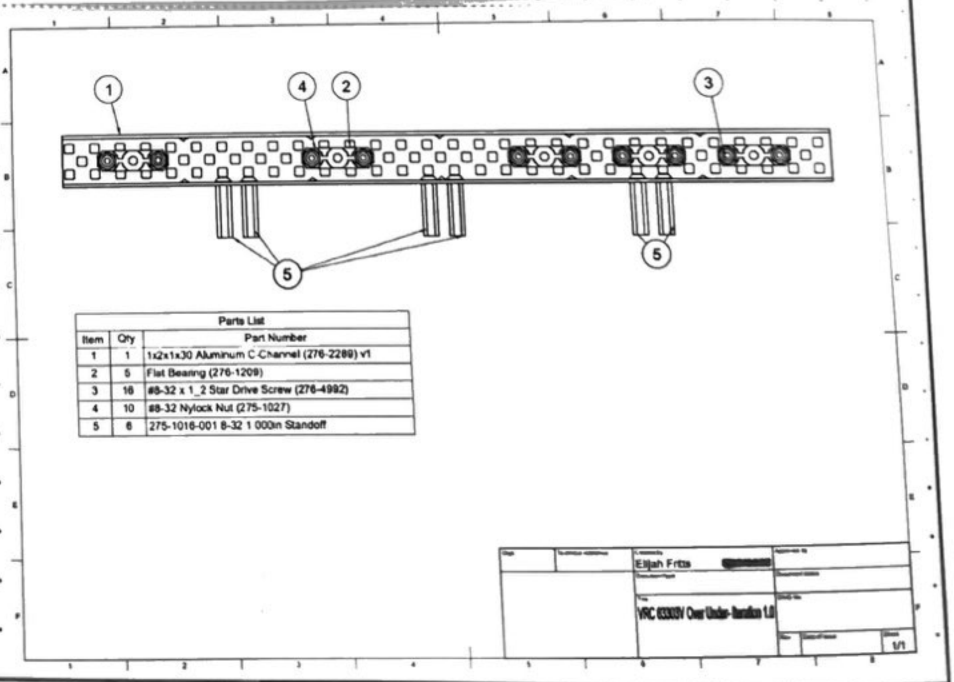


Parts List		
Item	Qty	Part Number
1	1	1x2x1x30 Aluminum C-Channel (276-2289) v1 (1) (1) (1)
2	5	Flat Bearing (276-1209)
3	16	#8-32 x 1_2 Star Drive Screw (276-4992)
4	10	#8-32 Nylock Nut (275-1027)

(6) Add 1.00 inch (one) standoffs to each of the six (6) Screws on the top of the c-channel. Repeat for the 2nd, 3rd, and 4th c-channels. On the bottom to the right is the image for the 2 c-channels with only 5 bearing flats. A total of twenty-four (24) standoffs used.



Parts List		
Item	Qty	Part Number
1	1	1x2x1x30 Aluminum C-Channel (276-2289) v1
2	7	Flat Bearing (276-1209)
3	20	#8-32 x 1_2 Star Drive Screw (276-4992)
4	14	#8-32 Nylock Nut (275-1027)
5	6	275-1016-001 8-32 1 000in Standoff



Parts List		
Item	Qty	Part Number
1	1	1x2x1x30 Aluminum C-Channel (276-2289) v1
2	5	Flat Bearing (276-1209)
3	16	#8-32 x 1_2 Star Drive Screw (276-4992)
4	10	#8-32 Nylock Nut (275-1027)
5	6	275-1016-001 8-32 1 000in Standoff

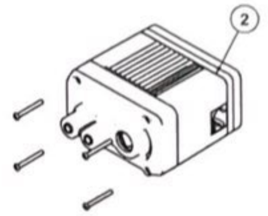
Adding Base Motors:

Date: 5/15

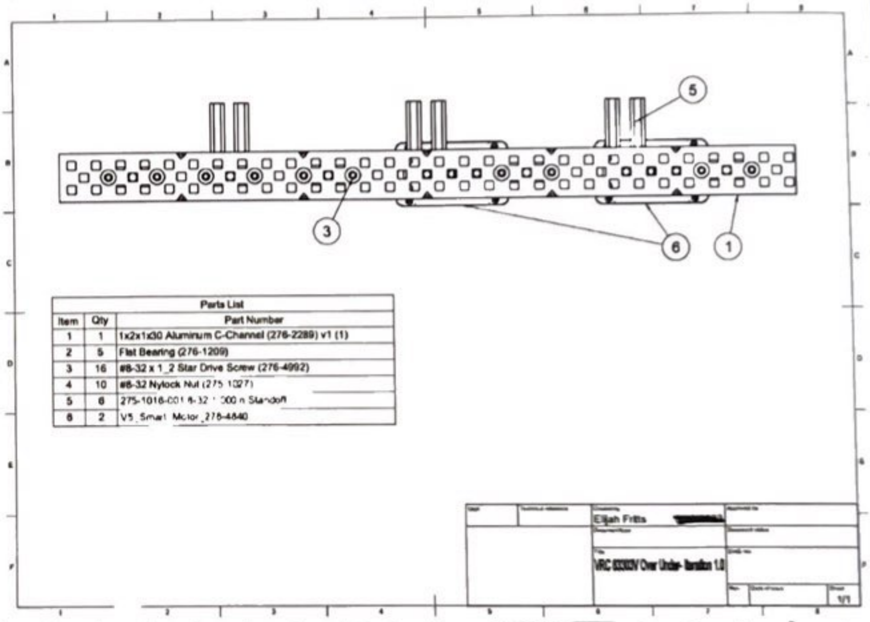
Goals for today → get motors attached to c-channels

⑦ Swap motor cartridges to blue high speed cartridges. Close using zipties for quick change. Repeat for a total of 4 motors.

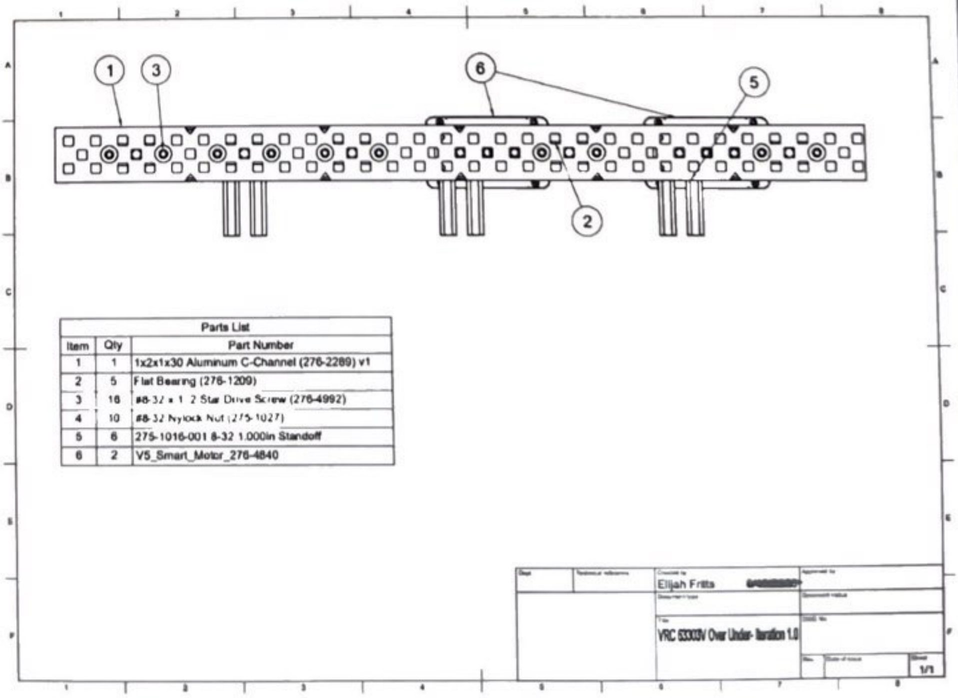
⑧ Place all 4 motors in the correct placement on the two c-channels with only 5 bearing flats. The CAD images to the right show this step on each c-channel. Where the missing 4 bearing flats would be is where each of the 4 motors go.



Parts List		
Item	Qty	Part Number
2	1	V5_Smart_Motor_ 276-4840



Parts List		
Item	Qty	Part Number
1	1	1x2x1x30 Aluminum C-Channel (276-2289) v1 (1)
2	5	Flat Bearing (276-1209)
3	16	#8-32 x 1_2 Star Drive Screw (276-4992)
4	10	#8-32 Nylock Nut (275 1027)
5	6	275-1016-001 8-32 1.000in Standoff
6	2	V5_Smart_Motor_276-4840



Parts List		
Item	Qty	Part Number
1	1	1x2x1x30 Aluminum C-Channel (276-2289) v1
2	5	Flat Bearing (276-1209)
3	16	#8-32 x 1 2 Star Drive Screw (276-4992)
4	10	#8-32 Nylock Nut (275-1027)
5	6	275-1016-001 8-32 1.000in Standoff
6	2	V5_Smart_Motor_276-4840

Iteration 2

Building

Problem

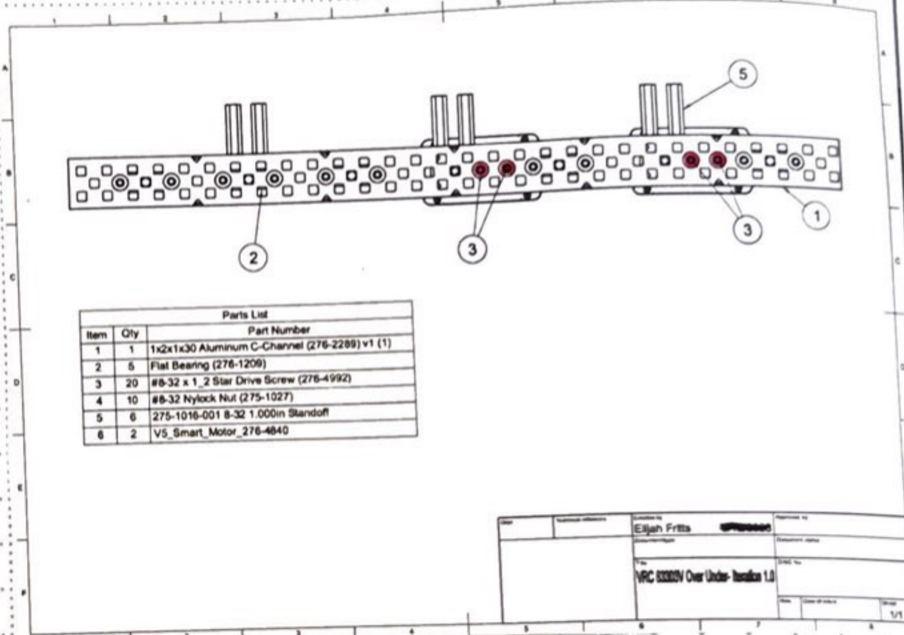
project: Adding Base Motors   designed by: Vanessa Perkins + Eli Fritts   witnessed by: Owen taylor   date: 5/15/2023

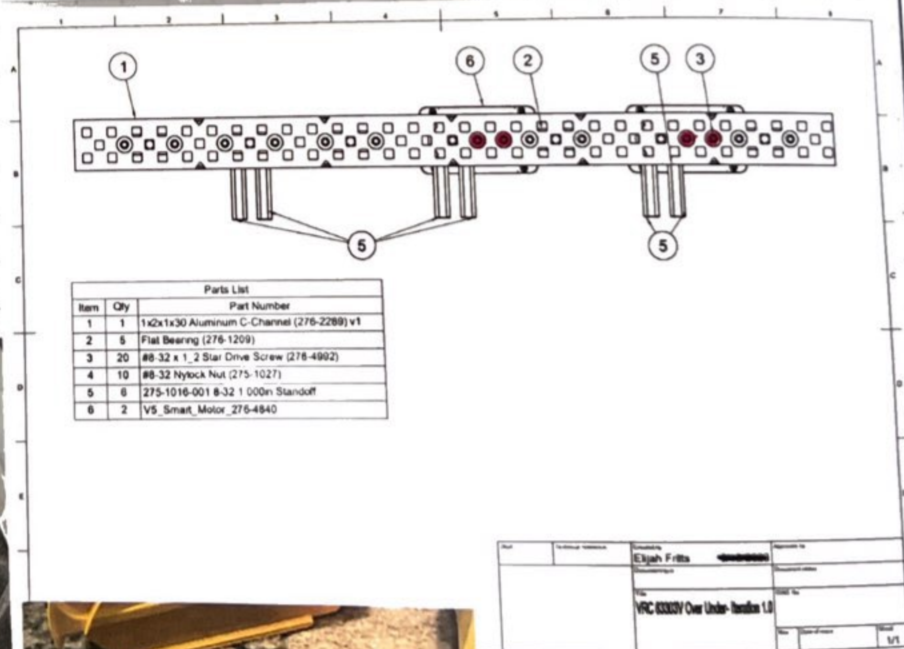
Date: 5/15

(Continued)

9 Add eight (8) half inch (0.500 inch) screws* to the motors. Tighten. To the right is the CAD images with the screws added to each of the motors. Below are images of our current robot outside of the CAD images we've been showing.

VP





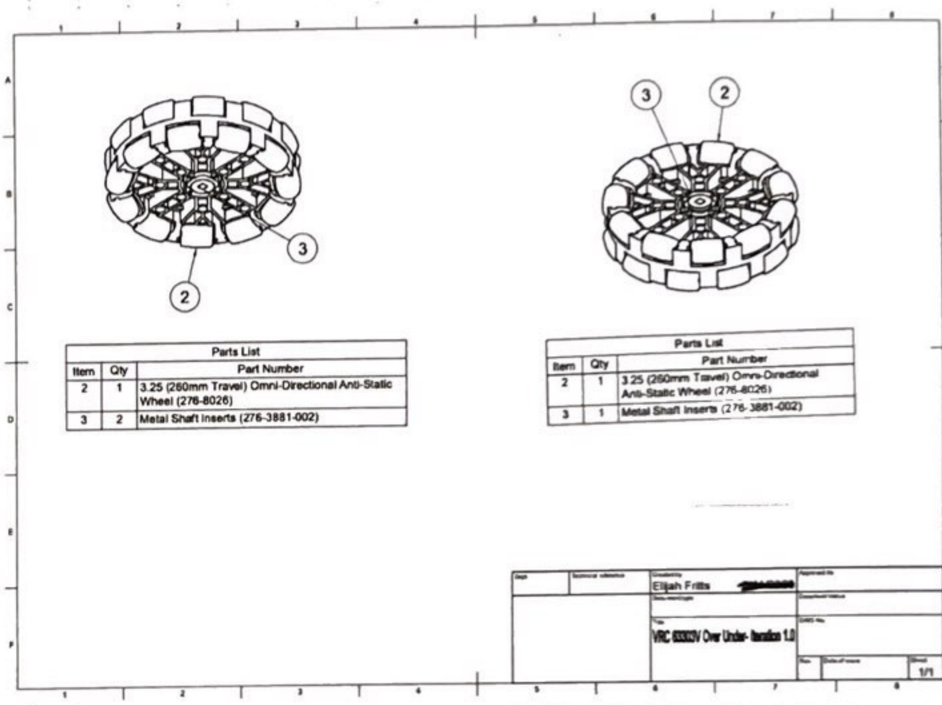
project Adding Base Motors designed by: Vanessa Perkins & Eli Fritts witnessed by: Owen Taylor date: 5/15/2023

Date: 5/16

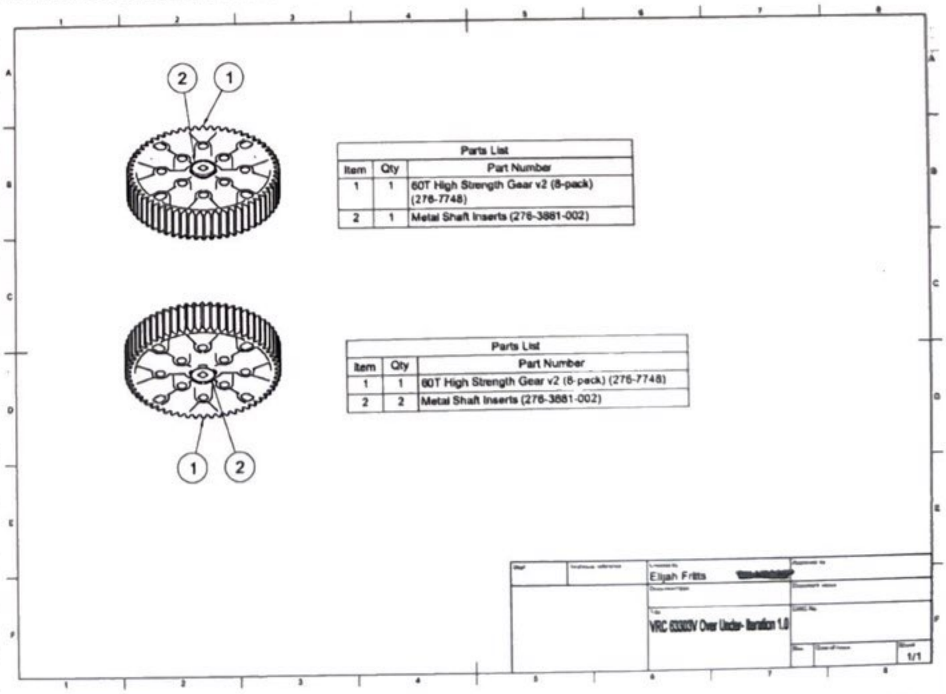
Attaching Gears to Wheels:

Goals for today → get the gears screwed on to all the wheels
→ begin spacing the wheels on axels

(10) We added metal shaft inserts to both sides of each wheel. We did this for six(6) omni wheels 3.25 inch diameter and two(2) traction wheels 3.25 inch diameter (not depicted in CAD) for a total of 8 wheels.



(11) Gather eight (8) 60 tooth high strength gears and add metal shaft inserts to each side of each gear for a total of sixteen (16) inserts being used. VP



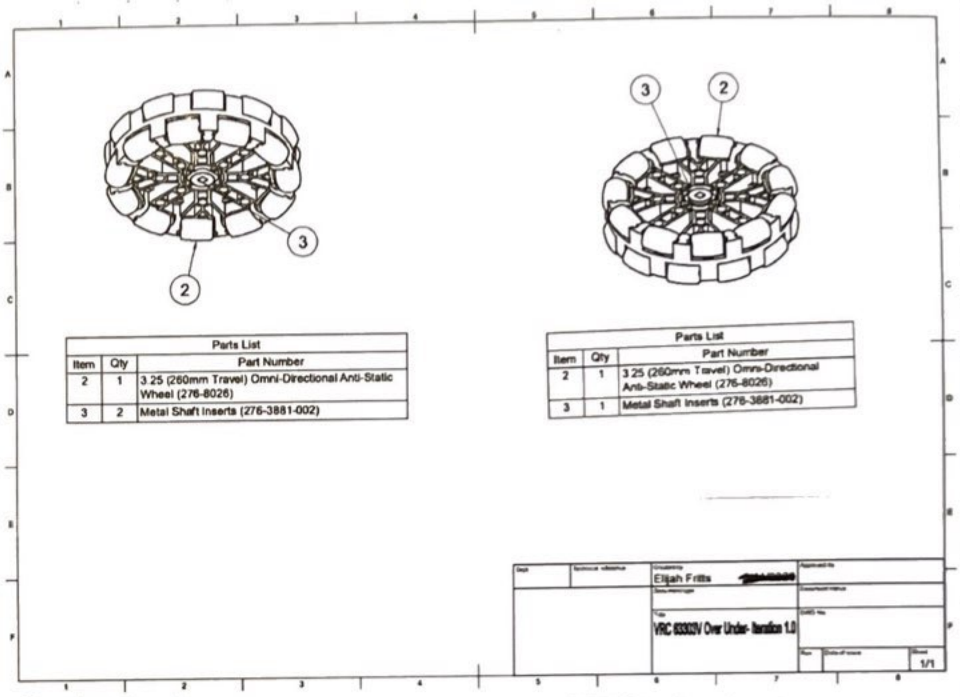
project: Attaching Gears to Wheels designed by: Vanessa Perkins + Eli Fritts witnessed by: Harris Perkins date: 5/16/2023

Date: 5/16

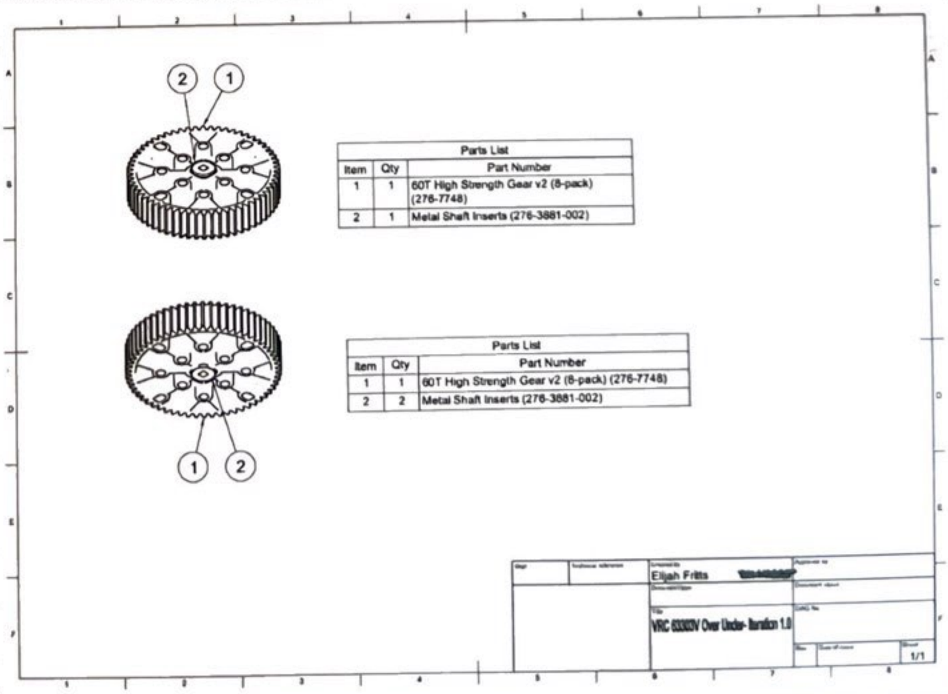
Attaching Gears to Wheels:

Goals for today → get the gears screwed on to all the wheels
→ begin spacing the wheels on axels

(10) We added metal shaft inserts to both sides of each wheel. We did this for six(6) omni wheels 3.25 inch diameter and two(2) traction wheels 3.25 inch diameter (not depicted in CAD) for a total of 8 wheels.



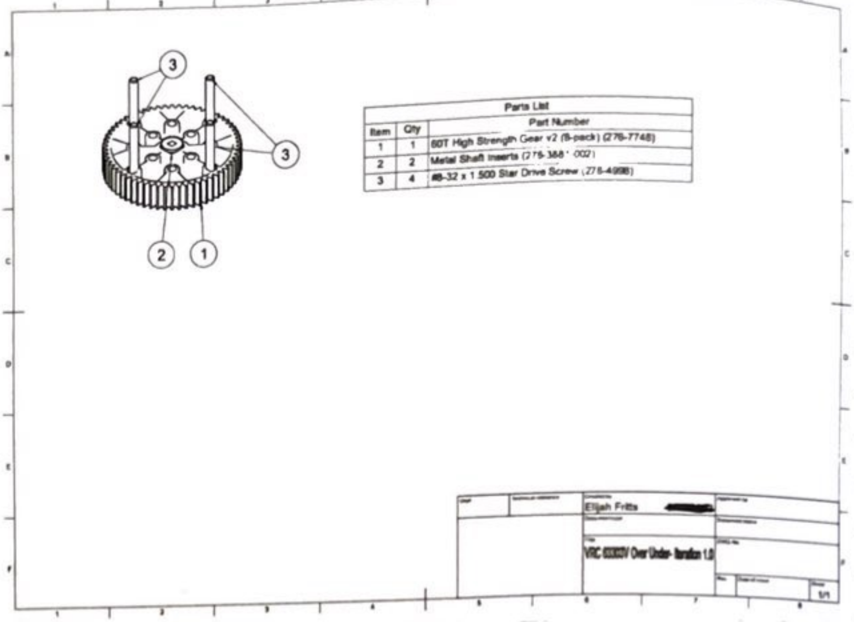
(11) Gather eight (8) 60 tooth high strength gears and add metal shaft inserts to each side of each gear for a total of sixteen (16) inserts being used. VP



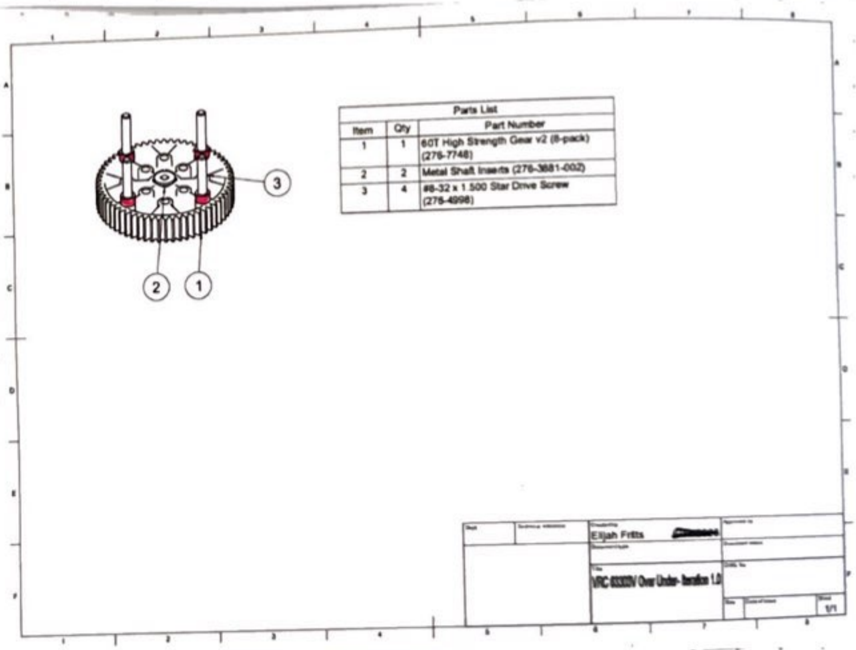
project: Attaching Gears to Wheels designed by: Vanessa Perkins + Eli Fritts witnessed by: Harris Perkins
date: 5/16/2023

(Continued)

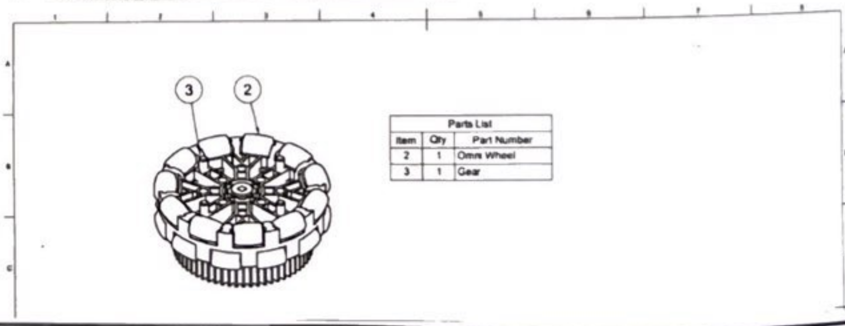
(12) Insert four (4) screws to each of the 8 gears in the depicted places to the right in the CAD images. The screws are 1.500 inch long and there are a total of thirty-two (32).



(13) Add 0.25 inch spacers* on each of the 1.500 inch screws for a total of thirty-two (32) spacers.



(14) Place the 6 omni and 2 traction wheels on top the 8 gears aligning the screws into the holes of the wheels.

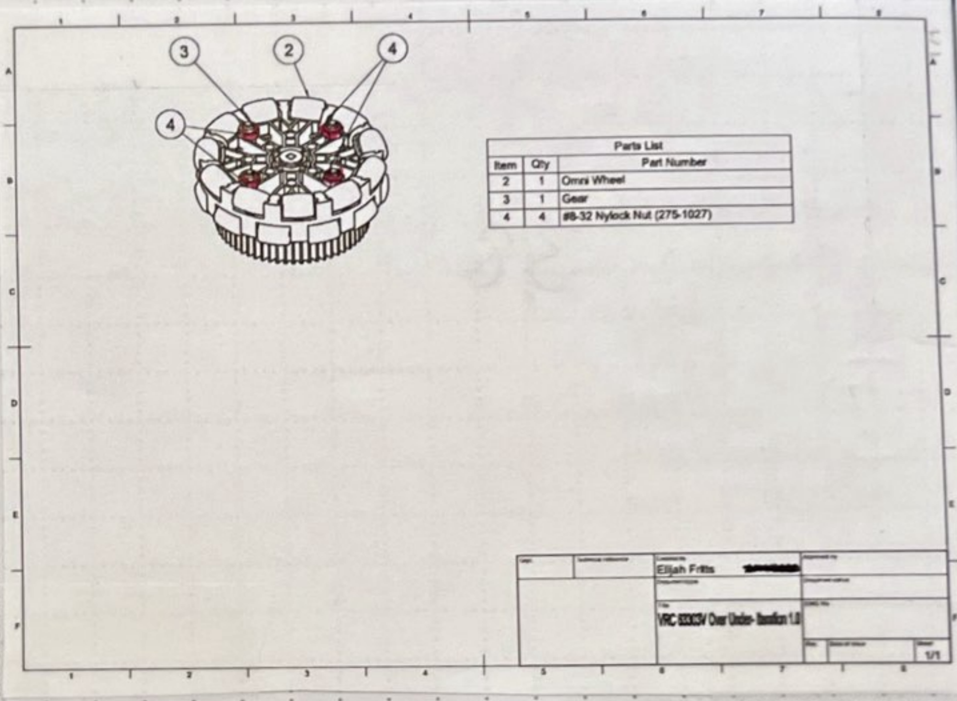


project Attaching Gears to Wheels (continued) designed by: Vanessa Perkins & Eli Fritts witnessed by: Harris Perkins date: 5/16/2023

Date: 5/16

(Continued)

(15) Add a nylock nut* to each of the 1.500 inch screws to attach the wheels and gears firm together. Repeat for all 8 wheels for a total of thirty-two (32) nylock nuts used.



To the left are pictures of this section of the build. The pictures only show one wheel, but we have eight in total, two traction and six omni.

<u>Purpose of Design:</u>

The purpose of these past six steps of building the base are crucial to the overall performance. By screwing the gears to the wheels, it helps with the following things:

- Keeping them both aligned during rotation
- Eliminating wiggling
- Reducing inconsistencies in programming

Without these steps, we'd run into future problems with the performance and consistency of our base's driving.

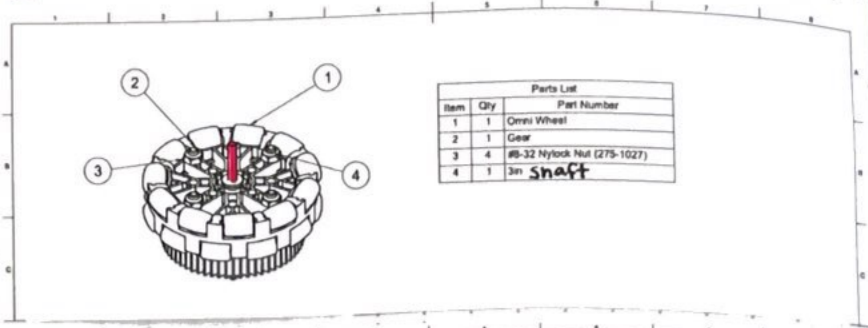
VP

project Attaching Gears to Wheels designed by: Vanessa Perkins + Eli Fritts witnessed by: Harris Perkins date: 5/16/2023

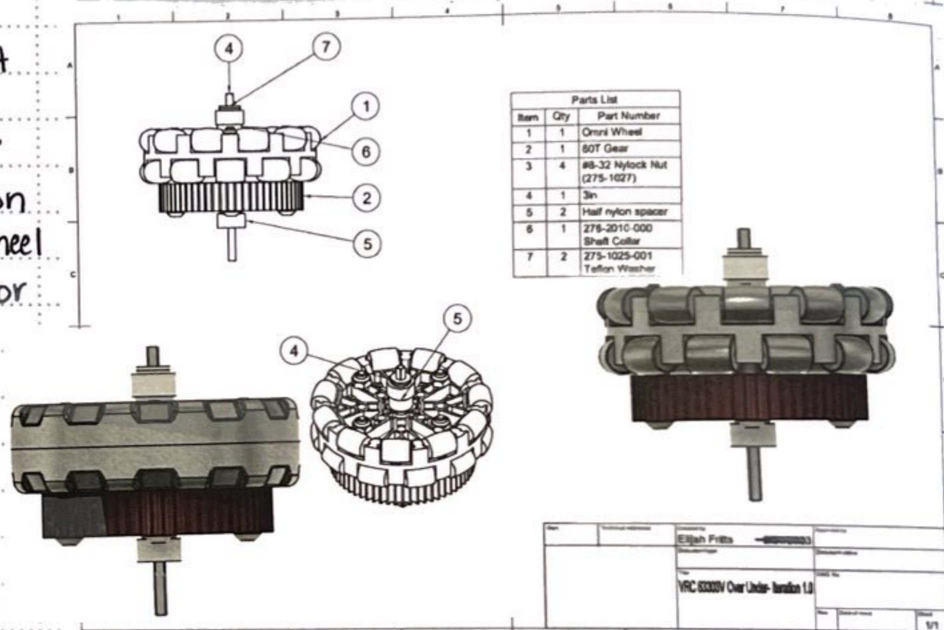
Wheel Spacing:

Date: 5/16

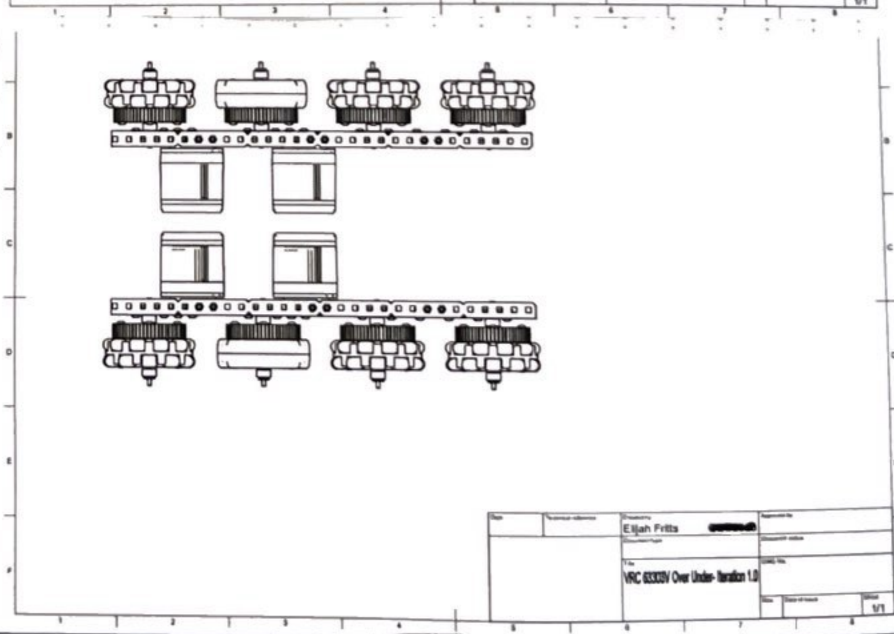
(16) Insert three (3) inch Shafts into each of the eight (8) wheels.



(17) Add a shaft collar, then 0.25 inch spacer and two (2) Teflon Washers to the wheel side of the axel for all 8 wheels. Add 0.25 inch spacer to the gear side of the axel for all 8 wheels. Total of sixteen (16) 0.25 inch spacers, sixteen (16) Teflon washers, eight (8) shaft collars.



(18) Insert the eight (8) wheel shafts with the correct spacing into the depicted locations of the two c-channels with motors to the right.



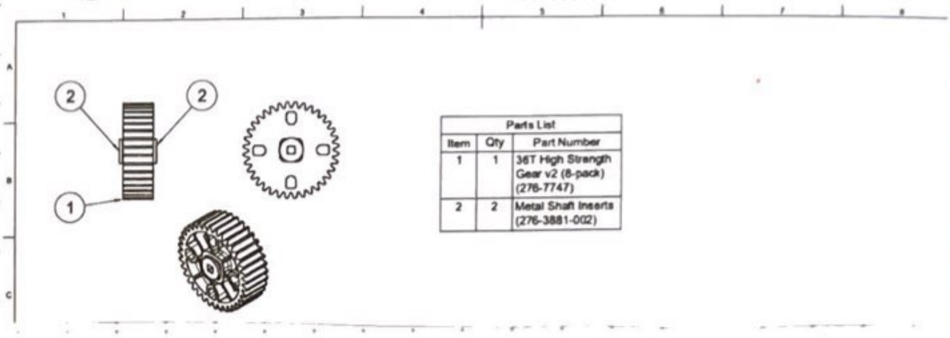
project: Wheel Spacing
designed by: Vanessa Perkins & Eli Fritts
witnessed by: Owen taylor
date: 5/16/2023

Date: 5/17

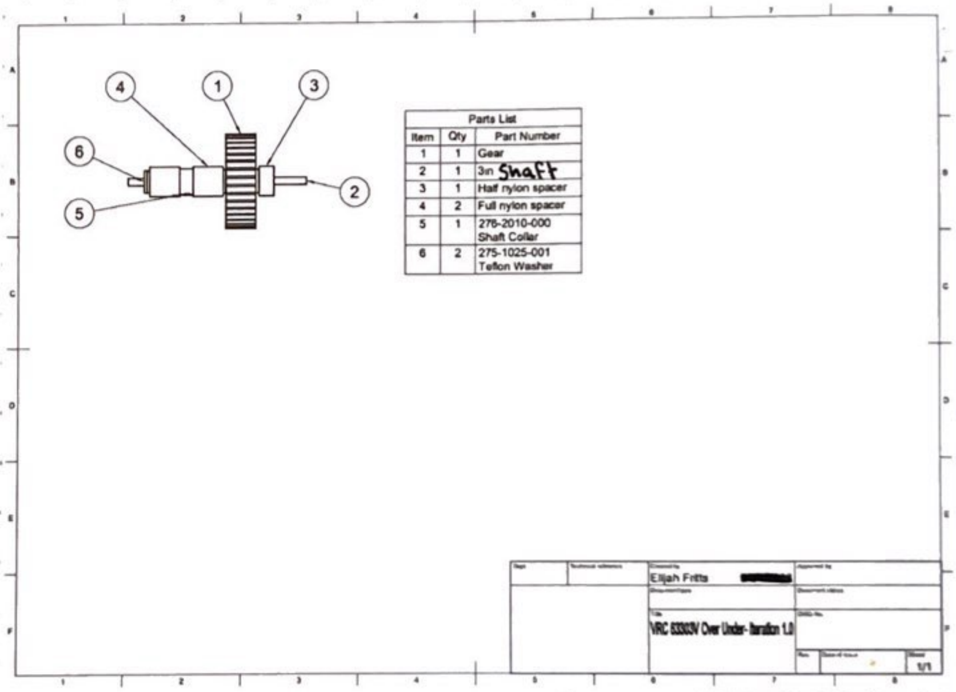
Finish Base Gearing:

Goals for today → get all shafts in place on the base

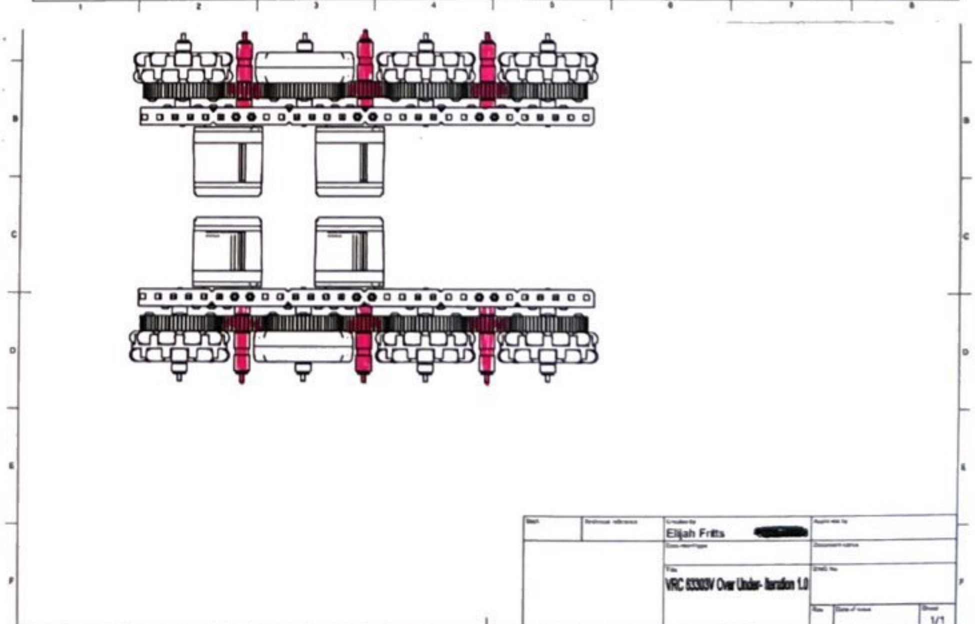
(19) Gather six(6) 36 tooth high strength gears.



(20) Add 0.500 inch spacer, then shaft collar, then another 0.500 inch spacer, and two (2) Teflon Washers to the left side of the three (3) inch shaft. Place 36t gear in middle, then add one (1) 0.250 inch spacer to the right side of the 36t gear. Repeat for all six (6) gears.



(21) Insert the six (6) gear shafts* with correct spacing into the depicted locations of the two (2) C-channels with motors to the right inbetween each of the wheels.



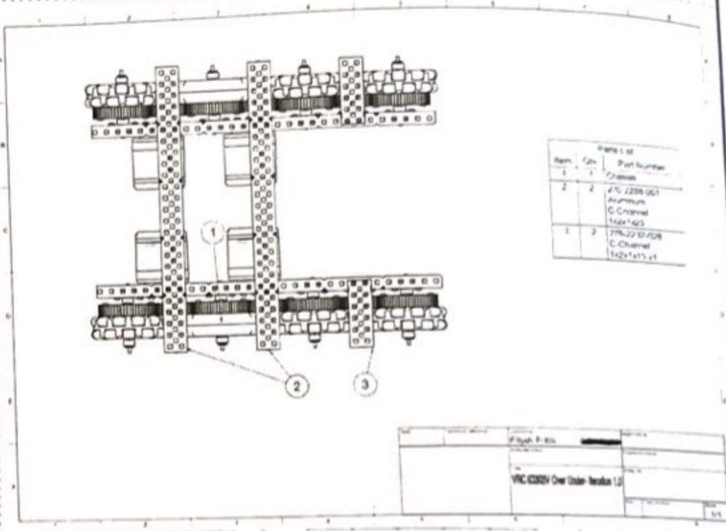
project Finish Base Gearing designed by: Vanessa Perkins + Eli Fritts witnessed by: Owen taylor date: 5/17/2026

Date: 5/18

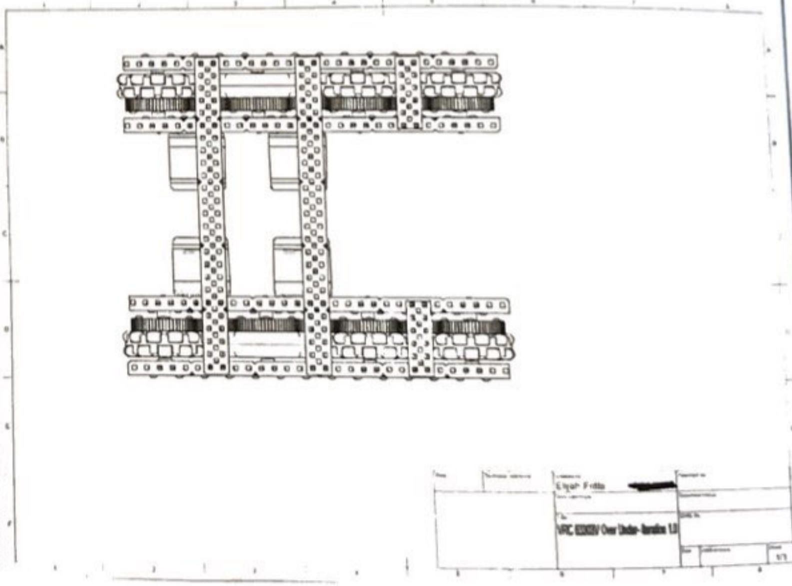
Connect Base C-channels:

Goals for today → Get base finished by screwing all the C-channels together
→ Document finished base pictures and CAD

22 Cut/Gather two (2) aluminum C-channels 25 holes long.
Also cut/gather two (2) 6 long aluminum C-channels. Place above the wheels and gears in locations to the right, right above the six (6) standoffs on each side.



23 Place the two (2) remaining aluminum 30 long C-channels from step 6 in their depicted places on the outside of the drive base chassis. The six (6) standoffs on each C-channel should line up with the perpendicularly placed C-channels from step 22.

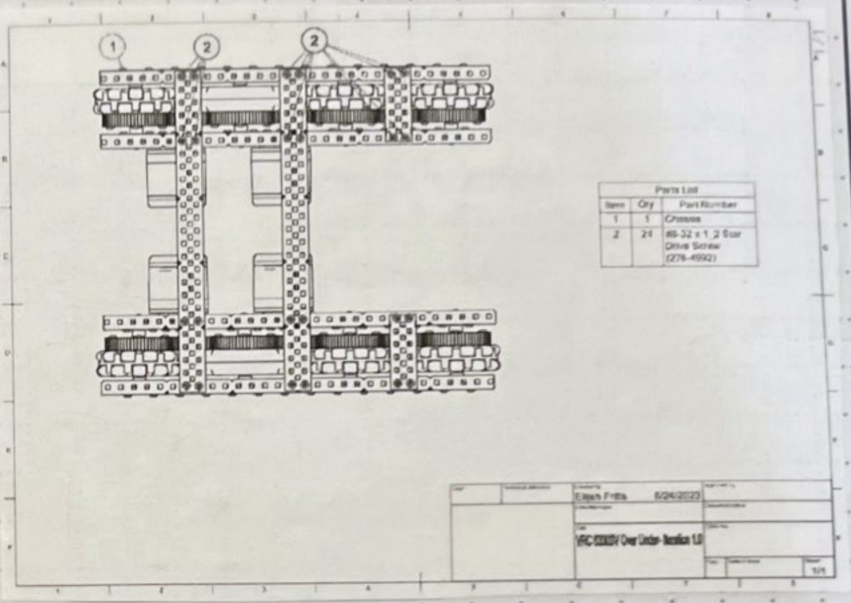


VP

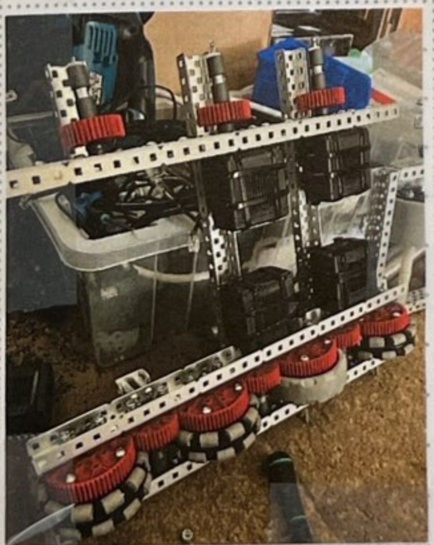
project: Connect Base C-channels designed by: Vanessa Perkins + Eli Fritts witnessed by: Harris Perkins
date: 5/18/2023

Date: 5/18

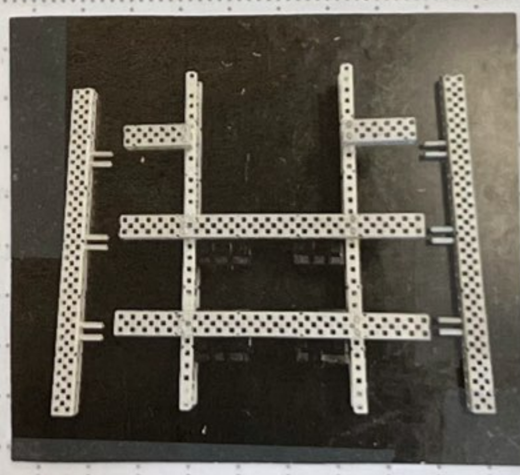
(Continued)

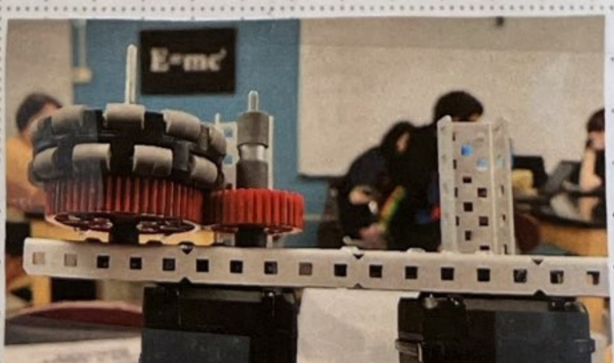


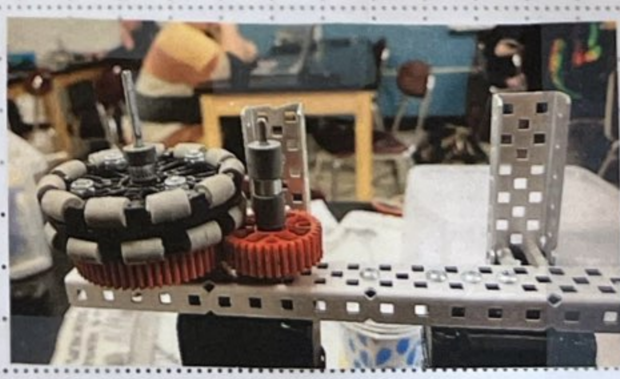
(24) Add twenty-four (24) half (0.5) inch screws to the depicted locations to tighten the loose c-channels on top to the standoffs below. These pieces will keep the base square, structured, held together, and provide space to build future subsystems off of.



Below are extra ~~pieces~~ pictures from this build. To the left is an image of all the C-channels used to construct the base. The rest of the images are of wheels, gears, and spacing.







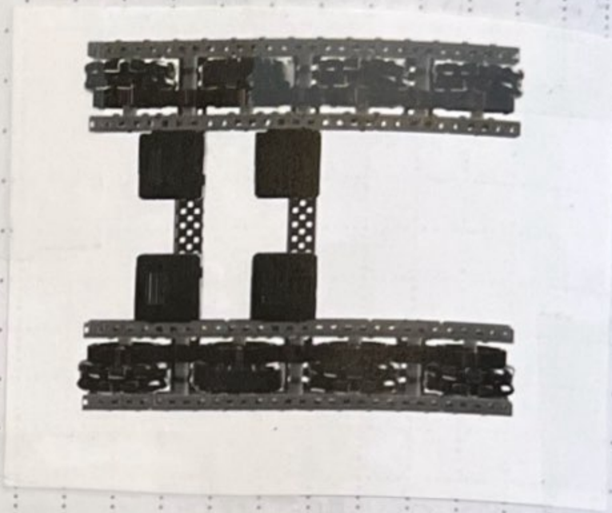
project: Connect Base C-channels (continued)
designed by: Vanessa Perkins + Eli Fritts
witnessed by: Owen taylor
date: 5/18/2023

Date: 5/18

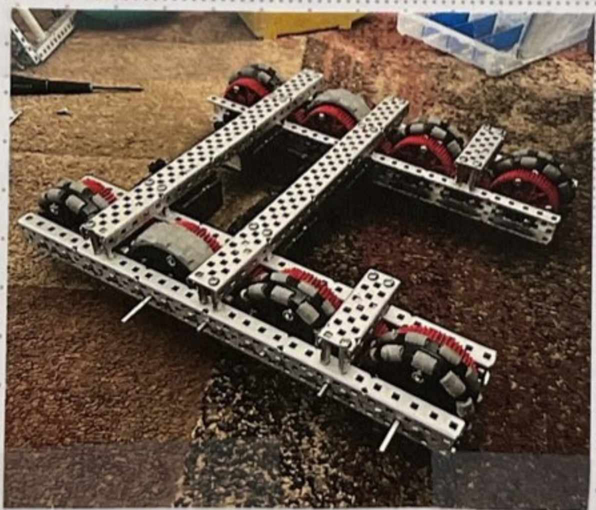
Finished Drivebase Images:

Below are CAD images from several different angles of our complete drivetrain/base complete as well as an actual image of our base fully constructed.

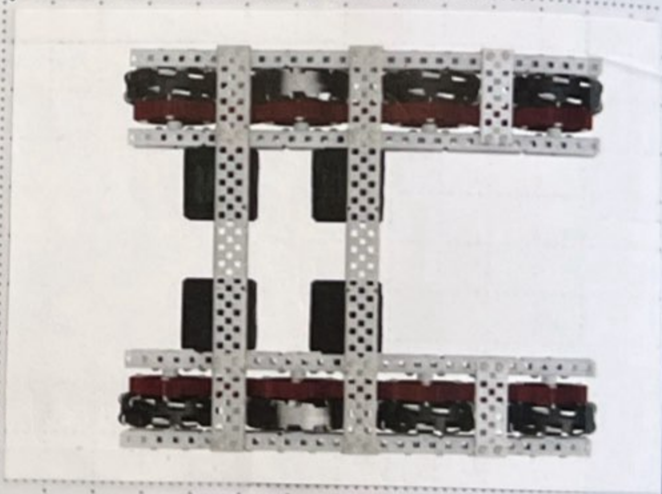
Bottom View

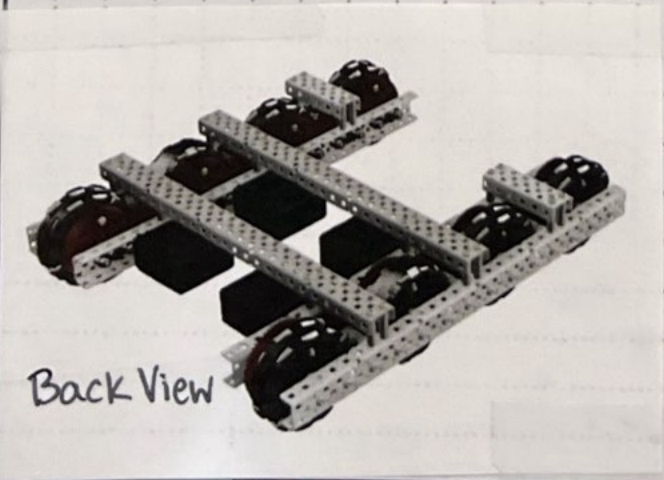


Actual Drivetrain

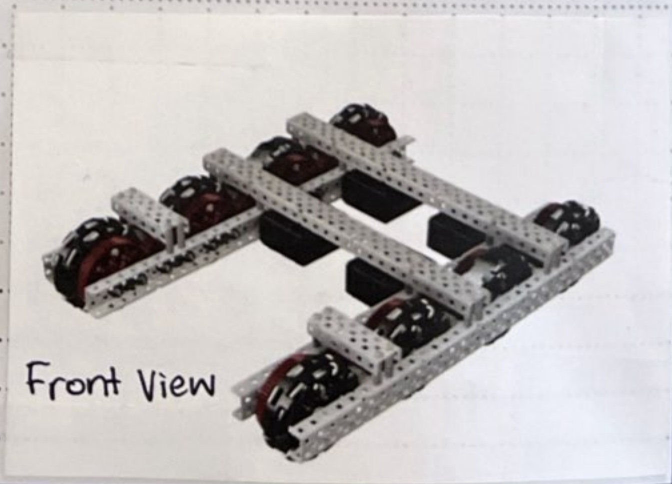


Top View





Back View



Front View

Complete Base Parts List:

Goals for today → work as a team to compile full list of every part for base to create a place for easier access to resources used

Metal:

- 4-30 long aluminum c-channels
- 2-25 long aluminum c-channels
- 2-6 long aluminum c-channels
- 14-3inch metal shafts

Wheels/Gears:

- 6 - 3.25 inch diameter Omni Wheels
- 2-3.25 inch diameter Traction Wheels
- 32 metal shaft inserts
- 8 - 60 tooth high strength gears
- 6 - 36 tooth high strength gears

Spacers:

- 54-0.250 inch spacers
- 28 Teflon Washers
- 12-0.500 inch spacers
- 14 shaft collars

Miscellaneous:

- 24 bearing flats
- 4 motors
- 4-600 rpm low torque, high speed, blue cartridges

Screws/Nuts/Standoffs:

- 104-0.500 inch screws
- 80 Nylock nuts
- 24 - 1.00 inch standoffs
- 32-1.500 inch screws

project Complete Base Parts List designed by: Vanessa Perkins witnessed by: Eli [illegible]
date: 5/18/2023

Brainstorming Scoring Mechanism Designs:

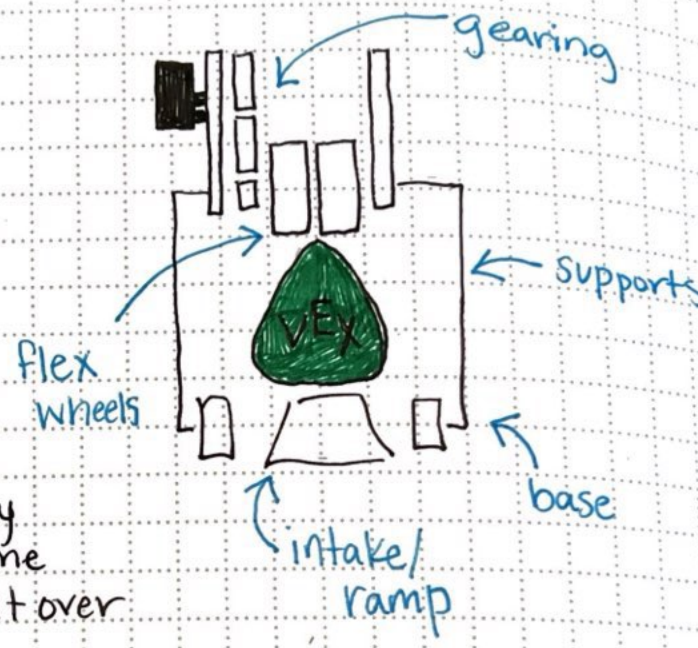
Date: 5/19

Goals for today → Research and Come up with atleast 3 Scoring mechanism design options to select between
→ Respect and listen to every team member's ideas

Design Option One: Vertical Flywheel

Specifics:

- 1 to 2 motors
- 60:12 gearing
- flex wheels



To the right is a rough sketch of a Vertical flywheel. The launch will be very low but might have some extra up spin to make it over the barrier better.

A downside to this design is that it is probably inconsistent since there is only a small surface area of the triball being controlled by the flywheel.

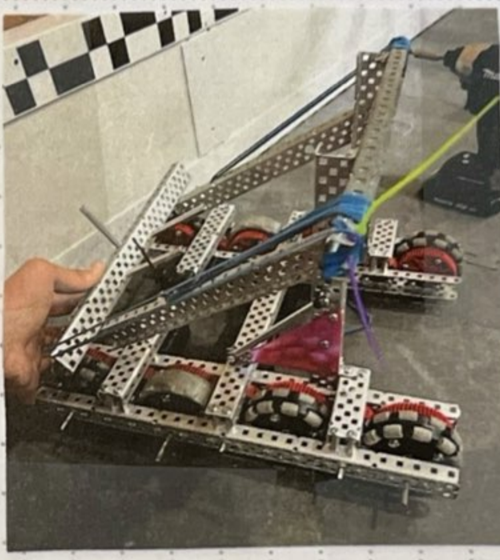
VP

project: Brainstorming Scoring Mechanism Designs designed by: Vanessa Perkins witnessed by: Eli Strik date: 5/19/2023

(Continued)

Design Option Two: Catapult

Specifics:
- 1 to 2 motors
- lots of rubberbands
- gearing with slip gear



To the right is a prototype catapult we built without the gearing. The purpose of the prototype was to test the strength and launch of the catapult.

The catapult has lots of strength and can get triballs over the barrier with ease.
The down side it that it would require much toning to get the correct angle of launch, rubberband tension, and slip gear size.
The catapult prototype is also very unstable every time it releases so it would require bracing and maybe anti-tips.

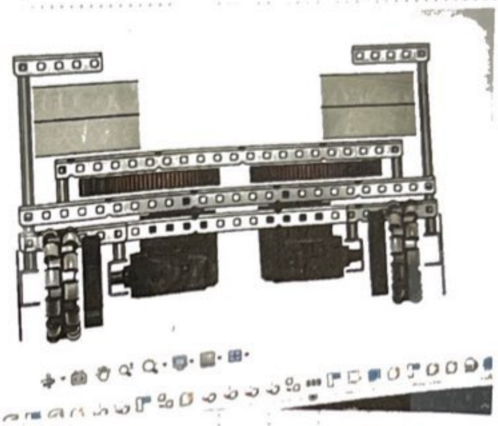
VP

project Brainstorming Scoring Mechanism Designs (continued) designed by: Vanessa Perkins witnessed by: Eli Stuke
date: 5/19/2023

(Continued)

Design Option Three: Horizontal Flywheel (Basic)

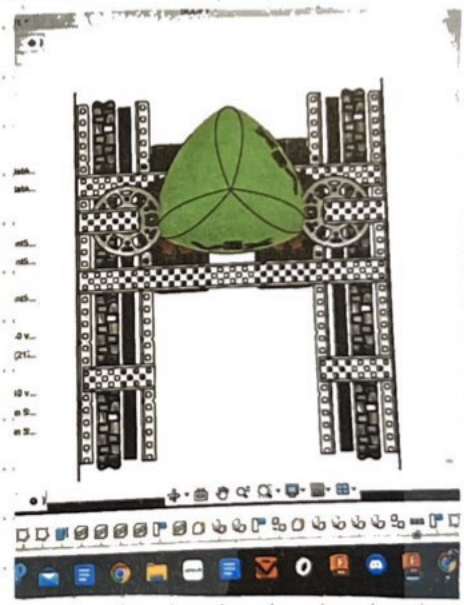
Specifics:
- 2 motors
- 60:12 gearing
- flex wheels
- 2 parts

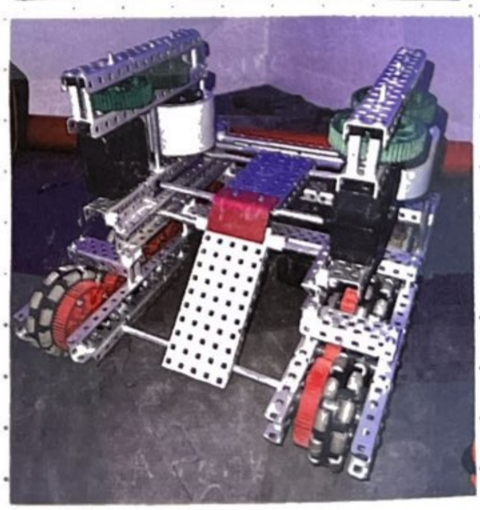


To the right are some CAD images of a horizontal flywheel. We also built a prototype, depicted on the lower right hand side.

The prototype helped us see that the horizontal works well to have two(2) sides spinning.

The horizontal flywheel would need an attach to adjust the angle to compensate for robots trying the block the low launch of the triballs.



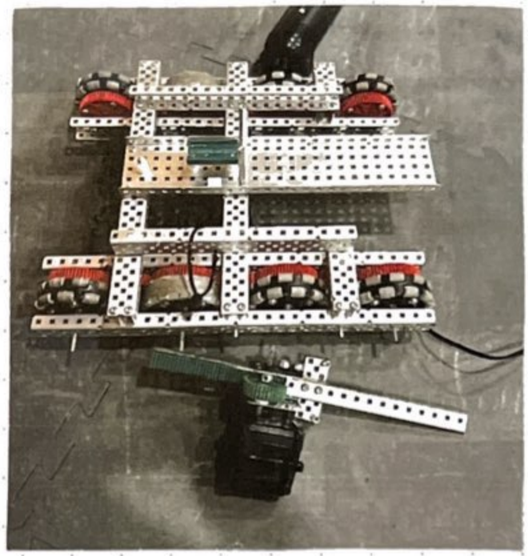


project: Brainstorming Scoring Mechanism Designs (Continued) designed by: Vanessa Perkins witnessed by: Owen taylor date: 5/19/2023

(Continued)

Design Option Four: Puncher

Specifics:
- 1 to 2 motors
- slip gear
- slidding/linear gearing
- rubberbands



To the right is a simple prototype of a puncher design that we built. We found that the puncher has a lot of power, but is inconsistent if the slip gear and linear, slidding gears are not aligned properly.

The puncher would launch triballs low to the ground every time unless we add something to change the angle.

VP

project Brainstorming Scoring Mechanism Designs (Continued) designed by: Vanessa Perkins witnessed by: Harris Perkins
date: 5/19/2023

Date: 5/19

Select Approach:

Goals for today → Choose a scoring mechanism design that best meets the requirements of the game.

We created a decision matrix of what aspects mattered the most on our scoring mechanism. Below is the matrix with the winning design.

Scoring Ideas	Speed	Consistency	Power	Durability	Size	Total
Vertical Flywheel	5	2	2	2	3	14
Catapult	3	4	5	2	3	17
Horizontal Flywheel	5	5	5	3	2	20
Puncher	2	2	4	2	5	15

* Speed: how fast it can score triballs
Consistency: how accurate the launch of triballs is
Power: how strong the triballs are launched so they can make it over the barrier
Durability: how it holds up against other robots and use
Size: the compactness and good use of space

Selected Design: Horizontal Flywheel

Explanation: The basic horizontal flywheel has the best speed, consistency, and power out of all the options. The flywheel will not be easy to break or require much toning and redesigning. The most important thing is the consistency because we need to be able to score triballs frequently. Not using rubberbands on the flywheel will make it more consistent than the puncher and catapult.

* Break down of each criteria aspect we looked at. A rank of zero (0) means bad and a rank of five (5) means good.

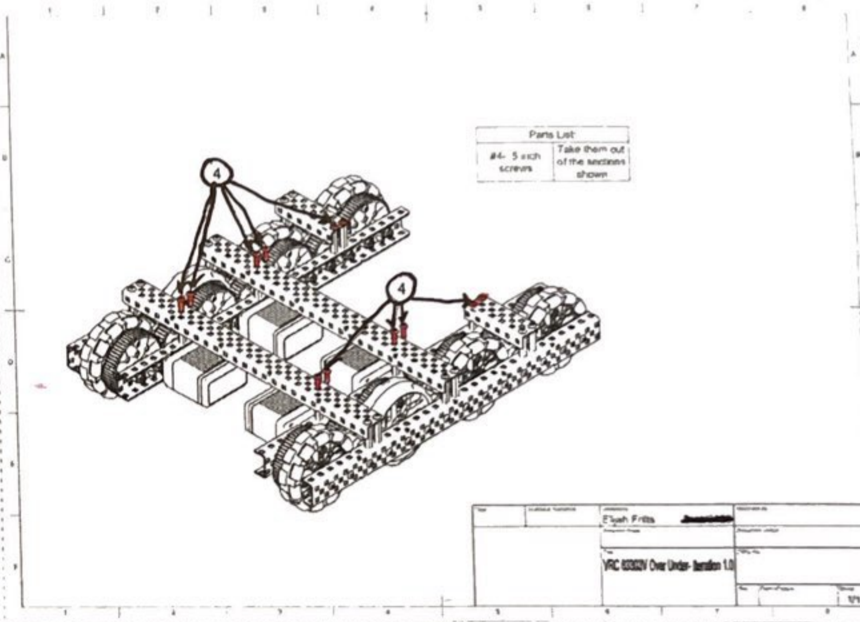
project Select Approach designed by: Vanessa Perkins witnessed by: Eli Fritts
date: 5/19/2023

Date: 5/20

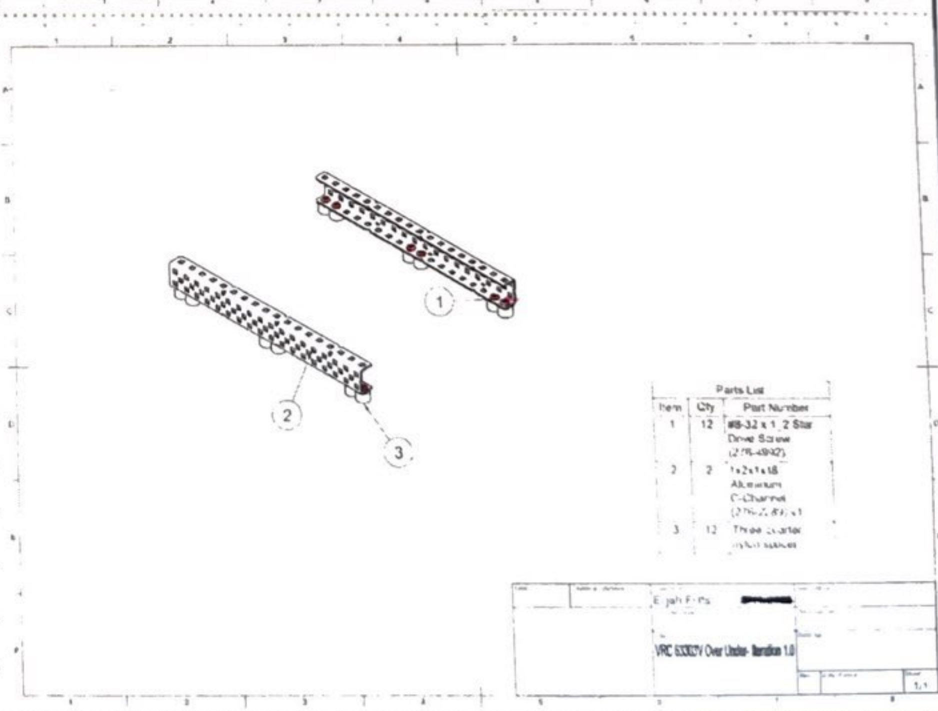
Flywheel Supports:

Goals for today → Build a structure off of our base for the flywheel subsystem to attach to
→ make the supports simple and light to conserve space and keep weight low

(1) Remove the eight (8) screws* depicted to the right from the C-channels on top of the base. Remove the last 4 screws from the front on the six (6) long C-channels. Remove a total of twelve (12) screws.



(2) Gather/cut two (2) eighteen (18) hole long aluminum C-channels. Add a 0.750 inch screw to each of the 12 holes* to the left. Add a 0.375 inch spacer to the bottom of each of those screws. Total of twelve (12) screws and twelve (12) spacers used.



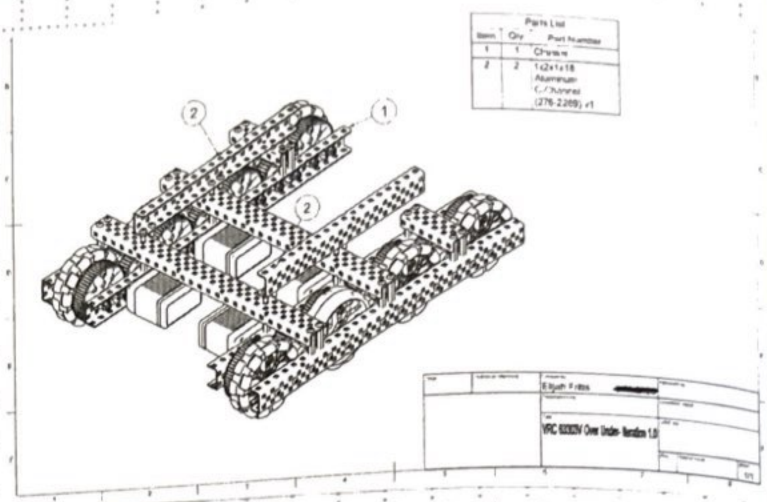
project Flywheel Supports designed by: Vanessa Perkins & Eli Fritts witnessed by: Owen Taylor
date: 5/20/2023

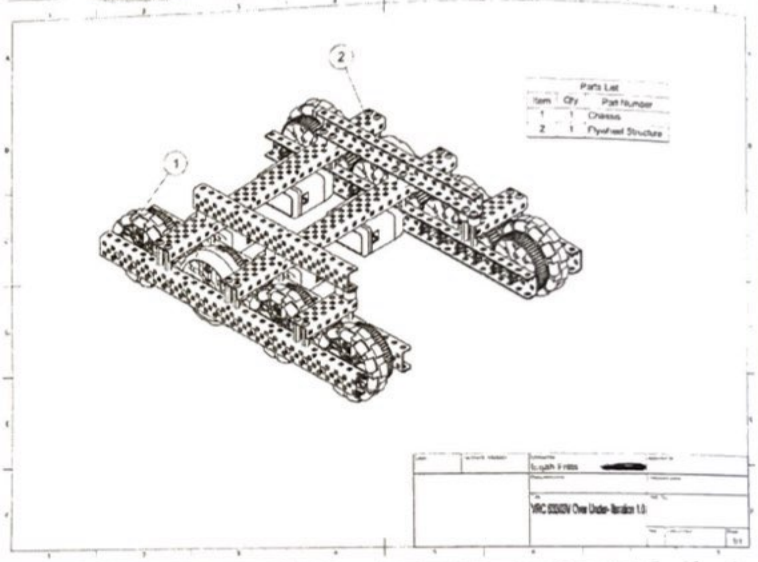
Iteration 2 Programming Building Problem

Date: 5/20

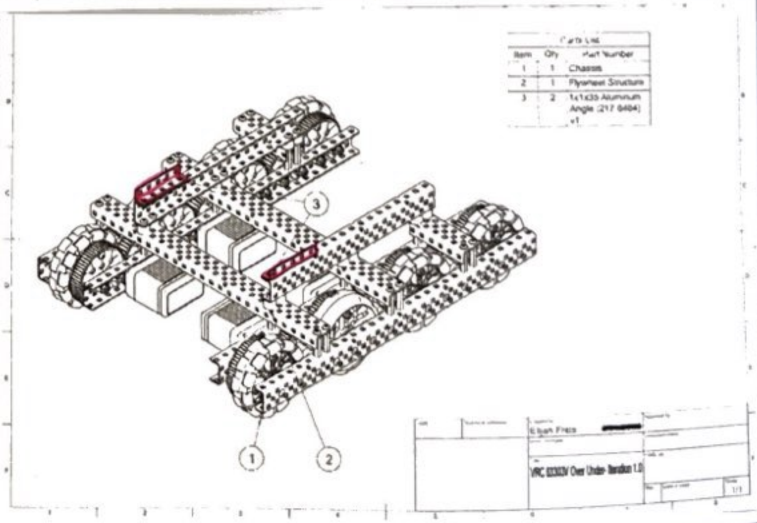
(Continued)

(3) Replace the two(2) C-channels from Step 2 into the locations of the removed screws from Step 1. Tighten so that new C-channels are parallel to the wheels and have spacers underneath. The 2nd picture to the right serves just to show the new metal completely added and tightened.





(4) Gather/cut two(2) five(5) hole long 1by1 Aluminum L angles. Place in the depicted locations to the right on top of the newly added C-channels towards the back end.
The new 1by1 angles are outlined in pink to more clearly see their placement.



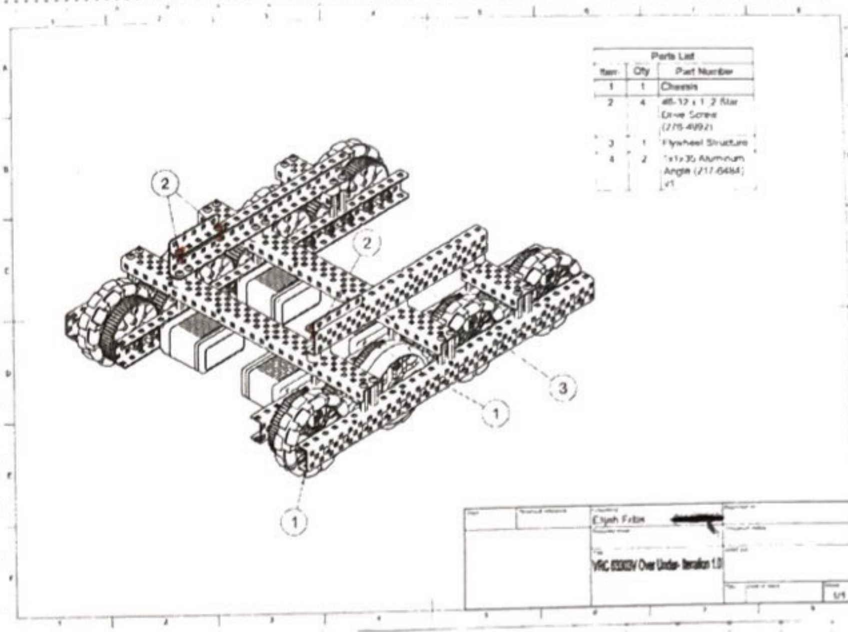
VP

project Flywheel Supports (Continued) designed by: Vanessa Perkins + Eli Fritts witnessed by: Harris Perkins date: 5/20/2023

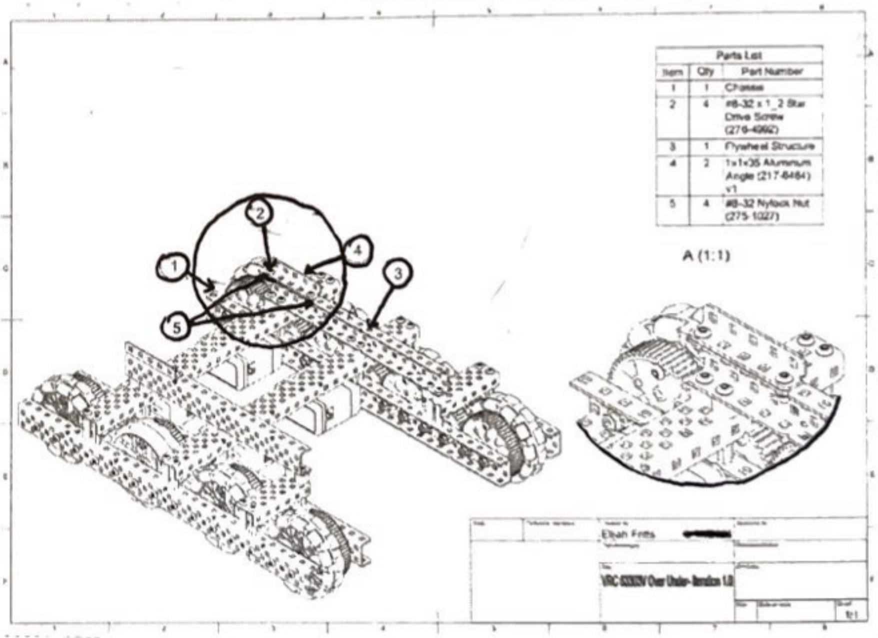
Date: 5/20

(Continued)

5. Add two (2) half inch screws* to each end on the 1by1 angles. Add two (2) nylock nuts* to the bottom and tighten. Repeat for the otherside for a total of four (4) screws and four (4) nylock nuts used.



To the right is a CAD image with a closer diagram of each of the parts just added for the right side. Mirror for the left side. Below is the complete parts list for the Flywheel Supports Structure.



Complete Parts List:

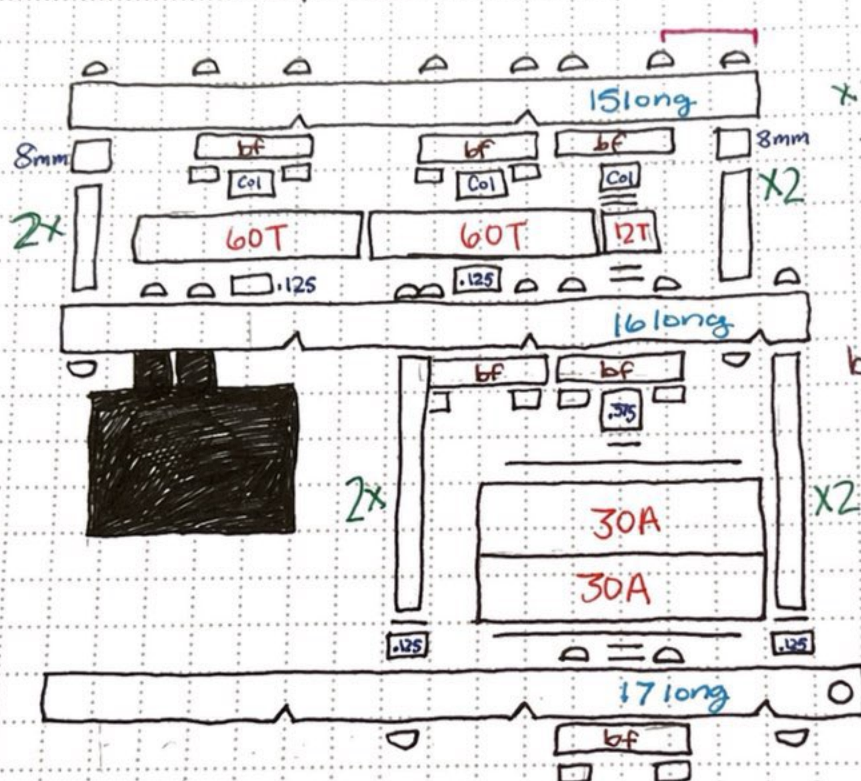
- 2 - 18 hole aluminum C-channels
- 2 - 5 hole 1by1 aluminum L-angles
- 12 - 0.375 inch spacers
- 12 - 0.750 inch screws
- 4 - 0.500 inch screws
- 4 - nylock nuts

project: Flywheel Supports (Continued) designed by: Vanessa Perkins & Eli Fritts witnessed by: Owen taylor date: 5/20/2023

Date: 5/25

Begin Building Flywheel:

Goals for today → finish atleast one side of the flywheel
→ attach flywheel to base once both sides are done being built



x2: This means to repeat twice because the 2nd time is hidden behind the first.

bf: bearing flats
—: Teflon washers
◠: screws
▭: Nylock nuts
▯: standoffs

Above is a diagram of the Left side from the Front of the flywheel. The parts list and images of the complete flywheel are on the next page (page 38). Below is the general steps we took to build each half of the flywheel.

① Gather two(2) fifteen(15) hole long aluminum c-channels, two(2) sixteen(16) hole long aluminum c-channels, and two(2) seventeen(17) hole long aluminum c-channels. Add the above depicted six(6) bearing flats to the insides for a total of twelve (12) bearing flats. Attach with half inch screws and nylock nuts on the outside holes.

② Gather ~~[illegible] (4)~~ shafts for each side, two(2) three(3) and one(1).
Begin adding the correct spacing from the above diagram with

project: Begin Building Flywheel designed by: Vanessa Perkins witnessed by: Eli F[illegible]
date: 5/25/2023

Date: 5/25

(Continued)

② Continued... the flywheel on the same axel as the 12 tooth gear.
③ Attach both motors to the sixteen (16) long c-channels on the center row in line with the bearing flats.
④ Insert the shafts with correct spacing to the correct locations on c-channels through the bearing flats.
⑤ Attach the fifteen (15) long c-channel to the sixteen (16) long c-channel using an 8mm spacer and 1 inch standoff for each of the four (4) corners for both sets. Use a 0.750 inch screw on one side and a 0.500 inch screw on the other.
⑥ Attach the sixteen (16) long c-channel to the seventeen (17) long c-channel using 3.000 inch standoffs, a Teflon washer, and 0.125 inch spacer. Use half inch screws for each side. Place in the depicted corners on the previous page for a total of four (4) attachment points on each set.
⑦ Attach both sets of flywheels with a twenty (20) hole long aluminum c-channel, four (4) two (2) inch standoffs, six (6) half inch screws and two (2) couplers. Remove the top outer half inch screws from the back of the flywheel and replace with the couplers. Set standoffs in a row in the highlighted place* on the previous page.
⑧ Attach to base of robot using two (2) half inch screws, two (2) teflon washers, and two (2) nylock nuts. Add screw then washer through the back hole of the 1 by 1 L-channel on the base then through the hinge point on the bottom of the flywheel then washer and nut. (Simple hinge point prototype to the left.)

project: Begin Building Flywheel designed by: Vanessa Perkins witnessed by: Owan taylor
date: 5/25/2023

Date: 5/26

Complete Flywheel Parts List:

Below is the complete list of parts for the flywheel build.

Metal:
- 2-15 long aluminum c-channels
- 2-16 long aluminum c-channels
- 2-17 long aluminum c-channels
- 20 long aluminum c-channel
- 4-2.5 inch shafts
- 2-6 inch shafts

Screws/nuts/Standoffs:
- 60-half inch screws
- 2-0.750 inch screws
- 26 Nylock nuts
- 8-1.000 inch standoffs
- 8-3.000 inch standoffs
- 4-2.000 inch standoffs

Wheels/Gears:
- 4-30A 3 inch flexwheels
- 4-Aluminum Versa Hubs/ Flywheel weights
- 4-60 tooth high strength gears
- 2-12 tooth high strength gears

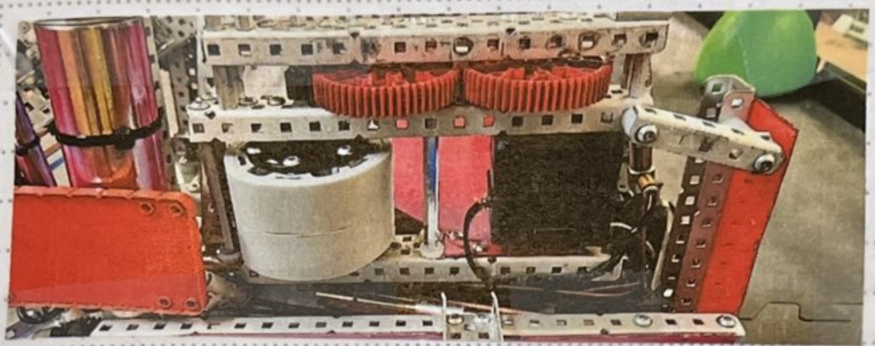
Miscellaneous:
- 2 motors
- 12 bearing flats
- 2 couplers
- 2-600 rpm low torque, high speed, blue cartridges

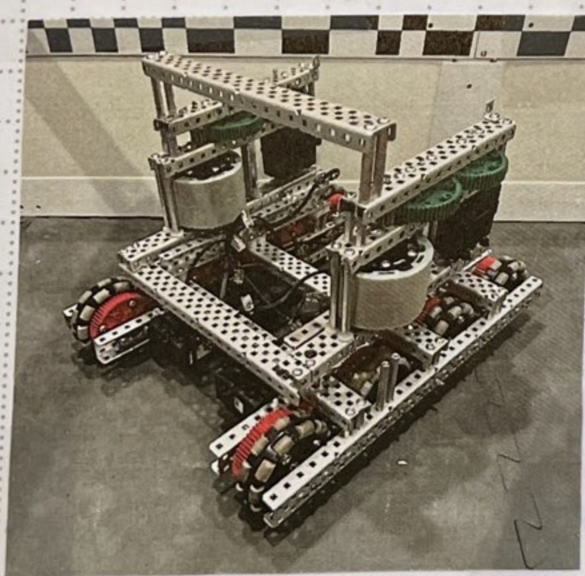
Spacers:
- 26 Teflon Washers
- 8-8mm spacers
- 6-0.125 inch spacers
- 2-0.375 inch spacers
- 6-shaft collars

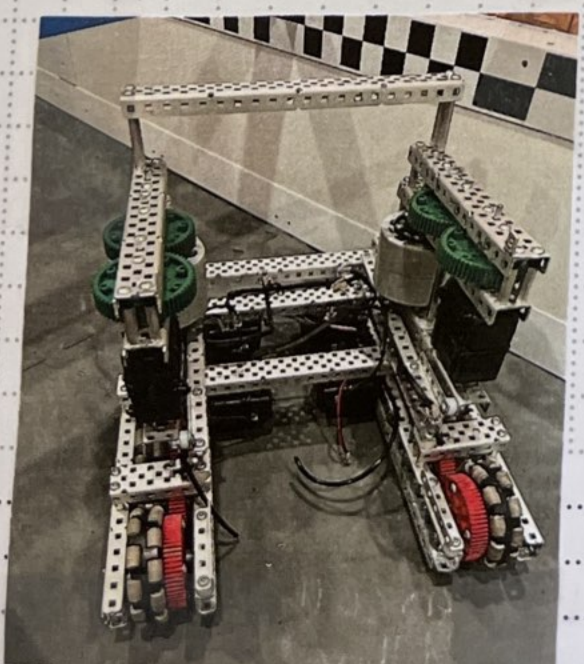
project Complete Flywheel Parts List designed by: Vanessa Perkins witnessed by: Eli Fritts
date: 5/26/2023

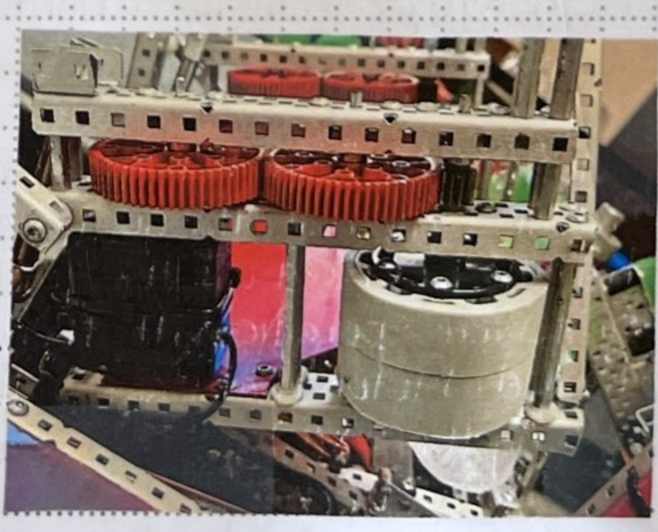
Date: 5/27

Finished Flywheel Images:









project Finished Flywheel Images designed by: Vanessa Perkins witnessed by: Harris Perkins

date: 5/27/2023

Date: 6/25

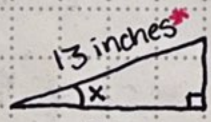
Testing Flywheel:

Goals for today → Program Flywheel
→ Setup and test flywheel for function and accuracy

Below is the set up and conditions for our testing as well as the data and conclusion.

Purpose: Test flywheel for function and accuracy of the build and programming when scoring in the goal from across the field.

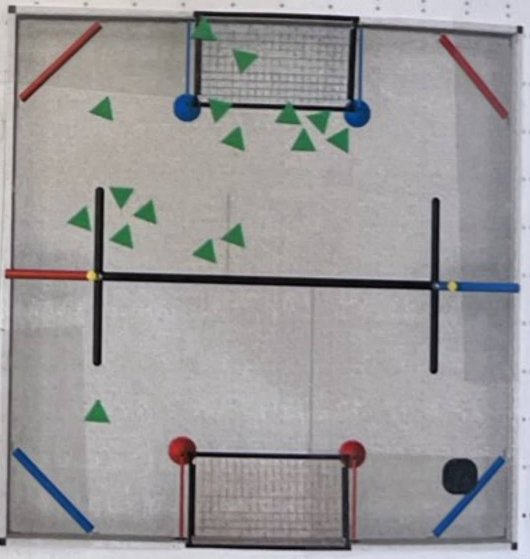
Apparatus: We set up the robot by one of the blue matchload zones at approximately a 13° angle based on our below trigonometry calculations and picture to the ~~left~~ right.



$\sin^{-1}(3/13) \doteq 13°$

* The thirteen (13) inches is the matchload zone barrier side with the twelve (12) inches representing the robot's measurement.

Data: To the right is a scatterplot of each of the sixteen (16) triballs and where they landed. The black square in the lower righthand corner represents the robot's location.



Conclusion: No more than three (3) triballs made it in the goal. Many triballs bounced off the goal and were sporadic with their launch. Motor heat was monitored during the six (6) minute testing duration so that wasn't the problem. Checked programming and found there was spin because one motor was at 100 pct while the other was at 95 pct.

project Testing Flywheel designed by: Vanessa Perkins witnessed by: Eli Frutto
date: 6/25/2023

Date: 6/25

(Continued)

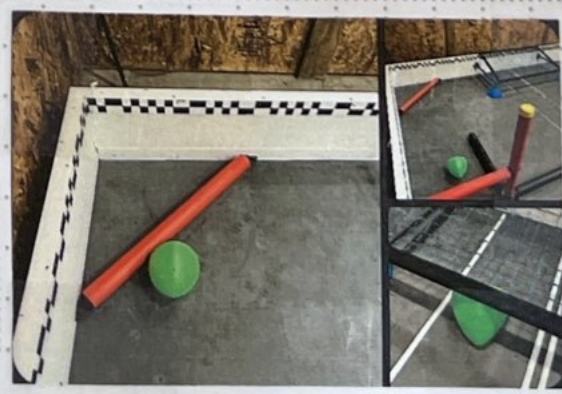
Goals for today → Fix error in program and retest Flywheel

Our previous test helped us discover that the flywheel programe was creating Spin because one side was at 100 pct while the other was at 95 pct. We fixed the error so both motors were set to 100 pct. Below were the results.

Purpose: Test to make sure program is fixed.

Apparatus: Same set up and conditions as Test 1 on page 40 with motor heat monitored and sixteen (16) triballs launched.

Data: To the right is a scatterplot of where each of the sixteen (16) triballs landed on the field. Below is a few images of where the triballs landed and were recorded on the plot above.



Conclusion: About seven (7) triballs made into the goal which showed improvement. Note that most of the errors were caused because of the wobble, meaning they hit the goal higher or the ground first leading them to bounce away.

Moving forward, our solution to improve the performance of the flywheel is to eventually add an aligner to reduce wobble since we now know the program is functioning to the best of it's ability.

project Testing Flywheel (Continued) designed by: Vanessa Perkins witnessed by: Eli [illegible]
date: 6/25/2023

Date: 6/26

Redesign Flywheel:

Goals for today → Redesign flywheel for better configurement for future subsystem builds

Identify the Problem/Challenge:

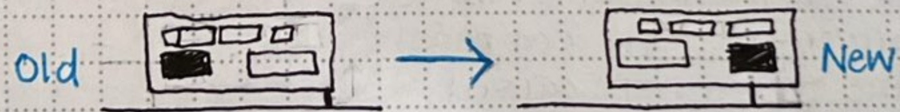
When planning for future subsystems, we realized that the placement and orientation of the flywheel on the base was unpractical and unefficient. It did allow proper room for the intakes and indexor particularly in the front.

Brainstorm:

1. Flip direction of flywheel from motors in front to motors in back.
2. Shorten/Change size of flywheel.
3. Extend the base size.
4. Change the hinge point to further in, instead of the furthest edge.

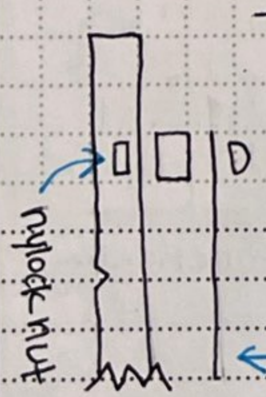
Select Option:

1 and 2 We decided on a combination of flipping the flywheel's orientation and hinging it further in to help slid in back in the robot further.



Build/Implement Solution:

- Detach flywheel from hinge points
  - Flip flywheel so the motors are towards the back of the base.
  - Attach with 0.500 inch screws and nylock nuts with a 0.250 inch spacer between the 17 long c-channel of flywheel and 1by1 5 long L-channel of base. Use two (2) screws, two (2) nylocks, and two (2) spacers total. The diagram is to the left.



← 1 by1 5long L-channel

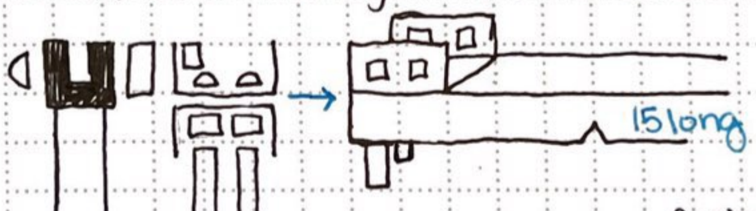
project Redesign Flywheel designed by: Vanessa Perkins witnessed by: Harris Perkins date: 6/26/2023

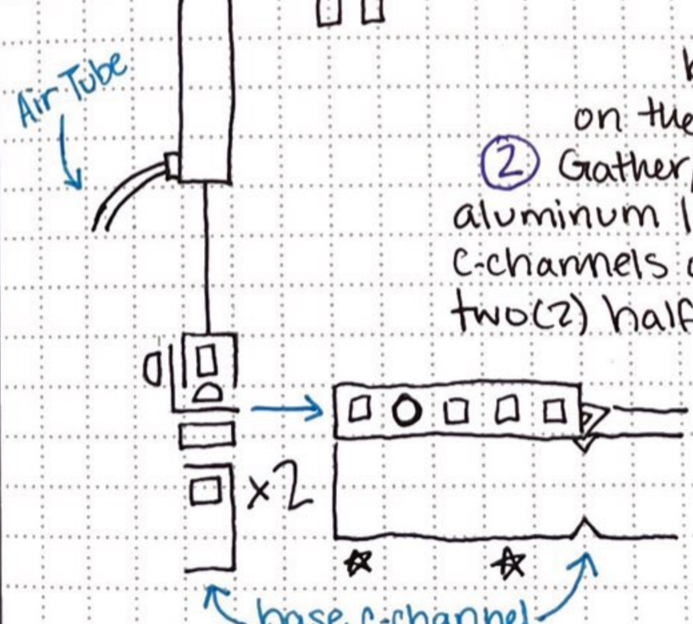
Date: 6/27

Attach Pistons to Flywheel:

Goals for today → Attach pistons to the back of the flywheel so it can raise up or down/change launch angles easily

To help improve the flexibility of our robot when it comes to strategies and offense, we want to add pneumatic pistons to the flywheel so we can change it's angle of launch during matches. The pistons will cause the flywheel to rotate up and down on the hinge point. When it is raised, we can launch matchload triballs over defensive triballs trying to block us. Below is the build diagram for one side.





① Gather/cut two(2) aluminum C-channels two(2) holes long. Attach to 15 long C-channel on top of flywheel by removing the two(2) screws on the end and replacing on top.

② Gather/cut two(2) five(5) hole long aluminum 1by1 L-channels. Attach to inside C-channels of base on the very edge using two(2) half inch screws for each side, a 0.125 spacer in the middle, and nylocks to tighten it.

③ Attach top of a Double Acting Piston (with only one(1) tube on lower end) to the two (2) long C-channel end inside hole using a one(1) inch screw, ~~0.125 inch~~ VP 4.6mm spacer inbetween, and a nylock nut.

★ → Holes that the half inch screw goes through to base C-channel with 0.125 inch spacer inbetween and a nylock nut holding it.

④ Attach bottom of piston to the 1by1 L-channel using a half inch screw, ~~0.125 inch screw inbetween~~ VP and a nylock nut on the inside. The circle represents the screw location on the 1by1 L-channel.

project: Attach Piston to Flywheel designed by: Vanessa Perkins witnessed by: Owen taylor

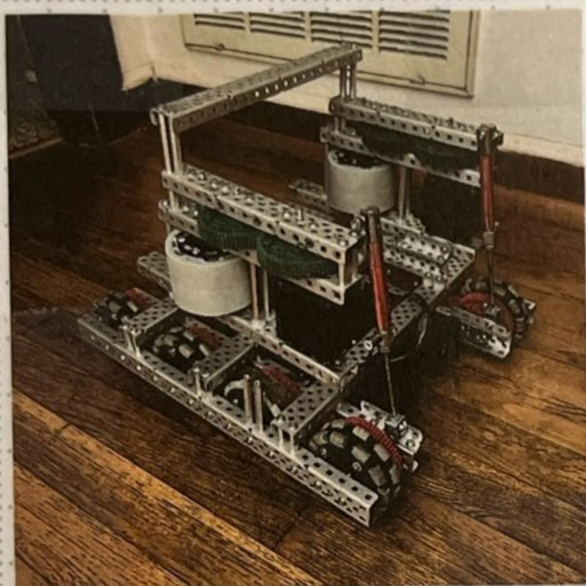
date: 6/27/2023

Iteration 2 Programming Building Problem

Date: 6/28

Flywheel Piston Parts List and Pictures:

Goals for today → Completely record the build of the pistons



Metal:
- 2-2 long aluminum c-channels
- 2-5 long 1by1 L-channels

Spacers:
- 2-4.6mm spacers
- 4-0.125 inch spacers

Nuts/Screws:
- 6-0.500 inch screws
- 2-1.000 inch screws
- 8 Nylock nuts

Miscellaneous:
- 2 Double Acting Pneumatic Pistons
- 2 Air Tubes for Pistons

The pistons remain extended when the flywheel is lowered. When the pneumatic pistons retract, the flywheel will tip back to raise the front and allow is shoot match loads over defensive robots trying to block us.

Date: 6/29

Brainstorming Indexor Ideas:

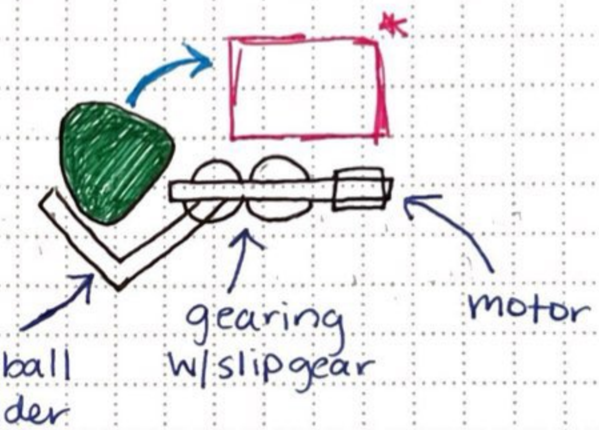
Goals for today → Research and come up with at least 3 indexor design options to select between

→ Respect and Listen to every team member's ideas

Design Option One: Geared Indexor

Specifics:

- 1 motor
- 1:1 gear ratio
- slip gear
- metal triball holder
  or
- molded Lexan triball holder



This indexor design uses a slip gear to help the robot reset into loading position then release and send the triball upwards towards the flywheel.

* The location of the flywheel on the robot.

The con of this design is that the 1:1 gear ratio means less teeth to grip on once the slip gear is cut. This design might require lots of adjusting too, but would allow us to do a ratchet off the gears for other mechanisms or to do a PTO or power take off for another mechanism

(Continued)

Design Option Two: Direct Intake

Specifics:

- Donot build any version of indexor

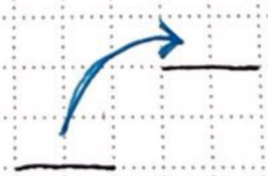
This design means we would skip the indexor and go straight to designing and building an intake that brings the triball from the field to the flywheel without the inbetween step.

The con is that this will be a big distance for a intake to move a triball, but it could potentially save motors.

Design Option Three: Chain bar lift

Specifics:

- 1 motor
- Chain
- Sprockets
- Forklift type bars



This design is very complex and chain can be unreliable as it breaks very easily, but the design keep triballs aligned and consistently oriented during its ascent to the flywheel. This is because there is no flipping or rotation as the other two designs.

project Brainstorming Indexor Ideas (Continued) designed by: Vanessa Perkins witnessed by: Harris Perkins date: 6/29/2023

Date: 7/1

Select Approach:

Goals for today → Choose an Indexor design that best meets the requirements of the game

We created a decision matrix of what aspects mattered the most on our indexor. Below is the matrix with the winning design.

Indexor Ideas	Speed	Consistency	Size	Total
Geared Indexor	4	3	5	12
Direct Intake	5	0	5	10
Chain Bar Lift	2	5	2	9

* Speed: how fast it moves triballs
Consistency: how well it moves triballs without getting misaligned or stuck
Size: how compact the indexor is and how well it fits inbetween the intakes and flywheel

Selected Design: Geared Indexor

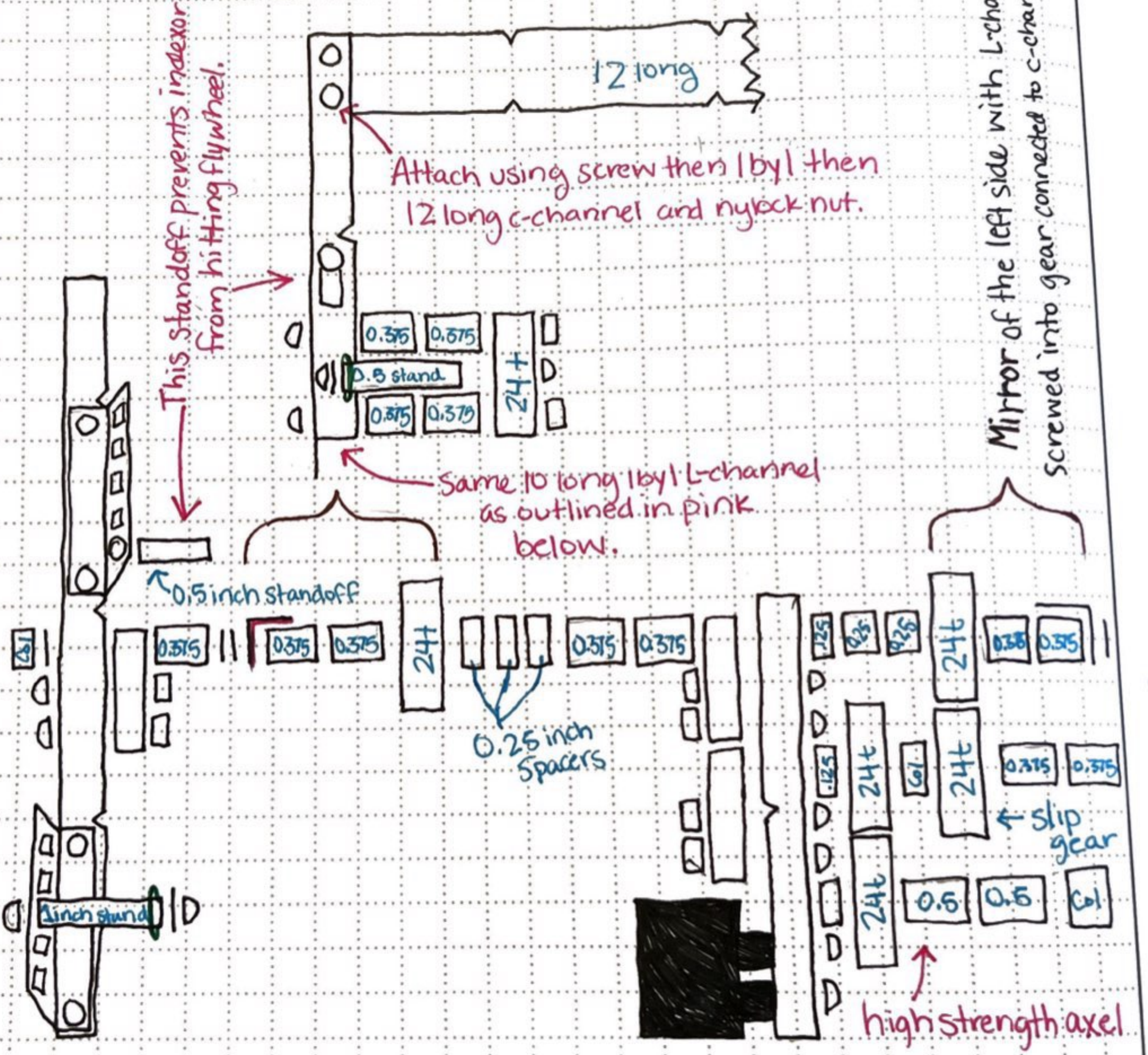
Explanation: The geared indexor design is a speedy, consistent and space friendly option. The Direct Intake is a close second but can not control the orientation of the triballs as the geared indexor can with a holder of some type. The geared indexor also allows us to build other mechanisms off of it easily if needed.

* Break down of each criteria aspect we looked at. A rank of zero (0) means bad and a rank of five (5) means good.

Date: 7/14

Begin Building Indexor:

Goals for today → Gather parts for Indexor
→ Begin assembling the gearing part of the Indexor



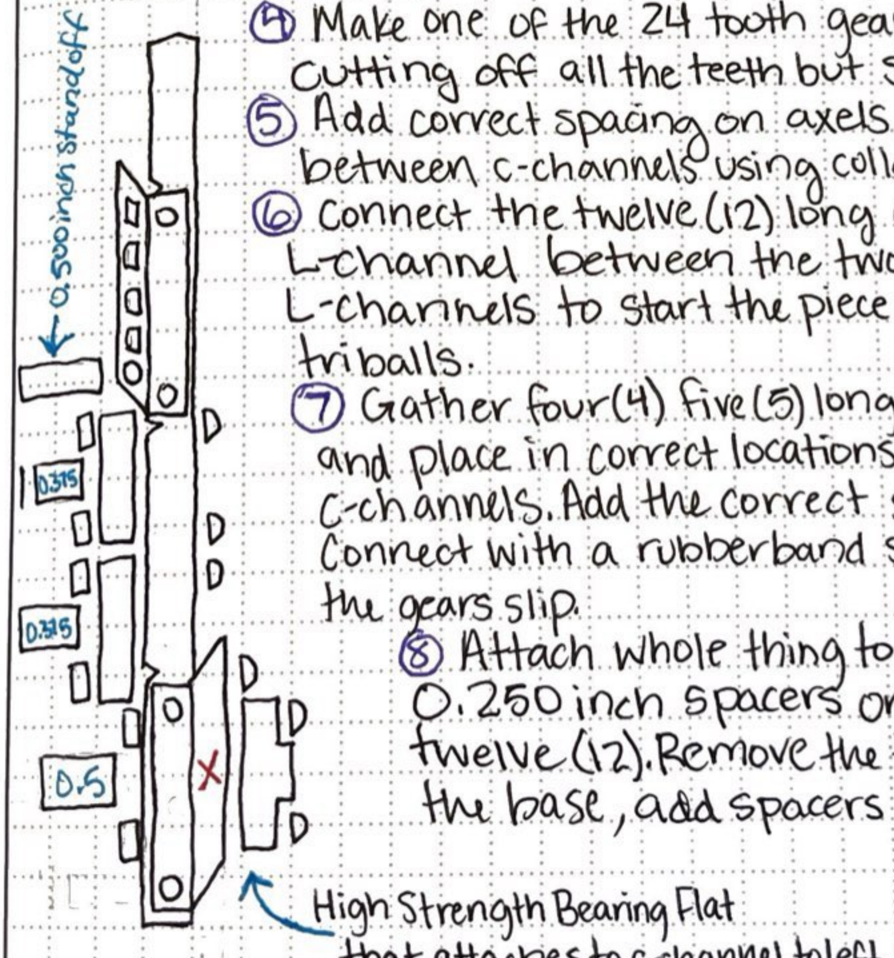
Above is a diagram of the indexor gearing with the first part of the indexor that holds the triballs at the top. The right side of the diagram continues onto page 49 with some additional information and steps on the build.

project Begin Building Indexor designed by: Vanessa Perkins witnessed by: Eli Fouts
date: 7/14/2023

Date: 7/14

(Continued)

1. Gather all metal including two (2) eighteen (18) long C-channels, one (1) ten (10) long c-channel, one twelve (12) long c-channel, one (1) twelve (12) long 1 by 1 L-channel, and two (2) ten (10) long 1 by 1 L-channels.
2. Add all six (6) bearing flats to their locations used two (2) screws and two (2) nuts each. Note: One (1) is a high strength bearing flat.
3. Attach the ten (10) long 1 by 1 L-channels to two (2) of the 24 tooth gears using the correct spacing and standoffs so the indexor will move with the rotation of the gear.
4. Make one of the 24 tooth gears a slip gear by cutting off all the teeth but six (6).
5. Add correct spacing on axels and tighten together between c-channels using collarlocks.
6. Connect the twelve (12) long c-channel and 1 by 1 L-channel between the two ten (10) long 1 by 1 L-channels to start the piece that will hold the triballs.
7. Gather four (4) five (5) long 1 by 1 L-channels and place in correct locations on the eighteen (18) long c-channels. Add the correct standoffs on each side. Connect with a rubberband so it will snap up when the gears slip.
8. Attach whole thing to base using six (6) 0.250 inch spacers on each side for a total of twelve (12). Remove the twelve inside screws from the base, add spacers and indexor, then replace screws.



X - The X is a mirror of the left side's screw, inch long standoff, Teflon washer, and screw with rubberband, but was left out of the drawing to not obstruct other parts view.

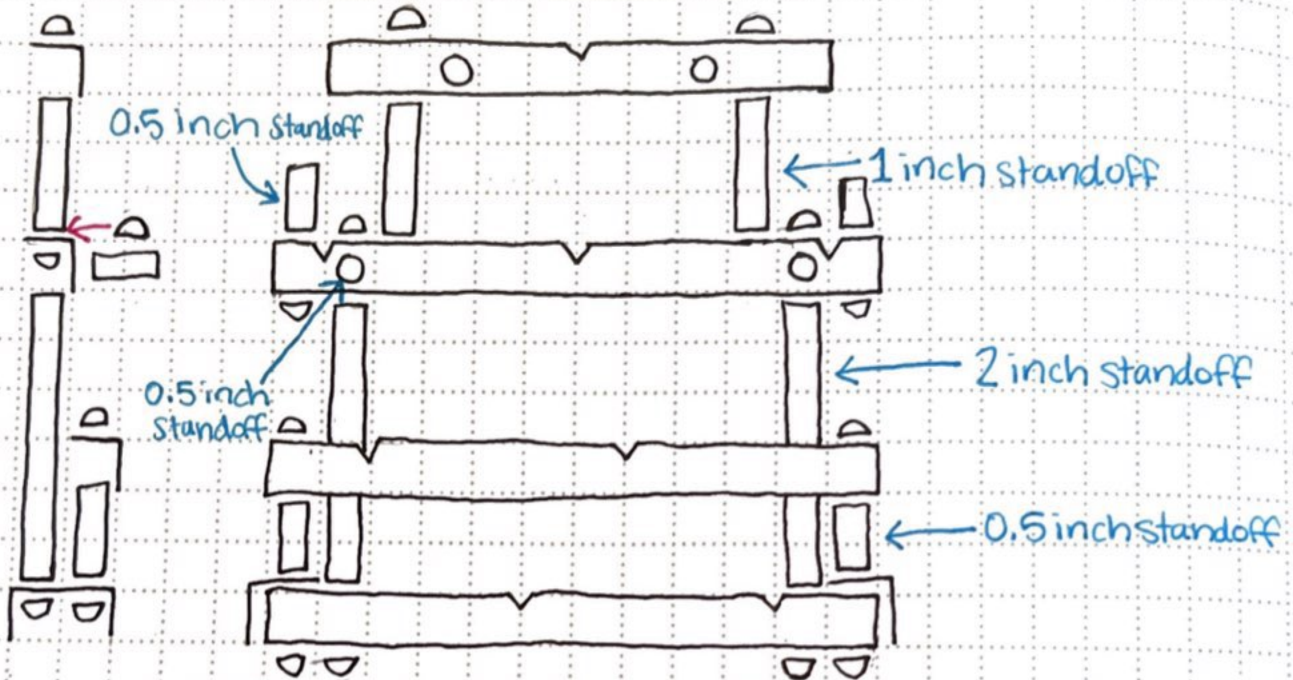
D - Rubberband goes between the half inch standoff and one inch standoff.

project: Begin Building Indexor (continued) designed by: Vanessa Perkins witnessed by: Harris Perkins date: 7/14/2023

Date: 7/15

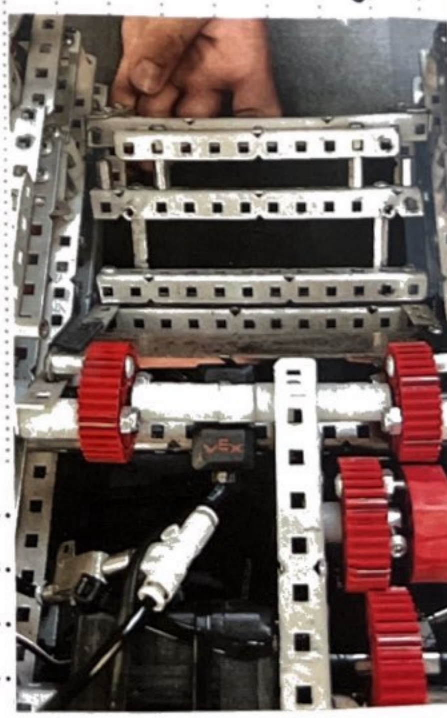
Finish Building Indexor:

Goals for today → finish the indexor
→ make sure triball fits well inside



Side View Top View

Above is the diagram for the rest of the indexor from two angles. To the right is an image of this part of the indexor close up and finished. The bottom 12 long c-channel and 2 1 by 1 L-channels are the same ones on page 48 attached to the 24 tooth gears.



Once complete, we tested it with a triball and found the problem that it would not stay aligned well inside the indexor to properly move to the flywheel.

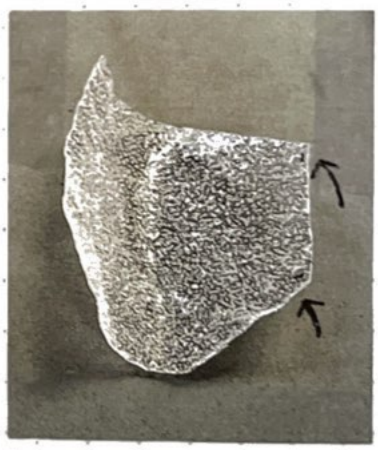
Date: 7/16

Indexor Triball Mold:

Goals for today → build something to hold the triballs in the indexor better
→ test different options for best one

Orginally, we would just have standoffs on the indexor to hold the triball as it moves up to the flywheel, but it was not consistent at keeping the triball aligned.

We brainstormed options to fix this alignment problem and decided on molding a sheet of Lexan to fit the triball. We drilled two (2) holes in the corner so it could be screwed onto the indexor using two (2) half inch screws and two (2) locknuts in the 3rd holes from the edge on the ten (10) long 1by1 L-channel. To the right are pictures of the finished mold. Below are the tests we did to get the right mold.



Test 1: Bake in oven at 500 degrees fahrenheit for six (6) minutes. Remove then press to shape of triball. Trim when cooled if needed.
Results: The Lexan did not get heated long enough so it remained too stiff to properly mold.
Test 2: Same steps except bake for nine (9) minutes at 500 degrees fahrenheit.
Results: Better results, but did not mold quick enough before it cooled completely. This result will be sufficient for the time being and helps the triballs align in the indexor much better than rigid, boxy metal and standoffs.

project Indexor Triball Mold designed by: Vanessa Perkins witnessed by: Harris Perkins
date: 7/16/2023

Complete Indexor Parts List:

Metal:
- 4-5 long 1by1 aluminum L-channels
- 3-10 long 1by1 aluminum L-channels
- 2-12 long 1by1 aluminum L-channels
- 10 long aluminum C-channel
- 12 long aluminum C-channel
- 2-18 long aluminum C-channel
- 1-8.5 inch drive shaft
- 1-4 inch drive shaft
- 1-4.5 inch high strength drive shaft

Screws/Nuts/Standoffs:
- 55 half inch screws
- 4-1.250 inch screws
- 22 nylock nuts
- 2-1.000 inch standoffs
- 2-2.000 inch standoffs
- 9-0.500 inch standoffs

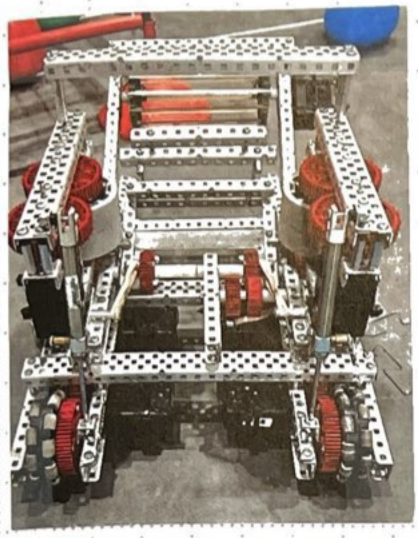
Miscellaneous:
- 1 motor
- 5 bearing flats
- 1 high strength bearing flat
- 2 rubber flaps with drilled hole on 1 end
- Lexan Mold → 2 half inch screws + 2 nylock nuts needed to attach
- 1-100rpm high torque, low speed, red cartridge
- 4 rubberbands

Spacers:
- 19-0.375 inch spacers
- 9 Teflon Washers
- 5-0.250 inch spacers
- 14-0.125 inch spacers
- 2 shaft collars
- 1 high strength shaft collar
- 1-0.125 inch high strength spacer
- 3-0.500 inch high strength spacers

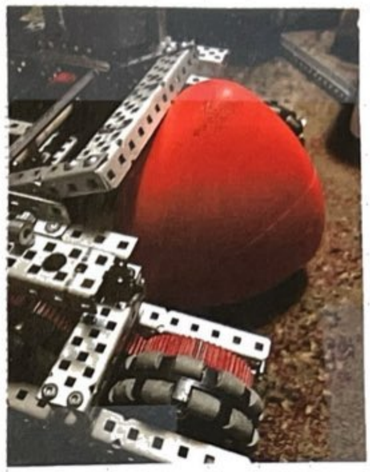
Gears:
- 5-24 tooth high strength gears

project: Complete Indexor Parts List designed by: Vanessa Perkins witnessed by: Harris Perkins
date: 7/16/2023

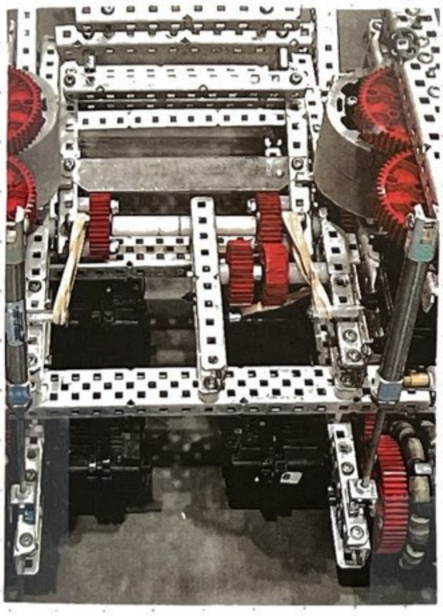
Finished Indexor Images:

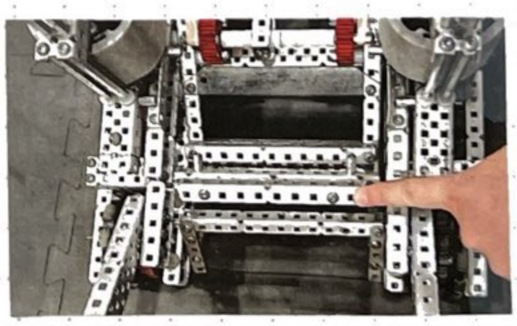


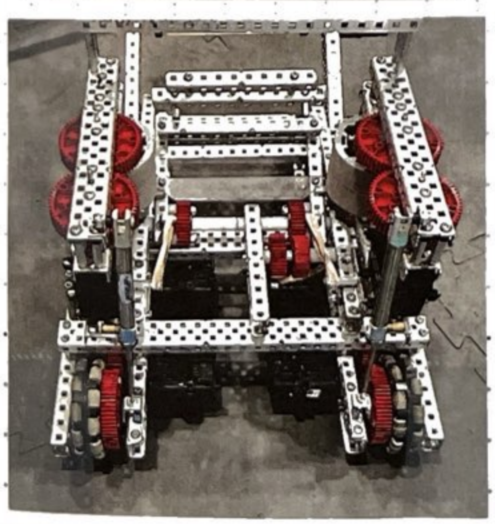
To the left are images take from the back of the robot looking at the indexor gearing ← mainly.



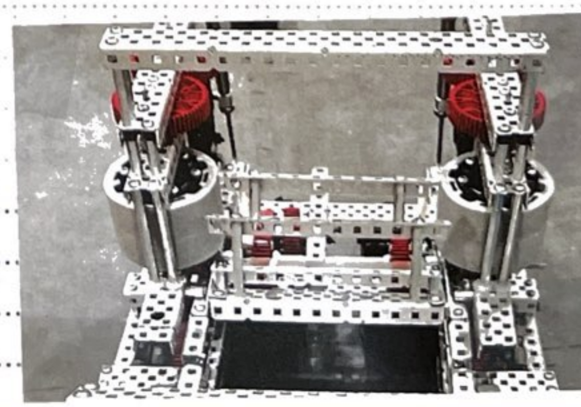
To the right are images of the front of the indexor where the triballs will sit and be lifted. →







VP



project Finished Indexor Images designed by: Vanessa Perkins witnessed by Owen taylor
date: 7/16/2023

Identify Problem and Brainstorm for Scoring:

Date: 7/16

Goals for today → Discuss the problems with the function of the robot when going to score
→ Brainstorm solutions to improve scoring performance

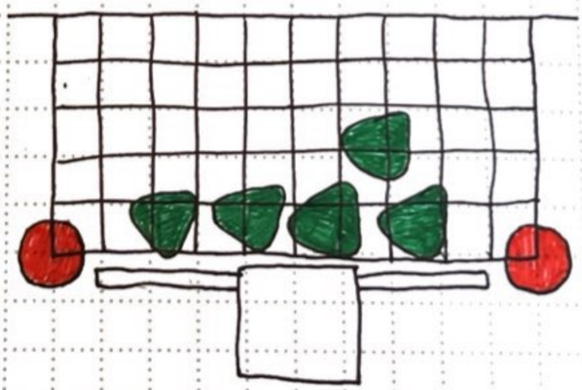
Identify the Problem:

The flywheel works well to launch triballs from the match load zone to in front of the goal. But when we drive our robot over to push them the rest of the way into the goal, it is to time consuming to go one by one.

Brainstorm:

We began researching what other teams were doing to efficiently push triballs into the goal.

We found a video from the YNOT Alliance with four robots involved. Specifically the FlapJack Robot Design was our focus for a solution. The robot had fold out wings made of metal to span the width of the goal and help push a large amount of triballs in at a time.



This is a simple, space efficient solution that will take minimal time to build and record in the notebook.

As well as scoring, when the opposite team is double zoning, this design can easily descore triballs by scooping them out using the reach and movement of the wings.

On the next page is the beginning of building our wings.

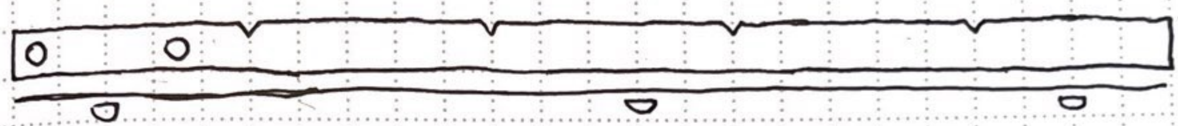
project Identify Problem and Brainstorm for Scoring designed by: Vanessa Perkins witnessed by: Harris Perkins date: 7/16/2023

Date: 7/16

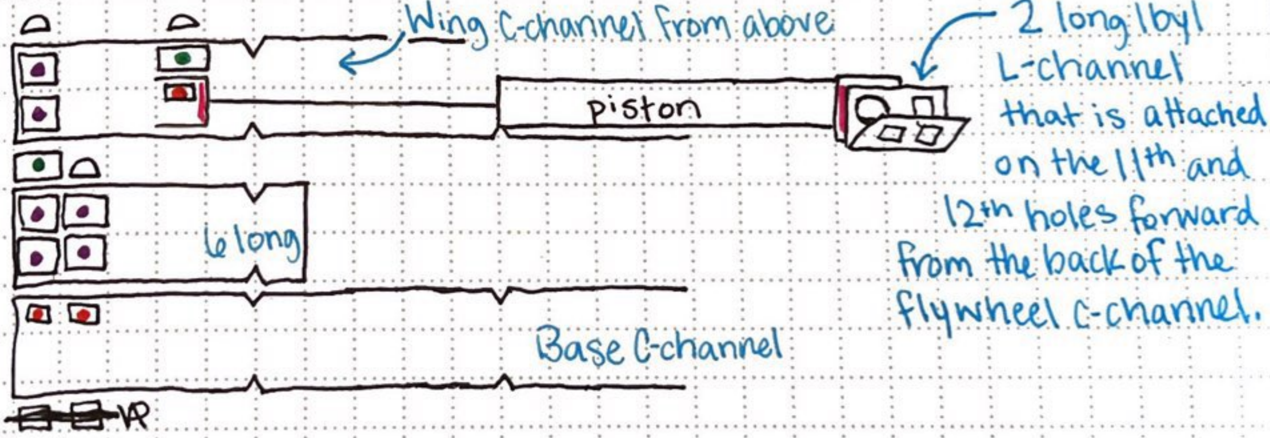
Begin Building Wings:

Goals for today → begin building wings
→ finish atleast one side

Below is a top view diagram of the right side wing with the Lexan down the side attached by three (3) half inch screws and nuts.



Below is the diagram of the wing from the inside side with all the spacing and hinging.



Parts Key:
0.375 inch spacer •
0.125 inch spacer •
Nylock nut •
Screws ⌓ or O
Rubberband \

(Lexan dimensions are located on page 56 with the parts list.)

To the left is a color coded key to tell which pieces are which easier. The long, outside screw is the hinge point that allows the pneumatic piston to push the wing in and out. We finished building the right side and lastly added a rubberband from end to end of the piston (denoted by the pink lines) to help pull the wings back in when the air is turned off.

We will mirror these diagrams for the left side wing on another day.

project: Solution: Begin Building Wings designed by: Vanessa Perkins witnessed by: Eli Fruitts
date: 7/16/2023

Date: 7/18

Complete Wings/Part List:

Goals for today → Build left side wing
→ Compile a parts list

We began the day building the left side wing which was an exact mirror of the diagram on page 55.

Then we wrote down and recorded in the notebook all the parts used for the set of wings. Below is the complete parts list.

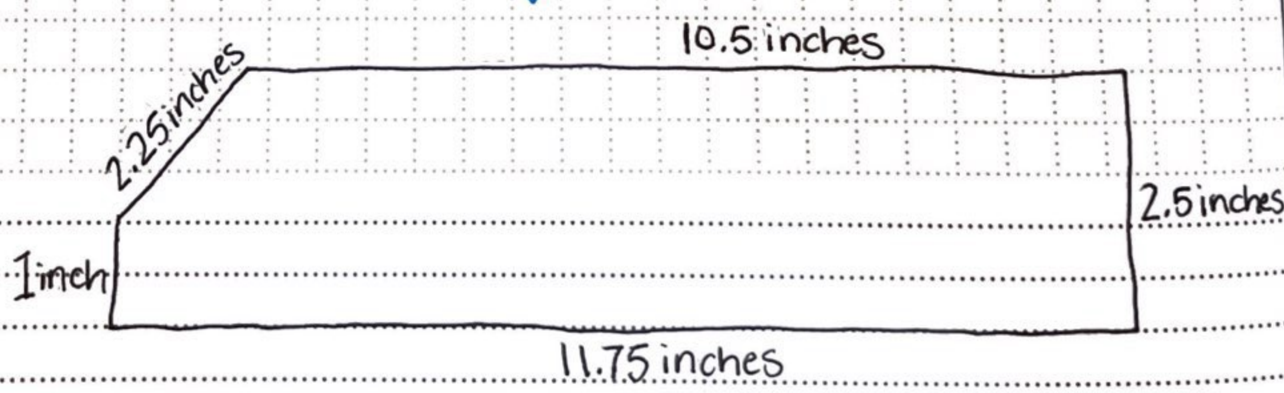
Metal:
- 2-6 long aluminum c-channels
- 2-24 long aluminum c-channels
- 2-2 long 1by1 aluminum L-channels

Screws/nuts:
- 2-2 inch screws
- 2-1 inch screws
- 14-half inch screws
- 9 Nylock nuts

Spacers:
- 12-0.375 inch spacers
- 4-0.125 inch spacers

Miscellaneous:
- 2-single acting pistons
- 2 rubberbands
- 2 Lexan sheets (dimensions below)

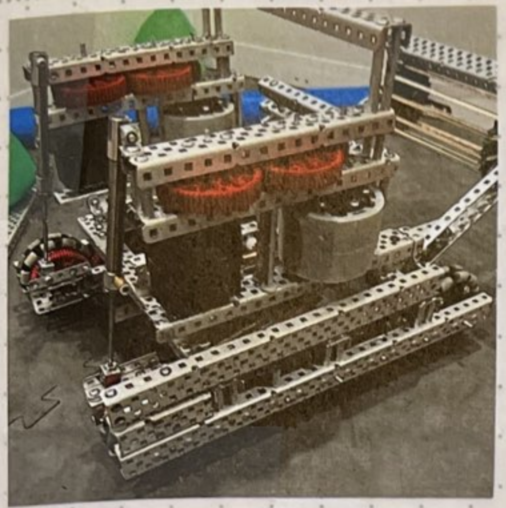
↓

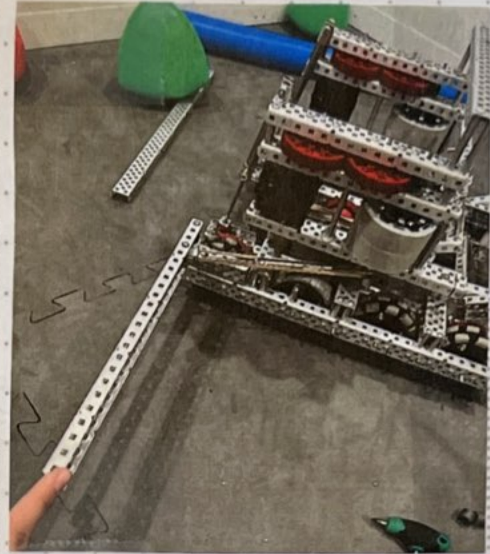


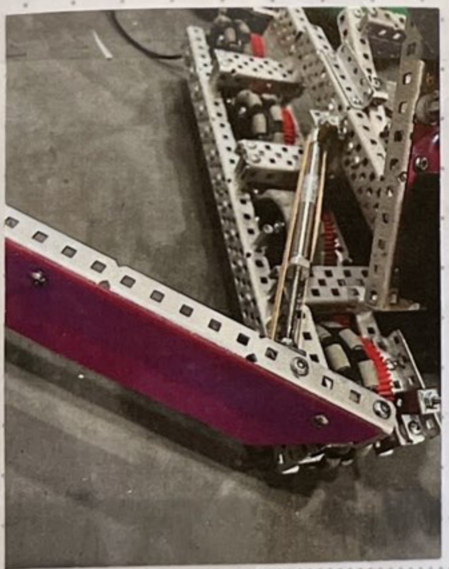
project Complete Wings/Part List designed by: Vanessa Perkins witnessed by: [illegible]
date: 7/18/2023

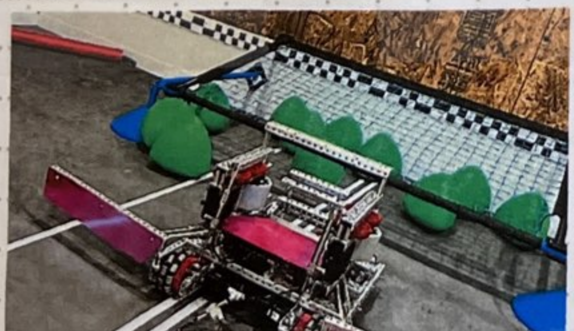
Date: 7/18

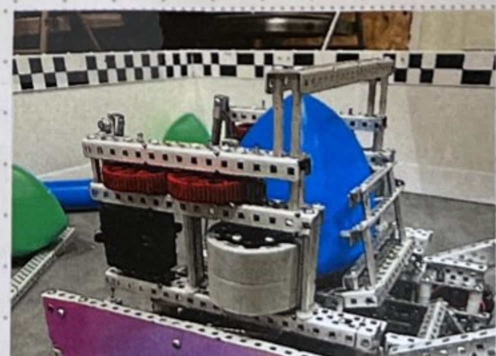
Finished Wings Images:











project Finished Wings Images designed by: Vanessa Perkins witnessed by: Eli Aritz

date: 7/18/2023

Date: 7/21

Identify Problem: Bending Pistons:

Goals for today ⟶ Discuss the problem of pistons bending and brainstorm solution options

Identify the Problem:

When driving around the field, the pistons on the back of the robot attached to the flywheel are unprotected. When Owen drives, there is high risk of the pistons being bent and rendered unuseable.

To prevent any possible damages, we wanted to take today to brainstorm and implement a solution for our exposed pistons.

Brainstorm:

① Move location of pistons to more sheltered location.
Con: Not a lot of relocation options.

② Ziptie a piece of metal to the piston that will move with it to block it from hits.
Con: Not very sturdy or permanent.

③ Add a bar to the back of the base to prevent ramming from bending anything.
Con: Will have to make cut outs for wheels

④ Make hinged, metal structure guards with lexan sheet covers that pivot with the movement of the piston.
Con: Will require complex arrangement of structure and hinging.

Solution:

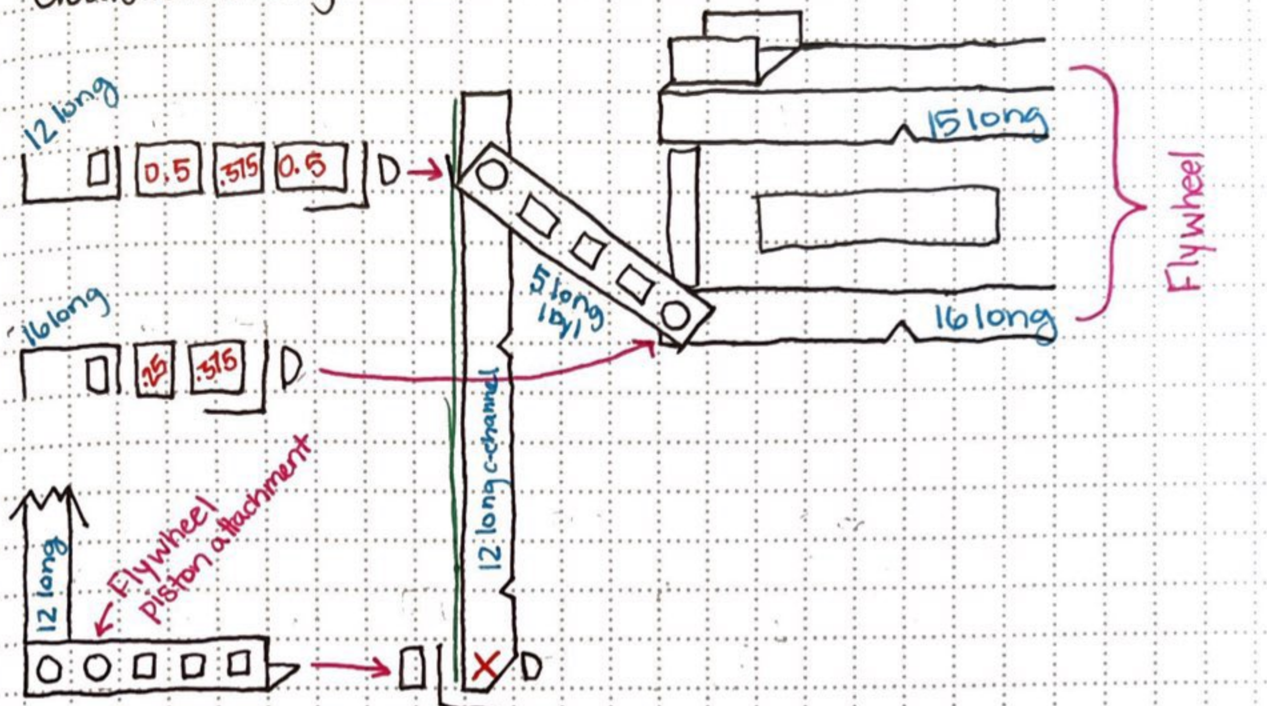
④ Build a hinged cover that pivots with the movement of the piston

project: Identify Problem: Bending Pistons designed by: Vanessa Perkins witnessed by: Owen Taylch
date: 7/21/2023

Solution: Build Piston Guards:

Date: 7/21

Goals for today →



The green represents where a 2inch by 5inch sheet of Lexan goes using two(2) half inch screws and two(2) nylock nuts.

The 5 long 1by1 L-channel and the bottom of the 12 long C-channel are the hinge points that tip back with the retraction of the pneumatic piston and tipping upward of the whole flywheel.

We repeated the above diagram mirrored for the otherside so both sides would have a guard.

X- Half inch screw goes through this hole into 5 long 1by1 L-channel attached to base then to nylock nut. The tip of the c-channel is filed to allow it to rotate on that screw inside the L-channel.

project Solution: Build PistonGuards designed by: Vanessa Perkins witnessed by: Owen Taylor
date: 7/21/2023

Complete Pistons Guards Parts List:

Below is all the parts used for the two(2) Piston guards built on page 59.

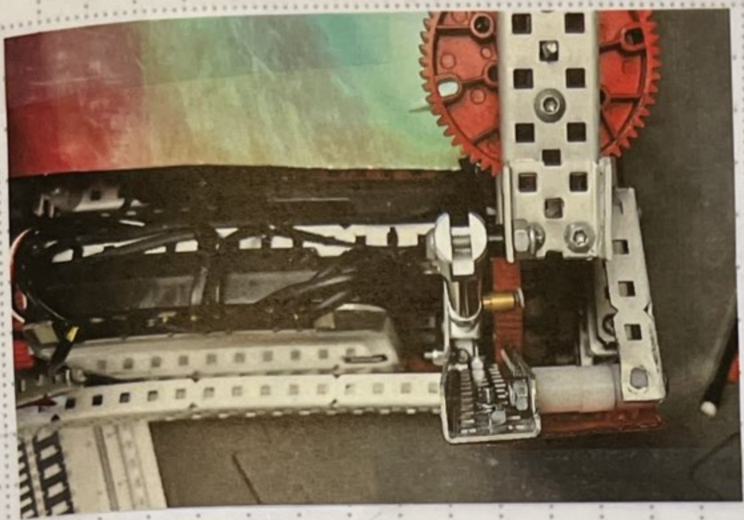
Metal:
- 2-12 long aluminum c-channels
- 2-5 long aluminum 1by1 L-channels

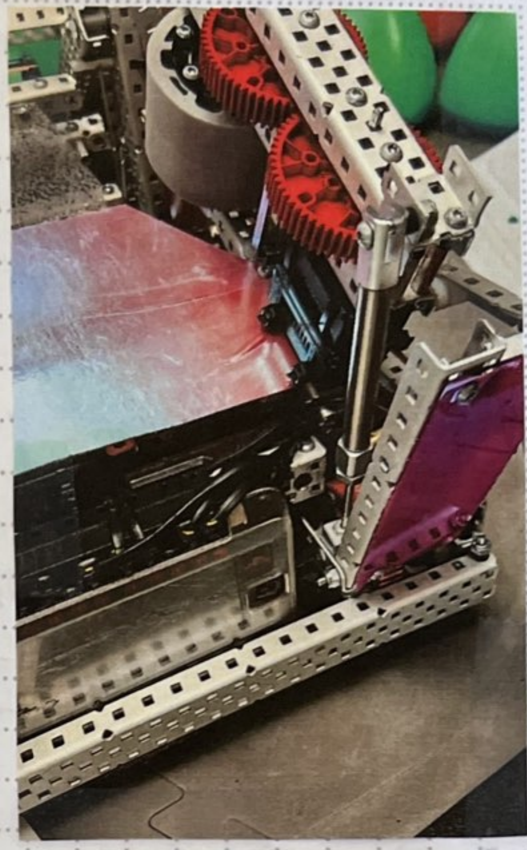
Screws/Nuts:
- 6-0.500 inch screws
- 2-1.500 inch screws
- 2-1.000 inch screws
- 10 Nylock nuts

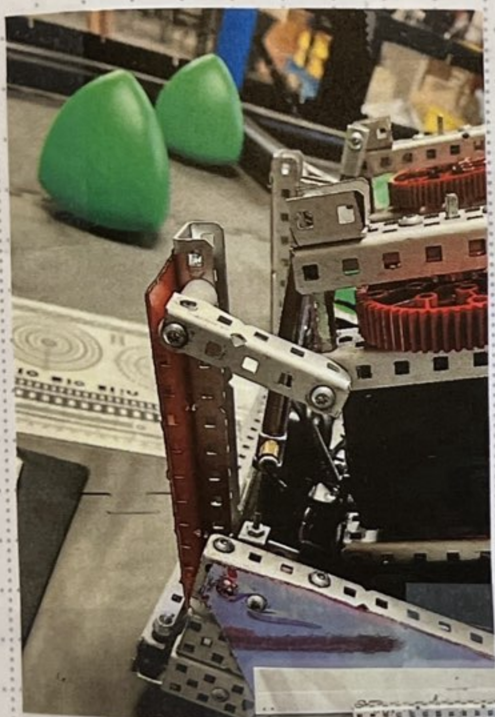
Spacers:
- 2-0.250 inch spacers
- 4-0.375 inch spacers
- 4-0.500 inch spacers

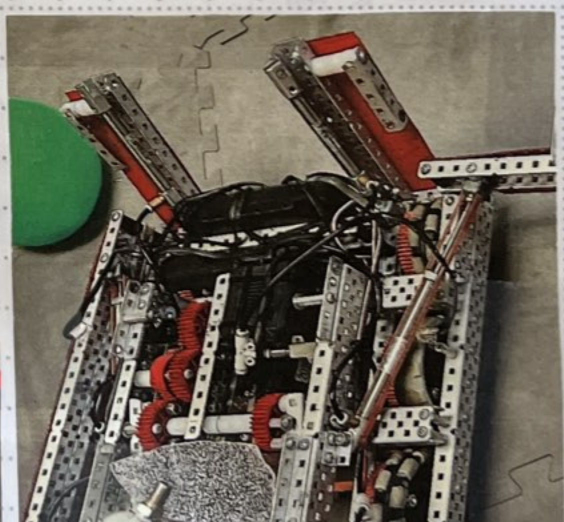
Miscellaneous:
- 2inch by 5inch Lexan Sheet

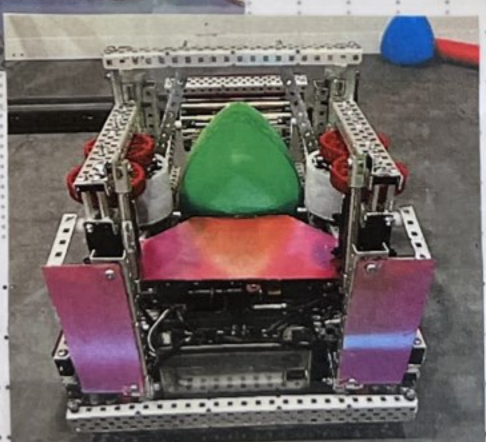
Finished Piston Guard Images:











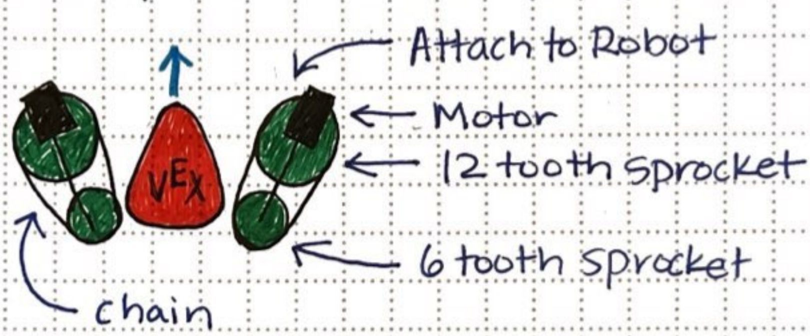
project: Finished Piston Guard Images designed by: Vanessa Perkins witnessed by: Harris Perkins
date: 7/21/2023

Brainstorming Intakes Designs:

Goals for today → Brainstorm atleast 3 intake design options
We can choose between
→ Listen and respect everyone's ideas

Design Option One: Horizontal Intake

Specifics:
- 2-5.5 watt motors/half motors
- chain
- 6:12 sprocket ratio



Above is a sketch of horizontal intakes. This design requires two (2) half motors since they can not be connected together. Attached to the 6 tooth sprocket axel will be either:

- chain flaps
- flex wheels
- rubber banded wheels

This robot has a long reach since the intakes are on the ground and can reach underneath the goals to score or descore.

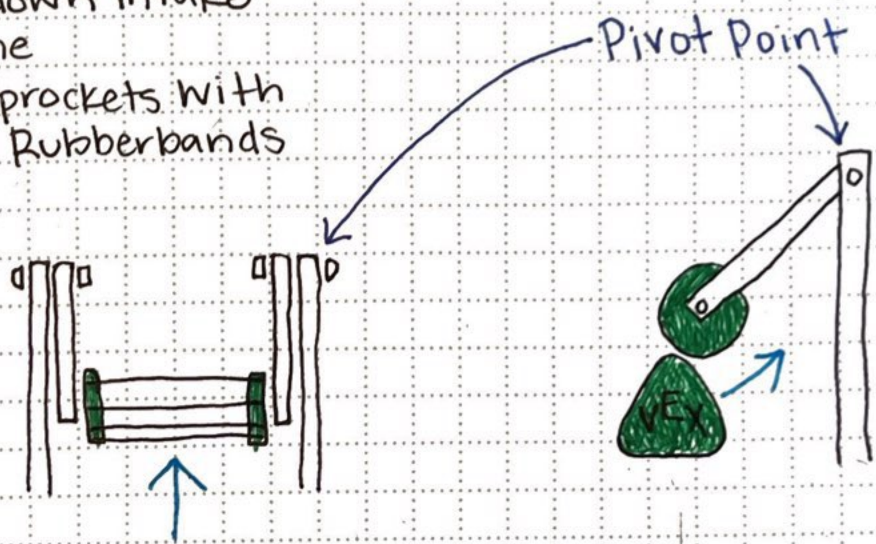
The con to this design is that chains are unreliable since they snap open easily when bumped.

project Brainstorming Intakes Designs designed by: Vanessa Perkins witnessed by: Owen taylor date: 7/22/2023

(Continued)

Design Option Two: Vertical Intake Variation 1

Specifics:
- 1 motor
- Angled down intake
- Combine
  - ↳ Sprockets with Rubberbands

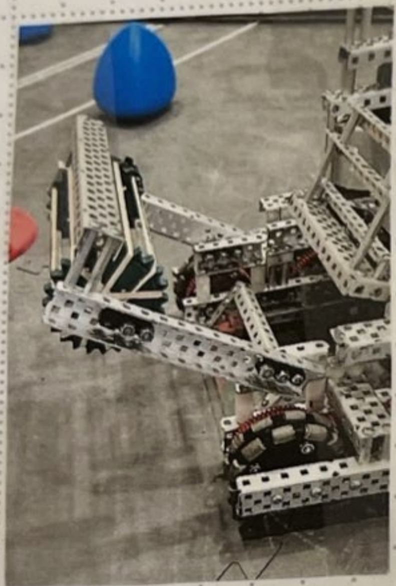
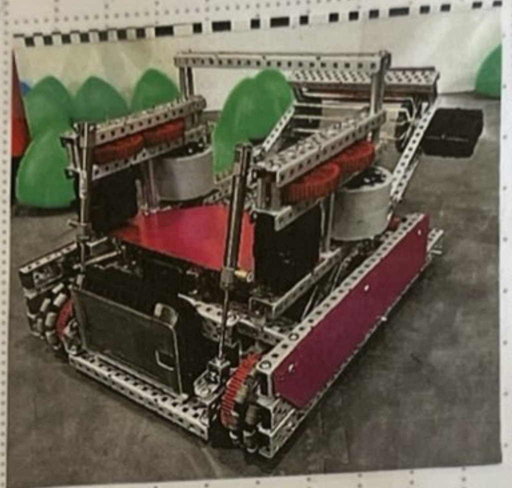


Above are sketches of this design from the front and side. The intake angles downward with gravity to create tension on the triball to help pick it up. The top is attached at a pivot point to allow the intake room to move up and down as the triball enters and exits the robot.

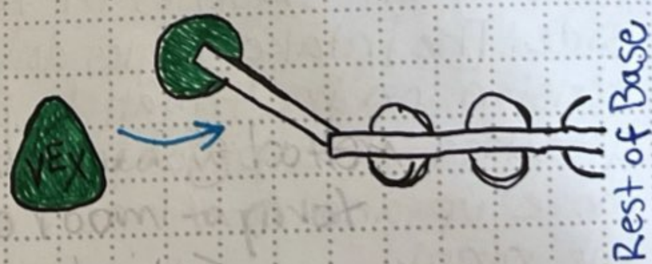
The sprocket size and rubberband amount will depend on the placement of the intake and the grip required.

(Continued)

Design Option Three: Vertical Intake Variation 2


Specifics:
- 1 motor
  - ↳ could do 11 watts or 1 half motor (5.5 watt) to save motors
- Angled up intake
- combine
  - ↳ Sprockets with rubberbands



Above are prototype photos of this intake variation as well as a simple sketch. The build and attachment of this intake is easier than option two, but does not allow room for pivoting and creating the correct tension needed to pick up triballs.

project: Brainstorming Intakes Designs (Continued) designed by: Vanessa Perkins witnessed by: Eli Fruitt date: 7/22/2023

Iteration 2 | Programming | Building

Select Approach:

Goals for today → Choose an intake design that best meets the requirements of the game so we can begin building

We created a decision matrix of what aspects mattered the most on our base. Below is the decision matrix with the winning intake.

Intake Ideas	Efficiency	Versatility	Size	Compression	Total
Horizontal	2	5	1	1	9
Vertical 1	4	5	3	4	16
Vertical 2	3	2	3	2	10

* Efficiency: how efficient the intake is at quickly moving triballs with little error.
Versatility: how well it can pick up triballs, hold them, score them into the goal, and put them in the flywheel
Size: does not take up a lot of space on robot or outside of the 18 inches.
Compression: how much grip and tension is on the triballs to help it be intaked and carried around.

Selected Design: Vertical Intake Variation 1

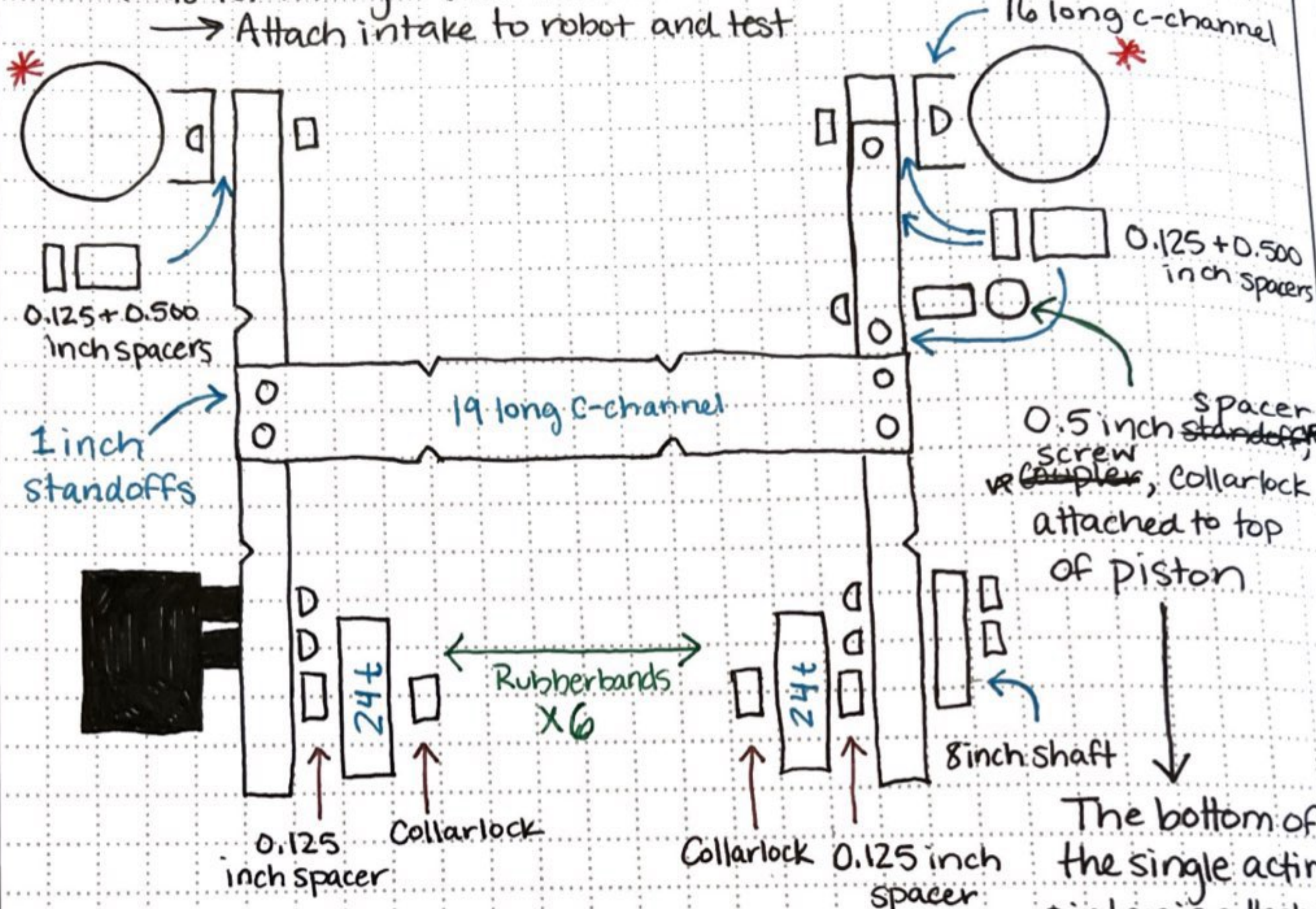
Explanation: We chose the Vertical Intake Design with the downward angle. The downward angle allows the smoothest intaking of triballs into the robot. This variation takes up the least amount of space. Variation 1 is better than 2 because the angle and ability to pivot help create a better transition of triballs into the robot.

* Breakdown of each criteria aspect we looked at. A rank of 0 (zero) means bad, and a rank of 5 (five) is good.

project: Select Approach designed by: Vanessa Perkins witnessed by: Harris Perkins
date: 7/22/2023

Begin Building Intake:

Goals for today → Build intake
→ Attach intake to robot and test



The bottom of the single acting piston is attached to the 3rd hole back of the inside c-channel of the base using a 0.500 inch screw.

*- The circles represent the two(2) air reservoirs that we have on our robot. They are attached using two(2) zipties each.

Above is a diagram of the intakes. The intakes are built off of the two(2) sixteen(16) long c-channels at the top that are attached perpendicular to the base. The two(2) c-channels are attached using two(2) half inch screws and nylocks per side to the end of the flywheel c-channels.

The 24 teeth gears are sprockets with rubberbands to make a combine.

project Begin Building Intake designed by: Vanessa Perkins witnessed by: [illegible]
date: 7/23/2023

Iteration 2 | Programming | Building

Complete Intake Parts List:

Metal:
- 5 long aluminum c-channel
- 2-15 long aluminum c-channels
- 2-16 long aluminum c-channels
- 19 long aluminum c-channel
- 8 inch shaft

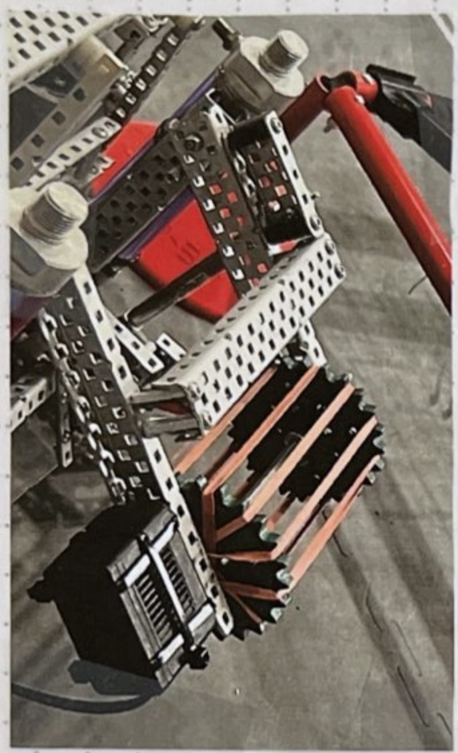
Screws/nuts:
- 12-0.500 inch screws
- 4-1.000 inch screws
- 6 Nylock nuts

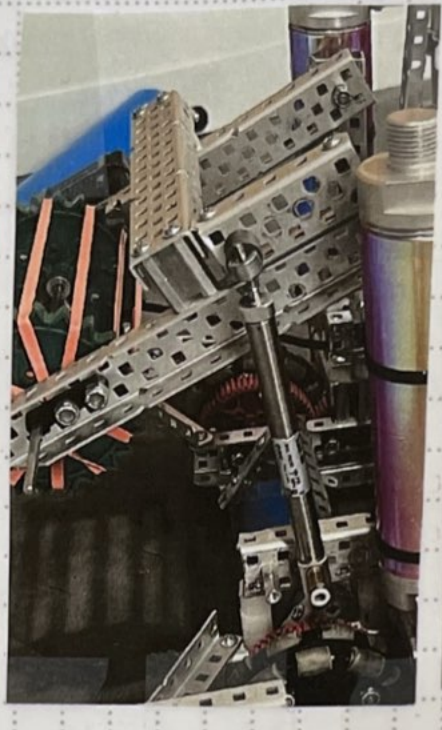
Spacing:
- 5-0.125 inch spacers
- 4-0.500 inch spacers
- 3 collarlocks

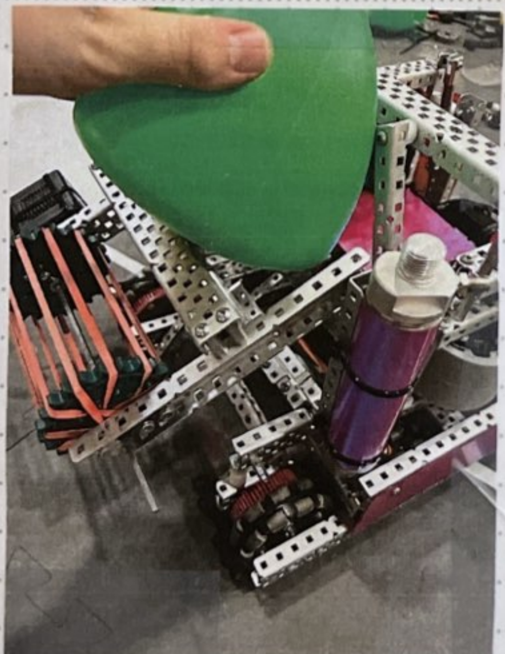
Miscellaneous:
- 2 air reserviors
- 2 bearing flats
- 4-1.000 inch standoffs
- 4 zipties
- 2-24 tooth sprockets
- 6 rubberbands
- 1 motor
- 1-600 rpm low torque, high speed, blue cartridge

project: Complete Intake Parts List
designed by: Vanessa Perkins
witnessed by: Owen taylor
date: 7/23/2023

Finished Intake Images:







VP

project Finished Intake Images designed by: Vanessa Perkins witnessed by: Harris Perkins

date: 7/23/2023

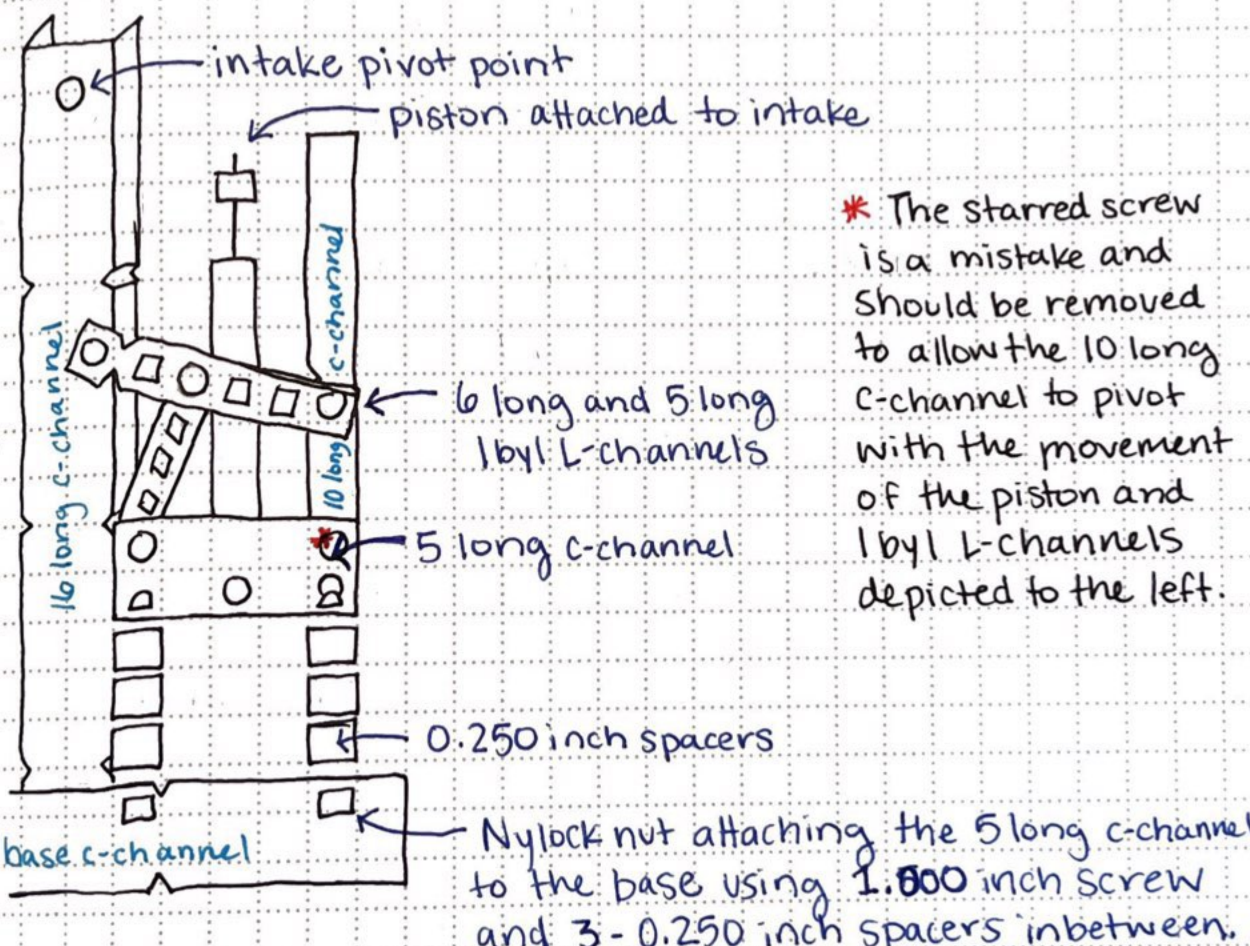
Date: 7/23

Right Side Front Piston Guard:

Goals for today ⟶ Add guard to intake piston

Because of the reasoning on pages 58-59 we are going to go ahead and build a guard around the front intake piston to prepare for them being hit or bent while driving.

Below is the diagram for the rightside guard from the inside of the base view.



* The starred screw is a mistake and should be removed to allow the 10 long c-channel to pivot with the movement of the piston and 1by1 L-channels depicted to the left.

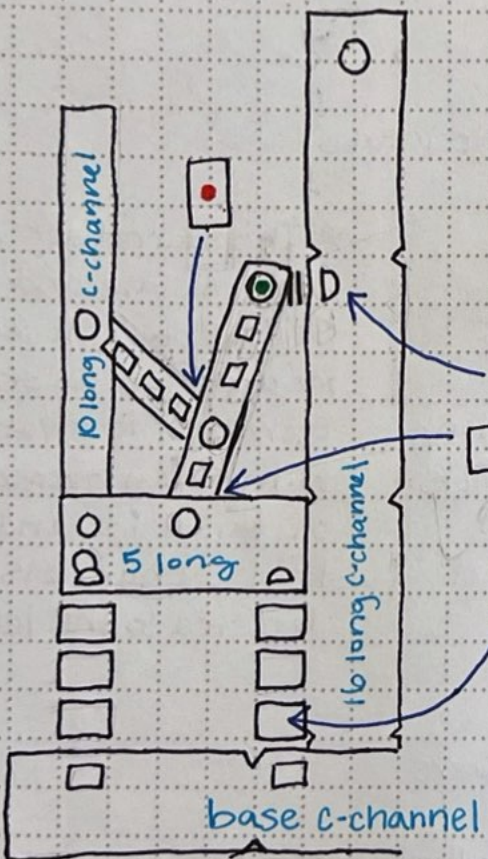
project: Right Side Front Piston Guard
designed by: Vanessa Perkins
witnessed by: Eli Fruits
date: 7/23/2023

Left Side Front Piston Guard:

Date: 7/23

Goals for today → Add front guard to the left side of the robot

Below is the diagram of the left side guard. Despite being no piston on the left the guard on the left side is to create balance and provide some protection to the robot while driving. This guard does not move in or out as it does not have to pivot with the piston and intake as the right side does.



● - 0.500 inch spacer in between the two(2) 1by1 L-channels connected with 0.750 inch screw and a nylock nut.

0.250 inch screw into collar shaft

0.375 inch spacer between 6 long 1by1 L-channel and 5 long c-channel

0.250 inch spacers that go in between the 5 long c-channel and base c-channel connected with a 1.000 inch screw and nylock nut.

● - At the end of the six(6) long 1by1 L-channel, a collar shaft is attached with a half inch screw and a nylock nut. A 0.250 inch screw attaches the collar shaft to the 16 long c-channel with two(2) teflon washers in between.

Complete Piston Guards Parts List:

Metal:
- 2 - 10 long aluminum c-channels
- 2 - 6 long aluminum 1 by 1 L-channels
- 2 - 5 long aluminum 1 by 1 L-channels
- 2 - 5 long aluminum c-channels

Spacers:
- 2 Teflon washers
- 12 - 0.250 inch spacers
- 1 - 0.375 inch spacer
- 1 - 0.500 inch spacer

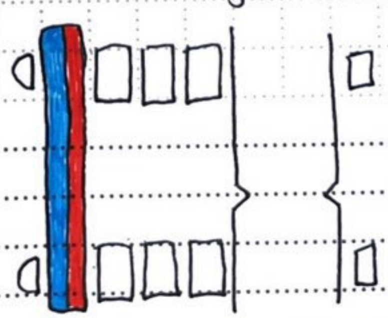
Screws/Nuts:
- 1 - 0.250 inch screw
- 12 - 0.500 inch screws
- 4 - 1.000 inch screws
- 15 Nylock nuts

Miscellaneous:
- 1 - collar shaft

* Note *

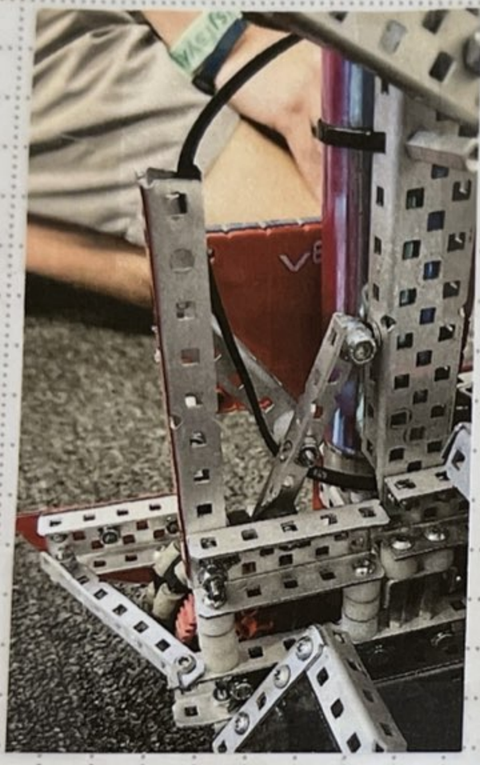
We added the license plates today to the front sides of the robot on the 10 long c-channels of the piston guards. The screws went through 3rd and 7th holes down from the top. Below are the parts required and a diagram.

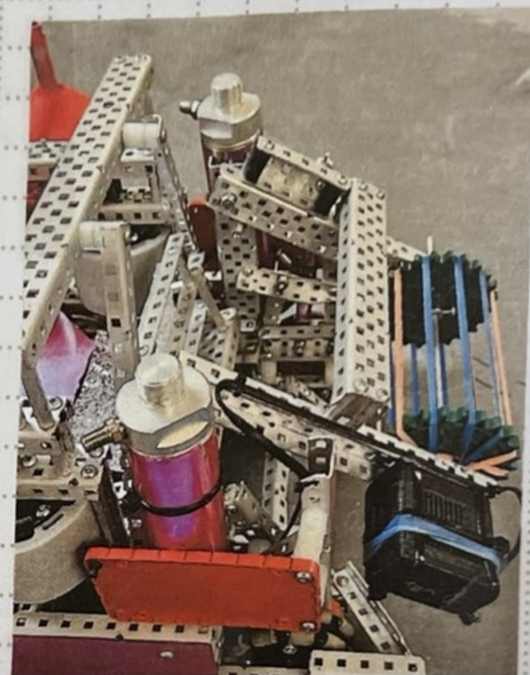
License Plate Parts:
- 12 - 0.250 inch spacers
- 4 - 1.500 inch screws
- 4 Nylock nuts
- 2 Red and 2 Blue License Plates

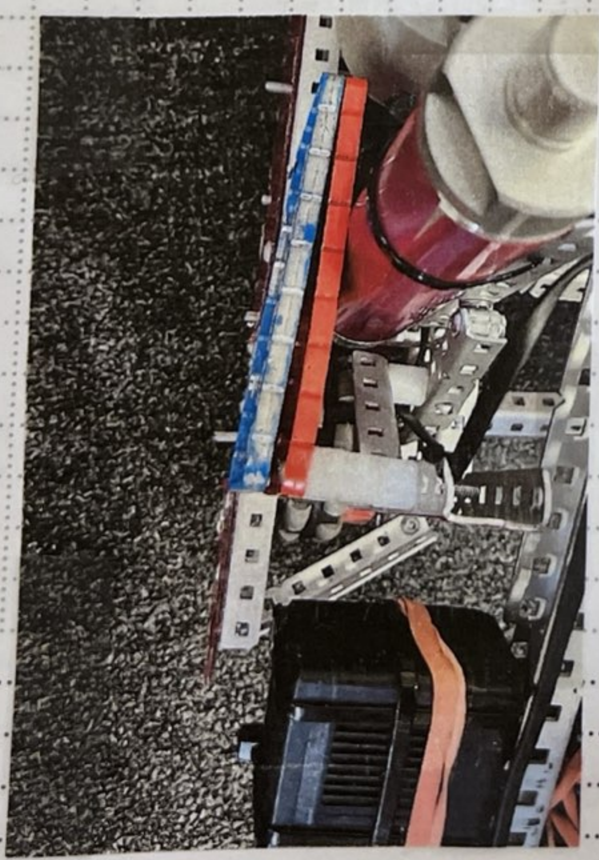


project: Complete Piston Guards Parts List designed by: Vanessa Perkins witnessed by: Harris Perkins date: 7/23/2023

Finished Piston Guards Images:







VP

project Finished Piston Guards Images designed by: Vanessa Perkins witnessed by: Harris Perkins

date: 7/24/2023

Triball Alignment: Points of Contact:

Goals for today → Revisit testing of flywheel

To improve the performance of the flywheel as tested and recorded on page 41, we will be identifying the problem of alignment.

Through hand feed testing, we found that two (2) points of contact on the triball provides the optimal launch. Two (2) points of contact equals more control of the triball as it leaves the flywheel as opposed to a crooked triball with less consistent points of contact.

The challenge of this is to be able to have a perfectly aligned triball every time using the intakes and indexor, not our hands.

The solution we decided on was some sort of aligner that would guide the triball to rotate into the right position in between the indexor and flywheel mechanisms.

* The image above shows the prime positioning of the triball between the two flywheels.

Brainstorming Aligner Designs:

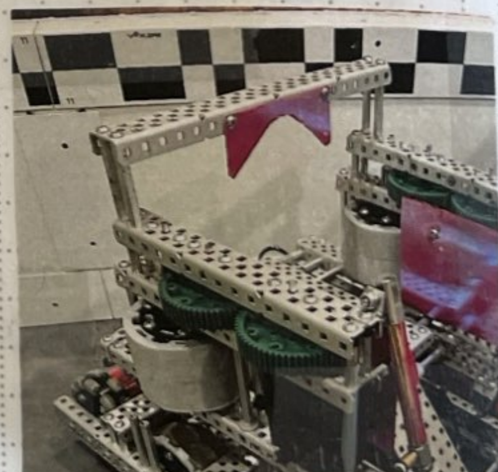
Goals for today → Come up with an aligner design
→ build, test, and adjust design as needed.

Problem/Challenge
Triballs are inconsistently placed into flywheel.
(More in depth analysis on page 73).

Brainstorm
① Lexan aligner
② Metal aligner

Select Option
①
Cut out a Lexan sheet to shape of triball and attach infront of flywheel

Build/Implement Solution
- gather 3.75 inch by 2.5 inch lexan
- cut out "tip of triball" shape
- screw on to robot



Test
The triball would still hit the Lexan aligner crooked and ~~get~~ messed and hung up. Not sturdy enough.

Redesign
②
A metal aligner will provide more support and stability and consistency since it will be a larger, more in depth aligner than a small piece of Lexan

Above is an image of our first aligner design attached to the top of the robot.

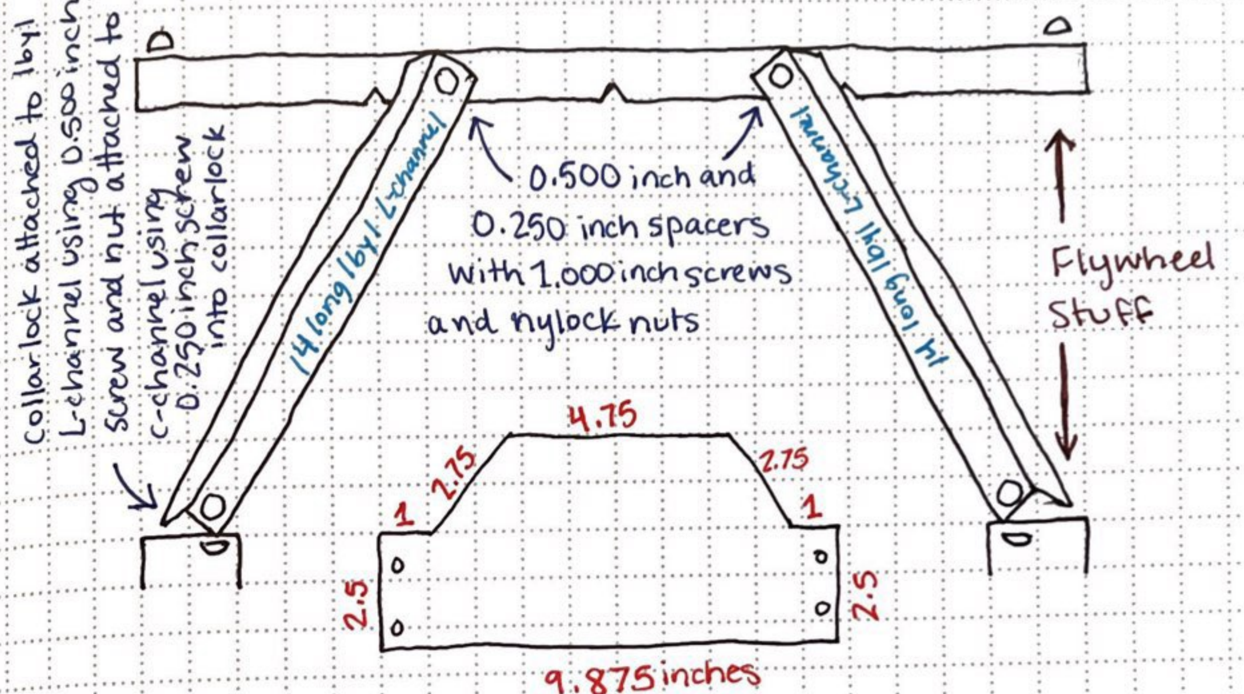
Build/Implement Solution
Go to page 75 for the build of the metal aligner.

project: Brainstorming Aligner Designs designed by: Vanessa Perkins witnessed by: Owen taylor
date: 7/24/2023

Date: 7/24

Build Metal Aligner:

Goals for today → build structure and attach to flywheel to align triballs



Above is the diagram for the metal aligner that attached to the front of the flywheel. We also diagramed the Lexan sheet that goes inbetween the flywheels and on top of the indexor gearing to prevent it from causing troubles. Above are the measurements for that Lexan in inches.

We tested the aligner and it worked very well at orienting the triball in just the way we needed it.

The Lexan sheet attached with four (4) half inch screws and nylock nuts on the bottom flywheel C-channels that the top of the aligner is also attached to.

project: Build Metal Aligner designed by: Vanessa Perkins witnessed by: Eli Fruits
date: 7/24/2023

Complete Aligner Parts List and Images:

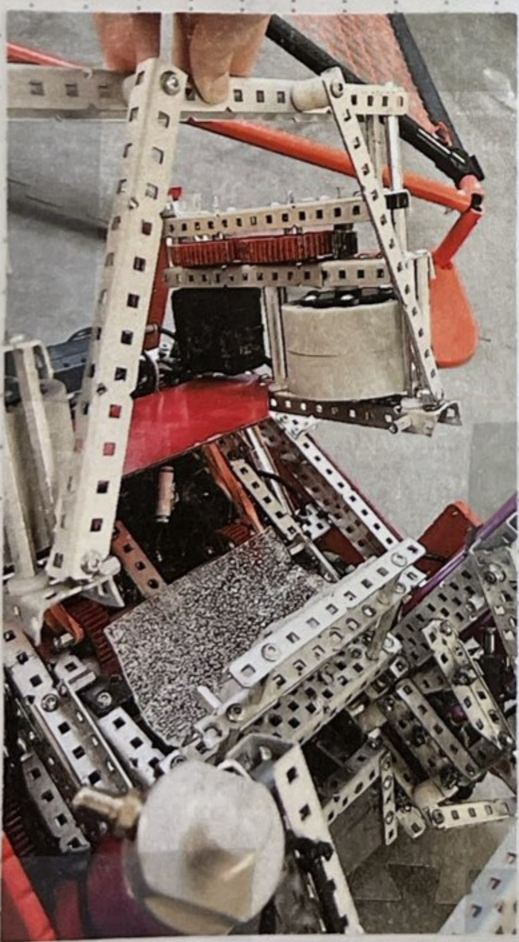
Metal:
- 2-14 long aluminum 1by1 L-channels

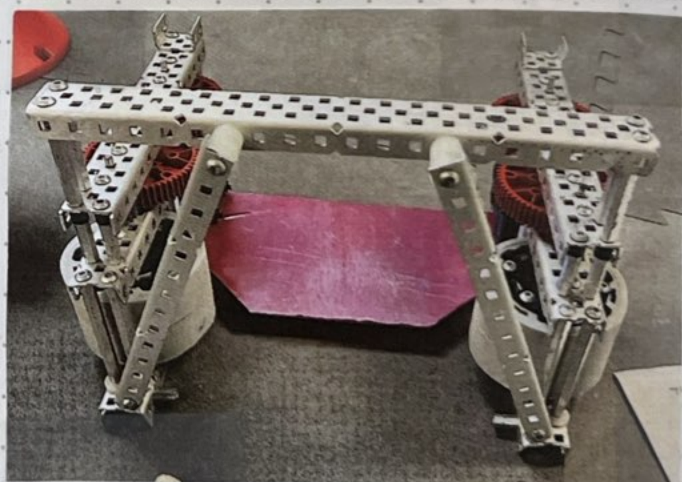
Spacers:
- 2- 0.250 inch spacers
- 2- 0.500 inch spacers
- 2- collar locks

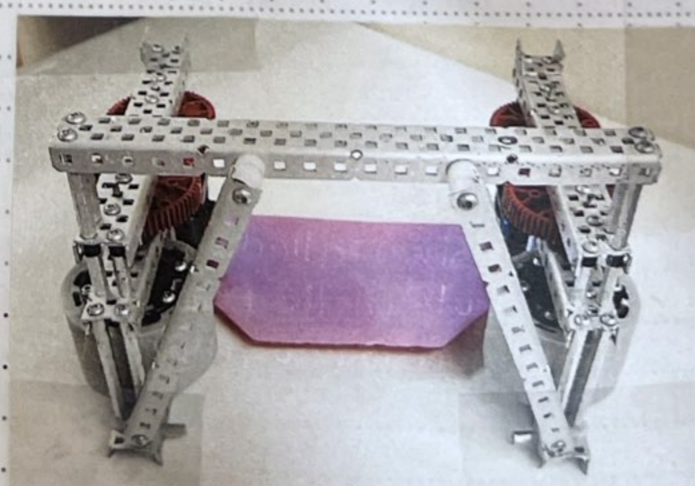
Lexan Sheet Parts:
- Lexan sheet with dimensions on page 75
- 4- 0.500 inch screws
- 4- Nylock nuts

Screws/Nuts:
- 2- 1.000 inch screws
- 2- 0.500 inch screws
- 2- 0.250 inch screws

- 4 Nylock Nuts







project: Complete Aligner Parts List and Images designed by: Vanessa Perkins witnessed by: Owen taylor

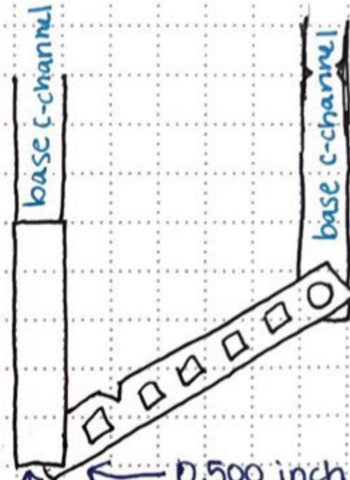
date: 7/24/2023

Date: 7/24

Build Funnels:

Goals for today → build funnels that are strong and will direct triballs into the intake instead of the wheels

Parts List:
- 2-7 long aluminum 1by1 L-channels
- 2-5 long aluminum C-channels
- 8-0.500 inch screws
- 8 Nylocks nuts

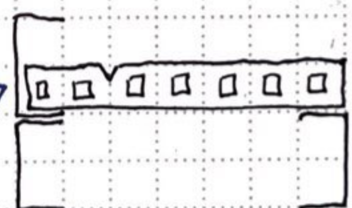


→ Side View
7 long 1by1 L-channel

← 0.500 inch screw and nut

5 long c-channel attached to base c-channel through last two holes using two(2) half inch screws and nuts.

Above is the part list for two(2) funnels on the front of the robot's base. To the right is a diagram of the left side funnel from an above view and in front view.



5 long c-channel that extends out past the base c-channel.

base c-channels

The right side funnel is mirrored and repeated exactly the same way as the left side. To the right is an above image of the right side funnel completely built and attached to the robot.

project Build Funnels designed by: Vanessa Perkins witnessed by: Harris Perkins
date: 7/24/2023

Date: 7/24

Identify Problem and Brainstorm:

Goals for today ⟶ record ideas for improving the robot's ability to drive over the barrier

Identify the Problem:

Since adding the flywheel and all other subsystems and components, the robot has become much heavier.

When trying to drive the base over the barrier now, the weight of the robot keeps it from being able to complete this task. Without the ability to climb the barrier, during matches our team will lose more time driving around them.

Brainstorm:

① Remove weight from the robot by using zipties and plastic screws and plastic nuts.

② Make long Lexan sleds that screw into the side of the base and allow the robot to drive up over the barrier in both directions.

③ Make two (2) Lexan sleds that just go on the front of the base to help drive and lift the robot over the barrier.

Select Option:

② We decided that a full length of the robot sled made out of Lexan would be the most efficient solution. On the next page is our build and implementation of our selected solution.

We decided that idea ① would require too much in depth building and might not create sufficient results.

project: Identify Problem and Brainstorm
designed by: Vanessa Perkins
witnessed by: Eli Fritts
date: 7/24/2023

Date: 7/24

Build Sleds:

Goals for today → Create a shape template
→ Cut out Lexan sled shapes and attach to robot's base
→ Test over the barrier

Below is an image of the CAD design of this sled that would help the robot climb the barrier from the front or back. The dimensions were approximately 16 inches long and 1 inch high in the middle.

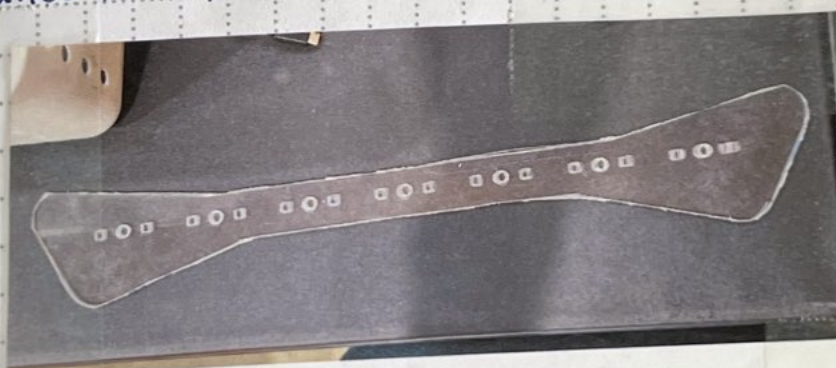
To keep precise cuts and measurements we tried to use a Cricut to cut the Lexan. However the Lexan was too thick and the screw and axel holes did not come out.

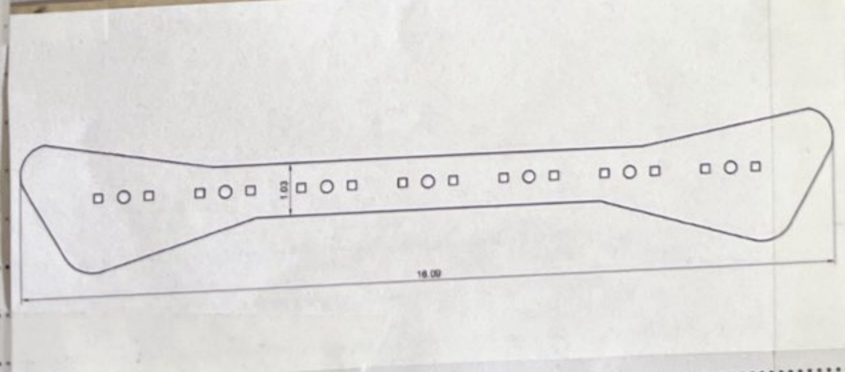
Below is the picture of our failed attempt.

We will be redesigns to the simpler sled option of option 3 located on page 78.

Failed Attempt

CAD





project: Build Sleds

designed by: Vanessa Perkins

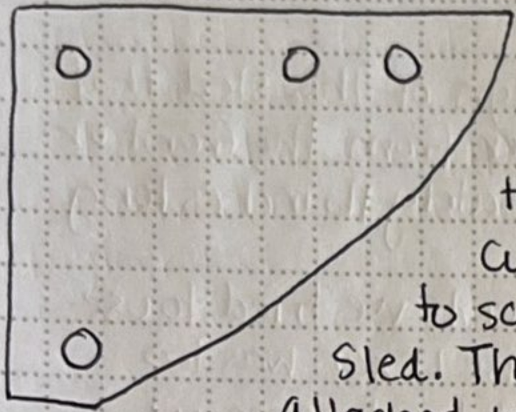
witnessed by: Owen Taylor

date: 7/24/2023

Date: 7/24

Redesign Sleds:

Goals for today ⟶ switch to the simpler sled design noted on page 78 under option 3

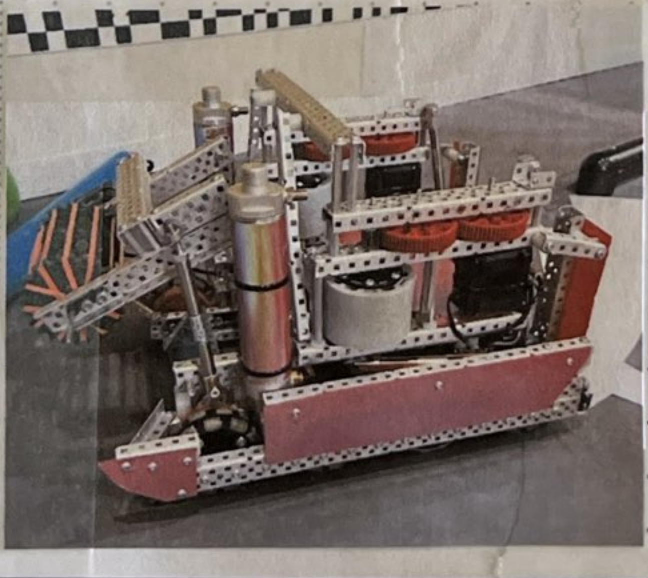


We could not find an entirely successful way to cut out the first sleds design option so we switched to a simpler design that we could cut out by hand. To the left is a to scale template of the left side Lexan sled. The right side is just mirrored. They were attached with four (4) half inch screws each and four (4) Nylock nuts each to the front of the robot on the outside c-channels.

Below are the images of the complete sleds attached to both sides of the front of the robot as well as the parts list. The sleds were tested and work well to allow us to climb the barrier.

Parts List:

- 2 Lexan sleds
- 8-0.500 inch screws
- 8 Nylock nuts



project Redesign Sleds designed by: Vanessa Perkins witnessed by: Harris Perkins
date: 7/24/2023

Date: 7/24

Brainstorming Hang Mechanisms:

Goals for today → Brainstorm a hanging mechanism within the constraints of the game and our resources

Identify the Problem:

We need a mechanism on the robot that can help us hang from the elevation bar. The biggest challenge will be to construct something that does not require a motor since we are currently at our max of eight (8).

Our goal is to atleast leave the ground.

Brainstorming:

① Switch the intake to a 5.5 watt motor to create an extra 5.5 motor to be used for a hanging mechanism such as claw.

② Add metal hooks to the top of the flywheel that will hang the robot from the horizontal elevation bar.

③ Add a stabilizing C-channel to the side of the robot so we can drive over the barrier and balance off of the ground against the elevation bar.

Select Option:

③ With our decreasing time before our scrimmage, option 3 will be the simpliest option for the moment. It meets our goal of leaving the ground and does not use too many resources.

The first two (2) ideas are not very easy to fine tune and build in our limited time.

The next page has our build and implementation of the chosen solution.

Iteration 2 | Programming | Building

project: Brainstorming Hang Mechanisms
designed by: Vanessa Perkins
witnessed by: Harris Perkins
date: 7/24/2023

Build Hang Mechanism:

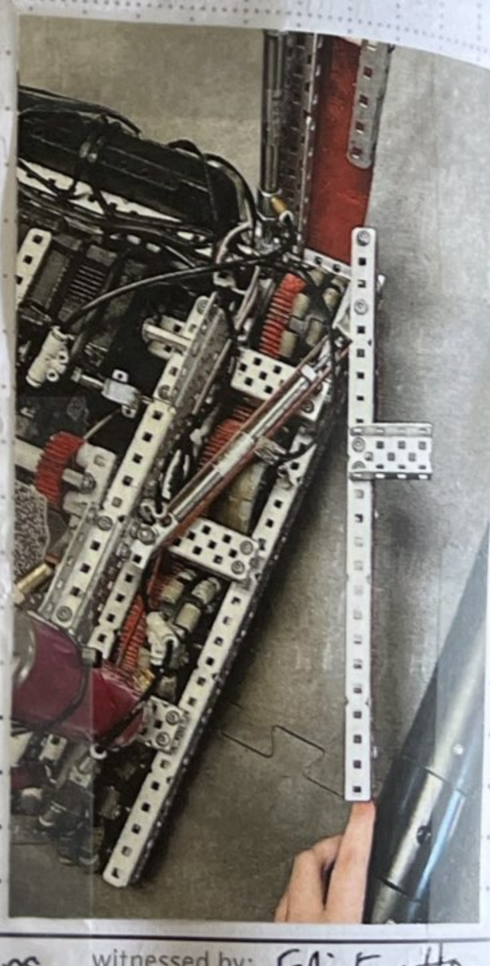
Date: 7/24

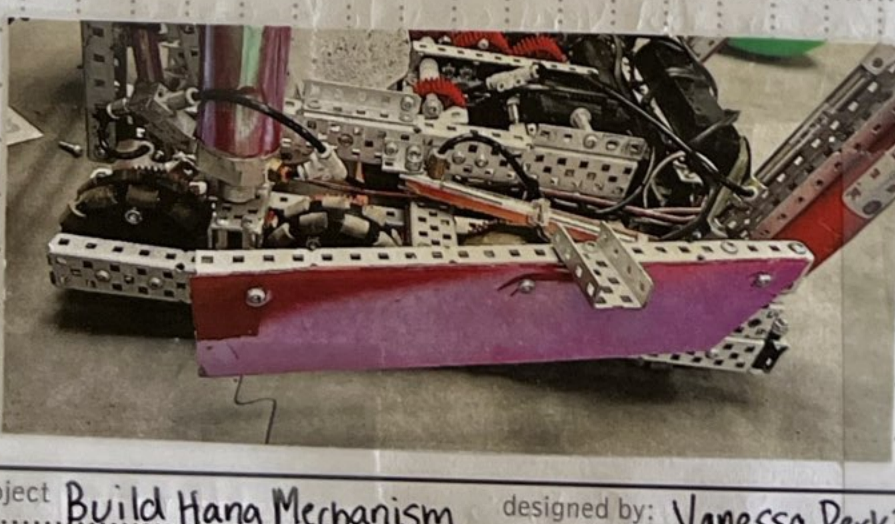
Goals for today ⟶ Build and Test the hang mechanism

Below are the images of the stabilizer to help us hang our robot on the edge of the barrier leaning against the elevation bar.

We used a four (4) long aluminum c-channel and attached it using two (2) 0.500 inch screws and two (2) Nylock nuts. The c-channel went through the tenth (10) and eleventh (11) holes out on the right side wing.

VP





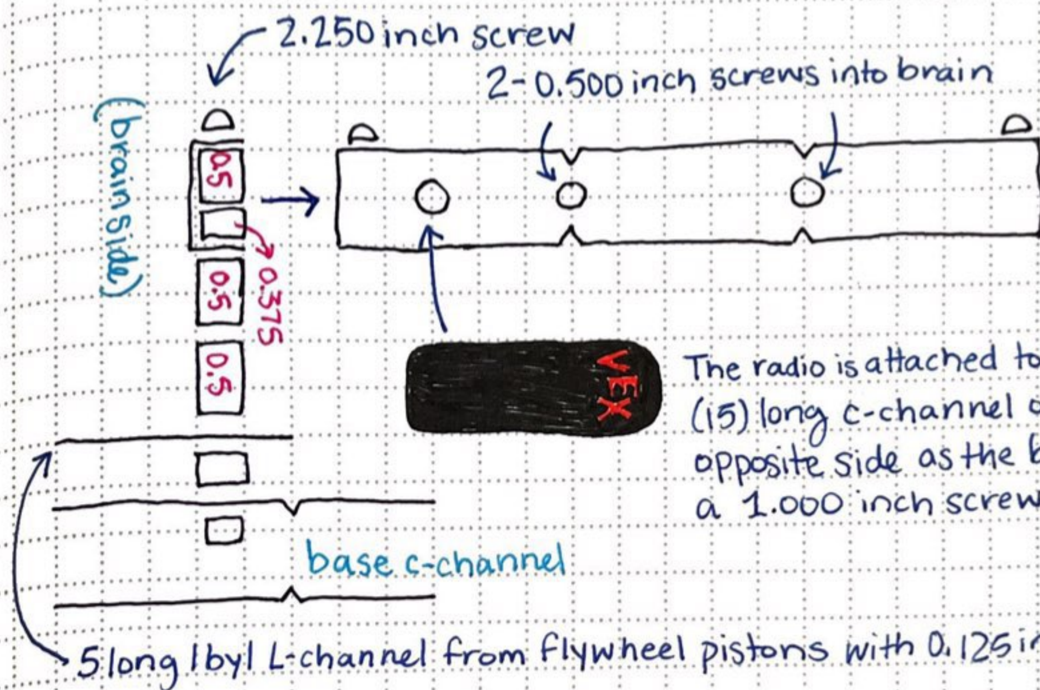
project: Build Hang Mechanism designed by: Vanessa Perkins witnessed by: Eli F[illegible]

date: 7/24/2023

Date: 7/25

Attach Brain to Robot:

Goals for today → Get the brain attached to the robot
→ Plug in all wires and motors and the battery



The radio is attached to the fifteen (15) long c-channel on the opposite side as the brain using a 1.000 inch screw.

Above is the diagram for how to attach the c-channel to the back of the robot's base for the brain to attach to. The diagram is mirrored for the right side. Above is also a front view of how the radio and brain attach to the c-channel. To the right is the parts list.

We also placed the battery on the bottom of the robot between the wheel motors using a loose ziptie as its holder.

Parts List:

- 15 long aluminum c-channel
- 2-0.500 inch screws
- 1-1.000 inch screw
- 2-2.250 inch screws
- 2 Nylock Nuts
- 6-0.500 inch spacers
- 2-0.375 inch spacers
- 1 radio transmitter
- 1 brain
- 1 large ziptie

project Attach Brain to Robot designed by: Vanessa Perkins witnessed by: Harris Perkins
date: 7/25/2023

Iteration 2 | Programming | Building

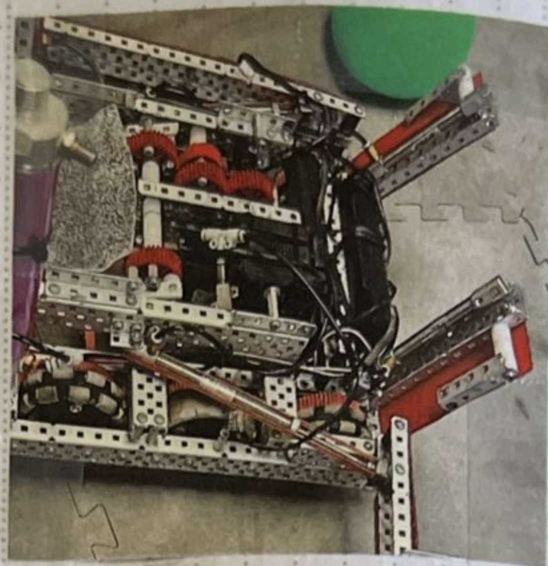
Date: 7/25

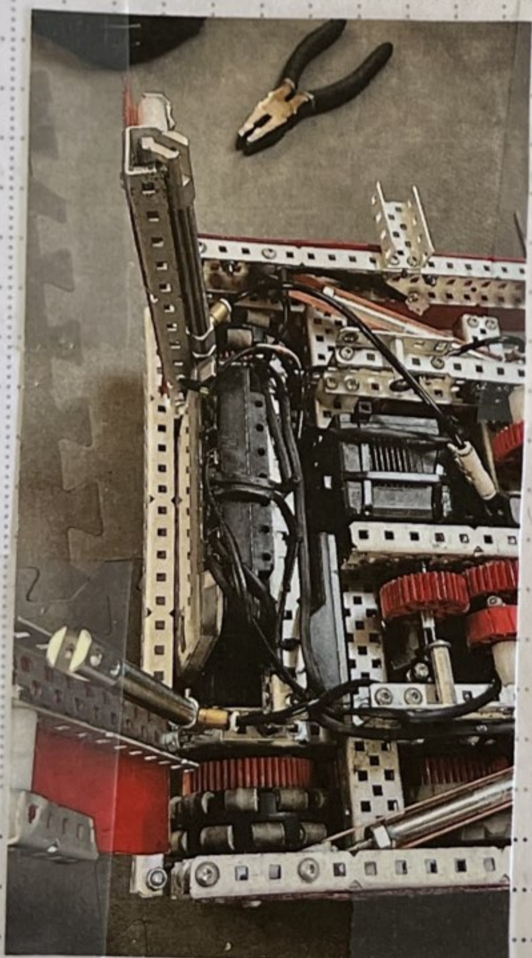
Brain Images:

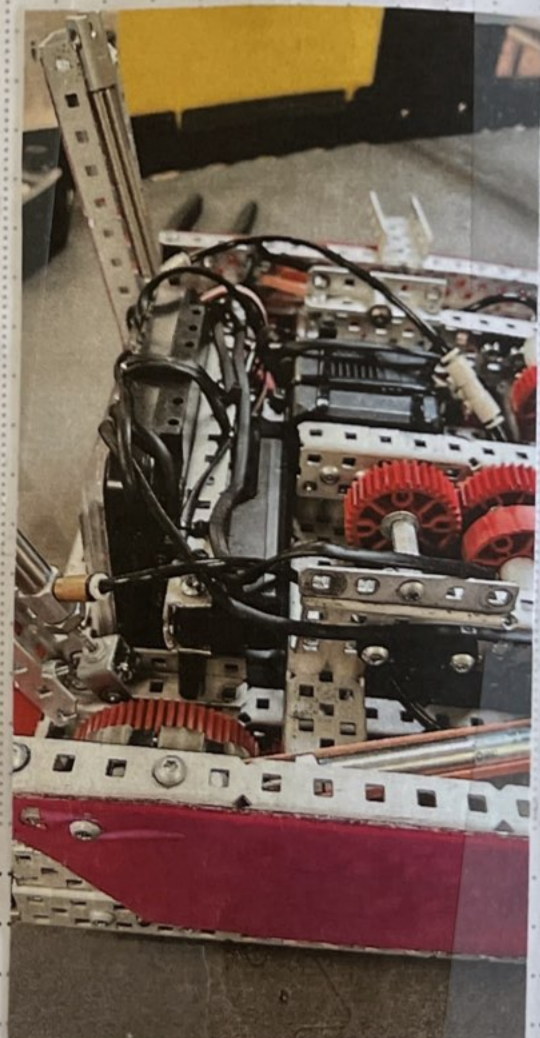
To the right and below are images of the added brain with the flywheel temporarily removed for better viewing and access.

The next step is to finish programming everything.

VP





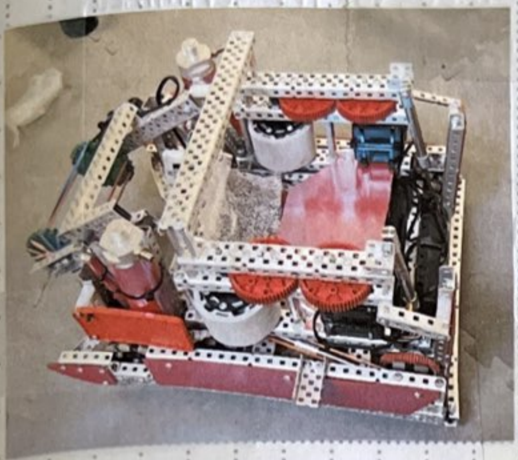


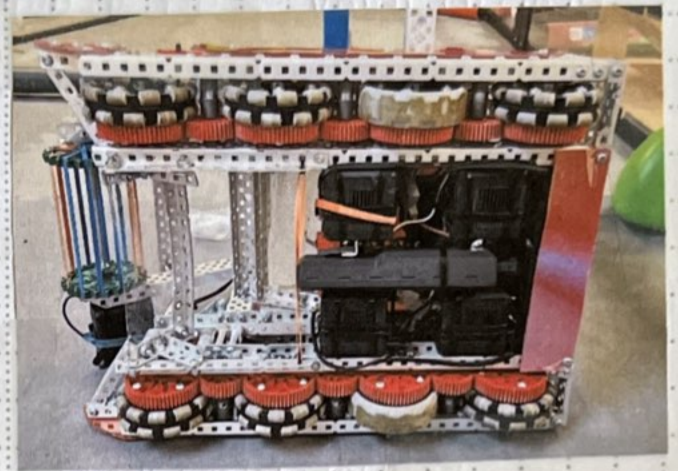
project Brain Images designed by: Vanessa Perkins witnessed by: Harris Perkins

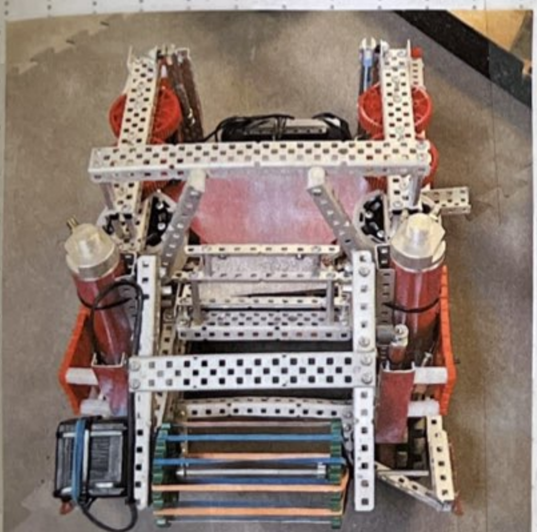
date: 7/25/2023

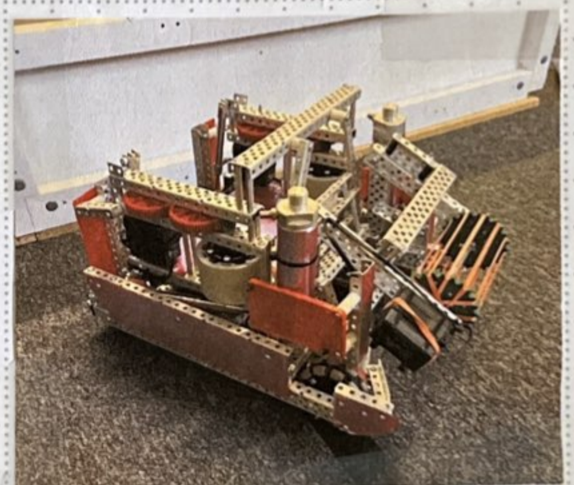
Date: 7/25

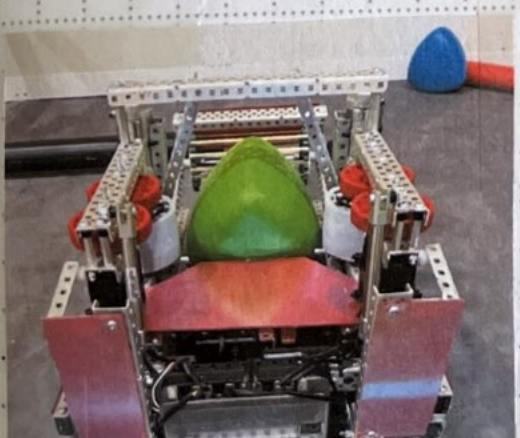
Complete Robot Pictures:

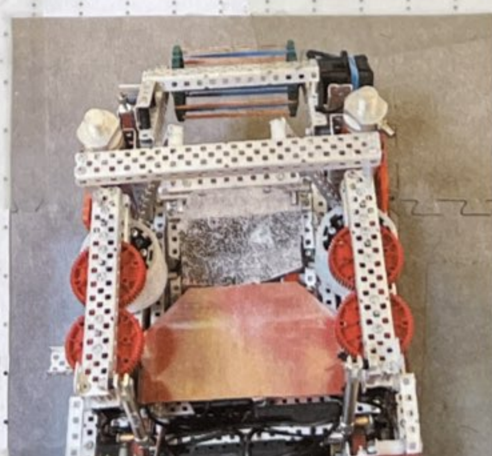












project: Complete Robot Pictures designed by: Vanessa Perkins witnessed by: Harris Perkins
date: 7/25/2023

Auton. Program Testing:

Date: 7/26

Goals for today → Get an autonomous for the scrimmage
→ Get all controls programmed

1. Set up controls.
2. Set goal for autonomous → Win Point is not essential
   → Score atleast 1 triball
   → Move robot to be ready for matchloads
3. Define a starting position: Middle tile of blue defensive side on edge of tile facing the Red Starting tiles
4. Begin programming Commands. Below is a chart of our ~~impuses~~ inputs and the results.

Input	Output
forward 100	good
turn left 90	not working
turn right 90	not working
change starting position to face backwards & start over for easier programming	
reverse 100	good
turn to right 90	good
indexor reverse 40%	no result
change: indexor forward 40%	no result
change: indexor forward 100%	worked but agressive
change: indexor forward 80% plus indexor stop	
flywheel 80	too agressive
change: flywheel 60	too weak
change: flywheels 70	too much
reverse 100	good, got closer to barrier
change: flywheel 60	too weak
change: reverse 50	medium distance so we can increase flywheel speed
chnage: flywheel 65	bit too much so don't drive as close to goal
change: reverse 46	closest to goal

The program has the robot reversing, turning 90 degrees, and picking up a triball from the center of the field, driving towards the barrier then launching the triball towards the goal for a score of 2 to 5 points depending on whether it makes it in or not each time.

project Auton. Program Testing designed by: Vanessa Perkins witnessed by: Eli Foutts
date: 7/26/2023

```
#include "vex.h"

// ---- START VEXCODE CONFIGURED DEVICES ----
// Robot Configuration:
// [Name] [Type] [Port(s)]
// Controller1 controller
// Drive_Front_Left motor 12
// Drive_Back_Left motor 1
// Drive_Front_Right motor 9
// Drive_Back_Right motor 5
// Intake motor 10
// Flywheels motor_group 6, 7
// Indexer motor 16
// Wings digital_out A
// Angler digital_out B
// Inertial3 inertial 3
// LimitSwitchC limit C
// IntakeUp digital_out D
// ---- END VEXCODE CONFIGURED DEVICES ----

using namespace vex;
competition Competition;

/*---------------------------------------------------------------------------*/
/* VEXcode Config */
/* */
/* Before you do anything else, start by configuring your motors and */
/* sensors using the V5 port icon in the top right of the screen. Doing */
/* so will update robot-config.cpp and robot-config.h automatically, so */
/* you don't have to. Ensure that your motors are reversed properly. For */
/* the drive, spinning all motors forward should drive the robot forward. */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* JAR-Template Config */
/* */
/* Where all the magic happens. Follow the instructions below to input */
/* all the physical constants and values for your robot. You should */
/* already have configured your robot manually with the sidebar configurer. */
/*---------------------------------------------------------------------------*/

Drive chassis(

//Specify your drive setup below. There are seven options:
//ZERO_TRACKER_NO_ODOM, ZERO_TRACKER_ODOM, TANK_ONE_ENCODER, TANK_ONE_ROTATION,
TANK_TWO_ENCODER, TANK_TWO_ROTATION, HOLONOMIC_TWO_ENCODER, and HOLONOMIC_TWO_ROTATION
//For example, if you are not using odometry, put ZERO_TRACKER_NO_ODOM below:
```

Iteration 2 | Programming | Building

project: Program   designed by: Harris Perkins   witnessed by: Vanessa Perkins
date: 7/26/2023

```
ZERO_TRACKER_NO_ODOM,

//Add the names of your Drive motors into the motor groups below, separated by commas,
i.e. motor_group(Motor1,Motor2,Motor3).
//You will input whatever motor names you chose when you configured your robot using
the sidebar configurer, they don't have to be "Motor1" and "Motor2".

//Left Motors:
motor_group(Drive_Front_Left,Drive_Back_Left),

//Right Motors:
motor_group(Drive_Front_Right,Drive_Back_Right),

//Specify the PORT NUMBER of your inertial sensor, in PORT format (i.e. "PORT1", not
simply "1"):
PORT3,

//Input your wheel diameter. (4" omnis are actually closer to 4.125"):
3.25,

//External ratio, must be in decimal, in the format of input teeth/output teeth.
//If your motor has an 84-tooth gear and your wheel has a 60-tooth gear, this value
will be 1.4.
//If the motor drives the wheel directly, this value is 1:
1.666,

//Gyro scale, this is what your gyro reads when you spin the robot 360 degrees.
//For most cases 360 will do fine here, but this scale factor can be very helpful when
precision is necessary.
360,

/*---------------------------------------------------------------------------*/
/* PAUSE! */
/* */
/* The rest of the drive constructor is for robots using POSITION TRACKING. */
/* If you are not using position tracking, leave the rest of the values as */
/* they are. */
/*---------------------------------------------------------------------------*/

//If you are using ZERO_TRACKER_ODOM, you ONLY need to adjust the FORWARD TRACKER
CENTER DISTANCE.

//FOR HOLONOMIC DRIVES ONLY: Input your drive motors by position. This is only
necessary for holonomic drives, otherwise this section can be left alone.
//LF: //RF:
PORT1, -PORT2,
```

project: Program designed by: Harris Perkins witnessed by: Vanessa Perkins
date: 7/26/2023

```
//LB: //RB:
PORT3, -PORT4,

//If you are using position tracking, this is the Forward Tracker port (the tracker
which runs parallel to the direction of the chassis).
//If this is a rotation sensor, enter it in "PORT1" format, inputting the port below.
//If this is an encoder, enter the port as an integer. Triport A will be a "1",
Triport B will be a "2", etc.
3,

//Input the Forward Tracker diameter (reverse it to make the direction switch):
2.75,

//Input Forward Tracker center distance (a positive distance corresponds to a tracker
on the right side of the robot, negative is left.)
//For a zero tracker tank drive with odom, put the positive distance from the center
of the robot to the right side of the drive.
//This distance is in inches:
-2,

//Input the Sideways Tracker Port, following the same steps as the Forward Tracker
Port:
1,

//Sideways tracker diameter (reverse to make the direction switch):
-2.75,

//Sideways tracker center distance (positive distance is behind the center of the
robot, negative is in front):
5.5

);

int current_auton_selection = 0;
bool auto_started = false;

void pre_auton(void) {
// Initializing Robot Configuration. DO NOT REMOVE!
vexcodeInit();
default_constants();


}

void autonomous(void) {
auto_started = true;
Intake.spin(forward,100,pct);
```

Iteration 2

Building

project Program designed by: Harris Perkins witnessed by: Vanessa Perkins
date: 7/26/2023

```
Flywheels.spin(reverse,65,pct);
chassis.drive_distance(-100);
chassis.right_swing_to_angle(90);
chassis.drive_distance(-46);
Indexer.spin(forward,80,pct);
wait(1.5,sec);
Indexer.stop();
}

/*---------------------------------------------------------------------------*/
/* */
/* User Control Task */
/* */
/* This task is used to control your robot during the user control phase of */
/* a VEX Competition. */
/* */
/* You must modify the code to add your own robot specific commands here. */
/*---------------------------------------------------------------------------*/

void usercontrol(void) {
// User control code here, inside the loop
while (1) {

chassis.control_arcade();

if(Controller1.ButtonR1.pressing() && Controller1.ButtonR2.pressing())
{Indexer.spin(forward,-40,pct);
Indexer.setStopping(hold);
}

else if(Controller1.ButtonUp.pressing())
{ Flywheels.setVelocity(90,percent);
Flywheels.spin(vex::directionType::fwd, 90, velocityUnits::pct);
Angler.set(true);
}


else if(Controller1.ButtonL1.pressing())
{ Flywheels.setVelocity(-100,percent);
Flywheels.spin(vex::directionType::fwd, -100, velocityUnits::pct);
Angler.set(false);
}

else if(Controller1.ButtonL2.pressing())
{ Intake.setVelocity(-100,percent);
Intake.spin(vex::directionType::fwd, -100, velocityUnits::pct);
IntakeUp.set(true);
```

project Program designed by: Harris Perkins witnessed by: Vanessa Perkins

date: 7/26/2023

```
}

else if(Controller1.ButtonR1.pressing())
{Indexer.spin(forward,100,pct);
waitUntil(LimitSwitchC.pressing());
Indexer.stop();
Indexer.setStopping(hold);
}

else if(Controller1.ButtonR2.pressing())
{Intake.spin(vex::directionType::fwd, 100, velocityUnits::pct);
IntakeUp.set(false);
}


else if(Controller1.ButtonX.pressing())
{Wings.set(true);
}
else if(Controller1.ButtonA.pressing())
{Wings.set(false);
}

else
{
Intake.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);
Indexer.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);
}

digital_out dig1 = digital_out(Brain.ThreeWirePort.A);

// Code for the Pneumatics
}

this_thread::sleep_for(10);
// The code for driving with the Bottom Back Right motor
// This is the main execution loop for the user control program.
// Each time through the loop your program should update motor + servo
// values based on feedback from the joysticks.

// ........................................................................
// Insert user code here. This is where you use the joystick values to
// update your motors, etc.
// ........................................................................

//Replace this line with chassis.control_tank(); for tank drive
//or chassis.control_holonomic(); for holo drive.
```

Iteration 2

Building

designed by: Harris Perkins witnessed by: Owen Taylor

project Program date: 7/26/2023

```
wait(20, msec); // Sleep the task for a short amount of time to
// prevent wasted resources.
}

//
// Main will set up the competition functions and callbacks.
//
int main() {
// Set up callbacks for autonomous and driver control periods.
Competition.autonomous(autonomous);
Competition.drivercontrol(usercontrol);

// Run the pre-autonomous function.
pre_auton();

// Prevent main from exiting with an infinite loop.
while (true) {
wait(100, msec);
}
}
```

The past five (5) pages have been our program for the robot with the autonomous towards the beginning and the driver controls towards the end.

If we plan to keep the robot after the scrimmage, we will program more in depth autons and develop our own PID instead of using the JAR Template for sake of time.

project Program designed by: Harris Perkins + Vanessa Perkins witnessed by: Eli Keith date: 7/26/2023

Date: 7/29

HOPE Scrimmage:

Goals for today → Test the performance of our robot to help us decide between keeping it or rebuilding
→ Run through at least three (3) matches

Below are the details of each match we ran with team 19589A Hope Robotics as well as our analysis of the overall robot performance.

Match 1: 110-52 (lose)
- Robot jammed because of Lexan model
- Needs sturdier wings that do not bow under pressure
- Needs better way than flywheel to move triballs to offensive zone

Match 2: 108-43 (lose)
- The triball matchloads bounce backwards too much

Match 3: 58-54
- Used a blocking strategy instead of using preloads and scoring

Match 4: 116-8 (2 versus 0)

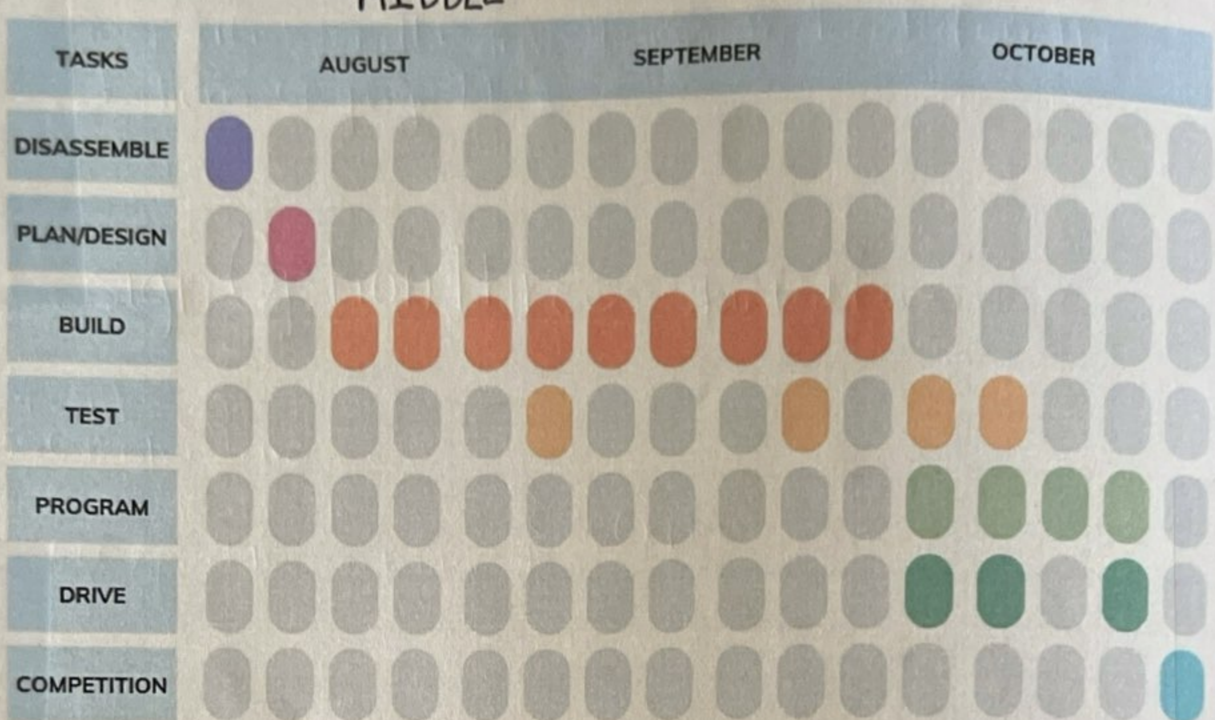
Analysis: Redesign and Rebuild robot because today was poor performance.

Date: 8/1

Timeline:

GANTT CHART

VR ~~EARLY~~ MIDDLE SEASON TIMELINE



Disassemble- Take apart robot #1 to reuse parts and save some budget

Plan/Design- Review the rules, research ideas, and select a robot design

Build- Begin assembling the pieces

Test- Intermittently test what we have built and change as necessary

Program- Set up controls and autonomies

Drive- Practice learning controls and strategies

Competition- Final performance of the robot at Sullivan East Middle School

project Timeline designed by: Vanessa Perkins witnessed by: Eli Strike

date: 8/1/2023

Iteration 2

Building

Brainstorming Base Designs:

Date: 8/2

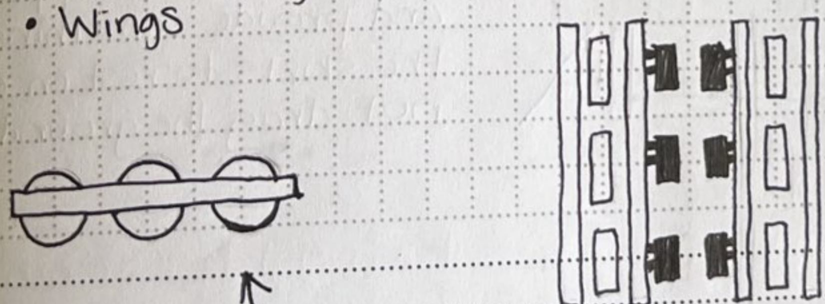
Goals for today → Research and come up with atleast 3 base design options to select from
→ Respect and listen to every team member's ideas

Design Option One: 6 Inch Robot Drive

Specifics:
- 6 Motor Drive
- Slim Design
- 6 Omni 3.25 inch wheels
- Gearing

Purpose:
The purpose of this robot is to have a slim and simple base design for future subsystems we will be adding. Keeping the entire build small and under a height of six (6) inches allows us to drive under the goals to score and descore triballs with ease.

On top of the base, this design could possibly have the following additionally subsystems:
- 1 motor catapult
- 1 motor intake
- Low hanging hook
- Wings



Rough Sketches of Side and Above View

(Continued)

Design Option Two: Mecanum

The specifics for this drive design are located on the bottom of page 11.

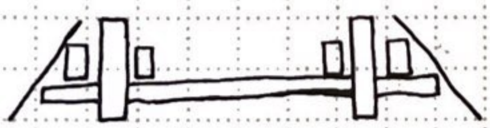
Design Option Three: Skirted Base

Specifics:

- 6 motor drive
- 36:60 gear ratio } 360 rpm
- 600 rpm blue cartridge }
- skirts

Purpose:

The purpose of this base is to be sturdy and durable to be able to hold several large subsystems in comparison to the 6 inch Robot Base. A long c-channel will go across the bottom of the base in between the wheels to act as support and provide something for the skirts to rest on and not drag the ground.



project Brainstorming Base Designs (Continued) designed by: Vanessa Perkins witnessed by: Eli Frutts date: 8/3/2023

VEX

Date: 8/3

Iteration 2

Building

Select Approach:

Goals for today ⟶ Chose a base design that best meets the requirements of the game so we can begin building

We created a decision matrix of what aspects mattered the most on our base. Below is that decision matrix with the winning base design.

Base Ideas	Speed	Torque	Space Efficient	Durability	Total
6 Inch Drive	5	3	4	3	15
Mecanum	3	5	2	2	12
Skirted Base	4	3	4	5	16

* Speed: how fast the base is.
Torque: the strength, ability to play defense, and withstand defense.
Space Efficient: structure and size of base to make is easier to navigate the field but still have adequate room for building future subsystems.
Durability: how well the base holds up against other robots and back to back matches.

Selected Design: Skirted Base

Explanation: The skirted base design is similar to our old base design with some slight adjustments to help it hold up to the rigor of the game. The skirt increases the durability of this base design without compromising speed, torque, or space. The 6 motor design with gearing makes driving around scoring faster and more efficient.

* Break down of each criteria aspect we looked at. A rank of 0(zero) means bad, and a rank of 5(five) is good.

witnessed by: Eli Foiths

project Select Approach designed by: Vanessa Perkins

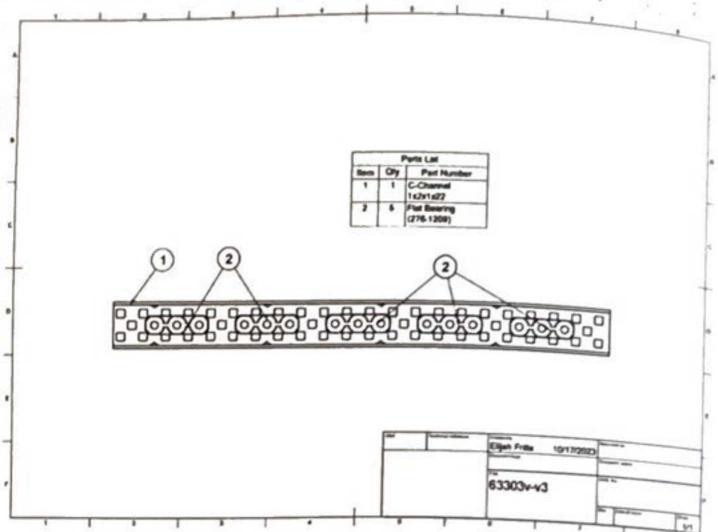
date: 8/3/2023

Date: 8/4

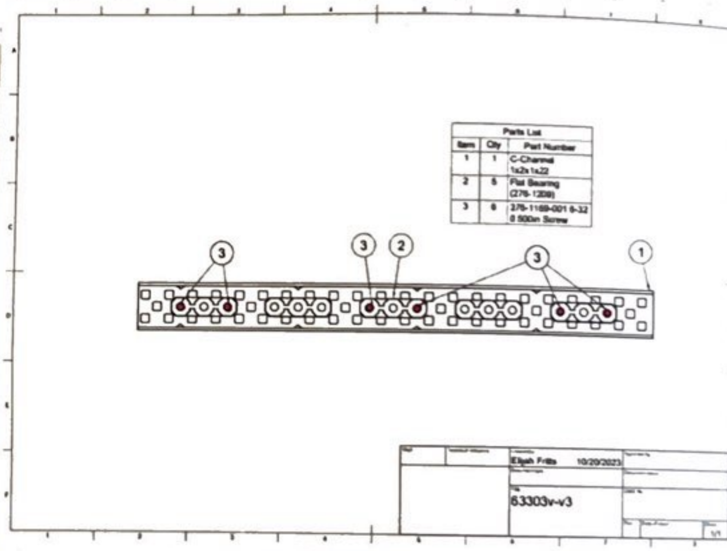
Begin Building Base:

Goals for today → Gather parts for base
→ Begin assembling
→ CAD along the way
→ Get base structure finished

(1) Gather or cut a twenty-two (22) long aluminum c-channel. Place five (5) bearing flats in the place depicted to the right.



(2) Gather six (6) 0.500 inch screws. Place screws into the outside holes of every other bearing flat. The screws are highlighted in pink* as well as labeled #3 in the CAD image to the right.



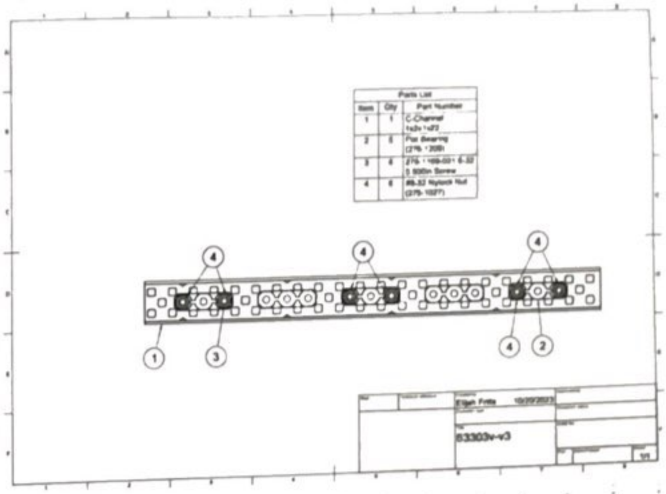
We used plastic screws to make the base more lightweight so we can drive faster and hang easier.

project: Begin Building Base
designed by: Vanessa Perkins + Eli Fritts
witnessed by: Harris Perkins
date: 8/4/2023

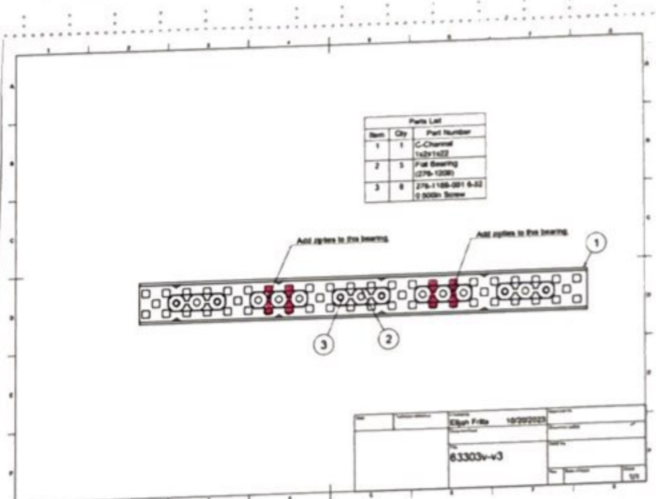
Date: 8/4

(Continued)

(3) Add six (6) nylock nuts to the half inch screws to tighten the three (3) bearing flats to the c-channel. The nylock nuts are labeled as #4.
The nuts we used also plastic like our screws.



(4) Gather four (4) zipties. Add two (2) zipties each to the remaining loose bearing flats around the center bearing flats holes inbetween the grooves and through the the c-channel holes above and below the bearing flats. An example one is drawn in pink* in each of the four correct locations.
The zipties also help to lighten the robot's base as well as keeping every thing tight without having to worry about screws coming loose.



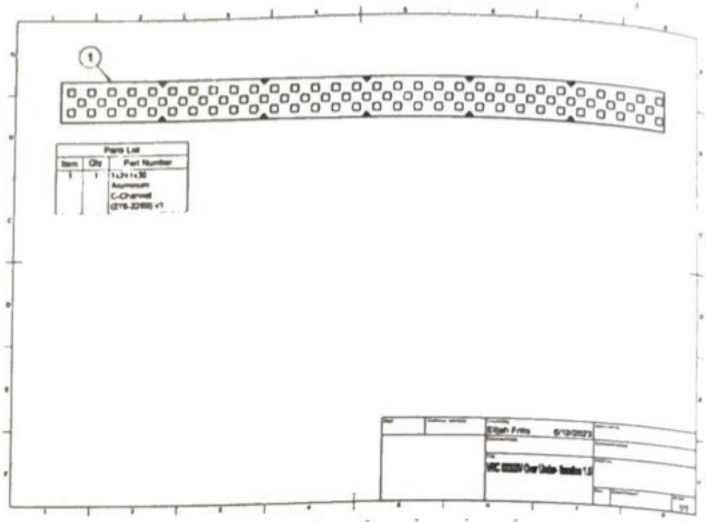
Building

project Begin Building Base (continued)
designed by: Vanessa Perkins & Eli Fritts
witnessed by: Owen taylor
date: 8/4/2023

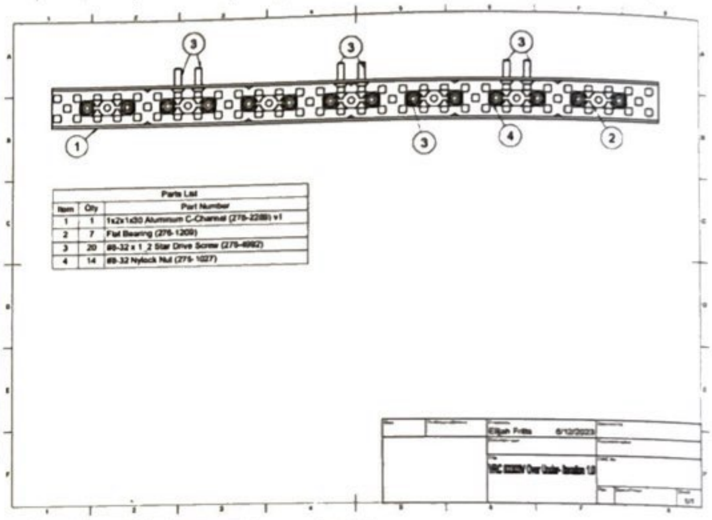
Date: 8/4

(Continued)

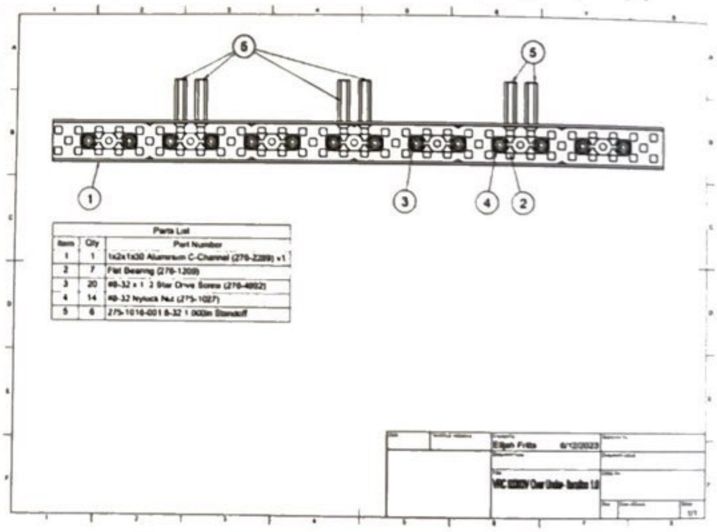
(5) Gather or cut a thirty (30) hole long aluminum c-channel to begin the other half of the base structure.



(6) Add seven (7) bearing flats to the c-channel in the depicted locations using fourteen (14) half inch screws and fourteen (14) nylock nuts. Those three pieces are labeled as numbers 2, 3, and 4. Add six (6) additional half inch screws through the depicted top holes.



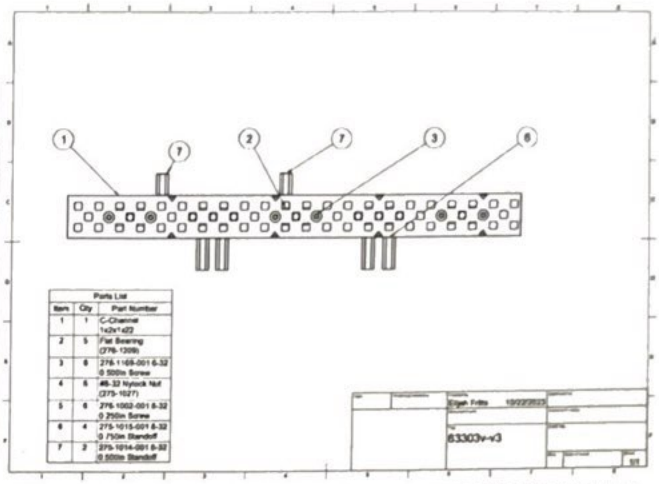
(7) Add one (1.000) inch standoffs to the top screws from step 6 for a total of six (6) standoffs on top of the c-channel.



Date: 8/4

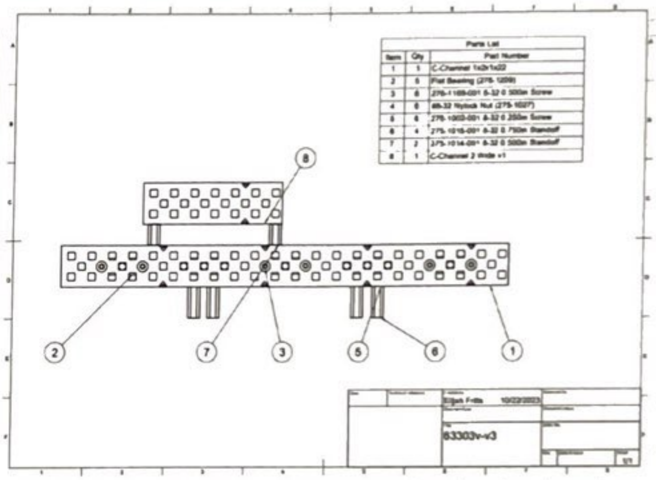
(Continued)

(8) Pick up the twenty-two (22) long c-channel from step 4 and add four (4) 0.750 inch standoffs to the bottom in the depicted locations. Use two (2) 0.250 inch screws to attach two (2) 0.500 inch standoffs to the top depicted locations.

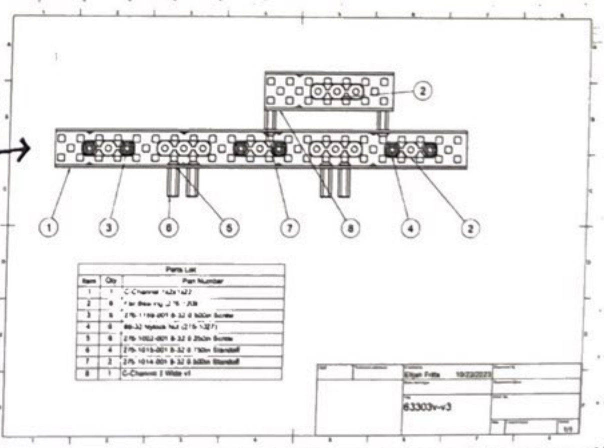


(9) Gather/cut a seven (7) long aluminum c-channel. Attach it to the two (2) half inch standoffs using two (2) 0.250 inch screws.

This piece is for a motor to attach to with gearing to the base.



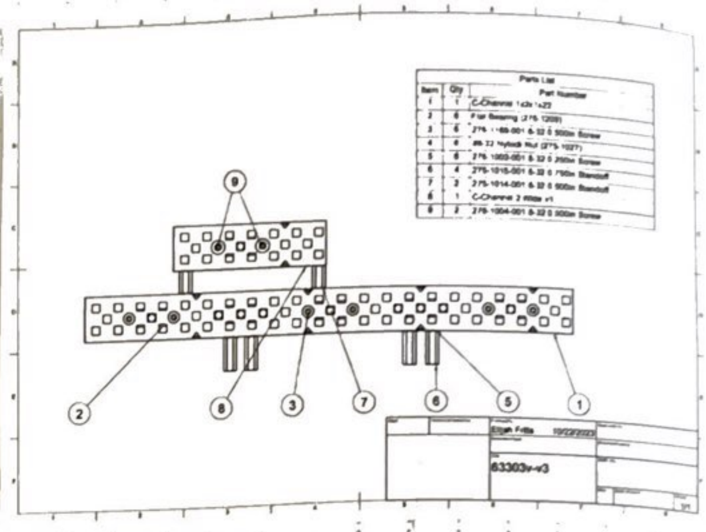
(10) Add a single bearing flat to the seven (7) long c-channel two (2) holes in from the front. →



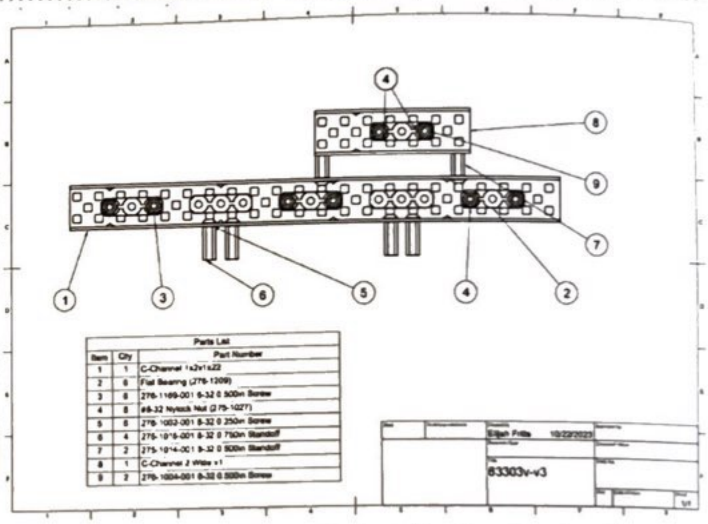
Date: 8/5

(Continued)

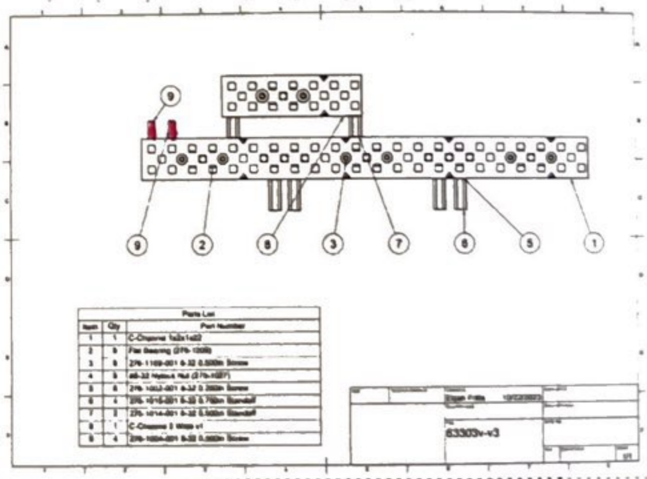
(11) Add two(2) 0.500 inch screws to the seven (7) long c-channel and into the bearing flat from step 10.



(12) Add two(2) nylock nuts to the screws from step 11 to tighten the bearing flat to the c-channel.



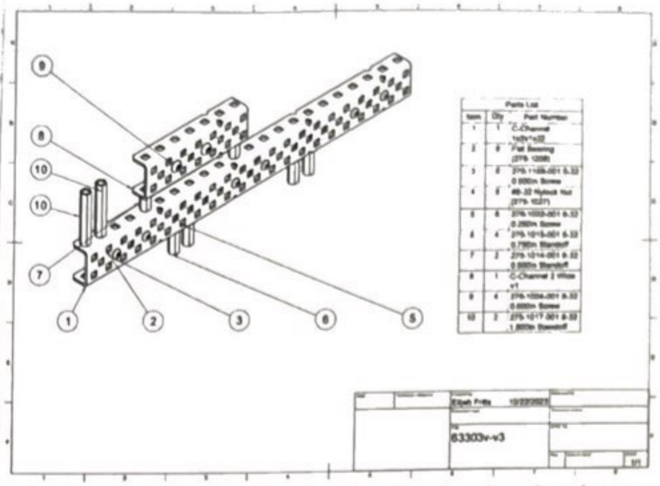
(13) Add two (2) half inch (0.500) screws to the top of the twenty-two(22) long c-channel through the last two holes. They are highlighted in pink* to the right.



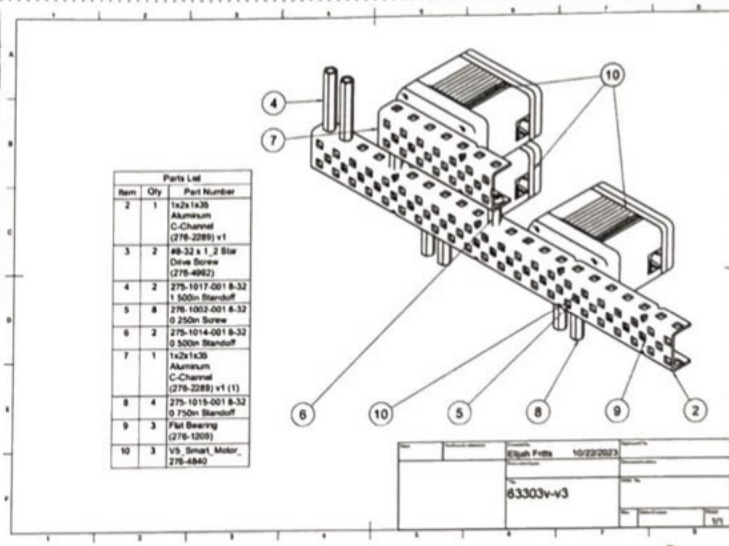
project Begin Building Base (Continued) designed by: Vanessa Perkins & Eli Fritts witnessed by: Harris Perkins date: 8/5/2023

(Continued)

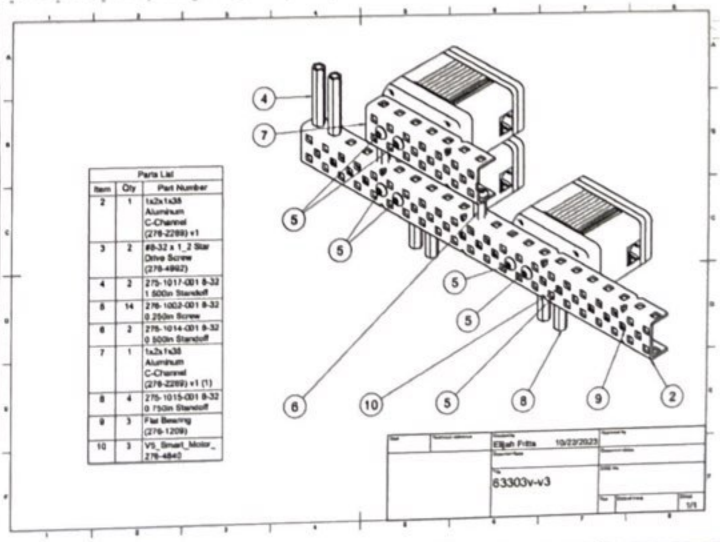
(14) Add 1.500 inch standoffs to the screws from Step 13. Uses a total of two (2) standoffs.



(15) Gather three (3) motors with the blue 600 rpm inserts. Place them in the depicted locations with one on the seven (7) long c-channel and two on the twenty-two long c-channel.



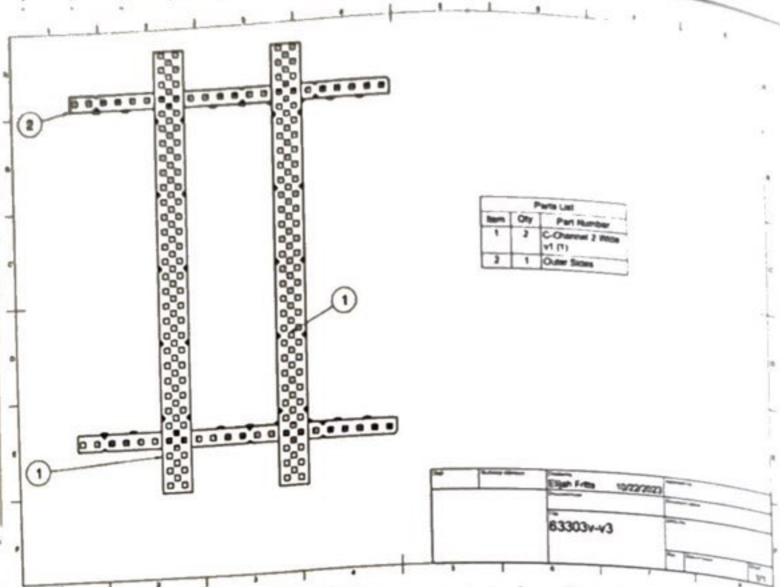
(16) Add six (6) 0.250 inch screws to the motors and tighten. Repeat all steps up through 16 for a total on two (2) twenty-two (22) long c-channels with motors and two (2) thirty (30) long c-channels.



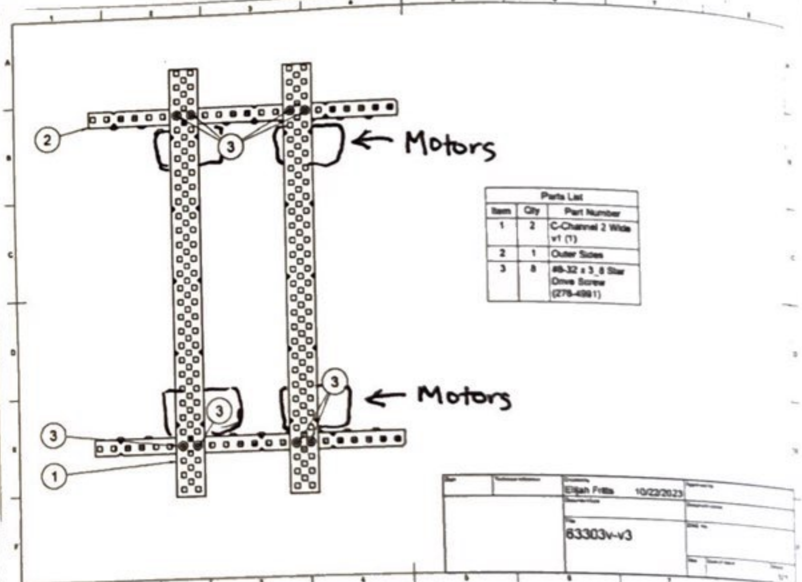
project: Begin Building Base (Continued) designed by: Vanessa Perkins + Eli Fritts witnessed by: Owen Taylor date: 8/5/2023

(Continued)

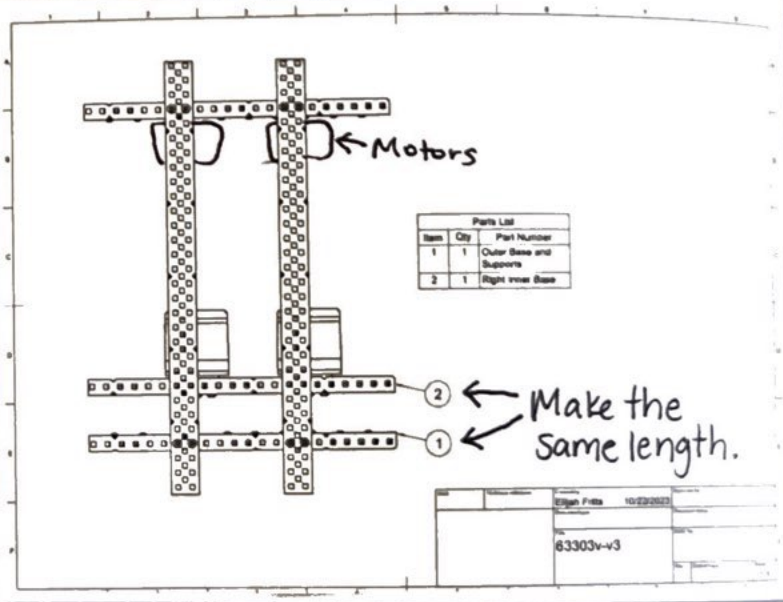
(17) Gather two (2) thirty (30) long aluminum C-channels to go underneath the base.



(18) Add eight (8) 0.250 inch screws to the thirty (30) long C-channels to connect them to the standoffs underneath on the motored C-channels from Step 16.



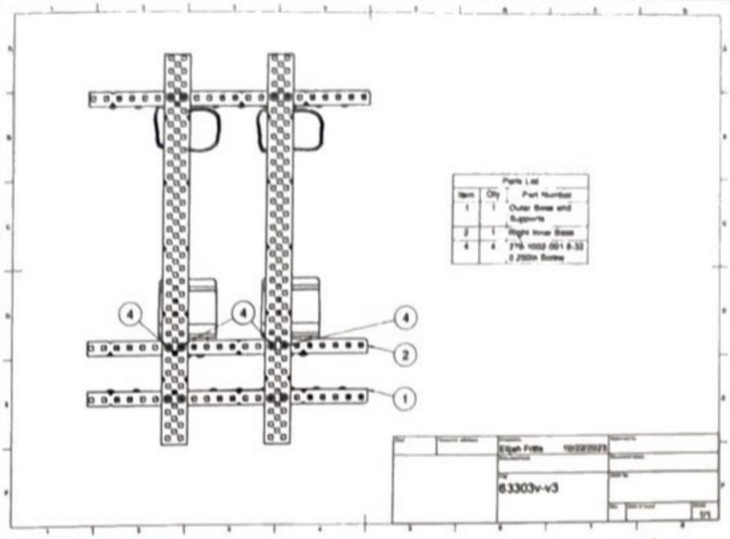
(19) We decided to redesign the thirty (30) long C-channels from Step 7 for shorter ones that match the twenty-two (22) long ones. Everything about the two (2) twenty-two long C-channels was duplicated except steps 15 and 16 with the motors.



project: Begin Building Base (Continued) designed by: Vanessa Perkins + Eli Fritts witnessed by: Harris Perkins date: 8/5/2023

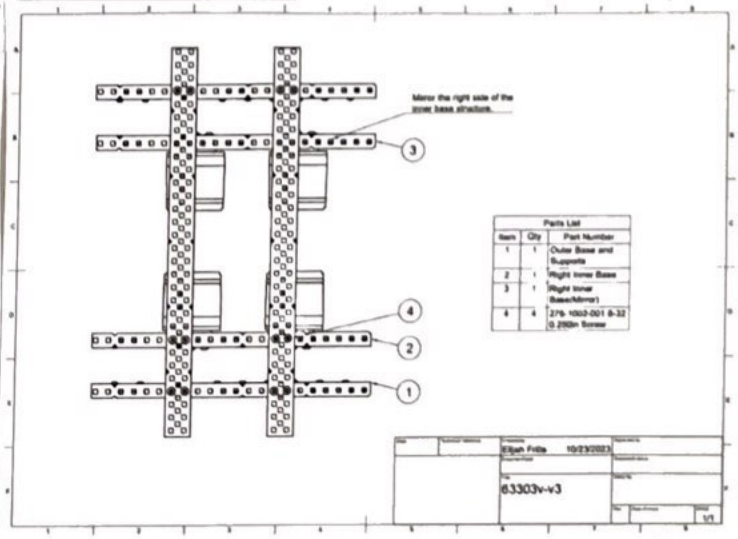
(Continued)

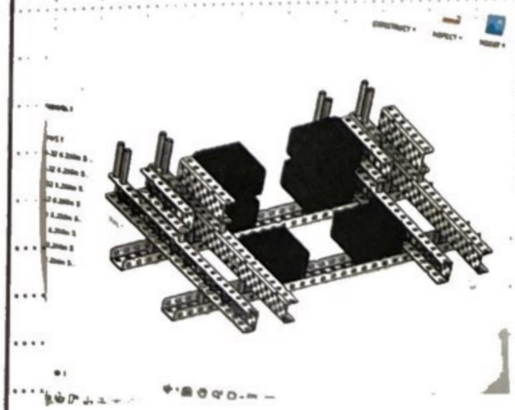
(20) Add the new twenty-two (22) long c-channel from step 19 to the thirty (30) long c-channels using four (4) 0.250 inch screws into the bottom standoffs.

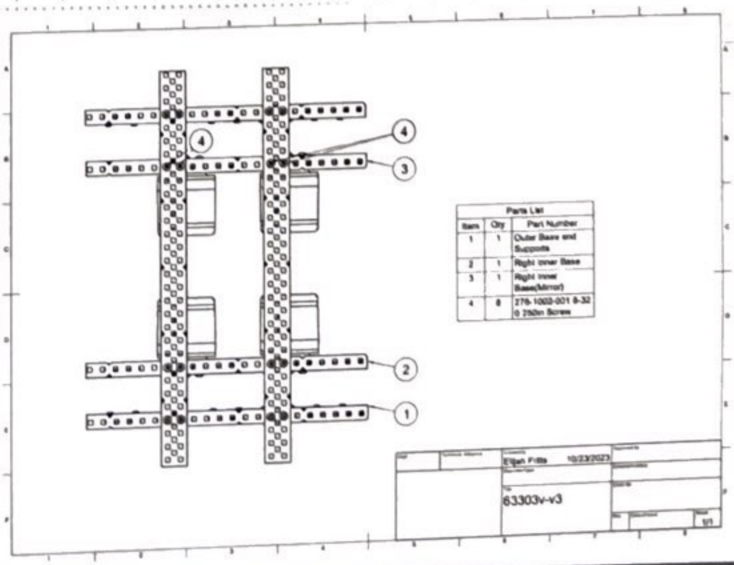


(21) Mirror step 20 with the last c-channels. Make sure the 30 long c-channel is attached underneath the other c-channels. The lower right image is a further image of step 21.

Below is a colored image of the CAD up to this point in time.



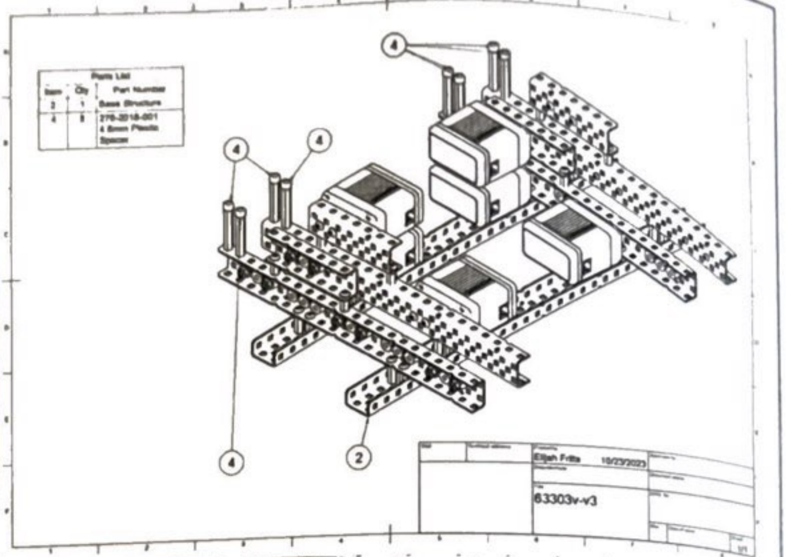




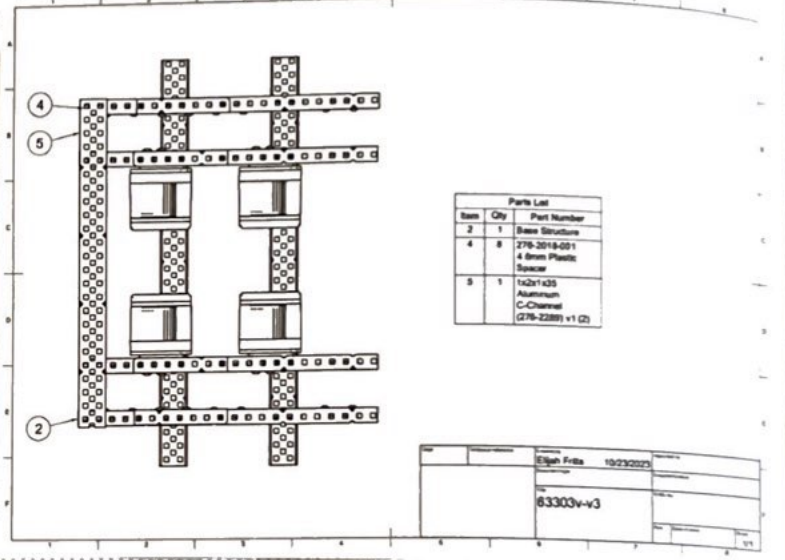
project: Begin Building Base (continued) designed by: Vanessa Perkins + Eli Fritts witnessed by: Harris Perkins date: 8/6/2023

(Continued)

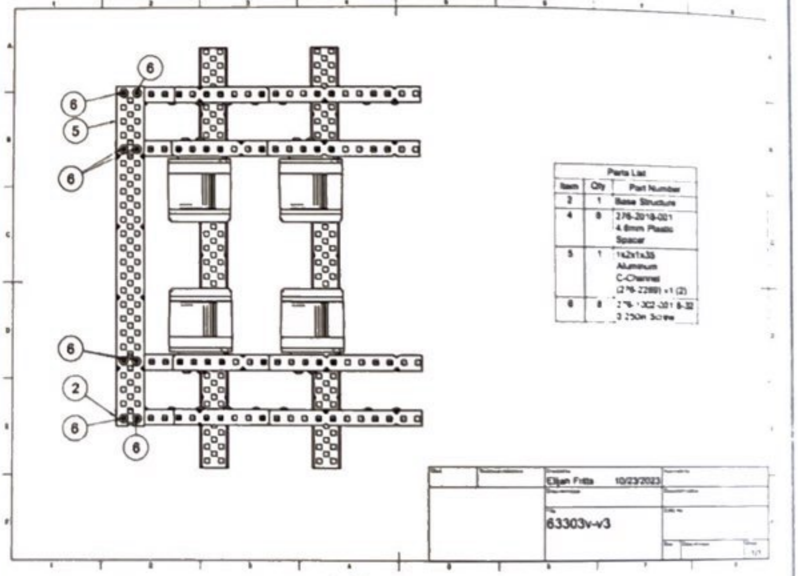
(22) Gather eight (8) 4.6mm spacers. Place one (1) on top each of the 1.500 ~~st~~ inch standoffs on the back of the base.



(23) Gather or cut one (1) twenty-four (24) long aluminum c-channel. Place in the depicted location to the right on top the spacers from step 22 on the back of the base.



(24) Gather eight (8) 0.250 inch screws to place in the top of the twenty-four (24) long c-channel and into the spacer and standoffs below.

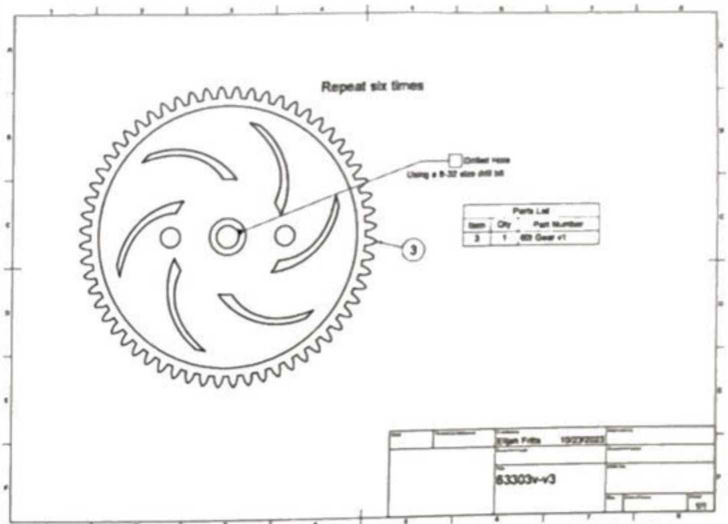


End of Base Structure

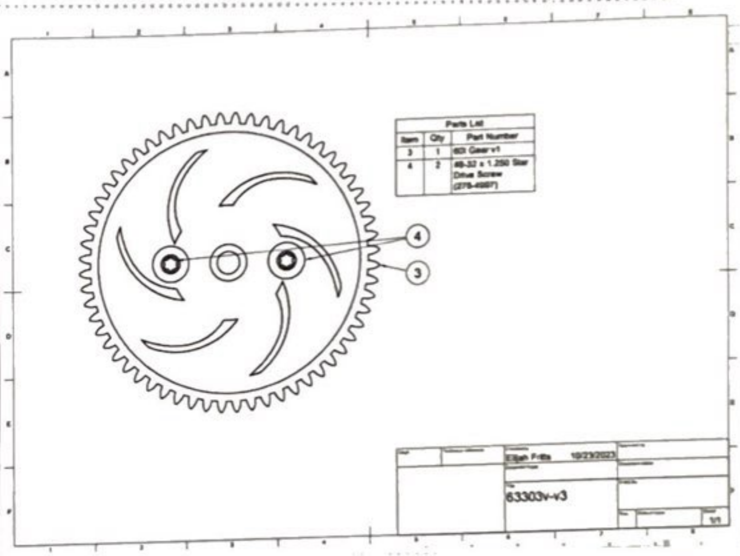
project: Begin Building Base (Continued) designed by: Vanessa Perkins + Eli Fritts witnessed by: Owen Taylor date: 8/16/2023

Adding Gears and Wheels:

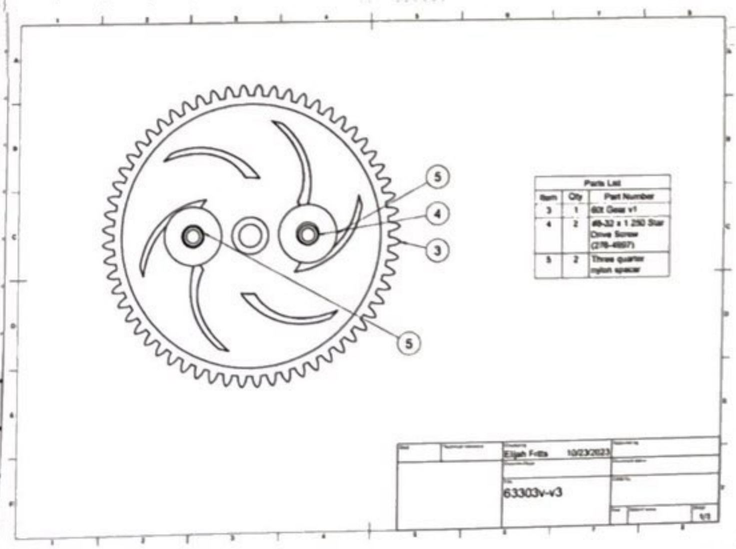
(1) Gather a 60 tooth gear and drill a hole through the center with the noted bit size for the axel to go through. Repeat for a total of six (6) gears.



(2) Insert two (2) 1.250 inch screws up through each gear for a total of twelve (12) screws.

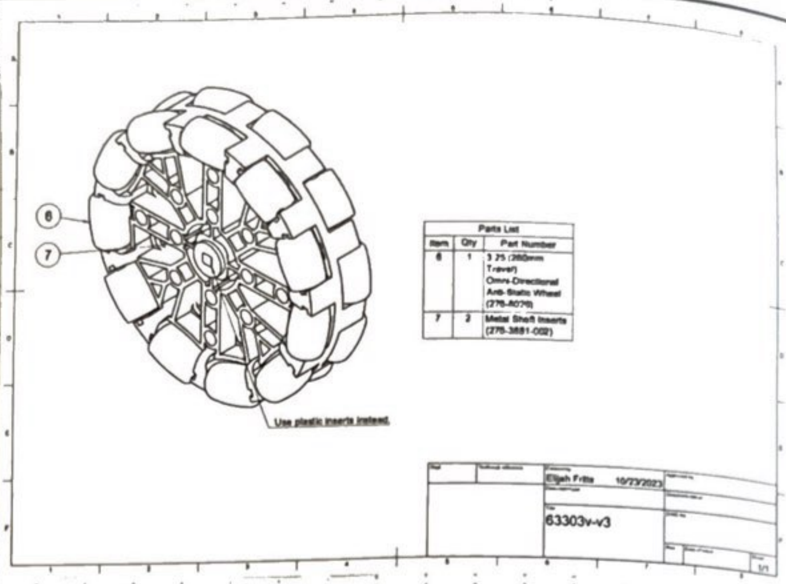


(3) Add one (1) 0.750 inch Spacer to each of the twelve (12) screws on top of the gears for a total of twelve (12) spacers.

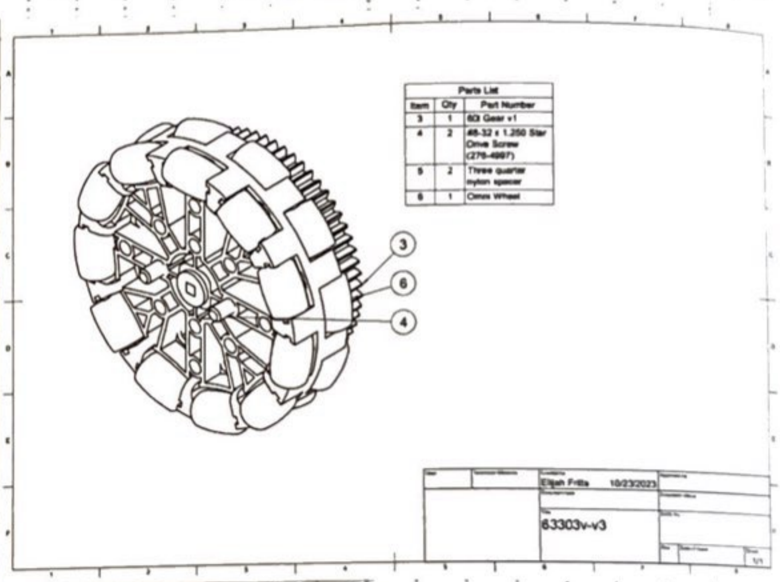


project Adding Gears and Wheels designed by: Vanessa Perkins + Eli Fritts witnessed by: Owen Taylor date: 8/6/2023

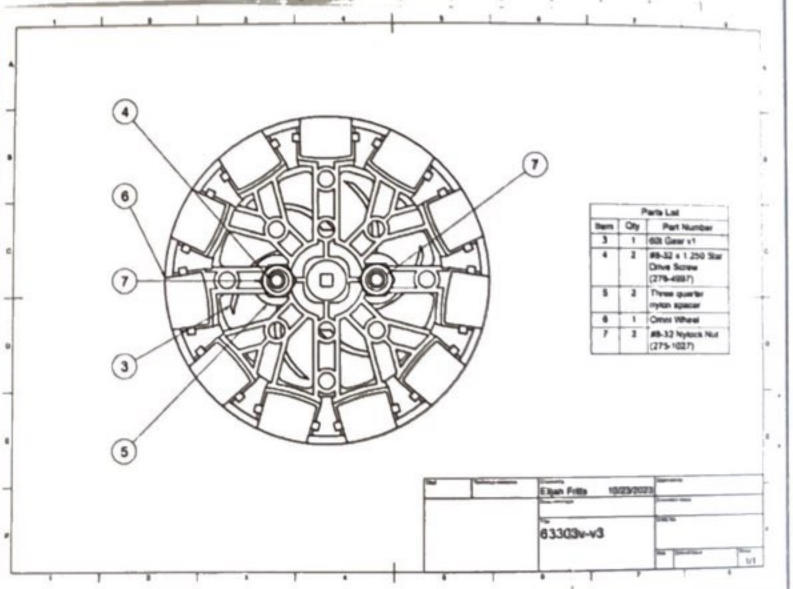
(Continued)



④ Gather six (6) wheels, four (4) omni and two (2) traction wheels. They all have a diameter of 3.25 inches. Use twelve (12) metal shaft inserts to place in each of the wheels.



⑤ Insert the wheels through the screws on the gears with the spacers from step 3 still inbetween. Repeat for all six (6) wheels.

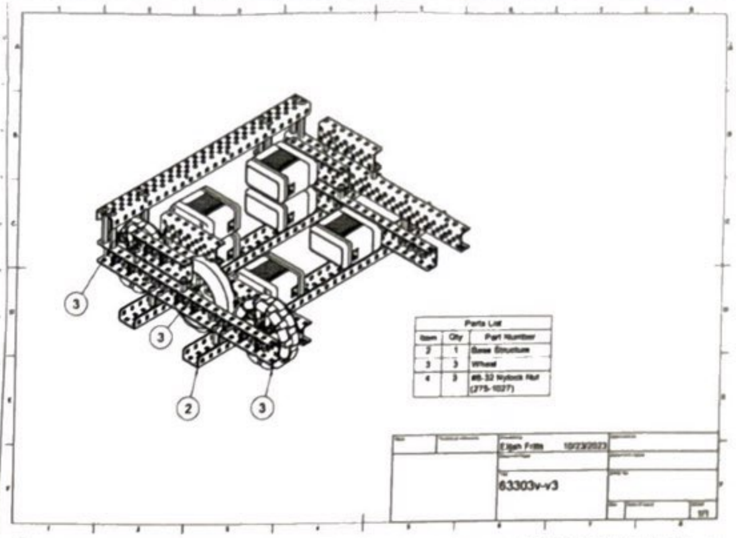


⑥ Gather twelve (12) nylock nuts and place on the screws to tighten the wheels down to the gears so both things will rotate together.

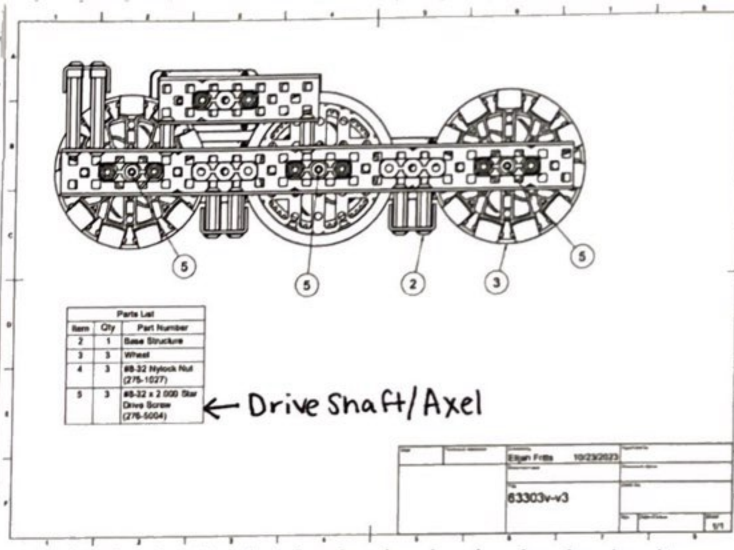
project: Adding Gears and Wheels (Continued) designed by: Vanessa Perkins + Eli Fritts witnessed by: Harris Perkins date: 8/6/2023

(continued)

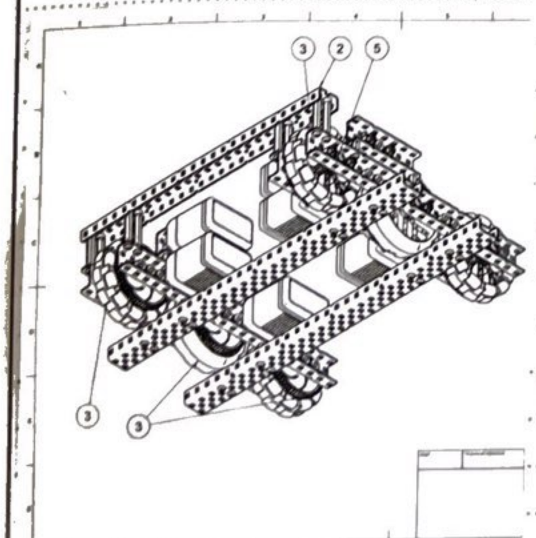
7 Place the wheels in the correct location in between the c-channels of the base.

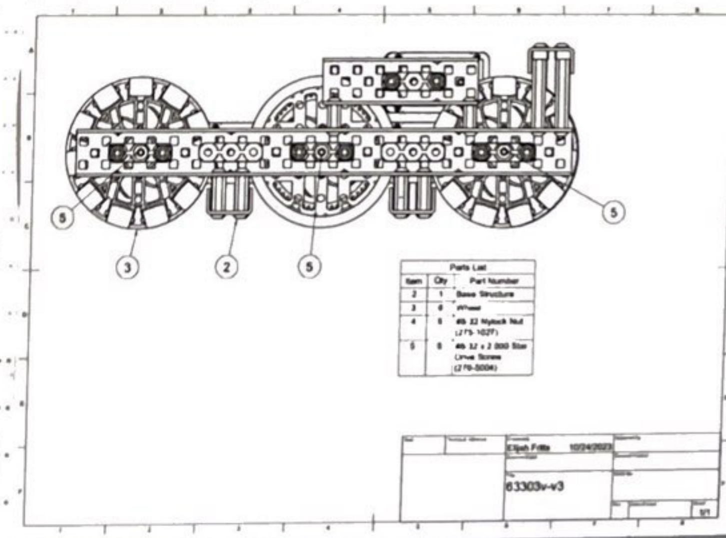


8 Insert 2.000 inch screws into the six (6) wheels with a loose nylock nut on each end for a total of six (6) screws and six (6) nylock nuts.
Below are images of the wheels added to the base.



← Drive Shaft/Axel

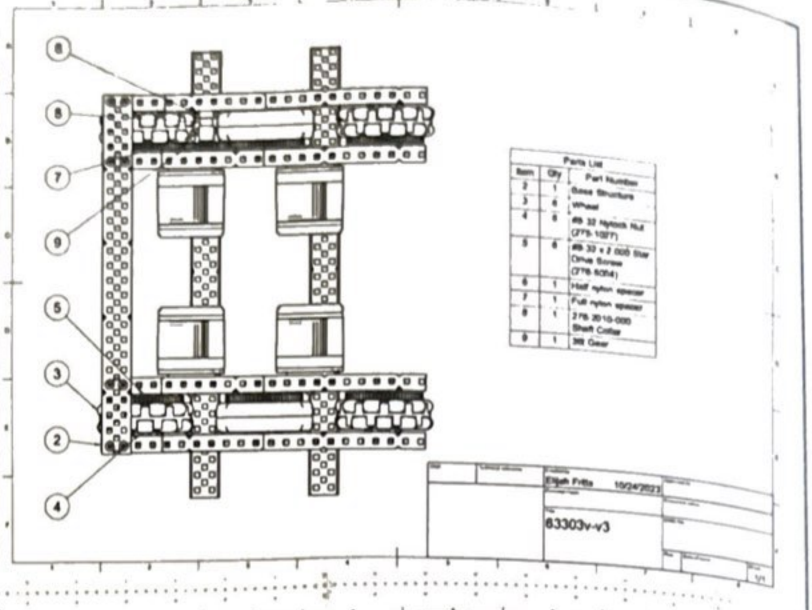




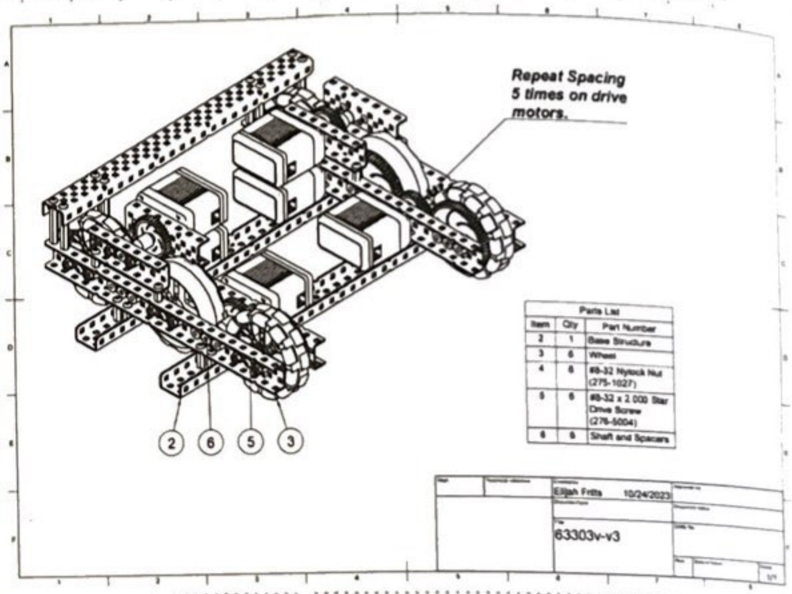
project: Adding Gears and Wheels (continued) designed by: Vanessa Perkins + Eli Fritts witnessed by: Owen Taylin date: 8/6/2023

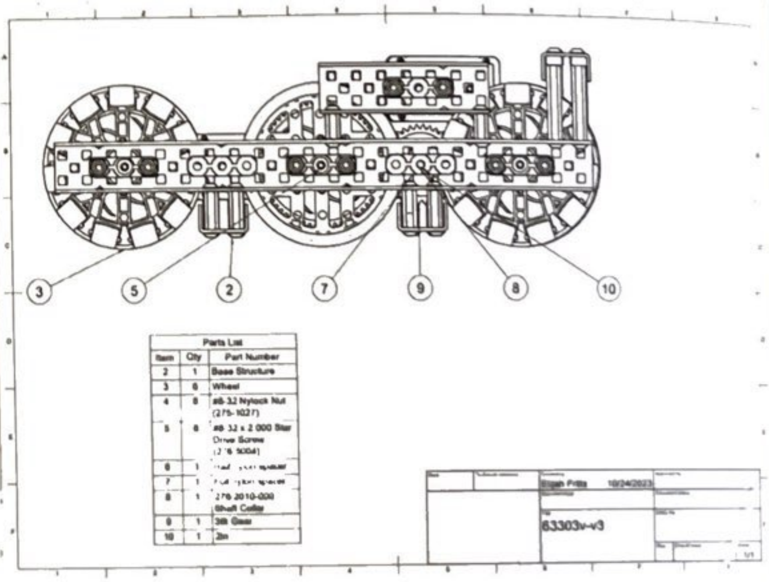
(Continued)

(9) Get a 3.00 inch drive shaft. Add a 36 tooth gear then a 0.500 inch spacer, shaft collar, and 0.250 inch spacer. Insert into a motor then tighten the shaft collar.



(10) Repeat step 9 Five (5) more times for the other remaining motors. The back motors will have their 36 tooth gears stacked on top of each other. In the bottom right corner is another angle of the base at the current moment for more detail.



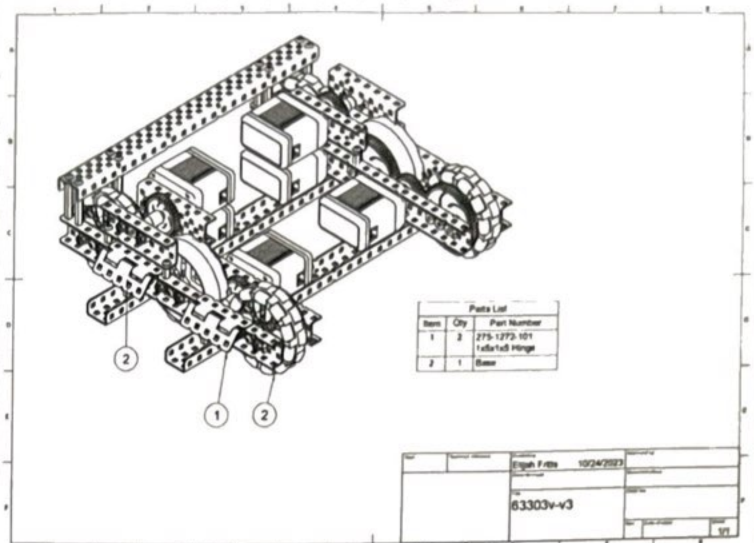


VP

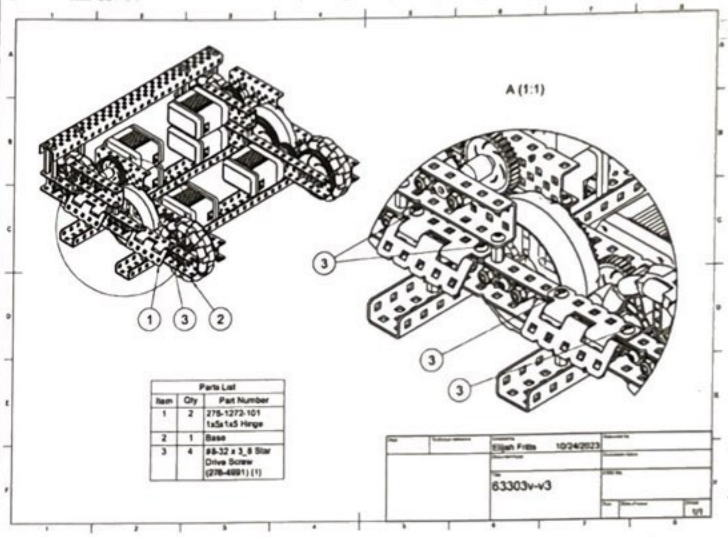
project Adding Gears + Wheels (continued) designed by: Vanessa Perkins + Eli Fritts witnessed by: Owen Taylor date: 8/6/2023

Adding Skirts to Base:

1. Gather and put in place two (2) 1x5 hinges on the outside C-channel of the base. They go three (3) holes from the front and five (5) from the back.

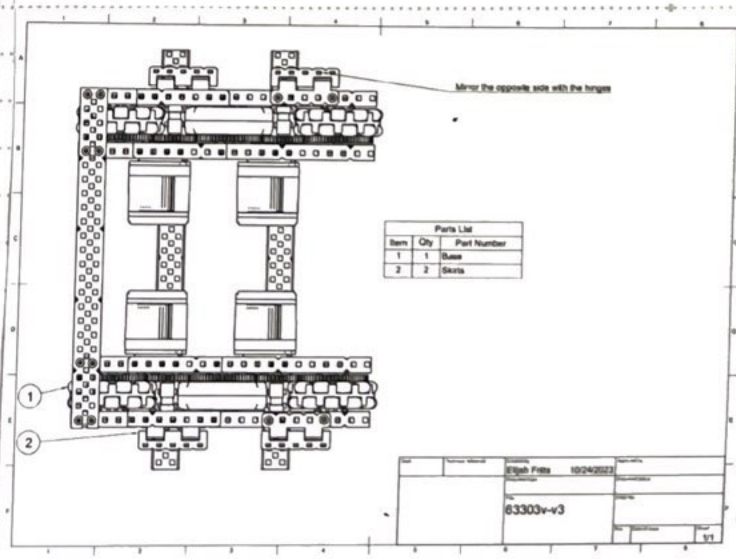


2. Add two (2) inch screws to each hinge for a total of four (4) screws. Use four (4) nylock nuts to tighten hinges to base.



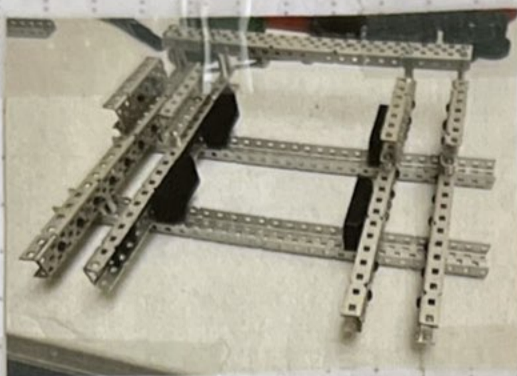
3. Cut out a sheet of Lexan for skirts. Drill four (4) holes, two (2) in the middle, one (1) on each end. Attach to a twenty-two (22) long 1 by 1 L-channel using four (4) 0.500 inch screws and nylock nuts. Attach Lexan and L-channel to hinges using four (4) 0.250 inch screws and nuts. Lexan Dimensions on page 113.

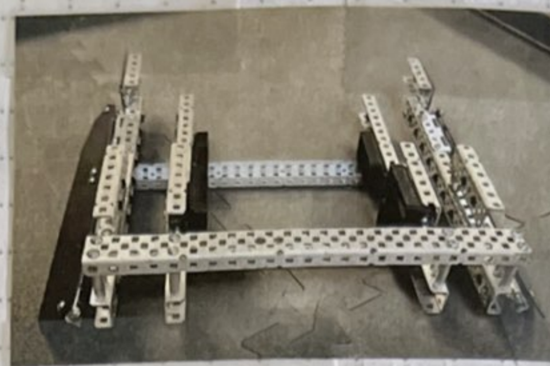
Not Depicted

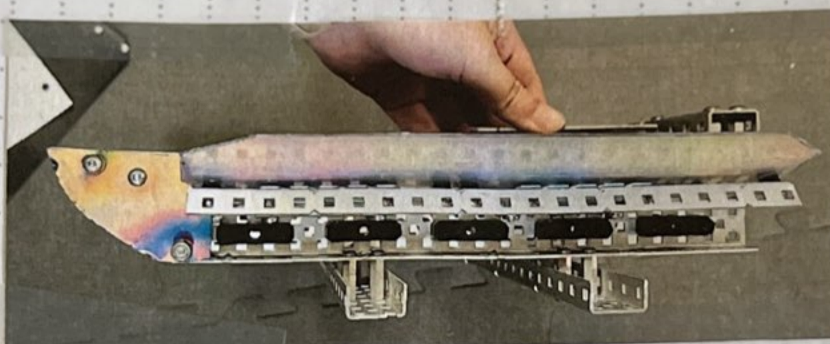


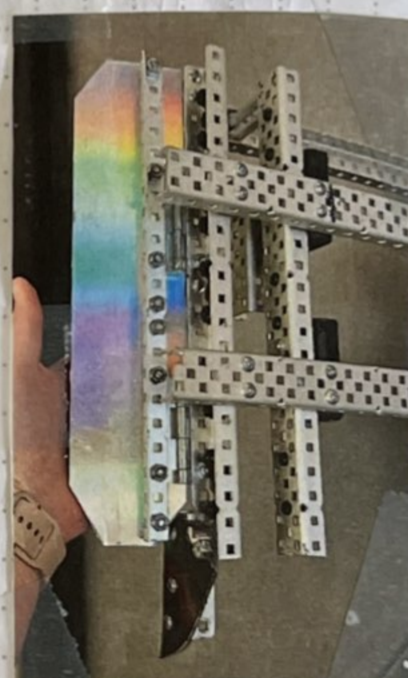
4. Repeat for opposite as depicted to the right.

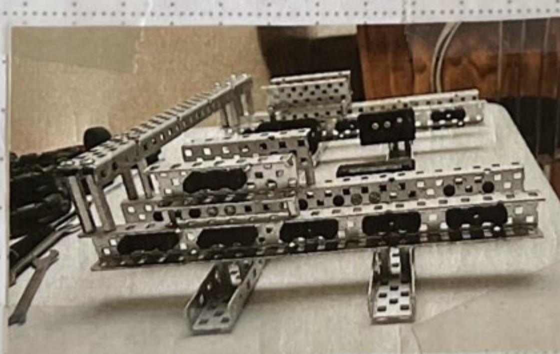
Skirted Base Images:

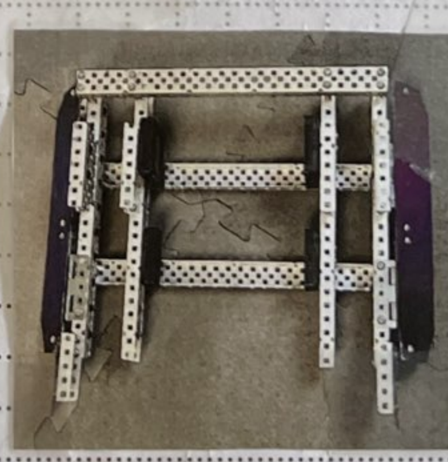


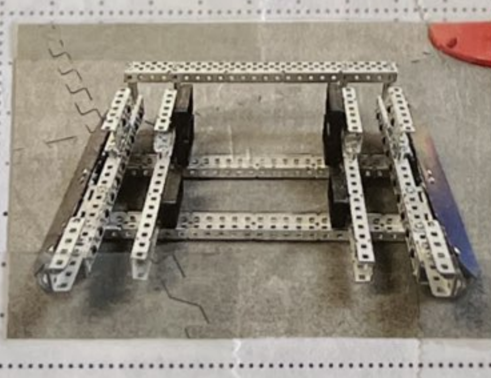








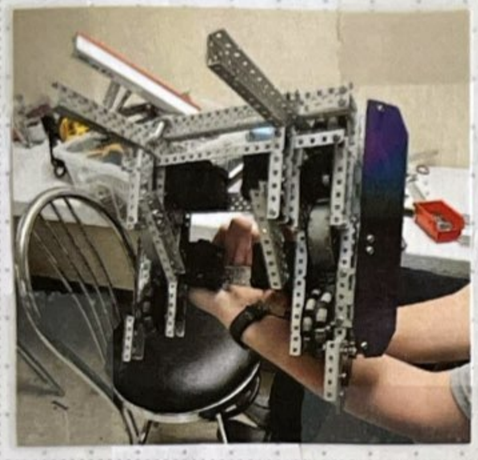


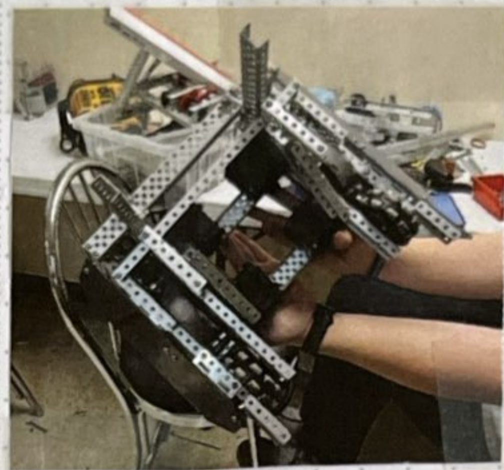


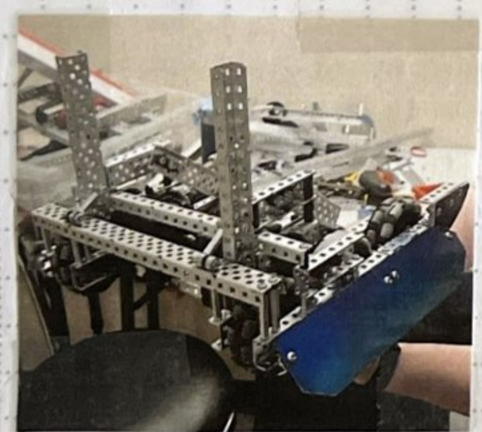
project Skirted Base Images designed by: Vanessa Perkins witnessed by: Harris Perkins

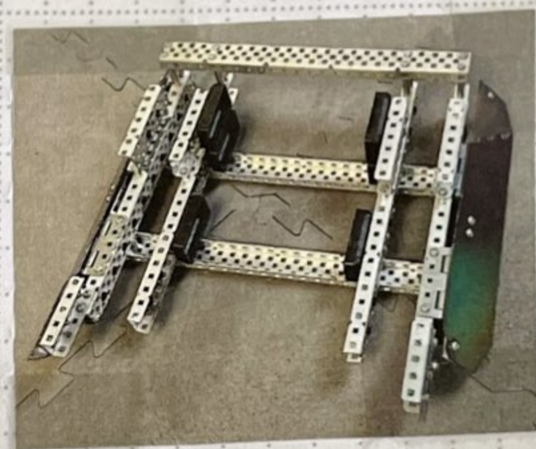
date: 8/9/2023

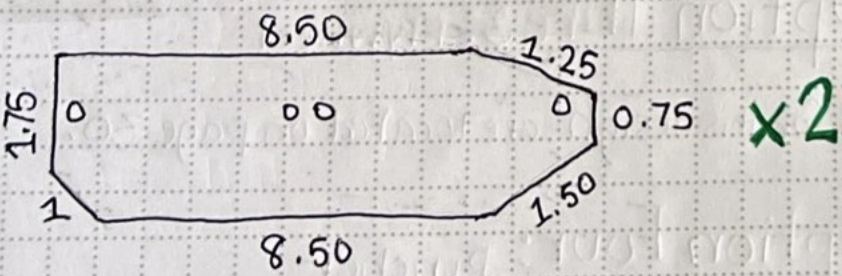
(Continued)











Above is the Lexan Sheet Dimensions from step 3 on page 111, We used two pieces of that size to make each of the skirts on the base.
The screwholes are also depicted above, for reference, on the Dimensions drawing.

Brainstorming Launch Mechanism Designs:

Date: 9/1

Goals for today → Come up with atleast 3 design options

Design Option One: Catapult

The specifics for this design are located on page 29.

Design Option Two: Hybrid

Specifics:
- 1 motor
- Slip gear
- 20 rpm
- rubberbands

Purpose:
The purpose of this catapult puncher hybrid design is to be able to launch triballs either from a normal catapult position during matches or be able to place triballs on the top to be punched with the catapult arm when using matchloads.

Design Option Three: Flywheel

The specifics for this design are located on page 30.

Design Option Four: Puncher

The specifics for this design are located on page 31.

VP

Select Approach:

Date: 9/1

Goals for today → Choose a launch mechanism design that best meets the requirements of the game.

We created a decision matrix of what aspects mattered the most on our launch mechanism. Below is the matrix with the winning design.

Ideas	Speed	Size	Force	Durability	Total
Catapult	3	2	4	2	11
Hybrid	5	2	5	3	15
Flywheel	5	1	4	1	11
Puncher	5	5	2	2	14

* Speed: how fast it launches triballs
Size: how compact the design is
Force: how strong it can launch the triballs across the field
Durability: how well it holds up to the repeated launch of matchloads

Selected Design: Hybrid

Explanation: We chose the Hybrid Catapult Puncher design because it will be the most efficient at launching matchloads as well as the triballs already on the field. The Hybrid Design is the most durable to the fast pace of the game. It also will provide the most force to get triballs over the barrier the best. The only challenge will be finding a good launch angle of the catapult arm.

* Break down of each criteria aspect we looked at. A rank of zero(0) means bad and a rank of five(5) means good.

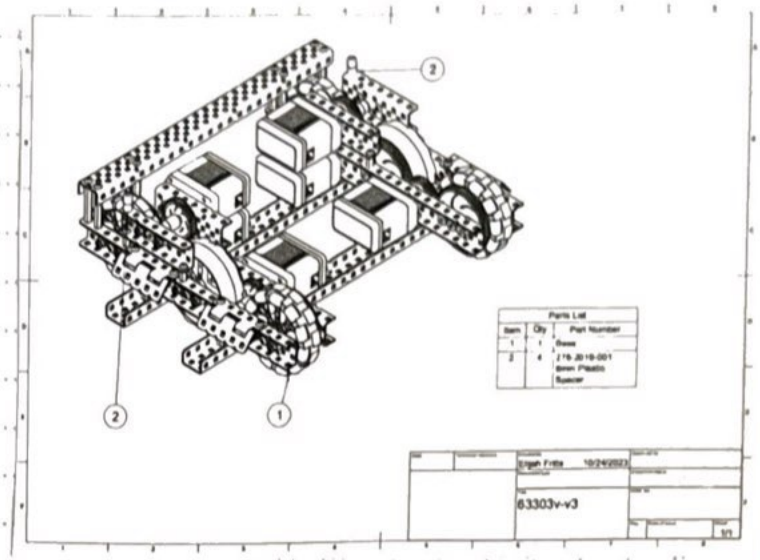
project: Select Approach designed by: Vanessa Perkins witnessed by: Eli [illegible]

date: 9/1/2023

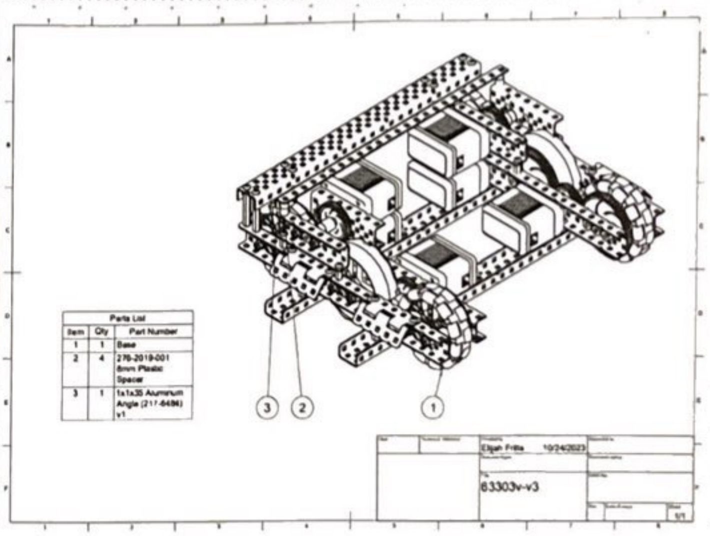
My Projects

Building Catapult Supports:

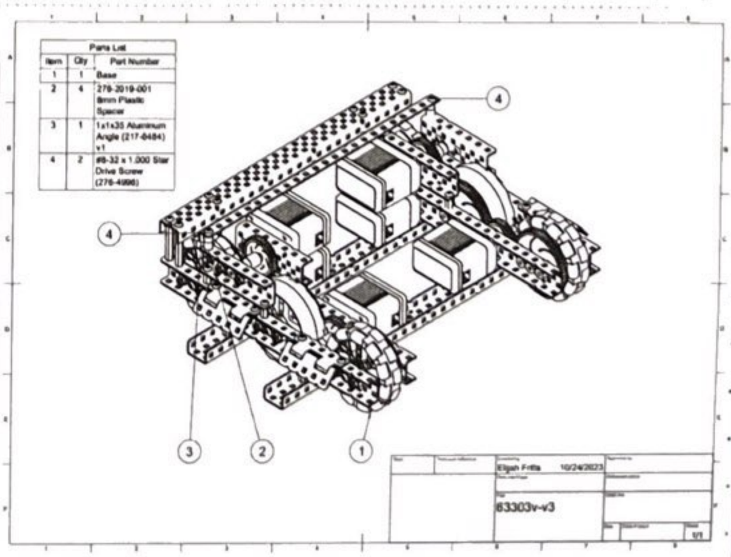
(1) Add two (2) 8mm spacers to the last holes of the outside base seven (7) long c-channels for a total of four (4) spacers used.



(2) Gather or cut a twenty-four (24) long aluminum 1by1 L-channel. Place on top of the step 1 spacers as depicted to the right.



(3) Add a 1.000 inch screw through the 1by1 and spacers and seven (7) long base c-channel on each side for a total of two (2) screws.

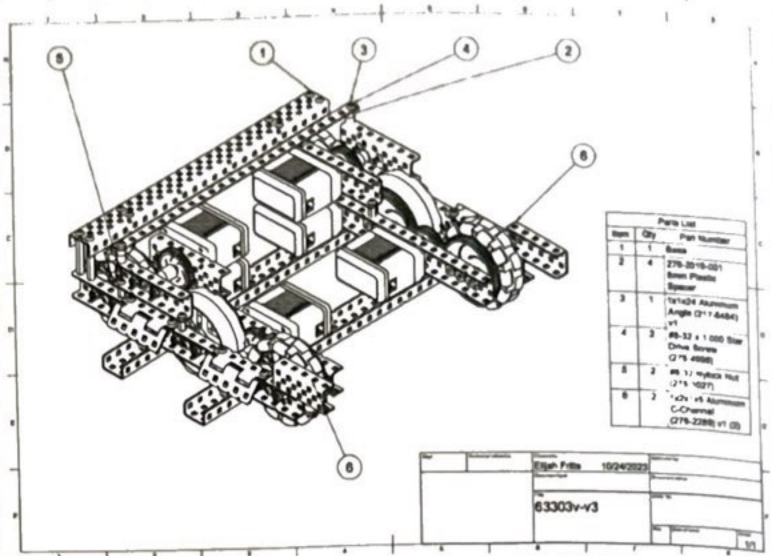


project: Building Catapult Supports designed by: Vanessa Perkins & Eli Fritts witnessed by: Owen Taylor date: 9/4/2023

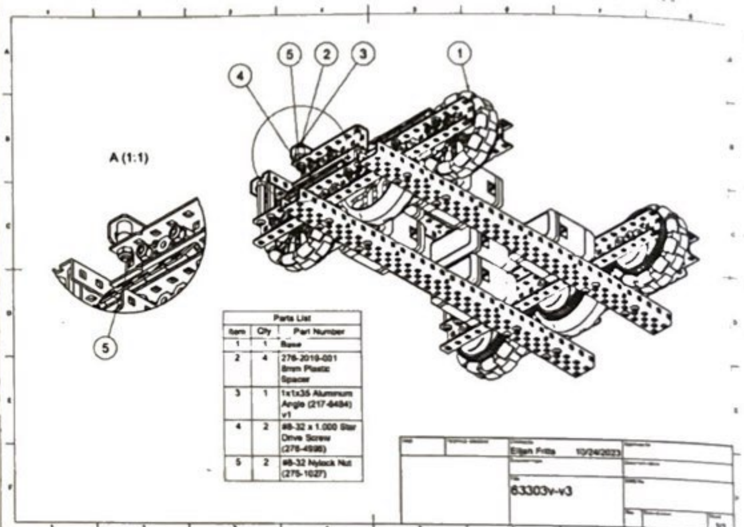
Date: 9/4

(Continued)

(4) Add two (2) nylock nuts to the screws from step 3 and tighten down. Gather/cut two (2) five (5) long aluminum C-channels and place in depicted locations at the front of the robot.



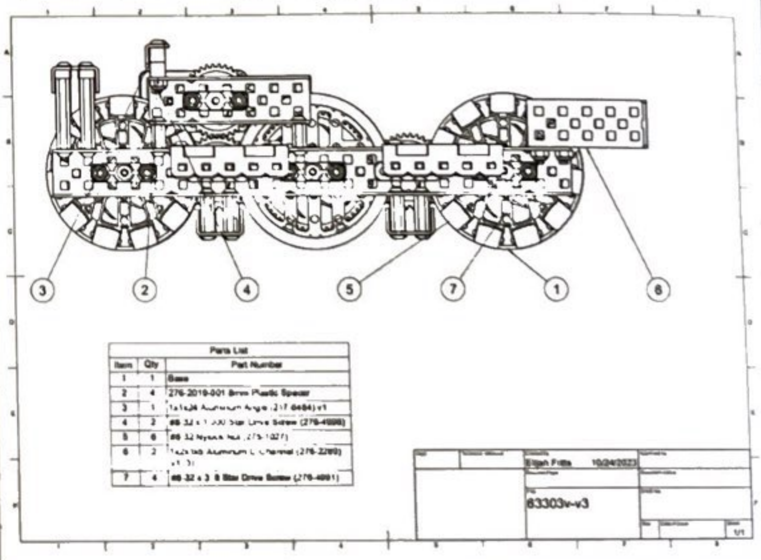
Further depiction of the beginning of step 4. →



(5) Add two (2) 0.250 inch screws to the five (5) long c-channels and two (2) nylock nuts each for a total of four (4) screws and nuts.

(6) Add lexan sleds to the 5 long c-channels on the front of the base.

Lexan sled dimensions are located on page 80.



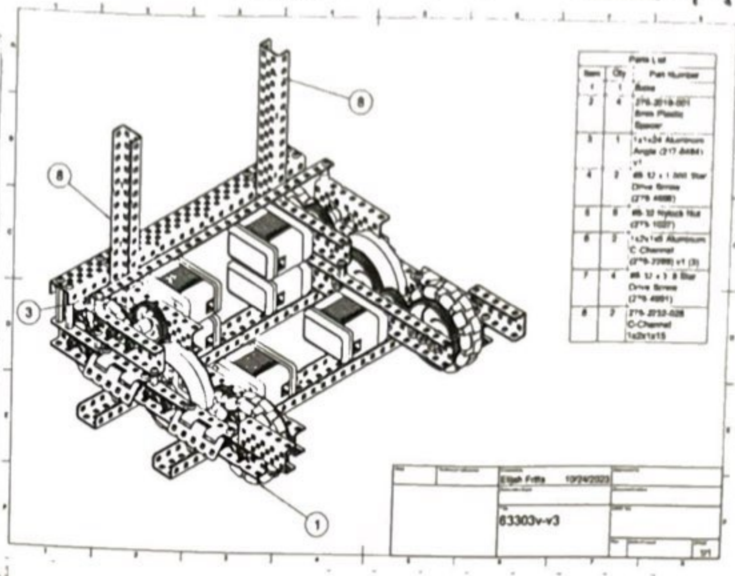
VR

project: Building Catapult Supports (continued) designed by: Vanessa Perkins + Eli Fritts witnessed by: Harris Perkins date: 9/4/2023

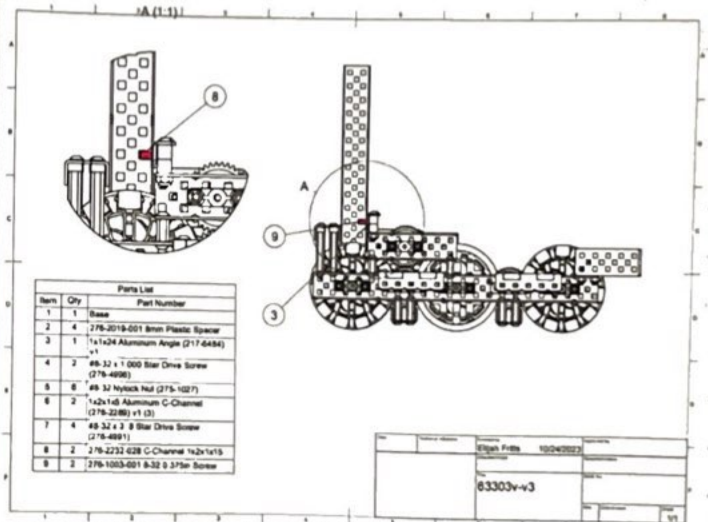
(Continued)

Date: 9/4

7. Gather or cut two(2) fifteen(15) long aluminum c-channels. Place in upright positions as depicted to the right in between the horizontal c-channel and L-channel.

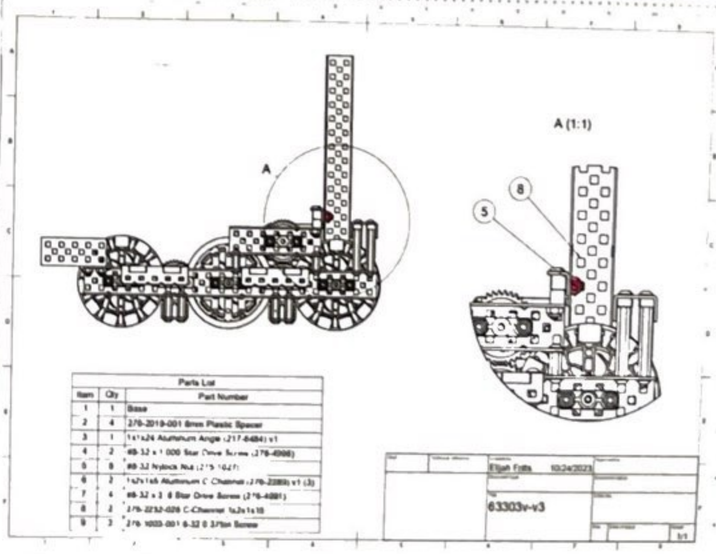


8. Add two(2) 0.375 inch screws* into the fifteen(15) long c-channel through the 1 by 1 L-channel.



VP

9. Add two(2) nylock nuts* to the screws from step 8 and tighten.



VP

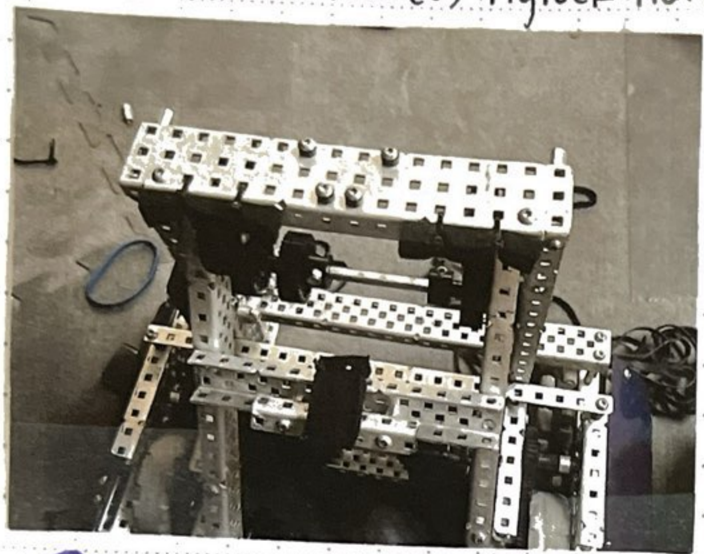
project Building Catapult Supports (Continued) designed by: Vanessa Perkins & Eli Fritts witnessed by: Harris Perkins date: 9/4/2023

Date: 9/5

Building Catapult:

Goals for today → build catapult arm and attach to supports
→ Test Catapult

(1) Attach a sixteen (16) hole long 1x3 c-channel to the top of the catapult supports using two(2) half inch screws and two (2) nylock nuts.



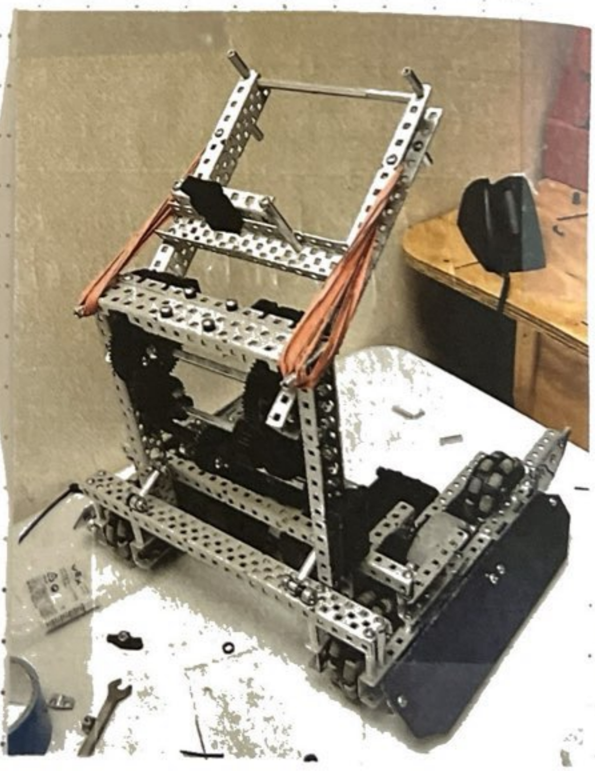
(2) Add two(2) 0.500 inch standoffs on the outside ends of the 1x3 c-channel using two(2) 0.250 inch screws.

(3) Add four(4) 0.125 inch spacers to the top of the 1x3 c-channel in the middle using four(4) 0.250 inch screws and nylock nuts for the triball to sit on.

(4) Add three (3) high strength bearing flats down each side of the supports for a total of six (6) using 0.500 inch screws and nylock nuts. Drill out holes in the metal for the high strength shafts.

(5) Attach a motor to the left side, bottom bearing flat as shown in the picture to the right.

(6) Add foam strips to the 1x3 c-channel to cushion the force of the catapult.



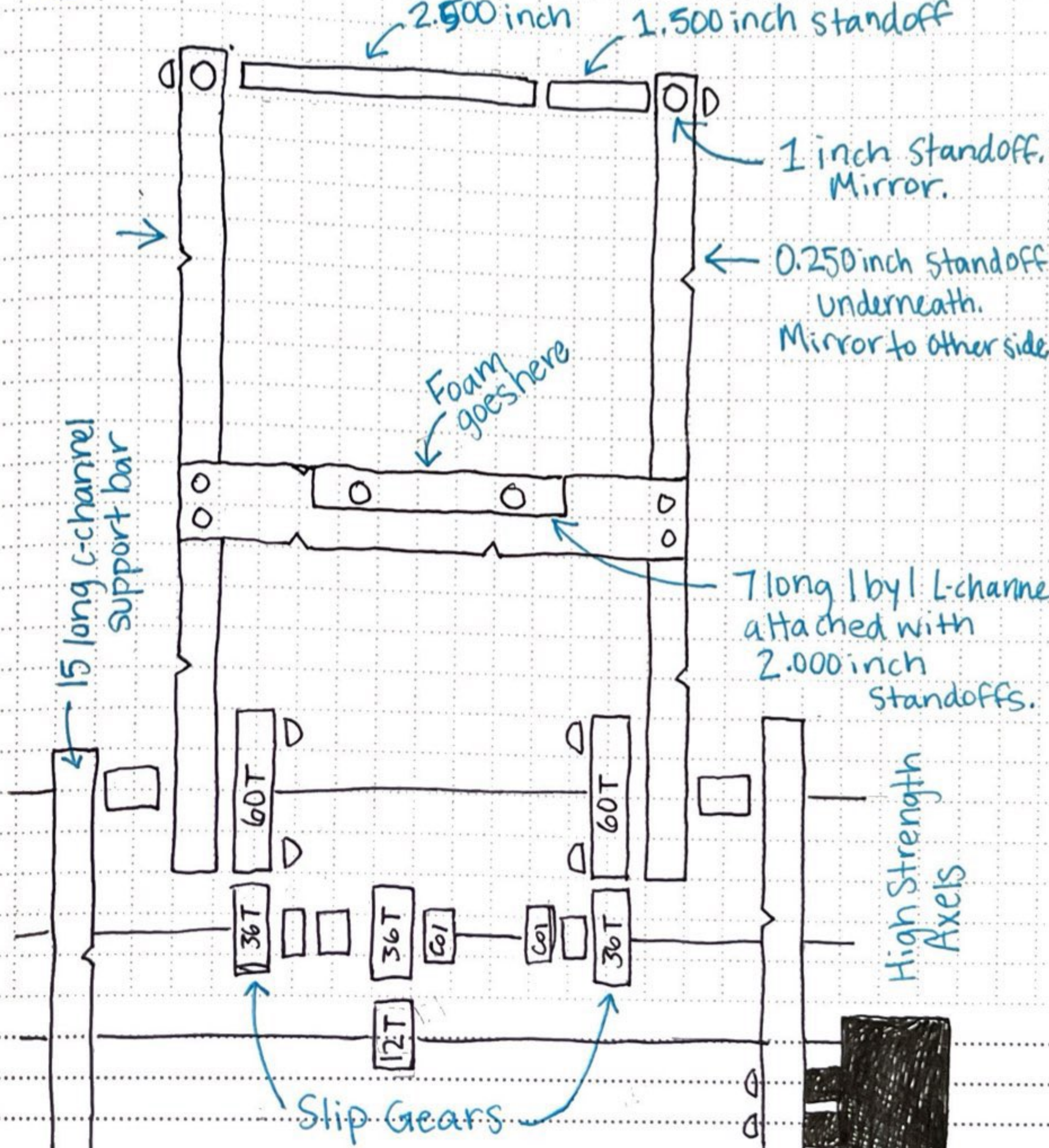
project Building Catapult designed by: Vanessa Perkins witnessed by: Eli [illegible]
date: 9/5/2023

(Continued)

Date: 9/6

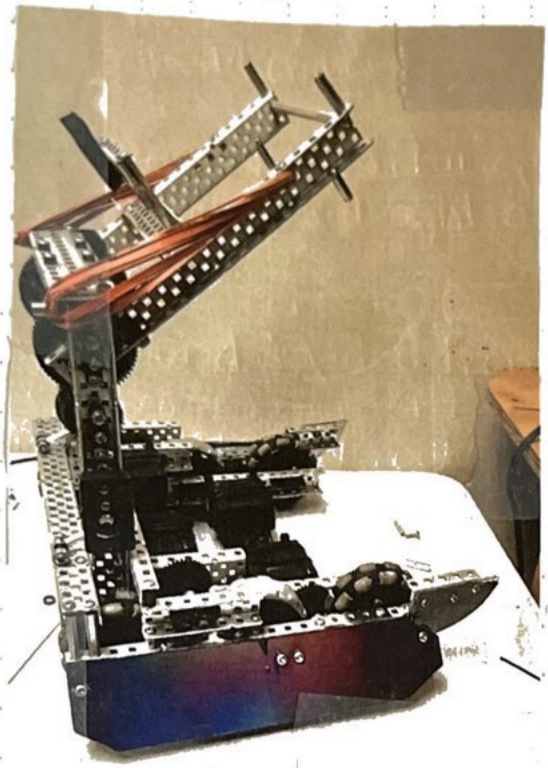
(6) Line up the gearing on the axels. Attach to an arm. Then place in the motor and supports. Gearing and arm diagram located below.

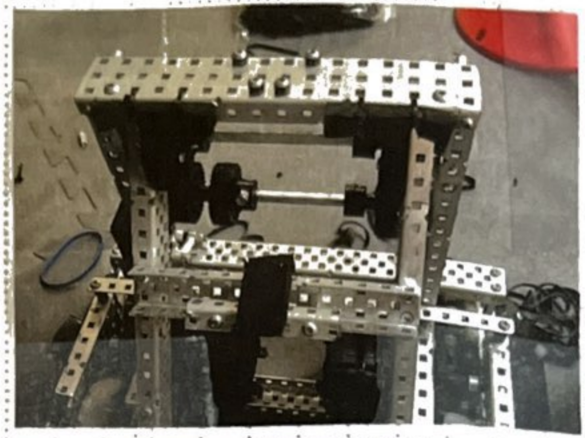
(7) Rubberband together to both standoffs.



(Continued)

Date: 9/6





Above are some more pictures of the catapult up to this point.

The triball can either be launched by being held in between the c-channels and standoffs of the catapult arm or on top of the four (4) spacers on the 1x3 C-channel which is the hybrid puncher aspect of this design.

VP

project: Building Catapult (Continued) designed by: Vanessa Perkins witnessed by: Harris Perkins date: 9/6/2023

Date: 9/8

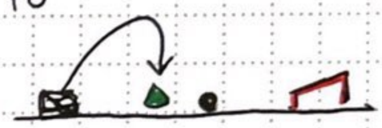
Testing Catapult:

Goals for today → Test the catapult
→ Discover any changes that may need to be done

Testing:

We began trying to launch triballs with both ways of the hybrid catapult and discovered several problems.

1. The catapult tosses the triball too high and doesn't have enough distance to make it over the barrier and towards the goal.



2. The puncher aspect ~~tha~~ VP of the catapult hits the triballs resting on the 1x3 c-channel at the wrong angle so they just go straight towards the ground.

No amount of changing height of triball or length of catapult arm or force of hit could fix this problem without a complete rebuild.

Results:

This catapult design is like pushing someone at shoulder height and trying to make then go up but your push angle knocks them to the ground instead.

VP

Redesign Launch Mechanism:

Date: 9/9

Goals for today → Brainstorm a new launch mechanism to build on our base in place of our current catapult

We began researching ways to make a catapult work to get the height, distance, force, and angle we wanted.

We found a video from the Talon Tussle, an early season tournament. Team 9364C had a robot with a lift that could get the height we needed and have the advantage of being able to shoot over defensive robots.

Our new plan is to build a lift with a new catapult on top.

VP

project: Redesign Launch Mechanism   designed by: Vanessa Perkins   witnessed by: Eli Fritts

date: 9/9/2023

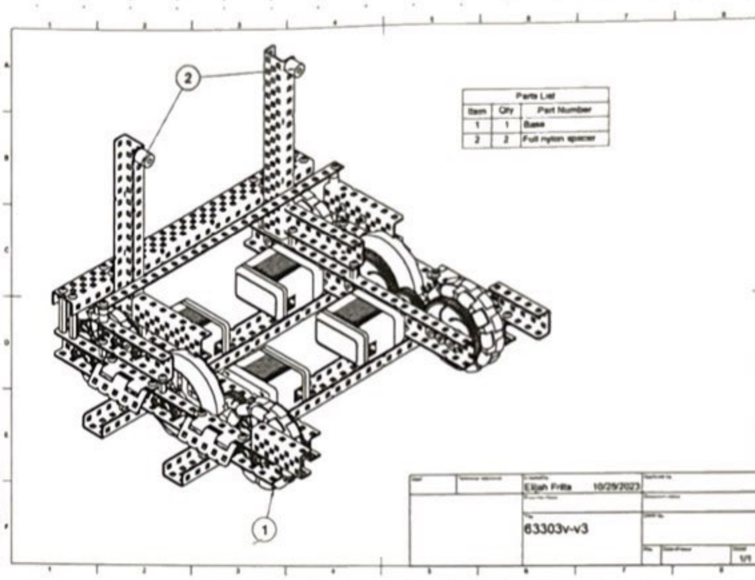
Begin Building Lift:

Date: 9/10

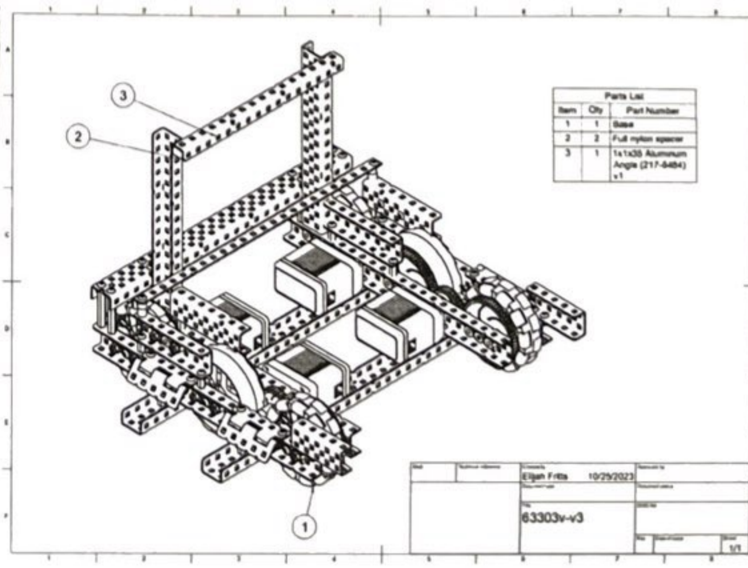
Goals for today → Get the metal added to the base where the lift will attach after disassembling the old catapult.

1. Remove all of the old catapult from the robot except the supports from page 119.

2. Add two (2) 1.000 inch spacers to the top of the uprights, one (1) per side.



3. Gather or cut a sixteen (16) long aluminum 1 by 1 L-channel. Place in the depicted location on top of the 1.000 inch spacers.

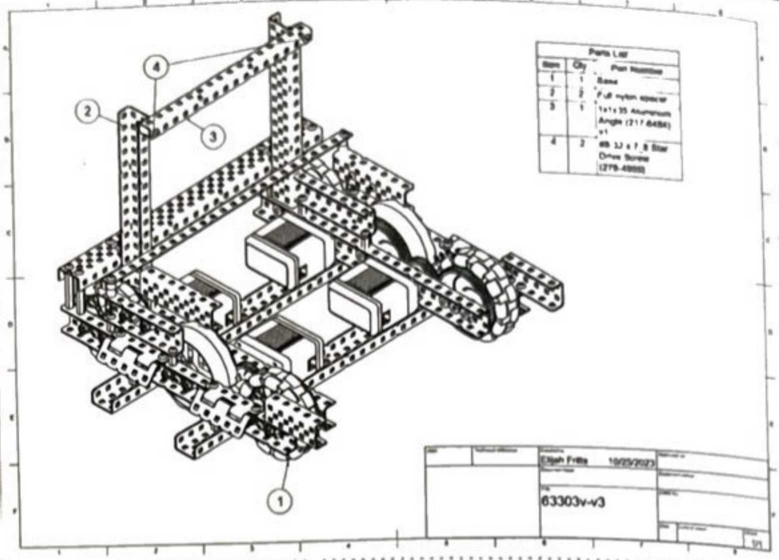


VP

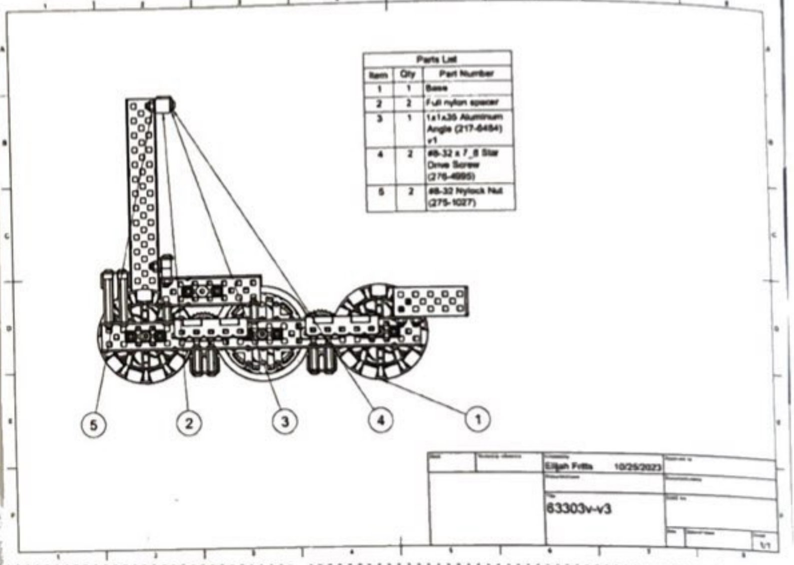
project: Begin Building Lift designed by: Vanessa Perkins & Eli Fritts witnessed by: Harris Perkins date: 9/10/2023

(Continued)

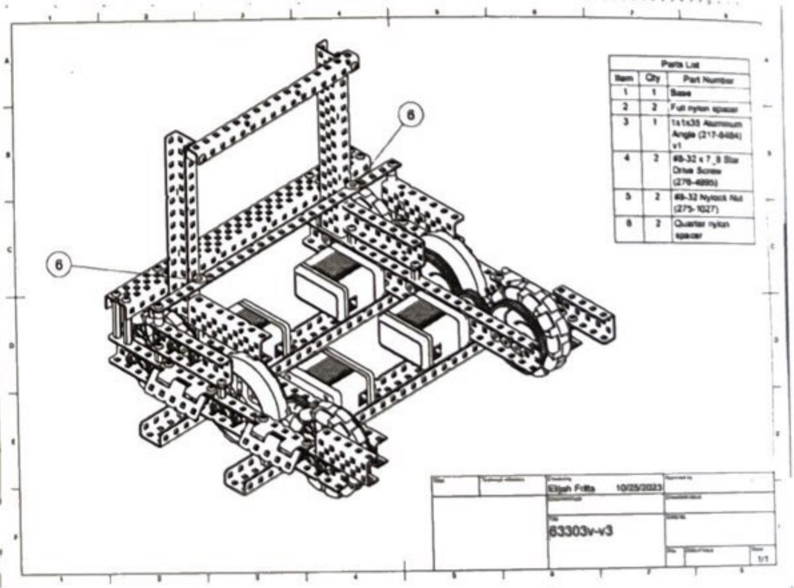
(4) Add two (2) 1.250 inch screws through the ends of the 1 by 1 L-channel into upright supports.



(5) Add two (2) nylock nuts to the screws from step 4 and tighten down.



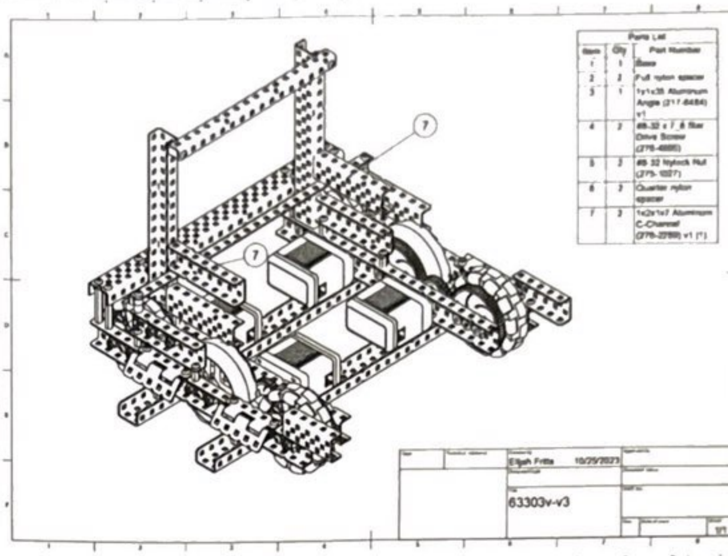
(6) Add a 0.250 inch spacer to each side of the horizontal 1 by 1 L-channel five (5) holes in on each side for a total of two (2) spacers.



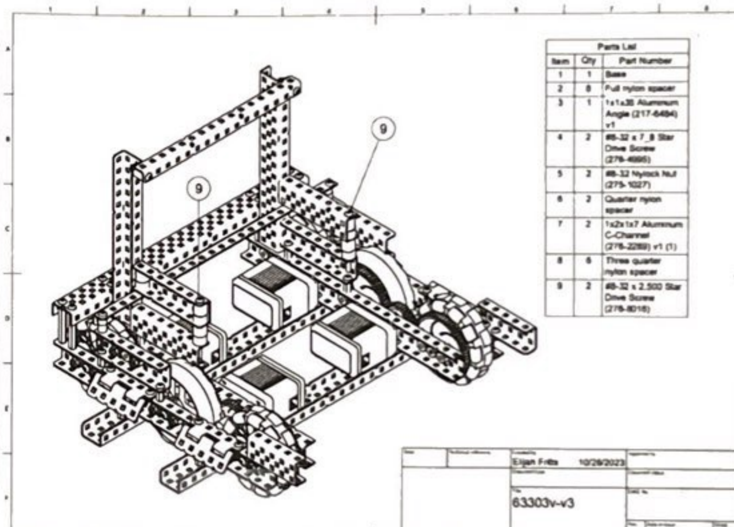
project: Begin Building Lift (Continued)
designed by: Vanessa Perkins & Eli Fritts
witnessed by: Harris Perkins
date: 9/10/2023

(Continued)

7 Gather or cut two (2) seven (7) long aluminum c-channels and place on top of horizontal 1by1 L-channel.

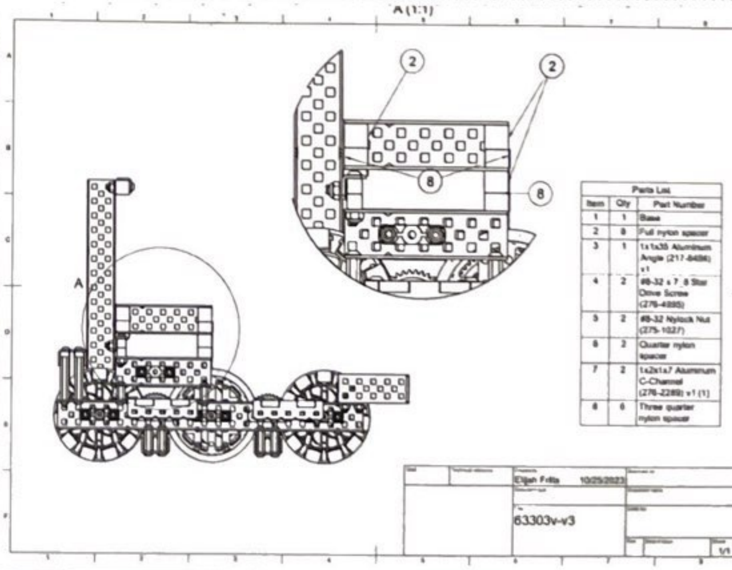


8 Gather ~~eight (8)~~ twelve (12) VP 0.375 inch Spacers and place four (4) on each corner of the seven (7) long c-channel in the front and two (2) each in the back. Below is further zoomed in CAD images of step 8.



~~9~~ VP

VP

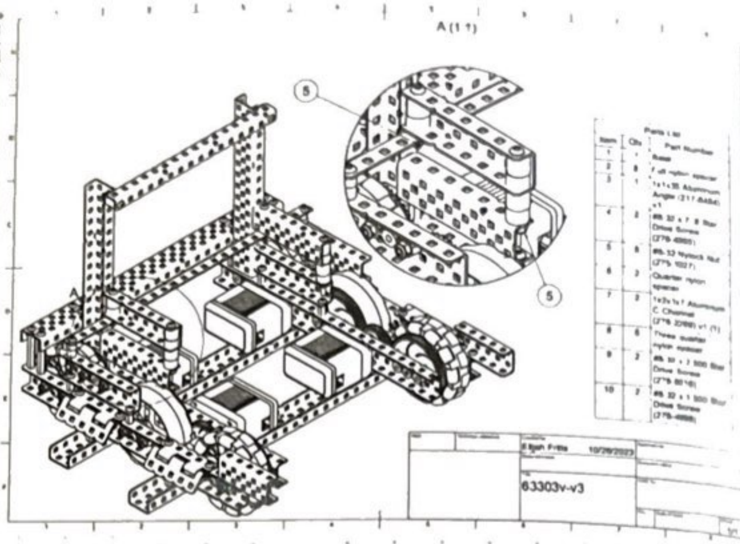


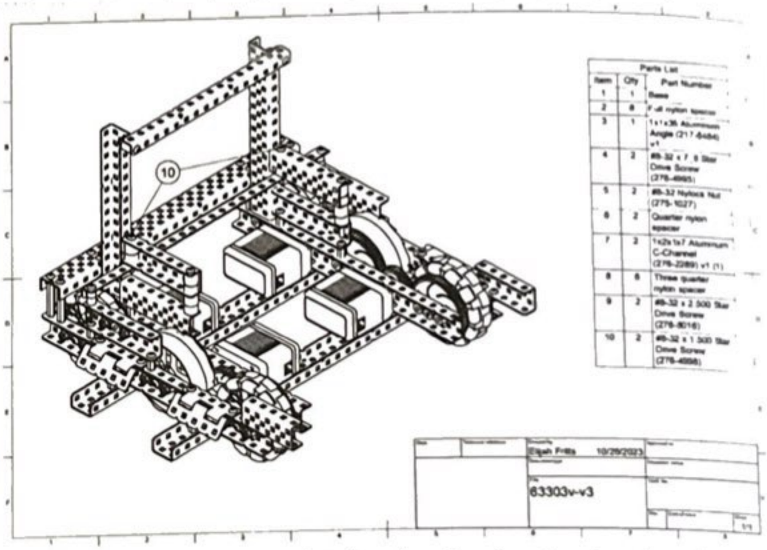
project Begin Building Lift (continued) designed by: Vanessa Perkins + Eli Fritts witnessed by: Owen taylor date: 9/10/2023

(Continued)

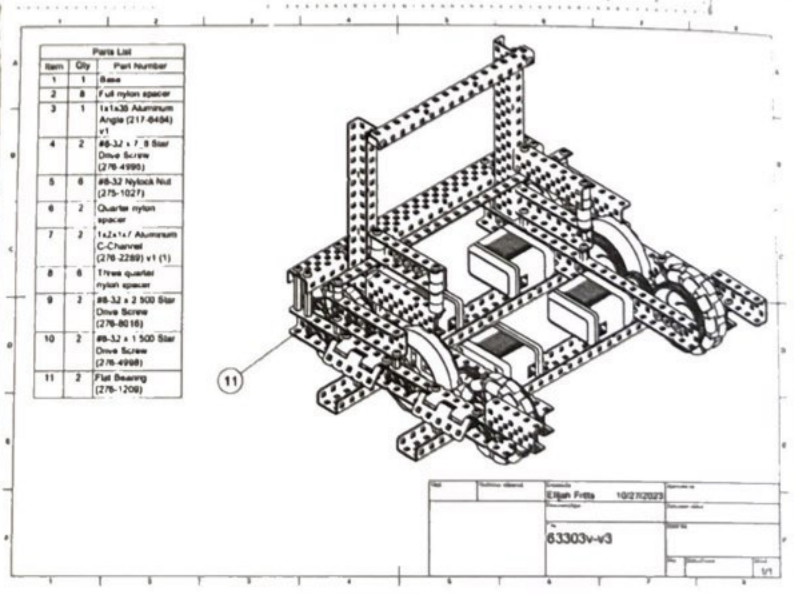
(9) Attach the c-channels and spacers with four (4) 1.500 inch screws. Add four (4) nylock nuts to the ends and tighten.
Below is another CAD image of step 9.

VP





(10) Add two (2) bearing flats to the seven (7) long c-channel in the depicted locations to the right.



Northeast TN VRC Season Kickoff: 63303V Rebuild Schedule

9/12/2023:

Harris: Start working on making a functional Drive PID, and taking notes on it for the EDN.
Owen: Start coming up with 3+ different driving strategies, one for every scenario for each robot (old and new) for EDN
Eli: Make the basis of the lift, with a prototype of it on the robot.
Vanessa: Work on the notebook.

9/13/2023:

Eli: Finalize the prototype, and start building the lift.
Vanessa: Work on the notebook.

9/14/2023:

Harris: Continue working on making a functional Drive PID, and taking notes on it for the EDN.
Owen: Finish coming up with 3+ different driving strategies, one for every scenario for each robot (old and new) for EDN
Eli: Get 50% of the lift done, gearing and ect.
Vanessa: Work on the notebook.

9/15/2023:

Eli: Start building up on the lift, and get it mostly done.
Vanessa: Work on the notebook.

9/16-17/2023:

Eli: Finish the lift and make an attaching point for the catapult, then start building the catapult.

9/18/2023:

Harris: Try to finish working on Drive PID, and learn more about automation for driver control.
Owen: Watch matches from Talon Tussle, configure more strategies against specific teams, and take overall notes for EDN.
Eli: Try to finish the catapult and start tuning it for cross-court shooting.
Vanessa: Work on the notebook.

9/19/2023:

Harris: Start programming overall controls and automation for the robot.
Owen: Continue watching matches from Talon Tussle, configure more strategies against specific teams, and take overall notes for EDN. As well as work with Harris to figure out controls.
Eli: Tune and finalize the catapult. Start evaluating the intake.
Vanessa: Work on the notebook.

project: Rebuild Schedule designed by: Eli Fritts witnessed by: Vanessa Perkins
date: 9/11/2023

9/20/2023:

Eli: Start building the intake structure.
Vanessa: Work on the notebook.

9/21/2023:

Harris: Continue programming overall controls and automation for the robot.
Owen: Make a plan for skills, and the possible routes.
Eli: Continue and almost finish intake.
Vanessa: Work on the notebook.

9/22/2023:

Eli: Continue and test intake.
Vanessa: Work on the notebook.

9/23-24/2023:

Eli: Finish building intake, and start and finish making a PTO for lift to intake.

9/25/2023:

Harris: Finish programming overall controls and automation for the robot.
Owen: Finish making a plan for skills, and the possible routes.
Eli: Test everything together, tune it, and start making wings.
Vanessa: Work on the notebook.

9/26/2023:

Harris: Start making lift automations.
Owen: Brainstorm autonomous ideas.
Eli: Finish making wings, and start making a vertical wing for touching the Loading Zone.
Vanessa: Work on the notebook.

9/27/2023:

Eli: Finish the vertical wing, and test the overall robot.
Vanessa: Work on the notebook.

9/28/2023:

Harris: Brainstorm autonomous ideas.
Owen: Test the robot, and make insightful critiques.
Eli: Make critiques and start working on the hang mechanism
Vanessa: Work on the notebook.

9/29/2023:

Eli: Continue working on the hang mechanism
Vanessa: Work on the notebook.

project: Rebuild Schedule (Continued) designed by: Eli Fritts witnessed by: Vanessa Perkins
date: 9/11/2023

Engineering Notebook

63303V
Team Number

Validation
Team Name

Johnson County High School
School

9/12/2023 | 1/17/2024 | 2 of
Start Date | End Date | Book #

VEX
ROBOTICS

REC
FOUNDATION

Table of Contents

Table of Contents

Table of Contents

Iteration 3

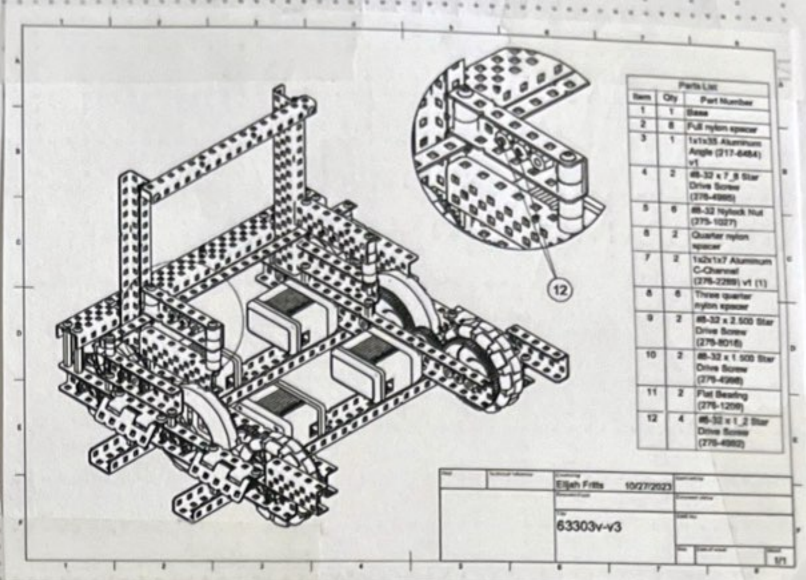
Begin Building Lift Continued:

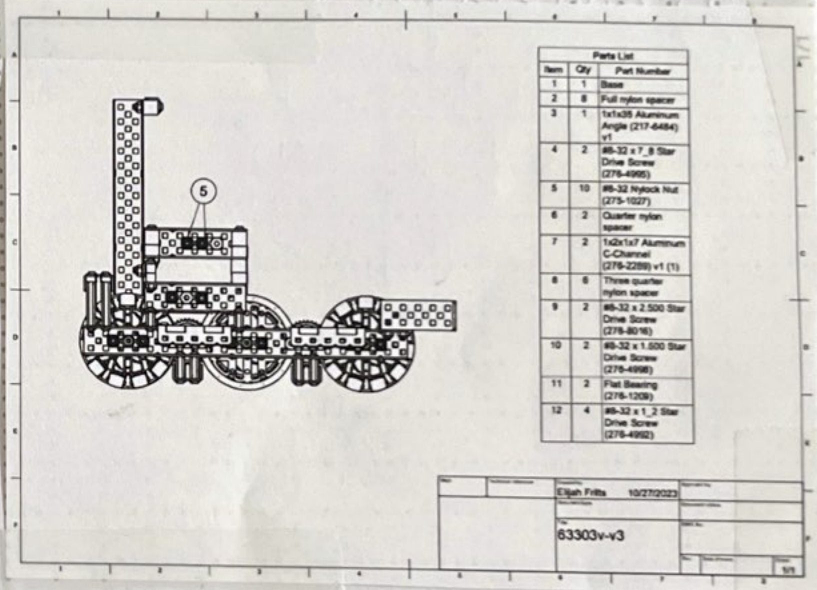
11 Add four (4) 0.500 inch screws to the two (2) bearing flats from step 12.

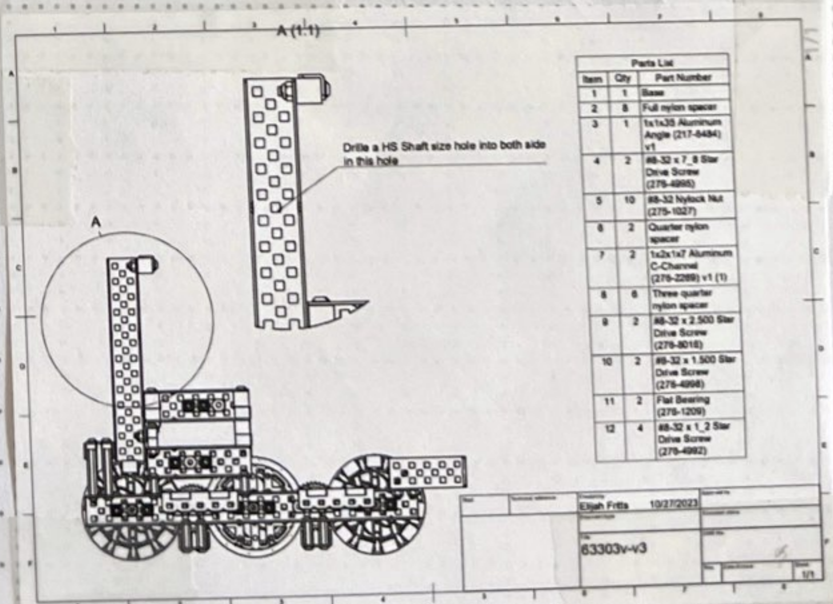
12 Add two (2) nylock nuts per side to the screws to tighten down bearing flats.

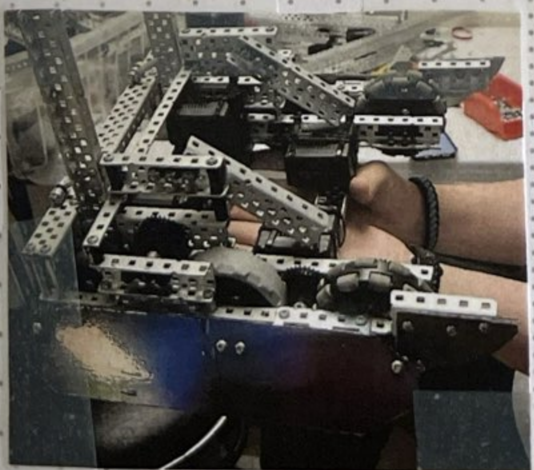
13 Drill out holes in the upright C-channels for the high strength shafts.

Below is an image of the robot up to this current point in time.



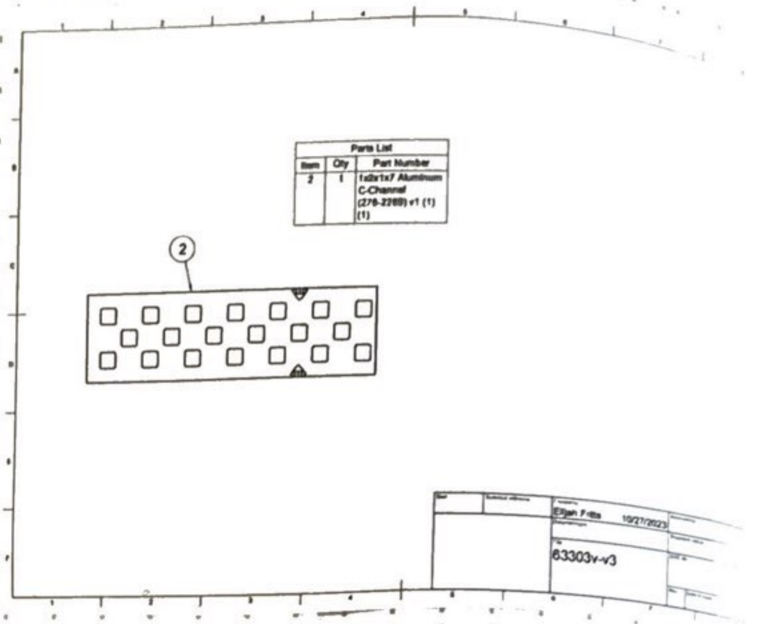




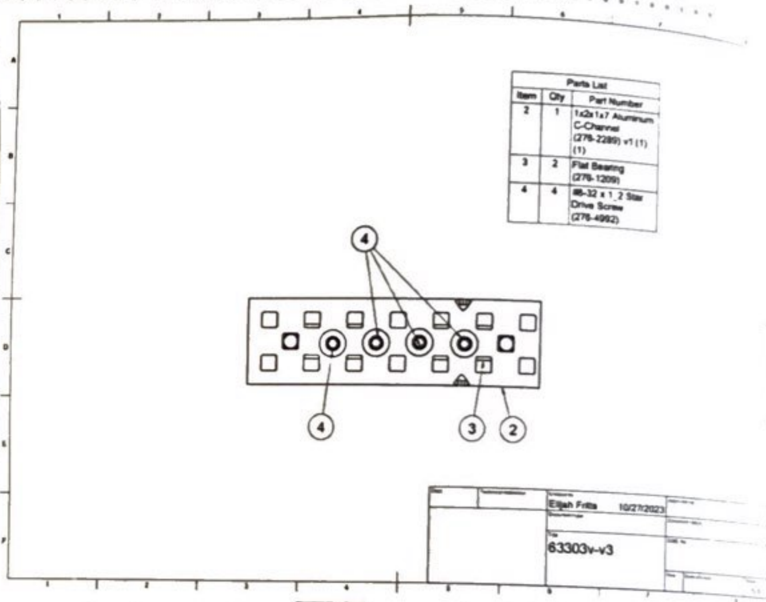


Assemble Lift Bars:

Goals for today → Get all the c-channels ready to be attached together

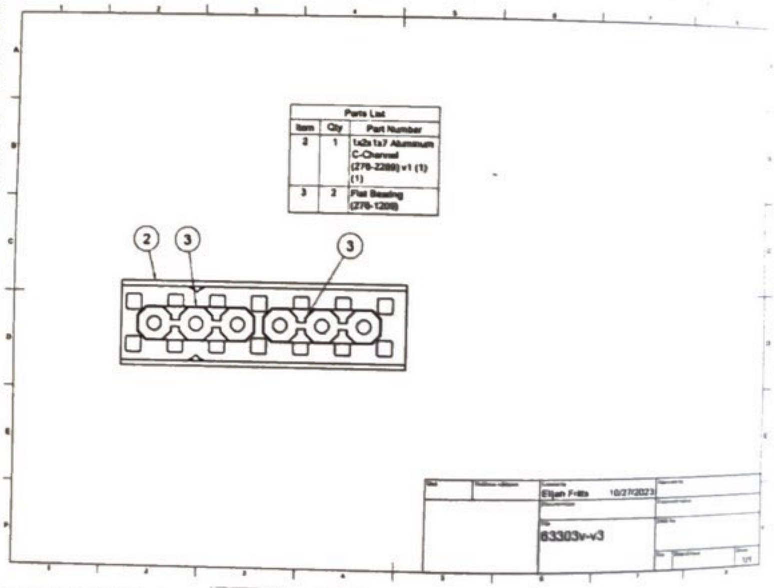


① Gather or cut a seven (7) long aluminum c-channel.



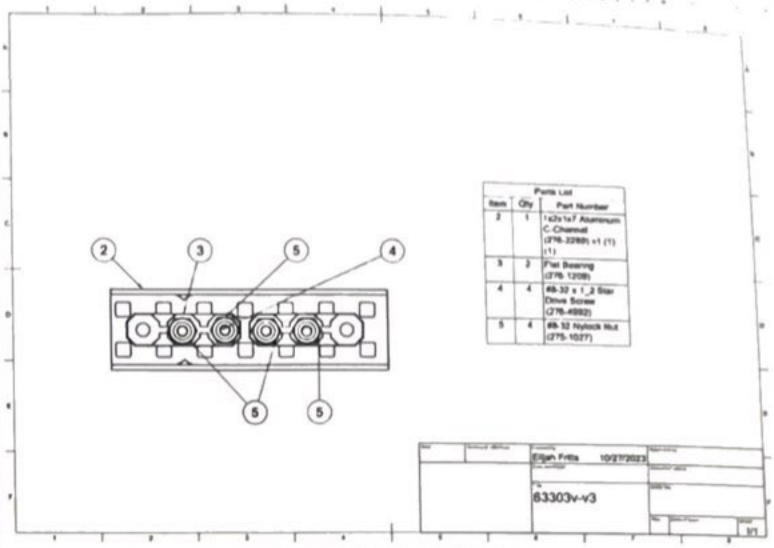
② Add two (2) bearing flats and two (2) 0.500 inch screws through each for a total of four (4) screws, leaving the outside holes open.

VP

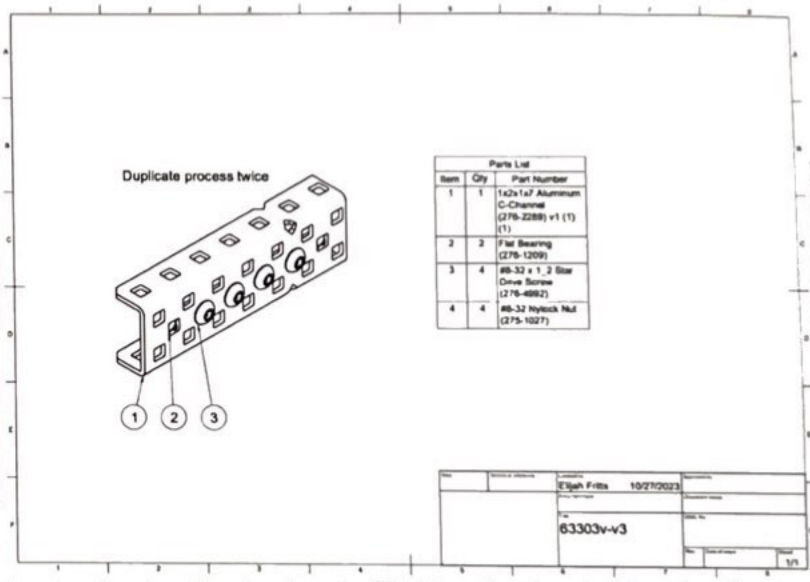


(continued)

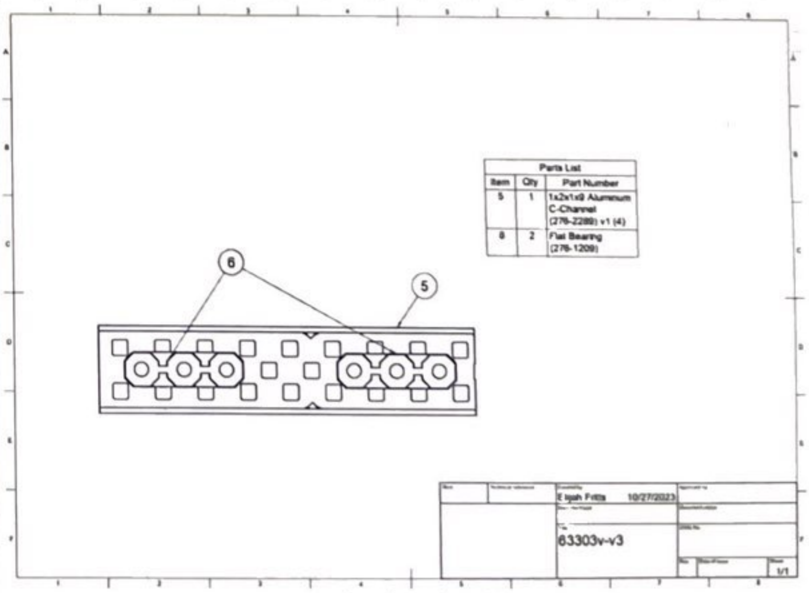
③ Add four (4) nylock nuts to the screws from step 2 and tighten.



④ Repeat steps 1-3 for a second c-channel with bearing flats attached.



⑤ Gather or cut a nine (9) long aluminum c-channel. Add two (2) bearing flats to the depicted locations to the right.

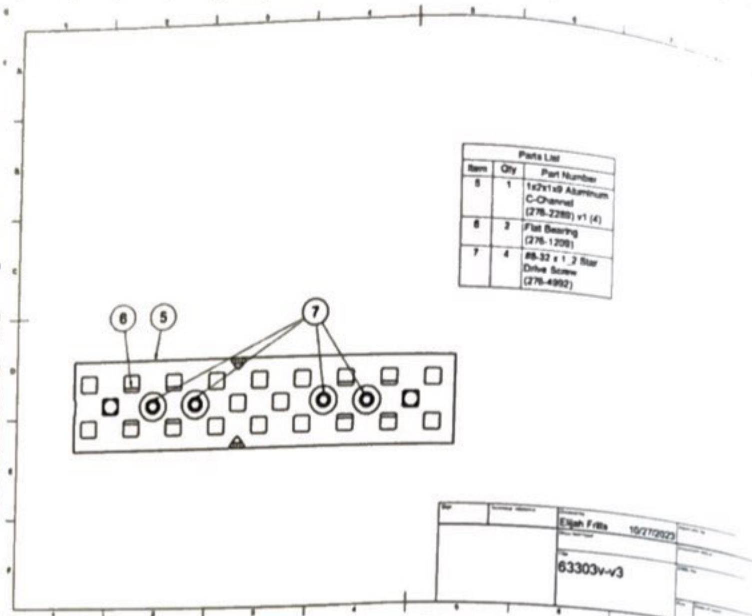


Project Assemble Lift Bars (continued) Name Vanessa Perkins & Eli Fritts
Date 9/12/2023

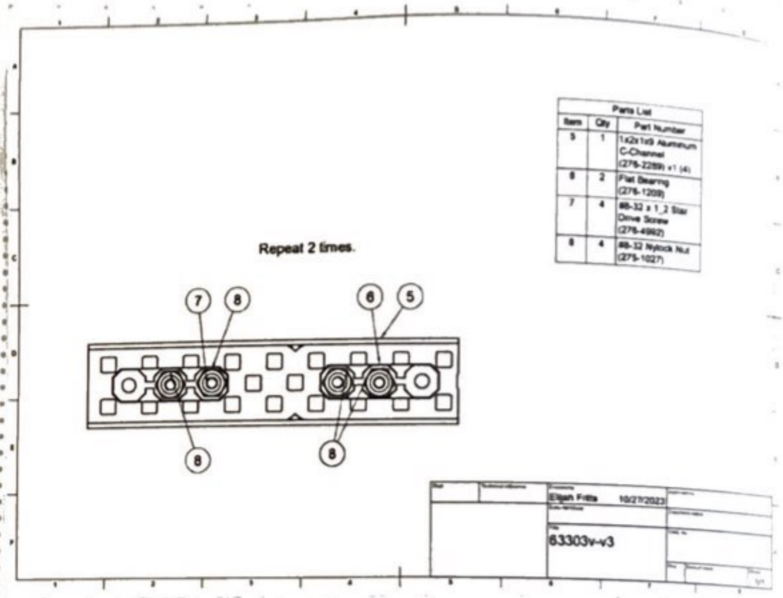
Iteration 3
Iteration 4

(Continued)

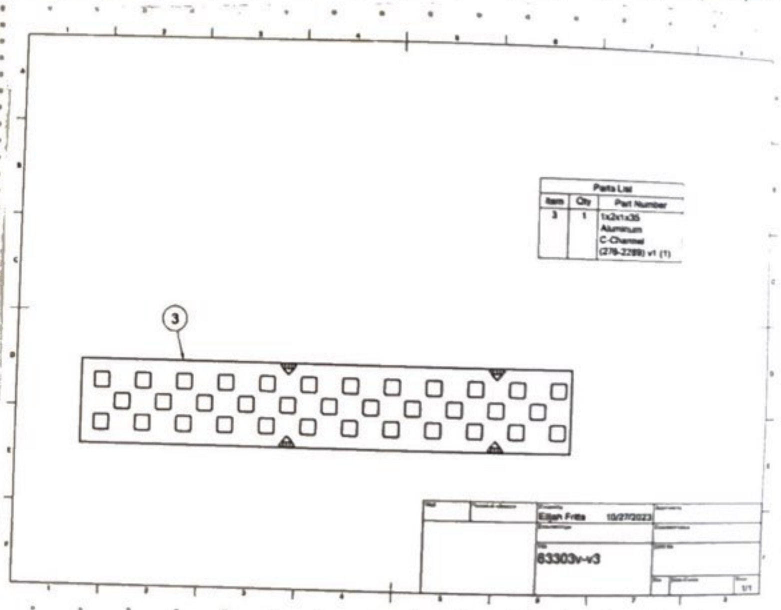
(6) Add four(4) 0.500 inch screws to the bearing flats. Leave outside holes open.



(7) Repeat steps 5-6 again for two(2) c-channels with bearing flats and screws.
Add eight(8) nylock nuts to both c-channels.



(8) Gather or cut a twelve(12) long aluminum c-channel. (Depicted below)

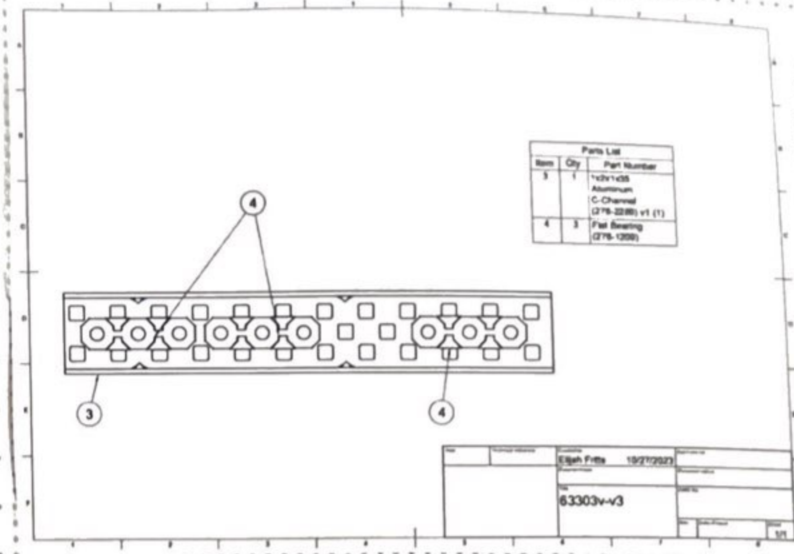


Iteration 3

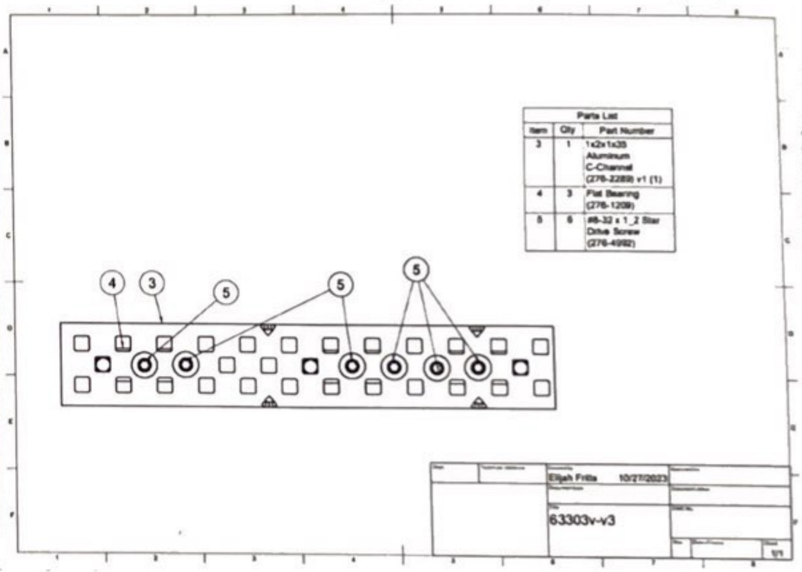
Iteration 4

(continued)

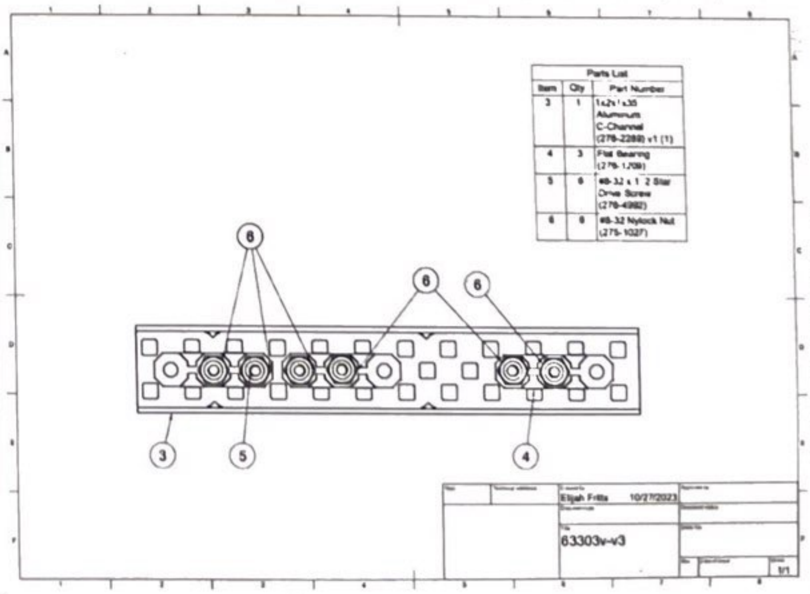
⑨ Add three (3) bearing flats to the c-channel in the depicted locations.



⑩ Add six (6) 0.500 inch screws to the bearing flats.



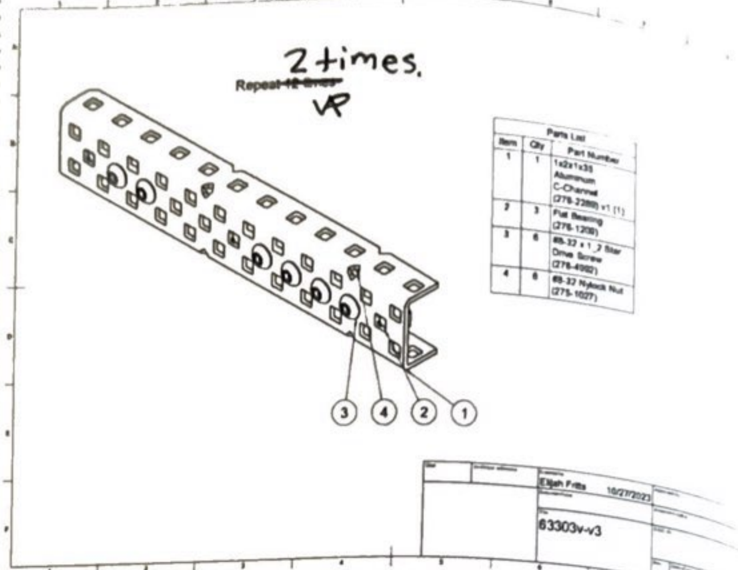
⑪ Add six (6) nylock nuts to the screws from step 10 and tighten down.



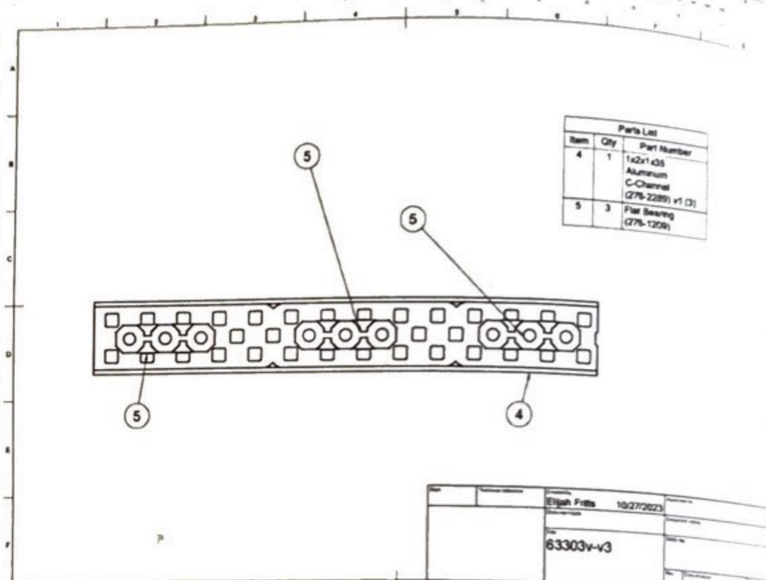
VP

(Continued)

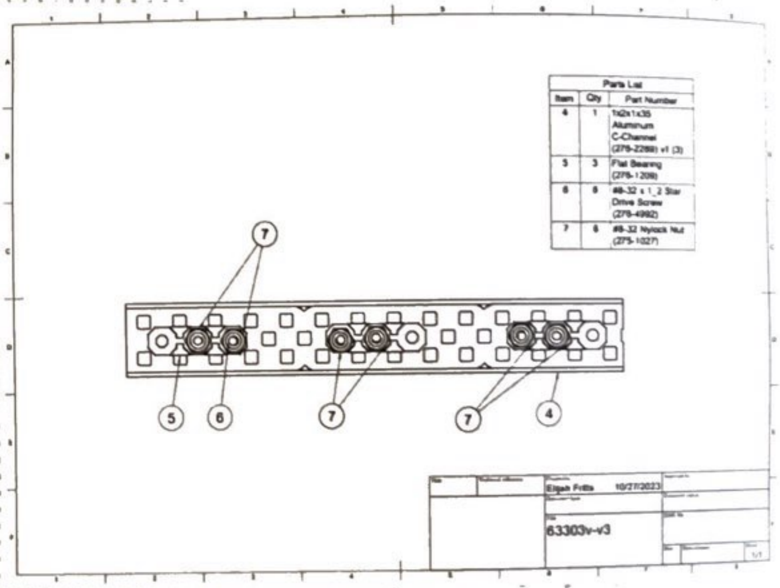
(12) Repeat steps 8-11 a second time for two (2) twelve (12) long c-channels with bearing flats attached.



(13) Gather or cut a fourteen (14) long aluminum c-channel. Add three (3) bearing flats in the depicted locations to the right.



(14) Add six (6) 0.500 inch screws and six (6) nylock nuts to the bearing flats and tighten.
Repeat steps 13-14 again for a total of two (2) c-channels with bearing flats attached.



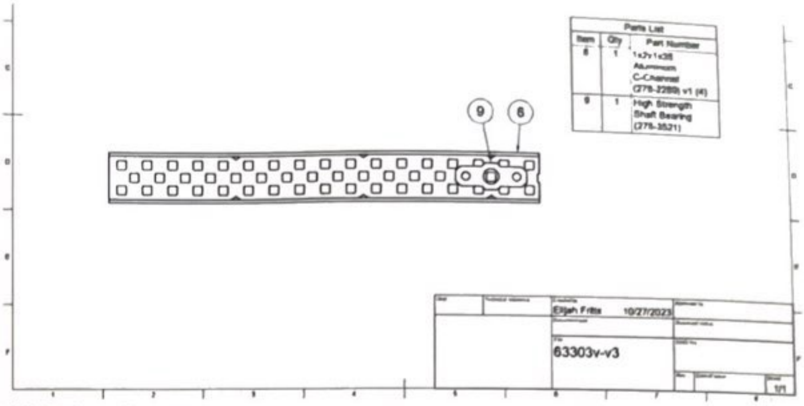
(continued)

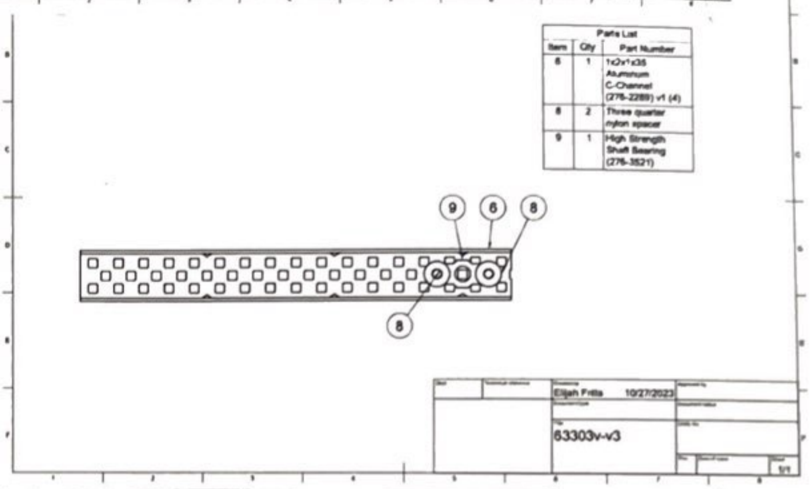
(15) Gather or cut a seventeen(17) long aluminum c-channel. Add a high strength bearing flat to the end.

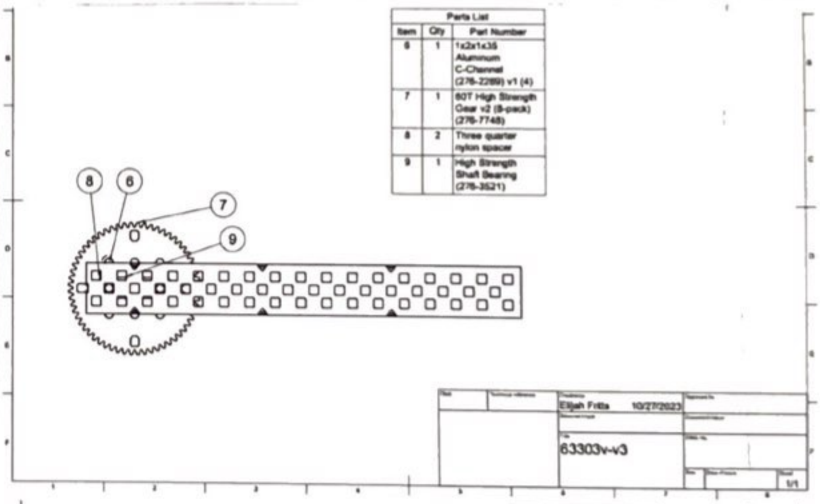
(16) Add two(2) 0.375 inch spacers to either side of the bearing flat.

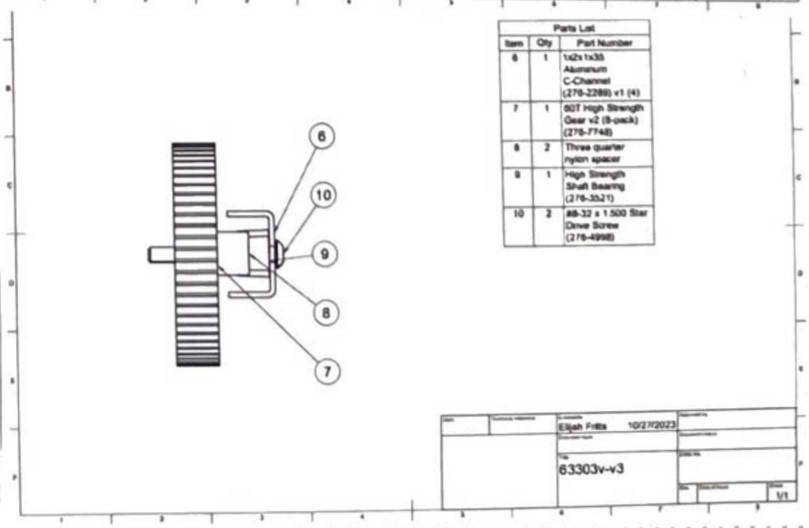
(17) Add a 60 tooth gear to the c-channel on top of the spacers from step 16.

(18) Add ~~of~~ two(2) 1.500 inch screws through the c-channel, bearing flat, spacers, and into the gear to keep them both rotating together with the shaft.







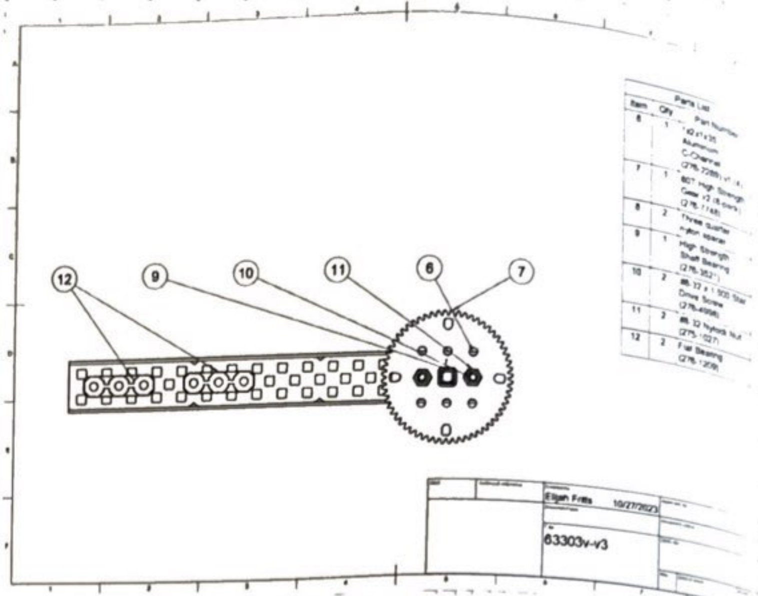


Project Assemble Lift Bars (continued) Name Vanessa Perkins + Eli Fritts

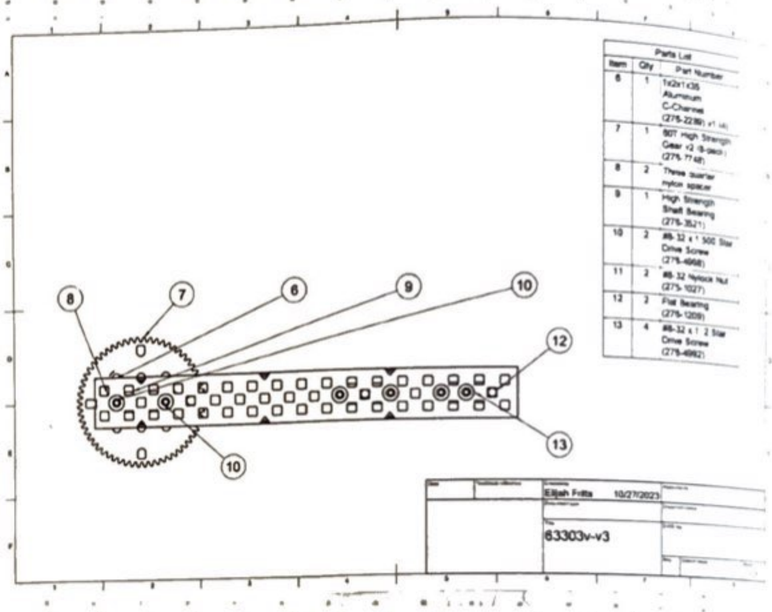
Date 9/12/2023

(Continued)

(19) Add two (2) nylock nuts to the screws from step 18. Add two (2) bearing flats in the depicted locations.



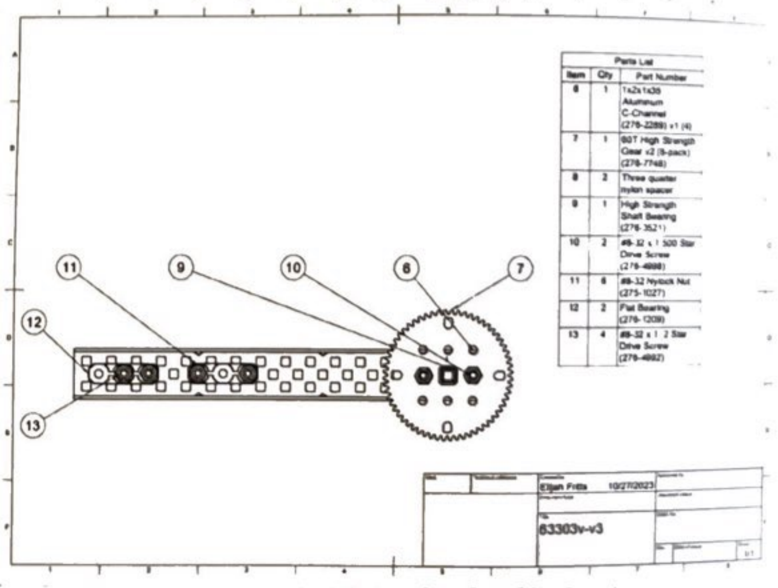
(20) Put four (4) 0.500 inch screws into the bearing flats in the depicted locations to the right.



(21) Add four (4) nylock nuts to the screws and tighten the bearing flats down.

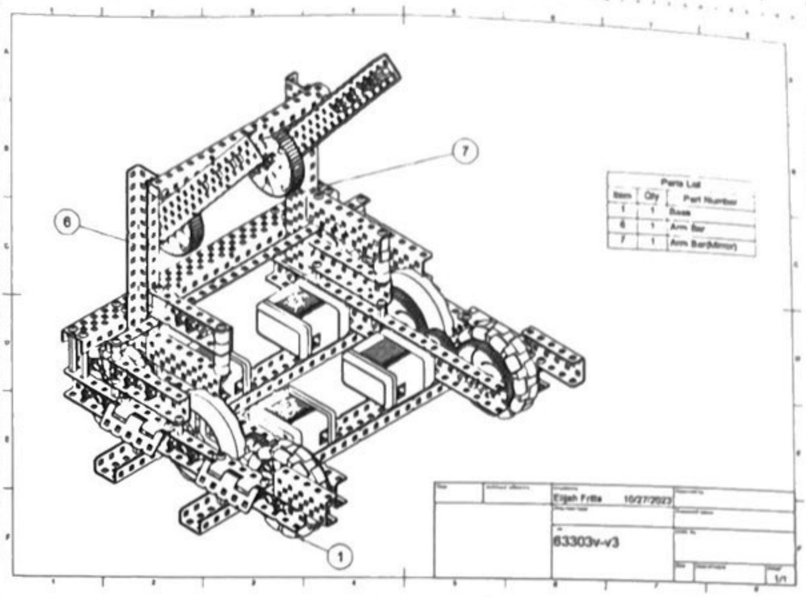
Repeat steps 15-21 again for two (2) c-channels.

~~(22) Repeats step 1-14 again to build the second half of the lift.~~ VP

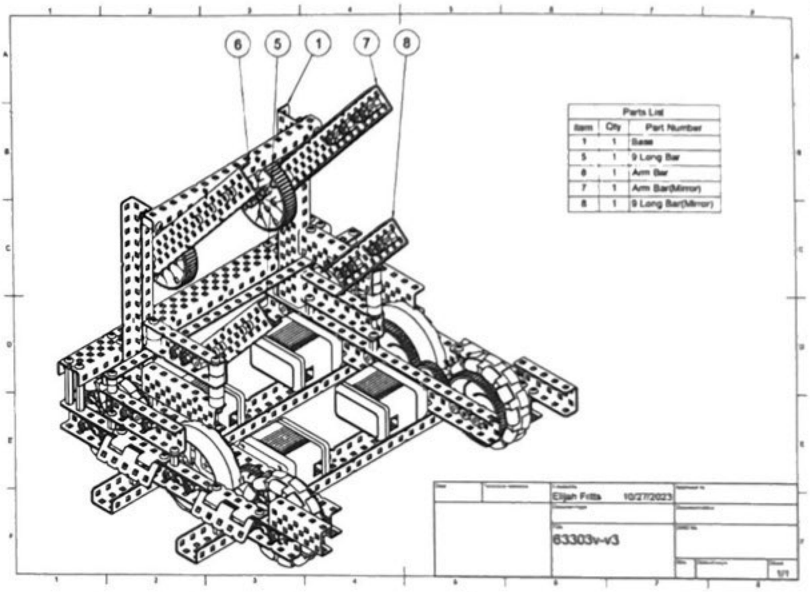


Attach Lift:

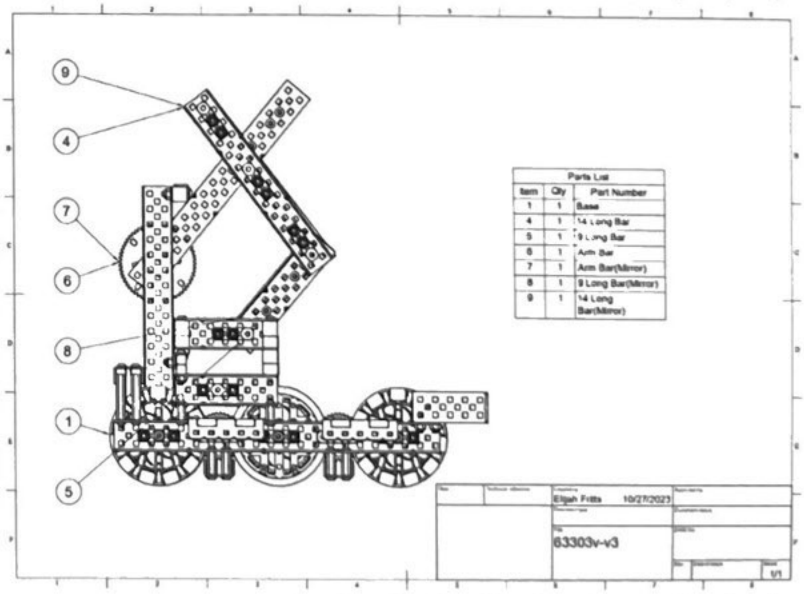
Goals for today → Get Lift fully attached to robot and working



① Put both arm bars in the depicted locations to the right where the high strength holes were drilled. Add nine (9) inch axel through supports + arms.

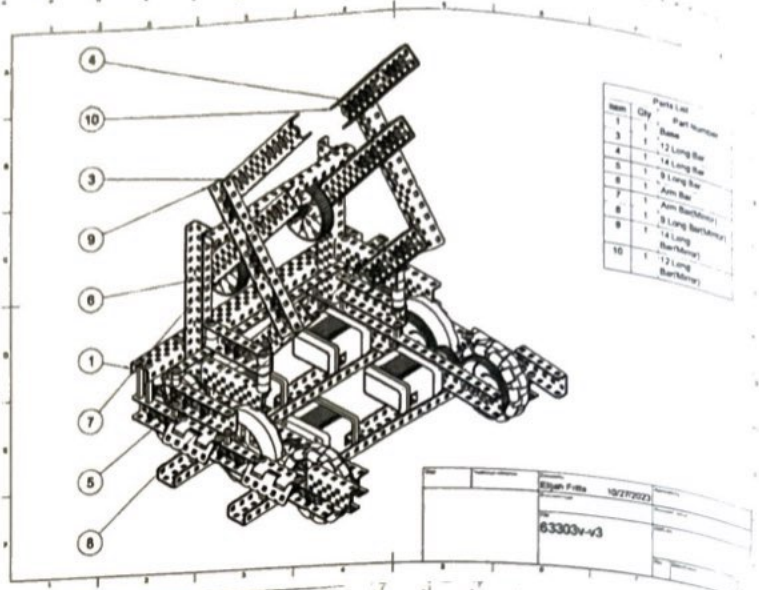


② Put the two(2) nine(9) long c-channels in place and attach with two(2) loose 1.000 inch screws and nylock nuts.

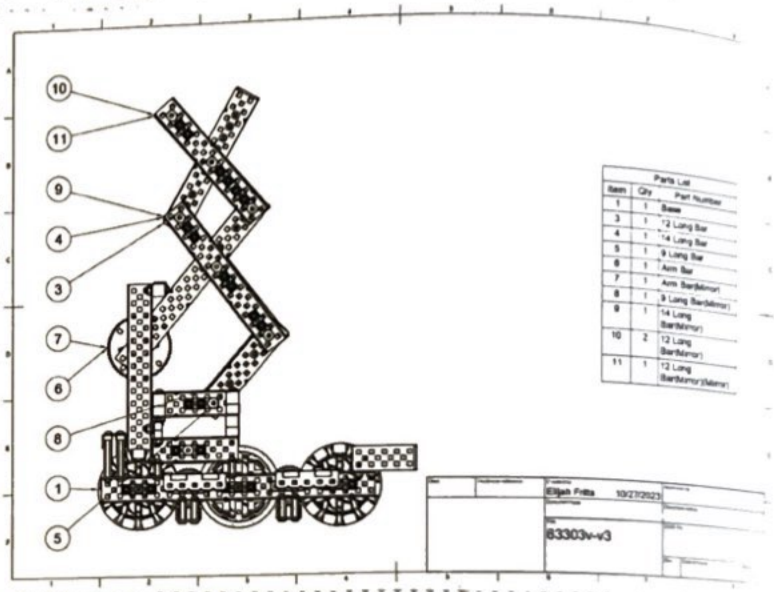


③ Add the two(2) fourteen(14) long bars to their place. Attach with two(2) loose 1.000 inch screws and nuts each in the bearing flat holes.

(Continued)



(4) Add the two(2) twelve(12) long bars to the top using two(2) loose 1.000 inch screws.



(5) Add two(2) twelve(12) long bars to the top and attach with two(2) loose 1.000 inch screws each.

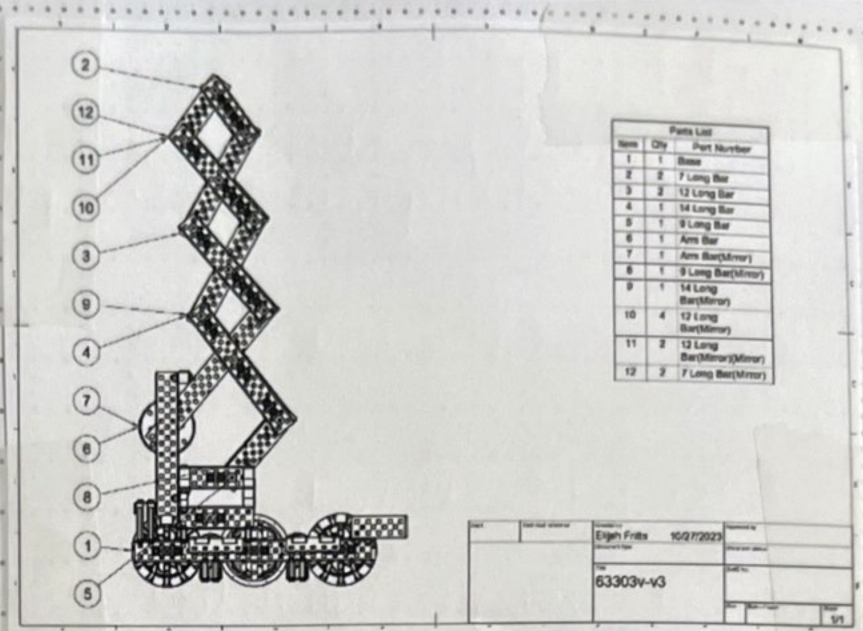
~~(6) Add~~

VP

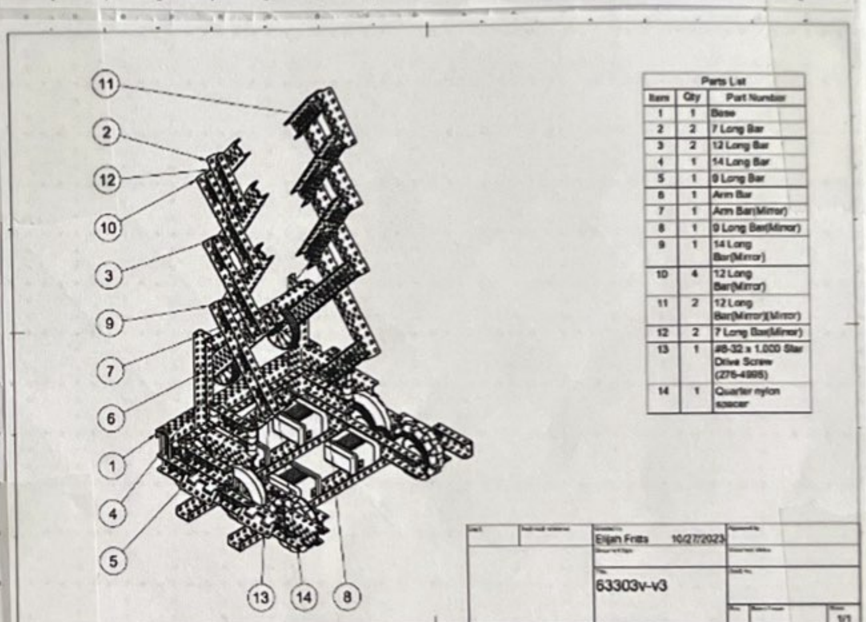
Project: Assemble Lift Bars (Continued)
Name: Vanessa Perkins & Eli Fritts
Date: 9/13/2023

(continued)

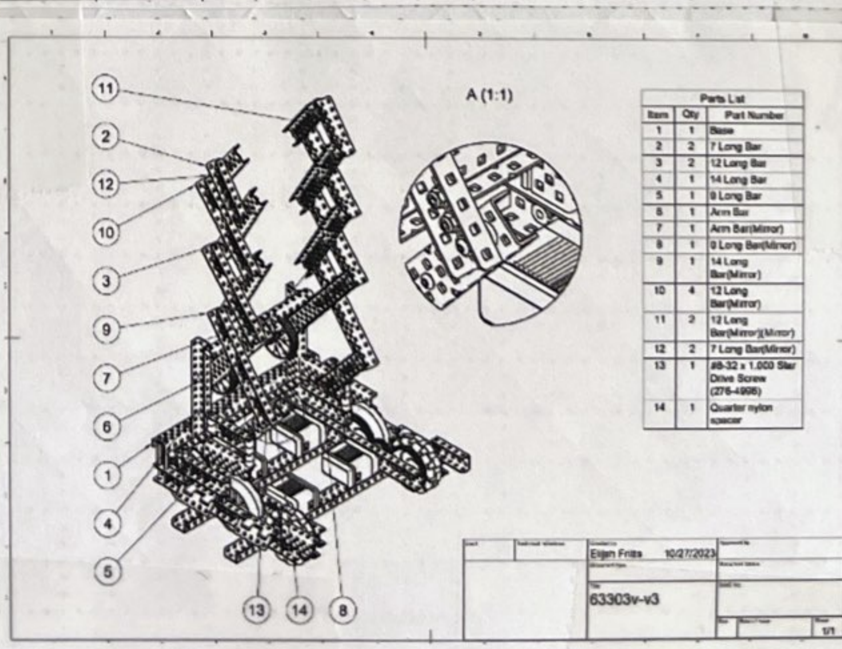
7. Add two (2) seven (7) long bars to the top on both sides diagionally using three (3) loose 1,000 inch screws each.



8. Make sure everything is attached correctly.



9. Add 36 tooth gear below 60 tooth gear. Attach motor to the 36 tooth gear.
Add support 1 by 1 L-channel bars in between the two sides of the lift.



Project Assemble Lift Bars (Continued) Name Vanessa Perkins + Eli Fritts
Date 9/13/2023

Test Lift:

Goals for today → Test Lift to make sure it works and is strong enough for our plans.

We tested the lift today and found the following things:

1. The lift works well moving up and down.
2. Inside 18 inches by 18 inches.
3. Not strong enough to lift the robot weight up for us to be able to hang.

VP

Project Test Lift

Date 9/18/2023

Name Vanessa Perkins

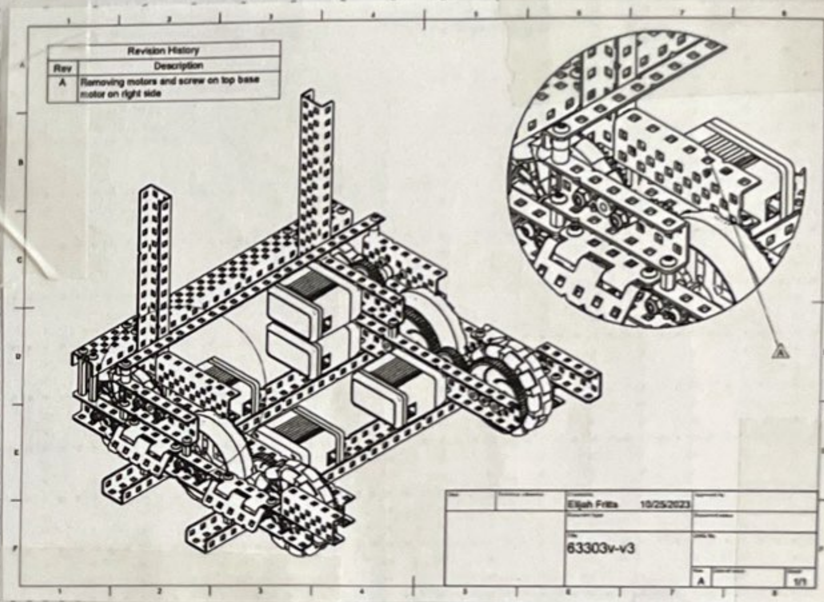
Lift Solution/Improvement:

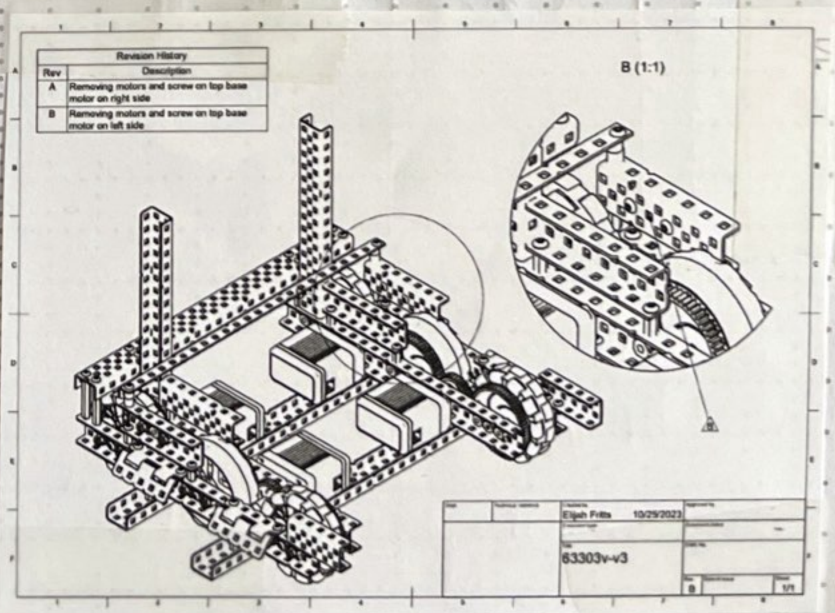
Goals for today → Brainstorm ways to make the lift stronger

To make our lift stronger for launching matchloads and hanging at the end of matches, we decided to take two (2) motors off the base and add one (1) to the other side of the lift.

Below are the CAD instructs for doing so.

VP

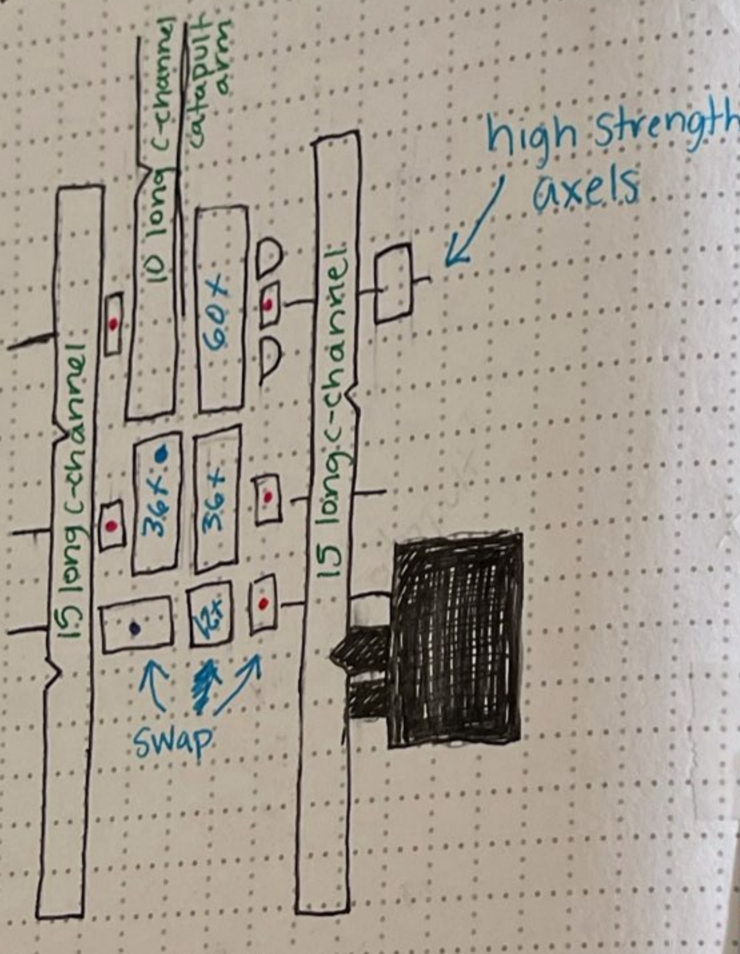


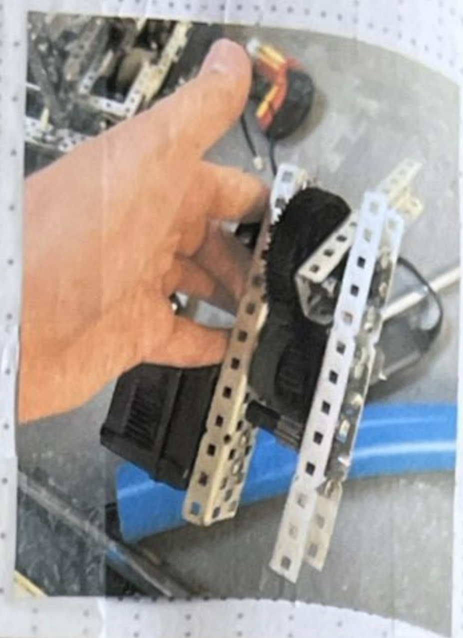


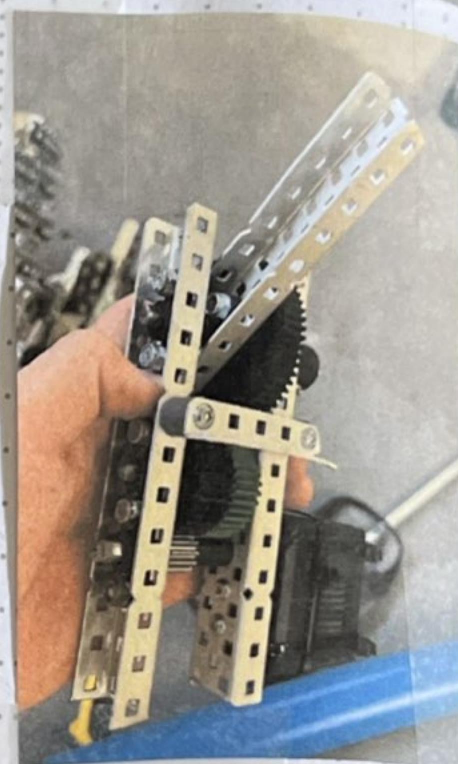
Building Mini Raised Catapult:

Goals for today → Get catapult built and slip gear tuned
→ Attach to lift

Today we began building a mini catapult to go on top of our lift. Below is a simple sketch of it with pictures to the right before we attached it to the robot.







• – collarlock
• – 0.125 inch spacer
• – slip gear
• – 0.500 inch spacer

The catapult arm is banded so it resets after the gear slips.

Project Building Mini Raised Catapult
Name Vanessa Perkins
Date 9/19/2023

(Continued)

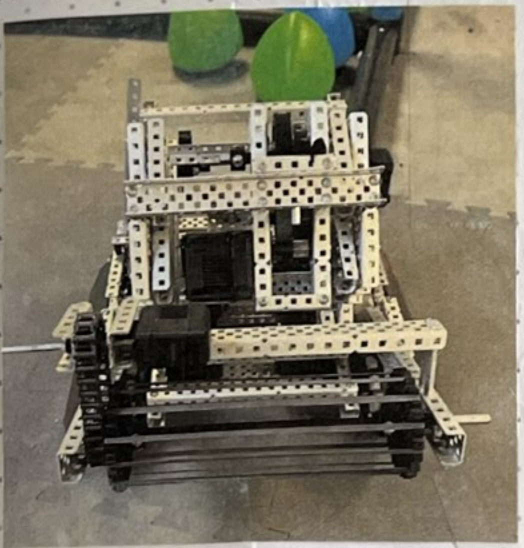
To the right is a complete parts list of the mini catapult.

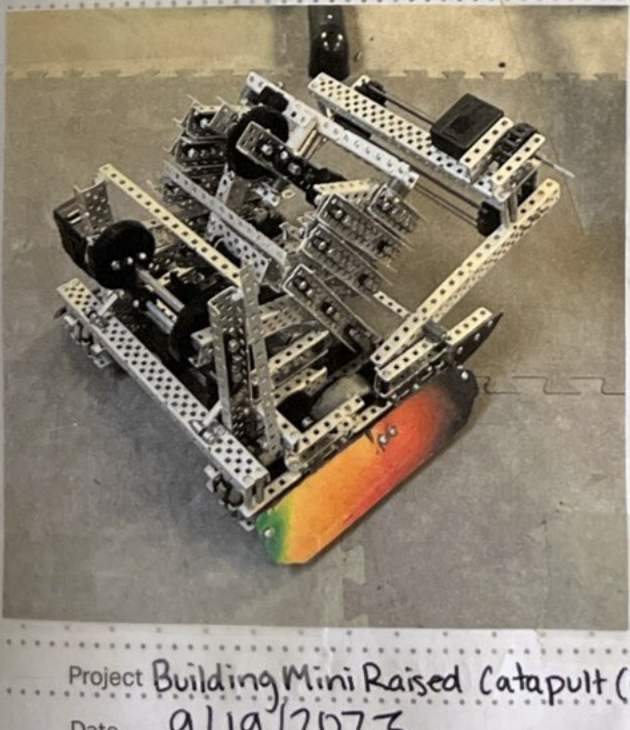
Below are the images of the finished catapult attached to the top of the lift to help us launch over defensive robots trying to block us.

Mini Catapult Parts List:

- 2 - 13 long aluminum c-channels
- 1 - 10 long aluminum c-channel
- 1 - 5 long aluminum 1by1 L-channel
- 1 - 60 tooth gear
- 2 - 36 tooth gears
- 1 - 12 tooth gear
- 6 - high strength bearing flats
- 1 motor
- 2 - 0.375 inch spacers
- 1 - high strength Collar Lock
- 3 - 3 inch high strength shafts
- 20 - 0.500 inch screws
- 18 - Nylock nuts
- 5 - 0.125 high strength spacers
- 1 - 0.500 inch high strength spacer

VP





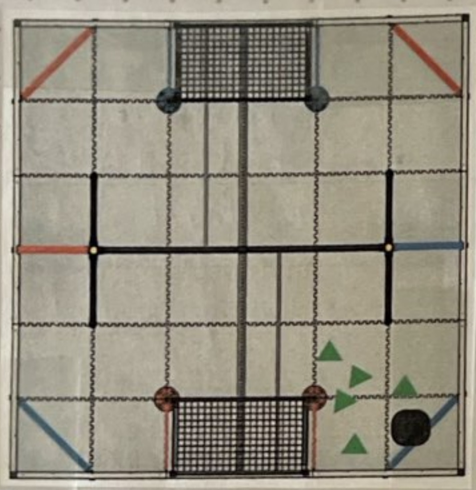
Test Catapult:

Goals for today → Run tests on the mini catapult with lift to make sure it can launch triballs over the barrier from the matchload corner

Purpose: Test catapult for function to make sure it has enough force and the right angle to make it across the field and over the barrier.

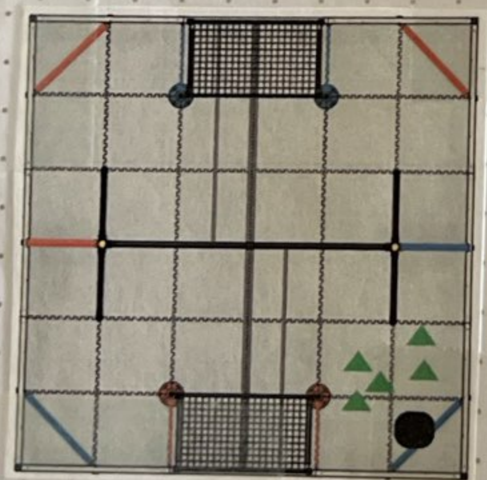
Apparatus: We set up the robot touching the matchload zone and angled towards the blue alliance goal. Each test we launched five(5) triballs and charted their landing locations.

Independent Variable: Number of Rubberbands



Test 1: Two(2) Rubberbands

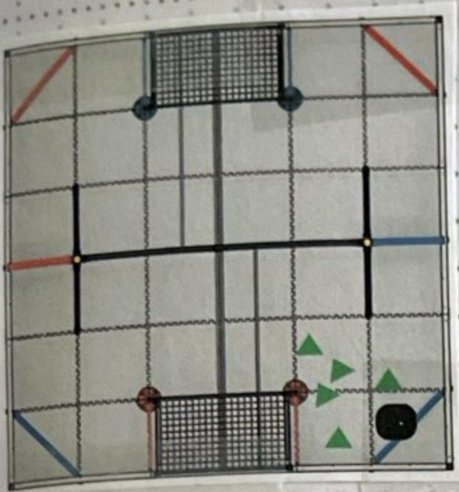
Results: Very in-consistent launching and mostly went straight down after leaving the robot.



Test 2: Four (4) Rubberbands

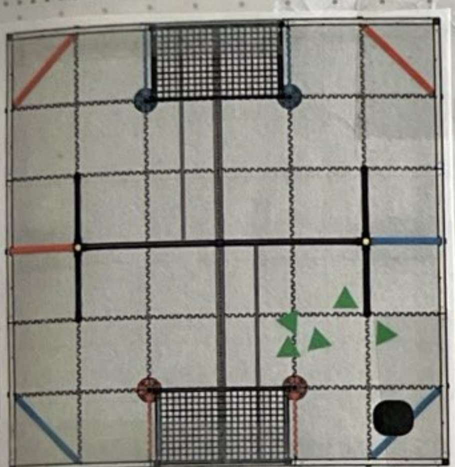
Results: Same results ~~of~~ as test number 1.

(Continued)



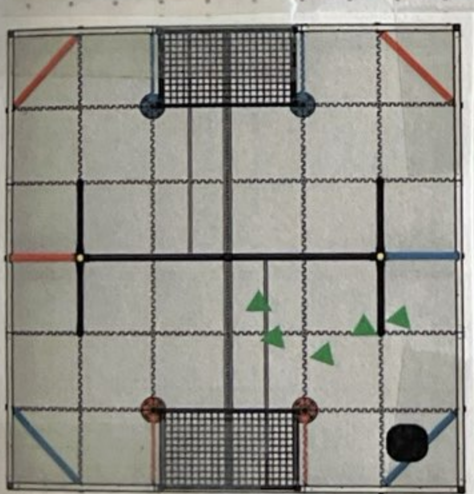
Test 3: Six (6) Rubberbands

Results: The increase of rubberbands is beginning to slow the catapult down as it tries to pull back.



Test 4: Eight (8) Rubberbands

Results: The triballs made it close to the barrier but still not over at all.



Test 5: Ten (10) Rubberbands

Results: Still not strong enough to make it over the barrier.

The tests had a zero percent success rate. The catapult is too hard to tune and require much more time to make efficient.

Brainstorming New Launch Mechanism:

Goals for today → Find a solution for our catapult
→ Or come up with new launch mechanism design

Problem/Challenge

With the confined space on the lift, we can't get the correct angle of launch on the catapult to make it over the barrier.

Brainstorm

1. Flywheel
2. Change the Lift
3. Puncher

Select Option

3

Build Puncher because the angle is perfect for the constraints of the lift.

Build/Implement Solution

- Build slip gear and linear slide puncher
- Attach to lift

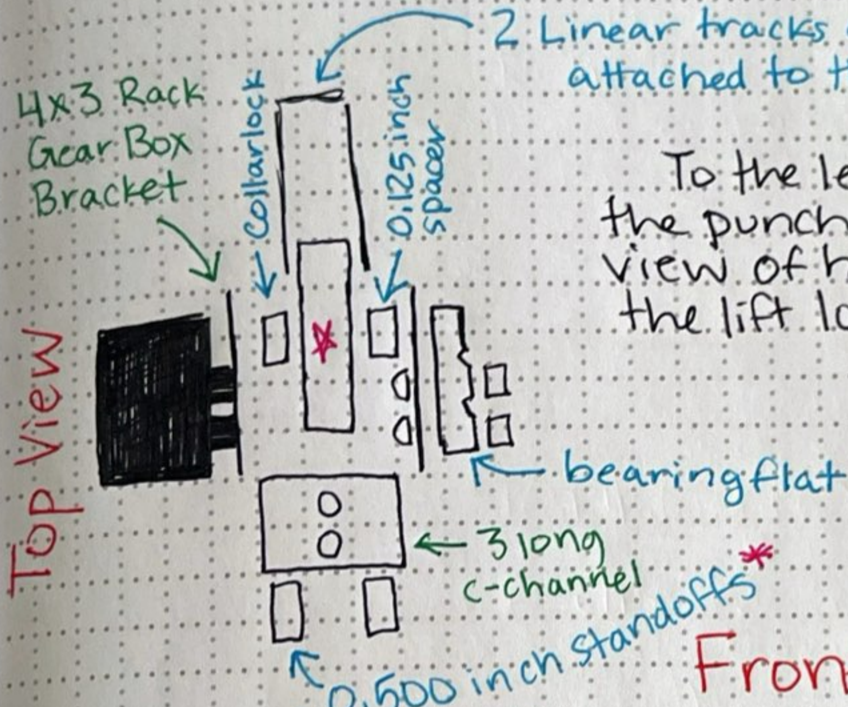
The build is located on page 19 and 20.

VP

Project Brainstorming New Launch Mechanism Name Vanessa Perkins
Date 9/21/2023

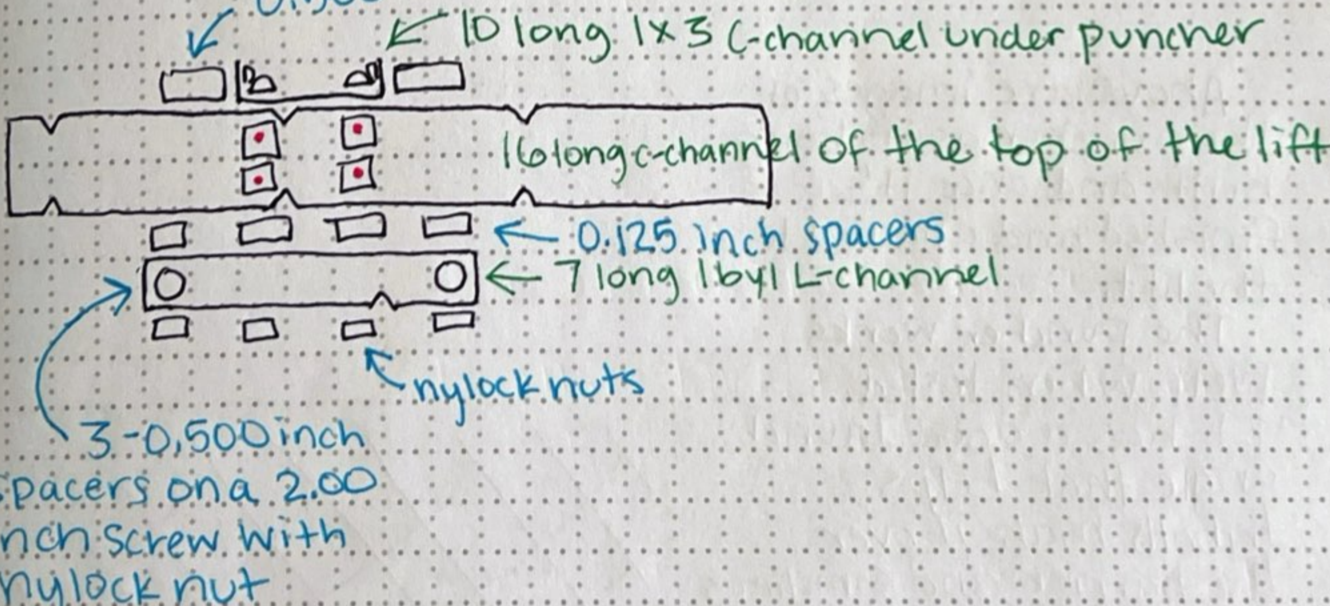
Build Puncher:

Goals for today → Build puncher
→ Attach to lift in place of mini catapult



To the left is a top view of the puncher with the front view of how it attaches to the lift located below.

Front View

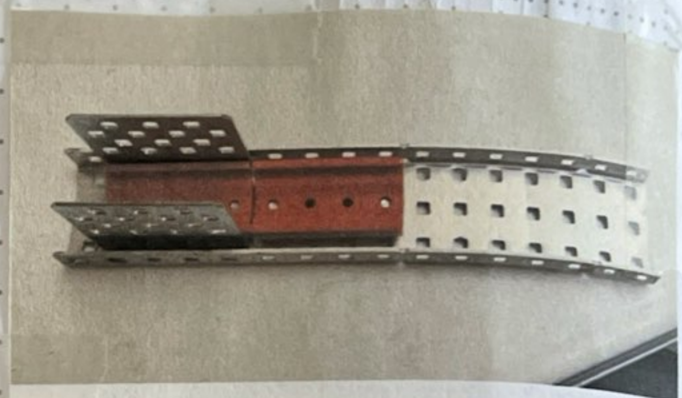


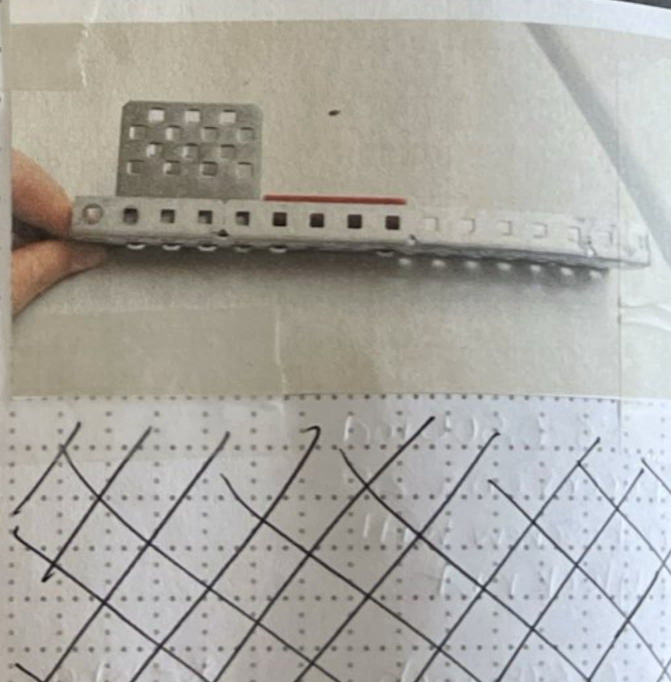
• - 0.375 inch Spacers in between the c-channel for support.

* - where two (2) rubber bands go across from front to back on each side.

✩ - 36 tooth gear with 20 teeth cut off for a slip gear.

(Continued)





Above are images of the puncher as it's being built and once it's finished and attached to the lift.

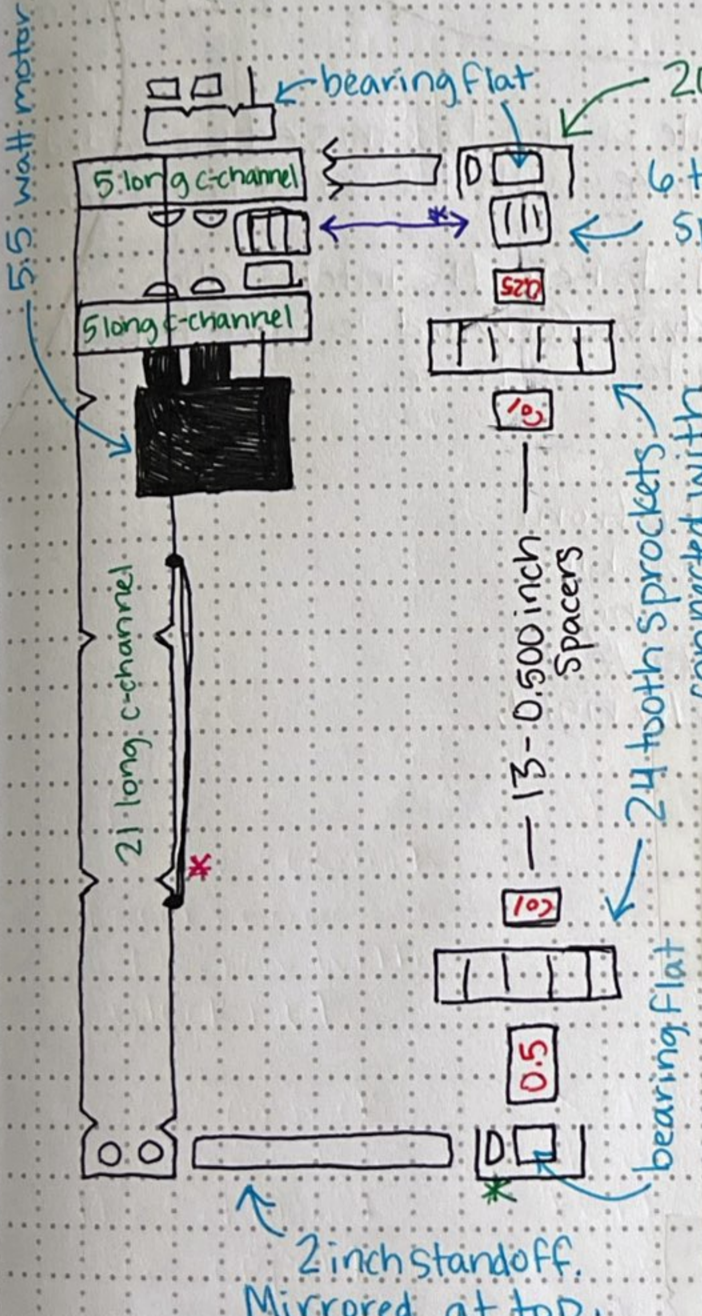
The puncher works well when tested. It has a good launch angle that helps triballs make it over the barrier and sometimes into the goal.

VP

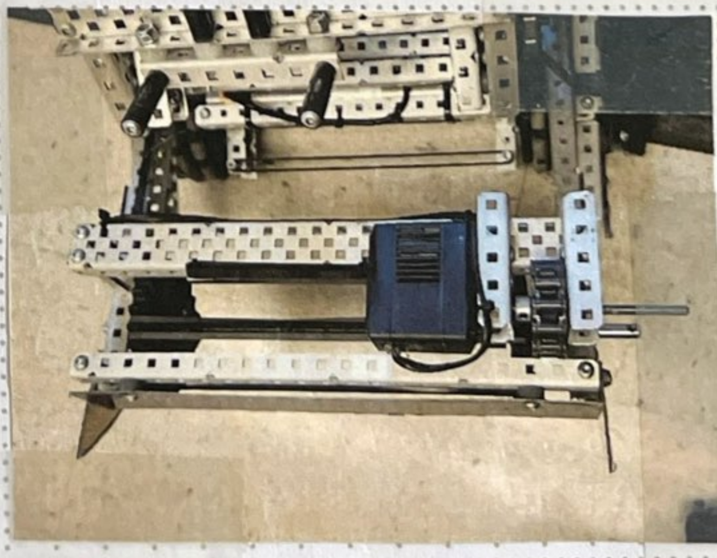
Build Intakes:

Goals for today → Assemble Intakes like our first robot
→ Attach intakes to front of robot

Below is a sketched diagram with an image of the intake. The 20 long c-channels attach to the base of the robot using one loose screw on each side to allow it to pivot up and down with the triballs and the goals.

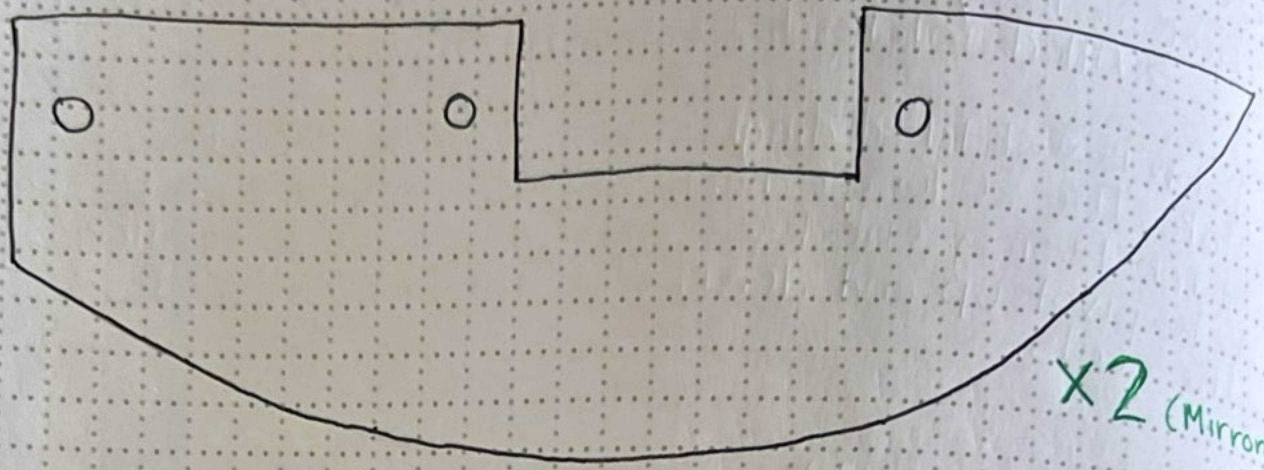


* Rubberband Ziptied to intake that hooks onto Puncher to make robot in 18 inches.



Project Build Intakes
Date 9/24/2023
Name Vanessa Perkins

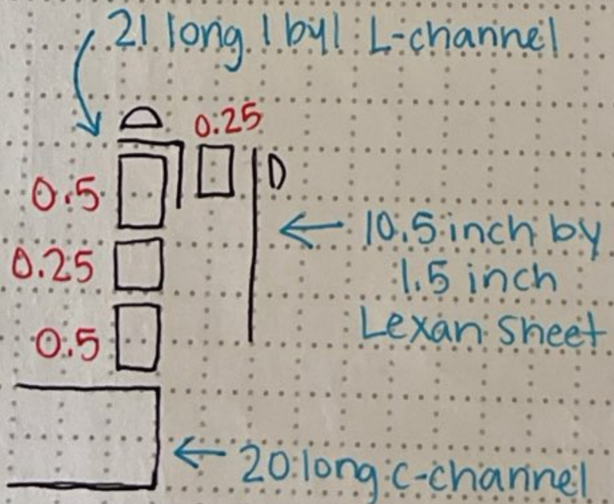
(Continued)

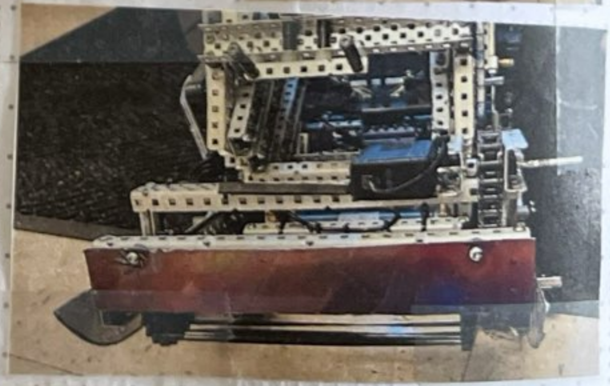


Above is a to scale template of the Lexan sled we added to the front sides of the intake with the screw holes for reference.

The purpose of these sleds is to help the intakes glide easier over the goal as we drive forward to score triballs without having to out take to save time.

Below is the diagram for the front intake guard to prevent the combine from getting entangled in any thing. It is a sketch from the left side with a front image to the right.





Anti-tips:

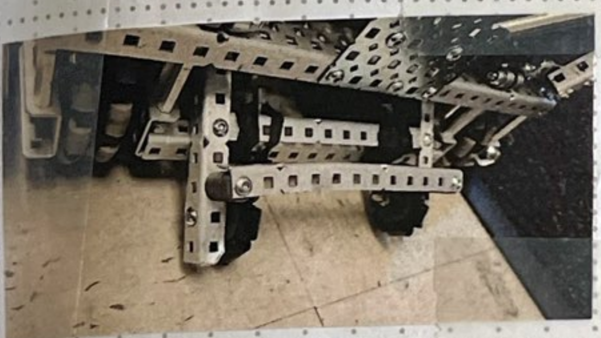
Goals for today → Solve the problem of tipping

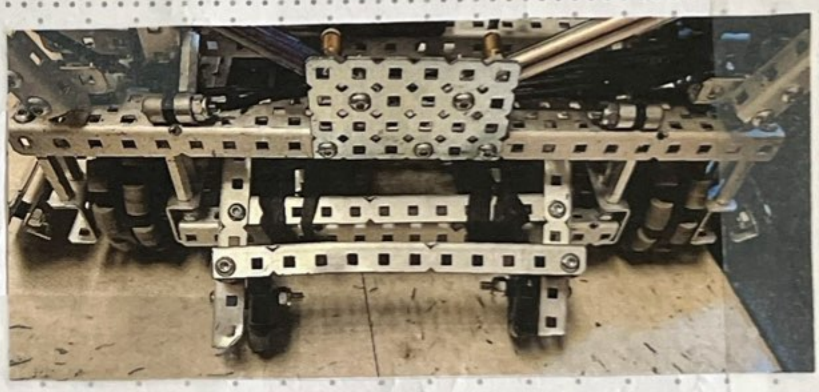
Problem/Challenge
With the height of our lift, the robot now tips over extremely easy.

Brainstorm
1. Add weight to base
2. Build Anti-tips
3. Shorten Lift

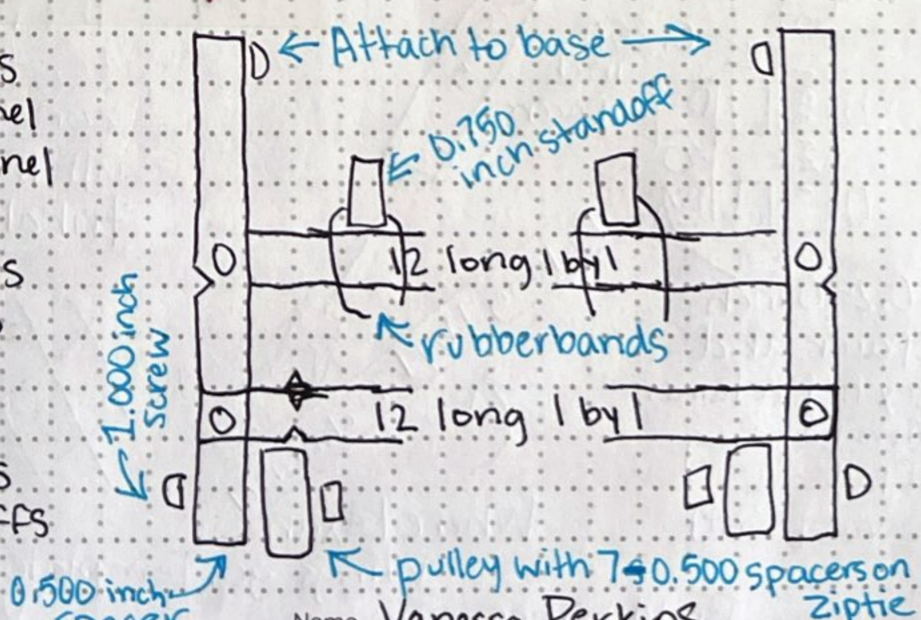
Select Option
2
Build and attach Anti-tips to back of the robot base

Build/Implement Solution
- gather parts
- choose mini wheels
- build structure
- assemble



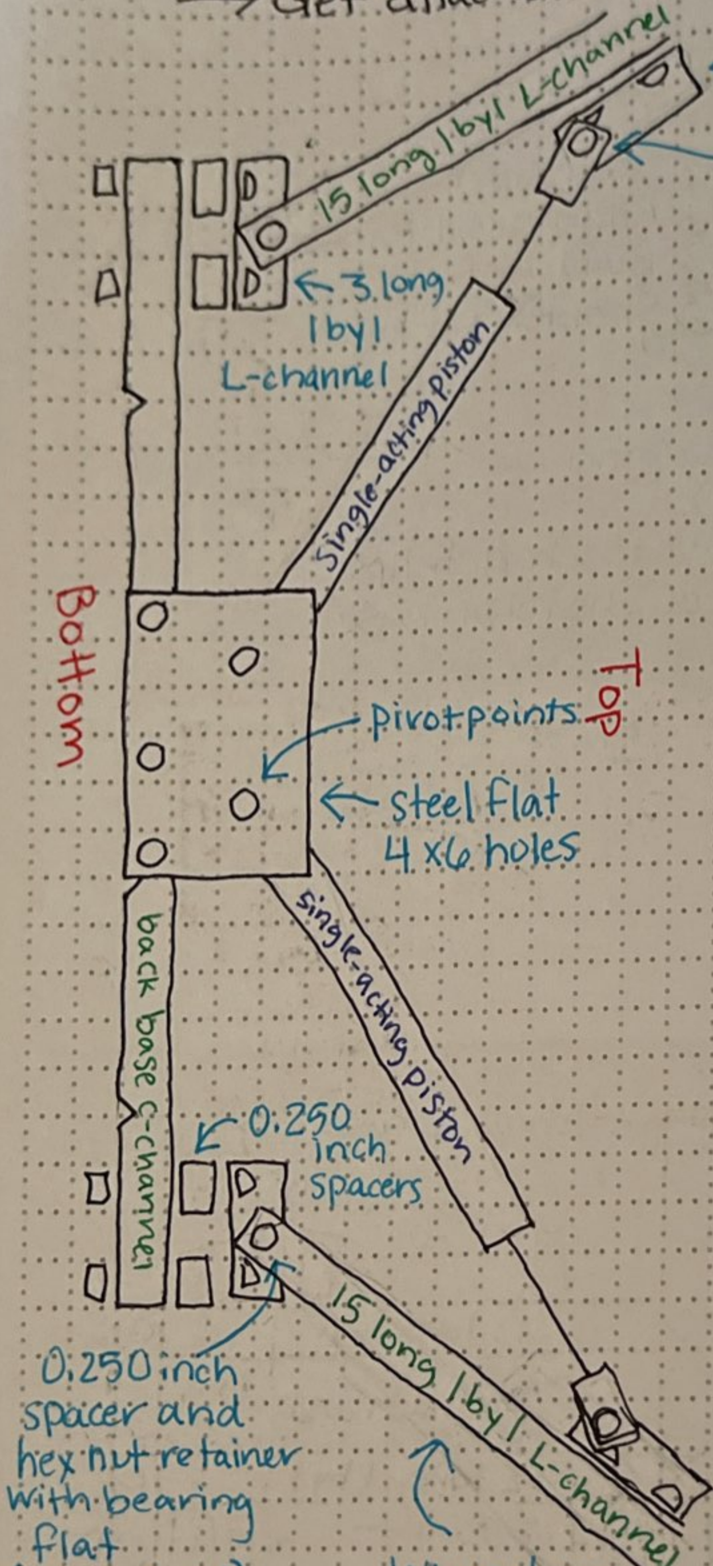


Anti-tips Parts List:
- 2-10 long 1by1 L-channels
- 1-12 long 1by1 L-channel
- 2-16 long 1by1 L-channel
- 2-pulleys
- 16-0.500 inch spacers
- 4-1.000 inch screws
- 2-zipties
- 8-nylock nuts
- 10-0.500 inch screws
- 4-0.750 inch standoffs
- 2 rubberbands



Build Wings:

Goals for today → Complete wings
→ Get attached to robot and test with triballs + goal



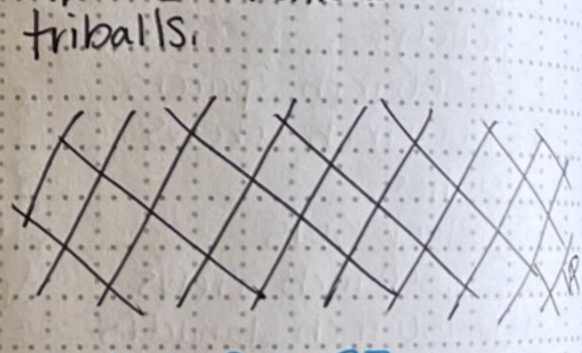
Where Lexan Wings go. Template on Page 27.

To the left is a labeled diagram of the wings and how they are attached to the robot base.

We went with vertical folding wings because it was all our robot had the space for because of the skirts and lift.

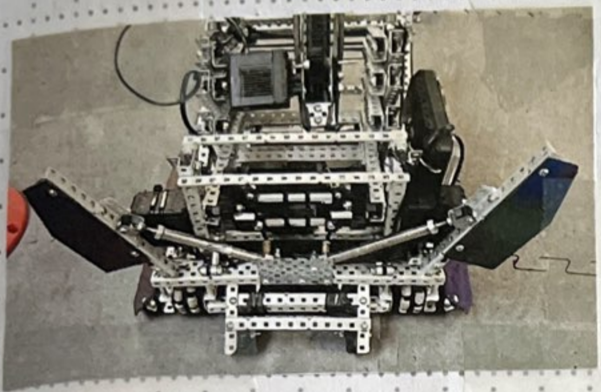
The wings fold back up when not being used to score large amounts of triballs at a time.

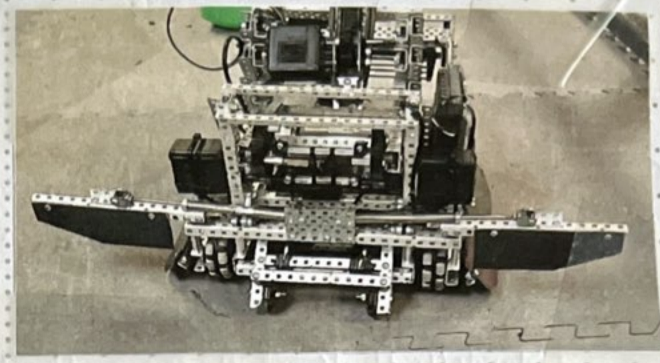
The wings can also be used for touching the matchload zone while launching all the matchload triballs.

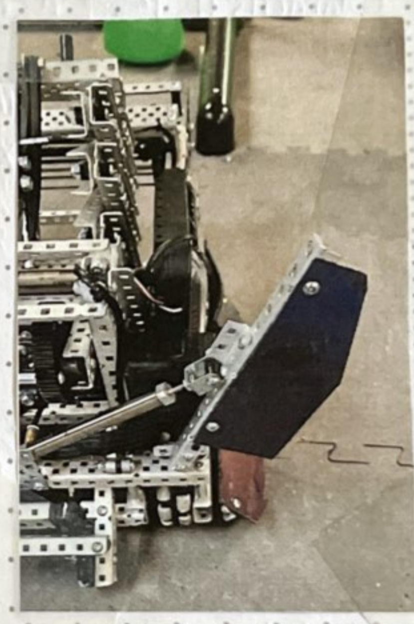


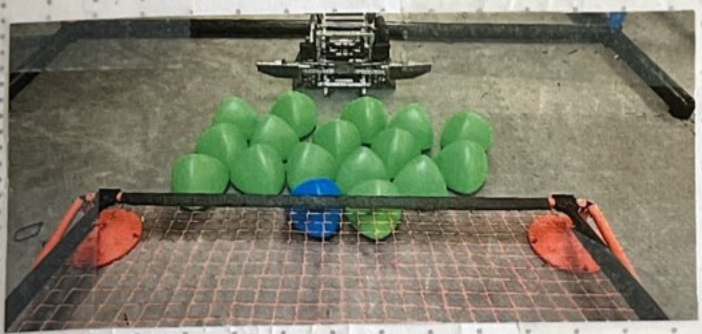
(Continued)

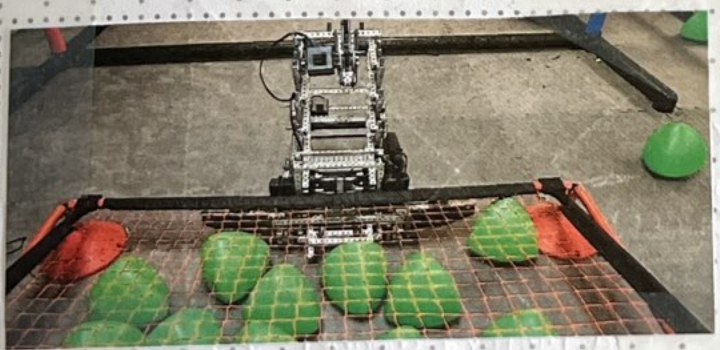
Below are images of the complete wings as well as a test with the goal and triballs before and after.

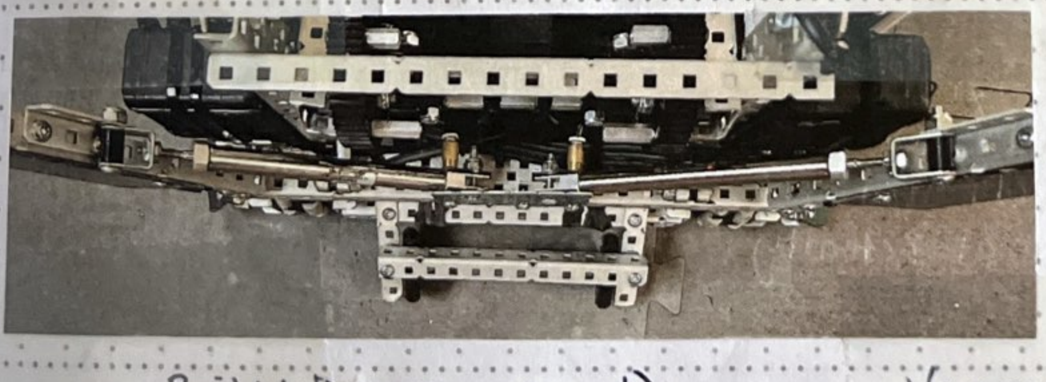










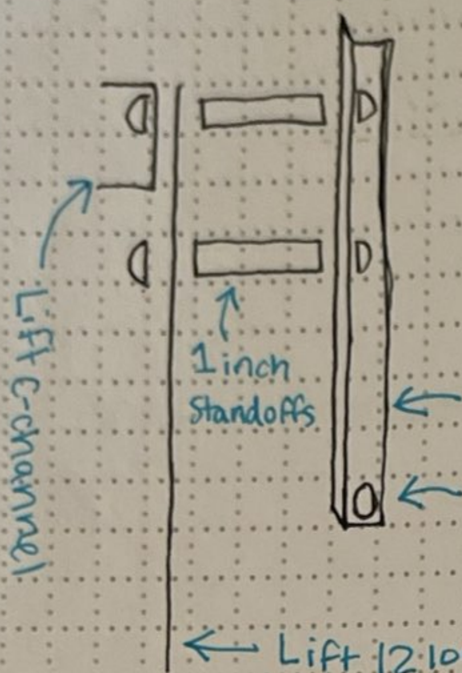


Project Build Wings (continued)   Name Vanessa Perkins

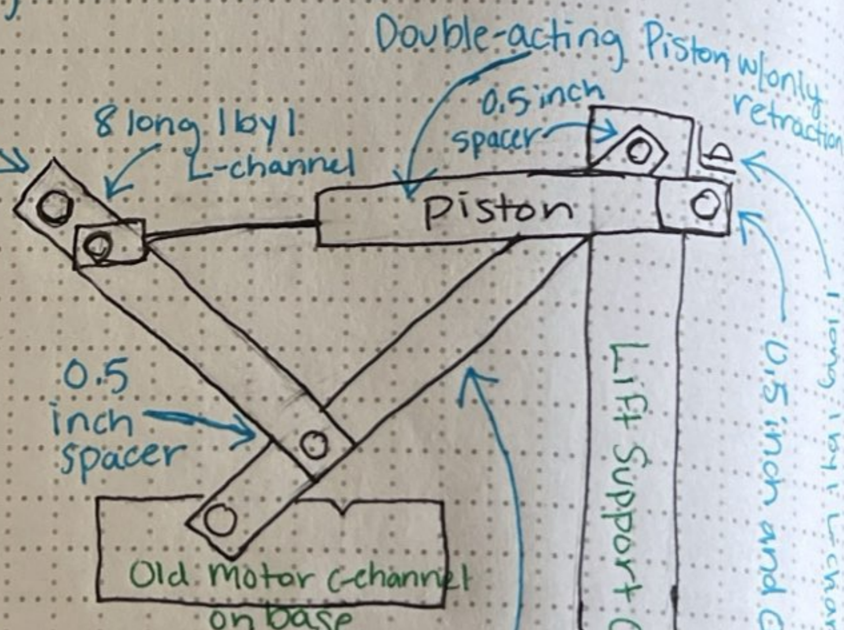
Date 10/23/2023

Build Hang Mechanism:

Goals for today → Build and test a hang mechanism
→ Have Owen practice driving and hanging



To the left is a diagram of the hang mechanism. It sticks off the lift and the standoff hooks onto the horizontal bar for an A-tier hang just above the ground when the lift lowers and compacts the robot.

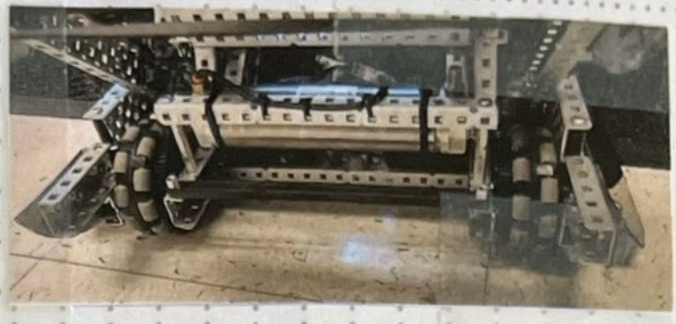


To the right is a diagram of the hang mechanism lock that hooks the lift locked down when the piston is pressed to keep the robot from falling down when the match ends and the lift brake no longer runs. The 1 inch standoff on the end is what actually latches the lift closed.

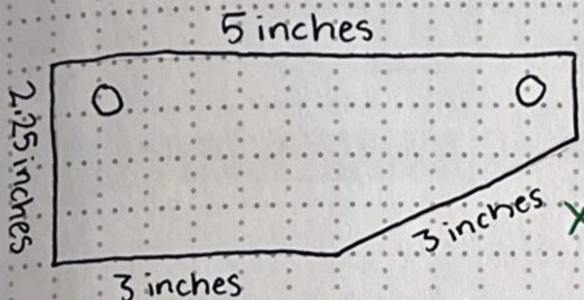
Project Build Hang Mechanism
Date 10/26/2023

Name Vanessa Perkins

(Continued)



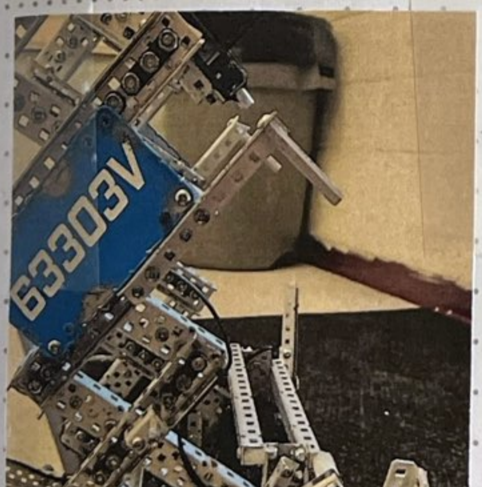
We added the air reservoir to the front of the robot underneath the intake and in between the wheels using a 16 long 1 by 1 L-channel attached with two(2) 2.000 inch standoffs and two(2) zipties looped through the L-channel. In the top right corner is a picture.



x2 (mirrored)

The Lexan dimensions for the wings on page 24 and 26.

Below are the images of the hang mechanism and the locking mechanism.

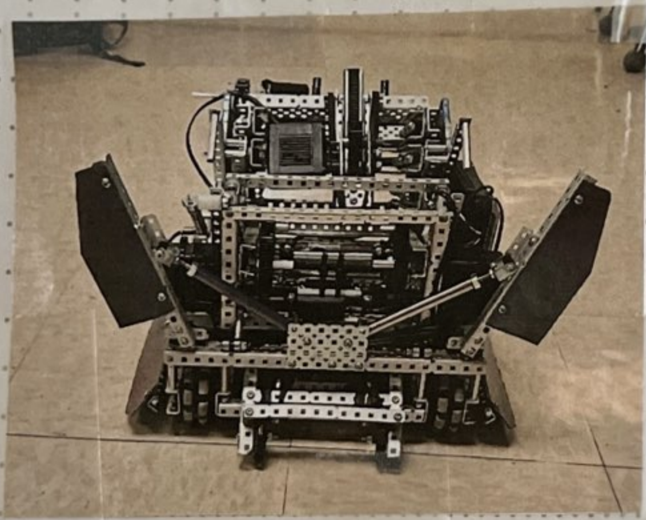


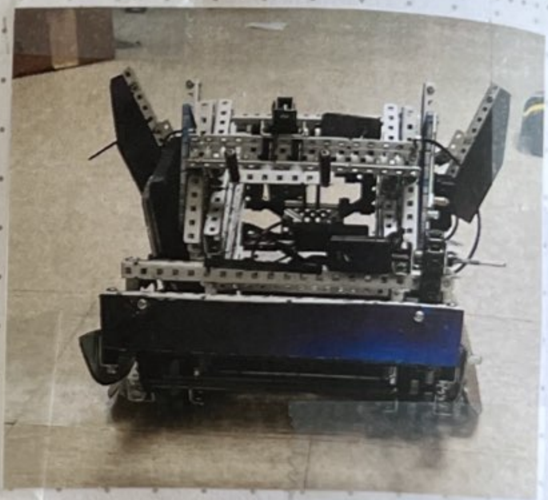
VP

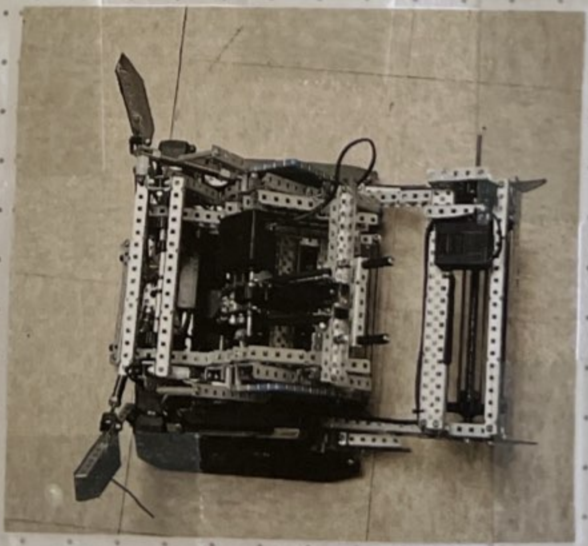
Project Build Hang Mechanism (continued) Name Vanessa Perkins

Date 10/26/2023

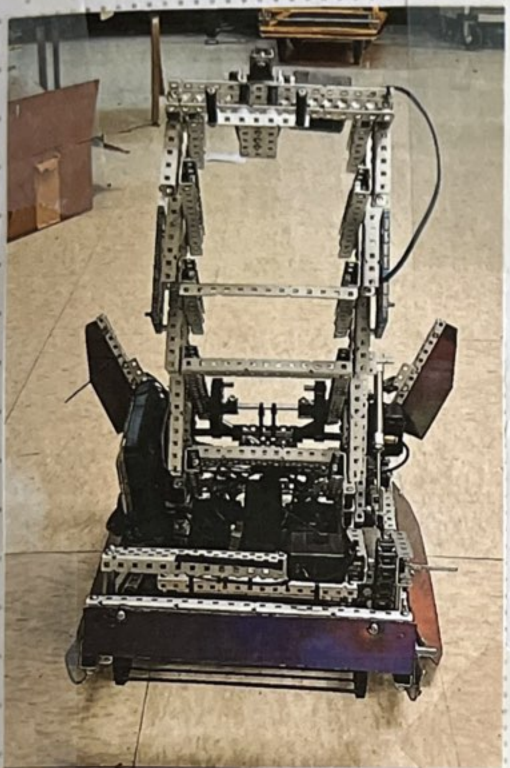
Finished Robot Images:

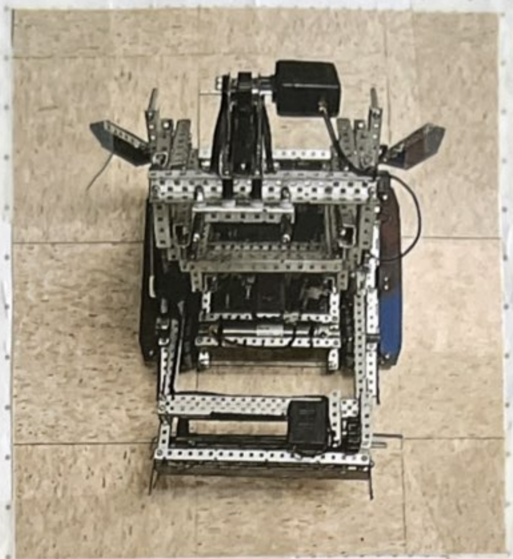


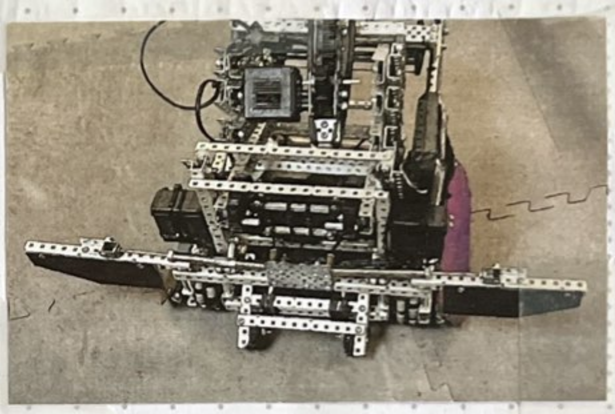


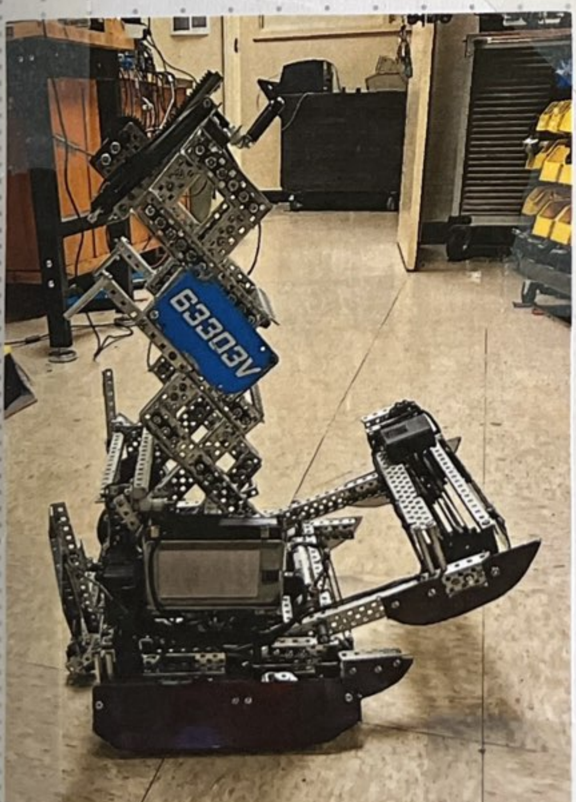


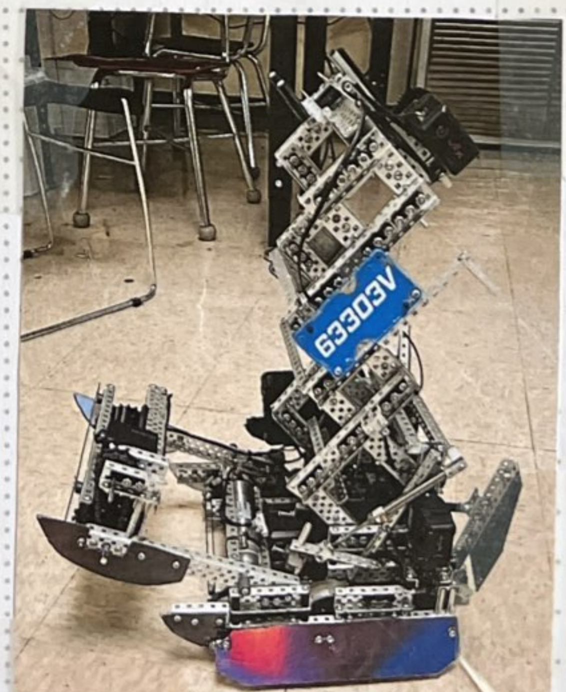
(Continued)











Project Finished Robot Images (Continued) Name Vanessa Perkins

Date 10/27/2023

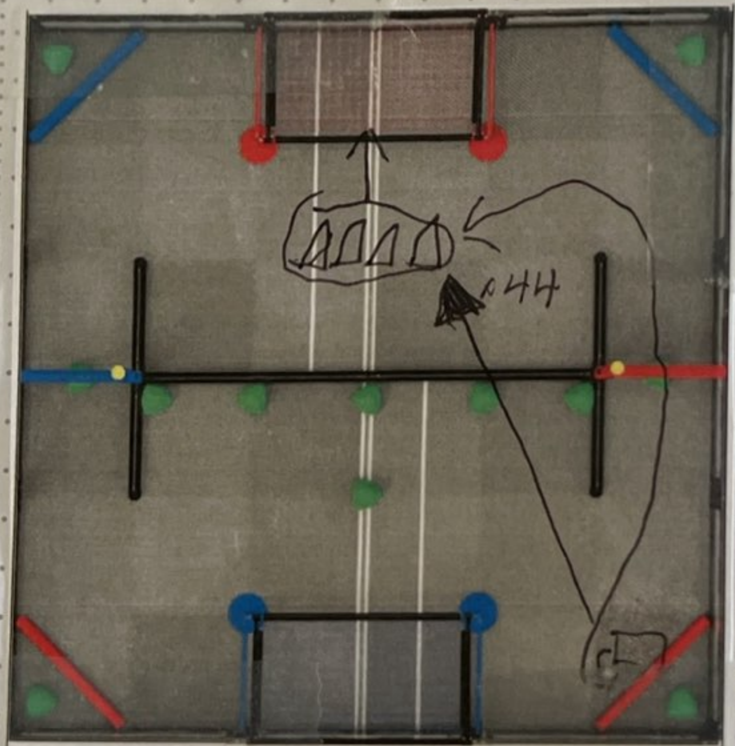
Driver Strategies:

Goals for today → Record Owen's Driver Strategies for Matches and Skills Runs
→ Have Owen practice all the strategies

Match Strategies:

1. Play offense for 1:30 by launching matchloads and pushing triballs into alliance goal. Go and hang on bar during last 15 seconds.
2. Play offense for 1:15 then go play defense for 20 seconds. Go hang during last 10 seconds.
3. Play defense for 1:25 by blocking the opponents trying to launch matchloads with our lift. Go and hang during last 20 seconds.

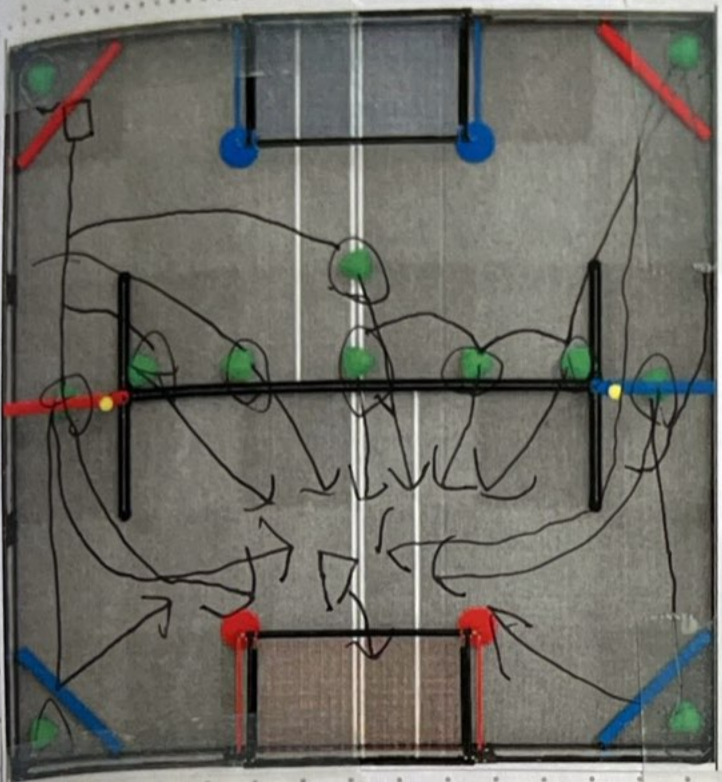
Skills Strategies/Maps:



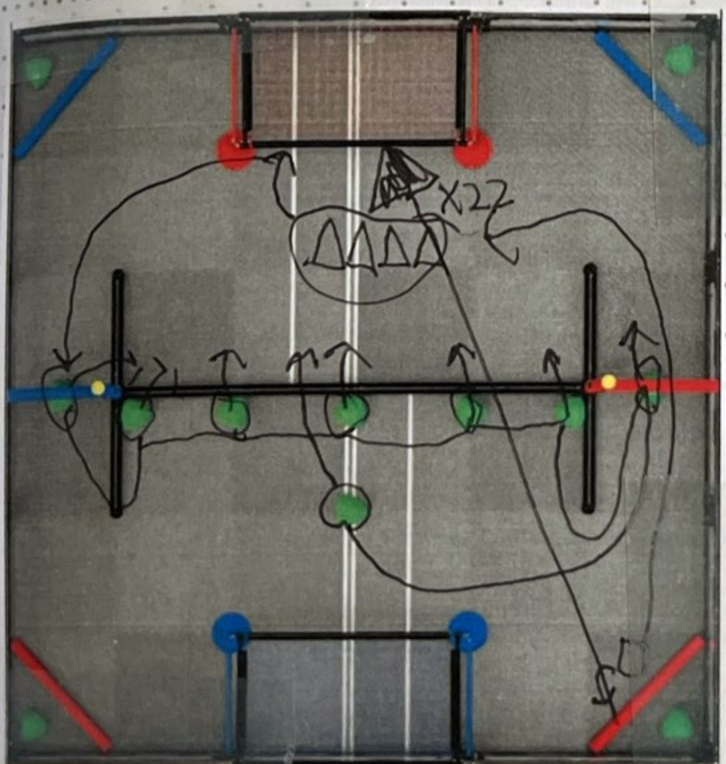
1. Begin at matchload corner. Launch all 44 triballs towards the opposite goal. Drive over and begin using the wings to push them swiftly into the goal.

VP

(Continued)



2. Focus on intaking and outtaking the triballs already on the field before using the matchloads. Hang on the bar with any remaining time.



3. Begin at matchload corner. Launch half of the matchloads to opposite goal. Move all triballs already on field to the goal and use wings to push all inside. Hang the robot on the bar with any remaining time.

VP

```
/*----------------------------------------------------------------------------*/
/* */
/* Module: main.cpp */
/* Author: VEX */
/* Created: Thu Sep 26 2019 */
/* Description: Competition Template */
/* */
/*----------------------------------------------------------------------------*/

// ---- START VEXCODE CONFIGURED DEVICES ----
// Robot Configuration:
// [Name] [Type] [Port(s)]
// Controller1 controller
// Drive_Front_Left motor 1
// Drive_Back_Left motor 20
// Drive_Front_Right motor 18
// Drive_Back_Right motor 15
// Intake motor 19
// Wings digital_out A
// Angler digital_out B
// LimitSwitchC limit C
// Lift motor_group 12, 16
// Puncher motor 13
// ---- END VEXCODE CONFIGURED DEVICES ----

#include "vex.h"

using namespace vex;

// A global instance of competition
competition Competition;

// define your global instances of motors and other devices here

/*---------------------------------------------------------------------------*/
/* Pre-Autonomous Functions */
/* */
/* You may want to perform some actions before the competition starts. */
/* Do them in the following function. You must return from this function */
/* or the autonomous and usercontrol tasks will not be started. This */
/* function is only called once after the V5 has been powered on and */
/* not every time that the robot is disabled. */
/*---------------------------------------------------------------------------*/

void pre_auton(void) {
// Initializing Robot Configuration. DO NOT REMOVE!
vexcodeInit();
```

```
// All activities that occur before the competition starts
// Example: clearing encoders, setting servo positions, ...
}
void moveForward(int velocity, int distance) {
Drive_Back_Left.spinFor(forward, distance, degrees, velocity, velocityUnits::pct,
false);
Drive_Back_Right.spinFor(forward, distance, degrees, velocity, velocityUnits::pct,
false);
Drive_Front_Left.spinFor(forward, distance, degrees, velocity, velocityUnits::pct,
false);
Drive_Front_Right.spinFor(forward, distance, degrees, velocity, velocityUnits::pct);
}

void moveBackward(int velocity, int distance) {
Drive_Back_Left.spinFor(reverse, distance, degrees, velocity, velocityUnits::pct,
false);
Drive_Back_Right.spinFor(reverse, distance, degrees, velocity, velocityUnits::pct,
false);
Drive_Front_Left.spinFor(reverse, distance, degrees, velocity, velocityUnits::pct,
false);
Drive_Front_Right.spinFor(reverse, distance, degrees, velocity, velocityUnits::pct);
}

void turnLeft(int velocity, int distance) {
Drive_Back_Left.spinFor(reverse, distance, degrees, velocity, velocityUnits::pct,
false);
Drive_Back_Right.spinFor(forward, distance, degrees, velocity, velocityUnits::pct,
false);
Drive_Front_Left.spinFor(reverse, distance, degrees, velocity, velocityUnits::pct,
false);
Drive_Front_Right.spinFor(forward, distance, degrees, velocity, velocityUnits::pct);
}

void turnRight(int velocity, int distance) {
Drive_Back_Left.spinFor(forward, distance, degrees, velocity, velocityUnits::pct,
false);
Drive_Back_Right.spinFor(reverse, distance, degrees, velocity, velocityUnits::pct,
false);
Drive_Front_Left.spinFor(forward, distance, degrees, velocity, velocityUnits::pct,
false);
Drive_Front_Right.spinFor(reverse, distance, degrees, velocity, velocityUnits::pct);
}
```

```
void halfLeft(int velocity, int distance) {
Drive_Front_Left.spinFor(reverse, distance, degrees, velocity,
velocityUnits::pct, false);
Drive_Back_Left.spinFor(reverse, distance, degrees, velocity,
velocityUnits::pct);
}

void halfRight(int velocity, int distance) {
Drive_Front_Right.spinFor(reverse, distance, degrees, velocity,
velocityUnits::pct, false);
Drive_Back_Right.spinFor(reverse, distance, degrees, velocity,
velocityUnits::pct, false);
}



/*---------------------------------------------------------------------------*/
/* */
/* Autonomous Task */
/* */
/* This task is used to control your robot during the autonomous phase of */
/* a VEX Competition. */
/* */
/* You must modify the code to add your own robot specific commands here. */
/*---------------------------------------------------------------------------*/

void autonomous(void) {

// ..........................................................................
// Insert autonomous user code here.
moveForward(50,100);
Wings.set(true);
moveBackward(50,100);
wait(.5,seconds);
Wings.set(false);
Puncher.spinFor(forward,250,degrees);
Puncher.setStopping(coast);
moveForward(30,100);
Intake.spin(reverse,100,pct);
wait(.75,seconds);
Intake.stop();
Intake.setStopping(hold);
moveForward(30,250); //this
```

```
turnRight(30,100);
moveForward(30,265); //this
turnRight(30,100);
Intake.spin(forward,100,pct);
moveForward(100,1100);
Intake.stop();
Intake.setStopping(hold);
moveBackward(50,750);
turnRight(50,400);
moveForward(50,1200);
turnLeft(50,185); //175
Intake.spin(reverse,100,pct);
moveForward(70,1750);
wait(.5,seconds);
Intake.stop();
Intake.setStopping(hold);
turnLeft(50,40);
moveBackward(50,3000);
turnRight(50,400);
Lift.spinFor(forward,1,turns);
moveForward(50,2500);


// ..............................................................................
}

/*--------------------------------------------------------------------------*/
/* */
/* User Control Task */
/* */
/* This task is used to control your robot during the user control phase of */
/* a VEX Competition. */
/* */
/* You must modify the code to add your own robot specific commands here. */
/*--------------------------------------------------------------------------*/

void usercontrol(void) {
// User control code here, inside the loop
while (1) {

Drive_Front_Left.spin(forward, Controller1.Axis3.position(percent) +
Controller1.Axis1.position(percent), percent); // The code for driving with the Front
Left motor

Drive_Back_Left.spin(forward, Controller1.Axis3.position(percent) +
Controller1.Axis1.position(percent), percent); // Left Back Drive Code
```

```
Drive_Front_Right.spin(forward, Controller1.Axis3.position(percent) -
Controller1.Axis1.position(percent), percent); // Right Front Drive Code

Drive_Back_Right.spin(forward, Controller1.Axis3.position(percent) -
Controller1.Axis1.position(percent), percent); // Right Back Drive Code


if(Controller1.ButtonDown.pressing())
{Lift.spin(forward,-50,pct);
Lift.setStopping(hold);
}


else if(Controller1.ButtonR2.pressing())
{ Intake.setVelocity(100,percent);
Intake.spin(vex::directionType::fwd, 100, velocityUnits::pct);
Intake.setStopping(hold);
}

else if(Controller1.ButtonR1.pressing())
{ Intake.setVelocity(-100,percent);
Intake.spin(vex::directionType::fwd, -100, velocityUnits::pct);
Intake.setStopping(hold);
}


else if(Controller1.ButtonUp.pressing())
{Lift.spin(vex::directionType::fwd, 100, velocityUnits::pct);
Lift.setStopping(hold);
}

else if(Controller1.ButtonX.pressing())
{Wings.set(true);
}

else if(Controller1.ButtonA.pressing())
{Wings.set(false);
}

else if(Controller1.ButtonL1.pressing())
{Puncher.spin(vex::directionType::fwd, 100, velocityUnits::pct);
Puncher.setStopping(hold);
}

else
{
```

```
Intake.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);
Lift.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);
Puncher.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);
}


}

digital_out dig1 = digital_out( Brain.ThreeWirePort.A );
if( Controller1.ButtonB.pressing() ) {
dig1.set( true );
}
else {
dig1.set( false ); //Code for the Pneumatics
}

this_thread::sleep_for(10);

// This is the main execution loop for the user control program.
// Each time through the loop your program should update motor + servo
// values based on feedback from the joysticks.

// ........................................................................
// Insert user code here. This is where you use the joystick values to
// update your motors, etc.
// ........................................................................

wait(20, msec); // Sleep the task for a short amount of time to
// prevent wasted resources.
}


//
// Main will set up the competition functions and callbacks.
//
int main() {
// Set up callbacks for autonomous and driver control periods.
Competition.autonomous(autonomous);
Competition.drivercontrol(usercontrol);

// Run the pre-autonomous function.
pre_auton();

// Prevent main from exiting with an infinite loop.
while (true) {
wait(100, msec);
```

Competition Analysis:

Goals for today → Review and Discuss the results of the Northeast TN VRC Season Kickoff Competition

Insights from Competiton	
Pros	Cons
Consistent hang, best hang present	Couldn't climb barrier
Puncher was fast & elimination	Skirts made robot too wide
Intakes worked decent	Auton program needed time to perfect
Structure was strong	Wings were hard to manage
Strong offense	Weak defensive play
Won Design Award	

Matches	Red Score	Blue Score	Alliance
Q-2	73	166	57711C
Q-14	57	67	24816H
Q-19	85	154	12876A
Q-23	167	36	24816H
Q-28	135	105	42652A
Q-34	130	116	12876C
Q-42	135	79	39840B
Q-48	148	100	42652B
QF 4.1	160	77	57711C
SF 2.1	120	124	57711C
F1	173	95	57711C

*The highlighted scores represent our alliance that match.

Our team went to a competition at Sullivan this weekend, October 28th, 2023.

Our overall performance was not dissappointing. We ranked 4th during Qualifying matches, got to the Finals, and won the design award.

In the end, it was a good first competition to help us recognize what we could improve upon for the remainder of the season.

Next Competition: November 4th, 2023

Project Competition Analysis Name Vanessa Perkins
Date 10/30/2023

New Goals:

Goals for today → Record our goals for the week before the next competition and any action plans required to complete these goals.

Goal Planner

Robot Goals

Make Three (3) Design changes to the Robot

Action plans

— Review performance —
— Brainstorm new ideas —
— Build efficiently —
– Keep a timeline and schedule –
— Test and Revise as needed —

Performance Goals

Rank Top Four (4)
Make it to the Finals
Win Design Award
FWP 70% of time
Highest OPR
Win Auton 100% of time

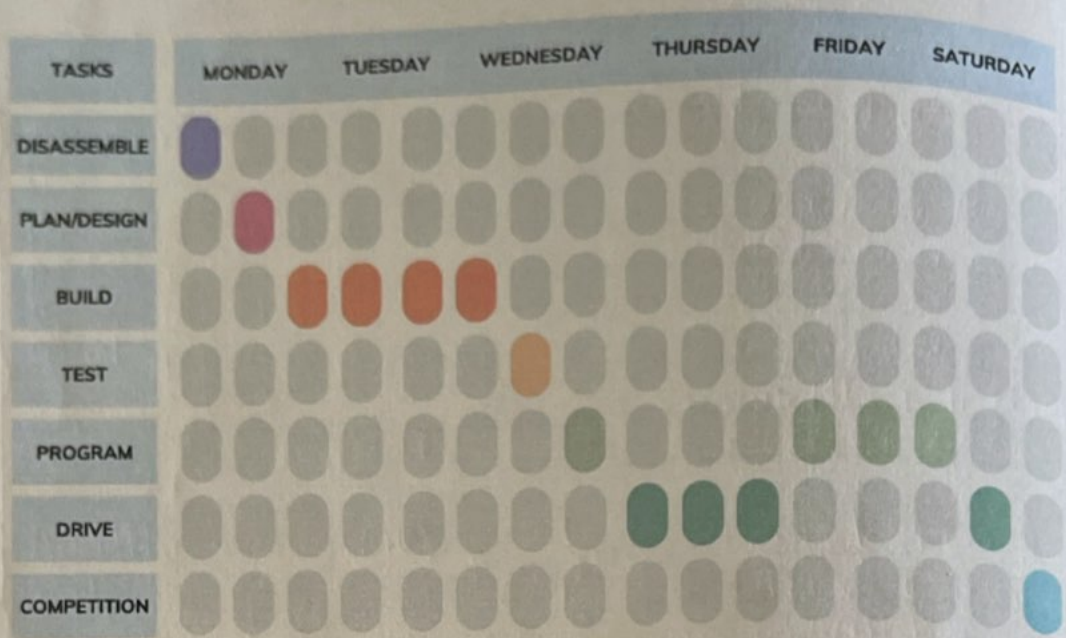
Action plans

— Scout efficiently —
— Choose a good alliance —
— Practice Interviews —
— Make program consistent —
— Practice Offense —
— Make program consistent —

Above is an organized chart to keep us on track for our up coming competition. We have limited time to prepare so our goals had to be reasonable.

Our biggest goal is that we qualify for STATES at this competition so we can focus on our driver and programming skills from now on out.

GANTT CHART
MIDDLE SEASON TIMELINE



Disassemble- Take apart only the necessary changes to robot #2 and reuse parts to save some budget
Plan/Design- Review the rules, research ideas, and decide on subsystems being redesigned
Build- Begin assembling the pieces
Test- Intermittently test what we have built and change as necessary
Program- Set up controls and autonomies
Drive- Practice learning controls and strategies
Competition- Final performance of the robot at Lakeway Christian Academy

Timeline Duration: 10/30 through 11/4

Brainstorming Subsystem Redesigns:

Goals for today → Take our Competition Analysis and come up with which subsystems we need to redesign.

Below is a list of the subsystems we plan to redesign along with our goals for how to improve each piece. In the lower left corner is a picture of our robot after dissassembling each of our decided subsystems.

1.) Base Under Bracing Bars: The bars currently prevent us from going over the barrier so we plan to move them to the top of the base so we can climb the barrier.

2.) Skirts: The skirts had little effect during defense and only made our base bulkier when trying to navigate the field. Improvement will include just removing the skirts entirely.

3.) Brain Mount: The brain is in an awkward location and will be move to the center inside of the robot for safety.

4.) Intakes: The intakes protruded very far on the front of the robot so we plan to adjust the angle and placement.

5.) Hanging Mechanism Locks: We only had one (1) piston lock on our hang mechanism before so it was hanging crooked. To improve this we will be adding a piston lock on the otherside as well.

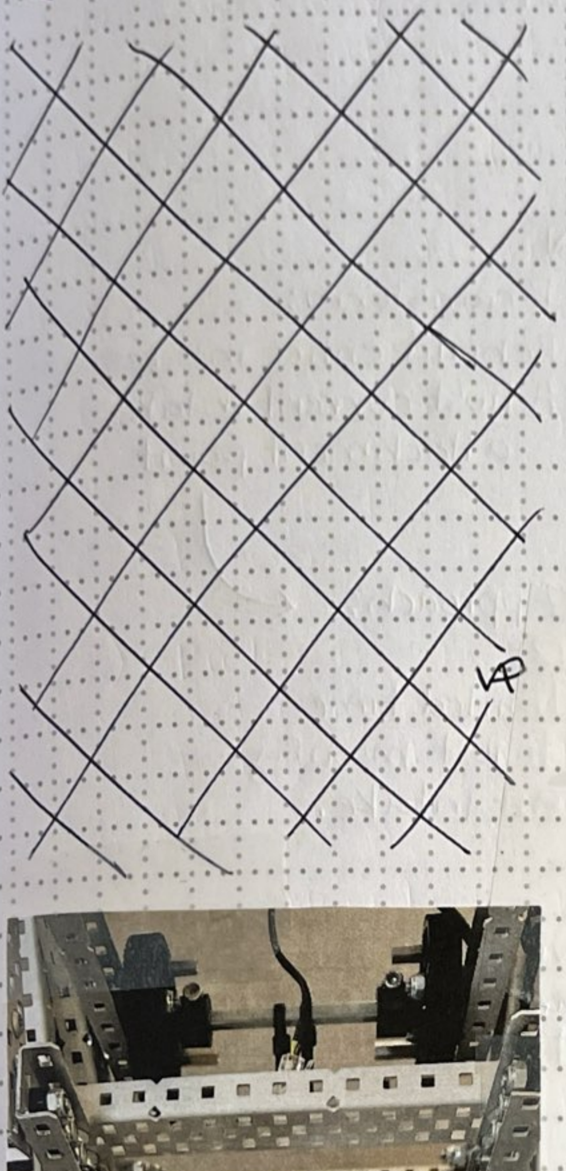
Redesign Base Bracing:

Goals for today → change base braces so the robot can drive over barrier
→ remove skirts
→ change Brain location

1. We removed the skirts from the base by unscrewing the hinges holding them on.
2. We removed the c-channel bracing under the base between the wheels that the skirts rested on so we could climb over the barrier.
3. Add 24 long c-channel across front of base on top of the seven (7) long c-channel the intakes where previously attached to by changing screws to 1.500 inches long.
4. Add the sixteen (16) long 1by1 L-channel through the 6th hole back on the inside of the base using a 0.750 inch standoff on each end with four (4) half inch screws
5. Add the twenty-four (24) long 1by1 L-channel four (4) holes forward from the back of the base using two (2) 8mm spacers on the outsides and 1.000 inch standoffs to attach it to the inner base c-channels.
6. Add the two (2) thirteen long 1by1 L-channels in between the L-channels from steps 4 and 5 using four (4) half inch screws and four (4) Nylock nuts. Add the two (2) inch standoff in the middle of the two (2) thirteen (13) long L-channels for extra bracing.
7. Attach Brain to the thirteen (13) long L-channels using four (4) half inch screws.

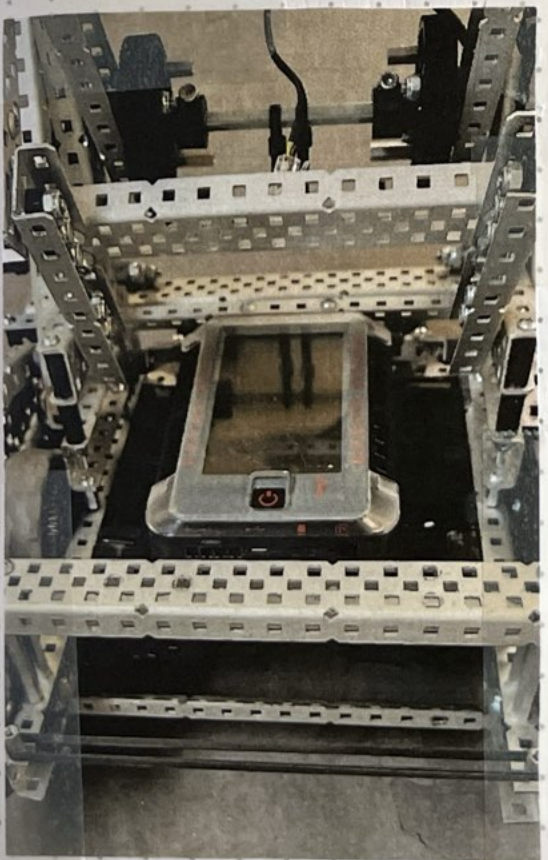
On page 43 is the complete parts list as well as pictures of the finished updated bracing and Brain location. It also shows the robot without the base skirts.

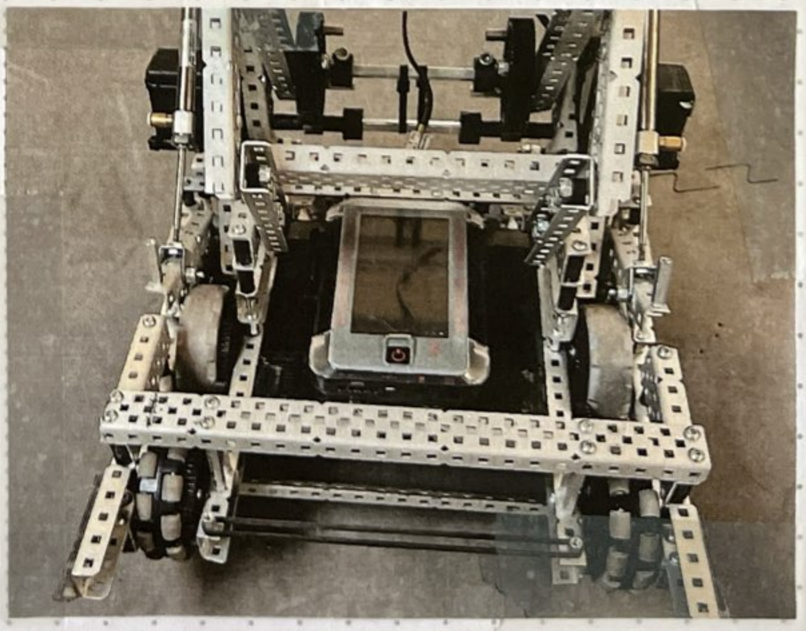
(continued)



Base Bracing Parts List:

- 24 long aluminum C-channel
- 24 long 1by1 aluminum L-channel
- 16 long 1by1 aluminum L-channel
- 2 - 0.750 inch standoffs
- 2 - 1.000 inch standoffs
- 1 - 2.000 inch standoff
- 4 - 8mm spacers
- 2 - 1.000 inch screws
- 4 - 1.500 inch screws
- 16 - 0.500 inch screws
- 2 - 13 long aluminum 1by1 L-channels.
- 6 Nylock nuts





Redesign Intakes:

Goals for today → Improve Intakes
→ Test Intakes

Problem/Challenge:
The Intakes stick out a bit from the robot and are at a slightly wrong angle.

Brainstorm:
① Rebuild entire intakes
② Adjust current intake's attachment point

Select Approach:
②
Due to limited time we chose to just modify our current intake.

Build/Implement Solution:
- Shorten Intake length
- Change Intake pivot point

Above is our process for improving the intakes. To implement the solution we added a four (4) long 1 by 1 aluminum L-channel to the top of the old intake mount location using two (2) half inch screws and nylock nuts. Add a 1.000 inch screw through the bottom fourteenth hole of the intake c-channels and add a loose nylock nut.

The c-channel from page 42 helps keep the intakes at the right angle while still being able to pivot upward.

Project Redesign Intakes
Date 10/31/2023
Name Vanessa Perkins

Redesign Hang Mechanism Lock:

Goals for today → brainstorm and select a way to improve the Hang Mechanism Lock
→ Build and test solution

Problem/Challenge:
The Hang is crooked and locks inconsistently.

Brainstorm:
1. Add a piston lock to the other side as well
2. Find new way to lock hang

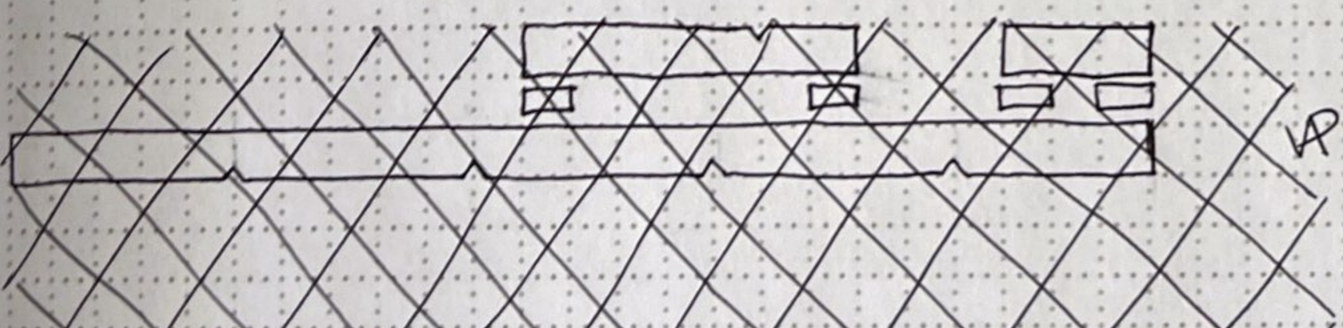
Select Approach:
1
A 2nd piston lock should balance our hang and make it more consistant.

Build/Implement Solution:
- Repeat/Mirror original hang mechanism lock

To improve our hang we mirrored the piston lock on to the left side of the lift.

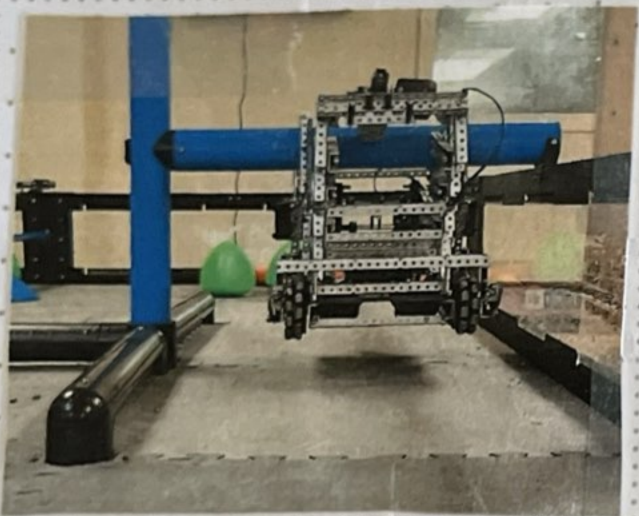
The instructions and details for this are located at the bottom of page 26.

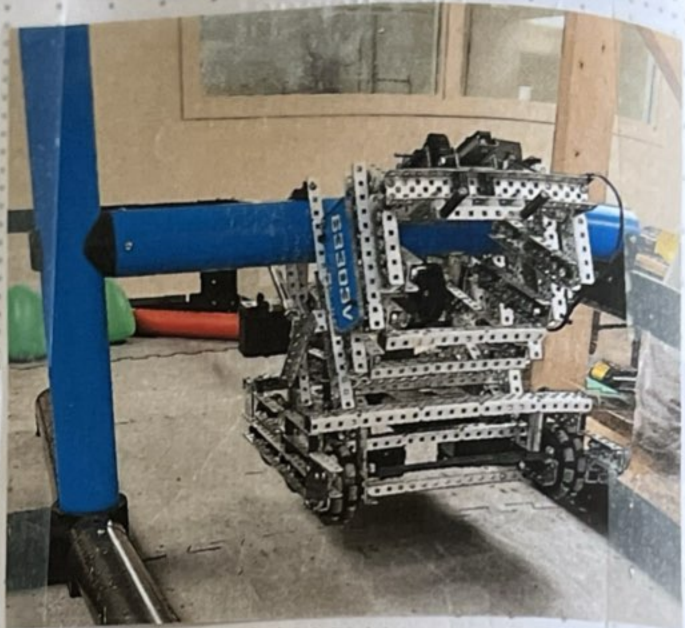
Pictures and testing the improving hang mechanism locks are located on page 46.



(Continued)

Below are the images of the complete hang mechanism locks with our test images to demonstrate how straight it is now.





Project Redesign Hang Mechanism Lock (Continued) Name Vanessa Perkins

Date 11/1/2023

Matchload Zone Arm:

Goals for today → Find Solution for how to touch matchload zone with robot

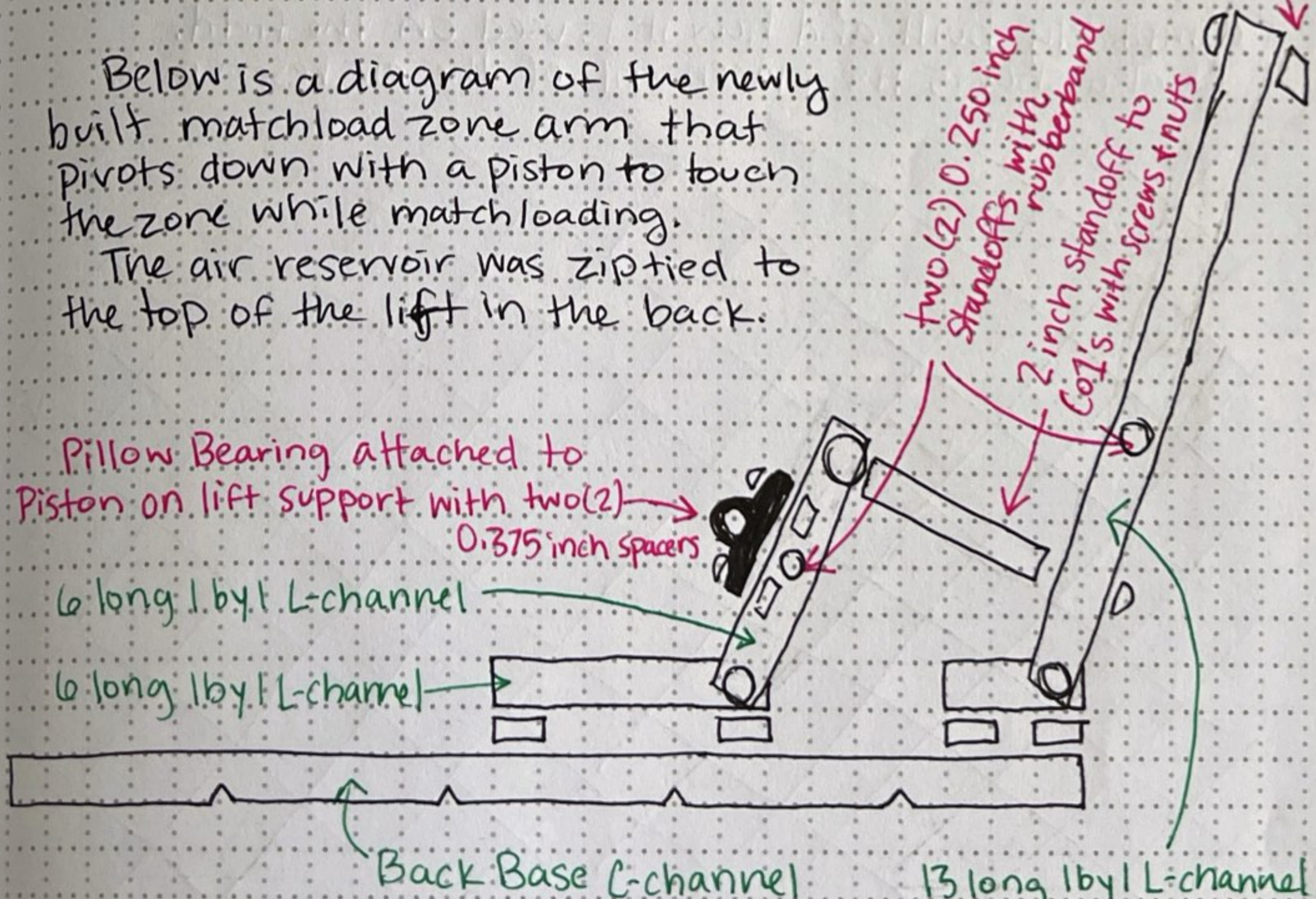
Problem/Challenge
How to touch matchload zone with robot.

Brainstorm
1. Build new wings
2. Build arm on one(1) side
3. Just touch zone with base

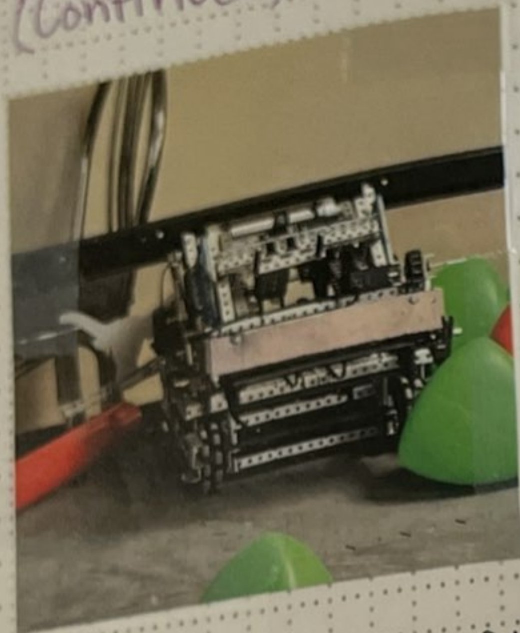
Select Option
2
Build an arm that extends and hooks the matchload zone.

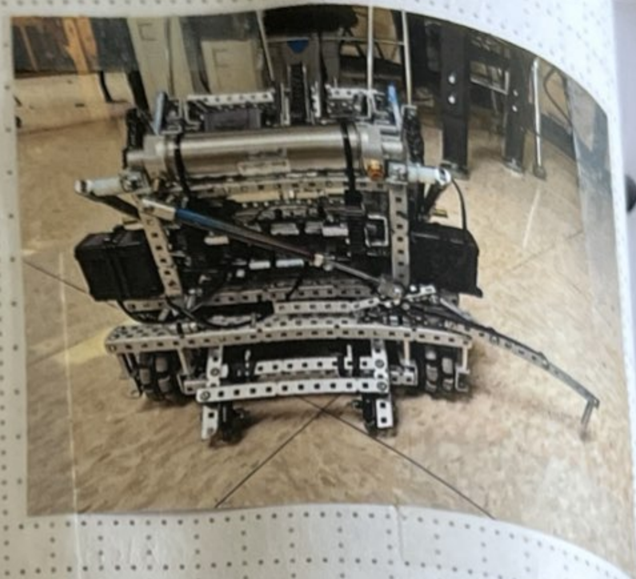
Build/Implement Solution
- Attach piston to a pivoting arm on back of the robot.

Below is a diagram of the newly built matchload zone arm that pivots down with a piston to touch the zone while matchloading.
The air reservoir was ziptied to the top of the lift in the back.



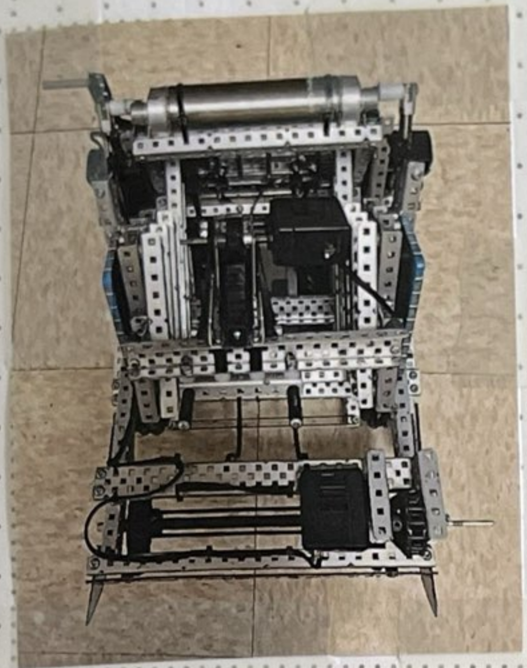
(Continued)



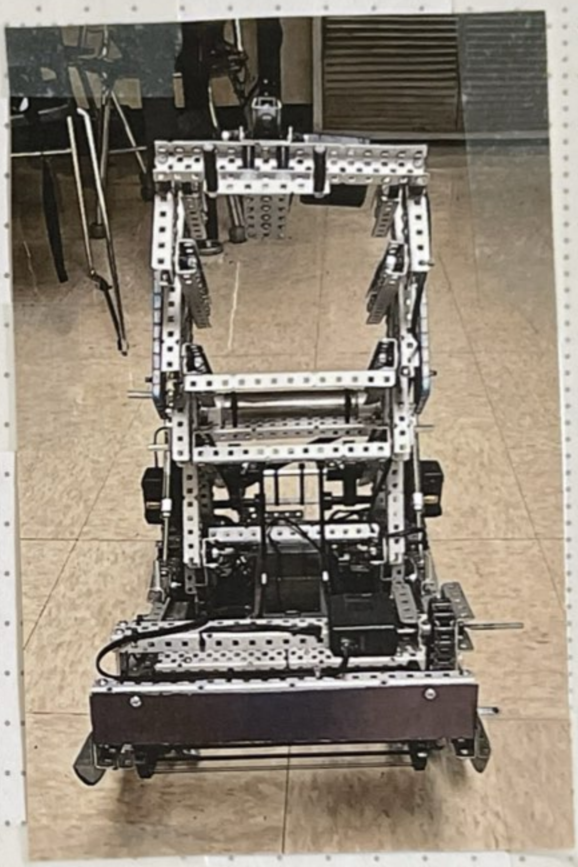


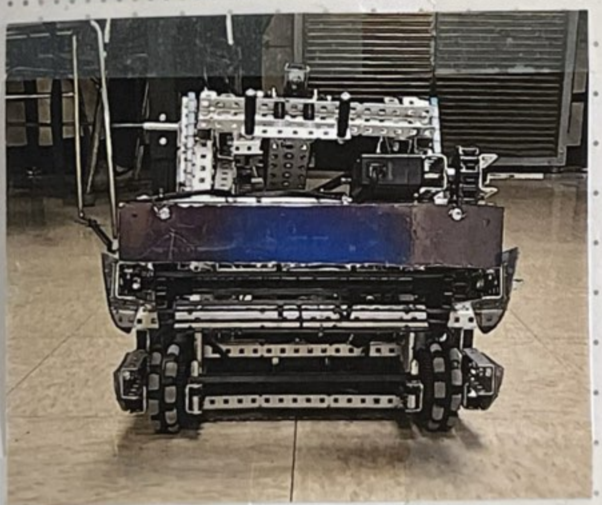
Above are images of the matchload zone arm once completely built and how it is used on the field. It also shows the placement of the air reserviors.

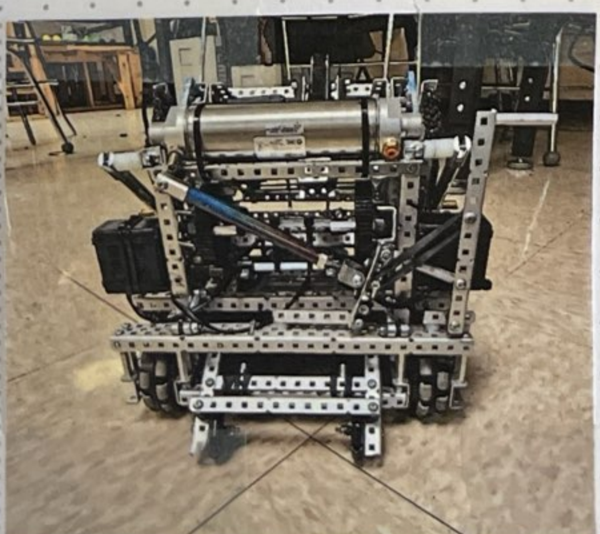
Finished Robot Images:



Below are the images of the complete robot for this upcoming competition on November 4th.







```
/*----------------------------------------------------------------------------
*/
/* */
/* Module: main.cpp */
/* Author: VEX */
/* Created: Thu Sep 26 2019 */
/* Description: Competition Template */
/* */
/*----------------------------------------------------------------------------
*/

// ---- START VEXCODE CONFIGURED DEVICES ----
// Robot Configuration:
// [Name] [Type] [Port(s)]
// Controller1 controller
// Drive_Front_Left motor 8
// Drive_Back_Left motor 7
// Drive_Front_Right motor 11
// Drive_Back_Right motor 15
// Intake motor 19
// Hook digital_out B
// Lift motor_group 20, 9
// Puncher motor 10
// Lock digital_out C
// ---- END VEXCODE CONFIGURED DEVICES ----

#include "vex.h"

using namespace vex;

// A global instance of competition
competition Competition;

// define your global instances of motors and other devices here

/*---------------------------------------------------------------------------*
/
/* Pre-Autonomous Functions */
/* */
/* You may want to perform some actions before the competition starts. */
/* Do them in the following function. You must return from this function */
/* or the autonomous and usercontrol tasks will not be started. This */
```

```
/* function is only called once after the V5 has been powered on and */
/* not every time that the robot is disabled. */
/*---------------------------------------------------------------------------*
/

void pre_auton(void) {
// Initializing Robot Configuration. DO NOT REMOVE!
vexcodeInit();

// All activities that occur before the competition starts
// Example: clearing encoders, setting servo positions, ...
}
void moveForward(int velocity, int distance) {
Drive_Back_Left.spinFor(forward, distance, degrees, velocity,
velocityUnits::pct, false);
Drive_Back_Right.spinFor(forward, distance, degrees, velocity,
velocityUnits::pct, false);
Drive_Front_Left.spinFor(forward, distance, degrees, velocity,
velocityUnits::pct, false);
Drive_Front_Right.spinFor(forward, distance, degrees, velocity,
velocityUnits::pct);
}

void moveBackward(int velocity, int distance) {
Drive_Back_Left.spinFor(reverse, distance, degrees, velocity,
velocityUnits::pct, false);
Drive_Back_Right.spinFor(reverse, distance, degrees, velocity,
velocityUnits::pct, false);
Drive_Front_Left.spinFor(reverse, distance, degrees, velocity,
velocityUnits::pct, false);
Drive_Front_Right.spinFor(reverse, distance, degrees, velocity,
velocityUnits::pct);
}

void turnLeft(int velocity, int distance) {
Drive_Back_Left.spinFor(reverse, distance, degrees, velocity,
velocityUnits::pct, false);
Drive_Back_Right.spinFor(forward, distance, degrees, velocity,
velocityUnits::pct, false);
Drive_Front_Left.spinFor(reverse, distance, degrees, velocity,
velocityUnits::pct, false);
```

```
Drive_Front_Right.spinFor(forward, distance, degrees, velocity,
velocityUnits::pct);
}

void turnRight(int velocity, int distance) {
Drive_Back_Left.spinFor(forward, distance, degrees, velocity,
velocityUnits::pct, false);
Drive_Back_Right.spinFor(reverse, distance, degrees, velocity,
velocityUnits::pct, false);
Drive_Front_Left.spinFor(forward, distance, degrees, velocity,
velocityUnits::pct, false);
Drive_Front_Right.spinFor(reverse, distance, degrees, velocity,
velocityUnits::pct, false);
}

void leftDrift(int velocityHigh, int velocityLow, int distance) {
Drive_Back_Left.spinFor(forward, distance, degrees, velocityLow,
velocityUnits::pct, false);
Drive_Back_Right.spinFor(forward, distance, degrees, velocityHigh,
velocityUnits::pct, false);
Drive_Front_Left.spinFor(forward, distance, degrees, velocityLow,
velocityUnits::pct, false);
Drive_Front_Right.spinFor(forward, distance, degrees, velocityHigh,
velocityUnits::pct,false);
}

void rightDrift(int velocityHigh, int velocityLow, int distance) {
Drive_Back_Left.spinFor(forward, distance, degrees, velocityHigh,
velocityUnits::pct, false);
Drive_Back_Right.spinFor(forward, distance, degrees, velocityLow,
velocityUnits::pct, false);
Drive_Front_Left.spinFor(forward, distance, degrees, velocityHigh,
velocityUnits::pct, false);
Drive_Front_Right.spinFor(forward, distance, degrees, velocityLow,
velocityUnits::pct);
}

/*----------------------------------------------------------------------------
/
/* */
/* Autonomous Task */
/* */
```

```
/* This task is used to control your robot during the autonomous phase of */
/* a VEX Competition. */
/* */
/* You must modify the code to add your own robot specific commands here. */
/*----------------------------------------------------------------------------*
/

void autonomous(void) {

// ..........................................................................
// Insert autonomous user code here.
Hook.set(true);
wait(.5,seconds);
moveBackward(30,400);
Hook.set(false);
//moveForward(60,200);
Drive_Back_Left.spin(forward, 30,pct);
Drive_Back_Right.spin(forward, 40,pct);
Drive_Front_Left.spin(forward, 30,pct);
Drive_Front_Right.spin(forward, 40,pct);
wait(1.3,seconds);
Drive_Back_Left.stop();
Drive_Back_Right.stop();
Drive_Front_Left.stop();
Drive_Front_Right.stop();
Puncher.spinFor(forward,270,degrees);
Lift.spinFor(reverse,1,turns);
moveForward(30,1300);
Intake.spin(forward);
wait(1,seconds);
Intake.stop(hold);
Lift.spinFor(forward,.85,turns);

// ..........................................................................
}

/*----------------------------------------------------------------------------*
/
/* */
/* User Control Task */
/* */
/* This task is used to control your robot during the user control phase of */
```

```
/* a VEX Competition. */
/* */
/* You must modify the code to add your own robot specific commands here. */
/*----------------------------------------------------------------------------
/

void usercontrol(void) {
// User control code here, inside the loop
while (1) {

Drive_Front_Left.spin(forward, Controller1.Axis3.position(percent) +
Controller1.Axis1.position(percent), percent); // The code for driving with
the Front Left motor

Drive_Back_Left.spin(forward, Controller1.Axis3.position(percent) +
Controller1.Axis1.position(percent), percent); // Left Back Drive Code

Drive_Front_Right.spin(forward, Controller1.Axis3.position(percent) -
Controller1.Axis1.position(percent), percent); // Right Front Drive Code

Drive_Back_Right.spin(forward, Controller1.Axis3.position(percent) -
Controller1.Axis1.position(percent), percent); // Right Back Drive Code



if(Controller1.ButtonDown.pressing())
{Lift.spin(forward,100,pct);
Lift.setStopping(hold);
}


else if(Controller1.ButtonR2.pressing())
{ Intake.setVelocity(100,percent);
Intake.spin(vex::directionType::fwd, 100, velocityUnits::pct);
Intake.setStopping(hold);
}

else if(Controller1.ButtonR1.pressing())
{ Intake.setVelocity(-100,percent);
Intake.spin(vex::directionType::fwd, -100, velocityUnits::pct);
Intake.setStopping(hold);
}
```

```
else if(Controller1.ButtonUp.pressing())
{Lift.spin(vex::directionType::fwd, -100, velocityUnits::pct);
Lift.setStopping(hold);
}

else if(Controller1.ButtonX.pressing())
{Hook.set(true);
}

else if(Controller1.ButtonA.pressing())
{Hook.set(false);
}

else if(Controller1.ButtonLeft.pressing())
{Lock.set(true);
}
else if(Controller1.ButtonL1.pressing())
{Puncher.spin(vex::directionType::fwd, 100, velocityUnits::pct);
Puncher.setStopping(hold);
}

else
{
Intake.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);
Lift.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);
Puncher.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);
}

}

digital_out dig1 = digital_out( Brain.ThreeWirePort.A );
if( Controller1.ButtonB.pressing() ) {
dig1.set( true );
}
else {
dig1.set( false ); //Code for the Pneumatics
}

this_thread::sleep_for(10);
```

Project Program (Continued) Name Harris Perkins

Date 11/3/2023

```
// This is the main execution loop for the user control program.
// Each time through the loop your program should update motor + servo
// values based on feedback from the joysticks.

// ........................................................................
// Insert user code here. This is where you use the joystick values to
// update your motors, etc.
// ........................................................................

wait(20, msec); // Sleep the task for a short amount of time to
// prevent wasted resources.
}


//
// Main will set up the competition functions and callbacks.
//
int main() {
// Set up callbacks for autonomous and driver control periods.
Competition.autonomous(autonomous);
Competition.drivercontrol(usercontrol);

// Run the pre-autonomous function.
pre_auton();

// Prevent main from exiting with an infinite loop.
while (true) {
wait(100, msec);
}
}
```

Project Program (Continued) Name Harris Perkins

Date 11/3/2023

Competition Analysis:

Goals for today → Review and Discuss the results of the LCA Fall Robotics Tournament

Insights from Competiton	
Pros	**Cons**
Won Design Award	Couldn't make it over to hang often
Puncher was fast & efficient	Didn't have wings
Intakes worked decent	Auton program needed time to perfect
Structure was strong	The Hang Lock Mechanism was in open
Can climb barrier	Weak defensive play

Matches	Red Score	Blue Score	Alliance
Q-2	114	123	9364H
Q-7	29	197	9364C
Q-14	63	103	57711C
Q-21	98	100	42652B
Q-24	179	89	19859A
Q-31	82	125	16859D
QF 4.1	161	43	9364D
SF 2.1	110	96	9364D

*The highlighted scores represent our alliance that match.

Our team went to a competition at LakeWay Christian Academy this weekend, November 4th, 2023.

Overall the tournament served as a chance to find ways to improve the robot, our interviews, skills, and teamwork.

We plan to make some minor changes to the robot and focus on driver and programming skills because we learned the deep importance of skills run at this competition.

Moving forward we plan to get some scrimmages in between competitions.

Next Competition: December 2nd, 2023

New Goals:

Goals for today → Record our goals for the time before our next competition and who is responsible for completing the goals

Robot Goals:

- Improve atleast 3 subsystems of our robot •
- Manage wires and pneumatics better to prevent any malfunctions during matches • •
- Get a skills autonomous of over 100 points •
- Get working autonomous programs for both sides of the field during matches •

Practice Goals:

- Have one (1) scrimmage • • • •
- Practice a driver skills run of over ~~150 points~~ 150 points. •
- Practice Interviews • • • •

Competition Goals:

- Tournament Champion •
- Excellence Award • • • •
- Skills Champion • •
- Better scouting •

Notebook Goals:

- Include Program Testing more • •
- Include More Charts •
- Add tabs for organization and quick access •

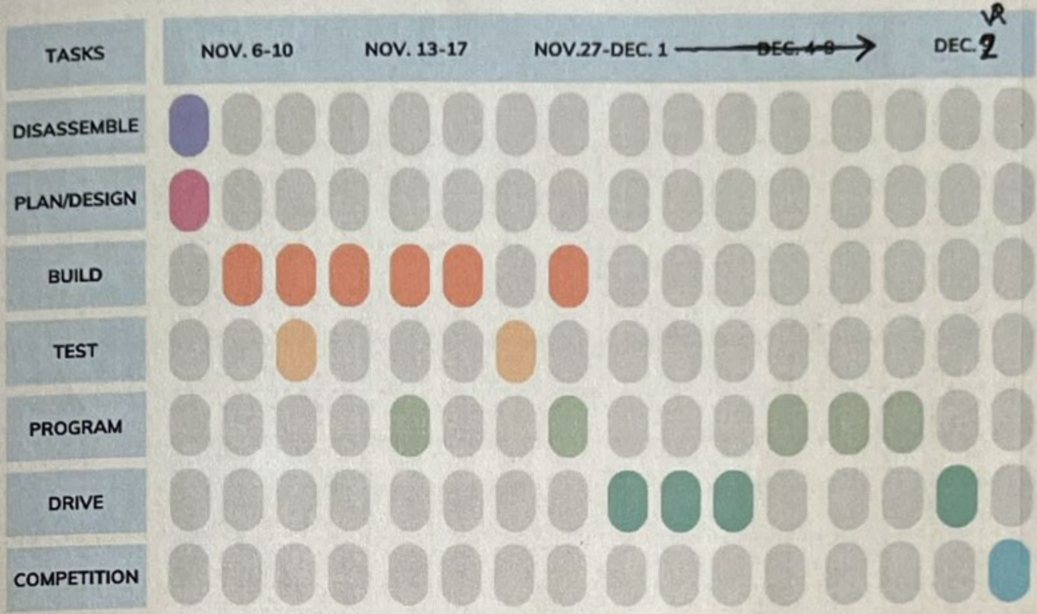
Responsibilities:

- Eli (Builder) •
- Owen (Driver) •
- Harris (Programmer) •
- Vanessa (Journalist) •

* The colored dot beside each goal correspond to which team member is responsible. Every color means a team effort and multiple means a collaboration.

Middle Season Timeline 2:

GANTT CHART
MIDDLE SEASON TIMELINE



Disassemble- Take apart only the necessary changes to robot #3 and reuse parts to save some budget

Plan/Design- Review the rules, research ideas, and decide on subsystems being redesigned

Build- Begin assembling the pieces

Test- Intermittently test what we have built and change as necessary

Program- Set up controls and autonomies

Drive- Practice learning controls and strategies as well as run full matches, scrimmages with teams, and skills

Competition- Final performance of the robot at Sullivan East Middle School

Brainstorming Intakes Redesign:

Goals for today → Analyze the performance of the intakes and look for ways to improve them

Problem/Challenge

Intakes are not entirely efficient all of the time.

Brainstorm

① Completely redesign intakes
② Take Inspiration from team 9364C's more horizontal intakes design at this past competition
③ Add ramp below intakes for better intaking, holding, and carrying of triballs

Select Approach

②

Take a more horizontal approach to our intake design

Build/Implement Solution

- Use current intake structure
- Change attachment angle to higher, more level, and farther out

Above is our plan for improving/redesigning the intakes. To further improve the intakes, we took the problem from brainstorming idea #3 and made a solution.

The solution was to move the rubberband on the front of the base up one hole.

VP

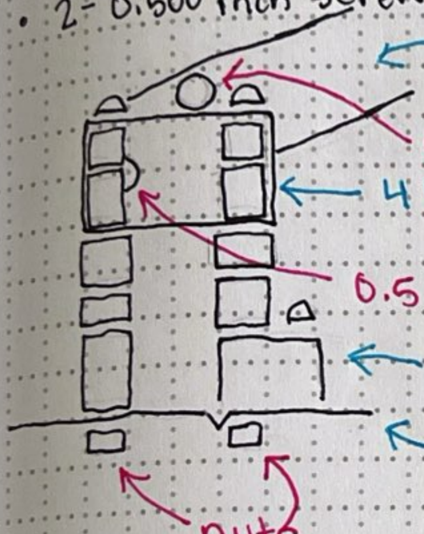
Working On Intakes:

Goals for today ⟶ Get intakes remounted better

Parts List:
- 2 - 4 long aluminum c-channels
- 10 - 0.500 inch spacers
- 8 - 0.375 inch spacers
- 4 - 0.125 inch spacers
- 4 - 2.000 inch screws
- 2 - 0.750 inch screws
- 8 - Nylock nuts
- 2 - 0.500 inch screws

(1) Detach intake from front of robot and set aside. Remove and discard the 1by1 L-channel that the intake was previously attached to.

(2) Follow the below diagram to add the four (4) long c-channels to the top of the outside base.



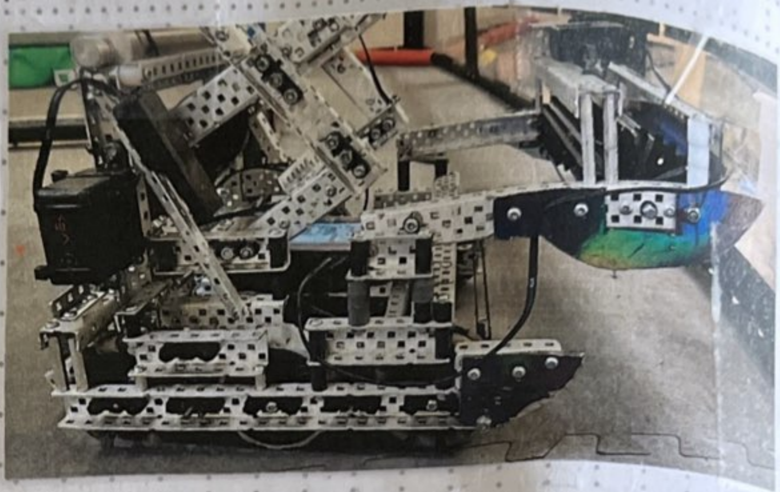
*This half inch screw is loose to allow it to pivot up as it picks up triballs or drives into the goals.

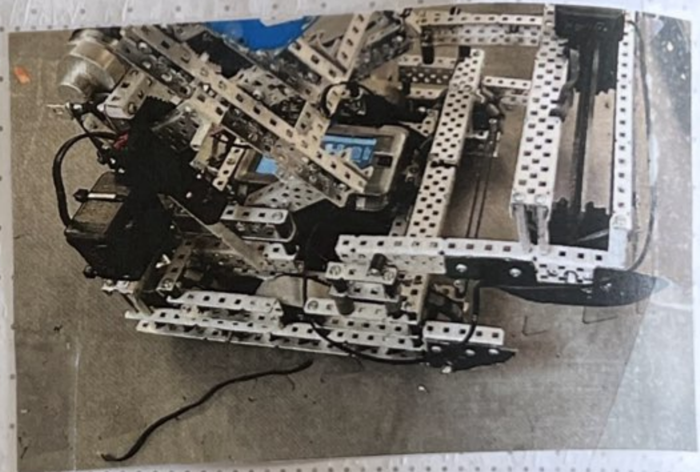
Below is a diagram of the intake c-channel with the specific screw holes labeled.

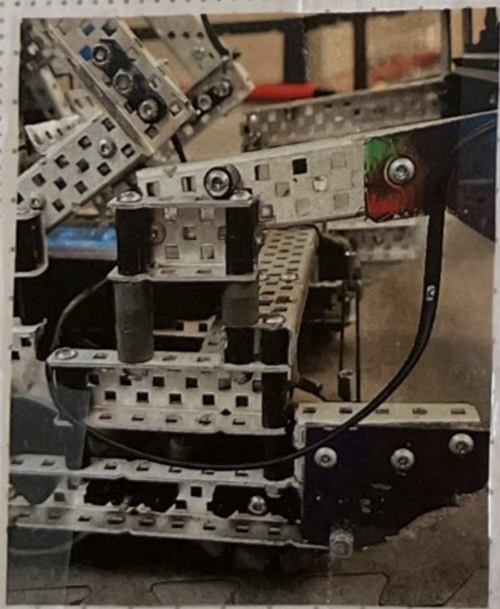
Finished Intakes Picture:

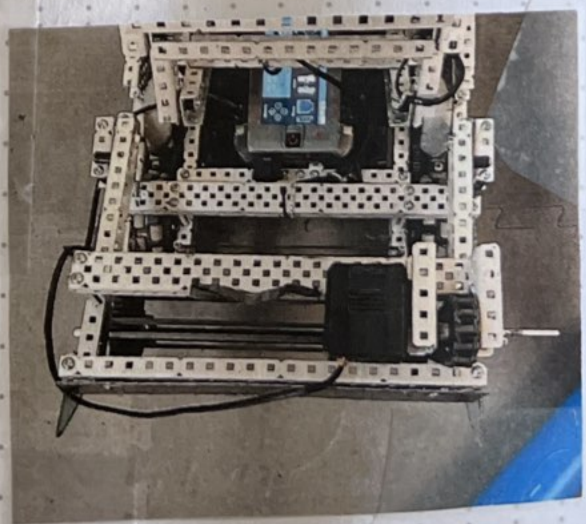
Below are the images of the finished intakes There are above and side angle images of all the changes we made on page 61.

VP









Brainstorm Wings Redesign:

Goals for today → Brainstorm New Wing Designs
→ Select Design to build and attach to robot

Problem/Challenge
Our last wings were not space efficient or sturdy in matches.

Brainstorm

1. 9364C Style Wings
   - Double Acting Pistons
   - Slanted Lexan wedges on front
   - overcenter lock
   - attached at front of robot
2. Vertical Wings
   - Fold down not out
   - Single Acting pistons
3. Simple Horizontal Wings
   - Attached at back of robot

Select Option
①
Build new wings that were inspired by team 9364C.

Build/Implement Solution
- Build wing structure
- Test and attach an overcenter lock
- Attach pistons
- Test performance

(Located on page 64-65.)

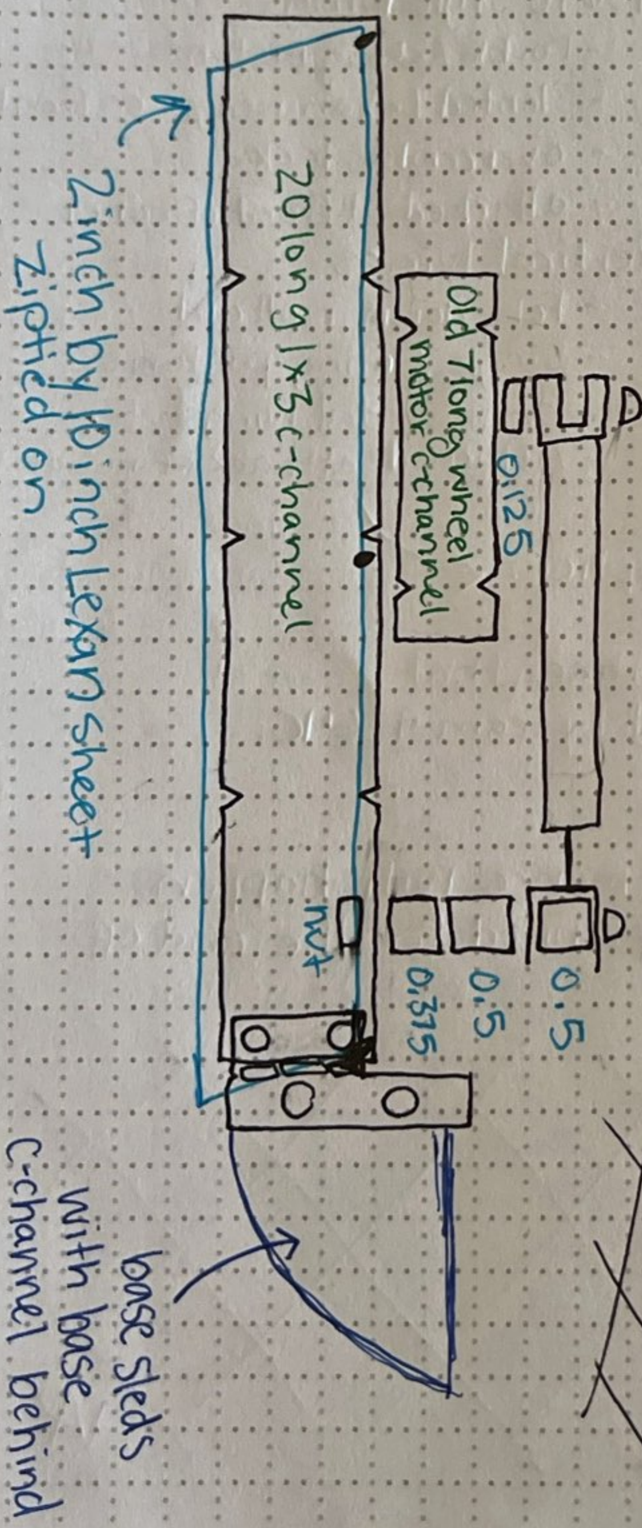
☆Test☆ The major test of performance will happen at a scrimmage with AGSI planned for the end of November.

VP

Project Brainstorm Wings Redesign Name Vanessa Perkins
Date 11/8/2023

Build Wings:

Goals for today ⟶ build wing structure
⟶ Attach all components
⟶ Test wings function



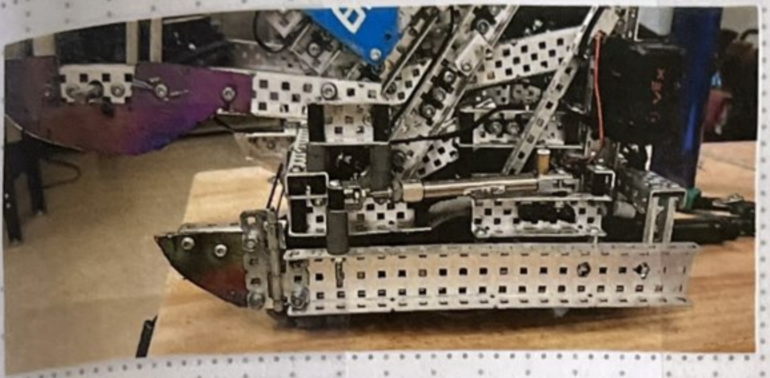
1. Gather 20 long 1x3 aluminum c-channel.
2. Cut two (2) holes off of the left side of a steel hinge as depicted in the lower left corner.
3. Attach the 3 long end of the hinge to the 1x3 c-channel using two (2) 0.5 inch screws and nylock nuts. Attach the 5 long end of the hinge to the base sleds in the same way.
4. Attach pneumatic piston to wings using spacing depicted below and a 2.500 inch screw with nylock nut. Attaches 3rd hole back from front.
5. Attach other end of piston to base using another 2.500 inch screw, nylock nut, and depicted spacing.
6. Attach Lexan sheet at an angle to to create the slanted wedge.

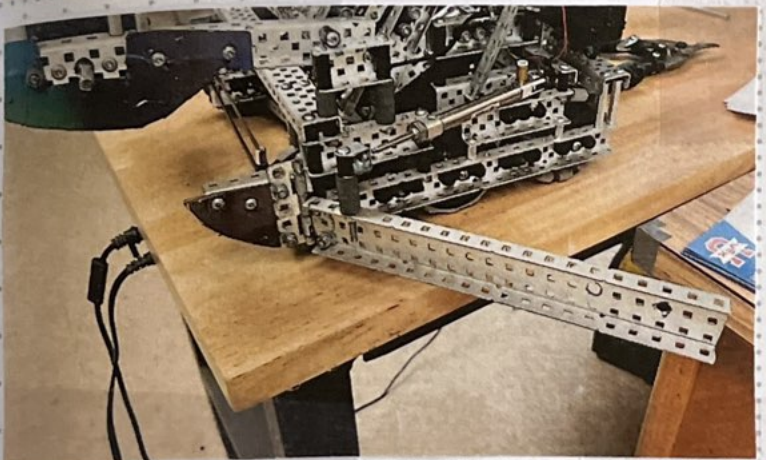
VP

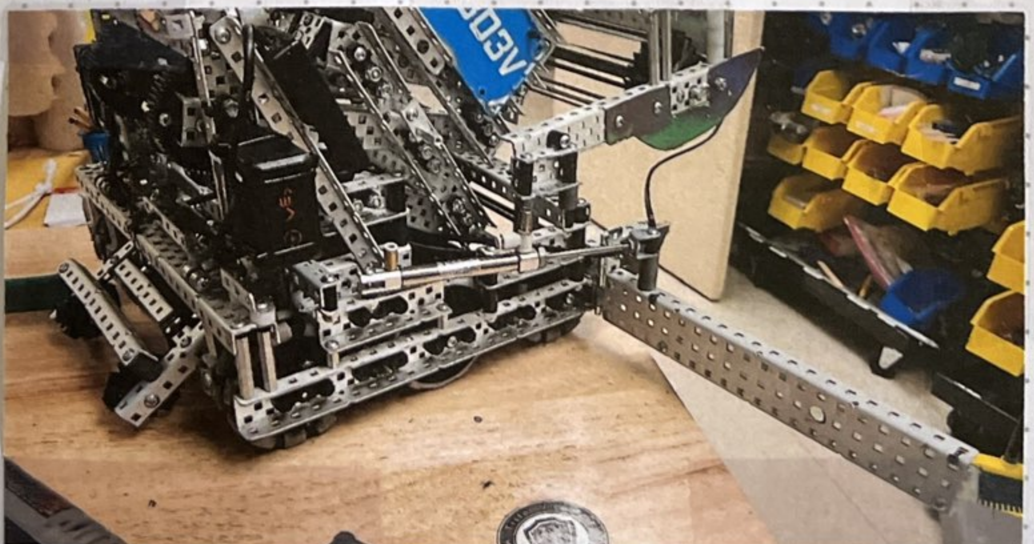
Project Build Wings
Date 11/9/2023
Name Vanessa Perkins

(Continued)

Below are images of the wings before we added Lexan so you can see the structure better. To the right is the complete parts list.



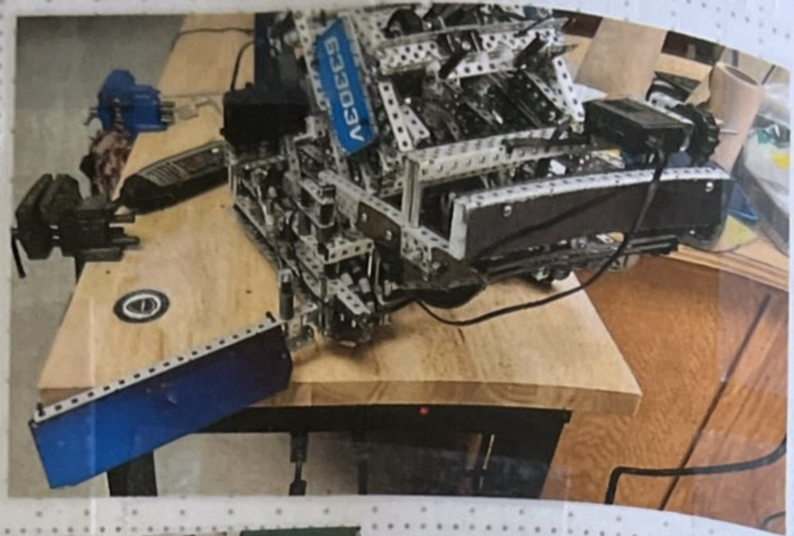


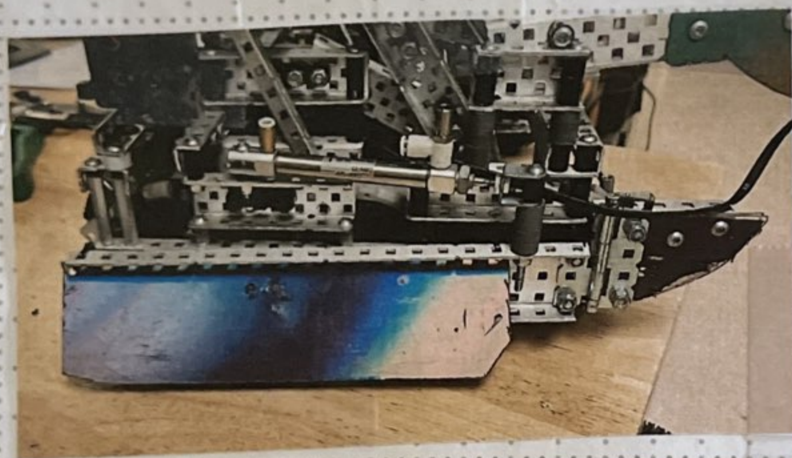


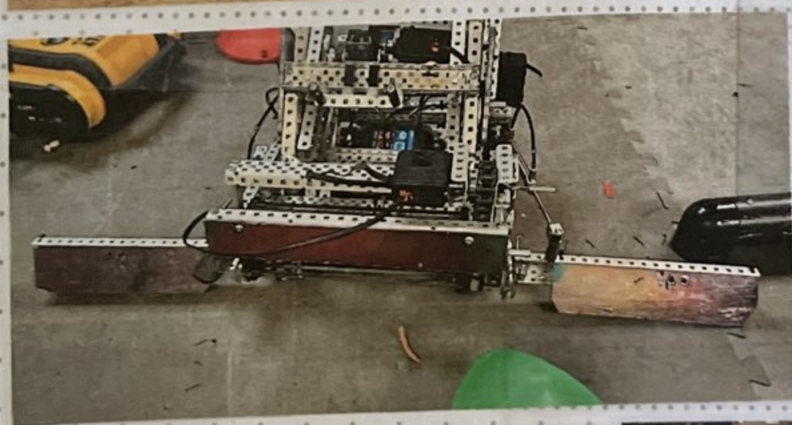
VP

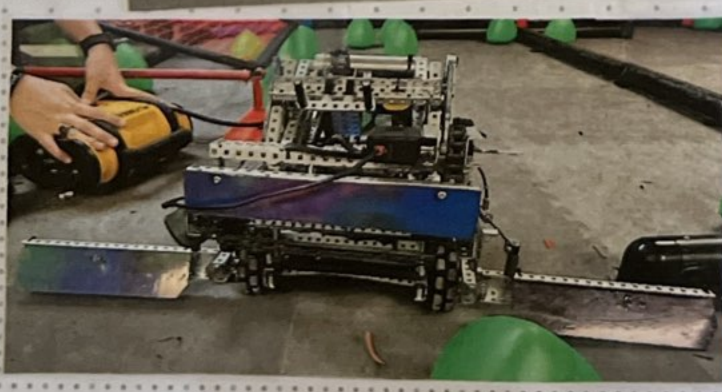
Finished Wings Pictures:

Below are the images of the finished wings from several different angles and positions to see all the components and the movement of the wings.



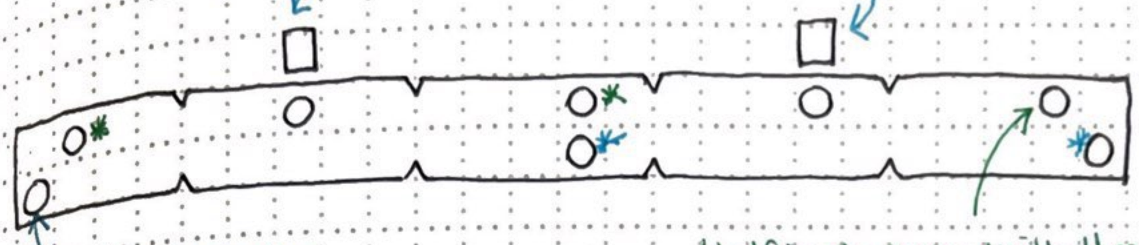






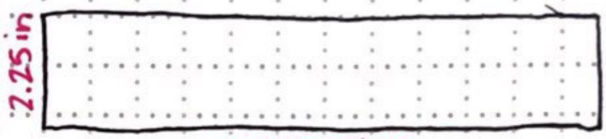
Building Back Wedge:

Goals for today → Attach Wedge sheet to the back of robot for defence and pushing triballs into goal



Back Wedge Parts List:

- 1 - 24 hole long aluminum c-channel
- 2.25 by 11.75 inch Lexan sheet
- 9 - 0.500 inch screws
- 3 - 4mm spacers
- 8 - 8mm spacers
- 4 - 1.000 inch screws
- 10 nylock nuts



Above is a diagram of how to build the back wedge. There is also a diagram of the Lexan sheet that is attached to the C-channel by three (3) half inch screws to create the wedge.

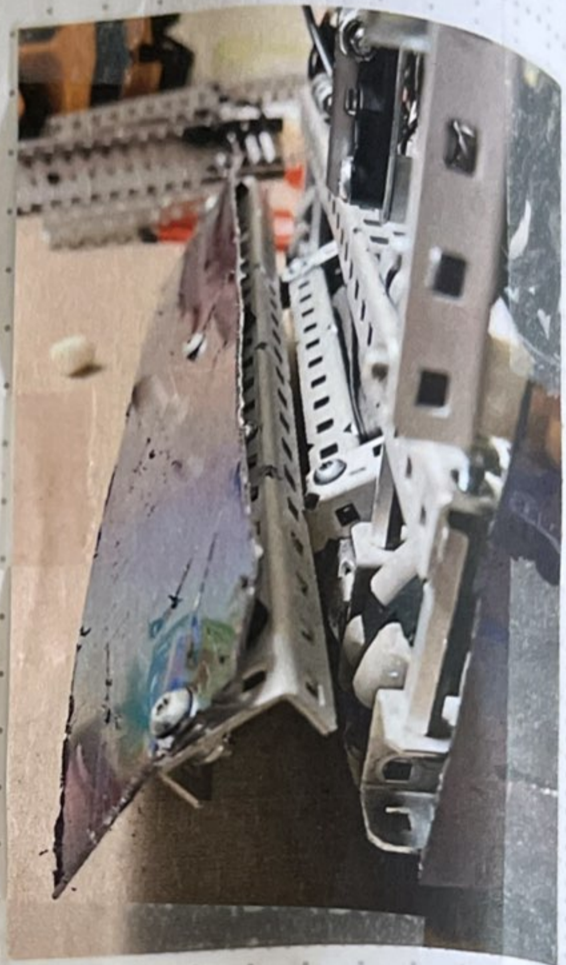
To the left is the complete parts list for the back wedge. The finished images are on page 68.

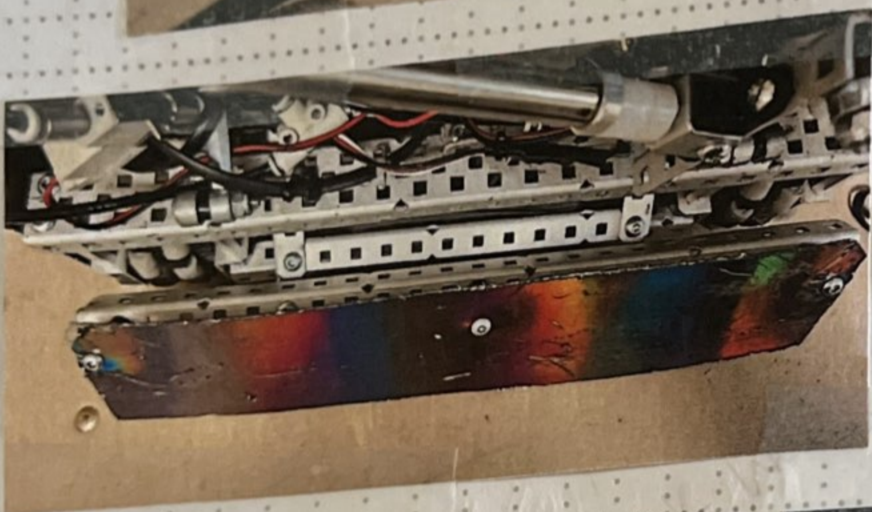
Owen did some test driving and found success in being able to push triballs into the goal with the wedge. At the scrimmage we can continue to test the function of the wedge.

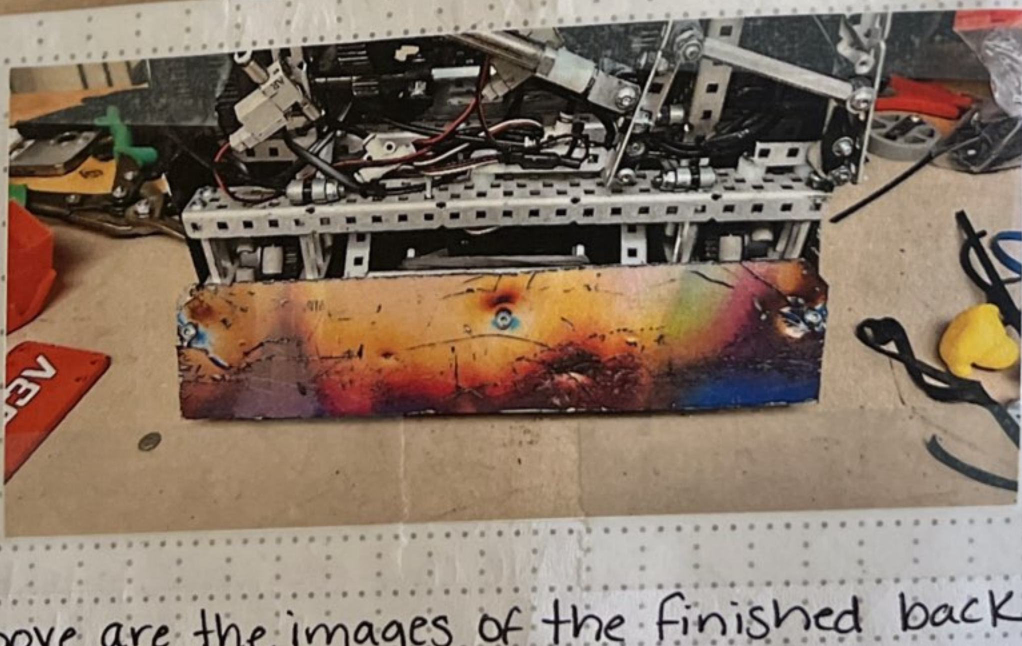
VP

Finished Back Wedge Images:

Goals for today → Finish documenting the back wedge images







Above are the images of the finished back wedge from various angles to see all of the details.

AGSI Scrimmage:

Goals for today → Build relationships and alliances
→ Test the robot and new strategies

We invited two AGSI, 57711, teams to our school today to get in some practice and testing before our next competition. Below is some information from this scrimmage.

Strategies to Test:

1. No Matchloading. Just throw over triballs on field to our goal.
   - When tested, this strategy kept the score low and very close. It might work better if we have an alliance instead of one on two.
2. Launch Half of the Matchloads. Score. Then Launch rest.
   - This strategy did not allow us to efficiently use our wings or back wedge to score several triballs all at once.
3. Roll Matchloads Off Front of Robot into Otherside of Field.
   - This strategy might have worked, but we did not score them quick enough so 57711X launched them back.

Subsystems to Test:

1. Wings.
   - The wings even with the overcenter lock were not very sturdy.
2. Back Wedge.
   - The wedge was very useful for scoring triballs.
3. Having no Matchload Zone Arm.
   - When 57711X played defense on us while matchloading, it took us over 50 seconds to launch all 22 triballs and most were inaccurate.
   - We learned our arm is crucial to performance.

Takeaways:

1. Strategies matter.
2. We need to switch to more stable vertical wings again.
3. We need the Matchload Zone Arm to withstand defensive robots.

Project AGSI Scrimmage
Name Vanessa Perkins
Date 11/22/2023

Hanging Mechanism Brainstorming:

Goals for today → Decide what to do about the hang mechanism

Problem/Challenge

We have not improved the hanging mechanism and it's exposed to damage on side of robot.

Brainstorming

① Design more enclosed hanging mechanism.
② Cover current hanging mechanism
③ Make higher tier hang.

Select Approach

②

Due to limited time, resources, and space on the robot, we will be keeping the current hanging mechanism but adding a Lexan cover.

Build/Implement Solution

- Keep Current Hang Mechanism
- Cut out and attach Lexan cover on each side of robot

→ The Instructions for the Hanging Mechanism and its lock are on pages 26-27 and pages 45-46. The cover portion of the solution is located on the next page with instructions and pictures.

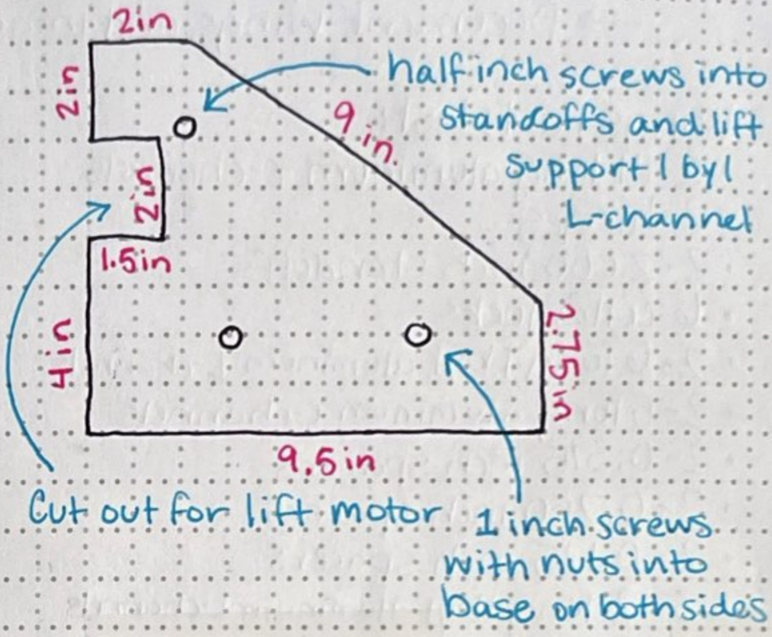
VP

Attaching Side Panels:

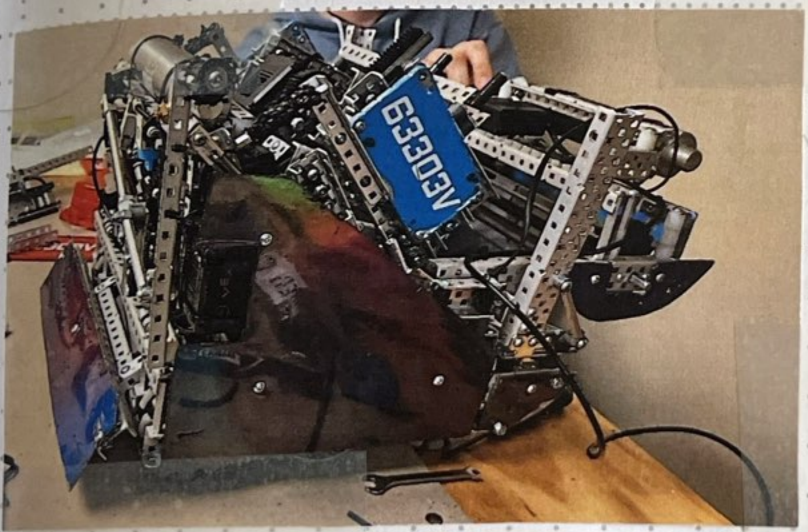
Goals for today → Add protection panels to hang mechanism

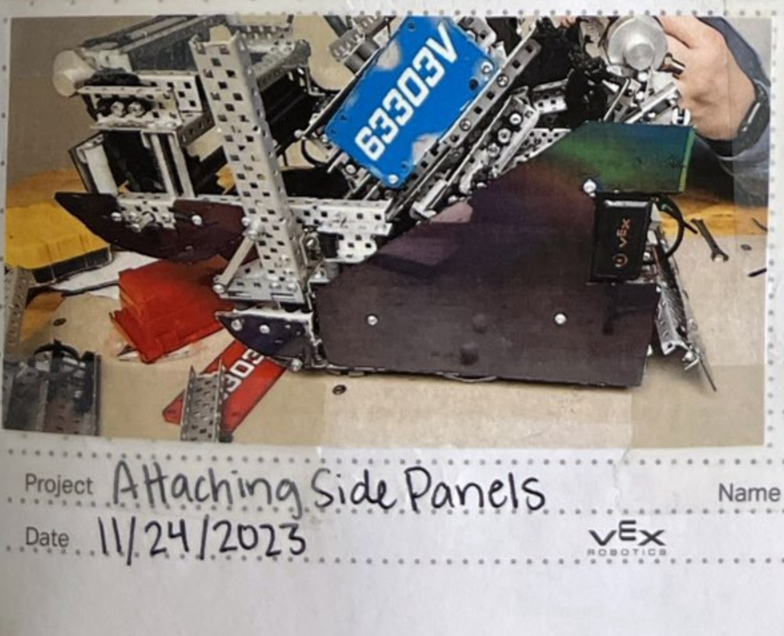
Side Panels Parts List:

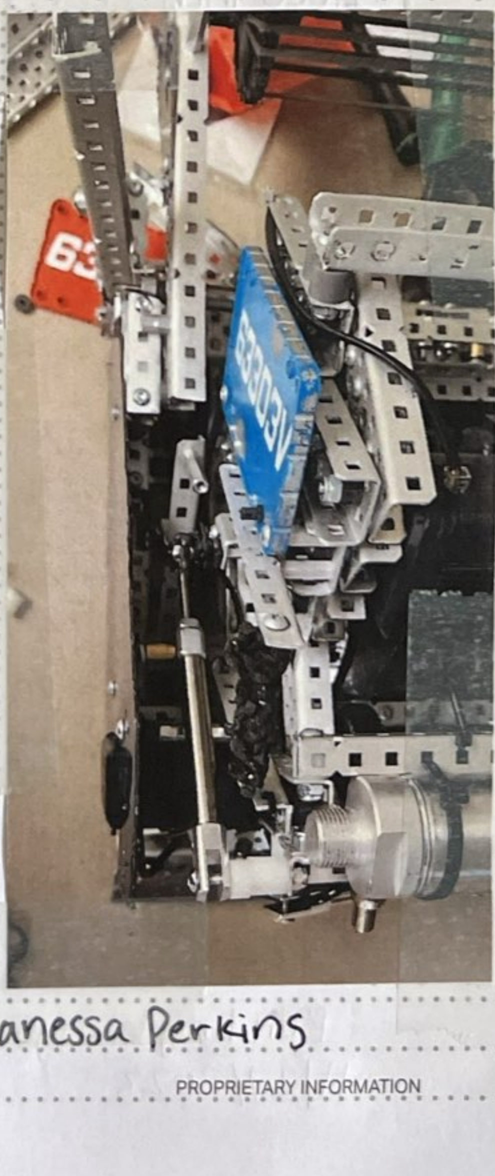
- 2-2.000 inch standoffs
- 4-1.000 inch screws
- 4 nylock nuts
- 4-0.500 inch screws
- Lexan Sheet x2 (Dimensions to the right)



To the right is the labeled dimensions with all attachment points and the finished pictures located below.







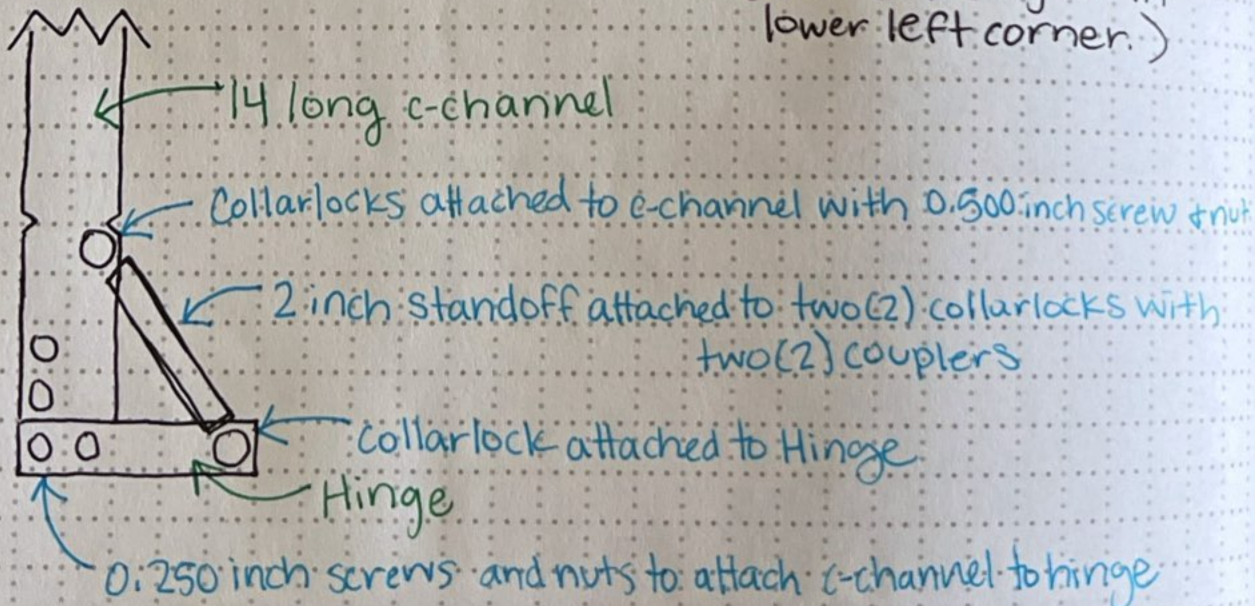
Building Wings:

Goals for today → Get wings built and attached
→ Test wings performance
→ Document wings and images

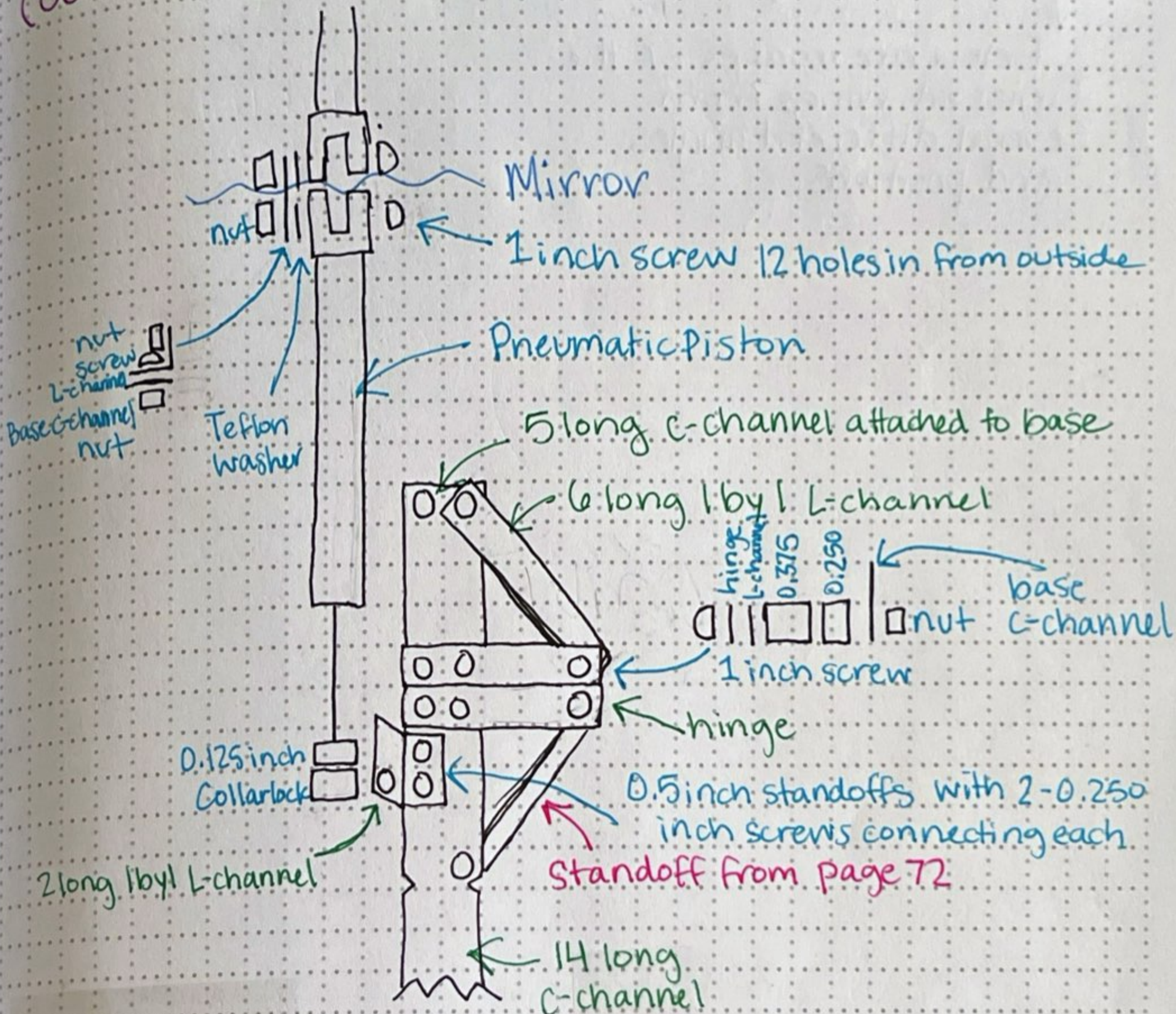
Wings Parts List:

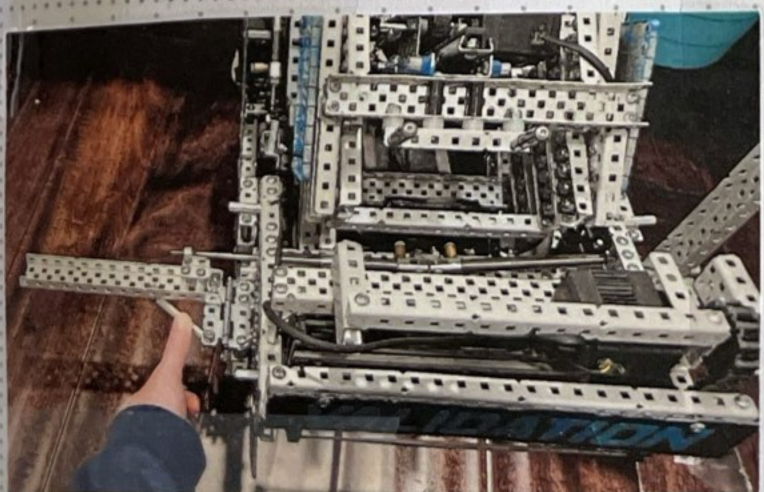
- 2-14 long aluminum C-channels
- 2-hinges
- 2-2.000 inch standoffs
- 6 collarlocks
- 2-6 long 1by1 aluminum L-channels
- 2-5 long aluminum C-channels
- 2-0.375 inch spacers
- 2-0.250 inch spacers
- 2-0.125 inch spacers
- 3-2 long 1by1 aluminum L-channels
- 4-1.000 inch screws
- 10-0.500 inch screws
- 12-0.250 inch screws
- 2 Single-Acting Pneumatic Pistons
- 16 nylock nuts
- 4 couplers

(Picture of diagram in lower left corner.)



(Continued)

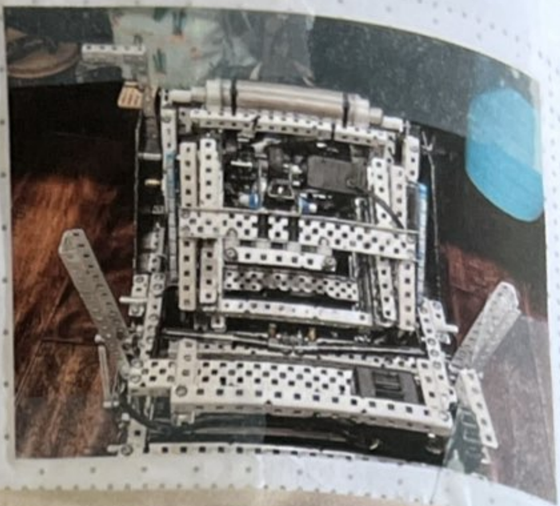


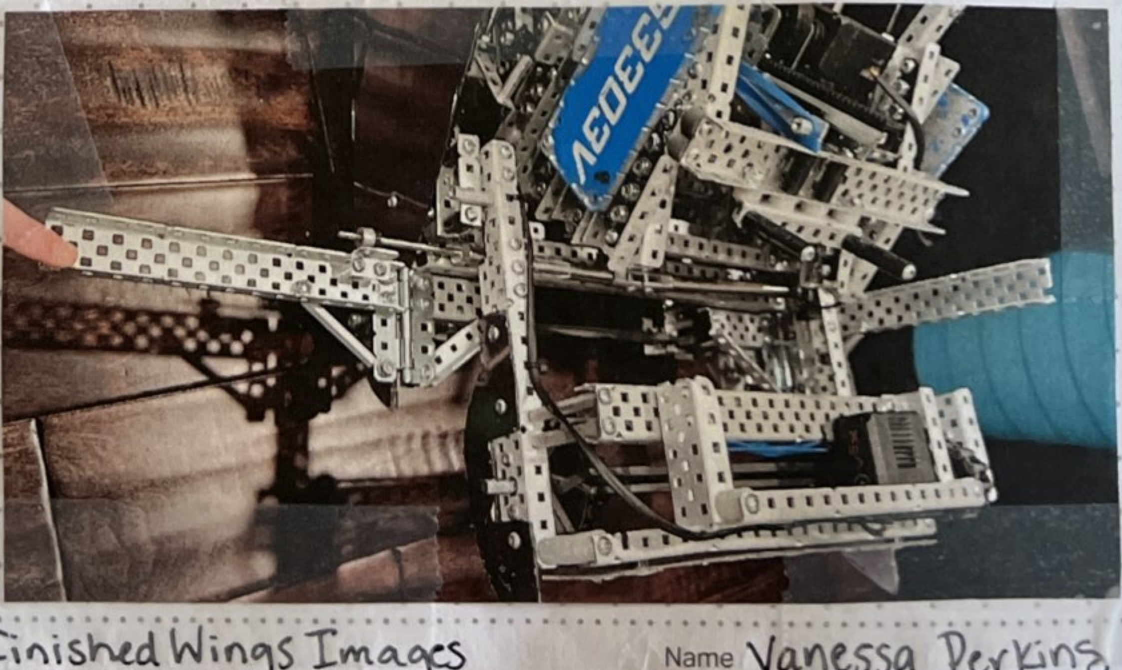


Above is the diagram for the right side wing. The left side is mirrored exactly.
To the left is an image of the wing folded out.

Finished Wings Images:

Below are images of the finished wings from Several different angles and positions.





Building Matchload Zone Arm:

Goals for today → Build and attach matchload mechanism
→ Test performance of mechanism

Matchload Zone Arm Parts List:

- 15 nylock nuts
- Single-acting Pneumatic Piston
- 4 - 0.375 inch spacers
- 2 - 0.250 inch spacers
- 2 - 1.500 inch screws
- 4 - 1.000 inch screws
- 1 Pillow Block bearing
- 2 - 0.125 inch spacers
- 3 - 8mm spacers
- 2 collar locks
- 2 Hex nut Retainers
- 4 Teflon Washers
- 1 Bearing Flat
- 8 - 0.500 inch screws
- 2 - 1.000 inch standoffs
- 2 - 0.500 inch standoffs
- 7 - 0.250 inch screws
- 1 rubberband
- 1 - 6 long 1by1 aluminum L-channel
- 1 - 3 long 1by1 aluminum L-channel
- 1 - 14 long 1by1 aluminum L-channel
- 1 - 5 long 1by1 aluminum L-channel
- 1 - 2by2 rounded steel plate triangle
- Lexan hook
- 2 zipties
- 2 Couplers

☆ Use the six(6) and three(3) long 1by1 L-channels attached to base from previous arm mechanism. ☆

- 1 - 2.000 inch standoff

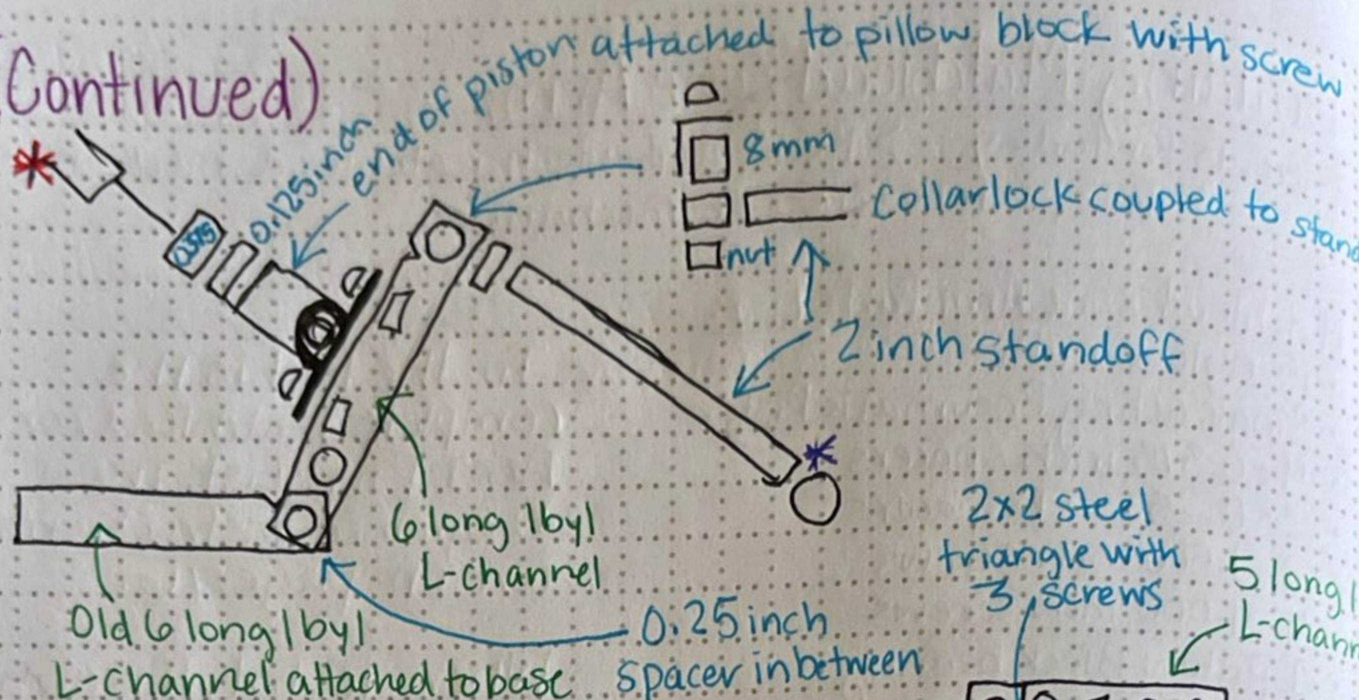
To the left is the Complete Matchload Zone Arm Parts List. The Diagramed instructions are on page 76.

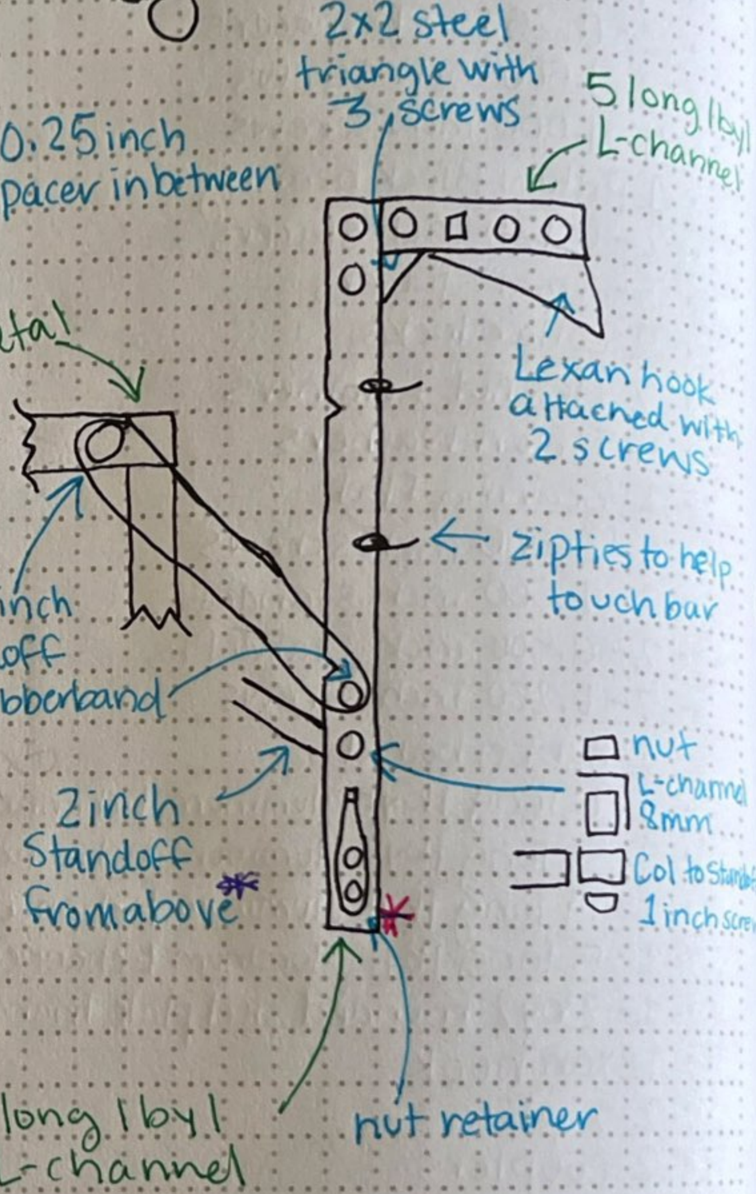
When tested, this arm allowed us to launch triballs more efficiently without being pushed by defensive robots.

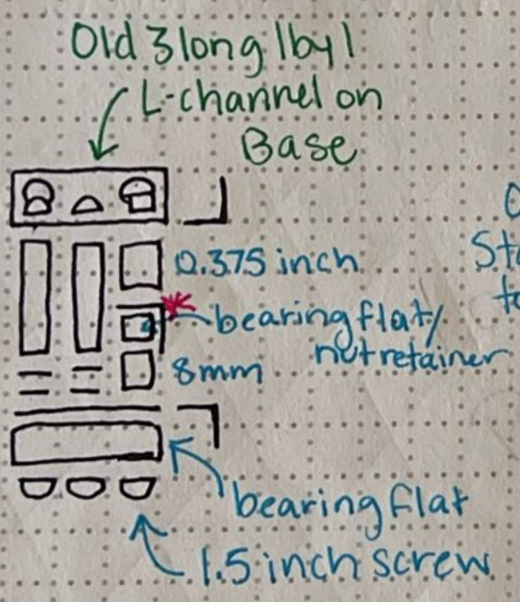
VP

Project Building Matchload Zone Arm
Name Vanessa Perkins
Date 11/28/2023

(Continued)







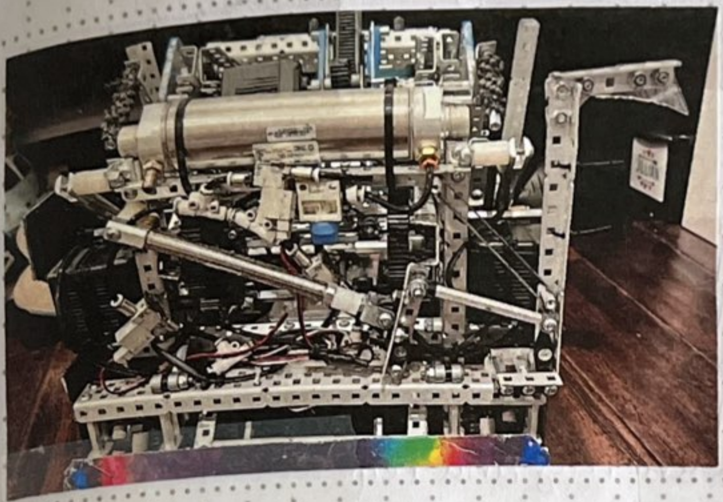
Above is Arm Diagram with parts list on previous page.

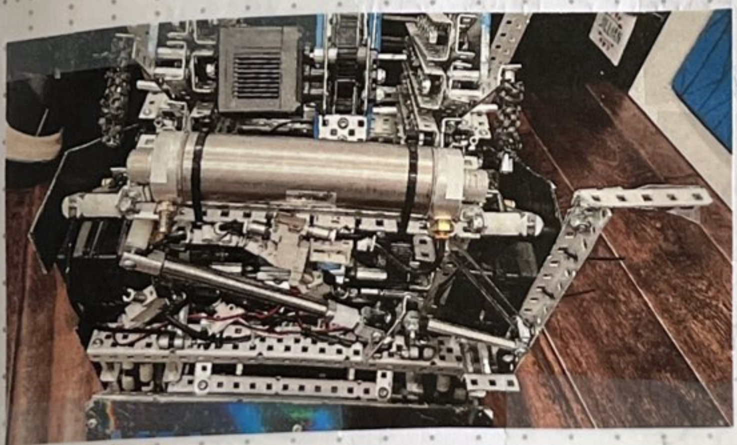
* Same 1by1 L-channel. Pivot Point.
* Piston attached to Lift Support 3 holes down with 1.5 inch screw, 0.375 and 0.25 inch spacer and nut.

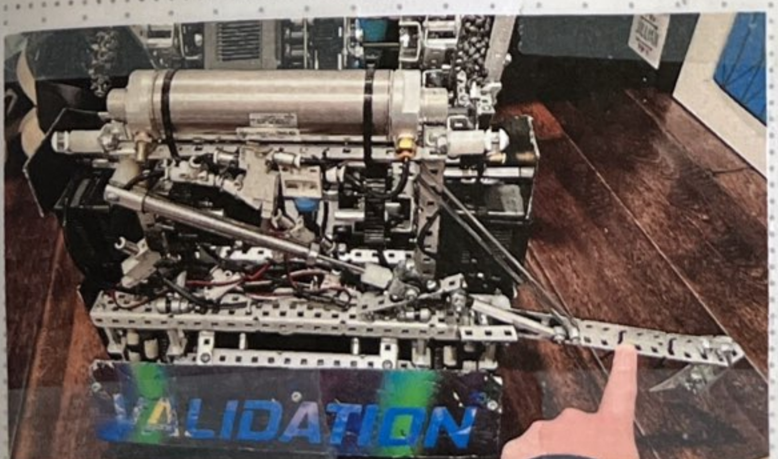
Project Building Matchload Zone Arm (continued) Name Vanessa Perkins
Date 11/28/2023

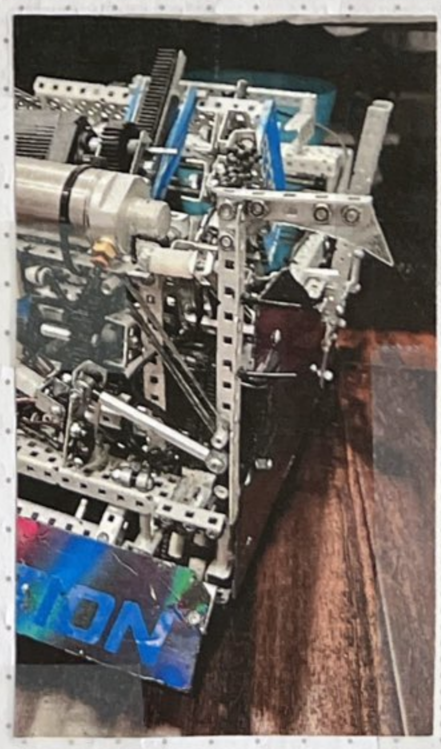
Finished Matchload Zone Arm Images:

Below are the Finished images of the Matchload Zone ArmMechanism.









VP

Skills Runs:

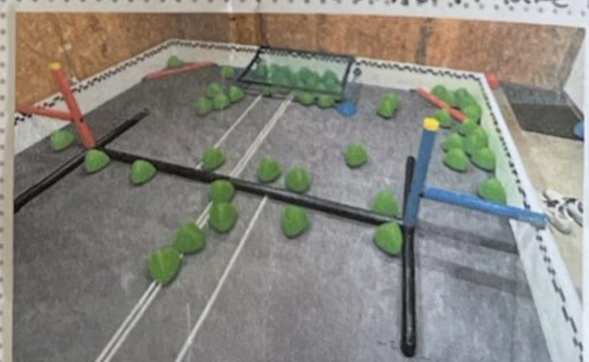
Goals for today → Practice Skills Strategies for Driver

Skills Run 1: 90 points
- Was not able to get many under goal due to short time.
- Improve by faster matchloading.

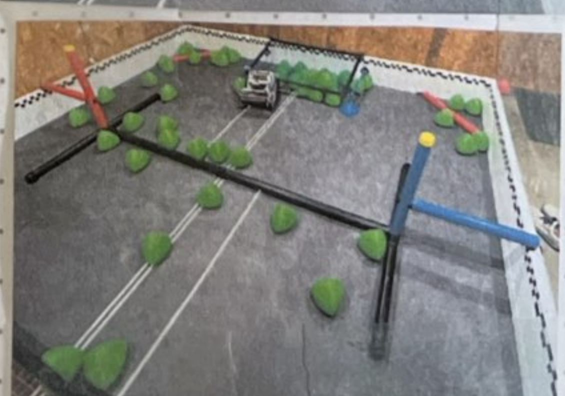
Skills Run 2: 114 points
- Better launch angle than first run so triballs don't get stuck in corner

Note: We found that when using five (5) of the Plastic Mill Brand Black Rubberbands, the triballs cross the barrier decently. However using four (4) of the Alliance Size 64 (same size as black) Blue Rubberbands, there is more elastic potential energy and requires one less band to launch across the barrier thus saving resources. Adding more Alliance brand rubber bands only makes it slower.

Skills Run 3: 130 points
- Aimed launch towards corner like Run 1 which make it difficult to push out of the corner
- End results to the right on top.



Skills Run 4: 115 points
- Aimed launch towards goal which made them bounce and scatter badly
- All skills runs use all matchloads provided.
- End results to the right on bottom.



Project Skills Runs
Date 11/29/2023

Name Vanessa Perkins

Programming Wedges:

Goals for today → Work on Skills/Match Autons
→ Make Adjustments to Robot to improve programming

Problem/Challenge

In programming we want to be able to push three(3) triballs over the barrier at once.

Brainstorm

1. Add wedges to wings
2. Use Matchload zone Arm
3. Push all three (3) with Intake

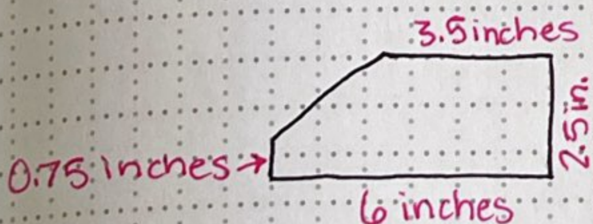
Build/Implement Solution

- Attach Lexan sheets at angles to the wings
- Build located below

Select Option

1

Add wedges to wings to help push triballs up and over barrier.



We attached two(2) three(3) long aluminum 1by1 L-channels to the wings using four(4) screws and nuts. Then the above Lexan shape was ziptied and rubberbanded down to the wing to create and angled wedge.

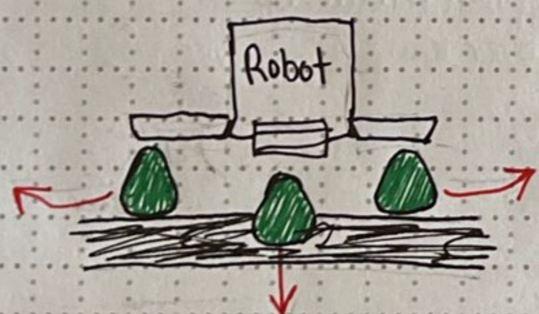
Project Programming Wedges
Name Vanessa Perkins
Date 11/30/2023

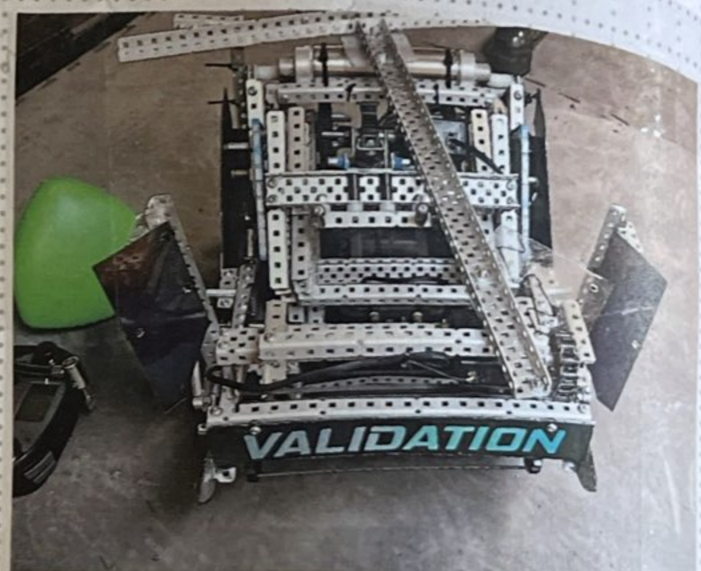
Test Programming Wedges:

Goals for ~~today~~ → Test Wedges on wings
→ Make changes or adjustments as needed

We tested the new wedges on the field with three (3) triballs against the barrier. We drove forward at full force.

Results: The Center triball being pushed by the intake made it over the barrier however the other two (2) being pushed by the wedges just slid out to the sides without crossing the barrier.



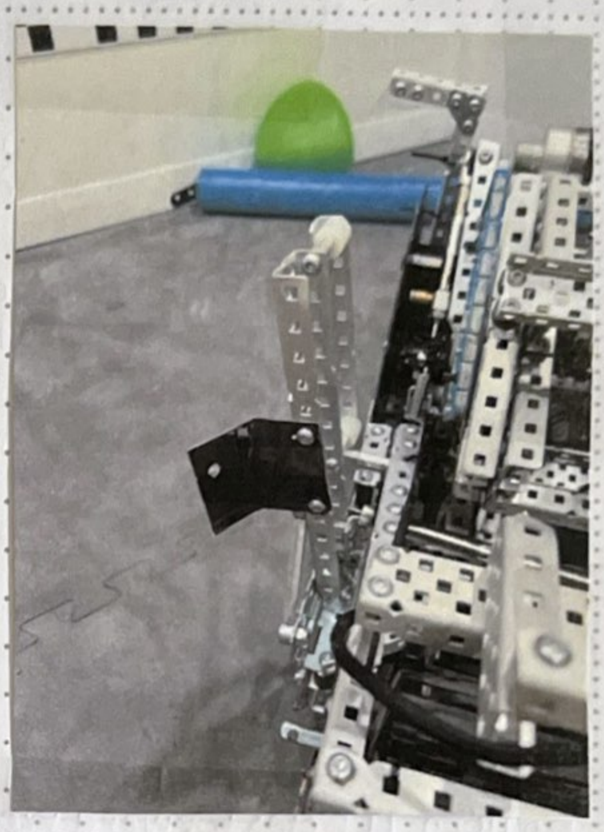


Above is a diagram to show the results. To the right are more images of the wedges to show that the results were because the wedges don't go all the way to the ground. But they can't since the wings are vertical and would make the robot too wide.

Solution: Remove wedges.

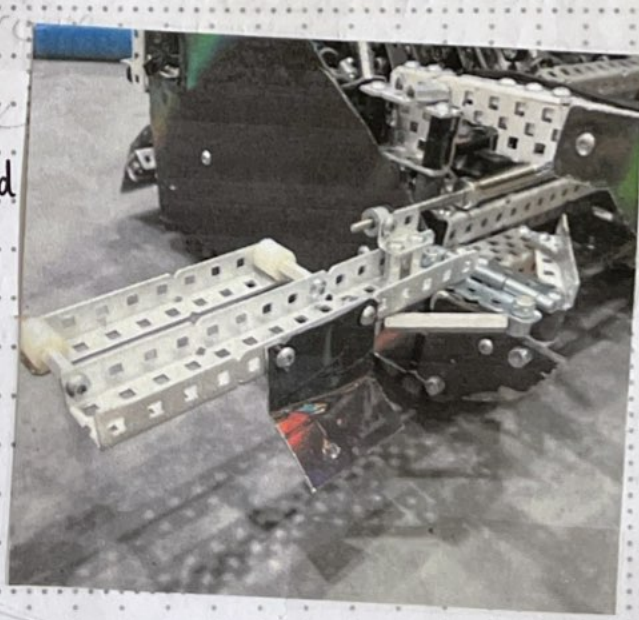
Improving Wings:

Goals for today → Make Wings more functional



Since the wedges didn't work, we added a bent Lexan rectangle to the right side wing to assist in programming during our attempt to get the win point by scooping the triball out of the matchload zone.

Then we added a nine(9) hole long aluminum 1by1 L-channel to the back of the same wing using two (2) half inch standoffs and two(2) half inch spacers. We used two(2) 0.250 inch screws and two (2) 0.750 inch screws.



This bar allows us to elevated on the barrier and ~~barrierand~~ VP touch our alliance elevation bar so it counts while allowing our alliance to also elevate at a higher tier.

This gives us multiple strategy options during matches.

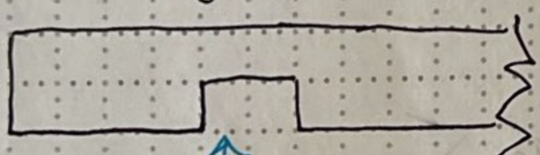
VP

Improving Intakes:

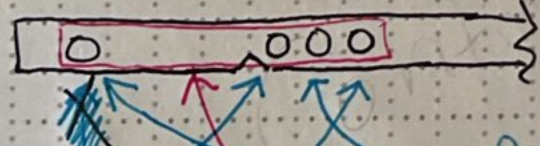
Goals for today → Make the intakes stronger

We discovered that since adding the wings with the pistons running through the center, the intakes were not able to lower completely so we had to cut a piece out and reinforce to make up for it.

To the right is an image of this improvement and the new bracing. Below are the instructions.



Cut 2 holes out of the bottom 4 holes forward from back on both sides of the intake



0.25 inch screw, 0.5 inch standoff, 8mm spacer, 0.5 inch screw

~~0.25 inch screw and nut~~

0.25 inch Screw and nut

7 long 1 by 1 L-channel goes inside intake C-channel.

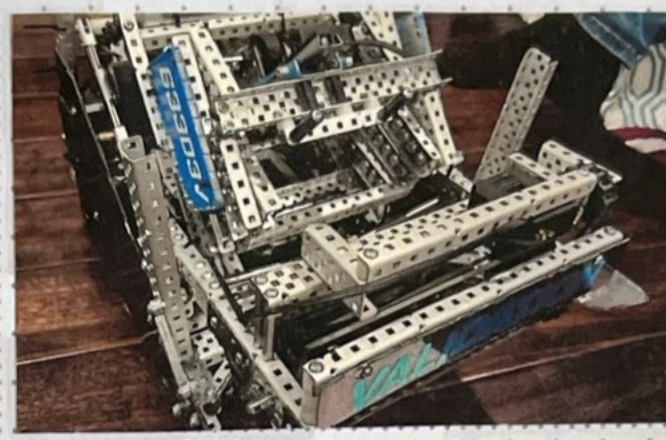
Mirror for each side.

Bracing Parts List:

- 2 - 7 long 1 by 1 L-channels
- 4 - 0.500 inch standoffs
- 4 - 8mm spacers
- 18 - 0.25 inch screws
- 4 - 0.500 inch screws
- 6 nylocknuts

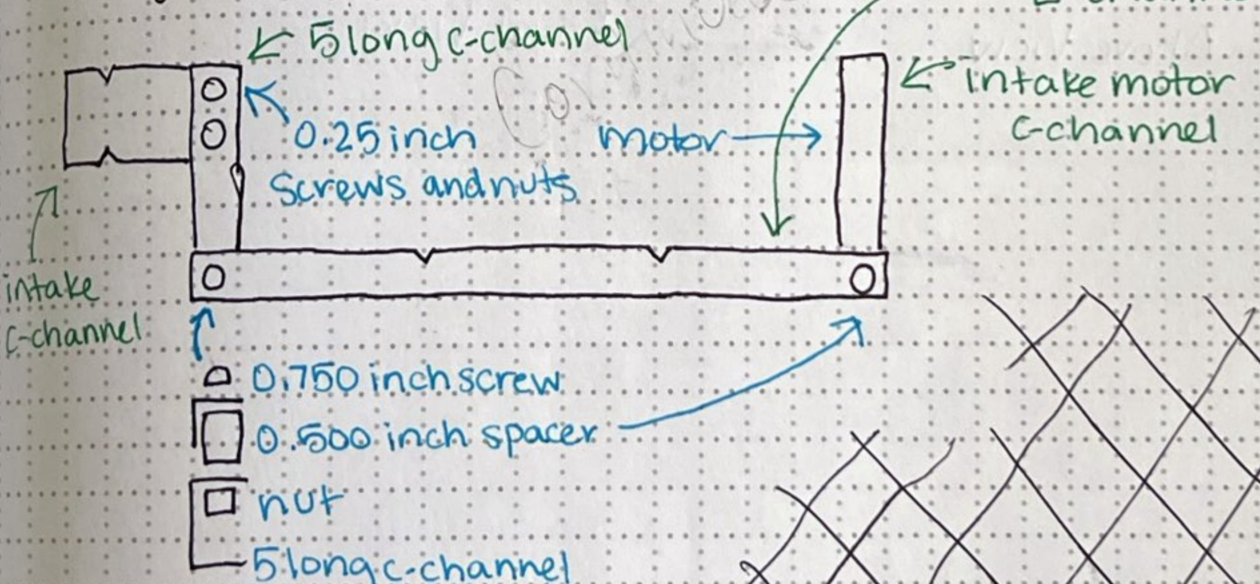
(continued)

To further increase the strength of the intake we added extra bracing on top.
To the right is the image and below are the instructions.



Top Intake Support Parts List:

- 1 - 5 hole long aluminum c-channel
- 1 - 15 hole long aluminum 1 by 1 L-channel
- 2 - 0.500 inch standoffs
- 2 - 0.750 inch screws
- 2 - 0.250 inch screws
- 4 nylock nuts

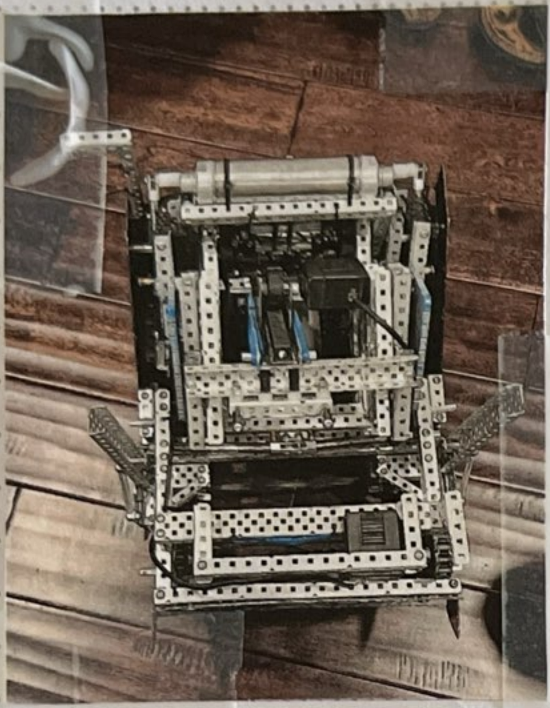


VP

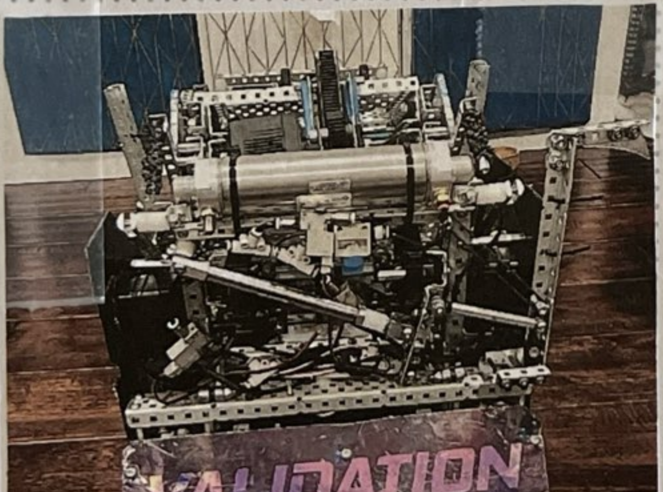
Full Robot Images:

Goals for today → Document complete robot from all angles and positions to record progress

Front View



Above View



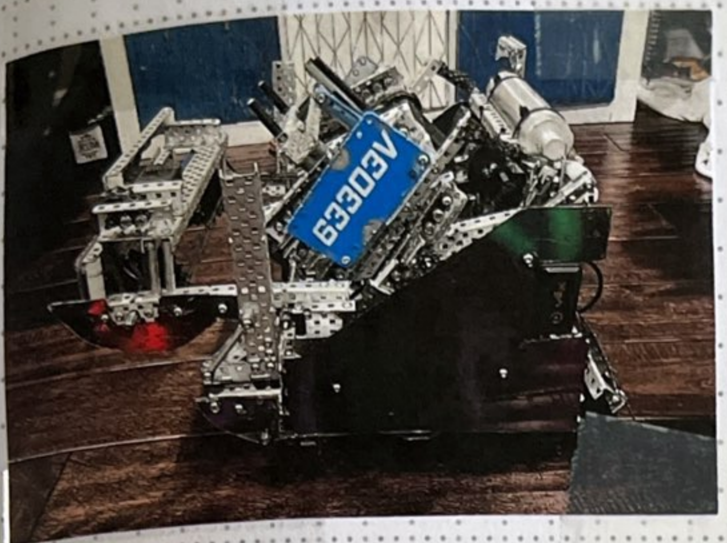
Back View

Right Side View

Project Full Robot Images
Date 12/1/2023

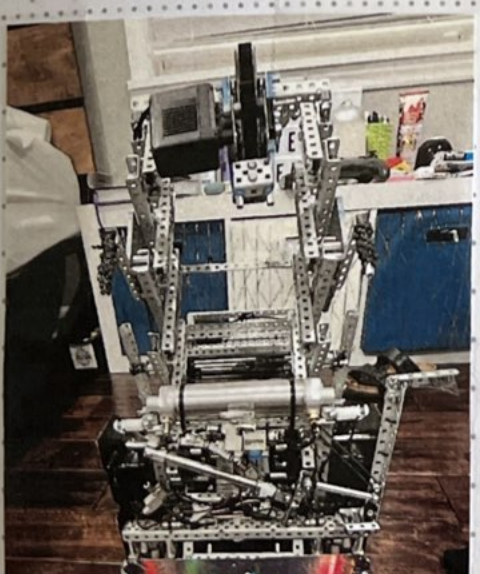
Name Vanessa Perkins

(Continued)

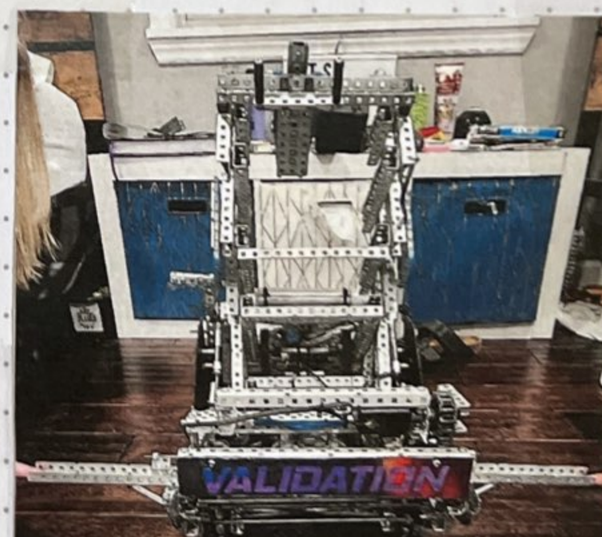


Left Side View

Left Side View, Raised Lift



Back View, Lift Raised



Front View, Extended Wings

Project Full Robot Images (Continued) Name Vanessa Perkins
Date 12/1/2023

```
#include "vex.h"

// ---- START VEXCODE CONFIGURED DEVICES ----
// Robot Configuration:
// [Name]               [Type]          [Port(s)]
// Controller1          controller
// Drive_Front_Left     motor           8
// Drive_Back_Left      motor           7
// Drive_Front_Right    motor           11
// Drive_Back_Right     motor           15
// Intake               motor           19
// Lift                 motor_group     20, 9
// Puncher              motor           10
// Lock                 digital_out     B
// Hook                 digital_out     C
// WingLeft             digital_out     D
// WingRight            digital_out     E
// Inertial             inertial        17
// ---- END VEXCODE CONFIGURED DEVICES ----

using namespace vex;
competition Competition;

/*-----------------------------------------------------------------------------
---*/
/*                          VEXcode Config
*/
/*
*/
/*  Before you do anything else, start by configuring your motors and
*/
/*  sensors using the V5 port icon in the top right of the screen. Doing
*/
/*  so will update robot-config.cpp and robot-config.h automatically, so
*/
/*  you don't have to. Ensure that your motors are reversed properly. For
*/
/*  the drive, spinning all motors forward should drive the robot forward.
*/
/*-----------------------------------------------------------------------------
---*/

/*-----------------------------------------------------------------------------
---*/
/*                          JAR-Template Config
*/
/*
*/
/*  Where all the magic happens. Follow the instructions below to input
*/
/*  all the physical constants and values for your robot. You should
*/
/*  already have configured your robot manually with the sidebar
configurer. */
```

```
/*---------------------------------------------------------------------------
---*/

Drive chassis(

//Specify your drive setup below. There are seven options:
//ZERO_TRACKER_NO_ODOM, ZERO_TRACKER_ODOM, TANK_ONE_ENCODER,
//TANK_ONE_ROTATION, TANK_TWO_ENCODER, TANK_TWO_ROTATION,
HOLONOMIC_TWO_ENCODER, and HOLONOMIC_TWO_ROTATION
//For example, if you are not using odometry, put ZERO_TRACKER_NO_ODOM
below:
ZERO_TRACKER_NO_ODOM,

//Add the names of your Drive motors into the motor groups below,
separated by commas, i.e. motor_group(Motor1,Motor2,Motor3).
//You will input whatever motor names you chose when you configured your
robot using the sidebar configurer, they don't have to be "Motor1" and
"Motor2".

//Left Motors:
motor_group(Drive_Front_Left,Drive_Back_Left),

//Right Motors:
motor_group(Drive_Back_Right,Drive_Front_Right),

//Specify the PORT NUMBER of your inertial sensor, in PORT format (i.e.
"PORT1", not simply "1"):
PORT17,

//Input your wheel diameter. (4" omnis are actually closer to 4.125"):
3.25,

//External ratio, must be in decimal, in the format of input teeth/output
teeth.
//If your motor has an 84-tooth gear and your wheel has a 60-tooth gear,
this value will be 1.4.
//If the motor drives the wheel directly, this value is 1:
0.6,

//Gyro scale, this is what your gyro reads when you spin the robot 360
degrees.
//For most cases 360 will do fine here, but this scale factor can be very
helpful when precision is necessary.
360,

/*---------------------------------------------------------------------------
---*/
/*                              PAUSE!
*/
/*
*/
/* The rest of the drive constructor is for robots using POSITION
TRACKING. */
```

Project Program (Continued)   Name Harris Perkins
Date 12/1/2023

Iteration 4 Building

```
/*  If you are not using position tracking, leave the rest of the values
as  */
/*  they are.
*/
/*-----------------------------------------------------------------------------
---*/

//If you are using ZERO_TRACKER_ODOM, you ONLY need to adjust the FORWARD
TRACKER CENTER DISTANCE.

//FOR HOLONOMIC DRIVES ONLY: Input your drive motors by position. This is
only necessary for holonomic drives, otherwise this section can be left
alone.
//LF:      //RF:
PORT1,     -PORT2,

//LB:      //RB:
PORT3,     -PORT4,

//If you are using position tracking, this is the Forward Tracker port
(the tracker which runs parallel to the direction of the chassis).
//If this is a rotation sensor, enter it in "PORT1" format, inputting the
port below.
//If this is an encoder, enter the port as an integer. Triport A will be a
"1", Triport B will be a "2", etc.
3,

//Input the Forward Tracker diameter (reverse it to make the direction
switch):
2.75,

//Input Forward Tracker center distance (a positive distance corresponds
to a tracker on the right side of the robot, negative is left.)
//For a zero tracker tank drive with odom, put the positive distance from
the center of the robot to the right side of the drive.
//This distance is in inches:
-2,

//Input the Sideways Tracker Port, following the same steps as the Forward
Tracker Port:
1,

//Sideways tracker diameter (reverse to make the direction switch):
-2.75,

//Sideways tracker center distance (positive distance is behind the center
of the robot, negative is in front):
5.5

);

int current_auton_selection = 0;
bool auto_started = false;
```

```
void pre_auton(void) {
  // Initializing Robot Configuration. DO NOT REMOVE!
  vexcodeInit();
  default_constants();


}

void autonomous(void) {
  auto_started = true;

  Puncher.spinFor(360,degrees);
  chassis.drive_distance(10);
  chassis.turn_to_angle(80);
  chassis.drive_distance(-15);
  Hook.set(true);
  Puncher.setVelocity(100,pct);
  Puncher.spinFor(46,turns);
  task::sleep(1000);
  Hook.set(false);
  chassis.turn_to_angle(-250);
  chassis.drive_distance(28);
  chassis.turn_to_angle(90);
  chassis.drive_distance(72);
  chassis.turn_to_angle(35);
  chassis.drive_distance(35);
  chassis.drive_distance(-36);
  chassis.drive_distance(36);
  chassis.drive_distance(-36);
  chassis.drive_distance(36);
  chassis.drive_distance(-36);



  /*
  chassis.drive_distance(24);
  chassis.turn_to_angle(-45);
  chassis.drive_distance(-36);
  chassis.right_swing_to_angle(-90);
  chassis.drive_distance(24);
  chassis.turn_to_angle(0);
  */
}

/*----------------------------------------------------------------------
---*/
/*
*/
/*                          User Control Task
*/
/*
*/
/*  This task is used to control your robot during the user control phase
of */
```

```
/*  a VEX Competition.
*/
/*
*/
/*  You must modify the code to add your own robot specific commands here.
*/
/*----------------------------------------------------------------------
---*/

void usercontrol(void) {
 // User control code here, inside the loop
 while (1) {



   Drive_Front_Left.spin(forward, Controller1.Axis3.position(percent) +
   Controller1.Axis1.position(percent), percent); // The code for driving
with the Front Left motor

   Drive_Back_Left.spin(forward, Controller1.Axis3.position(percent) +
   Controller1.Axis1.position(percent), percent); // Left Back Drive Code

    Drive_Front_Right.spin(forward, Controller1.Axis3.position(percent) -
   Controller1.Axis1.position(percent), percent); // Right Front Drive
Code

   Drive_Back_Right.spin(forward, Controller1.Axis3.position(percent) -
   Controller1.Axis1.position(percent), percent); // Right Back Drive Code



     if(Controller1.ButtonDown.pressing())
     {Lift.spin(forward,100,pct);
     Lift.setStopping(hold);
    }


     else if(Controller1.ButtonR2.pressing())
   { Intake.setVelocity(100,percent);
     Intake.spin(vex::directionType::fwd, 100, velocityUnits::pct);
     Intake.setStopping(hold);

     }

else if(Controller1.ButtonR1.pressing())
   { Intake.setVelocity(-100,percent);
     Intake.spin(vex::directionType::fwd, -100, velocityUnits::pct);
     Intake.setStopping(hold);
     }


    else if(Controller1.ButtonUp.pressing())
   {Lift.spin(vex::directionType::fwd, -100, velocityUnits::pct);
    Lift.setStopping(hold);
```

```
    }
else if(Controller1.ButtonX.pressing())
    {WingLeft.set(true);
    WingRight.set(true);
    }
else if(Controller1.ButtonA.pressing())
    {WingLeft.set(false);
    WingRight.set(false);
    }
else if(Controller1.ButtonLeft.pressing())
    {Hook.set(false);
    }
else if(Controller1.ButtonRight.pressing())
    {Hook.set(true);
    }
else if(Controller1.ButtonY.pressing())
    {Lock.set(true);
    }
else if(Controller1.ButtonL1.pressing())
    {Puncher.spin(vex::directionType::fwd, 100, velocityUnits::pct);
     Puncher.setStopping(hold);
    }

        else
    {
      Intake.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);
      Lift.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);
      Puncher.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);

      }

  }

digital_out dig1 = digital_out( Brain.ThreeWirePort.A );

    if( Controller1.ButtonB.pressing() ) {
      dig1.set( true );
    }
    else {
      dig1.set( false );                          //Code for the Pneumatics
    }

    this_thread::sleep_for(10);

    // This is the main execution loop for the user control program.
    // Each time through the loop your program should update motor + servo
    // values based on feedback from the joysticks.
```


Iteration4

Building

```
    //
    // Insert user code here. This is where you use the joystick values to
    // update your motors, etc.
    //

    //Replace this line with chassis.control_tank(); for tank drive
    //or chassis.control_holonomic(); for holo drive.
    chassis.control_arcade();

    wait(20, msec); // Sleep the task for a short amount of time to
                    // prevent wasted resources.
  }

//
// Main will set up the competition functions and callbacks.
//
int main() {
  // Set up callbacks for autonomous and driver control periods.
  Competition.autonomous(autonomous);
  Competition.drivercontrol(usercontrol);

  // Run the pre-autonomous function.
  pre_auton();

  // Prevent main from exiting with an infinite loop.
  while (true) {
    wait(100, msec);
  }
}
```

HP

Creating Lexan Dimension Charts:

Goals for today → Work as a team to trace and organize each of our Lexan pieces to make sure we are in dimensions

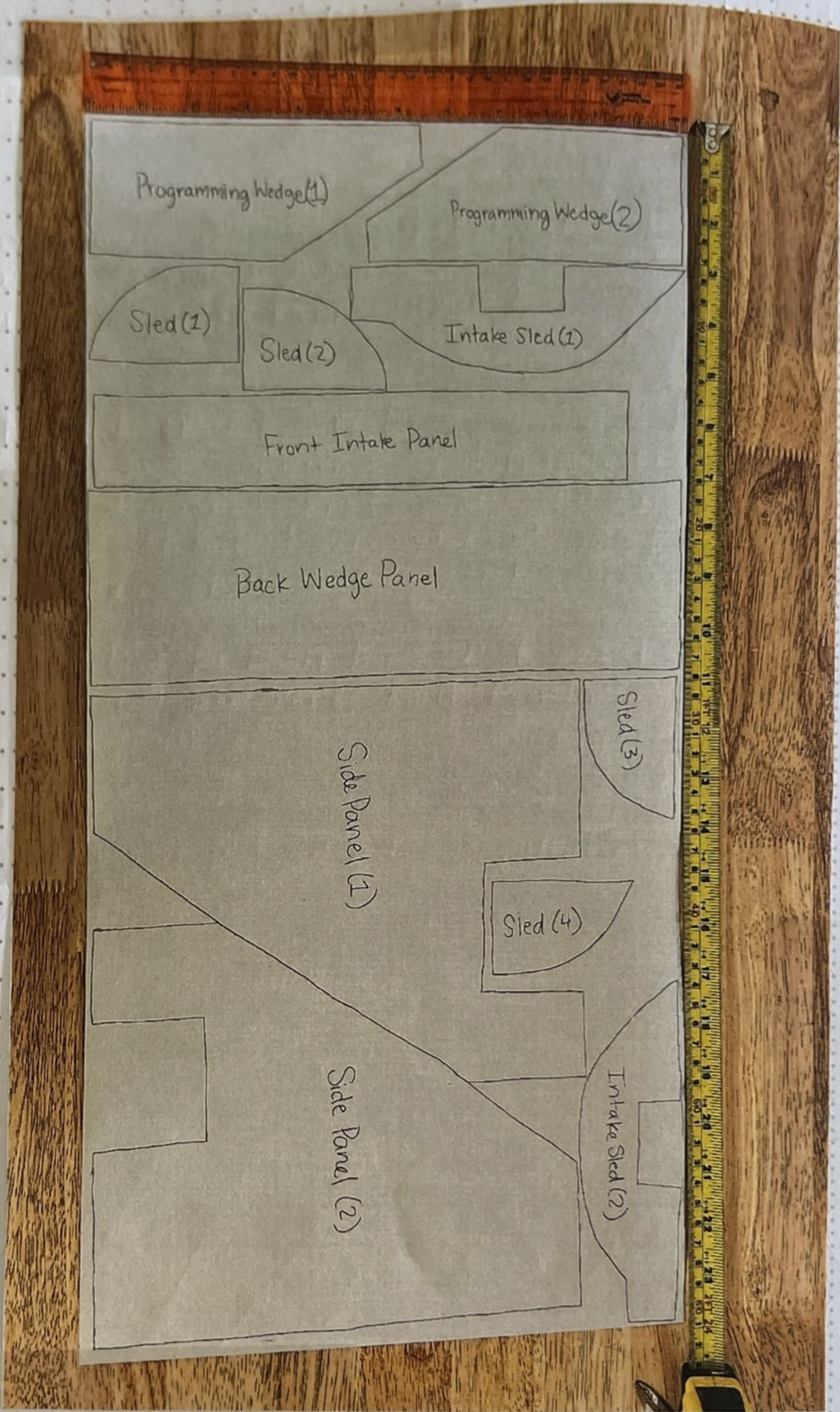
Robots are allowed to cut their pieces from a 12 inch by 24 inch sheet. Today we worked to better organize our measurements and double check that we fit inside the requirements.

On page 94 is our original chart with labeled pieces and a ruler and tape measurer for scaling purposes in the picture. This chart is accurate to the robot up until page 80 when we tested the programming wedges and recognized they needed to be revised.

On page 95 is our new and most up to date Lexan dimension chart that shows the removal of the two (2) programming wedges for the replacement of one (1) improved wedge.

We were successfully insided the requirements.

VP



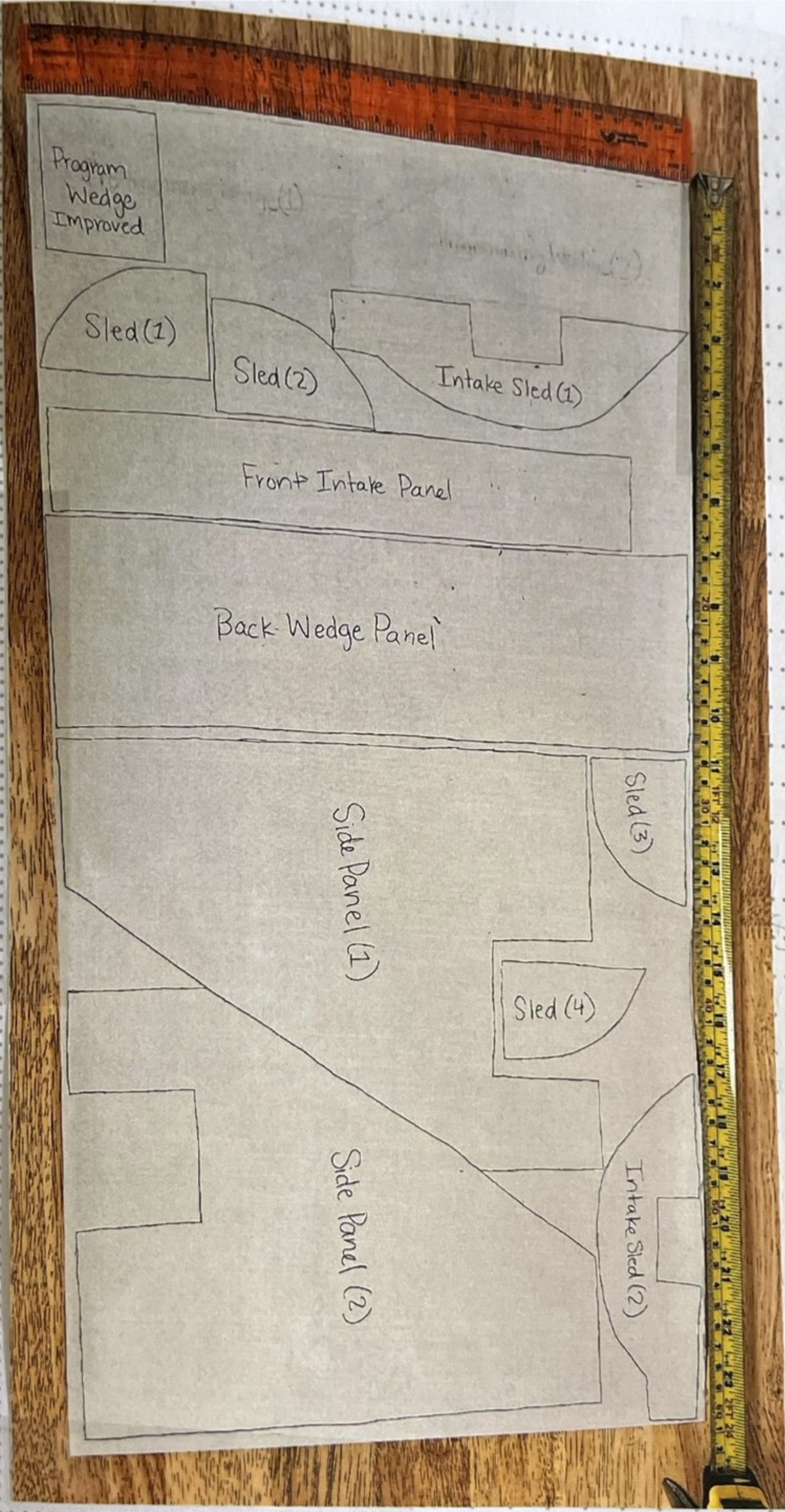
Project Lexan Dimensions Chart

Name Vanessa Perkins

Date 12/1/2023

VEX ROBOTICS

PROPRIETARY INFORMATION



Project Lexan Dimensions Chart (new) Name Vanessa Perkins

Date 12/1/2023

Competition Analysis:

Goals for today → Review and Discuss the results of the SEMS Winter Wonderland Competition

Above on the left is an image of the majority of our team with our alliance 57711X, Rocky. Above on the right is our robot with Rocky's robot on the left hand side.

With our alliance Rocky, we won the tournament. Our team also won the Design Award.

This competition helped us learn several things as listed below:

- Strategies are changing to where we need to be much faster at scoring in goals or else the opposing alliance will push them back over the barrier.
- Skills is very important in taking our team from winning the Design Award to winning the Excellence Award.

Project Competition Analysis Name Vanessa Perkins
Date 12/4/2023

Iteration 4

Building

New Goals:

Goals for today → plan out what we want to achieve before/at the next competition

Goal Planner

Robot Goals

Rebuild the robot to fit the new strategies of the game

Action plans

— Review performance —

— Brainstorm new ideas —

— Build efficiently —

– Keep a timeline and schedule –

— Test and Revise as needed —

Performance Goals

Rank Top Four (4)
Make it to the Finals
Win Design Award
FWP 70% of time
Highest OPR
Win Auton 100% of time

Action plans

— Scout efficiently —

— Choose a good alliance —

— Practice Interviews —

— Make program consistent —

— Practice Offense —

— Make program consistent —

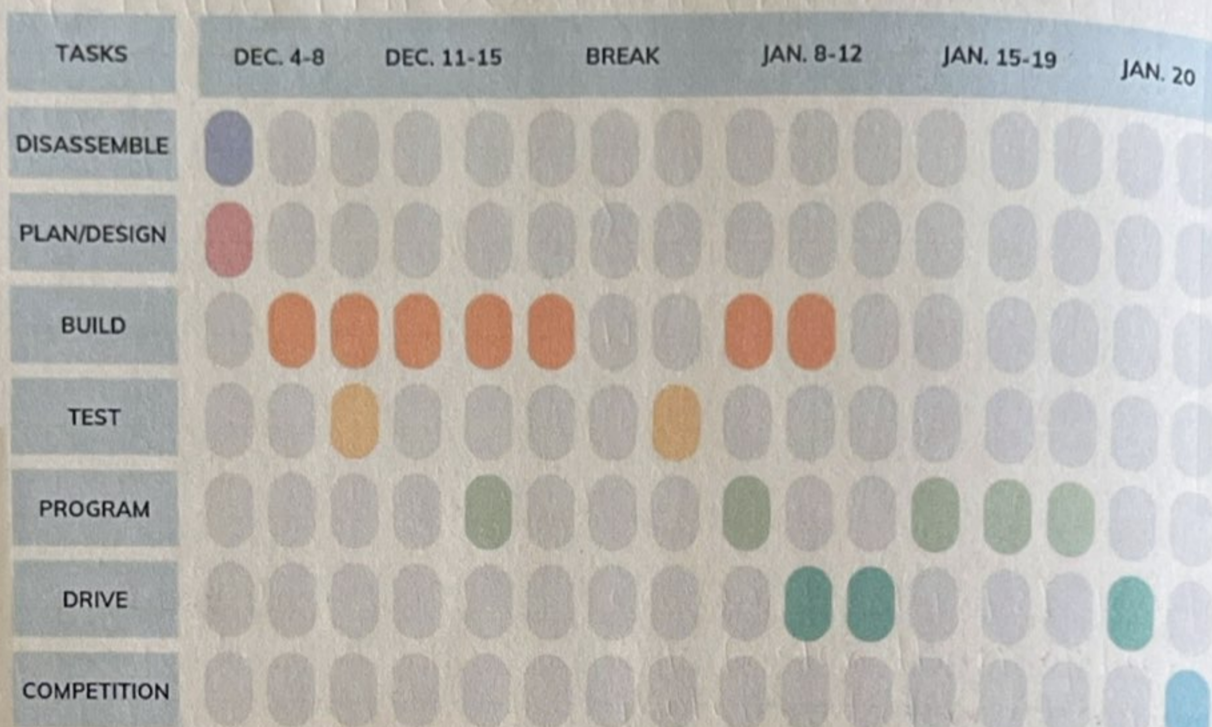
VP

Project New Goals
Date 12/5/2023

Name Vanessa Perkins

GANTT CHART

MIDDLE SEASON TIMELINE



Disassemble- Take apart the entire robot to reuse pieces for a new design

Plan/Design- Review the rules, research ideas, and decide on new subsystems to be built

Build- Begin assembling the pieces

Test- Intermittently test what we have built and change as necessary

Program- Set up controls and autonomies

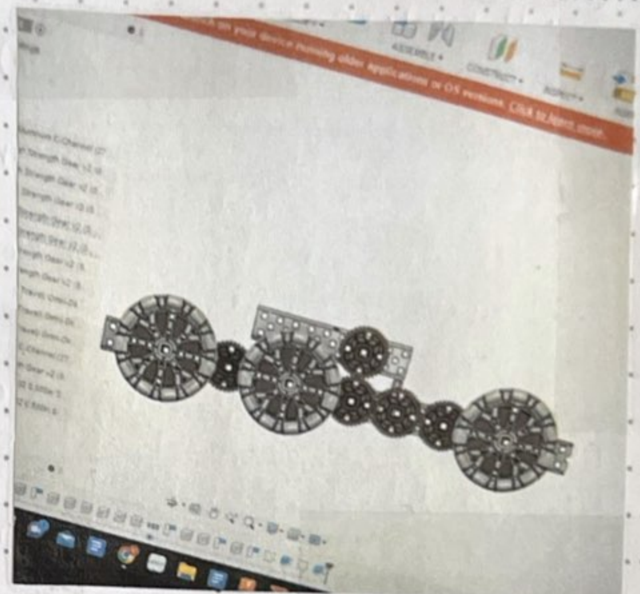
Drive- Practice learning controls and strategies as well as run full matches, scrimmages with teams, and skills

Competition- Final performance of the robot at Sullivan East Middle School

Brainstorm Base Ideas:

Goals for today ⟶ Come up with at least three base design ideas to choose from.

Base Idea One: Slip n' Slide Drive Base



Specifics:
- 6 motors
- 6 omni wheels (3.25 inch)
- 3 gear gap between last two (2) wheels
- 360 rpm
- 36:60 gear ratio

To the ~~left~~ right is a CAD rendering of what this base would look like. The gap between the back two (2) wheels is to improve transition over the barrier with the help of sleds. There will be a middle sled/secondary sled to prevent the base from getting stuck on the barrier.

Base Idea Two: Slip n' Slide Drive Base with 2 Gaps

Specifics:
- Same specifics as Idea One, but there is a gear gap in the front and back using 36 tooth gears.

This base design is similar to the first with some minor changes. This design, while continuing to improve the transition over the barrier, might make the base too long and have too much grey area of no control.

Iteration 4

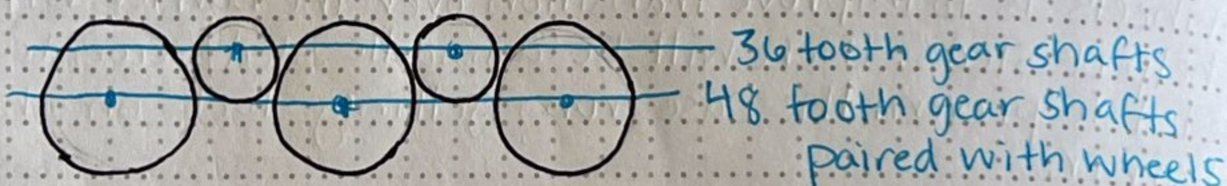
Building

(Continued)

Base Idea Three: Basic Drive (Summer Robot)

Specifics:
- 4 motors
- 6 omni wheels (3.25 inch)
- 450 rpm
- 600 cartridge geared down
- 36:48 gear ratio
- 48's on wheels, off set 36's
- Sleds on front and back

This base design is based on our first robot located on page 10 book 1. This base is faster than option one and two, but has less motors. Below is a rough sketch of this base design.



VP

Select Approach:

Goals for today → Review and Discuss our brainstorm options and Select a base most appropriate for the game

Below is our decision matrix to help us choose a base design as well as some explanations.

Base Ideas	Speed	Torque	Barrier Transition	Versitility	Total
Slip n'Slide	3	4	5	4	16
Slip n'Slide 2	3	3	4	5	15
Basic Drive	5	2	2	5	14

* Speed: how fast it drives
Torque: the Strength, ability to play defense, and withstand defense
Barrier Transition: how smoothly and efficiently the base Scales the barrier
Versitility: how efficiently can the base switch strategies and hold all the necessary subsystems

Selected Design: Slip n'Slide

Explanation: We chose the Slip n'Slide base design through the help of a decision matrix because of its overall performance and scoring. This design has a decent speed, good torque, excellent barrier transition, and good versitility to suit the game. This design will allow us to drive efficiently and have room for all future subsystems we will be adding.

* Break down of each criteria we rank on 0-5 (five good).

VP

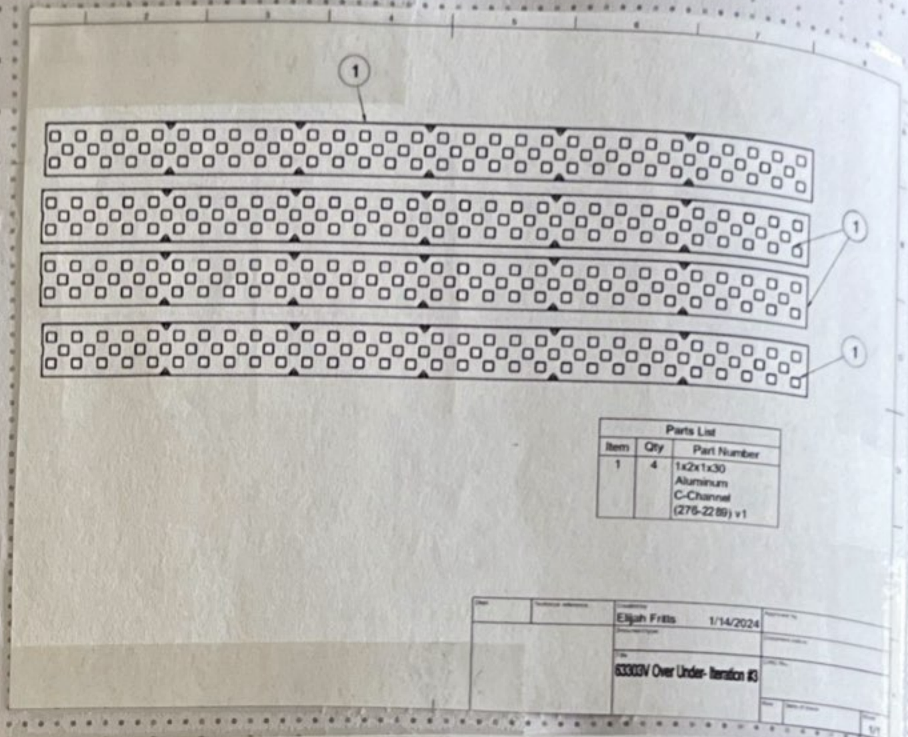
Iteration 4

Building

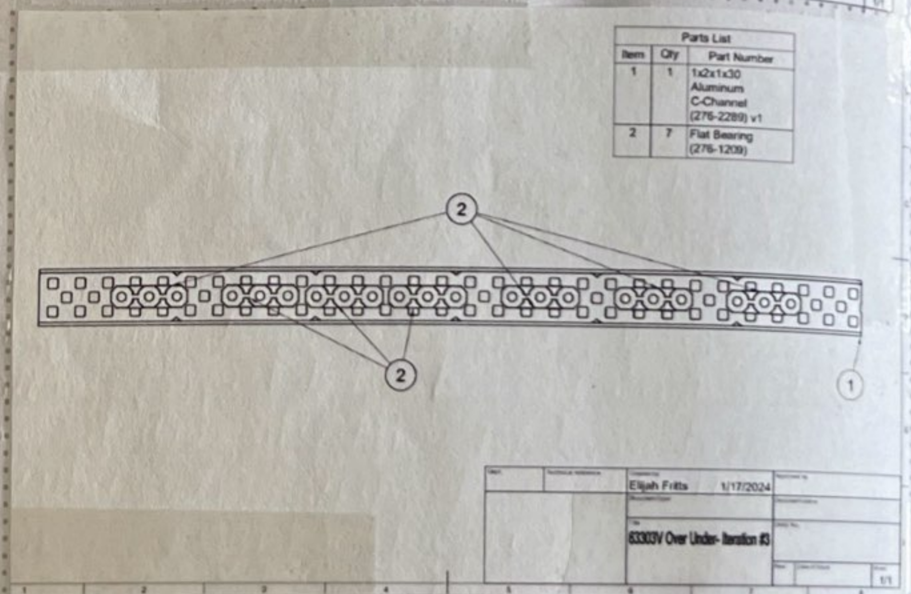
Begin Building Base:

Goals for today → Get the entire base structure built
→ Begin CAD of the robot

① Gather four (4) thirty (30) hole long aluminum c-channels to begin the base.



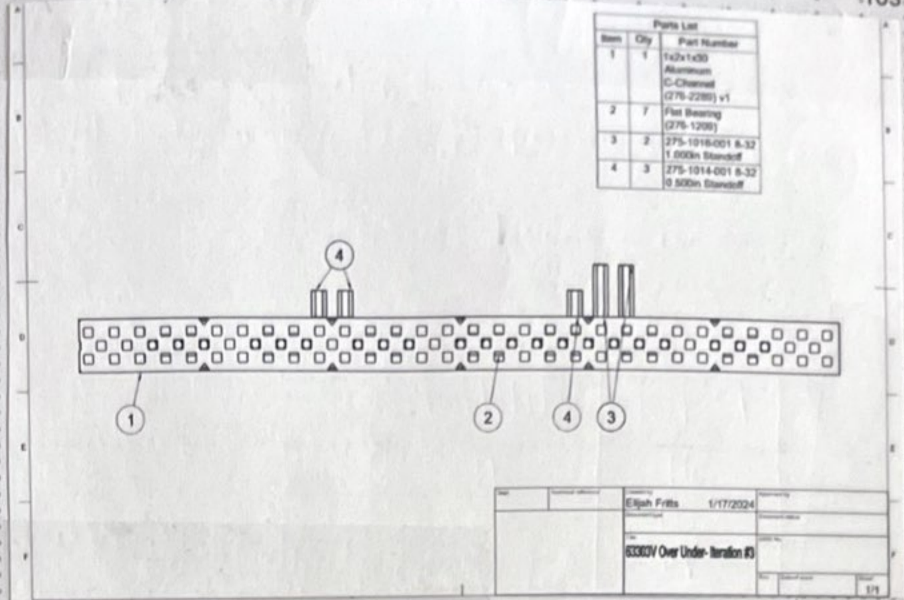
② Ziptie seven (7) bearing flats to the depicted locations on two (2) c-channels. Total of fourteen (14) zipties per c-channel.



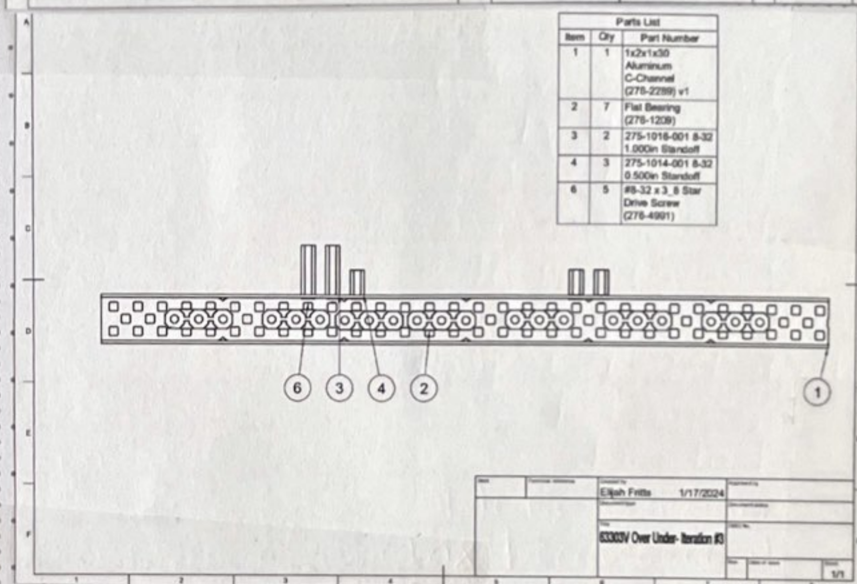
VP

(Continued)

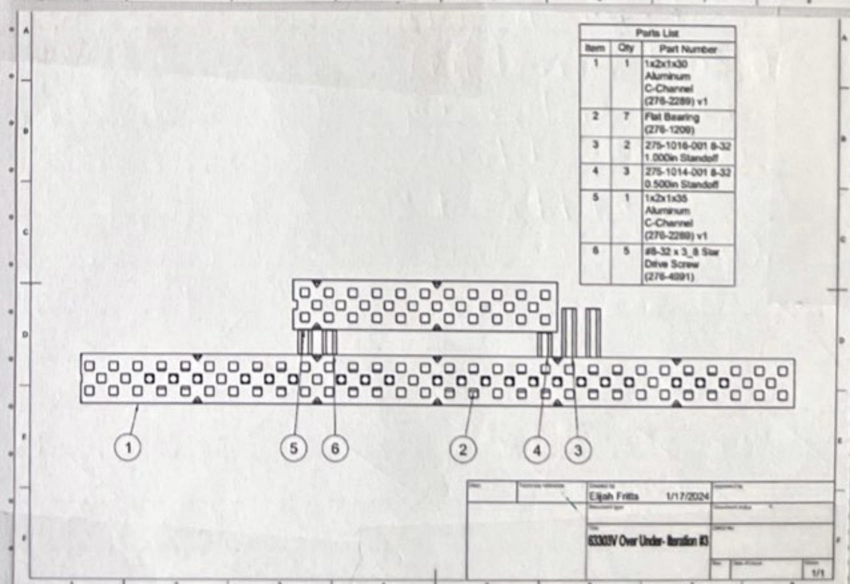
③ Attach the following c-channels to the depicted locations on all four (4) c-channels.



④ This step shows where the screws go into the standoffs.



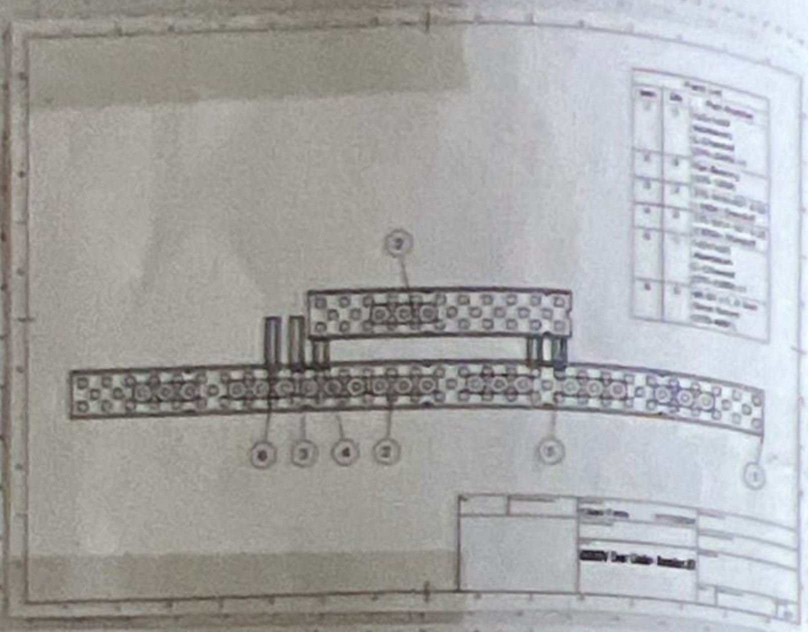
⑤ Gather or cut a eleven (11) hole aluminum c-channel and place in depicted location. Repeat for a total of four (4) times.



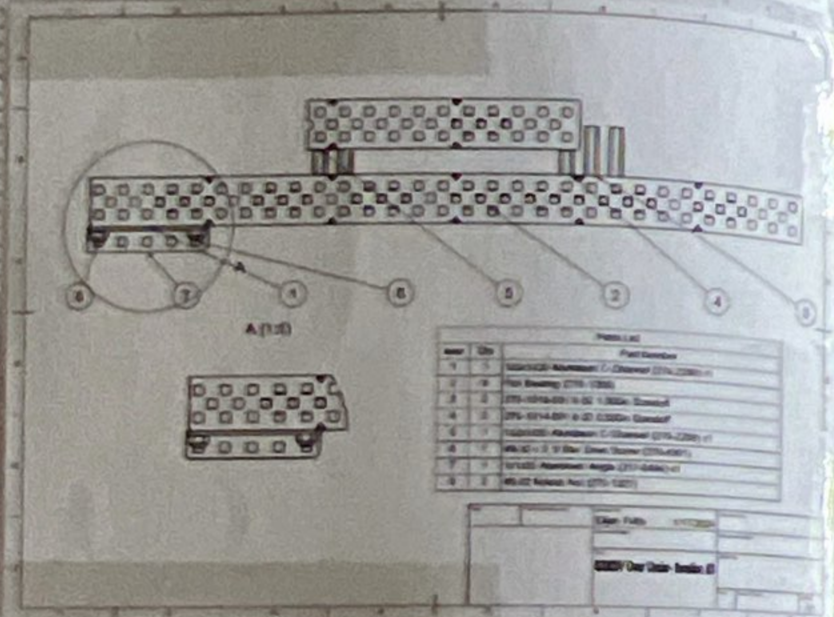
Building

(Continued)

(6) Add a bearing flat to the 11 long C-channel in the depicted location to two (2) C-channels.

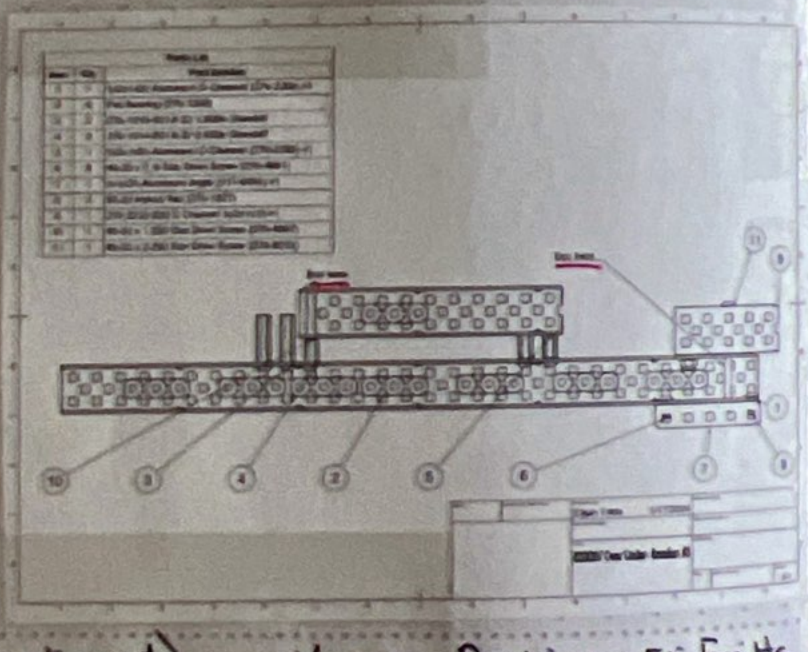


(7) Add a five (5) long 1 by 1 L-channel to the bottom of the base C-channel using two (2) screws and nylock nuts. Repeat for a second C-channel.



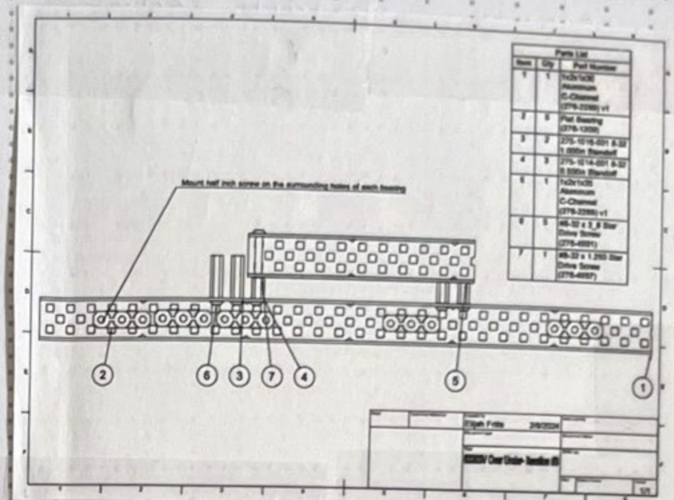
(8) Add a five (5) long aluminum C-channel above the L-channels from step 7 using the listed screws.

Box here means add a 0.5 and 0.375 inch spacer.

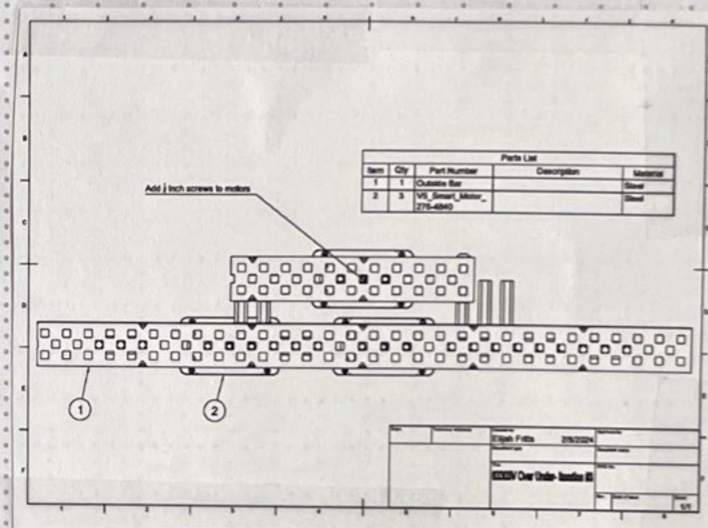


(Continued)

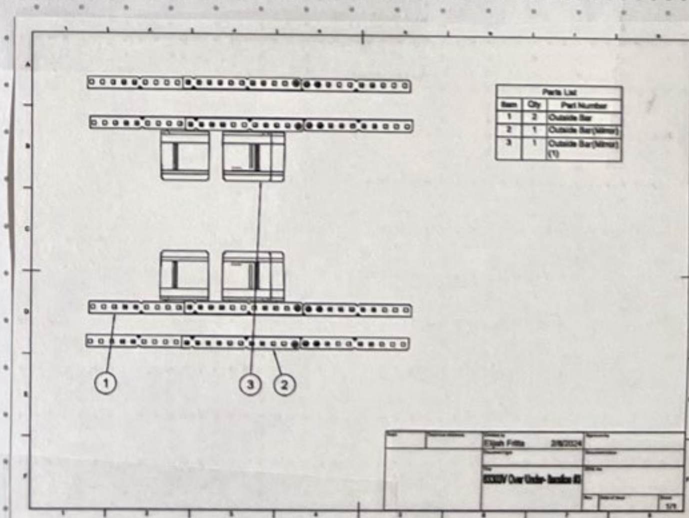
9. Place five(5) bearing flats on the two (2) c-channels with bearing flats already ziptied on. Use two(2) half inch screws per bearing flat.



10. Attach three(3) motors each to the two(2) c-channels from step 9 in between each of the bearing flats using half inch screws.



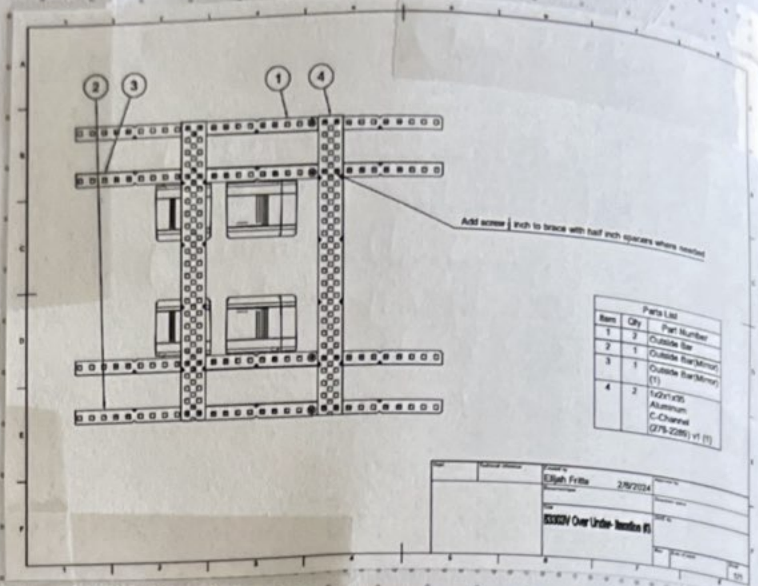
11. Lay the four(4) c-channels in the depicted positions to the right to prepare for wheels and bracing.

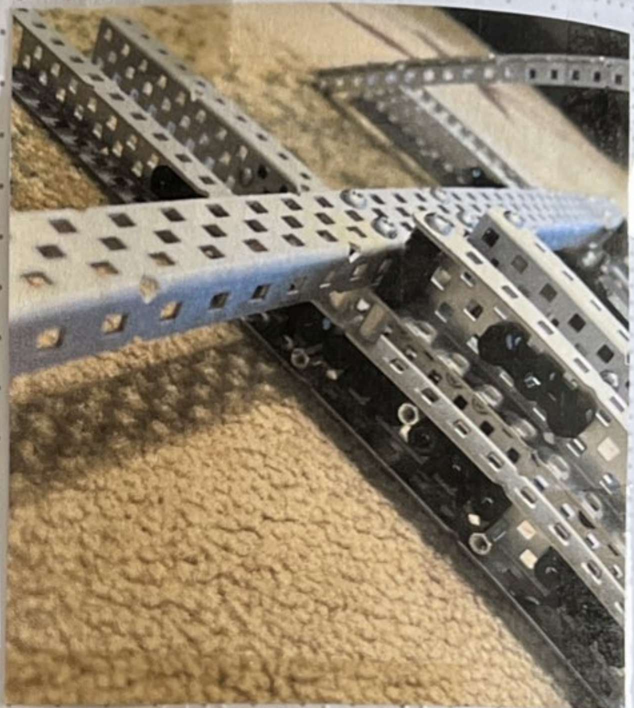


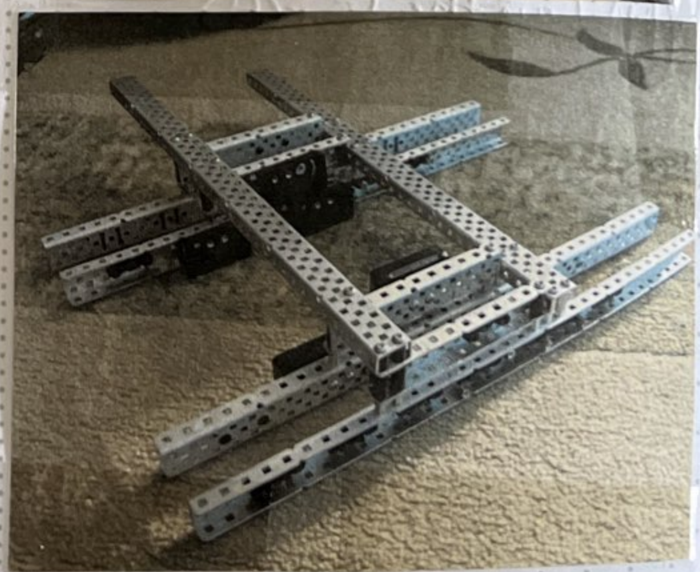
Building

(Continued)

12 Gather or cut two (2) twenty-four (24) long c-channels and screw into the depicted locations.



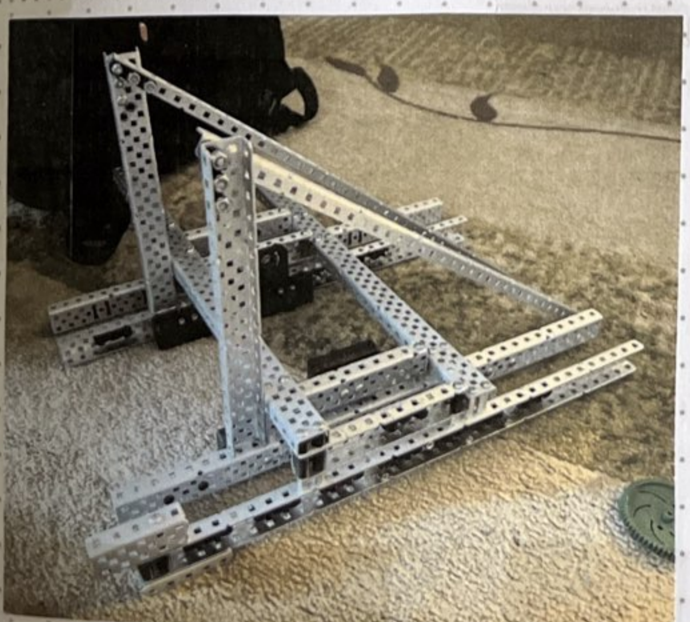




Project Begin Building Base (Continued) Name Vanessa Perkins + Eli Fritts
Date 12/7/2023

(Continued)

Below and on the previous page are pictures of the base up to this current point of the build from several different angles.



Project Begin Building Base (continued) Name Vanessa Perkins

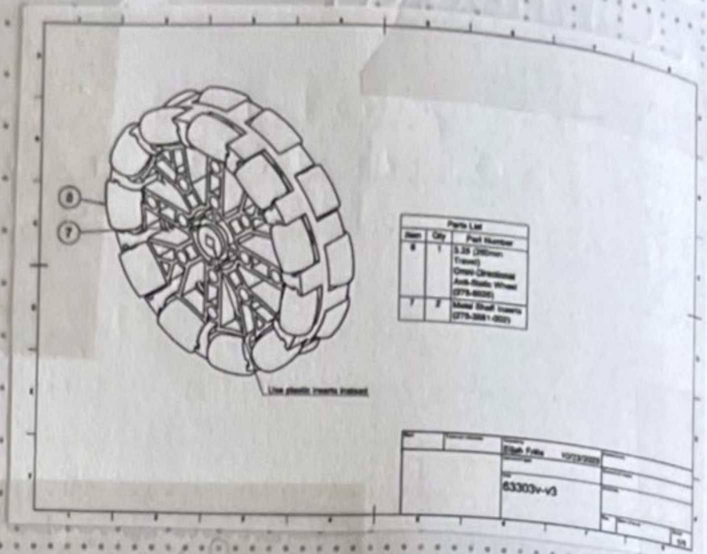
Date 12/7/2023

Building

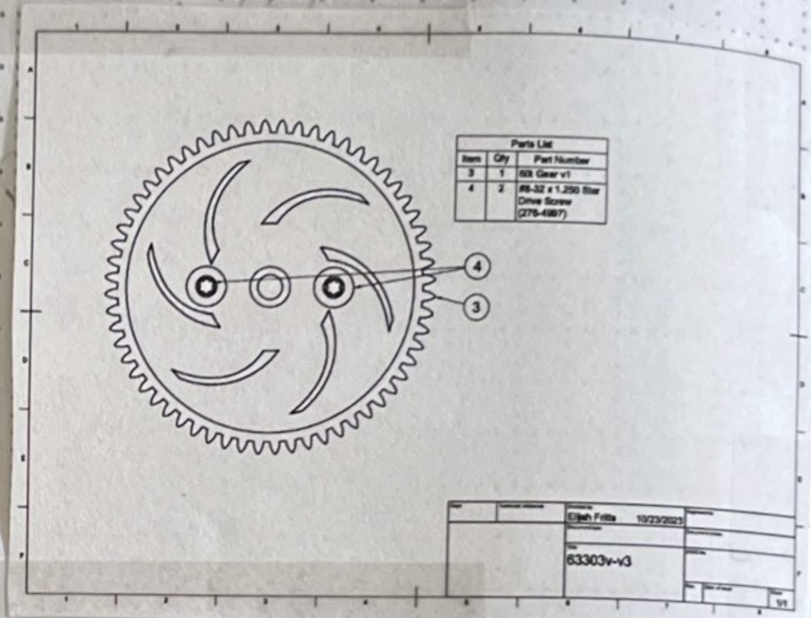
Wheels Spacing:

Goals for today → get the wheels ready to be added to the base

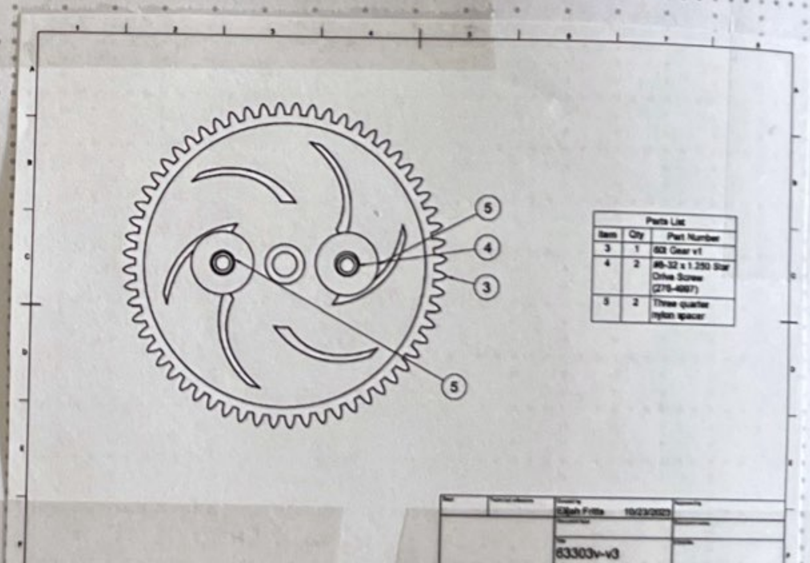
1. Gather four(4) 3.25 inch diameter omni wheels and two(2) traction wheels. Add metal shaft inserts to all six(6) wheels.



2. Gather three(3) 60 tooth gears per side of the base. Place two(2) 1.250 inch screws in each gear for a total of twelve(12) screws.



3. Add a 0.375 inch spacer to each of the screws in the gears.

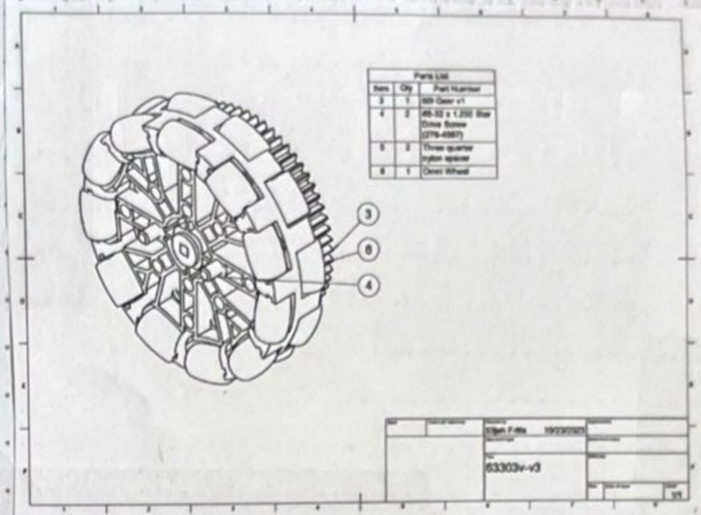


VP

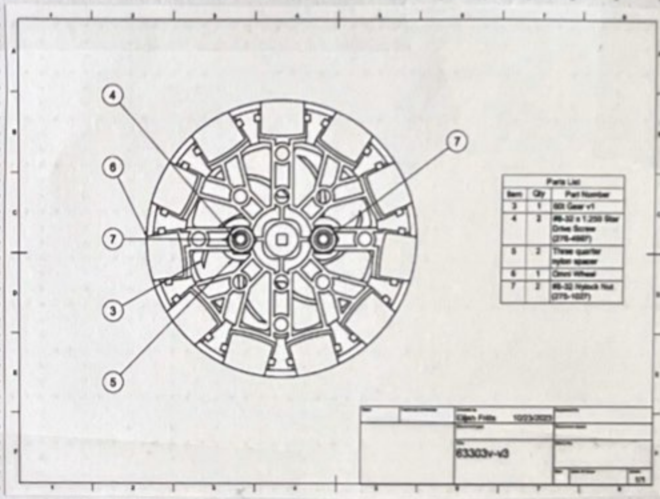
Project Wheel Spacing
Name Eli Fritts & Vanessa Perkins
Date 12/8/2023

(Continued)

4 Add a wheel to each of the gears through the screws.



5 Add a nylock nut to each of the screws then tighten.

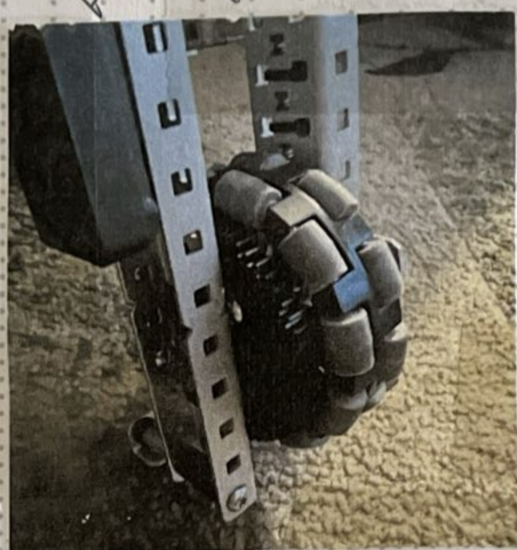


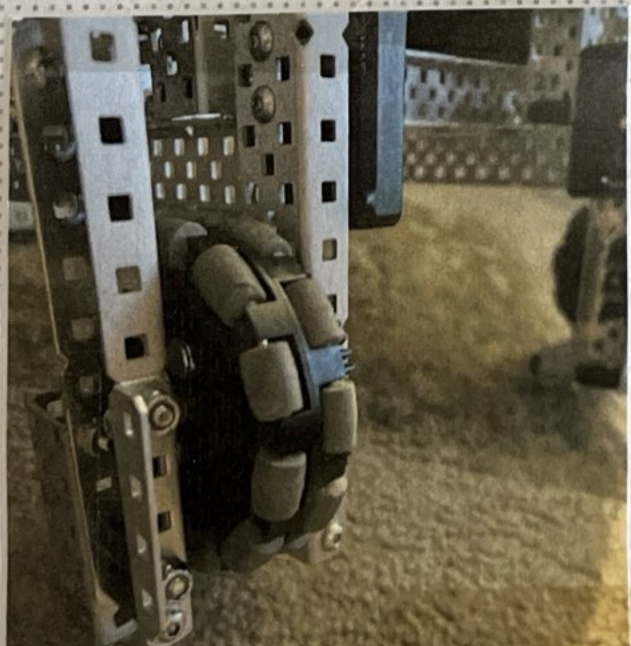
Below is a picture of a wheel and a gear before they got screwed together.

VP

(Continued)

Below are images of the wheels being attached to the base. A two (2) inch screw with a teflon washer and nylock nut are used to attach the wheel in the bearing flats.

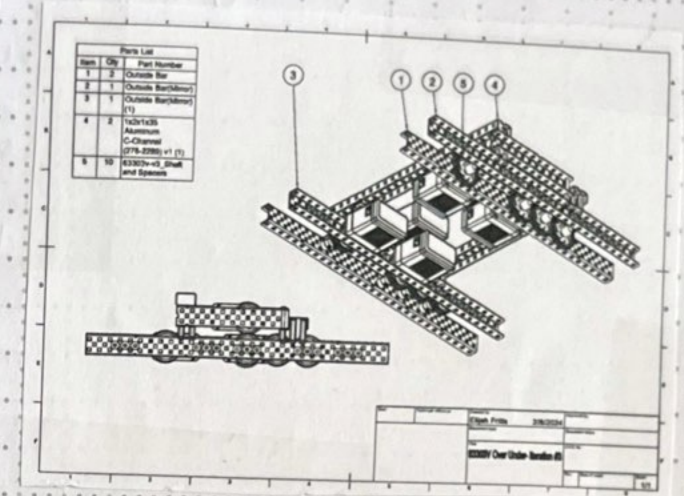




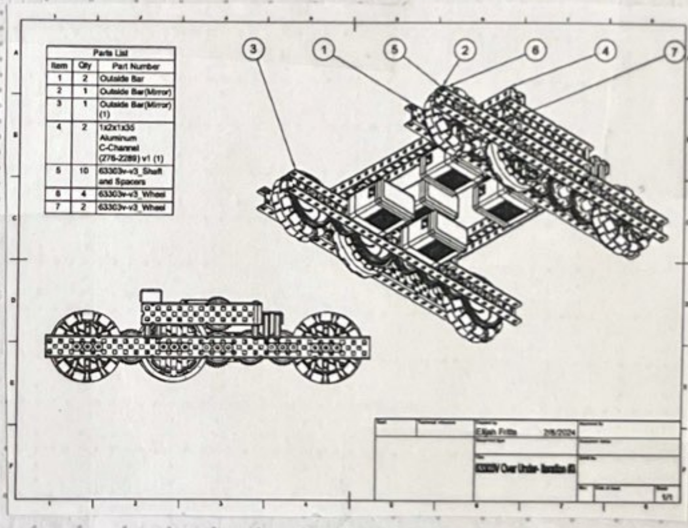
Finish Base:

Goals for today → finish building the base

1. Add five (5) 36 tooth gears to the depicted locations to the right on the ~~spacing~~ base into the motors using shafts and spacers.



2. To the right shows the wheels attached as described and seen on page 110. The traction wheels go in the middle.



VP

Building

Project Finish Base

Date 12/9/2023

Name Eli Fritts + Vanessa Perkins

Begin Building 4 Bar Lift:

Goals for today → Get the 4 bar build Started
→ Explain our decision process

Below is a decision matrix of what aspects mattered the most for our lift mechanism as well as an explaination to accompany the winning design.

Lift Ideas	Height	Space	Simplicity	Versitility	Total
Inverted 10 Bar	4	0	0	4	8
4 Bar	5	3	3	3	14
No Lift	0	5	4	0	9

* Height: how tall is it to block others and shoot over others
Space: how compact is the lift on the base and a good use of space
Simplicity: how complex and time consuming to build it is
Versitility: how easily can the design switch between Strategies during matches

Selected Design: 4 Bar
Explanation: The 4 Bar design provides the best compromise between height, space efficiency, simplicity, and versitility. The 4 Bar takes up less space and will be a little taller and simpler while still providing us with the opportunity to switch strategies. The only down side is that the 4 bar currently has no plans for a hanging mechanism.

VP

* Break down of each criteria aspect we looked at and ranked on a scale of 0-5 (five means good).

(Continued)

Below are the steps we took to begin building the 4 Bar Lift mechanism.

1. Gather/cut two (2) fifteen (15) hole long aluminum c-channels.
2. Attach both c-channels to the back base support perpendicularly using two (2) 0.500 inch screws and nylock nuts five (5) holes inward on each side of the base.
3. Gather/cut two (2) twenty-five (25) hole long aluminum c-channels. Gather/cut two (2) 1 by 1 aluminum L-channels of the same length.
4. Attach a bearing flat to each end of all four (4) pieces of metal from step 3.
5. Attach the ends of the L-channels to the top of the perpendicular supports from step 2.

VP

To the left is an image of the new robot currently with the old robot on the right.

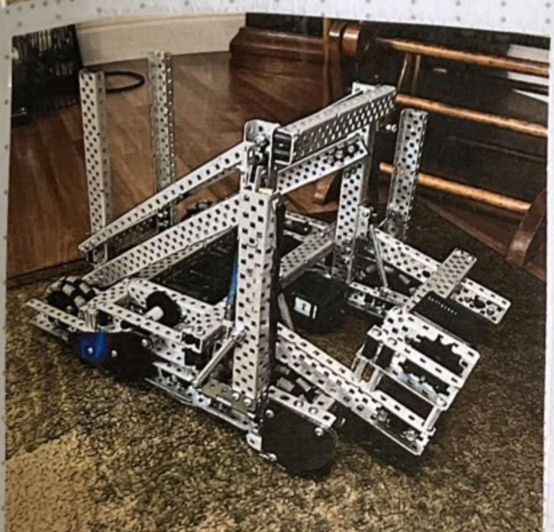
Building

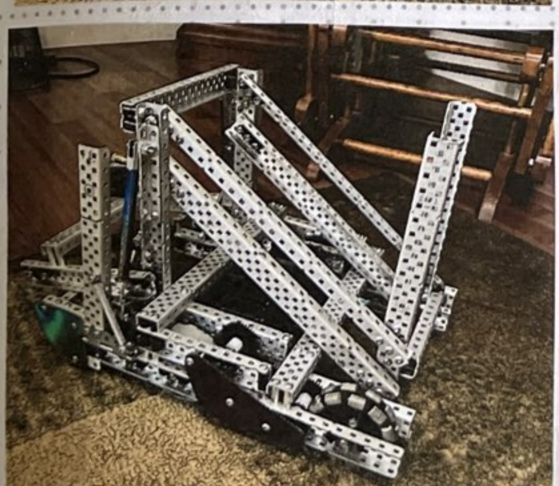
(Continued)

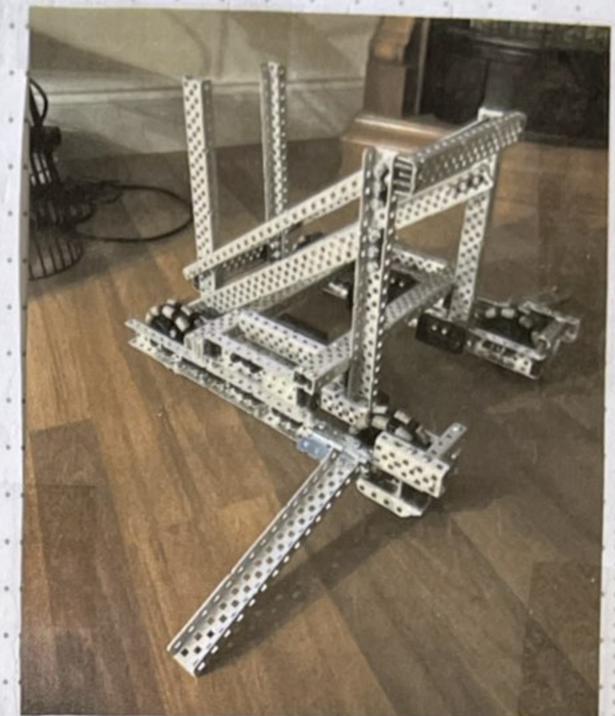
6. Attach the c-channels the same way to the same perpendicular c-channels three (3) holes lower.
7. Gather/cut two (2) twenty (20) hole long aluminum c-channels.
   Add a bearing flat to one end of each.
   Screw into the lower ends of the c-channels and 1by1 L-channels so they stay parallel.
8. Gather/cut a ten (10) hole long aluminum c-channel.
   Gather/cut a sixteen (16) long c-channel.
9. Attach the 10 long c-channel to the bottom of the lift using four (4) 0.5 inch spacers and screws.
10. Repeat step 9 at the top with the 16 long c-channel.

VP

Finished 4 Bar Lift Images:







Building

Project Finished 4 Bar Lift Images Name Vanessa Perkins
Date 12/11/2023

Brainstorm Launch Mechanism Ideas:

Goals for today → Research and come up with at least 3 launch mechanism design options to select from
→ Respect and listen to every team member's ideas

Design Option One: Puncher

Specifics:
- 1 motor
- slip gear
- 44 triballs in 28 seconds

Pictures are located on pages 19 and 20 with more specifics. A pro is that we won't have to rebuild the puncher since we already have one to attach.

Design Option Two: Catapult

The specifics for this design are located on page 29 book 1.

Design Option Three: Kicker

Specifics:
- 1 motor
- Slip gear
- More direct forward force than catapult

The Kicker design has a hammer on a slip gear that hits the triballs across the barrier. The Kicker is the most accurate of these three (3) designs if built properly.

Select Approach:

Goals for today → Choose a launch mechanism design that best meets the requirements of the game

We created a decision matrix of what aspects mattered the most on our launch mechanism. Below is that matrix with the winning mechanism and more details

Ideas	Speed	Accuracy	Force	Time to Build	Total
Puncher	5	4	5	5	19
Catapult	3	3	3	2	11
Kicker	4	5	5	2	16

Building

* Speed: How fast can the mechanism launch matchloads
Accuracy: How accurately can the triballs be launched across the field to create a good grouping for scoring in goal
Force: How powerful is the launch mechanism to make it over the barrier
Time to Build: How quickly can we build the mechanism, attach it to the robot, and get it running

Selected Design: Puncher
Explanation: We plan to use the puncher as our launch mechanism on this robot. After research, prototypes and looking at our schedule, we realized the puncher was the best decision for our team. Having a working puncher already availiable to us will allow us have more time for programming and driving practice.
Our team is already familiar with the function of a puncher which is another plus.

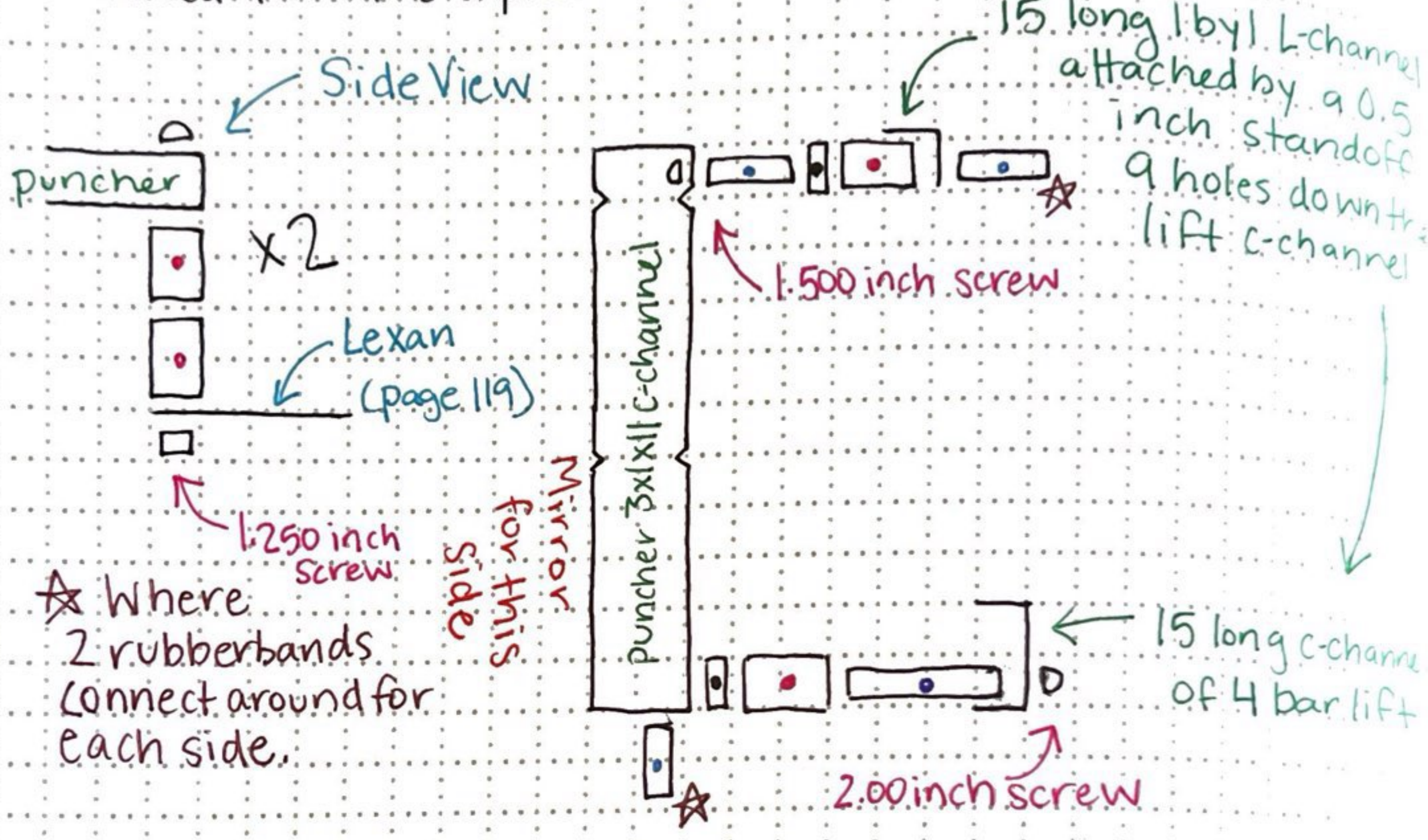
* Break down of each criteria aspect we looked at and ranked on a scale of 0-5 (five means good).

Project Select Approach
Date 12/12/2023
Name Vanessa Perkins

Attaching Puncher to 4 Bar:

Goals for today → Get the puncher functioning on the new robot

First we removed the puncher from the old robot. Below is a diagram of how we attached it to the new robot as viewed from the top.



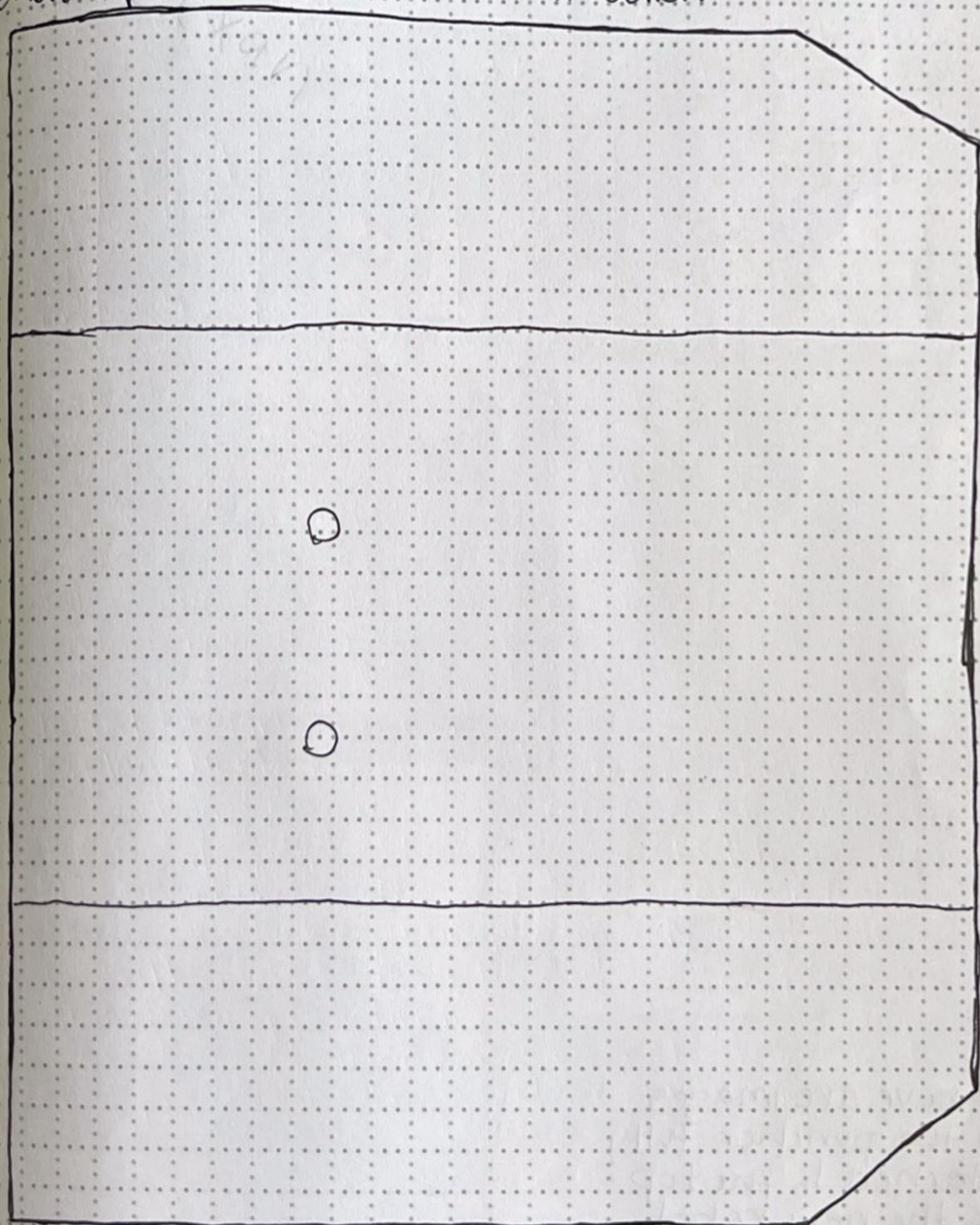
Parts List:

- Old Puncher (Built on page 19-20 book 2)
- 8 - 0.500 inch spacers
- 4 - 0.125 inch spacers
- 6 - 0.500 inch standoffs
- 2 - 1.000 inch standoffs
- ~~4 couplers~~ nylock nuts
- 2 - 15 long 1by1 L-channels
- 4 - 0.500 inch screws
- 2 - 1.500 inch screws
- 2 - 1.250 inch screws + 2 - 2.000 inch screws

To the left is a parts list of all the necessary mounting pieces. The colors coordinate with the pieces represented in the diagram above for both sides and views. On the next page is a template for the Lexan sheet as labeled above where the triball sits in front of the puncher.

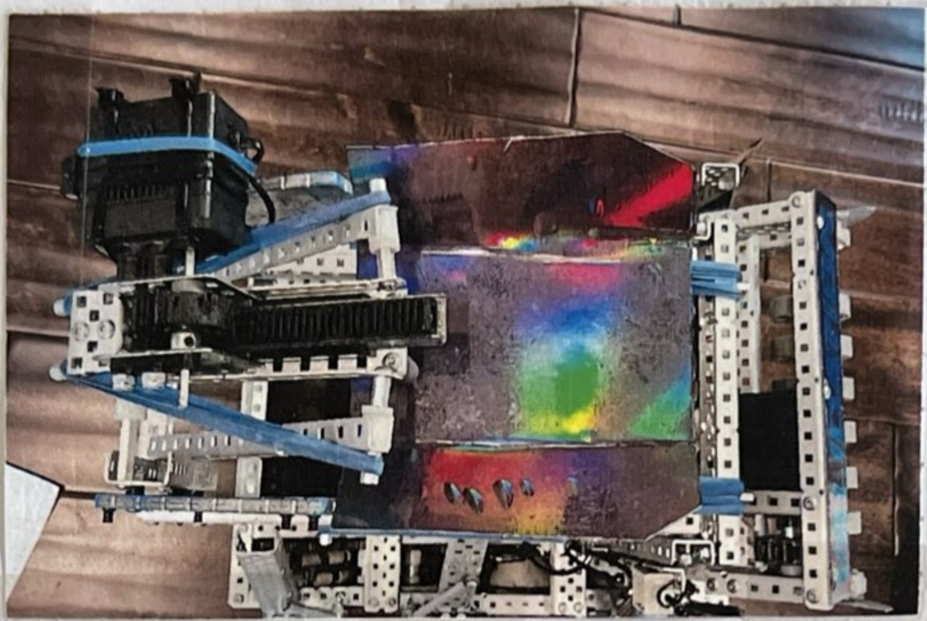
Lexan Sheet Template:

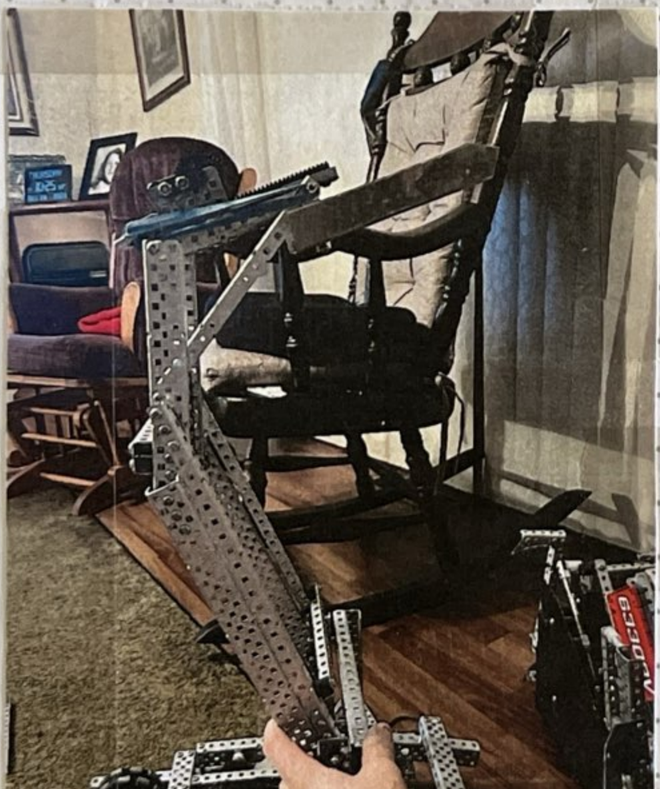
Below is the Lexan template for the triball to sit in front of the puncher with screw holes and lines were the Lexan bends upward to hold the triball better.



Building

Finished Puncher Pictures:





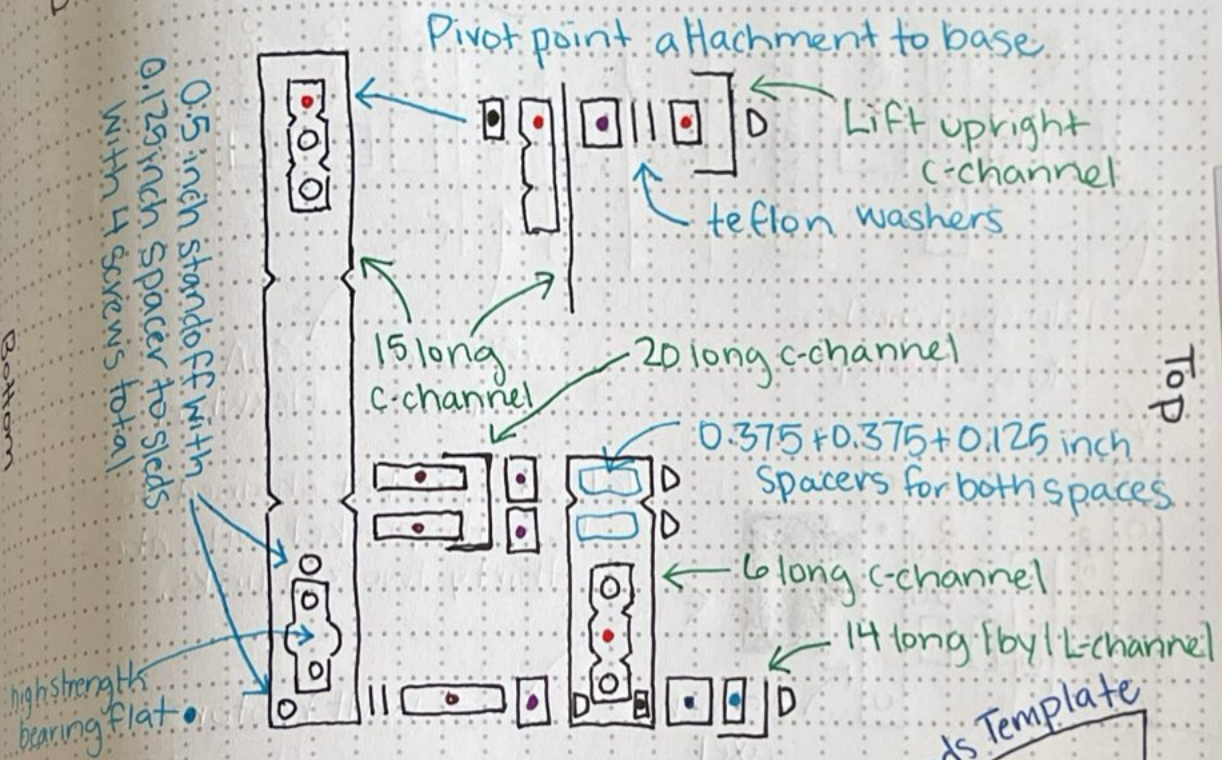
Above are images of the puncher fully attached to the top of the new robot.

Project Finished Puncher Pictures Name Vanessa Perkins
Date 12/13/2023

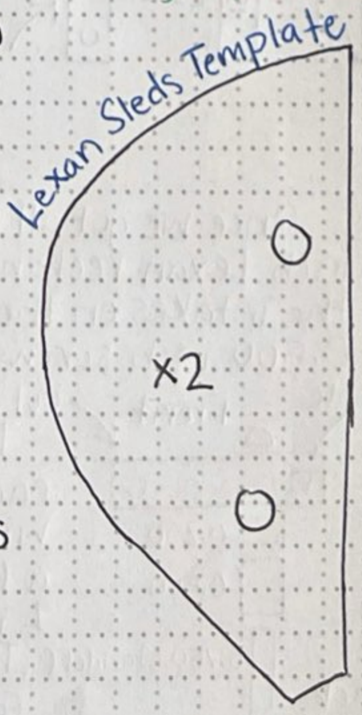
Begin Building Intakes:

Goals for today ⟶ Get the intake structure built and attached to robot

Below is a diagram of the intake structure, side view.



☆Mirror and Repeat☆



Parts List:
- 2-15 long aluminum c-channels
- 2-6 long aluminum c-channels
- 1-20 long aluminum c-channels
- 1-14 long aluminum 1 by 1 L-channel
- 6 bearing flats •
- 22 nylock nuts •
- 4-0.250 inch spacers •
- 6-1.000 inch standoffs •
- 4-0.500 inch standoffs
- 8 Teflon washers
- 2 high strength bearing flats •
- 10-0.375 inch spacers •
- 10-0.125 inch spacers •
- 2 Lexan sleds

Project Begin Building Intakes

Name Vanessa Perkins

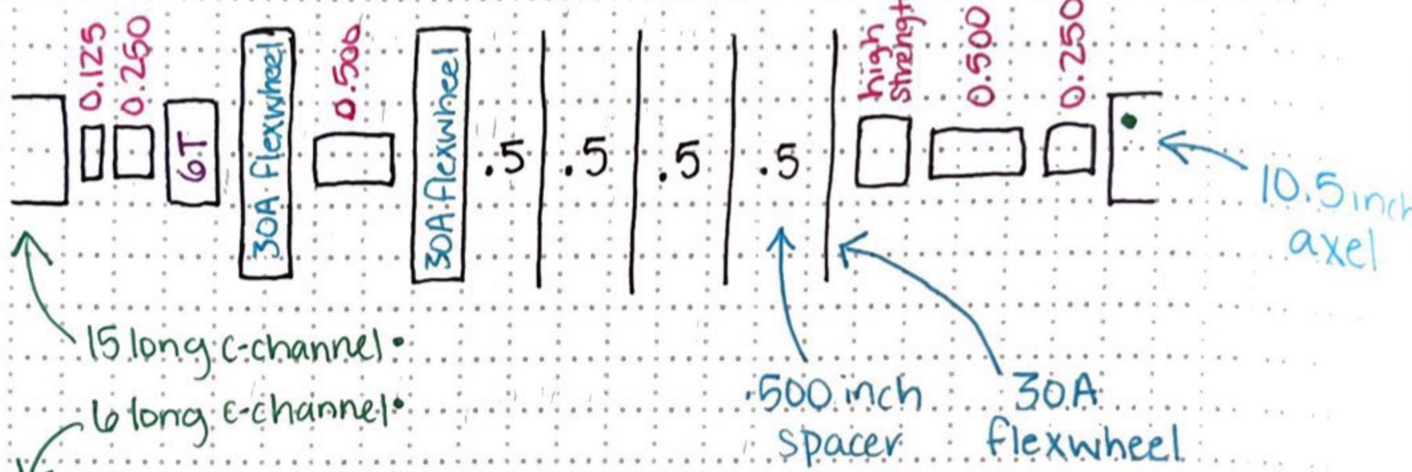
Date 12/14/2023

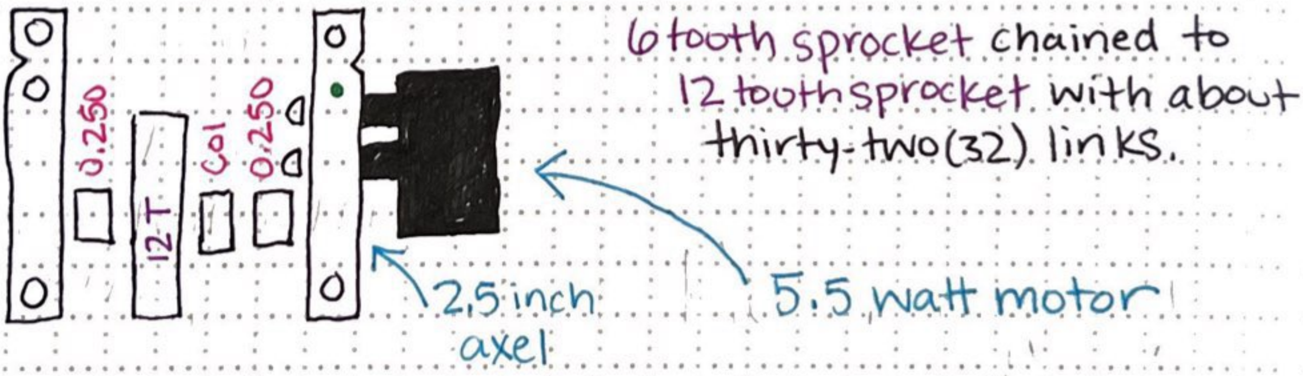
Building

(Continued)

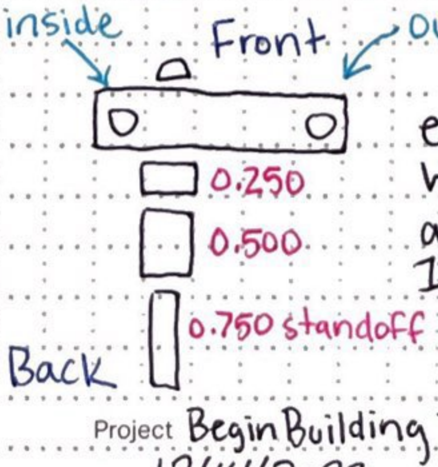
Goals for today → Get the intakes spacing put together

Below is a diagram of the intake spacing that goes on the structure from page 121.





Once we got all the spacing done, we added a 9 1/4 by 1 1/4 inch Lexan rectangle with our team name to the front of the intakes on the 14 long 1 by 1 L-channel using three (3) 0.500 inch screws and three (3) nylock nuts.

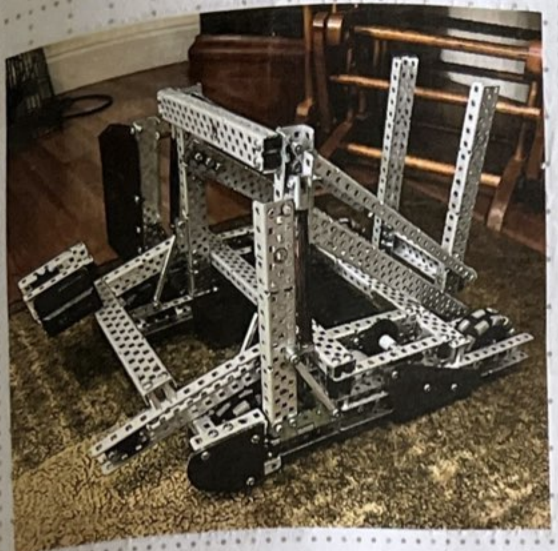
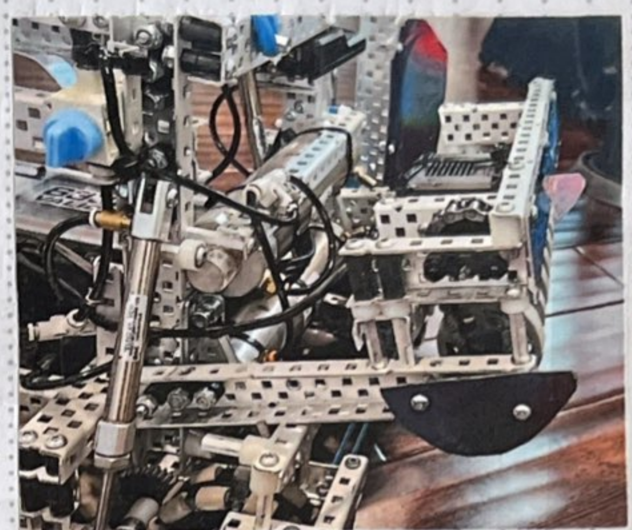
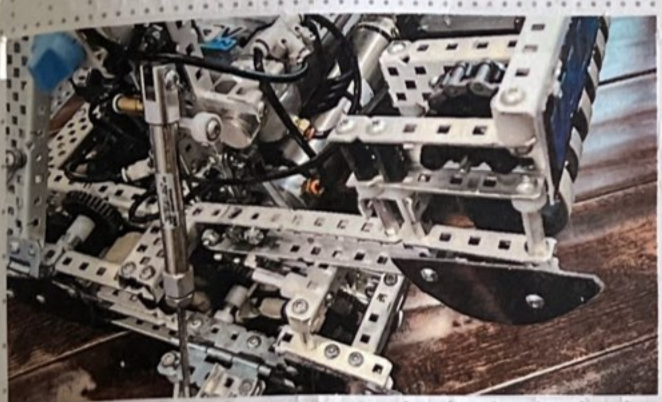


Then we added a five (5) long c-channel on each side of the base 2 holes back attached with a 1 inch standoff on the insides and a 0.5 inch spacer. The outsides used two (2) 1 inch standoffs coupled and a 0.500 inch spacer. The c-channel stops the wings from going down too far while the 0.750 inch standoff will stop the wings.

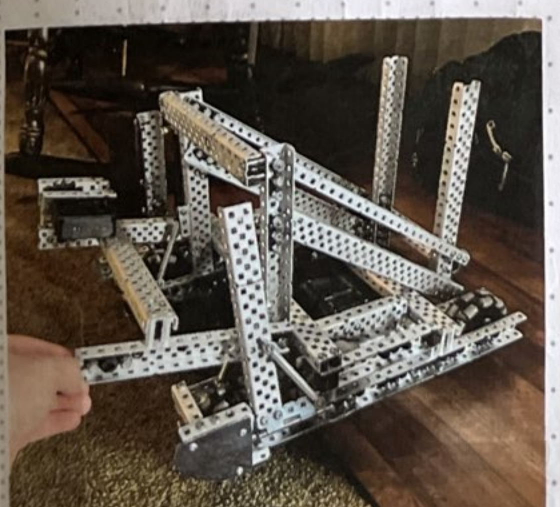
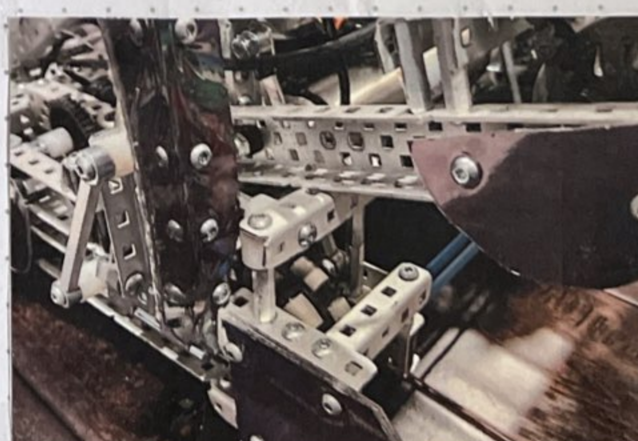
Project Begin Building Intakes (Continued) Name Vanessa Perkins

Date 12/14/2023

Finished Intake Pictures:

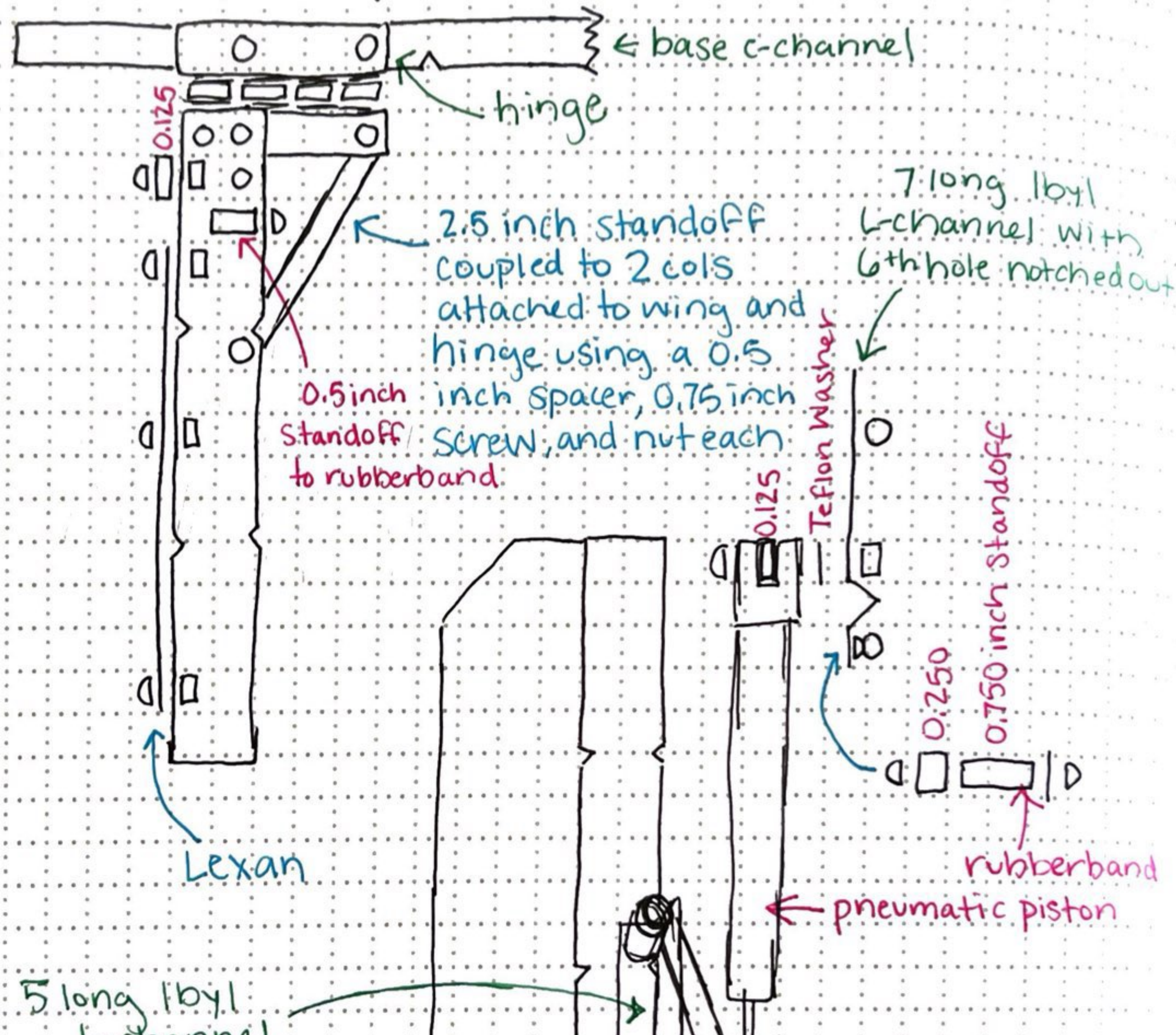
These pictures show the intakes in the build process and when they are completely done and attached to the robot.

Build Wings:

Goals for today → Get wings built
→ Attach wings to robot and pistons

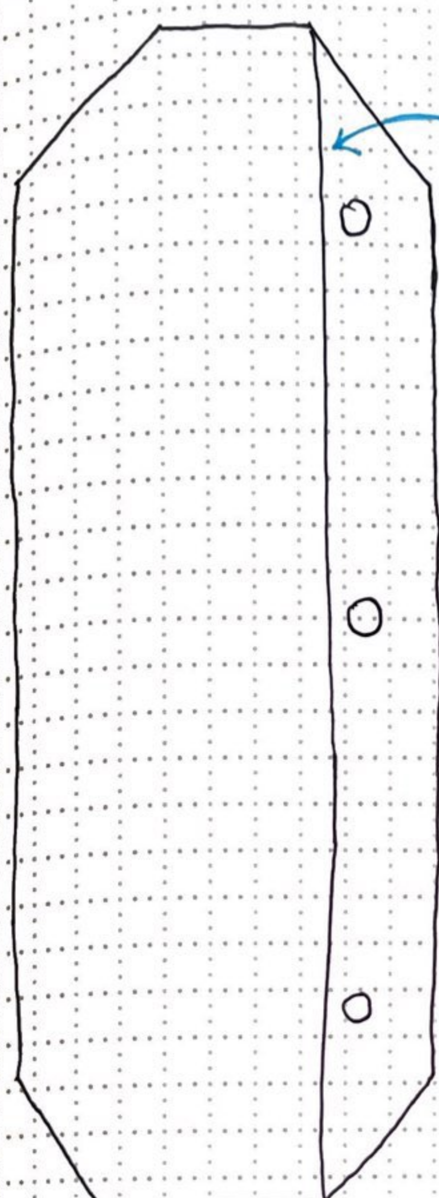
Below is the diagram for how to build the right side wing. The left side is mirrored. The top diagram is when it is folded down from an above view. The bottom diagram is when the wing is folded up from the outside.



Mirror and Repeat

Wings Lexan templates:

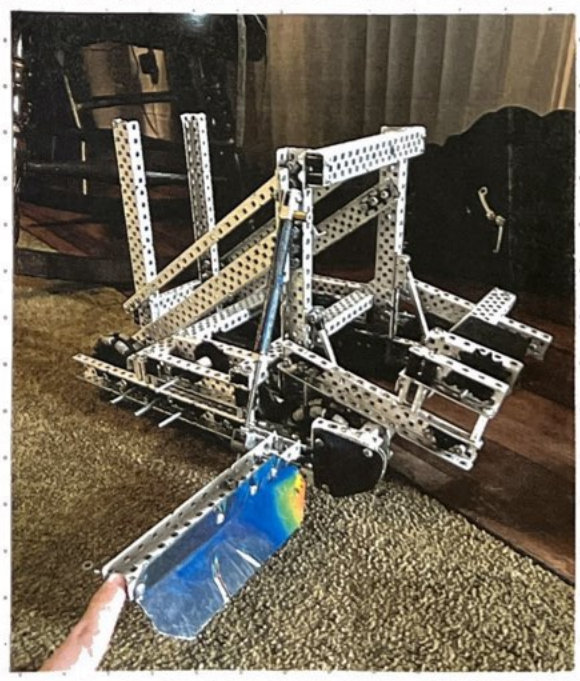
Below is a to scale template of the wings cut out of Lexan. The circles represent where we drilled holes to attach to the wings structure.



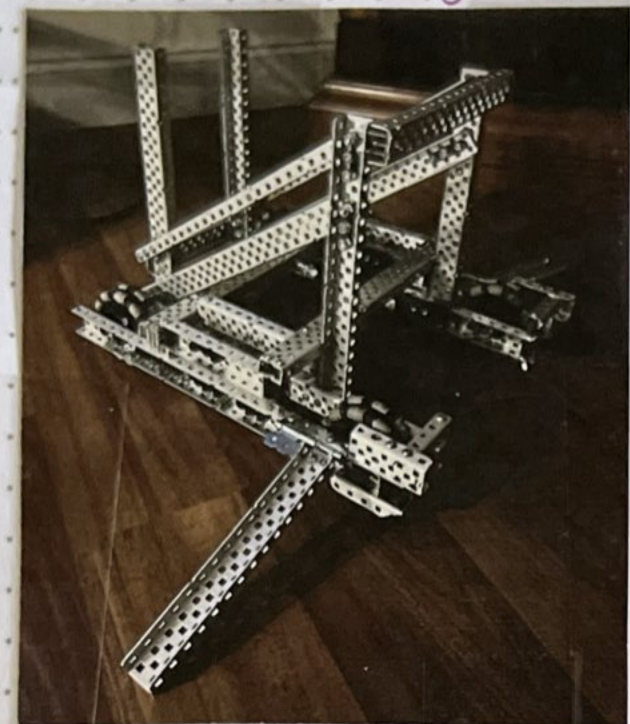
The line represents where we bent the Lexan at a 140° angle to help push triballs up and over the barrier.

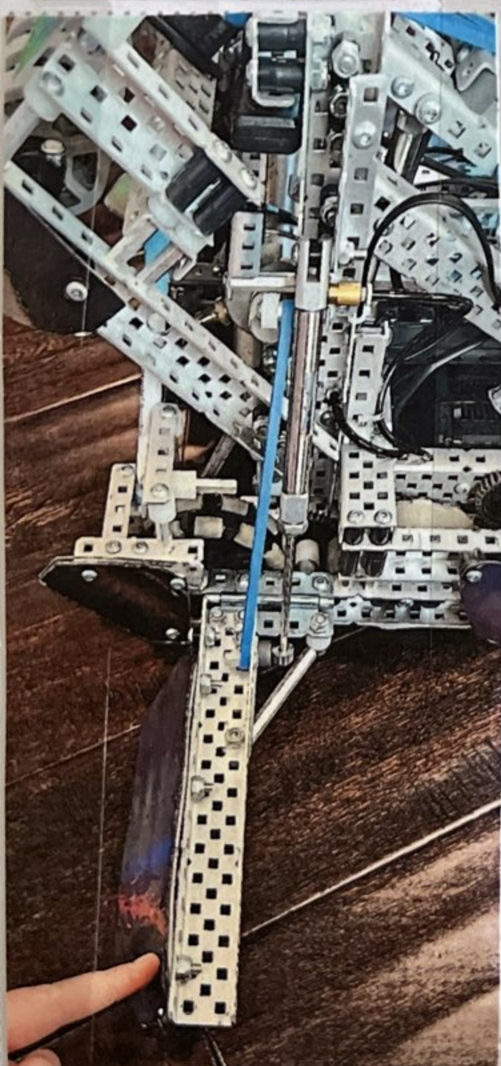
Cut out two (2) of these so there is one (1) for each wing.

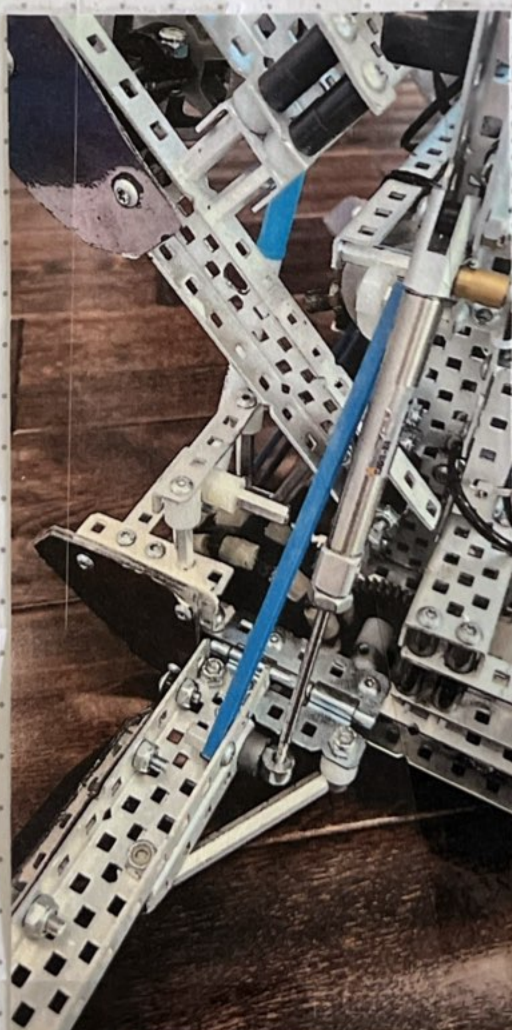
Below is an image of the Lexan sheet attached to the wing on the robot to show how it goes.

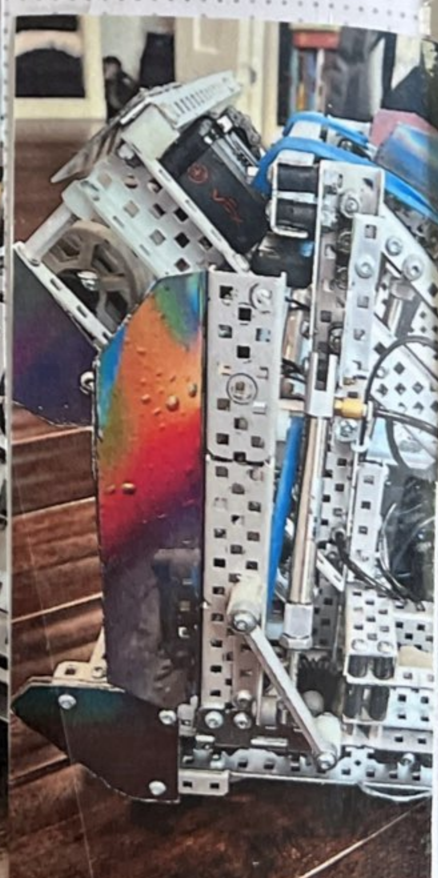


Finished Wings Pictures:





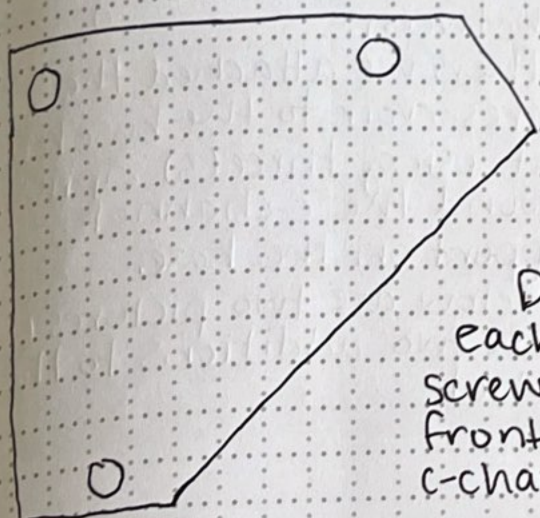




Attaching Sleds to Base:

Goals for today → Get the sleds made and attached to the base so we can drive over the barrier
→ work on other minor components of the robot

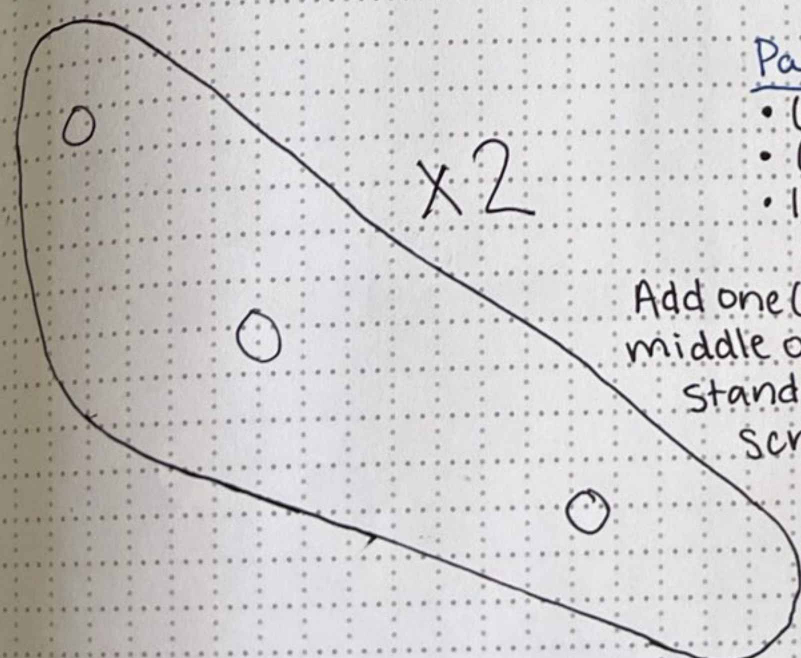
Below are templates for the front and middle sleds that we attached to the base with the screws and spacers necessary.



x4

Parts List:
- 6-0.500 inch screws
- 6 nylock nuts

Double two (2) Lexan sleds on each side and attach with the above screws in the three (3) holes to the front of the base through the c-channels.



Parts List:
- 6-0.500 inch standoffs
- 6-0.125 inch spacers
- 12-0.250 inch screws

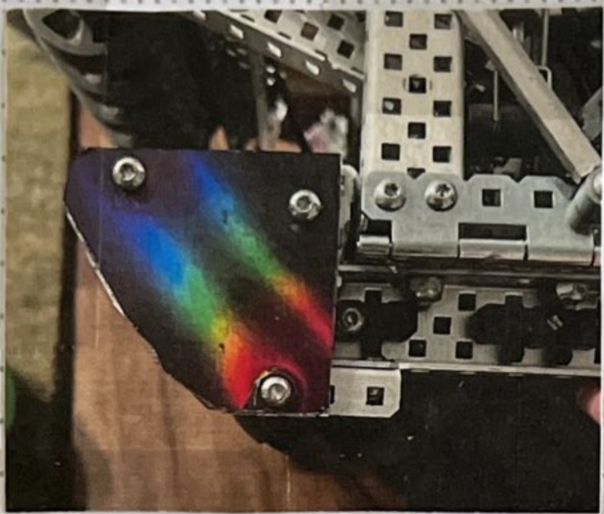
Add one (1) sled per side to the middle of the base using one standoff, one spacer, and two screws per hole. This sled helps the robot slide the rest of the way over the barrier once past the first two wheels.

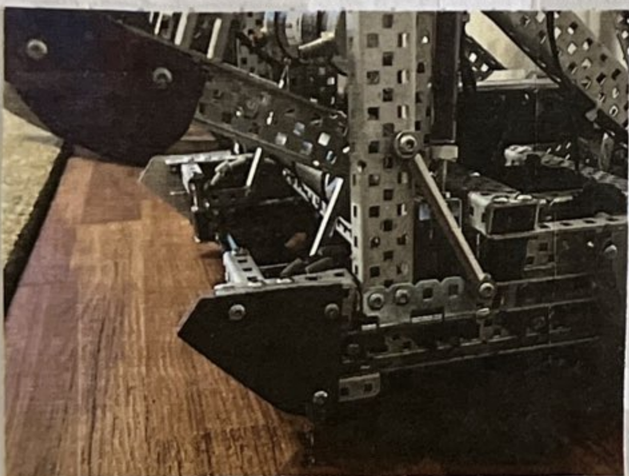
(Continued)

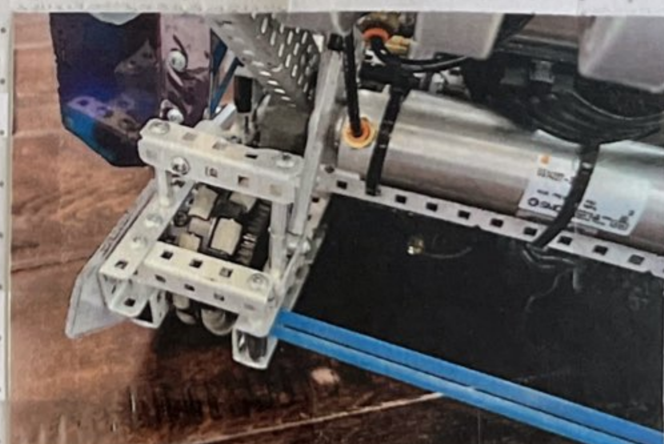
Below on the left side are images of the sleds attached to the base. Then we added a rubberband under the intakes to help hold triballs. We did this by adding two(2) 0.500 inch standoffs to the inside of the base with two(2) rubberbands stretching in between.

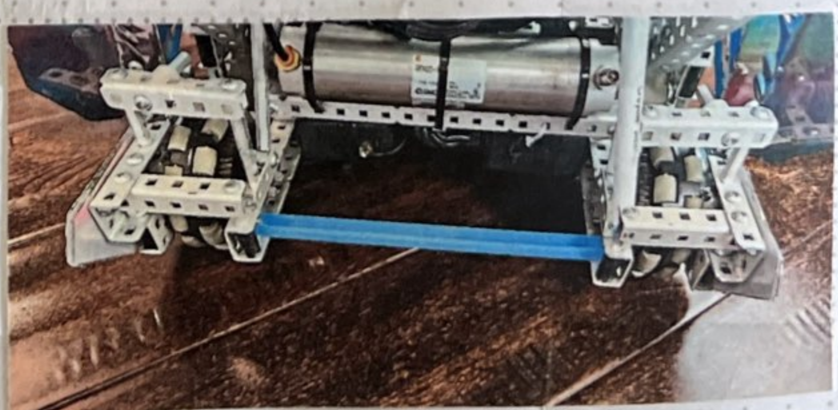
Then we attached the air reservoir to the base behind that using three(3) zipties around the c-channel support on the base.

Below are two pictures of these two additions to the robot.

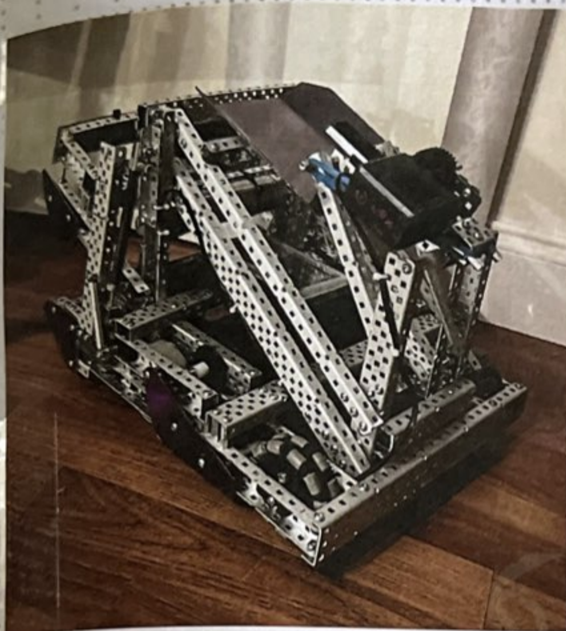


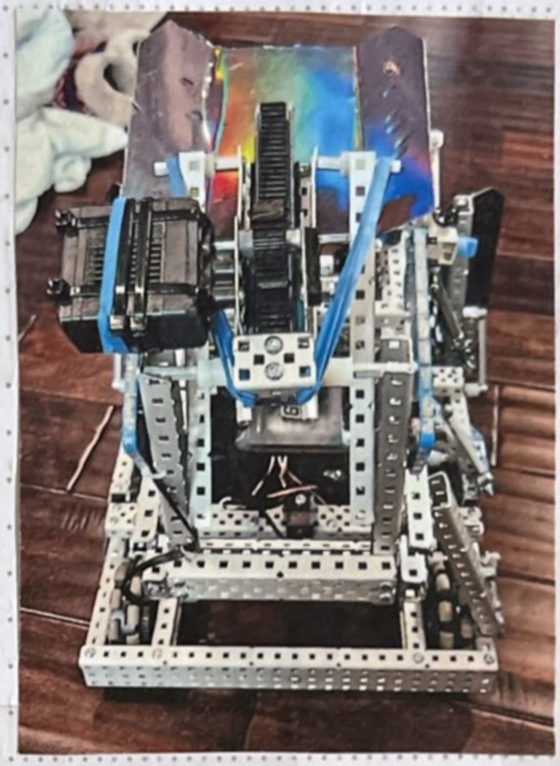


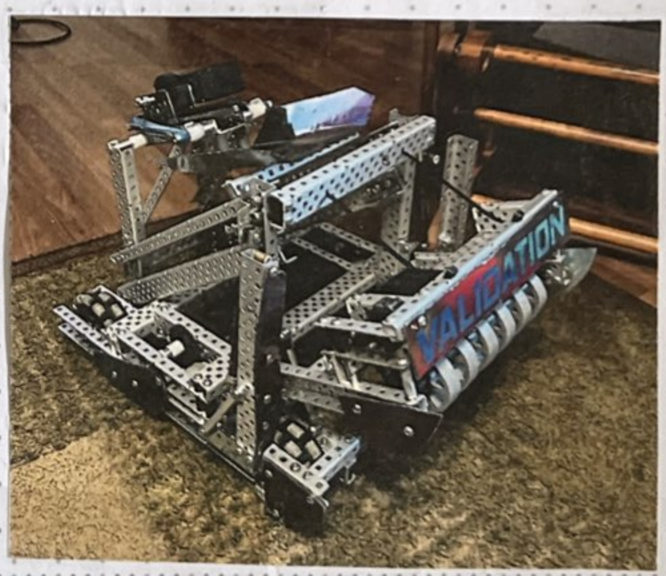




Full Robot Images:







VP

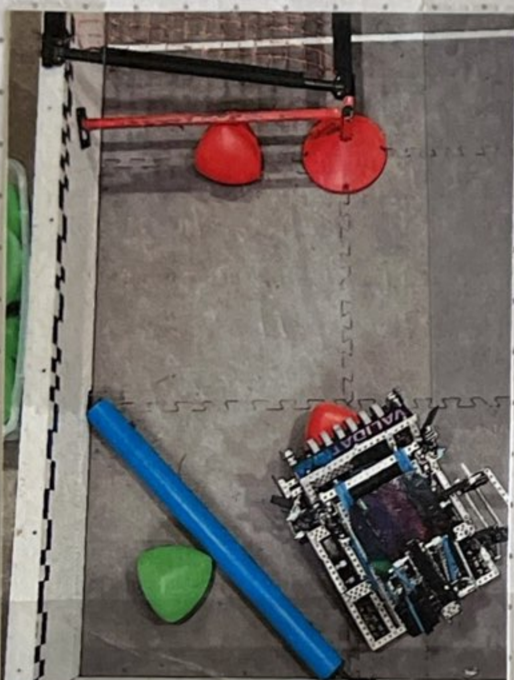
Project Full Robot Images
Name Vanessa Perkins
Date 12/20/2023

Testing Driver Skills Strategies:

Goals for today → Test different driver skills strategies to find the best one for Owen and our robot

To help find the best skills run strategy to practice, we ran three (3) tests that included different set ups and aiming of matchloads. The three (3) strategies are listed below with specifics and results.

Goal: 200 points in one driver skills run



Strategy 1:

Set up - depicted to left, 1 ~~match~~preload in intakes facing the red goal, 1 ~~match~~preload in front of red goal

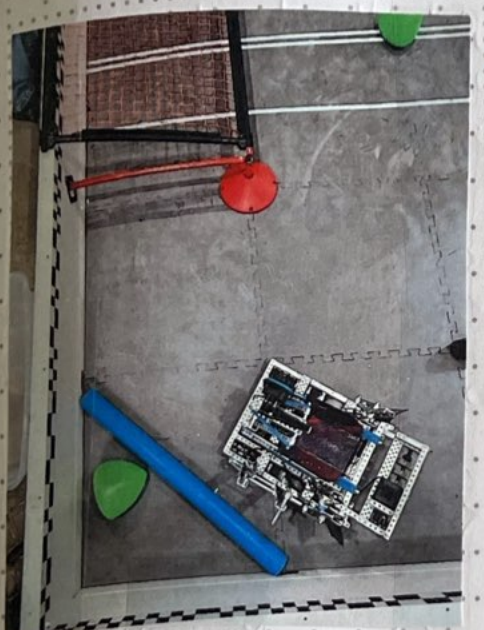
Aim - launch matchloads at center of the blue goal after scoring the two (2) preload alliance colored triballs in the nearside goal

Score - 145 points total, 23 triballs in the goals, field depicted below

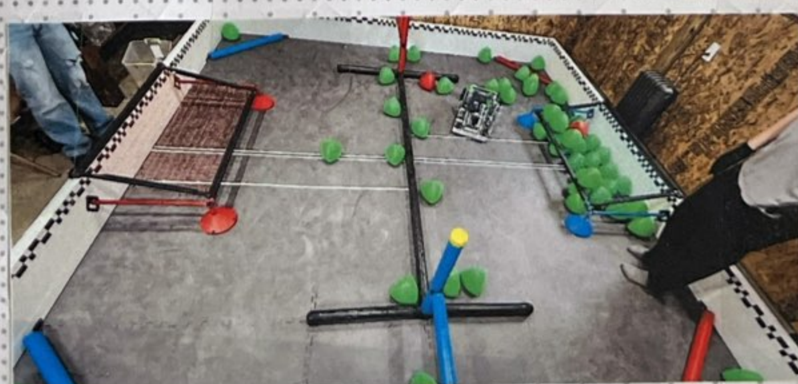
Results - scoring the preloads at the beginning before matchloading gave Owen trouble without having practice

(Continued)

Strategy 2:
Set up - same as strategy 1
Aim - launch matchloads at corner of field after scoring the two (2) preload alliance colored triballs in the nearside goal then plow matchloads into the goal
Score - 131 points total, 21 triballs in the goals
Results - aiming matchloads at the corner of the field isn't as consistent as the center of the goal



Strategy 3:
Set up - depicted to left, no preloads, back of robot facing the matchload zone
Aim - launch matchloads at center of the blue goal including the preloads then plow them all into the goal
Score - 132 points total, 20 triballs in the goal, field depicted below
Results - takeaway is that we need more practice plowing and matchloading



Project Testing Driver Skills Strategies (Contin.) Name Vanessa Perkins
Date 1/13/2024

Budget Update:

Goals for today → Review our plans and budget
→ Make adjustments as needed

At the beginning of the year our team created a budget. After this last competition, we wanted to review it.

At the moment we have a breakfast fundraiser planned for February to refill our funding. We are currently registered for States so we still need those travel costs.

To save money we decided to go to the local Signature Event, Bots at Bristol, than an out of state one that requires hotels.

We have done fairly well with reusing parts and not having to buy replacement parts.

Overall we hope that our fundraiser ~~to~~ VP will help us stay on track with our budget.

(Refer to old budget on page 9, book 1.)

VP

Project Budget Update
Name Vanessa Perkins
Date 1/14/2024

Brainstorming Robot Design Changes:

Goals for today → Review current game strategies and the performance of the robot
→ Brainstorm Design Improvements

Problem/Challenge

Our current robot performs well in practice, but doesn't fit current strategies.

Brainstorming

① Keep current robot
② Keep base and rebuild the rest
③ Completely Rebuild Robot

Select Approach

②

Due to limited time, and no necessary reason to rebuild the base, we will just be redoing the top subsystems.

Build/Implement Solution

- Disassemble almost everything except the base
- Decide on and build the new subsystems that are better for current game strategies.

VP

Project Brainstorming Robot Design Changes

Name Vanessa Perkins

Date 1/15/2024

VEX ROBOTICS

PROPRIETARY INFORMATION

Disassembling:

Goals for today → Get the robot ready for the rebuild

After a team meeting where we brainstormed our plans moving forward, we decided to rebuild some of our subsystems.

To prepare for this we disassembled everything except for the following parts we will be keeping and minorly updating to save time.

Keep: - base
- intakes
- front wings

Disassemble: - everything else

VP

Project Disassembling
Name Vanessa Perkins
Date 1/15/2024

Brainstorming Intake Improvements:

Goals for today → Discuss and decide on a way to improve the intakes without having to completely rebuild

Problem/Challenge

The flexwheel intakes aren't gripping the triball good enough.

Brainstorming

1. Switch to rubberband intakes
2. Adjust the angle
3. Add rubberbands underneath

Build/Implement Solution

Tomorrow we will build these.
- Keep structure
- Replace spacing and actual intake

Select Approach

1

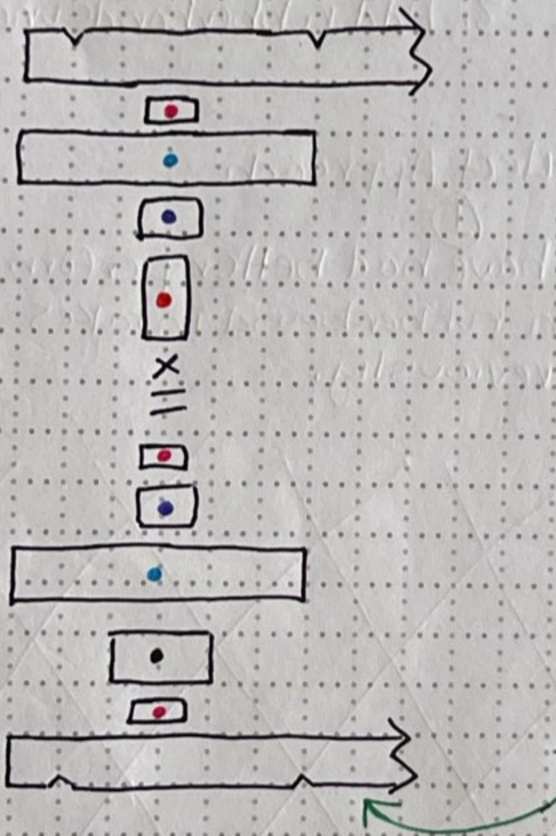
We have had better performance with rubberband intakes previously.

VP

Intake Update:

Goals for today → complete the minor adjustments
→ Test for improvement

Below is the diagram for how we implemented our chosen improvement of switching from flex wheels to rubberbands on the intakes.



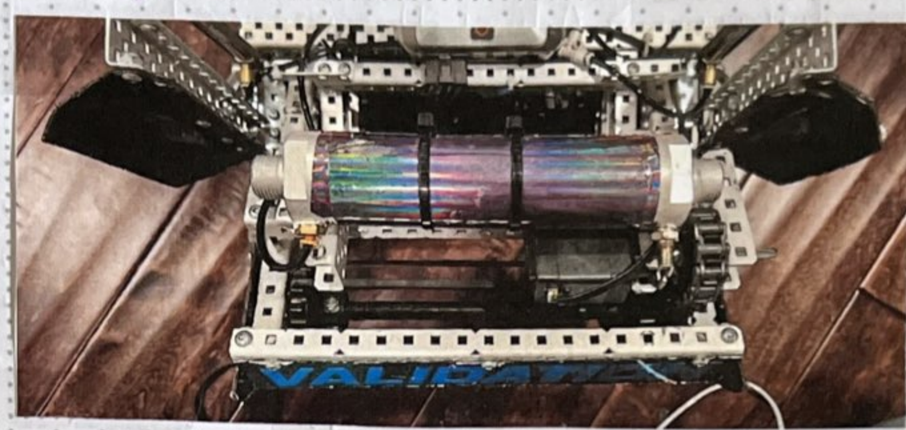
Parts List:

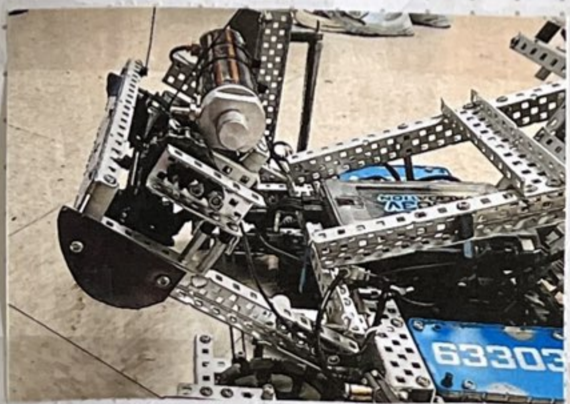
- Old Intake Structure (page 121-123 book 2)
- Old axel
- 3-0.125 inch high strength spacers
- 11-0.500 inch high strength spacers
- 2-18 tooth sprockets
- 1-6 tooth sprocket
- 2-high strength collar locks
- 4 rubberbands

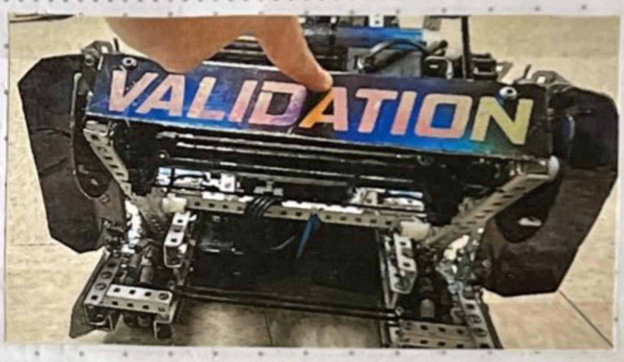
We removed the flex wheels and spacing off the old axel then we added the new above spacing. Then we rechained the 6 tooth sprocket to the motor. Lastly we stretched about four (4) rubberbands evenly around the two (2) 18 tooth sprockets to create a combine.

When tested, this intake had better results because the rubberbands form and grip better around the triballs.

(continued)





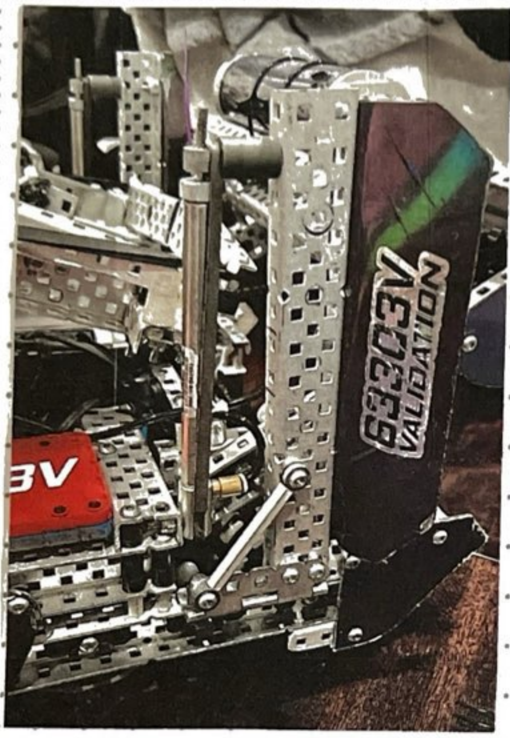


Above are images from several different angles of the finished, updated intakes.

We also left the air reservior ziptied to the top of the intakes which acts as a weight to help put more compression on the triballs to keep them inside the intakes.

Wings Update:

Goals for today → Adjust the wings to meet the new formate of the robot without having to completely rebuild



Since the 4 bar lift is no longer on the robot, we had nothing in the front to attach the end of the piston to that activates the wings.

Our solution, as depicted to the left, was to flip the piston and attach the bottom to the front of the c-channel base support that is just behind the wing.

Then the end of the piston attached to the end of the wing using a 0.500 inch ~~screw~~ VP spacer, col, and 0.750 inch screw through the 2nd to last hole.

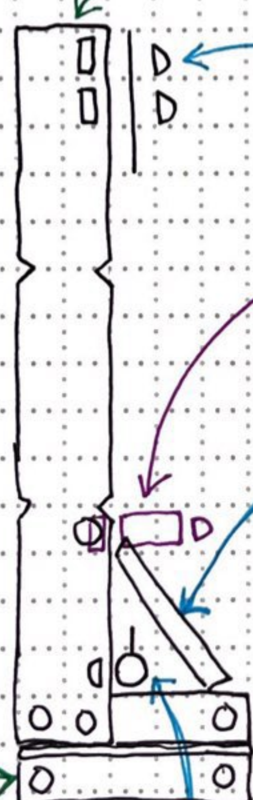
This was a quick and easy fix that functions just as well as the previous piston attachment method.

VP

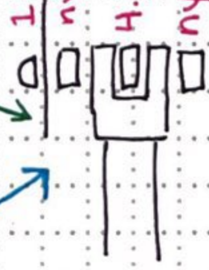
Building Matchload Mechanism:

Goals for today → Build and attach a mechanism that gets the triball out of the matchload zone in autonomous

Below is the diagram for how we built the matchload mechanism and attached it to the robot.



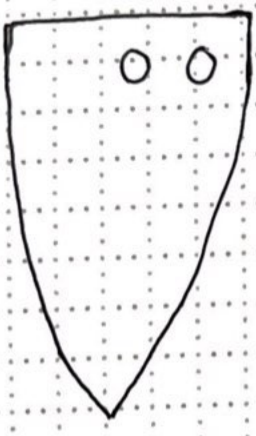
The other end of the piston is attached 3 holes down on the lift support with the pieces to the right.

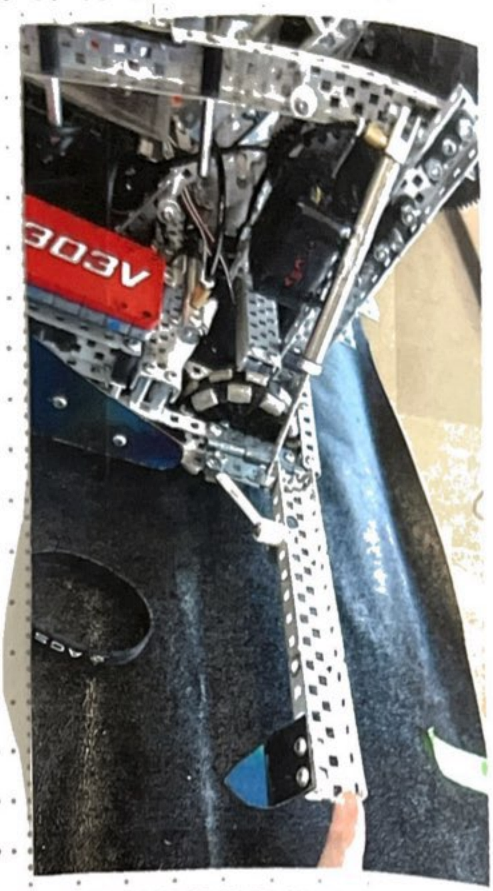


(Continued)

To the right are images of the finished matchload mechanisms attached to the robot.

Below is also a template of the Lexan hook we attached to the very end to lift triballs out of the corners during the autonomous.





×2

VP

Engineering Notebook

63303V
Team Number

Validation
Team Name

Johnson County High School
School

1/17/2024
Start Date

End Date

3
Book #

of

VEX ROBOTICS

REC FOUNDATION

Table of Contents

Table of Contents

Hang/Shooter Mechanism Design Options:

Goals for today → Come up with atleast three ideas for launch mechanisms to choose between

Design Option One: PTO Kicker/Hang

Specifics:

- 1 motor
- 1 pto mechanism
- slip gear for kicker
- ratchet for hang

This design allows us to build a more complex hang that is run ~~by~~ the same motor as the kicker.

The pto will be worked using a pneumatic piston.

The kicker will allow for fast matchloading.

VP

(Continued)

Design Option Two: Puncher

Specifics on Page 31, book 1.

Design Option Three: Catapult

Specifics on Page 29, book 1.

Select Approach:

Goals for today → Choose a launch mechanism design that best meets the requirements and current strategies of the game so we can begin building

We created a decision matrix of what aspects mattered the most on our launch mechanism. Below is the matrix and winning design.

Design Ideas	Speed	Versitility	Power	Total
PTO Kicker/Hang	4	5	4	13
Puncher	5	0	5	10
Catapult	3	0	3	6

Speed: how fast can it launch triballs
Versitility: how adaptable is it to different strategies or functions
Power: how strong is the design

Selected Design: PTO Kicker/Hang
Explanation: This design ultimately won because of its versitility. With a six (6) motor drive, we have to use our remaining two (2) motors wisely. This design is an effective and fairly simple way to share and save motors.

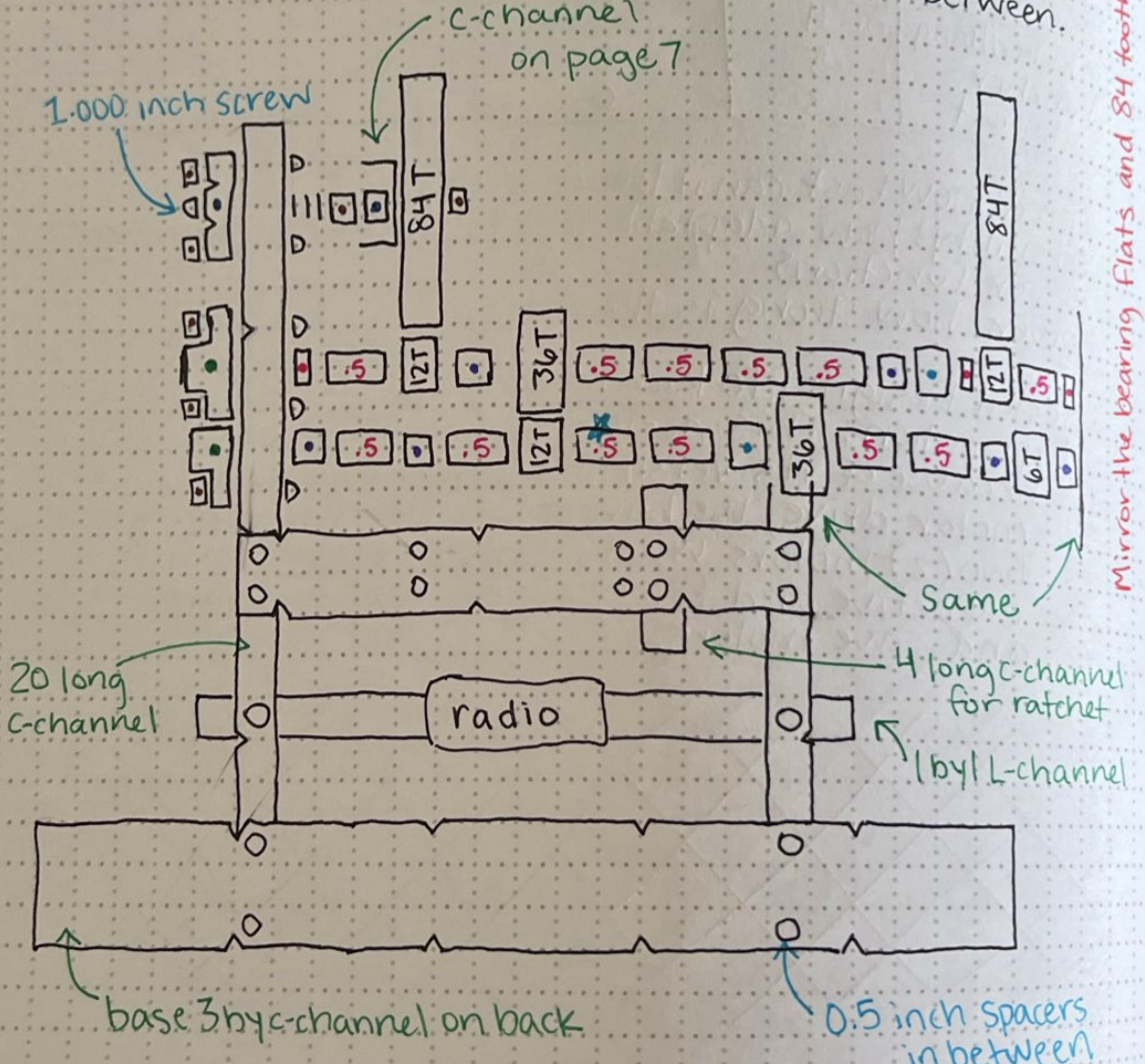
VP

Begin Hang Mechanism Gearing:

Goals for today → Begin building the hang mechanism structure and gearing

Below is a diagram for how we built the hang mechanism structure and gearing. The gearing is Spaced out to help distinguish between pieces.

The 1by1 with the radio attached was attached to the hang mechanism using a 0.5 inch and 0.375 inch spacer in between.



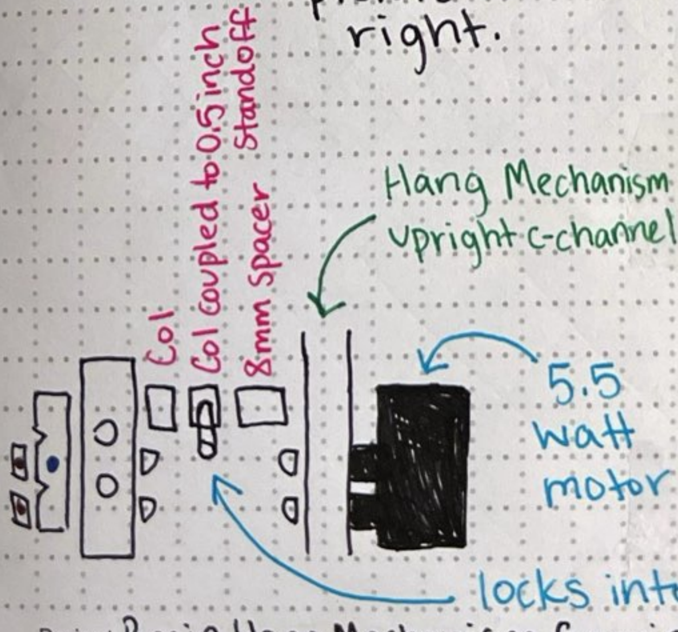
(Continued)

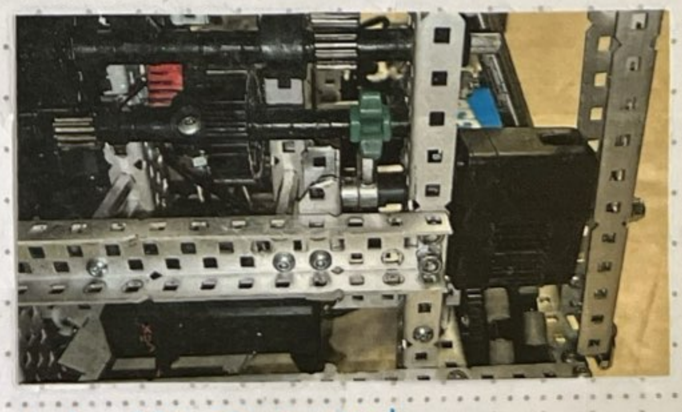
Parts List:

- 2-20 long aluminum c-channels
- 1-14 long aluminum c-channel
- 1-16 long 1by1 aluminum L-channel
- 1-4 long aluminum c-channel
- 2-84 tooth gears
- 3-12 tooth gears
- 2-36 tooth gears
- 1-6 tooth sprocket
- 4 bearing flats •
- 4 high strength bearing flats • (2 of them with 1 hole cut off)
- 3-0.125 inch spacers •
- 18-0.500 inch spacers
- 6-0.250 inch spacers •
- 2-0.375 inch spacers
- 6 Teflon Washers
- 2 High strength collar locks •
- 2 Collar locks
- 1-8 mm spacer
- radio
- 1 coupler
- 1-0.500 inch standoff
- 1-5.5 watt motor
- 28 nylock nuts •
- 2-1.000 inch screws
- screws
- 2-7 inch high strength shafts
- 1-2 inch shaft

VP

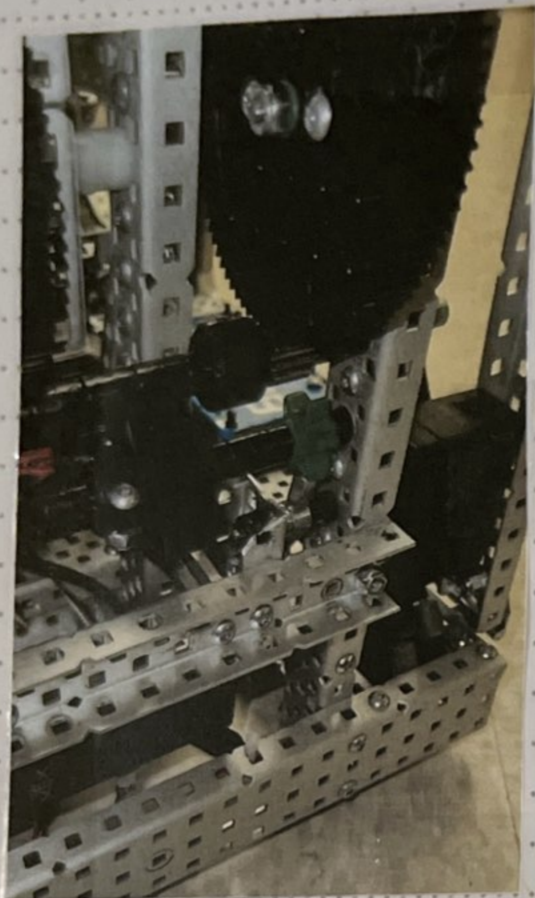
Ratchet directions below with picture to the right.

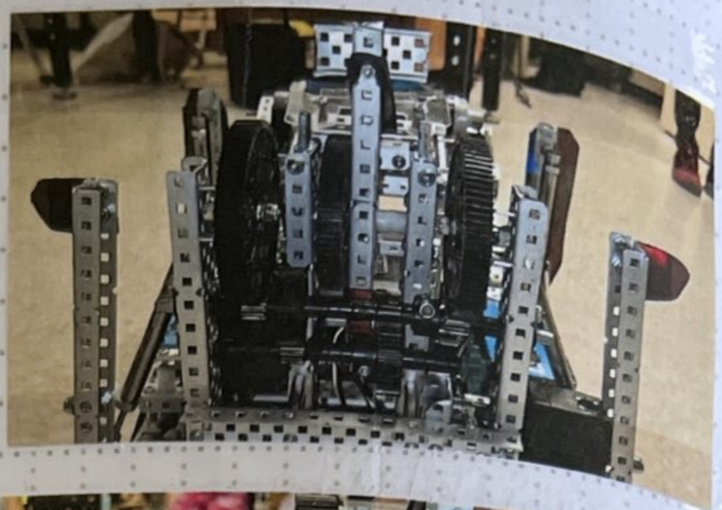


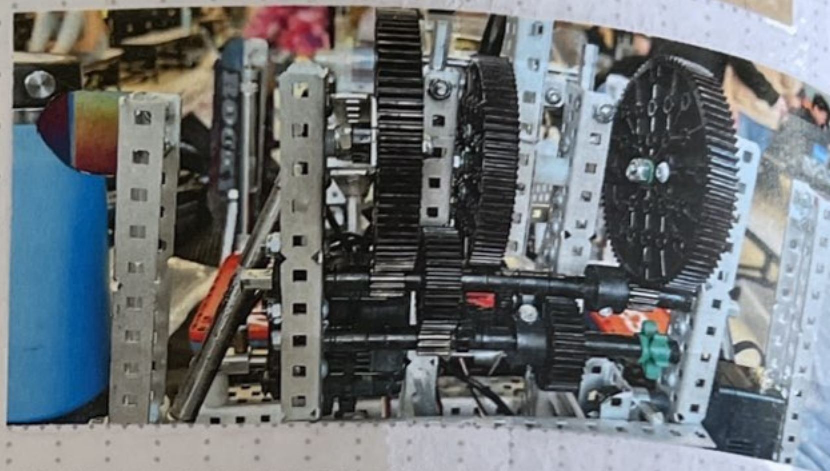


Project Begin Hang Mechanism Gearing (continued)
Name Vanessa Perkins
Date 1/18/2024

(Continued)

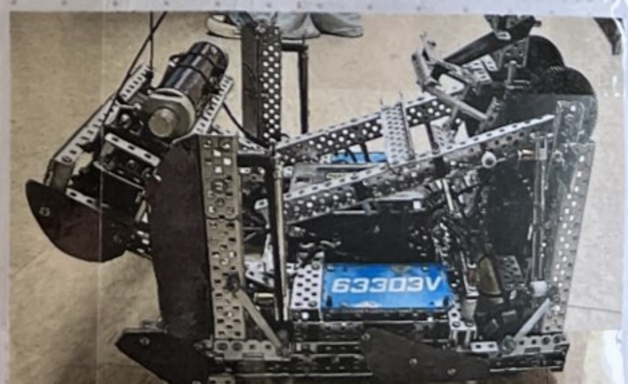






Above are images of the hang mechanism gearing attached to the robot from several different angles. On the left is a close up of the ratchet engaged to keep the lift from moving.

VP

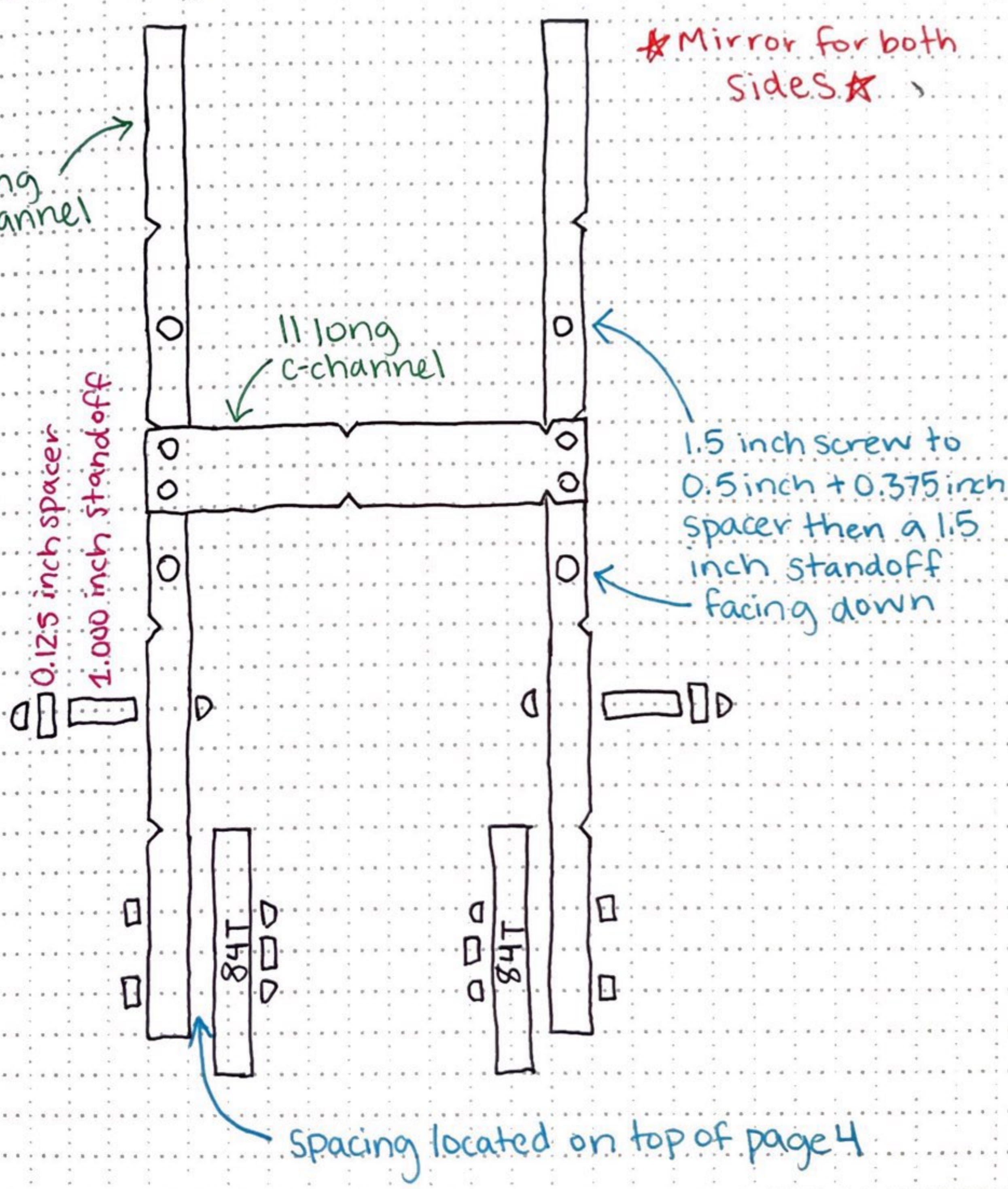


Finished Hang Mechanism:

Goals for today → Get the hang mechanism aspect of the robot finished

Below is the diagram for building the arm that attaches to the gearing from page 4. The arm is the physical part that hangs on the elevation bar with the help of gearing.

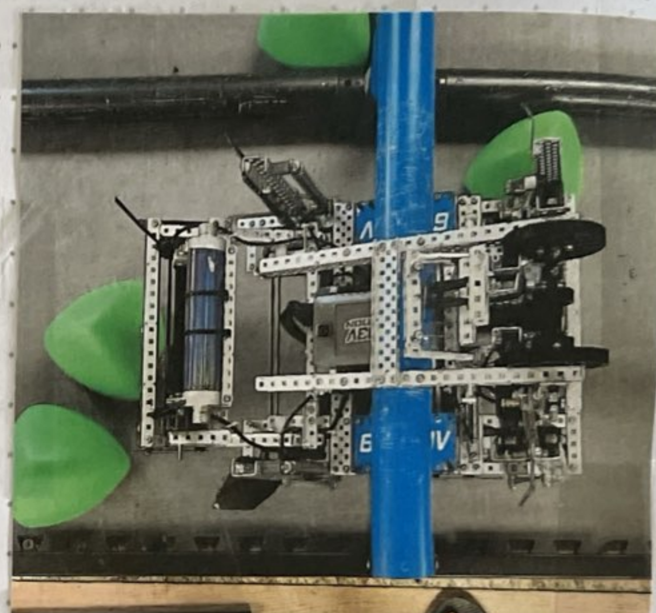
Side view of the finished arm on robot

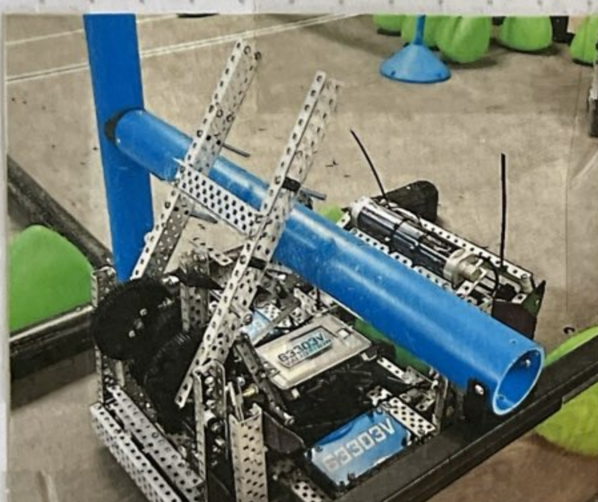


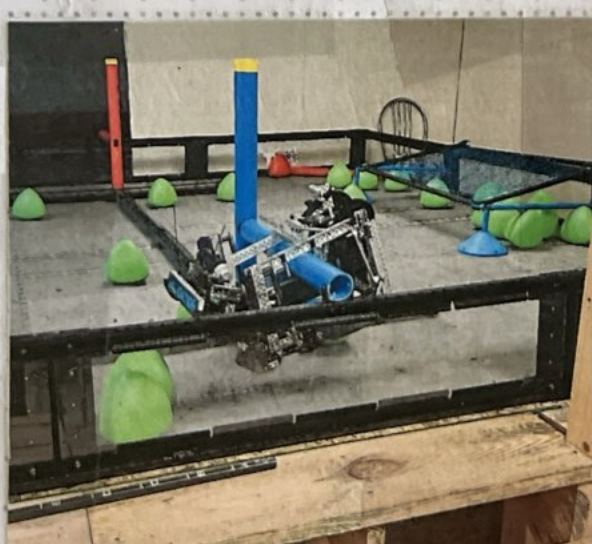
Testing Hang Mechanism:

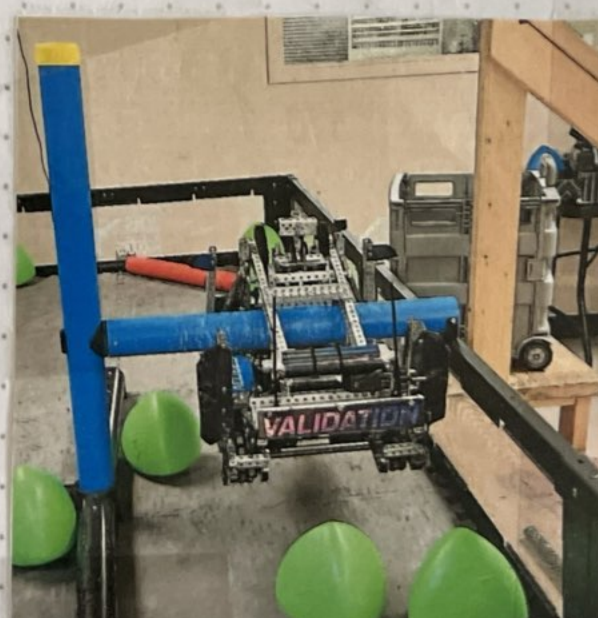
Goals for today → Make sure the mechanism works and the robot is strongenough to elevate itself
→ adjust as necessary

Below are images of our hang being tested. It worked well and did not need any adjustments. Owen just drives up to the elevation bar with the hang mechanism raised. Then he hooks on, lowers the arm as the robot raises. Lastly he runs the ratchet motor to keep the hang mechanism from releasing when the match ends.







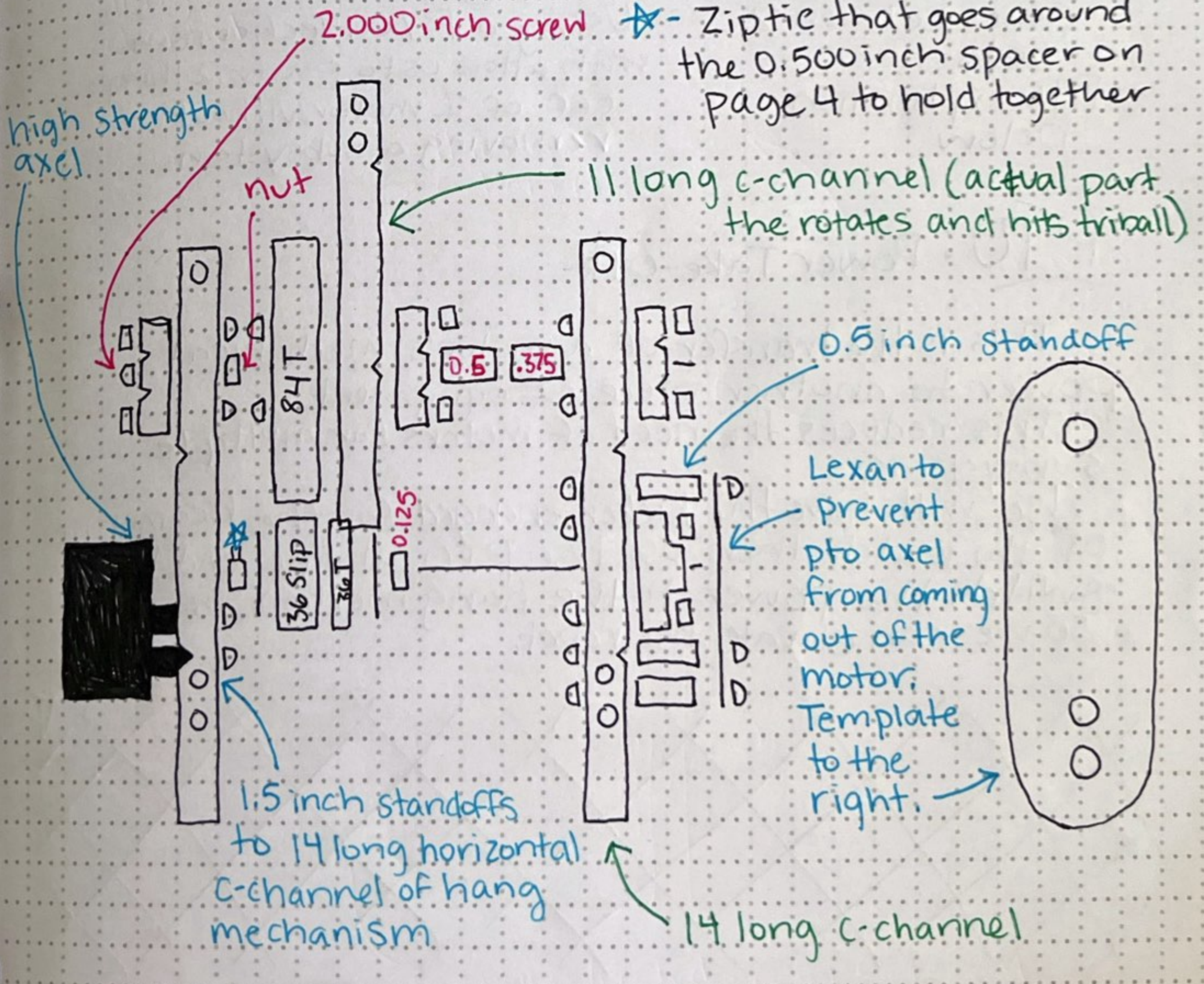


Kicker Gearing:

Goals for today → Begin the gearing for the kicker
→ Attach to the hang mechanism

Below is the diagram for building the kicker gearing and attaching it to the hang mechanism gearing.

For the 36 tooth gear we took off 12 (twelve) teeth to create a slip gear.



Designing PTO:

Goals for today → Solve our challenge of needing more motors to power the subsystems we're building

Problem/Challenge
We are out of motors to run all of our subsystems seperately.

Brainstorm
1. Remove a subsystem
2. Incorporate PTO

Select Approach
2
Using a PTO Mechanism with allow us to power 2 things off of 1 motor without removing a subsystem

Build/Implement Solution
- Use piston to move gears between 2 sets
- PTO explanation below

PTO: Power Take-Off

This is the transfer of a motor's mechanical power to another piece of equipment

This reduces the need of motors for multiple subsystems.

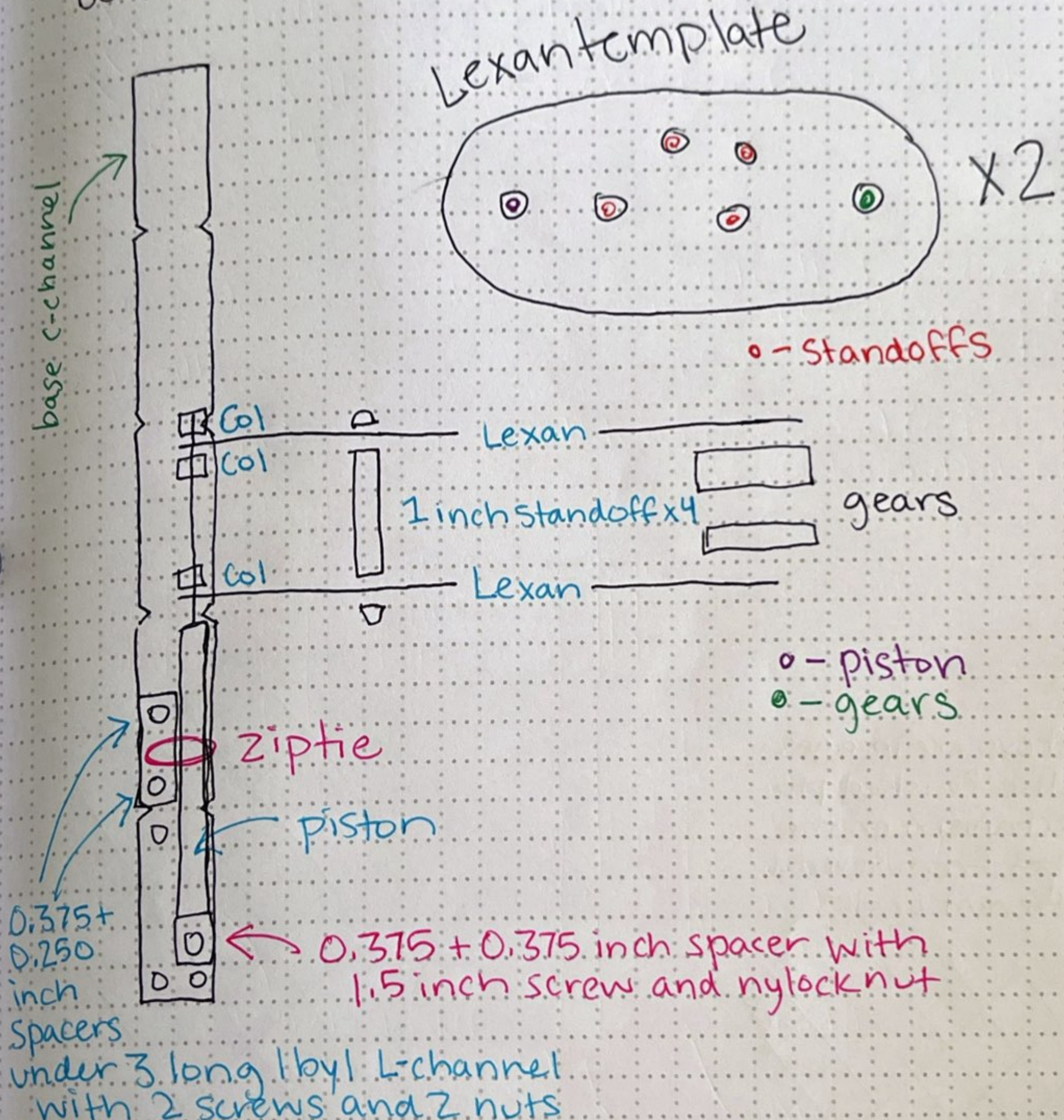
We will have the kicker engaged for the beginning of the match then use the PTO at the end to switch the power to the hang mechanism so we can elevate the robot.

VA

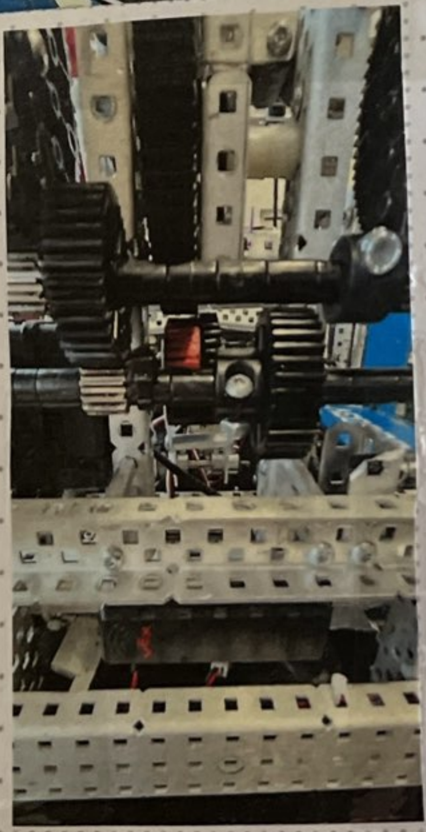
Building PTO:

Goals for today → build the pto and get it attached to the gearing and robot

Below is the diagram for how we built the pto.



PTO Images:



Above are images of the finished pto mechanism on the robot from several different angles.

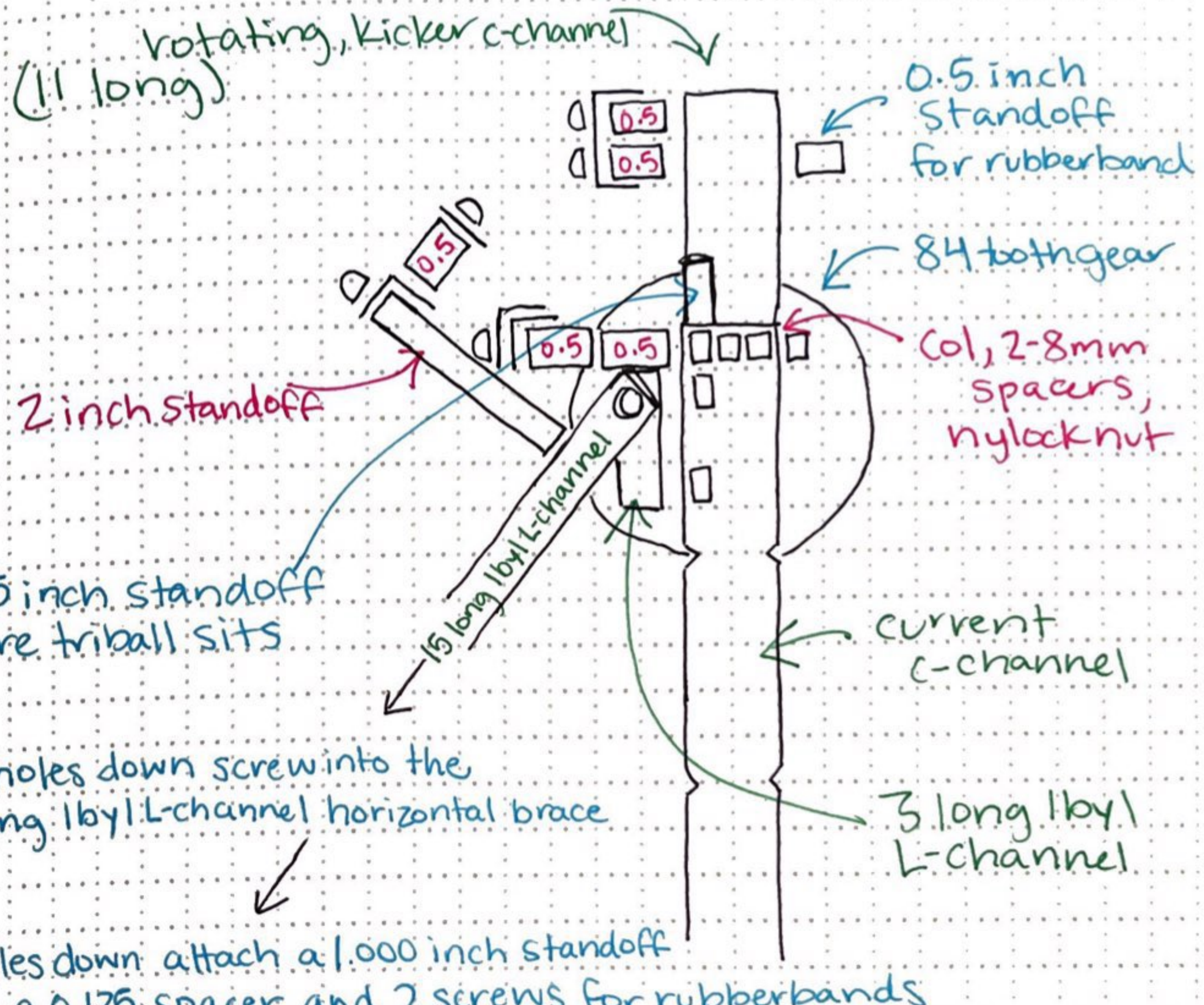
VP

Project PTO Images
Name Vanessa Perkins
Date 1/20/2024

Building the Kicker:

Goals for today → Build the kicker structure and attach to the robot

Below is a diagram from the side of how we began to build the kicker mechanism that will launch triballs.



6 holes down screw into the 8 long 1by1 L-channel horizontal brace

↓

11 holes down attach a 1.000 inch standoff with a 0.125 spacer and 2 screws for rubberbands

↓

15th hole down add a 0.500 inch spacer and 2.000 inch standoff into the base c-channel

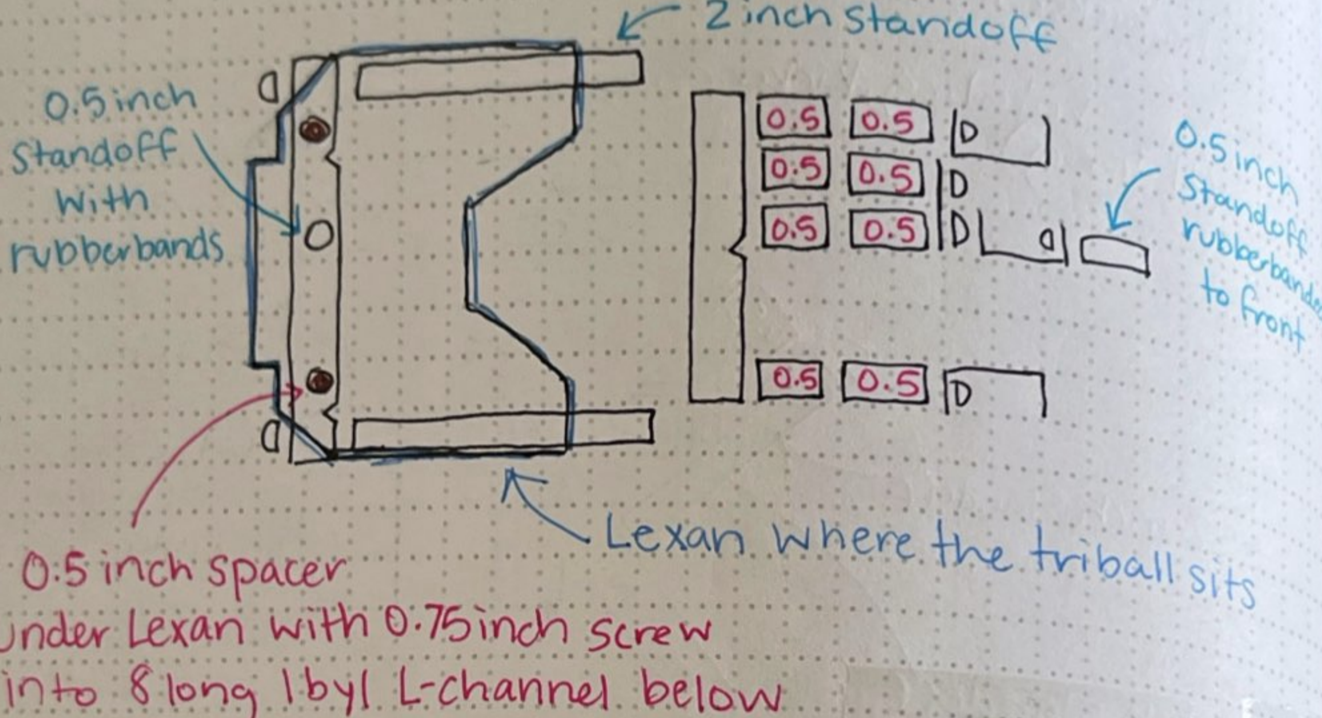
Project Building the Kicker

Name Vanessa Perkins

Date 1/21/2024

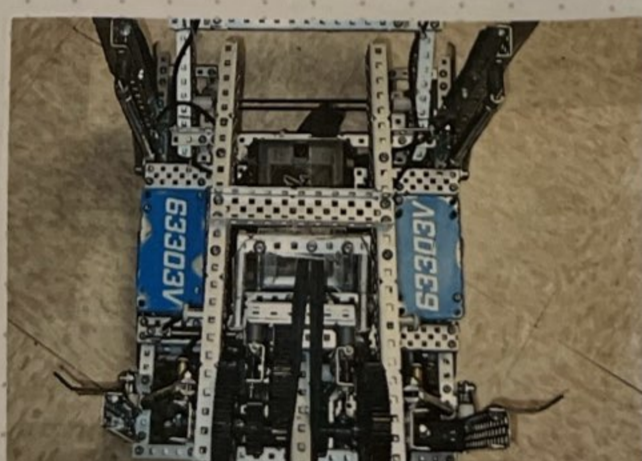
(Continued)

Below is the rest of how to build the kicker in a diagram. It is from the top view. This part of the build is where the triball sits to be hit and launched.



Below are pictures of the top part of the kicker on the robot.

• - where triball sits



Project Building the kicker (Continued)

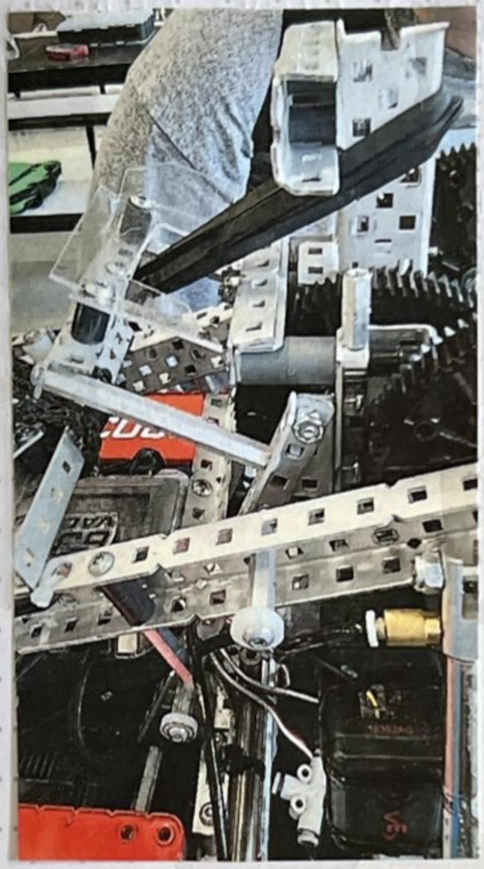
Date 1/21/2024

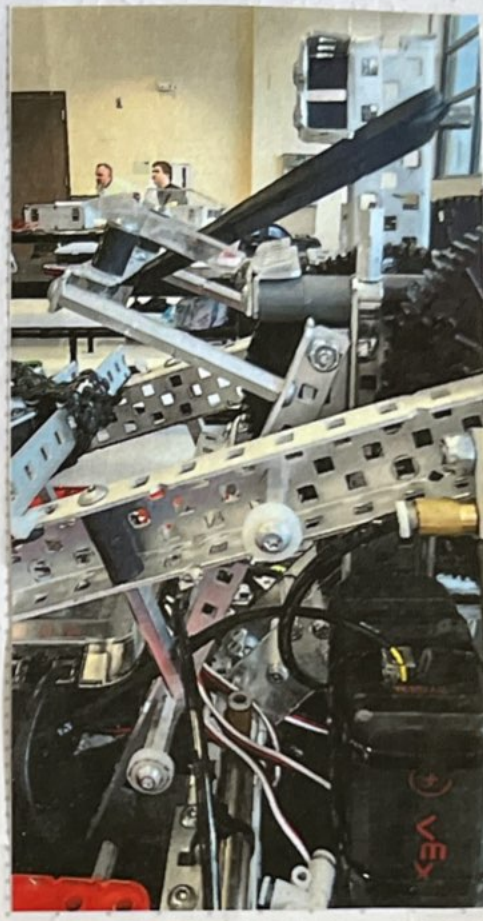
Name Vanessa Perkins

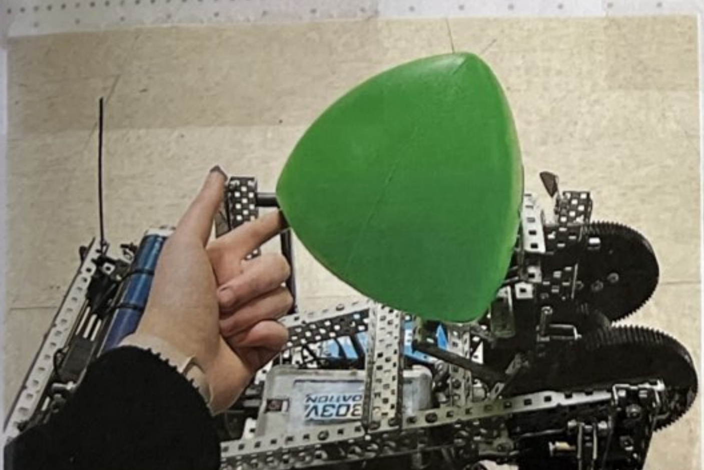
Finished Kicker Images:

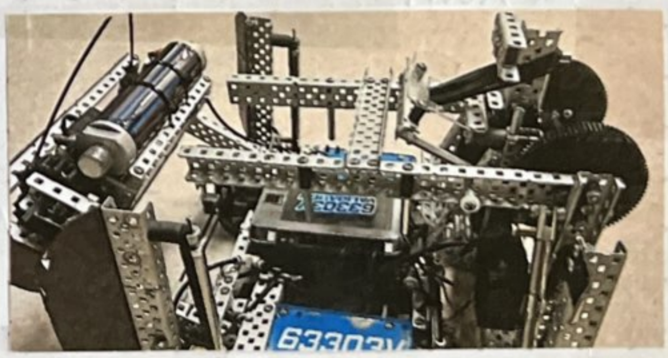
Below are images of the finished kicker from various angles as well as one picture with a triball on top of the kicker to show its placement on the robot.

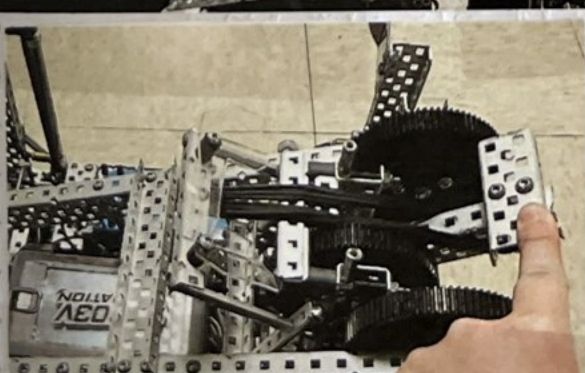
VP

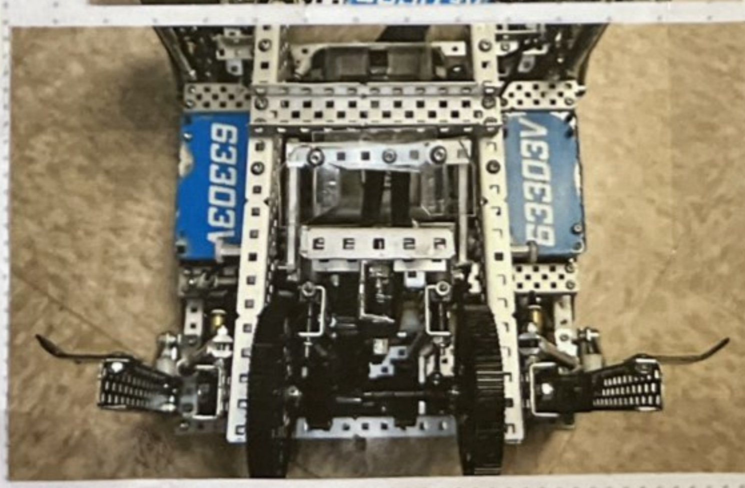












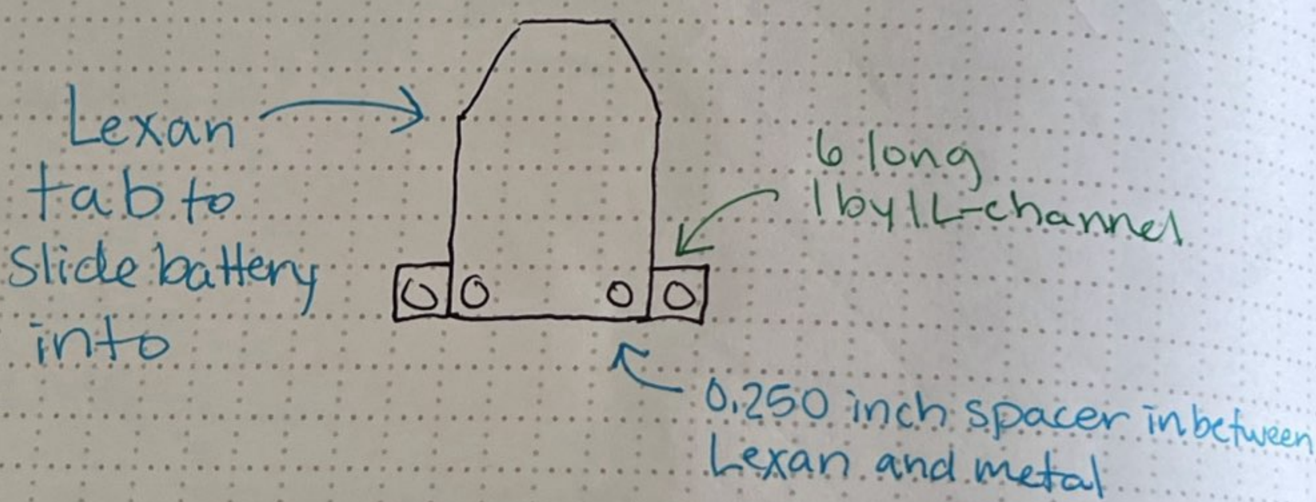
Attaching Brain and Battery:

Goals for today ⟶ Get brain, battery, and the license plates attached to the robot

First we added a brain to the base by screwing it to the very center using four (4) screws.

Next we added license plates to the sides of the brain facing flat. We screwed the blue ones down and then use pins to keep the red ones on top for easy switching.

Below is the diagram and instructions for attaching the battery. We added it under the base.



The above L-channel attaches to the base using a 0.375 + 0.250 spacer + 0.500 inch standoff to the base C-channel on each side.

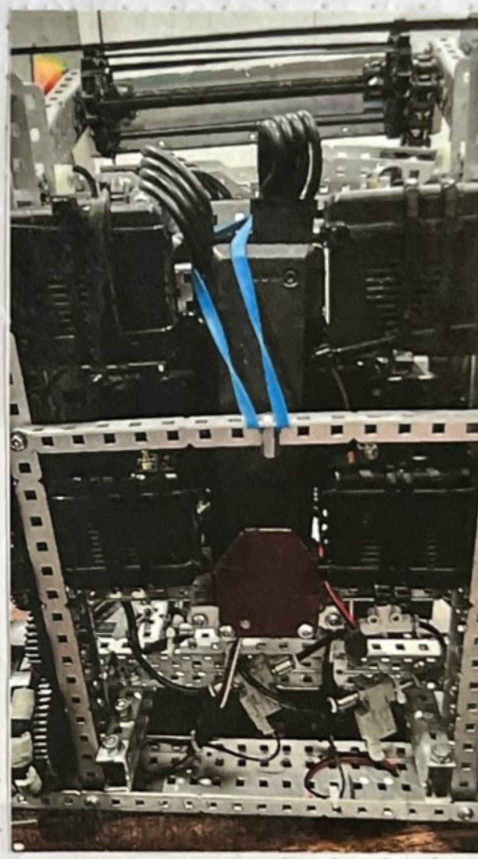
We ziptied a rubberband to the top of the base by the brain and pull it to a half inch standoff under the base to help secure the battery.

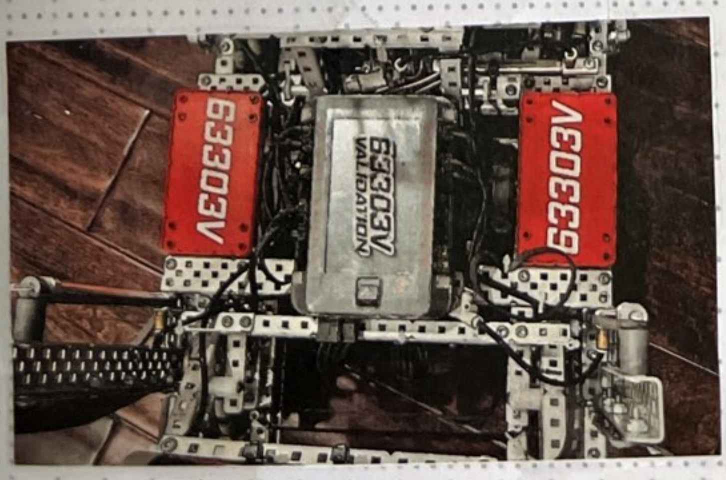
Project Attaching Brain and Battery
Date 1/22/2024
Name Vanessa Perkins

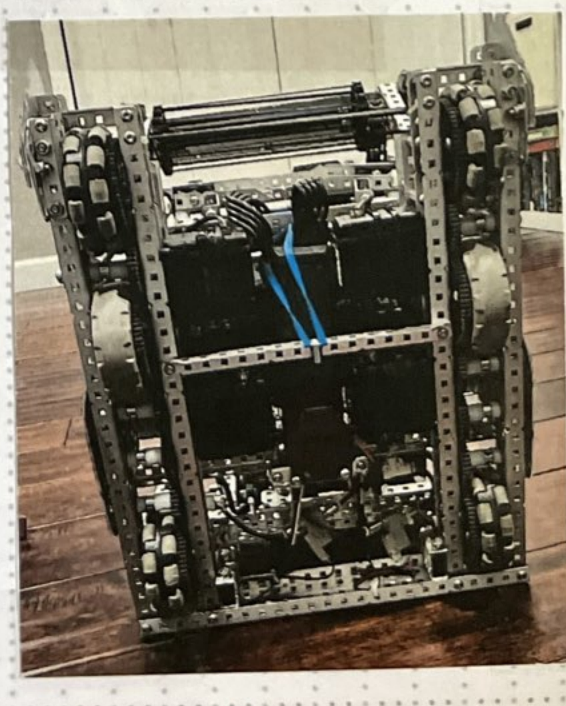
(continued)

Below are images of the complete battery, brain, and license plates.

VP







Project Attaching Brain and Battery
Name Vanessa Perkins
Date 1/22/2024

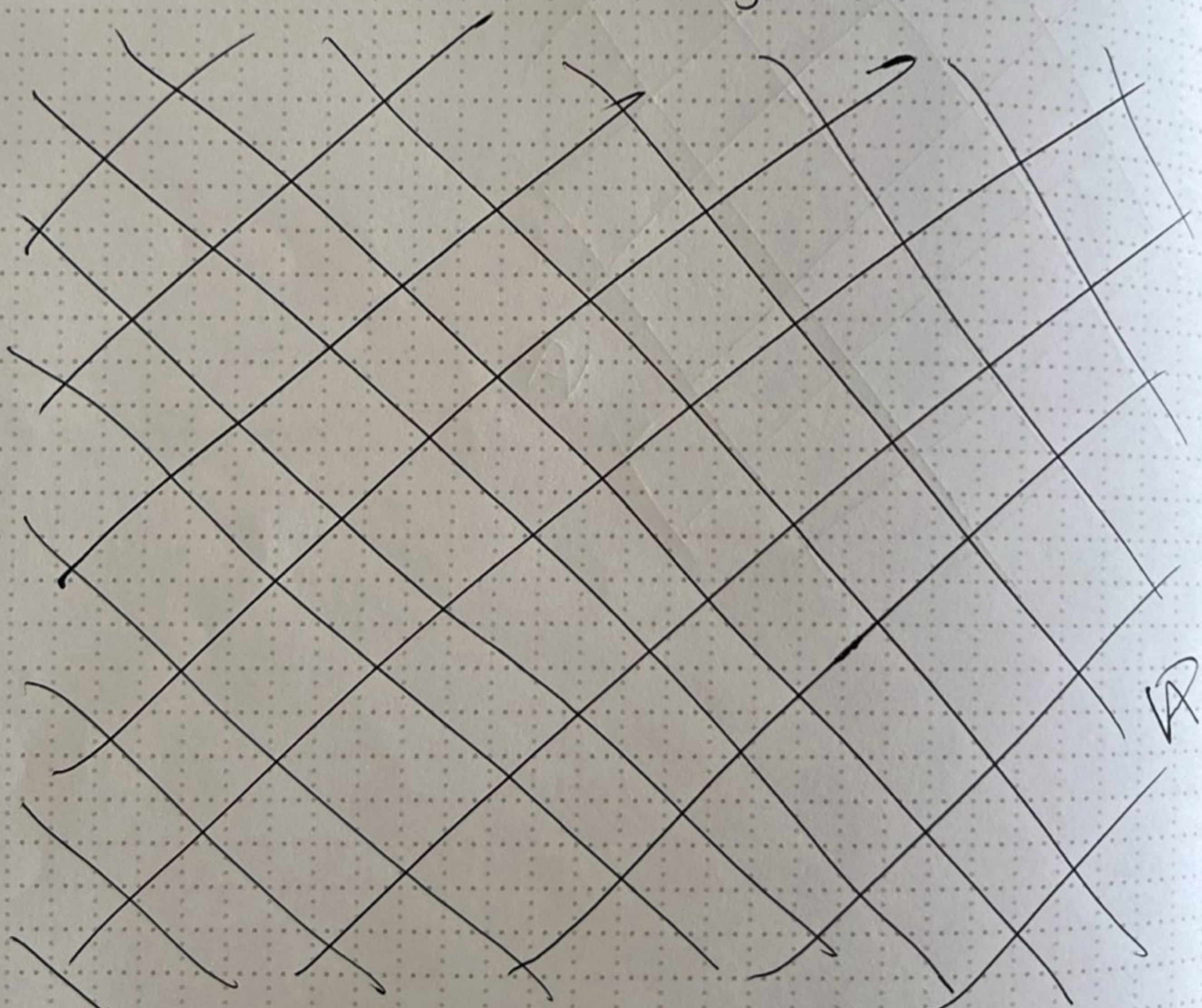
Finishing Robot:

Goals for today → Finish up the robot with wiring and programming

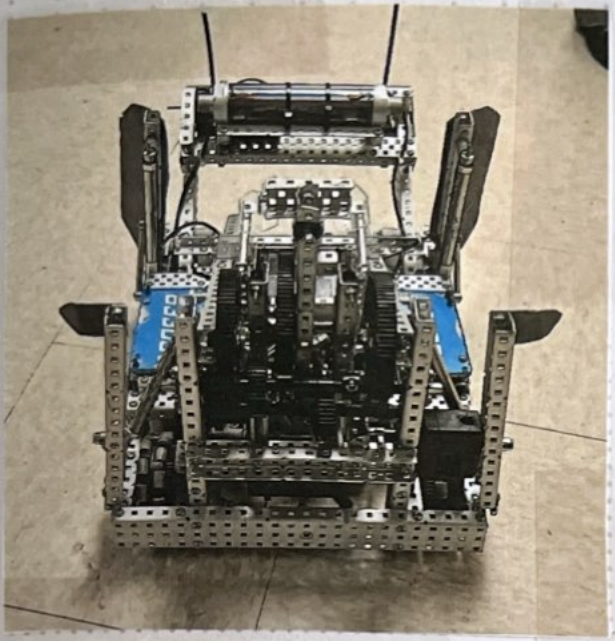
Today Harris and Owen wired the robot while Eli hooked up all the pneumatic tubing and air reserviors to the pistons.

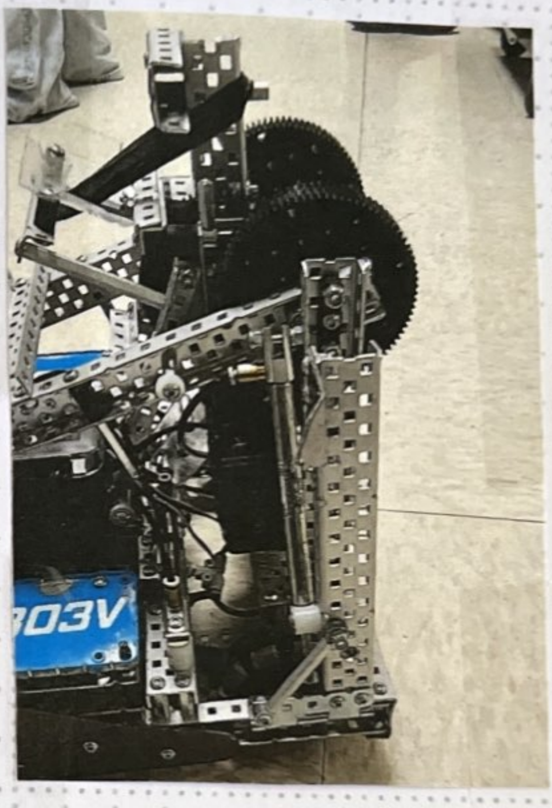
Then Harris programmed the motors and controls to what Owen wanted.

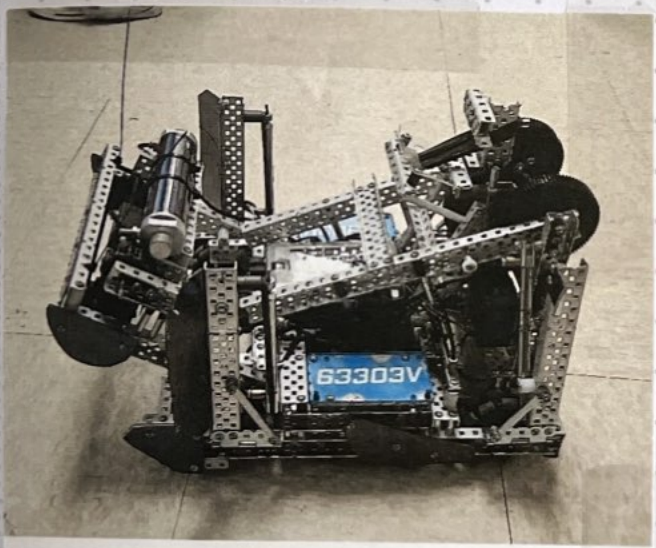
Then Harris began mapping auton runs.

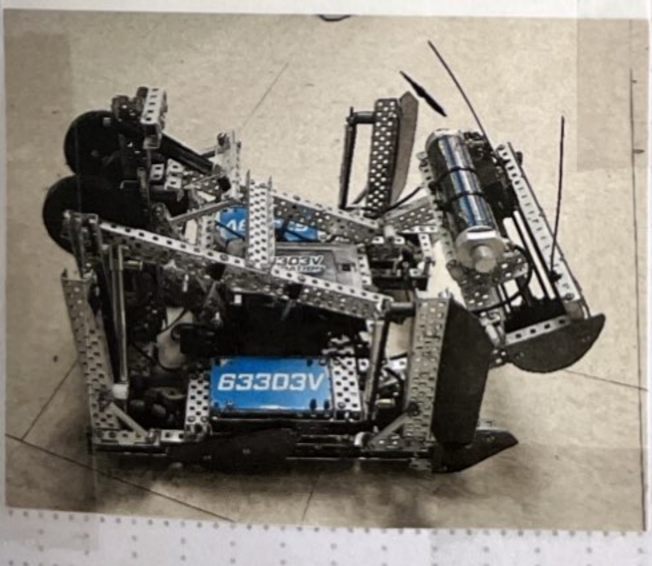


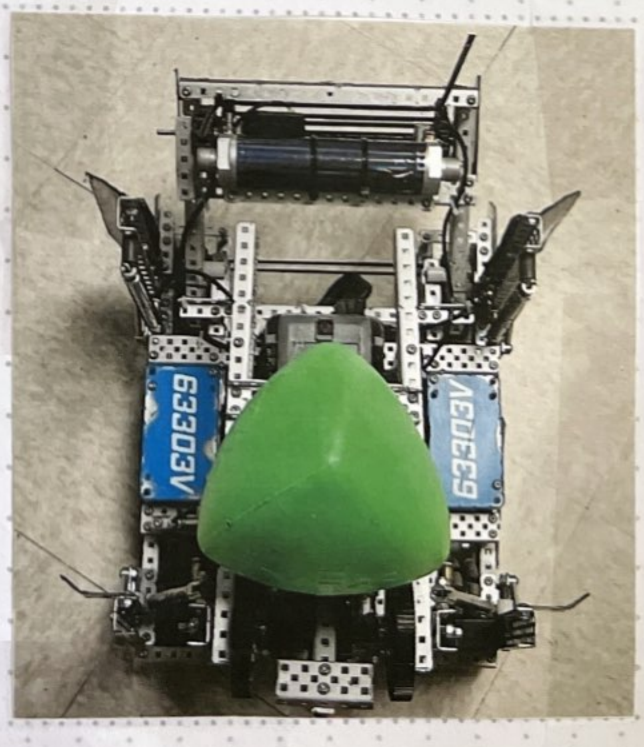
Finished Robot Images:



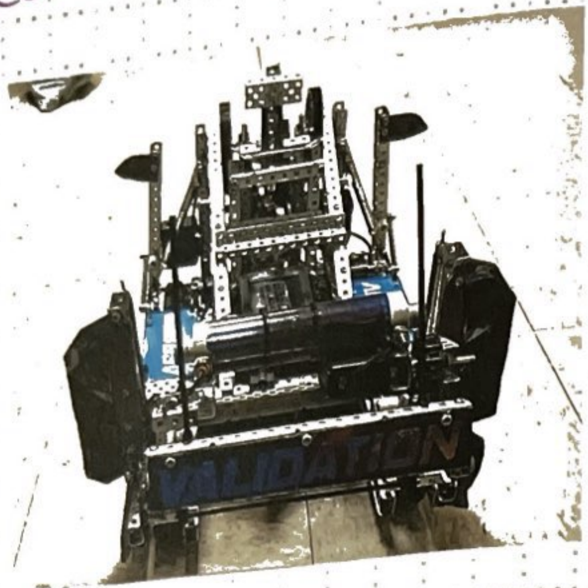


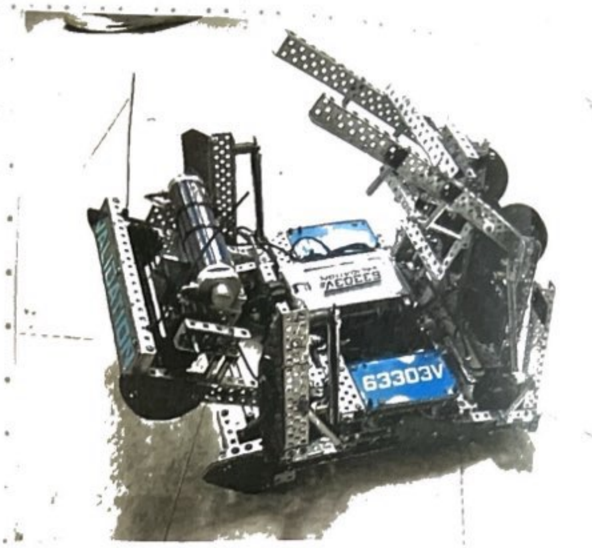


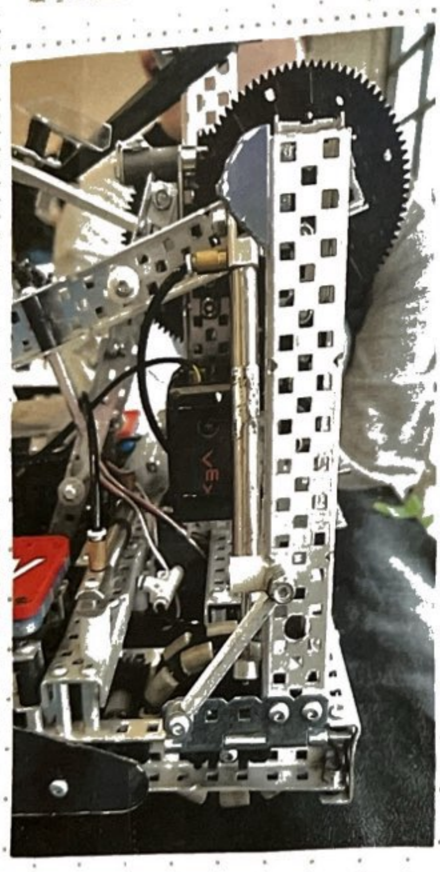


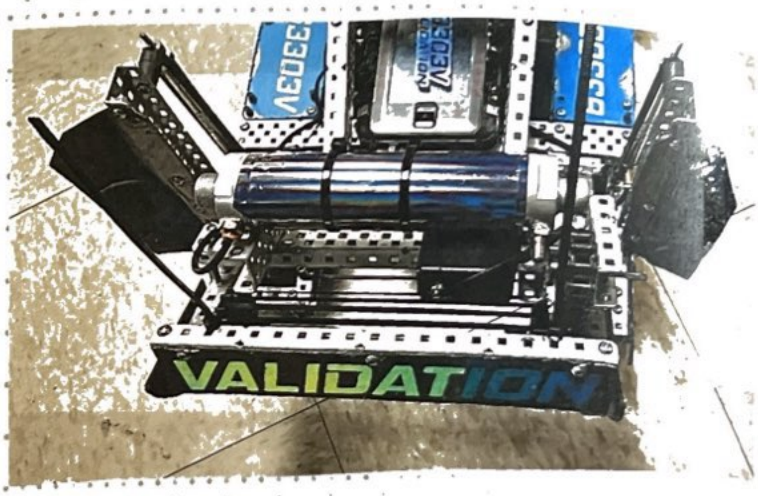


(Continued)









LP

Working on Skills Programming:

Goals for today → practice driver skills strategies

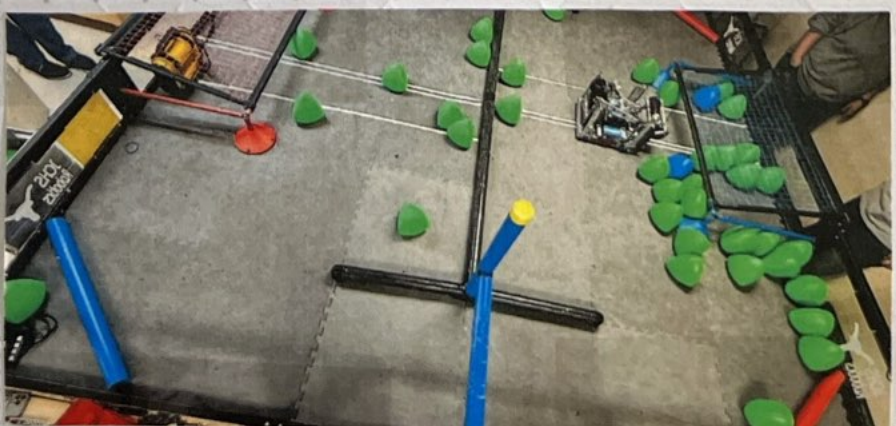
Today we tested different driver skills strategies such as plowing to the left, right, or center first. Below are the results as well as a picture of one of our test runs.

Plow Left: This allows him to better push stray triballs that didn't matchload correctly.

Plow Right: This gives Owen more time at the goal but then he misses triballs on the close side.

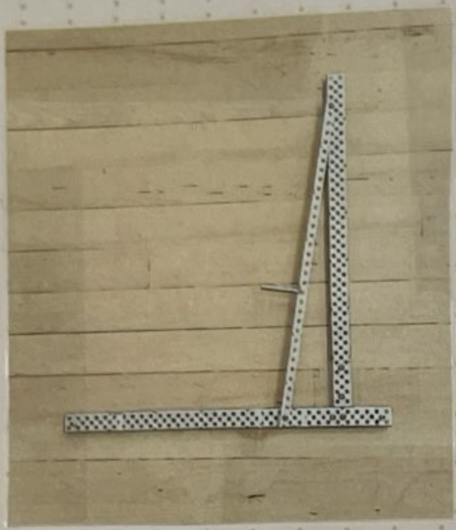
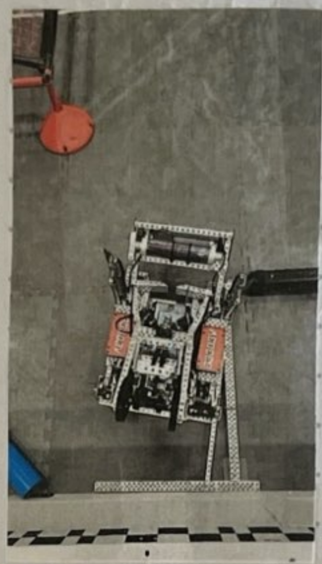
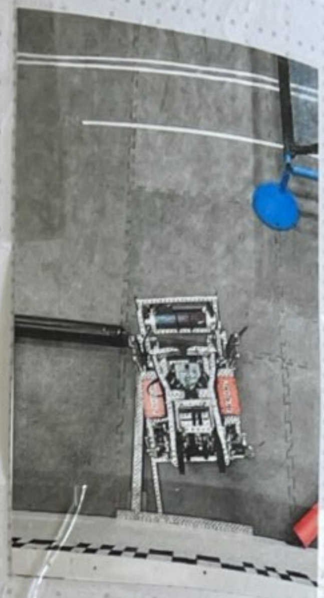
Center Plow: This is a time consuming strategy because it is hard for the robot to drive over the barrier while trying to push triballs as well.

Overall plowing to the left side of the field is the most effective strategy.



Autonomous Aligner:

Goals for today → build an aligner that can make our programs more consistent

To make our autonomous runs more consistent, we built this aligner out of spare c-channels.
The center image shows the alignment position for the close side auton.
The right image shows the alignment position for the far side auton.
This aligner has proved very effective.

VP

Project Autonomous Aligner
Name Vanessa Perkins
Date 1/20/2024

Close Side Auton

```
#include "vex.h"

// ---- START VEXCODE CONFIGURED DEVICES ----
// Robot Configuration:
// [Name]               [Type]          [Port(s)]
// Controller1          controller
// Kicker               motor           10
// Intake               motor           9
// DriveFrontLeft       motor           3
// DriveCenterLeft      motor           1
// DriveBackLeft        motor           4
// DriveFrontRight      motor           17
// DriveCenterRight     motor           18
// DriveBackRight       motor           11
// FrontWings           digital_out     A
// BackWingLeft         digital_out     F
// BackWingRight        digital_out     C
// PTO                  digital_out     E
// Inertial             inertial        16
// Kicker2              motor           12
// ---- END VEXCODE CONFIGURED DEVICES ----

using namespace vex;
competition Competition;

/*---------------------------------------------------------------------------*/
/*                              VEXcode Config                               */
/*                                                                           */
/*  Before you do anything else, start by configuring your motors and        */
/*  sensors using the V5 port icon in the top right of the screen. Doing     */
/*  so will update robot-config.cpp and robot-config.h automatically, so     */
/*  you don't have to. Ensure that your motors are reversed properly. For    */
/*  the drive, spinning all motors forward should drive the robot forward.   */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/*                              JAR-Template Config                          */
/*                                                                           */
/*  Where all the magic happens. Follow the instructions below to input      */
/*  all the physical constants and values for your robot. You should         */
/*  already have configured your robot manually with the sidebar configurer. */
/*---------------------------------------------------------------------------*/

Drive chassis(

//Specify your drive setup below. There are seven options:
//ZERO_TRACKER_NO_ODOM, ZERO_TRACKER_ODOM, TANK_ONE_ENCODER, TANK_ONE_ROTATION,
TANK_TWO_ENCODER, TANK_TWO_ROTATION, HOLONOMIC_TWO_ENCODER, and HOLONOMIC_TWO_ROTATION
//For example, if you are not using odometry, put ZERO_TRACKER_NO_ODOM below:
ZERO_TRACKER_NO_ODOM,

//Add the names of your Drive motors into the motor groups below, separated by commas,
i.e. motor_group(Motor1,Motor2,Motor3).
//You will input whatever motor names you chose when you configured your robot using
the sidebar configurer, they don't have to be "Motor1" and "Motor2".

//Left Motors:
motor_group(DriveFrontLeft,DriveCenterLeft,DriveBackLeft),

//Right Motors:
motor_group(DriveFrontRight,DriveCenterRight,DriveBackRight),
```

```
//Specify the PORT NUMBER of your inertial sensor, in PORT format (i.e. "PORT1", not
simply "1"):
PORT16,

//Input your wheel diameter. (4" omnis are actually closer to 4.125"):
3.25,

//External ratio, must be in decimal, in the format of input teeth/output teeth.
//If your motor has an 84-tooth gear and your wheel has a 60-tooth gear, this value
will be 1.4.
//If the motor drives the wheel directly, this value is 1:
0.6,

//Gyro scale, this is what your gyro reads when you spin the robot 360 degrees.
//For most cases 360 will do fine here, but this scale factor can be very helpful when
precision is necessary.
360,

/*---------------------------------------------------------------------------*/
/*                              PAUSE!                                       */
/*                                                                           */
/*  The rest of the drive constructor is for robots using POSITION TRACKING. */
/*  If you are not using position tracking, leave the rest of the values as  */
/*  they are.                                                                */
/*---------------------------------------------------------------------------*/

//If you are using ZERO_TRACKER_ODOM, you ONLY need to adjust the FORWARD TRACKER
CENTER DISTANCE.

//FOR HOLONOMIC DRIVES ONLY: Input your drive motors by position. This is only
necessary for holonomic drives, otherwise this section can be left alone.
//LF:      //RF:
PORT1,     -PORT2,

//LB:      //RB:
PORT3,     -PORT4,

//If you are using position tracking, this is the Forward Tracker port (the tracker
which runs parallel to the direction of the chassis).
//If this is a rotation sensor, enter it in "PORT1" format, inputting the port below.
//If this is an encoder, enter the port as an integer. Triport A will be a "1",
Triport B will be a "2", etc.
3,

//Input the Forward Tracker diameter (reverse it to make the direction switch):
2.75,

//Input Forward Tracker center distance (a positive distance corresponds to a tracker
on the right side of the robot, negative is left.)
//For a zero tracker tank drive with odom, put the positive distance from the center
of the robot to the right side of the drive.
//This distance is in inches:
-2,

//Input the Sideways Tracker Port, following the same steps as the Forward Tracker
Port:
1,

//Sideways tracker diameter (reverse to make the direction switch):
-2.75,
```

Project Close Side Auton Program (Contin.) Name Harris Perkins
Date 1/26/2024

```
//Sideways tracker center distance (positive distance is behind the center of the
robot, negative is in front):
5.5

);

int current_auton_selection = 0;
bool auto_started = false;

void pre_auton(void) {
 // Initializing Robot Configuration. DO NOT REMOVE!
 vexcodeInit();
 default_constants();

 }

void autonomous(void) {
 auto_started = true;
  FrontWings.set(true);
 wait(0.2,sec);
 FrontWings.set(false);
 Intake.spin(reverse,100,pct);
 wait(0.75,sec);
 Intake.setStopping(hold);
 Intake.stop();
 chassis.drive_distance(48);
 Intake.spin(forward,100,pct);
 wait(0.5,sec);
 chassis.turn_to_angle(10);
 chassis.drive_distance(-36);
 chassis.right_swing_to_angle(170);
 chassis.drive_distance(13);
 chassis.turn_to_angle(90);
 BackWingRight.set(true);
 wait(0.5,sec);
 chassis.turn_to_angle(87);
 BackWingRight.set(false);
 chassis.drive_distance(20);
 Intake.spin(reverse,100,pct);
 chassis.drive_distance(22);


}

/*---------------------------------------------------------------------------*/
/*                                                                           */
/*                              User Control Task                            */
/*                                                                           */
/*  This task is used to control your robot during the user control phase of */
/*  a VEX Competition.                                                       */
/*                                                                           */
/*  You must modify the code to add your own robot specific commands here.   */
/*---------------------------------------------------------------------------*/

void usercontrol(void) {
 // User control code here, inside the loop
 while (1) {

   DriveFrontLeft.spin(forward, Controller1.Axis3.position(percent) +
   Controller1.Axis1.position(percent), percent); // The code for driving with the
Front Left motor

   DriveBackLeft.spin(forward, Controller1.Axis3.position(percent) +
```

```
Controller1.Axis1.position(percent), percent); // Left Back Drive Code

DriveCenterLeft.spin(forward, Controller1.Axis3.position(percent) +
Controller1.Axis1.position(percent), percent); // Left Center Drive Code
 DriveFrontRight.spin(forward, Controller1.Axis3.position(percent) -
Controller1.Axis1.position(percent), percent); // Right Front Drive Code

DriveCenterRight.spin(forward, Controller1.Axis3.position(percent) -
Controller1.Axis1.position(percent), percent); // Right Center Drive Code

DriveBackRight.spin(forward, Controller1.Axis3.position(percent) -
Controller1.Axis1.position(percent), percent); // Right Back Drive Code



    if(Controller1.ButtonR1.pressing())
  { Intake.setVelocity(100,percent);
    Intake.spin(vex::directionType::fwd, 100, velocityUnits::pct);
    Intake.setStopping(hold);

    }

else if(Controller1.ButtonR2.pressing())
   { Intake.setVelocity(-100,percent);
     Intake.spin(vex::directionType::fwd, -100, velocityUnits::pct);
     Intake.setStopping(hold);
     }



else if(Controller1.ButtonX.pressing())
   {FrontWings.set(true);
   }

else if(Controller1.ButtonA.pressing())
   {FrontWings.set(false);
   }

else if(Controller1.ButtonDown.pressing())
   {BackWingLeft.set(false);
   }

else if(Controller1.ButtonLeft.pressing())
   {BackWingLeft.set(true);
   }

else if(Controller1.ButtonRight.pressing())
   {BackWingRight.set(false);
   }

else if(Controller1.ButtonUp.pressing())
   {BackWingRight.set(true);
   }

else if(Controller1.ButtonY.pressing())
   {PTO.set(true);
   Kicker.spin(vex::directionType::fwd,10, velocityUnits::pct);
    Kicker.setStopping(hold);
   }

else if(Controller1.ButtonL1.pressing())
   {Kicker.spin(vex::directionType::fwd, 100, velocityUnits::pct);
```

```
    Kicker.setStopping(hold);
    Kicker2.spin(vex::directionType::fwd, 100, velocityUnits::pct);
    Kicker2.setStopping(hold);
  }

else if(Controller1.ButtonL2.pressing())
  {Kicker.spin(vex::directionType::fwd, -100, velocityUnits::pct);
    Kicker.setStopping(hold);
    Kicker2.spin(vex::directionType::fwd, -100, velocityUnits::pct);
    Kicker2.setStopping(hold);
  }

      else

  {
    Intake.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);

    Kicker.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);

    Kicker2.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);

    }

    // This is the main execution loop for the user control program.
    // Each time through the loop your program should update motor + servo
    // values based on feedback from the joysticks.

    // ........................................................................
    // Insert user code here. This is where you use the joystick values to
    // update your motors, etc.
    // ........................................................................

    //Replace this line with chassis.control_tank(); for tank drive
    //or chassis.control_holonomic(); for holo drive.


    wait(20, msec); // Sleep the task for a short amount of time to
                    // prevent wasted resources.
  }
}

//
// Main will set up the competition functions and callbacks.
//
int main() {
  // Set up callbacks for autonomous and driver control periods.
  Competition.autonomous(autonomous);
  Competition.drivercontrol(usercontrol);

  // Run the pre-autonomous function.
  pre_auton();

  // Prevent main from exiting with an infinite loop.
  while (true) {
    wait(100, msec);
  }
```

Project Close Side Auton Program (Continued) Name Harris Perkins

Date 1/26/2024

Far Side Auton

```
#include "vex.h"

// ---- START VEXCODE CONFIGURED DEVICES ----
// Robot Configuration:
// [Name]               [Type]          [Port(s)]
// Controller1          controller
// Kicker               motor           10
// Intake               motor           9
// DriveFrontLeft       motor           3
// DriveCenterLeft      motor           1
// DriveBackLeft        motor           4
// DriveFrontRight      motor           17
// DriveCenterRight     motor           18
// DriveBackRight       motor           11
// FrontWings           digital_out     A
// BackWingLeft         digital_out     F
// BackWingRight        digital_out     C
// PTO                  digital_out     E
// Inertial             inertial        16
// Kicker2              motor           12
// ---- END VEXCODE CONFIGURED DEVICES ----

using namespace vex;
competition Competition;

/*---------------------------------------------------------------------------*/
/*                             VEXcode Config                                */
/*                                                                           */
/*  Before you do anything else, start by configuring your motors and        */
/*  sensors using the V5 port icon in the top right of the screen. Doing     */
/*  so will update robot-config.cpp and robot-config.h automatically, so     */
/*  you don't have to. Ensure that your motors are reversed properly. For    */
/*  the drive, spinning all motors forward should drive the robot forward.   */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/*                             JAR-Template Config                           */
/*                                                                           */
/*  Where all the magic happens. Follow the instructions below to input      */
/*  all the physical constants and values for your robot. You should         */
/*  already have configured your robot manually with the sidebar configurer. */
/*---------------------------------------------------------------------------*/

Drive chassis(

//Specify your drive setup below. There are seven options:
//ZERO_TRACKER_NO_ODOM, ZERO_TRACKER_ODOM, TANK_ONE_ENCODER, TANK_ONE_ROTATION,
TANK_TWO_ENCODER, TANK_TWO_ROTATION, HOLONOMIC_TWO_ENCODER, and HOLONOMIC_TWO_ROTATION
//For example, if you are not using odometry, put ZERO_TRACKER_NO_ODOM below:
ZERO_TRACKER_NO_ODOM,

//Add the names of your Drive motors into the motor groups below, separated by commas,
i.e. motor_group(Motor1,Motor2,Motor3).
//You will input whatever motor names you chose when you configured your robot using
the sidebar configurer, they don't have to be "Motor1" and "Motor2".

//Left Motors:
motor_group(DriveFrontLeft,DriveCenterLeft,DriveBackLeft),

//Right Motors:
motor_group(DriveFrontRight,DriveCenterRight,DriveBackRight),
```

Project Far Side Auton Program

Date 1/26/2024

Name Harris Perkins

```
//Specify the PORT NUMBER of your inertial sensor, in PORT format (i.e. "PORT1", not
simply "1"):
PORT16,

//Input your wheel diameter. (4" omnis are actually closer to 4.125"):
3.25,

//External ratio, must be in decimal, in the format of input teeth/output teeth.
//If your motor has an 84-tooth gear and your wheel has a 60-tooth gear, this value
will be 1.4.
//If the motor drives the wheel directly, this value is 1:
0.6,

//Gyro scale, this is what your gyro reads when you spin the robot 360 degrees.
//For most cases 360 will do fine here, but this scale factor can be very helpful when
precision is necessary.
360,

/*---------------------------------------------------------------------------*/
/*                              PAUSE!                                        */
/*                                                                            */
/*  The rest of the drive constructor is for robots using POSITION TRACKING. */
/*  If you are not using position tracking, leave the rest of the values as  */
/*  they are.                                                                 */
/*---------------------------------------------------------------------------*/

//If you are using ZERO_TRACKER_ODOM, you ONLY need to adjust the FORWARD TRACKER
CENTER DISTANCE.

//FOR HOLONOMIC DRIVES ONLY: Input your drive motors by position. This is only
necessary for holonomic drives, otherwise this section can be left alone.
//LF:      //RF:
PORT1,     -PORT2,

//LB:      //RB:
PORT3,     -PORT4,

//If you are using position tracking, this is the Forward Tracker port (the tracker
which runs parallel to the direction of the chassis).
//If this is a rotation sensor, enter it in "PORT1" format, inputting the port below.
//If this is an encoder, enter the port as an integer. Triport A will be a "1",
Triport B will be a "2", etc.
3,

//Input the Forward Tracker diameter (reverse it to make the direction switch):
2.75,

//Input Forward Tracker center distance (a positive distance corresponds to a tracker
on the right side of the robot, negative is left.)
//For a zero tracker tank drive with odom, put the positive distance from the center
of the robot to the right side of the drive.
//This distance is in inches:
-2,

//Input the Sideways Tracker Port, following the same steps as the Forward Tracker
Port:
1,

//Sideways tracker diameter (reverse to make the direction switch):
-2.75,
```

```
//Sideways tracker center distance (positive distance is behind the center of the robot, negative is in front):
5.5

);

int current_auton_selection = 0;
bool auto_started = false;

void pre_auton(void) {
 // Initializing Robot Configuration. DO NOT REMOVE!
 vexcodeInit();
 default_constants();

 }

void autonomous(void) {
 auto_started = true;
  Intake.spin(reverse,100,pct);
 wait(0.75,sec);
 Intake.setStopping(hold);
 Intake.stop();
 BackWingRight.set(true);
 Intake.spin(forward,100,pct);
 chassis.drive_distance(50);
 chassis.turn_to_angle(92);
 Intake.spin(reverse,100,pct);
 BackWingRight.set(false);
 wait(0.5,sec);
 chassis.turn_to_angle(240);
 Intake.spin(forward,100,pct);
 chassis.drive_distance(25);
 chassis.drive_distance(-24);
 chassis.turn_to_angle(95);
 Intake.spin(reverse,100,pct);
 chassis.drive_distance(-10);
 chassis.turn_to_angle(300);
 Intake.spin(forward,100,pct);
 chassis.drive_distance(9);
 chassis.turn_to_angle(95);
 FrontWings.set(true);
 Intake.spin(reverse,100,pct);
 chassis.drive_distance(45);
 chassis.drive_distance(-15);


}

/*---------------------------------------------------------------------------*/
/*                                                                           */
/*                              User Control Task                            */
/*                                                                           */
/*  This task is used to control your robot during the user control phase of */
/*  a VEX Competition.                                                       */
/*                                                                           */
/*  You must modify the code to add your own robot specific commands here.   */
/*---------------------------------------------------------------------------*/

void usercontrol(void) {
 // User control code here, inside the loop
 while (1) {

   DriveFrontLeft.spin(forward, Controller1.Axis3.position(percent) +
```

Project Far Side Auton Program (contin.) Name Harris Perkins

Date 1/26/2024

```
Controller1.Axis1.position(percent), percent); // The code for driving with the Front Left motor

DriveBackLeft.spin(forward, Controller1.Axis3.position(percent) +
Controller1.Axis1.position(percent), percent); // Left Back Drive Code

DriveCenterLeft.spin(forward, Controller1.Axis3.position(percent) +
Controller1.Axis1.position(percent), percent); // Left Center Drive Code
DriveFrontRight.spin(forward, Controller1.Axis3.position(percent) -
Controller1.Axis1.position(percent), percent); // Right Front Drive Code

DriveCenterRight.spin(forward, Controller1.Axis3.position(percent) -
Controller1.Axis1.position(percent), percent); // Right Center Drive Code

DriveBackRight.spin(forward, Controller1.Axis3.position(percent) -
Controller1.Axis1.position(percent), percent); // Right Back Drive Code



    if(Controller1.ButtonR1.pressing())
  { Intake.setVelocity(100,percent);
    Intake.spin(vex::directionType::fwd, 100, velocityUnits::pct);
    Intake.setStopping(hold);

    }

else if(Controller1.ButtonR2.pressing())
  { Intake.setVelocity(-100,percent);
    Intake.spin(vex::directionType::fwd, -100, velocityUnits::pct);
    Intake.setStopping(hold);
    }



else if(Controller1.ButtonX.pressing())
   {FrontWings.set(true);
   }

else if(Controller1.ButtonA.pressing())
   {FrontWings.set(false);
   }

else if(Controller1.ButtonDown.pressing())
   {BackWingLeft.set(false);
   }

else if(Controller1.ButtonLeft.pressing())
   {BackWingLeft.set(true);
   }

else if(Controller1.ButtonRight.pressing())
   {BackWingRight.set(false);
   }

else if(Controller1.ButtonUp.pressing())
   {BackWingRight.set(true);
   }

else if(Controller1.ButtonY.pressing())
   {PTO.set(true);
   Kicker.spin(vex::directionType::fwd,10, velocityUnits::pct);
    Kicker.setStopping(hold);
```

```
    }

else if(Controller1.ButtonL1.pressing())
   {Kicker.spin(vex::directionType::fwd, 100, velocityUnits::pct);
    Kicker.setStopping(hold);
    Kicker2.spin(vex::directionType::fwd, 100, velocityUnits::pct);
    Kicker2.setStopping(hold);
   }

else if(Controller1.ButtonL2.pressing())
   {Kicker.spin(vex::directionType::fwd, -100, velocityUnits::pct);
    Kicker.setStopping(hold);
    Kicker2.spin(vex::directionType::fwd, -100, velocityUnits::pct);
    Kicker2.setStopping(hold);
   }

       else
   {
     Intake.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);

     Kicker.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);

     Kicker2.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);

     }



    // This is the main execution loop for the user control program.
    // Each time through the loop your program should update motor + servo
    // values based on feedback from the joysticks.

    // ........................................................................
    // Insert user code here. This is where you use the joystick values to
    // update your motors, etc.
    // ........................................................................

    //Replace this line with chassis.control_tank(); for tank drive
    //or chassis.control_holonomic(); for holo drive.

    wait(20, msec); // Sleep the task for a short amount of time to
                    // prevent wasted resources.
  }
}

//
// Main will set up the competition functions and callbacks.
//
int main() {
  // Set up callbacks for autonomous and driver control periods.
  Competition.autonomous(autonomous);
  Competition.drivercontrol(usercontrol);

  // Run the pre-autonomous function.
  pre_auton();

  // Prevent main from exiting with an infinite loop.
  while (true) {
    wait(100, msec);
  }
}
```

Project Far Side Auton Program (continued) Name ~~Varre~~ Harris Perkins
Date 1/26/2024

Skills Program

```
#include "vex.h"

// ---- START VEXCODE CONFIGURED DEVICES ----
// Robot Configuration:
// [Name]               [Type]          [Port(s)]
// Controller1          controller
// Intake               motor           9
// DriveFrontLeft       motor           3
// DriveCenterLeft      motor           1
// DriveBackLeft        motor           4
// DriveFrontRight      motor           17
// DriveCenterRight     motor           18
// DriveBackRight       motor           11
// FrontWings           digital_out     A
// BackWingLeft         digital_out     F
// BackWingRight        digital_out     C
// PTO                  digital_out     E
// Inertial             inertial        16
// Kicker               motor           10
// Kicker2              motor           12
// ---- END VEXCODE CONFIGURED DEVICES ----

using namespace vex;
competition Competition;

/*---------------------------------------------------------------------------*/
/*                             VEXcode Config                                */
/*                                                                           */
/*  Before you do anything else, start by configuring your motors and        */
/*  sensors using the V5 port icon in the top right of the screen. Doing     */
/*  so will update robot-config.cpp and robot-config.h automatically, so     */
/*  you don't have to. Ensure that your motors are reversed properly. For    */
/*  the drive, spinning all motors forward should drive the robot forward.   */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/*                             JAR-Template Config                           */
/*                                                                           */
/*  Where all the magic happens. Follow the instructions below to input      */
/*  all the physical constants and values for your robot. You should         */
/*  already have configured your robot manually with the sidebar configurer. */
/*---------------------------------------------------------------------------*/

Drive chassis(

//Specify your drive setup below. There are seven options:
//ZERO_TRACKER_NO_ODOM, ZERO_TRACKER_ODOM, TANK_ONE_ENCODER, TANK_ONE_ROTATION,
TANK_TWO_ENCODER, TANK_TWO_ROTATION, HOLONOMIC_TWO_ENCODER, and HOLONOMIC_TWO_ROTATION
//For example, if you are not using odometry, put ZERO_TRACKER_NO_ODOM below:
ZERO_TRACKER_NO_ODOM,

//Add the names of your Drive motors into the motor groups below, separated by commas,
i.e. motor_group(Motor1,Motor2,Motor3).
//You will input whatever motor names you chose when you configured your robot using
the sidebar configurer, they don't have to be "Motor1" and "Motor2".

//Left Motors:
motor_group(DriveFrontLeft,DriveCenterLeft,DriveBackLeft),

//Right Motors:
motor_group(DriveFrontRight,DriveCenterRight,DriveBackRight),
```

```
//Specify the PORT NUMBER of your inertial sensor, in PORT format (i.e. "PORT1", not simply "1"):
PORT16,

//Input your wheel diameter. (4" omnis are actually closer to 4.125"):
3.25,

//External ratio, must be in decimal, in the format of input teeth/output teeth.
//If your motor has an 84-tooth gear and your wheel has a 60-tooth gear, this value will be 1.4.
//If the motor drives the wheel directly, this value is 1:
0.6,

//Gyro scale, this is what your gyro reads when you spin the robot 360 degrees.
//For most cases 360 will do fine here, but this scale factor can be very helpful when precision is necessary.
360,

/*---------------------------------------------------------------------------*/
/*                               PAUSE!                                      */
/*                                                                           */
/*  The rest of the drive constructor is for robots using POSITION TRACKING. */
/*  If you are not using position tracking, leave the rest of the values as  */
/*  they are.                                                                */
/*---------------------------------------------------------------------------*/

//If you are using ZERO_TRACKER_ODOM, you ONLY need to adjust the FORWARD TRACKER CENTER DISTANCE.

//FOR HOLONOMIC DRIVES ONLY: Input your drive motors by position. This is only necessary for holonomic drives, otherwise this section can be left alone.
//LF:      //RF:
PORT1,     -PORT2,

//LB:      //RB:
PORT3,     -PORT4,

//If you are using position tracking, this is the Forward Tracker port (the tracker which runs parallel to the direction of the chassis).
//If this is a rotation sensor, enter it in "PORT1" format, inputting the port below.
//If this is an encoder, enter the port as an integer. Triport A will be a "1", Triport B will be a "2", etc.
3,

//Input the Forward Tracker diameter (reverse it to make the direction switch):
2.75,

//Input Forward Tracker center distance (a positive distance corresponds to a tracker on the right side of the robot, negative is left.)
//For a zero tracker tank drive with odom, put the positive distance from the center of the robot to the right side of the drive.
//This distance is in inches:
-2,

//Input the Sideways Tracker Port, following the same steps as the Forward Tracker Port:
1,

//Sideways tracker diameter (reverse to make the direction switch):
-2.75,
```

```
//Sideways tracker center distance (positive distance is behind the center of the
robot, negative is in front):
5.5

);

int current_auton_selection = 0;
bool auto_started = false;

void pre_auton(void) {
 // Initializing Robot Configuration. DO NOT REMOVE!
 vexcodeInit();
 default_constants();


 }

void autonomous(void) {
 auto_started = true;
  chassis.drive_distance(10);
 chassis.left_swing_to_angle(69);
 Intake.spin(reverse,100,pct);
 chassis.drive_distance(-19);
 Intake.stop();
 BackWingRight.set(true);
 Kicker.spin(fwd,100,pct);
 Kicker2.spin(reverse,100,pct);
 wait(25,seconds);
 BackWingRight.set(false);
 Kicker.setStopping(hold);
 Kicker2.setStopping(hold);
 Kicker.stop();
 Kicker2.stop();
 chassis.drive_distance(5);
 chassis.turn_to_angle(-45);
 chassis.drive_distance(-18);
 chassis.turn_to_angle(90);
 Intake.spin(reverse,100,pct);
 chassis.drive_distance(73);
 chassis.turn_to_angle(45);
 FrontWings.set(true);
 chassis.drive_distance(25);
 FrontWings.set(false);
 chassis.drive_distance(10);
 chassis.turn_to_angle(0);
 chassis.drive_distance(8);
 chassis.drive_distance(-10);
 chassis.turn_to_angle(-80);
 chassis.drive_distance(35);
 FrontWings.set(true);
 chassis.left_swing_to_angle(69);
 chassis.drive_distance(15);
 chassis.turn_to_angle(90);
 chassis.drive_distance(5);
 FrontWings.set(false);
 chassis.drive_distance(-15);
 chassis.turn_to_angle(170);
 chassis.drive_distance(-20);
 chassis.turn_to_angle(90);
 FrontWings.set(true);
 chassis.drive_distance(25);
 chassis.drive_distance(-10);

}
```

```
/*---------------------------------------------------------------------------*/
/*                                                                           */
/*                            User Control Task                              */
/*                                                                           */
/*  This task is used to control your robot during the user control phase of */
/*  a VEX Competition.                                                       */
/*                                                                           */
/*  You must modify the code to add your own robot specific commands here.   */
/*---------------------------------------------------------------------------*/

void usercontrol(void) {
 // User control code here, inside the loop
 while (1) {

   DriveFrontLeft.spin(forward, Controller1.Axis3.position(percent) +
   Controller1.Axis1.position(percent), percent); // The code for driving with the
Front Left motor

   DriveBackLeft.spin(forward, Controller1.Axis3.position(percent) +
   Controller1.Axis1.position(percent), percent); // Left Back Drive Code

   DriveCenterLeft.spin(forward, Controller1.Axis3.position(percent) +
   Controller1.Axis1.position(percent), percent); // Left Center Drive Code
    DriveFrontRight.spin(forward, Controller1.Axis3.position(percent) -
   Controller1.Axis1.position(percent), percent); // Right Front Drive Code

   DriveCenterRight.spin(forward, Controller1.Axis3.position(percent) -
   Controller1.Axis1.position(percent), percent); // Right Center Drive Code

   DriveBackRight.spin(forward, Controller1.Axis3.position(percent) -
   Controller1.Axis1.position(percent), percent); // Right Back Drive Code



     if(Controller1.ButtonR1.pressing())
   { Intake.setVelocity(100,percent);
     Intake.spin(vex::directionType::fwd, 100, velocityUnits::pct);
     Intake.setStopping(hold);

     }

 else if(Controller1.ButtonR2.pressing())
    { Intake.setVelocity(-100,percent);
      Intake.spin(vex::directionType::fwd, -100, velocityUnits::pct);
      Intake.setStopping(hold);
      }



else if(Controller1.ButtonX.pressing())
   {FrontWings.set(true);
   }

else if(Controller1.ButtonA.pressing())
   {FrontWings.set(false);
   }

else if(Controller1.ButtonDown.pressing())
   {BackWingLeft.set(false);
   }
```

```
else if(Controller1.ButtonLeft.pressing())
   {BackWingLeft.set(true);
   }

else if(Controller1.ButtonRight.pressing())
   {BackWingRight.set(false);
   }

else if(Controller1.ButtonUp.pressing())
   {BackWingRight.set(true);
   }

else if(Controller1.ButtonY.pressing())
   {PTO.set(true);
    Kicker.spin(vex::directionType::fwd,10, velocityUnits::pct);
    Kicker.setStopping(hold);
   }

else if(Controller1.ButtonL1.pressing())
   {Kicker.spin(vex::directionType::fwd, 100, velocityUnits::pct);
    Kicker.setStopping(hold);
    Kicker2.spin(vex::directionType::fwd, -100, velocityUnits::pct);
    Kicker2.setStopping(hold);
   }

else if(Controller1.ButtonL2.pressing())
   {Kicker.spin(vex::directionType::fwd, -100, velocityUnits::pct);
    Kicker.setStopping(hold);
    Kicker2.spin(vex::directionType::fwd, 100, velocityUnits::pct);
    Kicker2.setStopping(hold);
   }

      else
   {
    Intake.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);

    Kicker.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);

    Kicker2.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);

    }

    // This is the main execution loop for the user control program.
    // Each time through the loop your program should update motor + servo
    // values based on feedback from the joysticks.

    // ........................................................................
    // Insert user code here. This is where you use the joystick values to
    // update your motors, etc.
    // ........................................................................

    //Replace this line with chassis.control_tank(); for tank drive
    //or chassis.control_holonomic(); for holo drive.

    wait(20, msec); // Sleep the task for a short amount of time to
```

```
                  // prevent wasted resources.
  }
}

//
// Main will set up the competition functions and callbacks.
//
int main() {
  // Set up callbacks for autonomous and driver control periods.
  Competition.autonomous(autonomous);
  Competition.drivercontrol(usercontrol);

  // Run the pre-autonomous function.
  pre_auton();

  // Prevent main from exiting with an infinite loop.
  while (true) {
    wait(100, msec);
  }
}
```

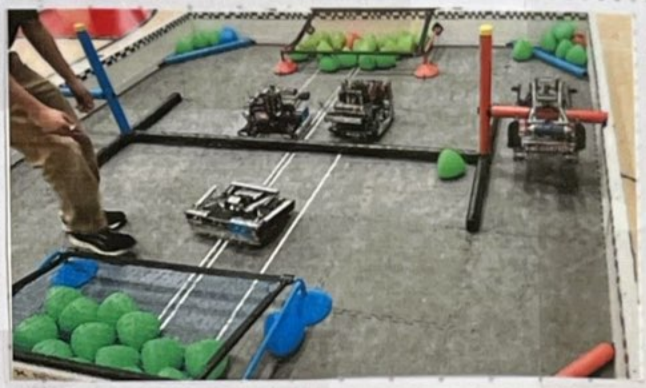
VP

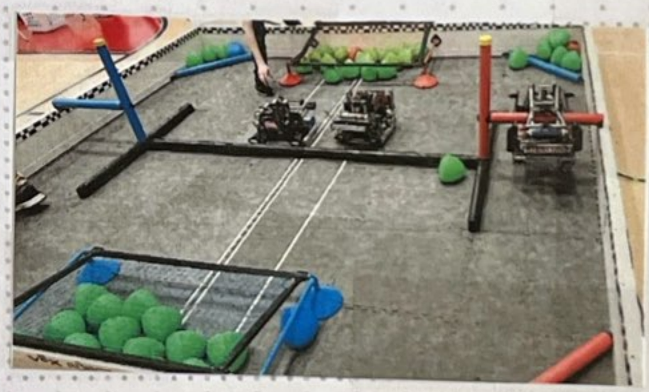
Project Skills Auton Program (continued) Name Harris Perkins

Date 1/26/2024

Competition Analysis:

Goals for today → Review our robot and team performance and ~~this~~ this competition to help plan for the future





This competition was an overall success despite Harris' absence. We won Tournament Champions with team 57711X ROCKY as well as the Judges' Award.

Above are pictures from awards and from our finals match where we were able to elevate.

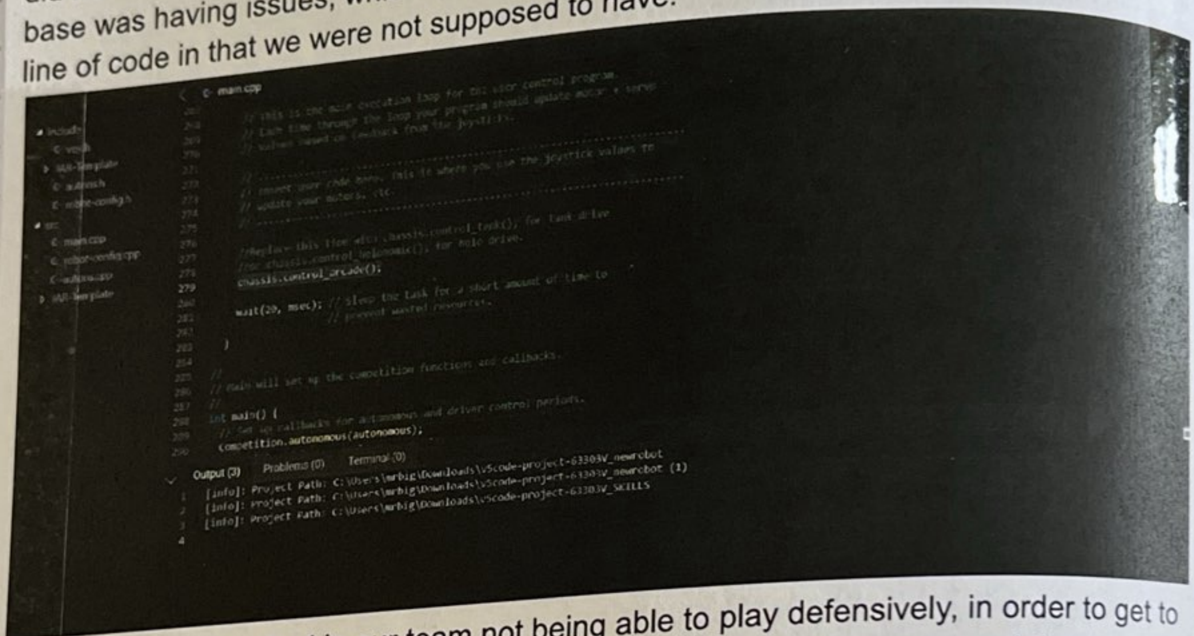
On the next few pages are in depth analyses of each of our matches to help us find areas of improvement in the future.

VP

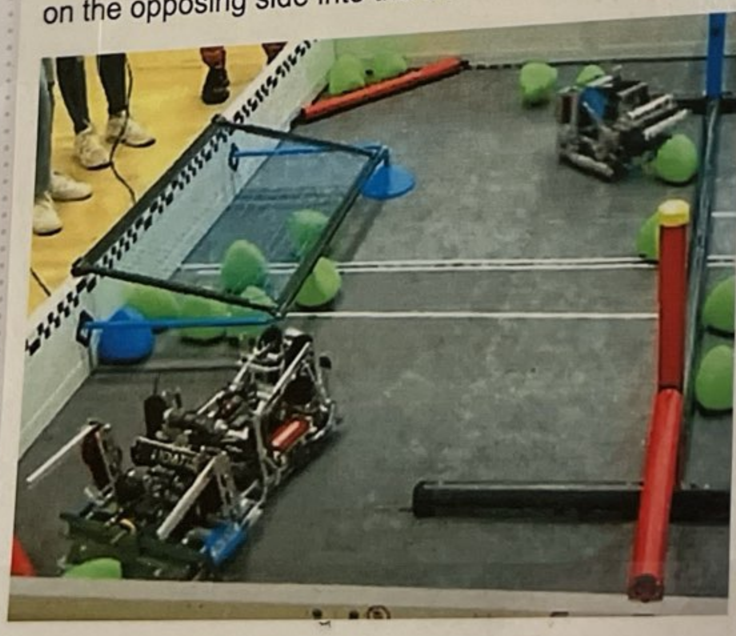
Match Analyses:

Qualifier #3	Jan 27th at 10:14 AM	42652D	**63303V**	71	57711X	42652B	129

In this match, we were allianced with a robot primarily made for hanging, our strategy going in was to bowl, block the other team bowling, and push triballs back over. However, in autonomous we forgot to set up the corner tribal, resulting in the loss of the AWP. We did however score one tribal on the opposing side. When the match started our drive base was having issues, which we later discovered was a minor coding error. Leaving a line of code in that we were not supposed to have:



This minor error resulted in our team not being able to play defensively, in order to get to our match load bar to bowl. However, in this instant, we should have scored the triballs on the opposing side into the other zone before 57711X was able to score them. We wasted nearly 20 seconds trying to get to our bar, when we could have scored points by moving the triballs back over. This is one strategy we plan to perfect, moving triballs back over. Our next mistake was match loading in order to compensate for our loss in points when bowling would have given us more points. This is because our triballs were pushed right back over. Ultimately, our drivebase was the biggest issue, resulting in our loss, along with poor strategy.



Project Match Analyses

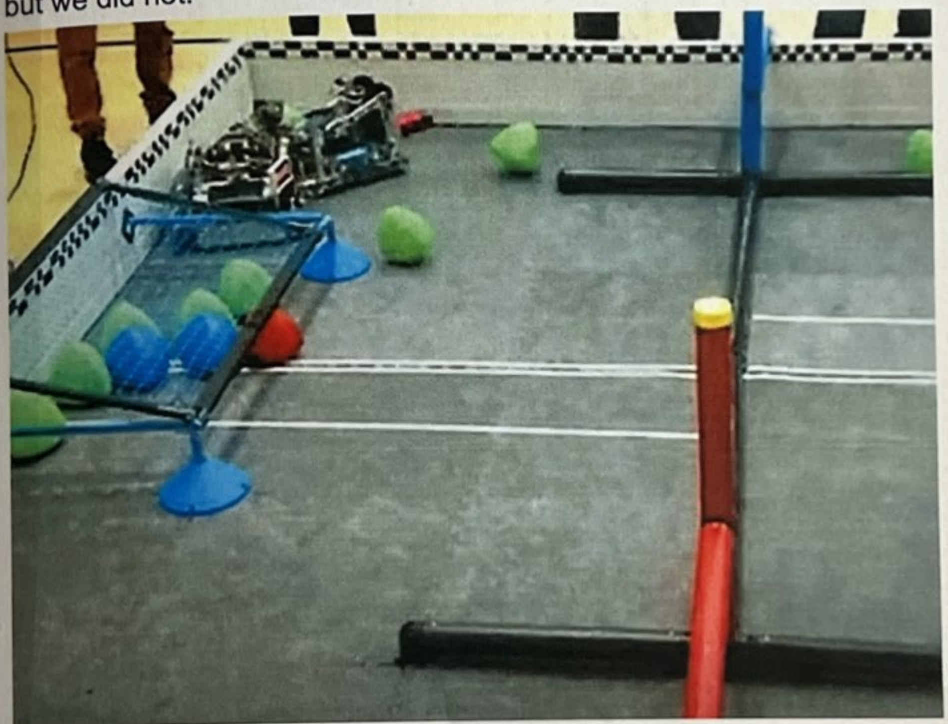
Date 1/29/2024

Name Eli Fritts

(Continued)

Qualifier #7	Jan 27th at 10:36 AM	24816R	16859D	39	**63303V**	24816T	65

In this match we were allianced with a low profile robot meant for descoring, our strategy was to bowl and for the to score as needed while waiting to descore. Our AWP auton worked flawlessly scoring two triballs on the opposing side. We received the full win point in this match. In this match we scored what we had after our autonomous through the alley, however, was not able to bowl as much as we would have liked due to the defense of the opposing team blocking the alley and our loading station. In this scenario, we should have drove to the other station, and we need to practice this versatility in our matches. In the image below, we could easily have scored more through the other alley, but we did not.



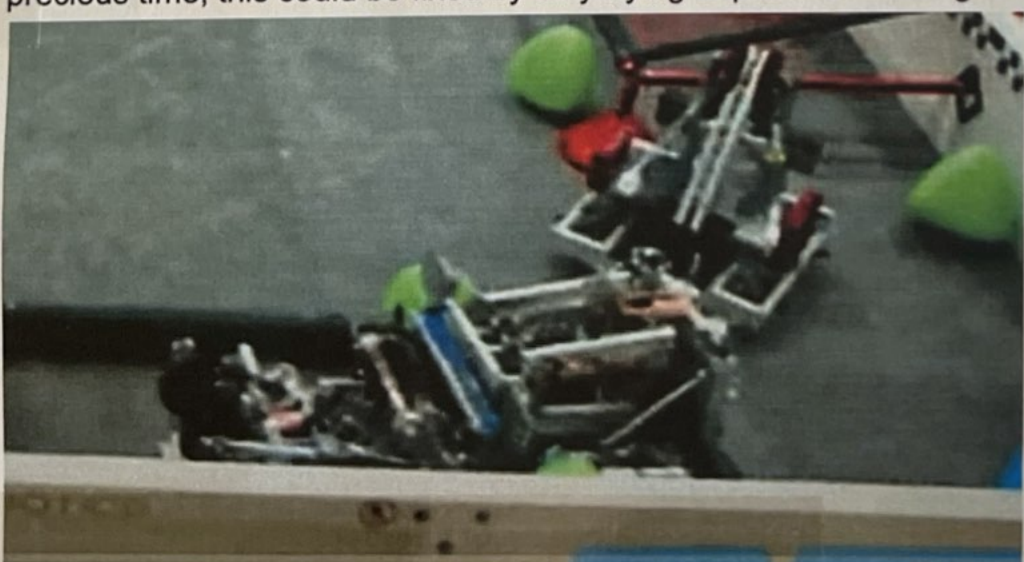
VP

Qualifier #9	Jan 27th at 10:47 AM	42652C	42652A	35	**63303V**	16859A	72

In this match, our autonomous worked amazing, however, we did not get the AWP because when setting up the robot we accidentally knocked over the triball in the zone and did not see it. Our partner was a basic shooter and push bot, so we decided to follow the same strategy and bowl. We scored three triballs in autonomous. In this match we did great, stopping our opposing team from bowling by pushing them in the match load zone, and scoring what was on our side. However, we could have improved by using the other matchloading zone to bowl as we found ourselves stuck in the alley, unsure what to do.

Qualifier #14	Jan 27th at 11:13 AM	**63303V**	42652D	67	24816H	42652A	63

In this match we were paired with a basic push and hang robot. In autonomous, our robot performed perfectly scoring three over, completing the AWP. As we began to score, our alliance partner was in the way, and the opposing team started to matchload. So, we decided to block, unsuccessfully. This wasted time allowed the other teams to catch up and score. Soon after, we began to push triballs back over, evening the playing field, but this was not enough. In order to gain back points, we attempted to go through the alley, but we decided to continue to fight in the alley, when we ultimately wasted precious time, this could be fixed by only trying to push for so long.



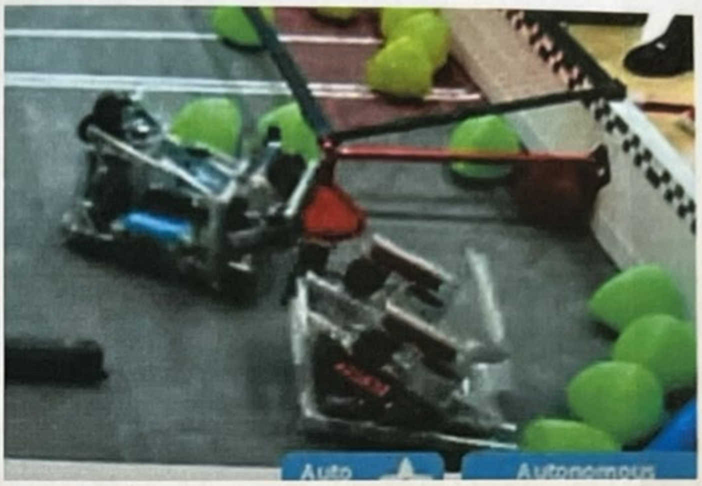
Project Match Analyses (Continued) Name Eli Fritts

Date 1/29/2024

(Continued)

Qualifier #18	Jan 27th at 11:39 AM	63303B	16859A	63	63303A	**63303V**	80

In this match we got a perfect win point, allianced with a push bot. In this match, we began with bowling, and it was very effective, even using the other matchload bar and stopping some bowling. Overall this was a great match, but at the end we were not watching opposing bowling and almost let the opposing alliance score a lot of points.We also got slightly overwhelmed by matchloading, but we need to practice using our wings better.

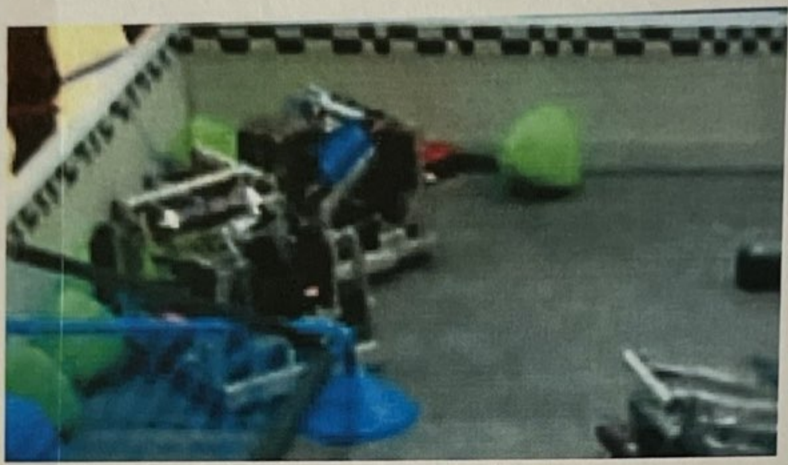


Qualifier #23	Jan 27th at 1:38 PM	**63303V**	63303B	37	24816A	2536A	73

In this match, we decided to have our alliance partner matchload. This was a huge mistake, as it was immediately countered, and it left us without a good plan. In addition, we were pinned in the corner for over five seconds, as well as entangled for one minute of the match, with a poorly built robot that could easily get stuck with other people. Overall, we should have utilized the other alleyway, and stuck to bowling. However, our autonomous worked flawlessly pushing three triballs over and scoring the AWP.

Qualifier #30	Jan 27th at 2:13 PM	24816H	**63303V**	67	63303D	57711X	77

In this match, our win point worked perfectly, pushing three over. We went in with the strategy we always used, bowling. However, we also made sure to counter other bowling, but when doing so, we were pinned for 13 seconds of the match, which was a lot of time we could have used to score. So, we were not able to bowl and score to the best of our ability, making us lose by a short margin.



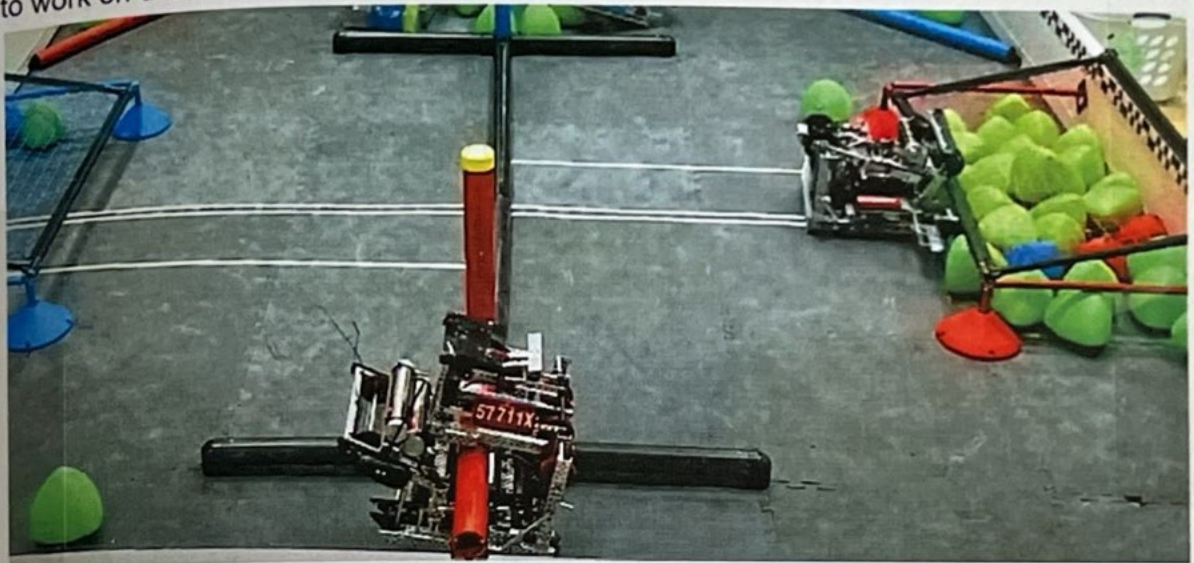
Qualifier #33	Jan 27th at 2:35 PM	42652C	16859D	82	**63303V**	57711C	84

In this match, our autonomous worked great pushing three over, but our zipties slightly missed the hang bar. Starting out, we scored the triballs from autonomous, and went back to bowl, however there was major miscommunication, as we were not sure to bowl or match load. We also got blocked and did not go to the other alley, making conditions worse. Ultimately teaching us the value of communication.

Project Match Analyses (continued) Name Eli Fritts
Date 1/29/2024

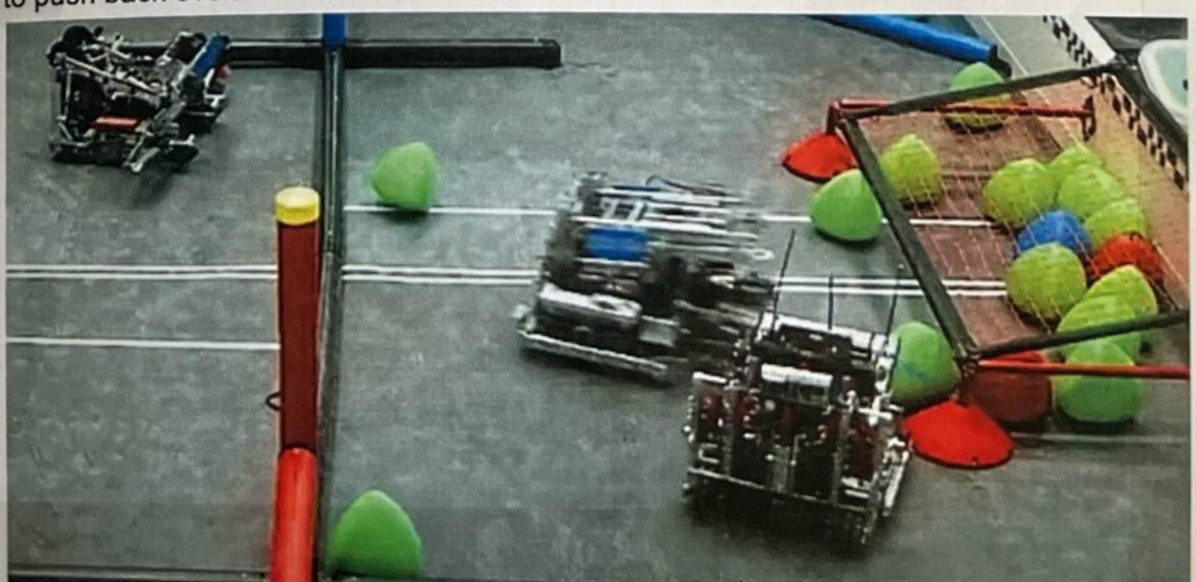
QF #1-1	Jan 27th at 3:28 PM	57711X	**63303V**	151	63303D	16859D	22

In this match, our autonomous worked perfectly scoring all three on the opposing side. We did very good at bowling, however was swarmed by defense, so we know we need to work on defense evasion.



SF #1-1	Jan 27th at 3:52 PM	57711X	**63303V**	114	63303A	24816A	56

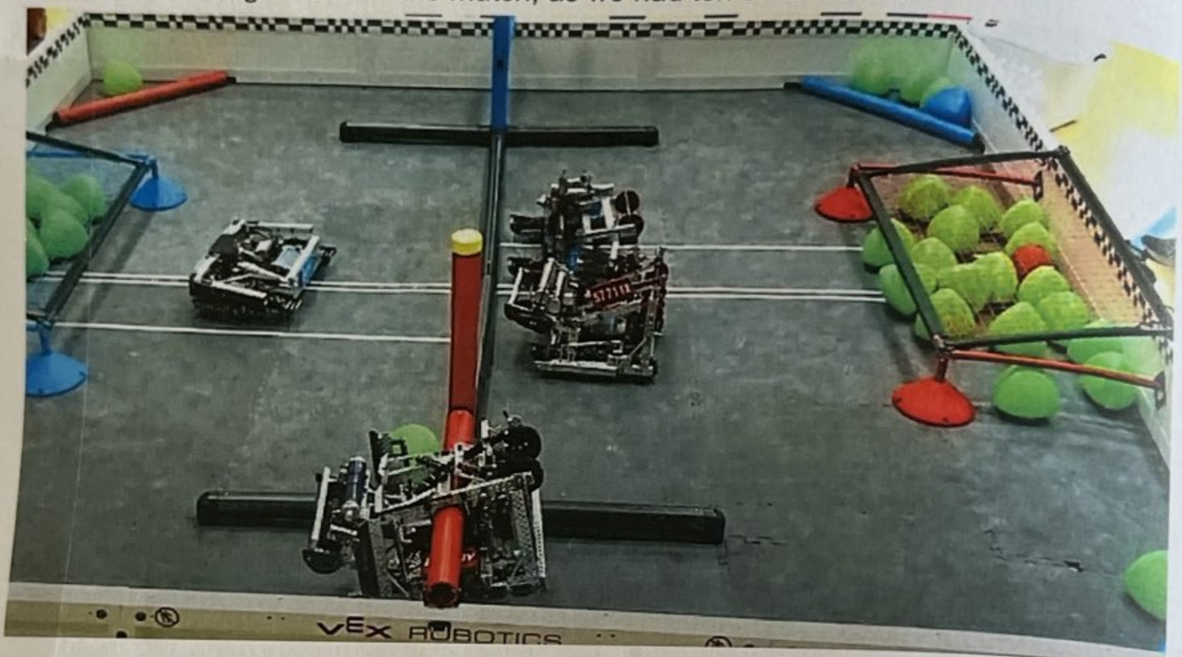
In this match, our autonomous worked perfectly pushing two triballs over. To start the match, we began by pushing back triballs that were being matchloaded, however, we then shifted to scoring as our alliance partner shot some triballs over to score more in a safe manner. Overall, it worked very well, however we need to practice using our wings to push back over.



(Continued)

Final #1-1	Jan 27th at 4:14 PM	57711X	**63303V**	118	24816T	57711C	77

In this match, the field was set up wrong, and we missed one triball, however, it still worked overall. Overall, we bowled and match loaded some, but match loading was our biggest mistake, giving the other team some points. Our hang worked well too, but we should have hanged later in the match, as we had ten seconds left.



Project Match Analyses (Continued) Name Eli Fritts
Date 1/29/2024

Bots at Bristol: 63303V Schedule

1/29/2024: (Eli, Vanessa, Harris)
Location: High School
Time: Regular Schedule

1/30/2024: (Eli, Vanessa, Harris)
Location: Garage
Time: 3:15-6:00 PM

1/31/2024: (Eli, Owen)
Location: Garage
Time: 3:15-5:30 PM

1/01/2024: (Eli, Vanessa, Harris, Owen)
Location: Garage
Time: 3:15-7:00 PM

1/03/2024: (Eli, Vanessa, Harris)
Location: Garage
Time: 1:00-7:00 PM

1/05/2024: (Eli, Vanessa, Harris, Owen)
Location: High School
Time: Regular Schedule

1/06/2024: (Eli, Vanessa, Harris, Owen)
Location: Garage
Time: 3:15-7:00 PM

1/07/2024: (Eli, Vanessa, Harris, Owen)
Location: Garage
Time: 3:15-5:30 PM

1/08/2024: (Eli, Vanessa, Harris, Owen)
Location: Garage
Time: 3:15-8:30 PM

Timeline and Goals:

Goals for today → Plan out our time and goals for the next competition

Timeline

We have two (2) weeks.
Week 1 will be set aside for building updates.
Week 2 will be split evenly for bothing programming and driving.
Our competition date is Feb 9-10.

Goals

Since this is a Signature Event, our goals are specific to helping us try to qualify for WORLDS.

- Have a 400 point skills total.
- Top 3 in Skills
- Excellence Award
- Matchloading in Skills under 30 seconds.
- Practice Interviews
- Win point autonomas
- Unique robot
- Higher hang/Doublehang

VP

Brainstorming Robot Changes:

Goals for today → Come up with ways to change and improve the robot

Below are some ideas we came up with as a team of how to minorly improve the robot that we will try to get to before the next competition.

- Higher hang/elevation.
- 5.5 watt motor to the PTO.
- Vision sensor under Kicker to help with matchloading.
- Get Intakes to hold triballs better.
- Fold out wings instead of fold down.
- Get a win point on our own.

PID/Programming Notes:

Goals for today → Have Harris study programming more, while the robot is being rebuilt, so he can improve his programs

Advanced JAR Notes

- The most common usage of angle values in autons is written as:
chassis.turn_to_angle(90);
- Added overload methods and member variables can be used to maximize and perfect autonomous programming
- To start the overload method, turn_max_voltage, being a number 0-12, can be put behind the set angle for the turn:
chassis.turn_to_angle(90, 3);
- If another value needs to be specified the overload method may be overly complica[illegible] since all variables in between are needed as well
- If turn_timeout needs to be set than also does turn_settle_error and turn_settle_tim[illegible] even if they stay the default values, resulting in something like:
chassis.turn_to_angle(90, 10, 1, 200, 10000);
- If it is too difficult to read or write that several times over then member variables ca[illegible] be more useful
- Member variables eliminate the unnecessary variables between the ones that are actually changing:
chassis.turn_timeout = 10000;
chassis.turn_to_angle(90);
- The member variables carry over to all the lines below it unless changed:
chassis.turn_max_voltage = 3;
chassis.turn_to_angle(90);
chassis.turn_to_angle(0);
- The member variables can change to another desired variable or back to default:
chassis.turn_max_voltage = 3;
chassis.turn_to_angle(90);
default_constants();
chassis.turn_to_angle(0);
- The left and right drive motor groups are also able to change separately:
chassis.DriveR.spin(fwd, 12, volt);

My plan is to implement the overload methods and member variables in to all of my programs to improve accuracy and speed in movements.

Project PID/Programming Notes

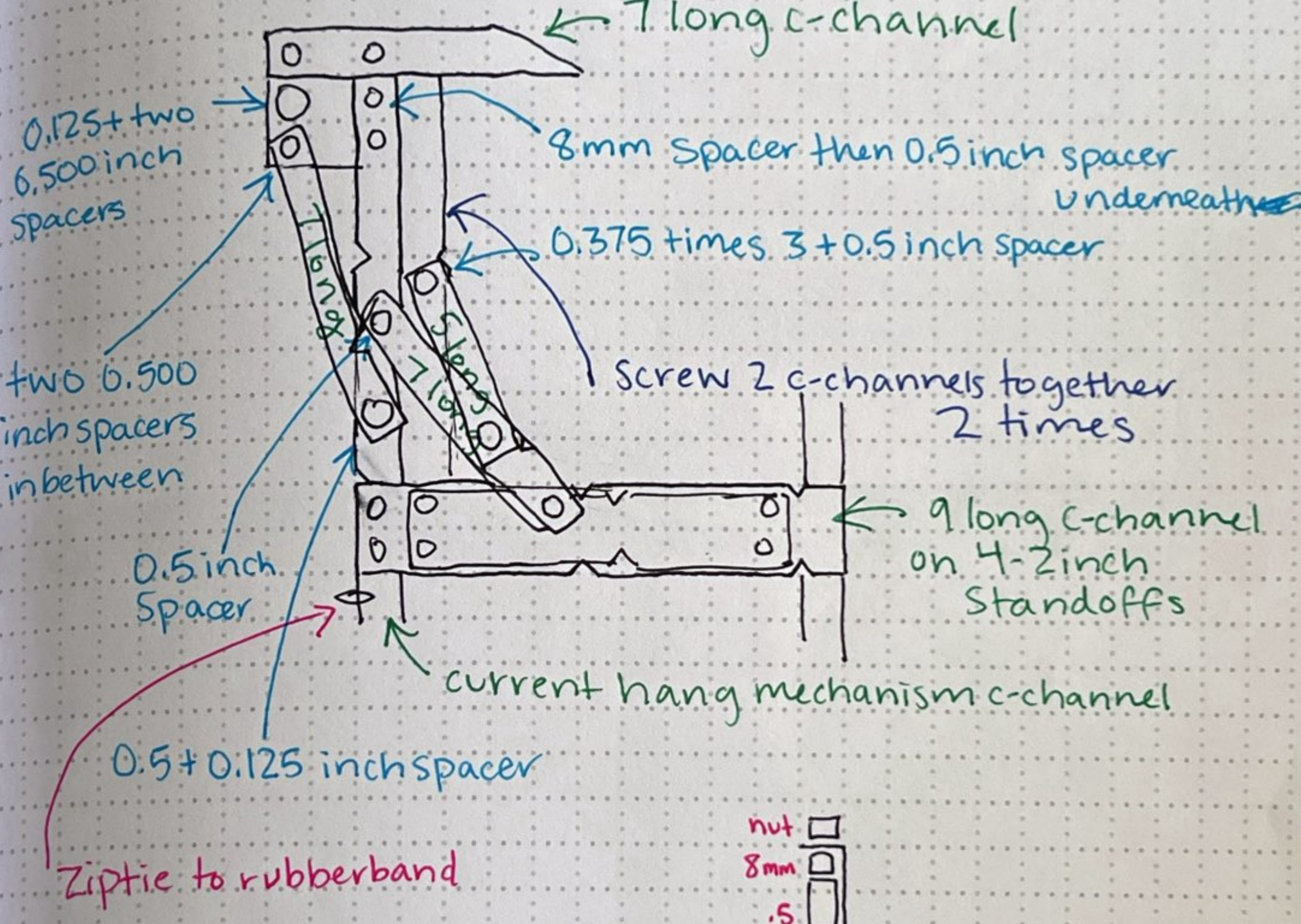
Date 1/30/2024

Name Harris Perkins

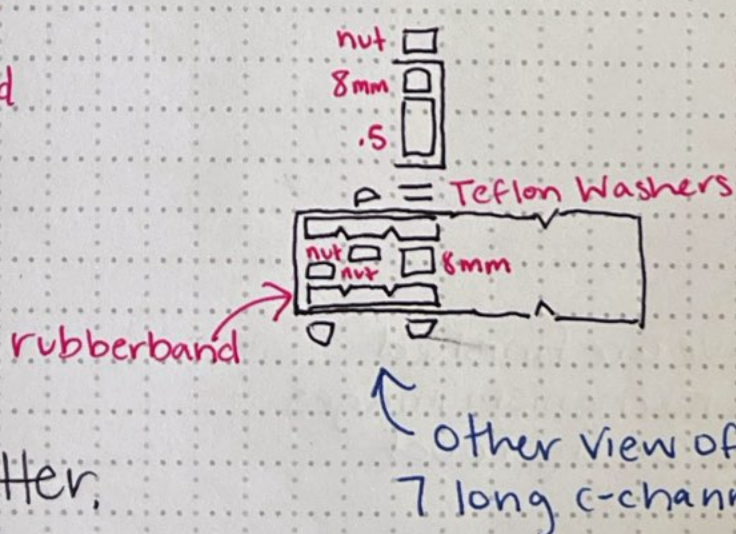
Building C/D Tier Hang:

Goals for today → Add a new, higher way to elevate to our robot so we can double hang with an alliance

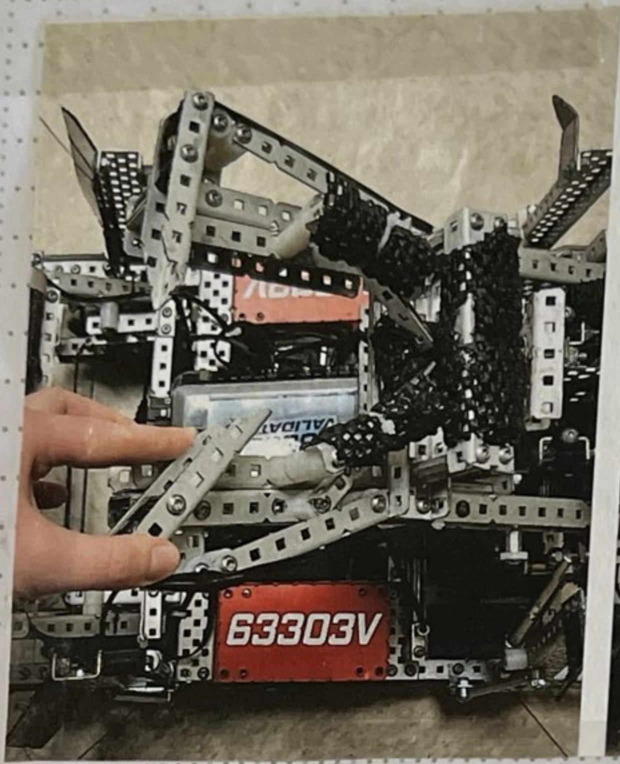
Below is a diagram for how we built the C/D Tier Hang Mechanism. Mirror the left to the right.

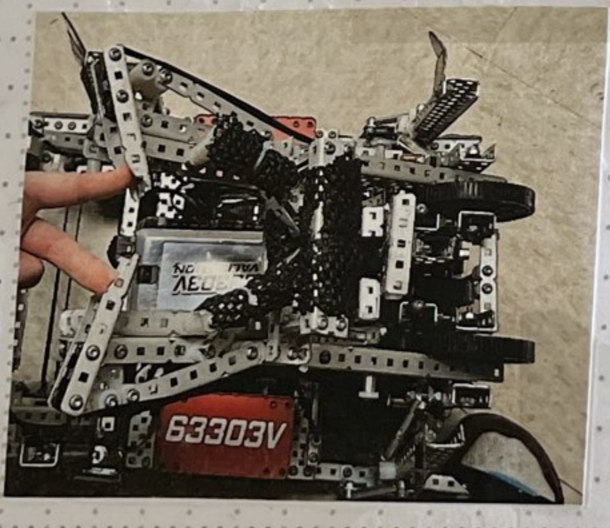


Add Mesh grip around all this metal to grip elevation bar better.



Finished Hang Images:



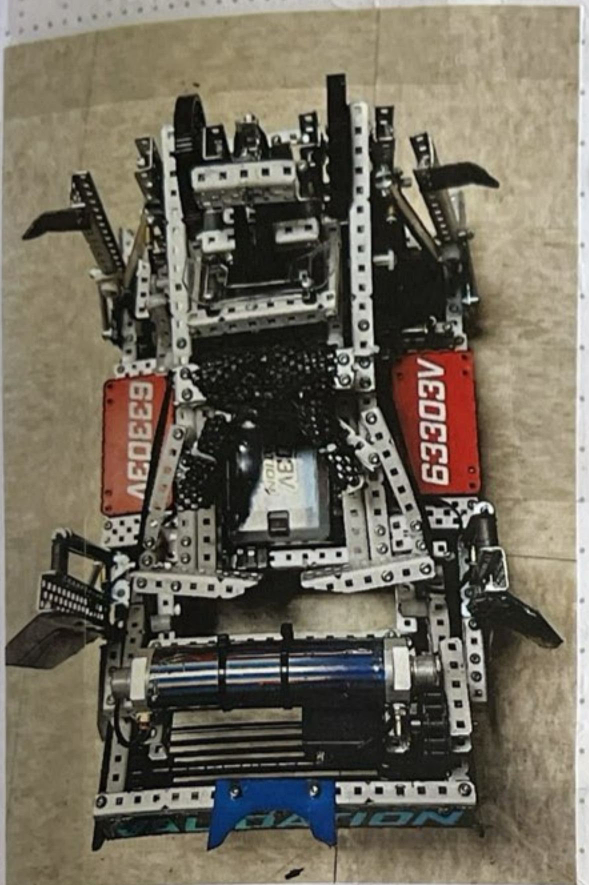


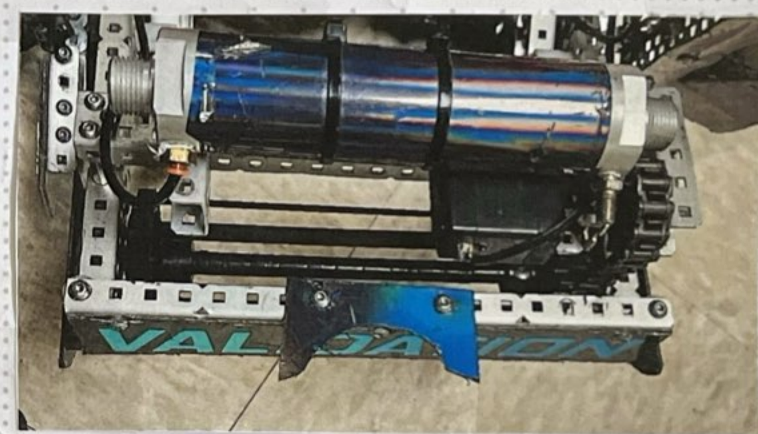
Above are finished hang mechanism images of it on the robot.

Project Finished Hang Images   Name Vanessa Perkins

Date 1/30/2024

(Continued)





Above are more images of the hang from different angles as well as images of a Lexan curve we added on top of the intakes to help line up the hang on the elevation bar.

VP

Project Finished Hang Images (contin.) Name Vanessa Perkins
Date 1/30/2024

Testing New Hang Mechanism:

Goals for today → Test the new hang mechanism
→ Make adjustments as needed

Testing

When testing the hang we found that
1. The robot struggles to drive up to the elevation bar.
2. The Kicker/hang motor is not quite strong enough the lift the robot quickly.

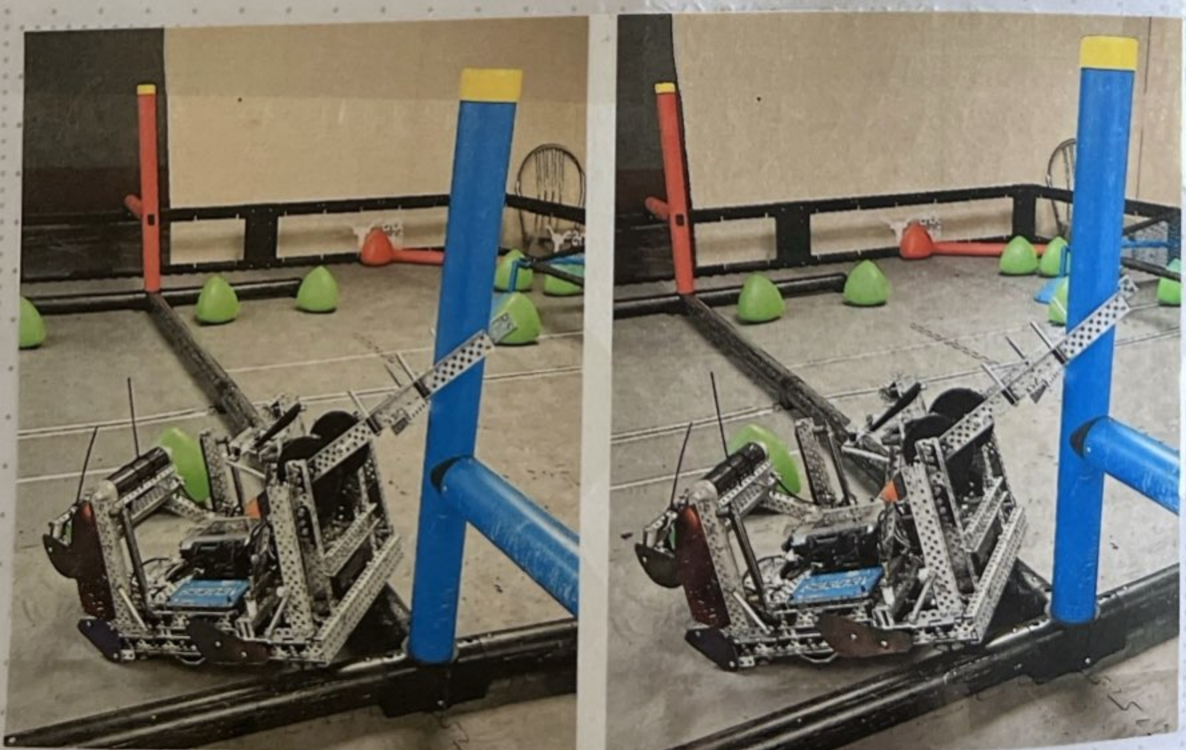
Brainstorming

1. Add sleds in the back
2. Add a 5.5 watt motor to the hang mechanism to increase the strength

Implement Solution

We will add sleds to the back of the base and another motor to the hang so our elevation is smoother and quicker.

Below is how we went above testing the hang through our build process.

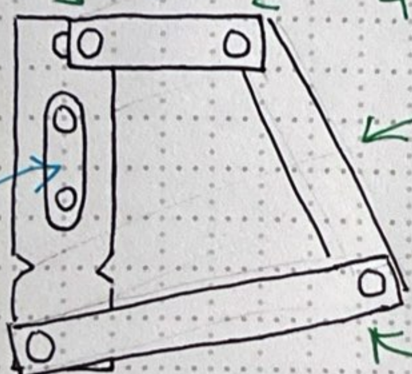


Project Testing New Hang Mechanism Name Vanessa Perkins
Date 1/31/2024

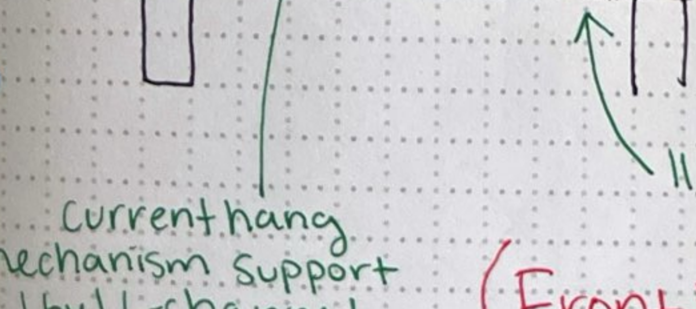
Adding 5.5 Watt Motor:

Goals for today → Improve the speed and strength of the hang mechanism/kicker by adding a motor

Below is a diagram from the front and side view of how we added another motor. We chained a 6 tooth sprocket to a 12 tooth sprocket to gear the 200 rpm 5.5 watt motor down to 100 rpm to match the current motor. The total power is 16.5 watt.



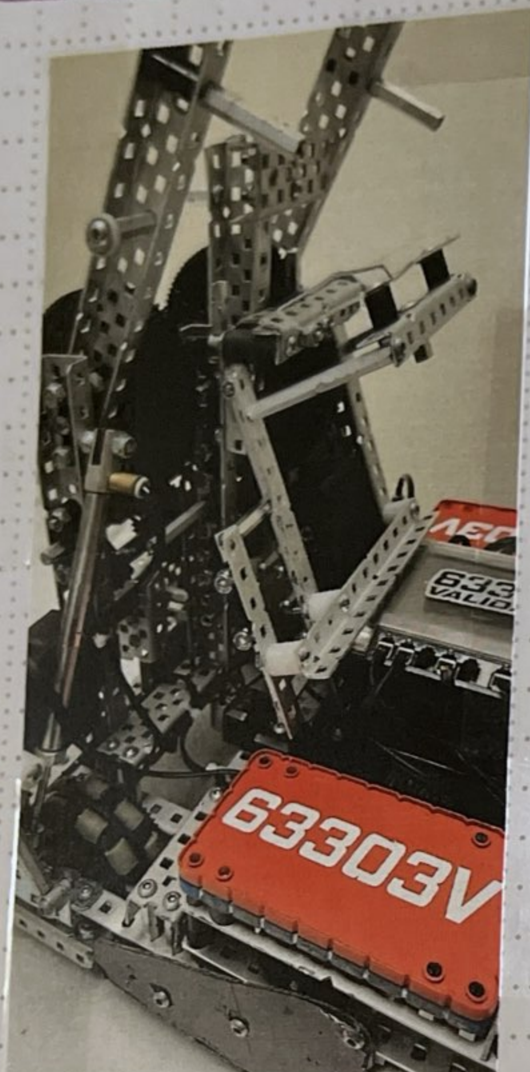
(Side View)

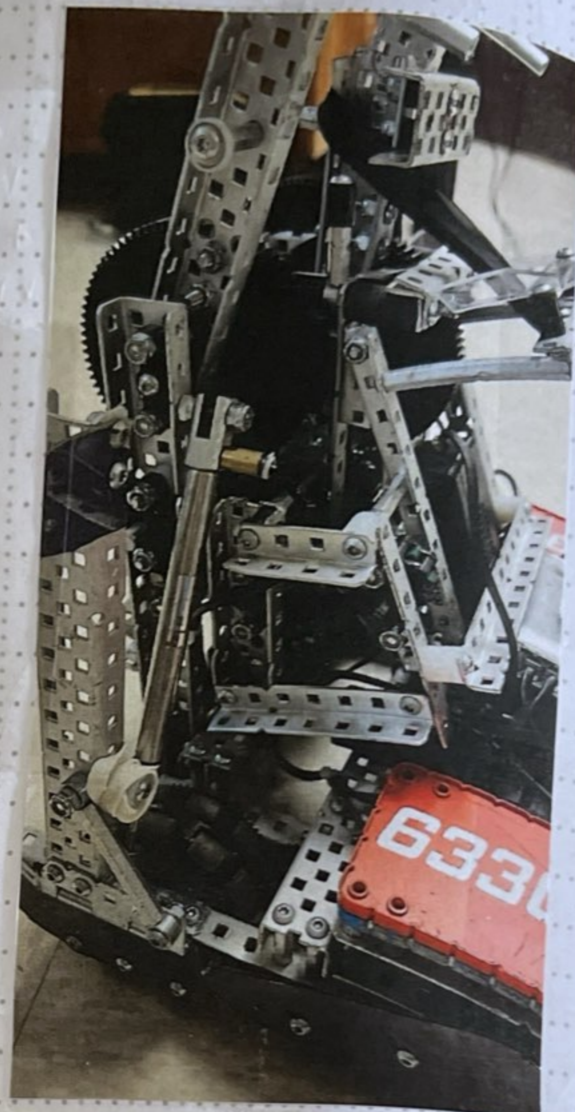


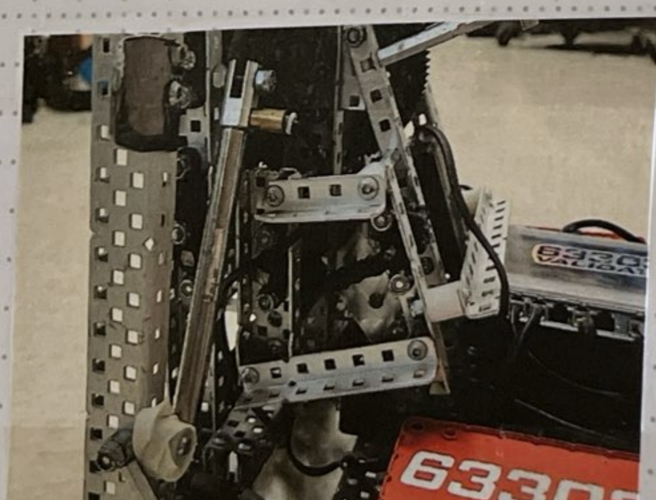
(Front View)

Project Adding 5.5 Watt Motor
Date 2/1/2024
Name Vanessa Perkins

5.5 Watt Motor to Hang Images:





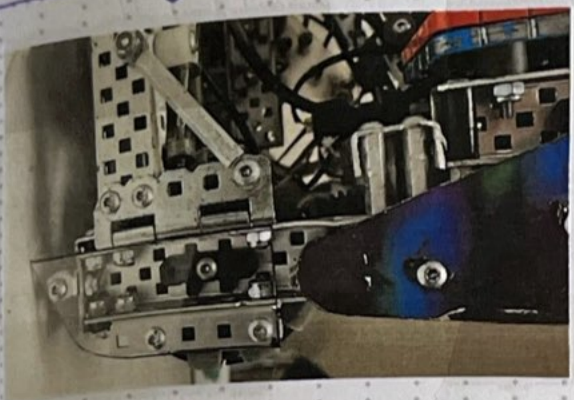


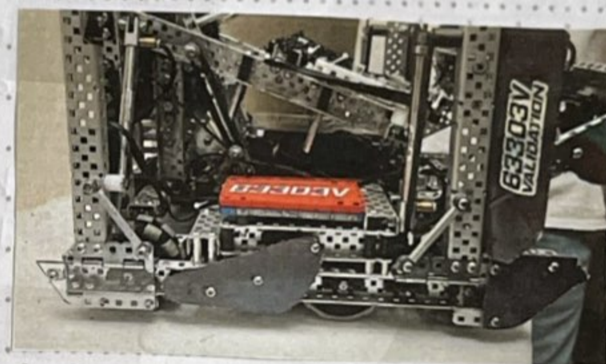
Project 5.5 Watt Motor to Hang Images
Name Vanessa Perkins
Date 2/1/2024

Back Sleds Prototypes:

Goals for today → Find a sled design that allows our robot to get up to the elevation bar and fix the problem from page 54

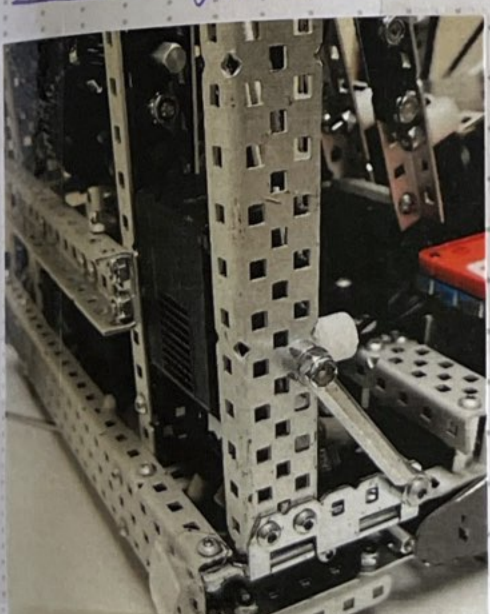
Prototype One: Basic Sleds



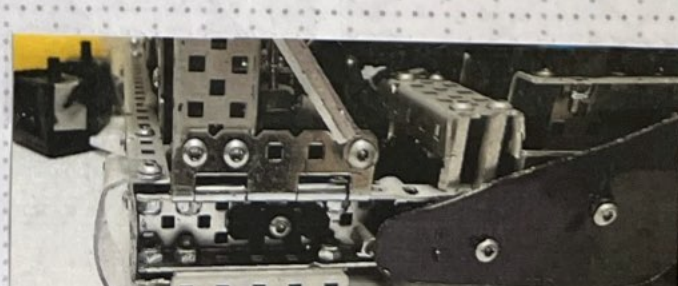


This first sled is a basic rounded Lexan sled. When tested it was too blunt and not smooth/round enough to drive up over the barrier.

Prototype Two: Bumper Sleds



These bumper sleds were smoother, but did not have enough height to push over the top of the barrier. We need a mixture of prototype 1 and 2.



(Continued)

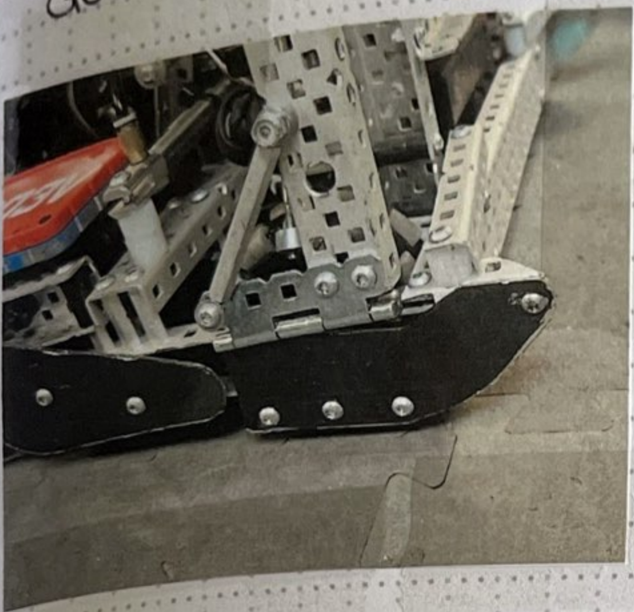
Prototype Three: Mixed Sleds

This sled is a mixture of the last two with height and smooth curve. It worked well and we will be making some final adjustments on the next page to make them our official sleds. Below and to the right are images of the current sleds.

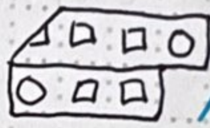
VP

Finish Back Sleds:

Goals for today → Finish Back Sleds prototype



Today we finished the official sleds. Below are the parts list, Lexan template, and diagrams.



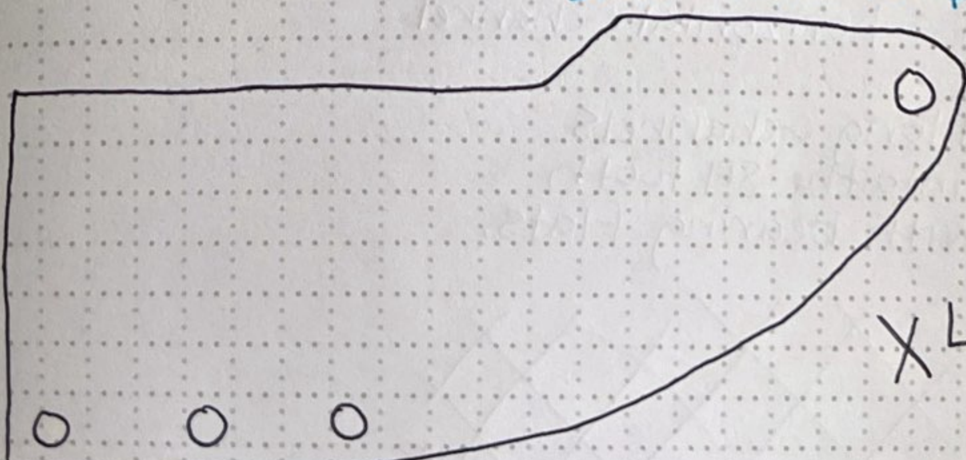
4 long 1by1 L-channel cut to fit to base and sled

Parts List:

- 2-4 long 1by1 L-channels
- 2-7 long 1by1 L-channels
- 4-0.500 inch spacers
- 4-0.750 inch screws
- 6-0.500 inch screws
- 10 nylock nuts
- 4-Lexan sleds

Use this with spacers to stabilize sleds

Lexan Sled Template

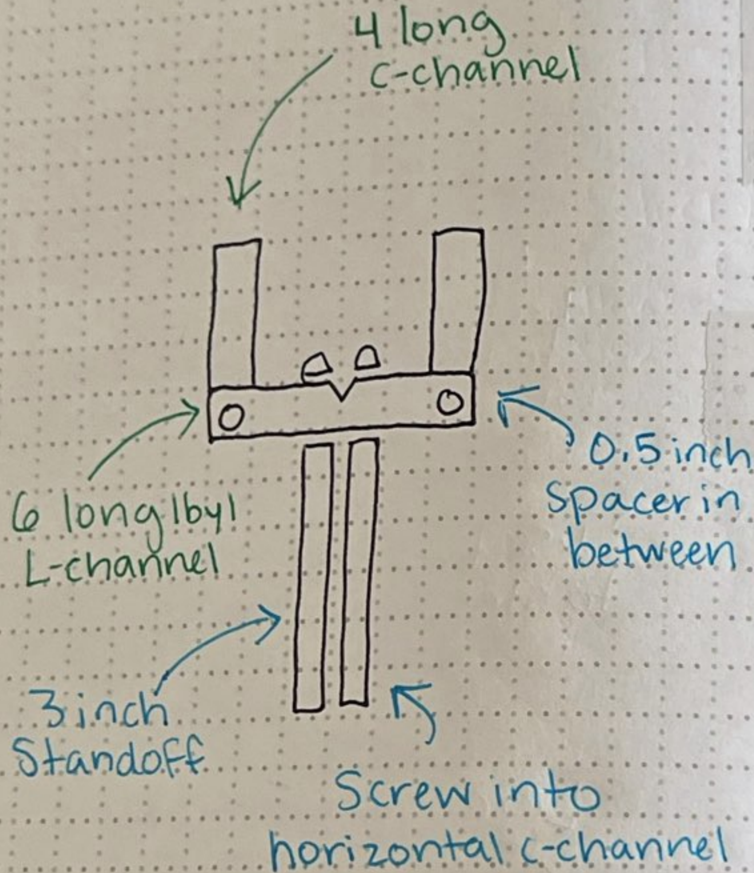


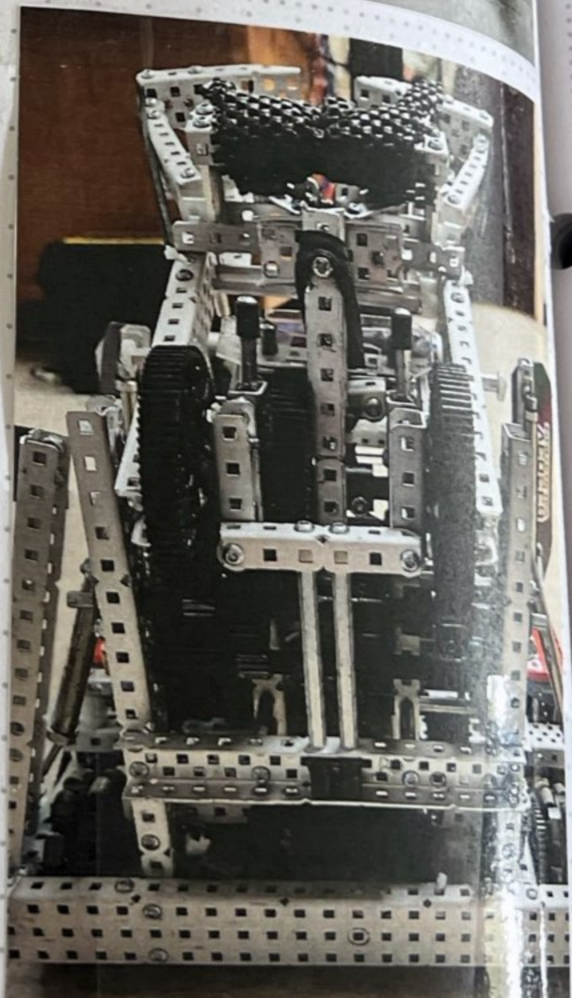
X4

Add Back Bracing to Gears:

Goals for today → Brace the gears so they stop slipping so much

Below are images of the complete bracing as well as the diagram of how to build it so the hang mechanism gears don't slip.





The 4 long c-channels screws into the 84 tooth gears with bearing flats.

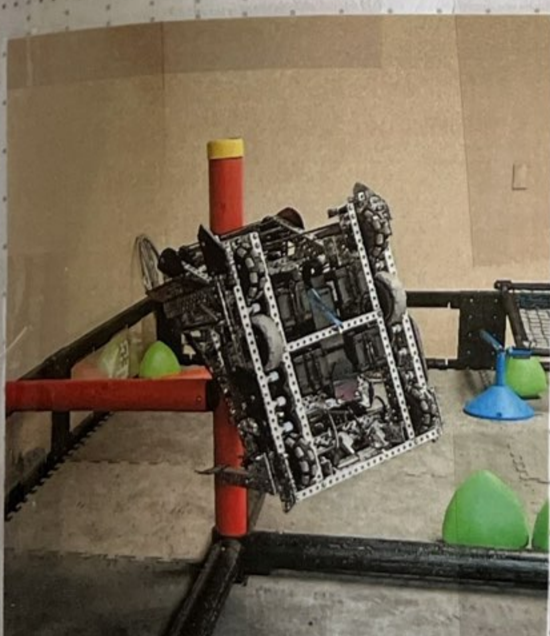
VP

Testing New Hang Mechanism Improved:

Goals for today → Test the hang again with the new improvements
→ adjust as needed

To the left are images of our higher C/D tier hang working so that we can potientially double hang with our alliances.

No adjustments were necessary.



VP

Skills Triball Tray:

Goals for today → Find a way to make skills matchloading faster

Problem/Challenge

We need a faster way to matchload during skills.

Brainstorming

1. Practice a ton
2. Grippy gloves
3. Bigger tray
4. Slow down kicker

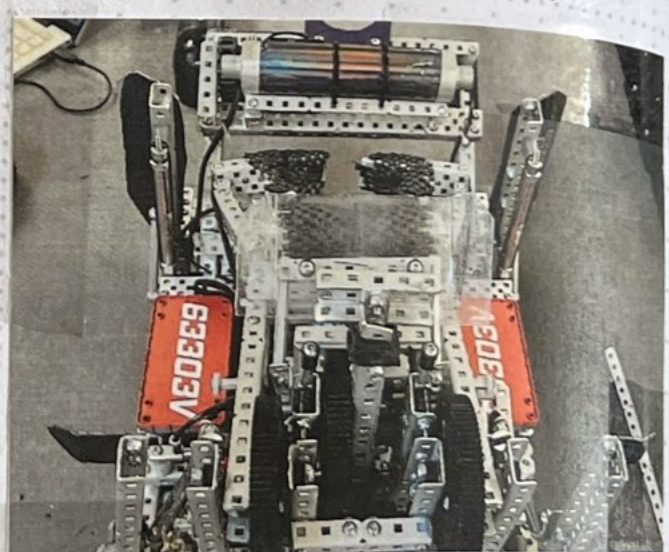
Build/Implement Solution

Make a Lexan tray that we can screw on for skills instead of the normal little one. Below are images of this tray being implemented on the robot and with a triball on top.

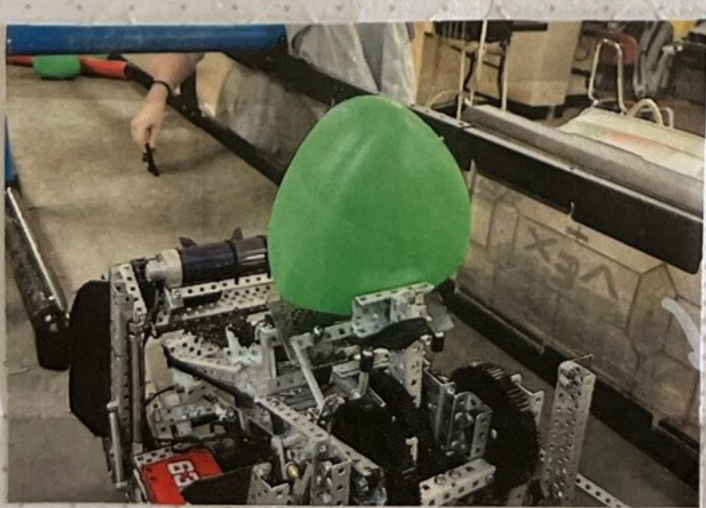
Select Approach

3

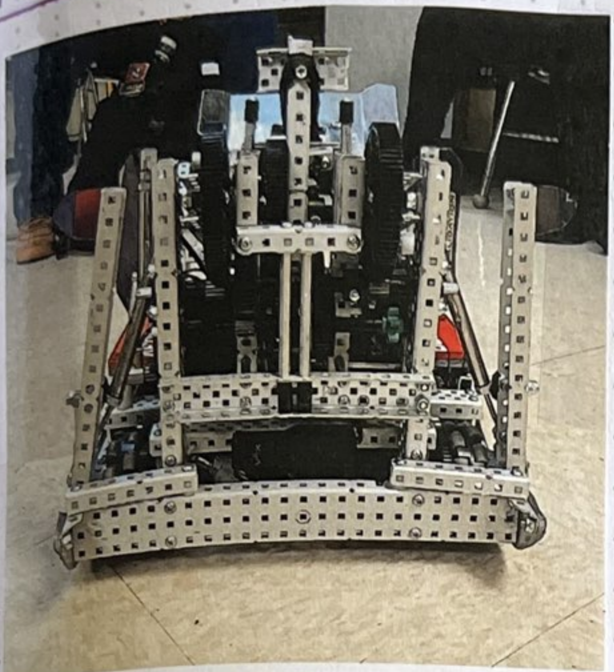
Attach a bigger Lexan tray so you can place them faster.

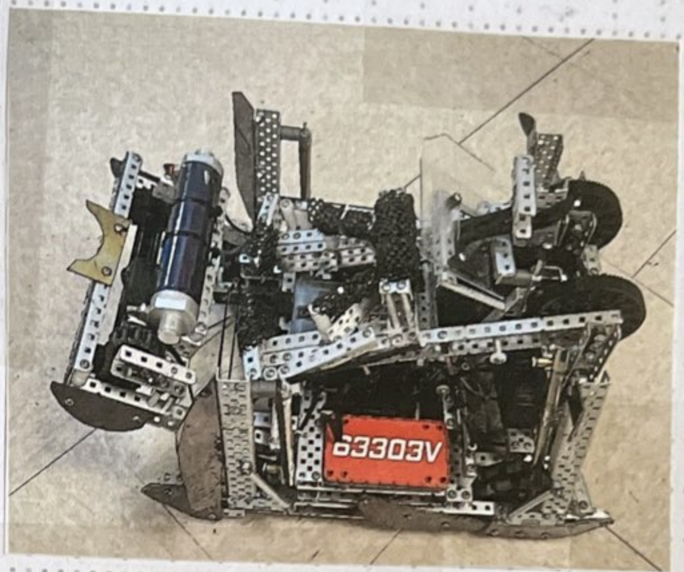


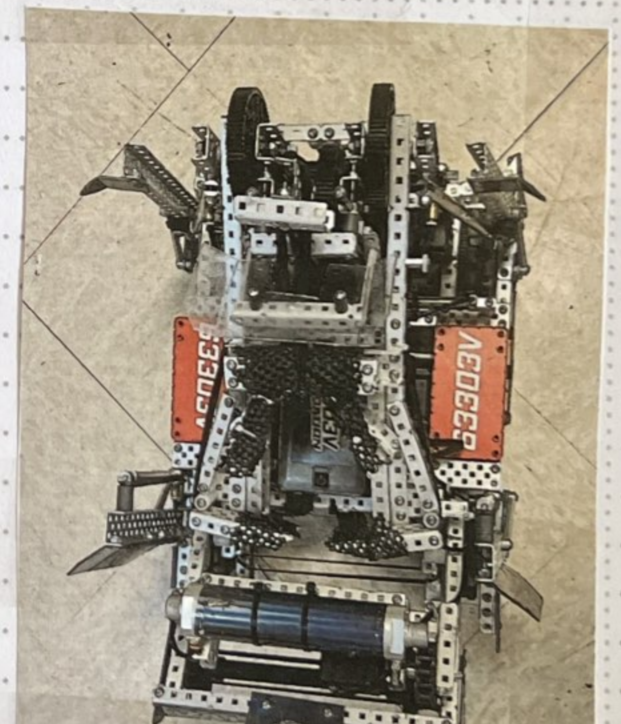
VP

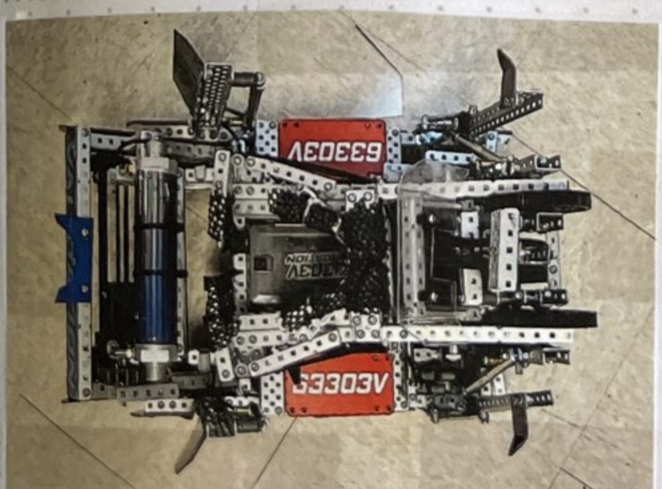


Full Robot Images:









Close Side Auton

```
#include "vex.h"

// ---- START VEXCODE CONFIGURED DEVICES ----
// Robot Configuration:
// [Name]               [Type]          [Port(s)]
// Controller1          controller
// Kicker               motor           10
// Intake               motor           9
// DriveFrontLeft       motor           3
// DriveCenterLeft      motor           1
// DriveBackLeft        motor           4
// DriveFrontRight      motor           17
// DriveCenterRight     motor           18
// DriveBackRight       motor           11
// FrontWings           digital_out     G
// BackWingLeft         digital_out     A
// BackWingRight        digital_out     C
// PTO                  digital_out     E
// Inertial             inertial        16
// Kicker2              motor           12
// ---- END VEXCODE CONFIGURED DEVICES ----

using namespace vex;
competition Competition;

/*---------------------------------------------------------------------------*/
/*                              VEXcode Config                               */
/*                                                                           */
/*  Before you do anything else, start by configuring your motors and        */
/*  sensors using the V5 port icon in the top right of the screen. Doing     */
/*  so will update robot-config.cpp and robot-config.h automatically, so     */
/*  you don't have to. Ensure that your motors are reversed properly. For    */
/*  the drive, spinning all motors forward should drive the robot forward.   */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/*                            JAR-Template Config                            */
/*                                                                           */
/*  Where all the magic happens. Follow the instructions below to input      */
/*  all the physical constants and values for your robot. You should         */
/*  already have configured your robot manually with the sidebar configurer. */
/*---------------------------------------------------------------------------*/

Drive chassis(

//Specify your drive setup below. There are seven options:
//ZERO_TRACKER_NO_ODOM, ZERO_TRACKER_ODOM, TANK_ONE_ENCODER, TANK_ONE_ROTATION, TANK_TWO_ENCODER, TANK_TWO_ROTATION, HOLONOMIC_TWO_ENCODER, and HOLONOMIC_TWO_ROTATIO
//For example, if you are not using odometry, put ZERO_TRACKER_NO_ODOM below:
ZERO_TRACKER_NO_ODOM,

//Add the names of your Drive motors into the motor groups below, separated by commas, i.e. motor_group(Motor1,Motor2,Motor3).
//You will input whatever motor names you chose when you configured your robot using the sidebar configurer, they don't have to be "Motor1" and "Motor2".

//Left Motors:
motor_group(DriveFrontLeft,DriveCenterLeft,DriveBackLeft),

//Right Motors:
motor_group(DriveFrontRight,DriveCenterRight,DriveBackRight),
```

```
//Specify the PORT NUMBER of your inertial sensor, in PORT format (i.e. "PORT1", not
simply "1"):
PORT16,

//Input your wheel diameter. (4" omnis are actually closer to 4.125"):
3.25,

//External ratio, must be in decimal, in the format of input teeth/output teeth.
//If your motor has an 84-tooth gear and your wheel has a 60-tooth gear, this value
will be 1.4.
//If the motor drives the wheel directly, this value is 1:
0.6,

//Gyro scale, this is what your gyro reads when you spin the robot 360 degrees.
//For most cases 360 will do fine here, but this scale factor can be very helpful when
precision is necessary.
360,

/*---------------------------------------------------------------------------*/
/*                              PAUSE!                                       */
/*                                                                           */
/*  The rest of the drive constructor is for robots using POSITION TRACKING. */
/*  If you are not using position tracking, leave the rest of the values as  */
/*  they are.                                                                */
/*---------------------------------------------------------------------------*/

//If you are using ZERO_TRACKER_ODOM, you ONLY need to adjust the FORWARD TRACKER
CENTER DISTANCE.

//FOR HOLONOMIC DRIVES ONLY: Input your drive motors by position. This is only
necessary for holonomic drives, otherwise this section can be left alone.
//LF:      //RF:
PORT1,     -PORT2,

//LB:      //RB:
PORT3,     -PORT4,

//If you are using position tracking, this is the Forward Tracker port (the tracker
which runs parallel to the direction of the chassis).
//If this is a rotation sensor, enter it in "PORT1" format, inputting the port below.
//If this is an encoder, enter the port as an integer. Triport A will be a "1",
Triport B will be a "2", etc.
3,

//Input the Forward Tracker diameter (reverse it to make the direction switch):
2.75,

//Input Forward Tracker center distance (a positive distance corresponds to a tracker
on the right side of the robot, negative is left.)
//For a zero tracker tank drive with odom, put the positive distance from the center
of the robot to the right side of the drive.
//This distance is in inches:
-2,

//Input the Sideways Tracker Port, following the same steps as the Forward Tracker
Port:
1,

//Sideways tracker diameter (reverse to make the direction switch):
-2.75,

//Sideways tracker center distance (positive distance is behind the center of the
robot, negative is in front):
```

Project Close Side Auton Program (Continued) Name Harris Perkins
Date 2/9/2024

```
5.5
};

int current_auton_selection = 0;
bool auto_started = false;

void pre_auton(void) {
 // Initializing Robot Configuration. DO NOT REMOVE!
 vexcodeInit();
 default_constants();

 }

void autonomous(void) {
 auto_started = true;
  Intake.spin(reverse,100,pct);
 wait(0.75,sec);
 Intake.setStopping(hold);
 Intake.stop();
 chassis.drive_distance(48);
 Intake.spin(forward,100,pct);
 wait(0.5,sec);
 chassis.turn_to_angle(10);
 chassis.drive_distance(-36);
 chassis.right_swing_to_angle(170);
 chassis.drive_distance(13);
 chassis.turn_to_angle(90);
 BackWingRight.set(true);
 wait(0.5,sec);
 chassis.turn_to_angle(87);
 BackWingRight.set(false);
 chassis.drive_distance(20);
 Intake.spin(reverse,100,pct);
 chassis.drive_distance(23);


}

/*---------------------------------------------------------------------------*/
/*                                                                           */
/*                              User Control Task                            */
/*                                                                           */
/*  This task is used to control your robot during the user control phase of */
/*  a VEX Competition.                                                       */
/*                                                                           */
/*  You must modify the code to add your own robot specific commands here.   */
/*---------------------------------------------------------------------------*/

void usercontrol(void) {
 // User control code here, inside the loop
 while (1) {

   DriveFrontLeft.spin(forward, Controller1.Axis3.position(percent) +
   Controller1.Axis1.position(percent), percent); // The code for driving with the
Front Left motor

   DriveBackLeft.spin(forward, Controller1.Axis3.position(percent) +
Controller1.Axis1.position(percent), percent); // Left Back Drive Code

   DriveCenterLeft.spin(forward, Controller1.Axis3.position(percent) +
Controller1.Axis1.position(percent), percent); // Left Center Drive Code
   DriveFrontRight.spin(forward, Controller1.Axis3.position(percent) -
```

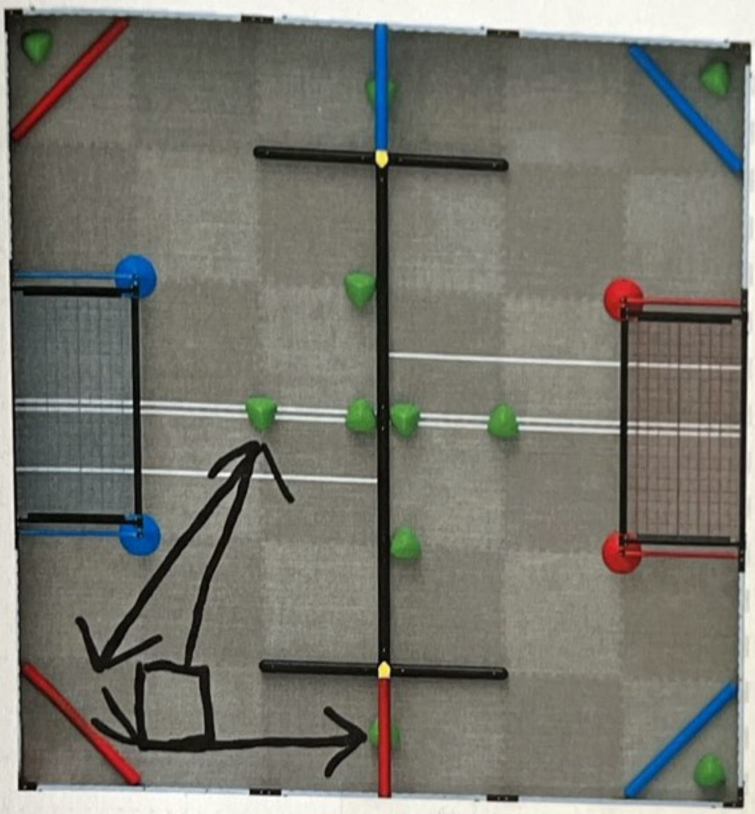
Project Close Side Auton Program (Contin.) Name Harris Perkins
Date 2/9/2024

```
Controller1.Axis1.position(percent), percent); // Right Front Drive Code

DriveCenterRight.spin(forward, Controller1.Axis3.position(percent) -
Controller1.Axis1.position(percent), percent); // Right Center Drive Code

DriveBackRight.spin(forward, Controller1.Axis3.position(percent) -
Controller1.Axis1.position(percent), percent); // Right Back Drive Code


  if(Controller1.ButtonR1.pressing())
{ Intake.setVelocity(100,percent);
  Intake.spin(vex::directionType::fwd, 100, velocityUnits::pct);
  Intake.setStopping(hold);

  }

else if(Controller1.ButtonR2.pressing())
  { Intake.setVelocity(-100,percent);
  Intake.spin(vex::directionType::fwd, -100, velocityUnits::pct);
  Intake.setStopping(hold);
  }



else if(Controller1.ButtonX.pressing())
  {FrontWings.set(true);
  }

else if(Controller1.ButtonA.pressing())
  {FrontWings.set(false);
  }

else if(Controller1.ButtonDown.pressing())
  {BackWingLeft.set(false);
  }

else if(Controller1.ButtonLeft.pressing())
  {BackWingLeft.set(true);
  }

else if(Controller1.ButtonRight.pressing())
  {BackWingRight.set(false);
  }

else if(Controller1.ButtonUp.pressing())
  {BackWingRight.set(true);
  }

else if(Controller1.ButtonY.pressing())
  {PTO.set(true);
  Kicker.spin(vex::directionType::fwd,10, velocityUnits::pct);
  Kicker2.spin(vex::directionType::fwd,10, velocityUnits::pct);
  }

else if(Controller1.ButtonL1.pressing())
  {Kicker.spin(vex::directionType::fwd, 60, velocityUnits::pct);
   Kicker2.spin(vex::directionType::fwd, 60, velocityUnits::pct);
  }

else if(Controller1.ButtonL2.pressing())
  {Kicker.spin(vex::directionType::fwd, -100, velocityUnits::pct);
```

```
      Kicker2.spin(vex::directionType::fwd, -100, velocityUnits::pct);
    }

      else
    {
      Intake.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);

      Kicker.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);

      Kicker2.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);

    }

    // This is the main execution loop for the user control program.
    // Each time through the loop your program should update motor + servo
    // values based on feedback from the joysticks.

    // ........................................................................
    // Insert user code here. This is where you use the joystick values to
    // update your motors, etc.
    // ........................................................................

    //Replace this line with chassis.control_tank(); for tank drive
    //or chassis.control_holonomic(); for holo drive.


    wait(20, msec); // Sleep the task for a short amount of time to
                    // prevent wasted resources.
  }
}

//
// Main will set up the competition functions and callbacks.
//
int main() {
  // Set up callbacks for autonomous and driver control periods.
  Competition.autonomous(autonomous);
  Competition.drivercontrol(usercontrol);

  // Run the pre-autonomous function.
  pre_auton();

  // Prevent main from exiting with an infinite loop.
  while (true) {
    wait(100, msec);
  }
}
```

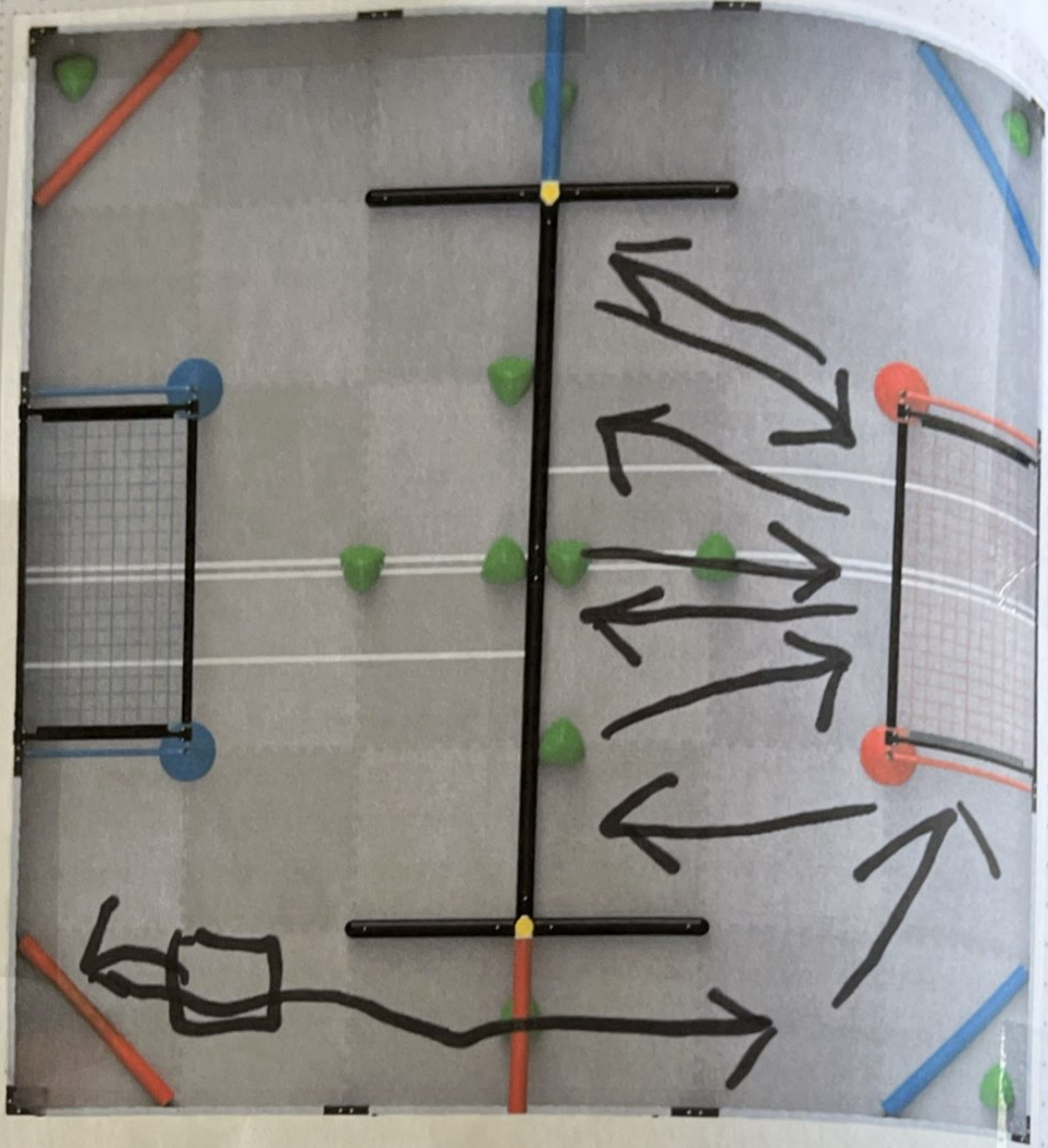
Close Side Auton Path:

The robot starts in the corner of the second tile. It goes for the triball in the middle of the field and returns back to the corner to remove the triball behind the matchload bar. After it removes the triball it pushes all three to the other side of the field and touches the horizontal hang bar.



Above is the path that our close side autonomous goes through to help us get towards a win point.

HP



Above is the path of our skills run with a description below.

The robot starts in the corner of the second tile and backs up to the match load bar. All 46 triballs can be launched across the field with the kicker in 25 seconds. After those 25 seconds the robot drives under the hang bar and pushes triballs into the right side of the goal. The robot then goes around to the front of the goal and pushes at three different angles to ensure that as many triballs as possible are scored.

Project Skills Auton Path

Date 2/9/2024

Name Harris Perkins

Competition Analysis:

Goals for today → Review our competition performance

On the 9th and 10th we had a signature event in Bristol. We had some ups and downs during our matches, but were glad to meet new teams and get some practice with our interviews.

We learned a few new strategies and were able to practice driving in skills.

We ended the competition in the round of 16 with team 24816T as our alliance. We also won the BUILD AWARD for our robot's durability and build.

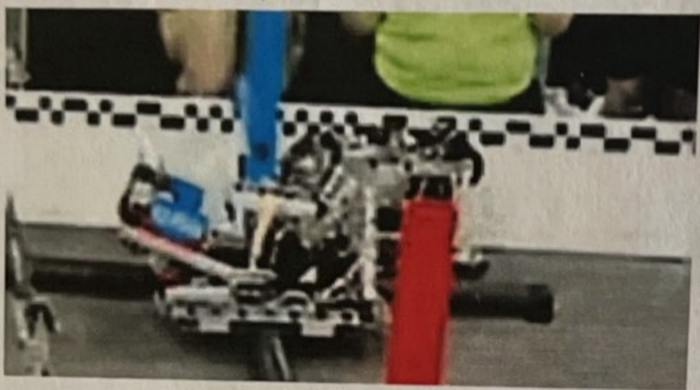
Below are some pictures from the competition.

Bristol Match Analyses:

Goals for today → Review our matches and look for ways to improve

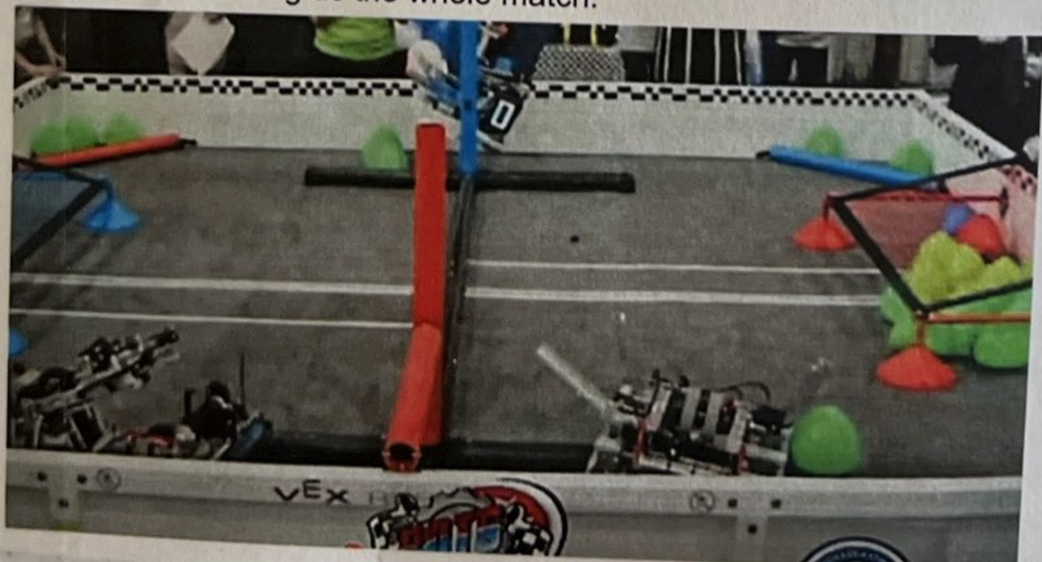
Qualifier #15	Feb 9th at 11:34 AM	8349U	50252A	76	**63303V**	52455A

In this match, we had an excellent autonomous and it worked flawlessly, however, we struggled bowling with defense. In this scenario, we should have used the other tunnel or done one at a time over the barrier. In addition, we need a faster hang so we can get more last-second points.



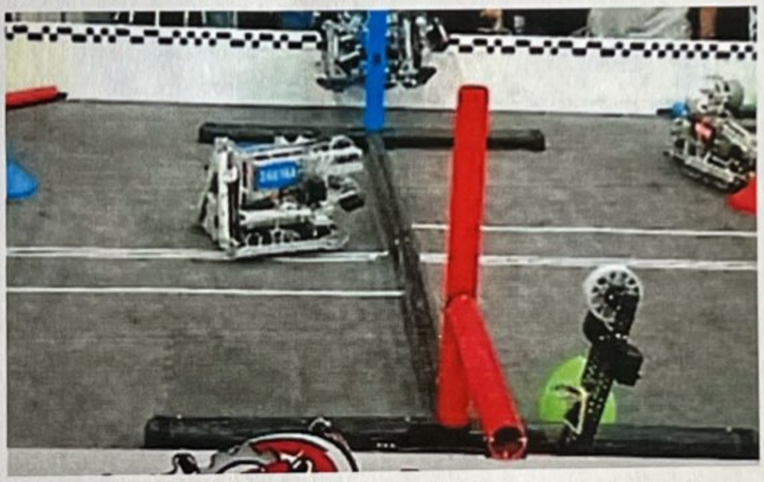
Qualifier #28	Feb 9th at 12:20 PM	44252R	**63303V**	80	16859C	57249A	79

In this match, our autonomous worked perfectly getting us another AWP. In this match our bowling was amazing, however, we did hold two in the field at once getting another warning. But, we were very ahead in the match until we could not get a last-second hang, almost costing us the whole match.



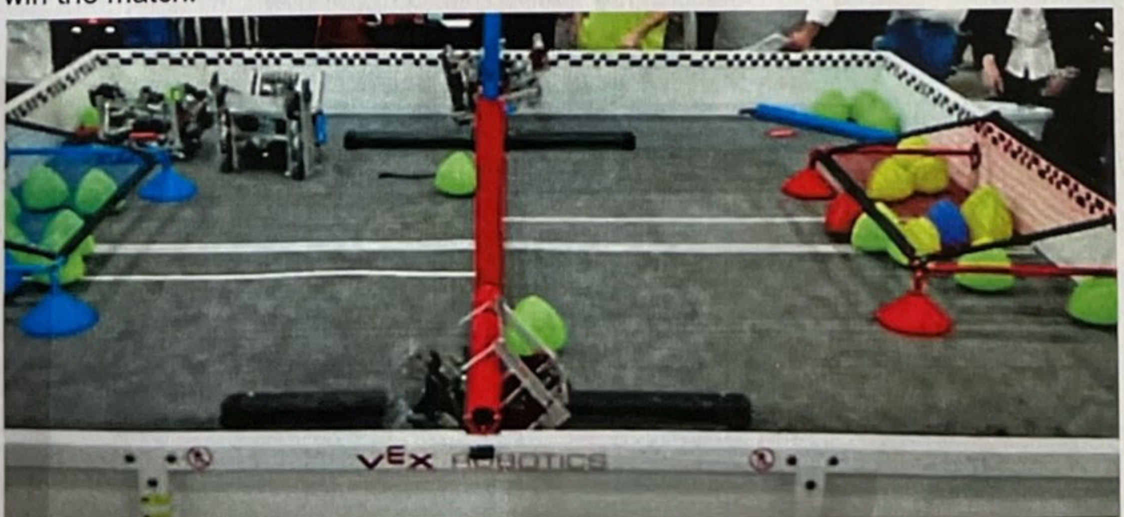
Qualifier #36	Feb 9th at 1:47 PM	40994F	50810E	59	24816A	**63303V**	100

In this match, we lost autonomous and the win point as we just barely missed descoring our triball. However, our bowling was much better as we were very consistent with it. Then we gave ourselves roughly 10 seconds to hang, allowing us to hang and win the match.



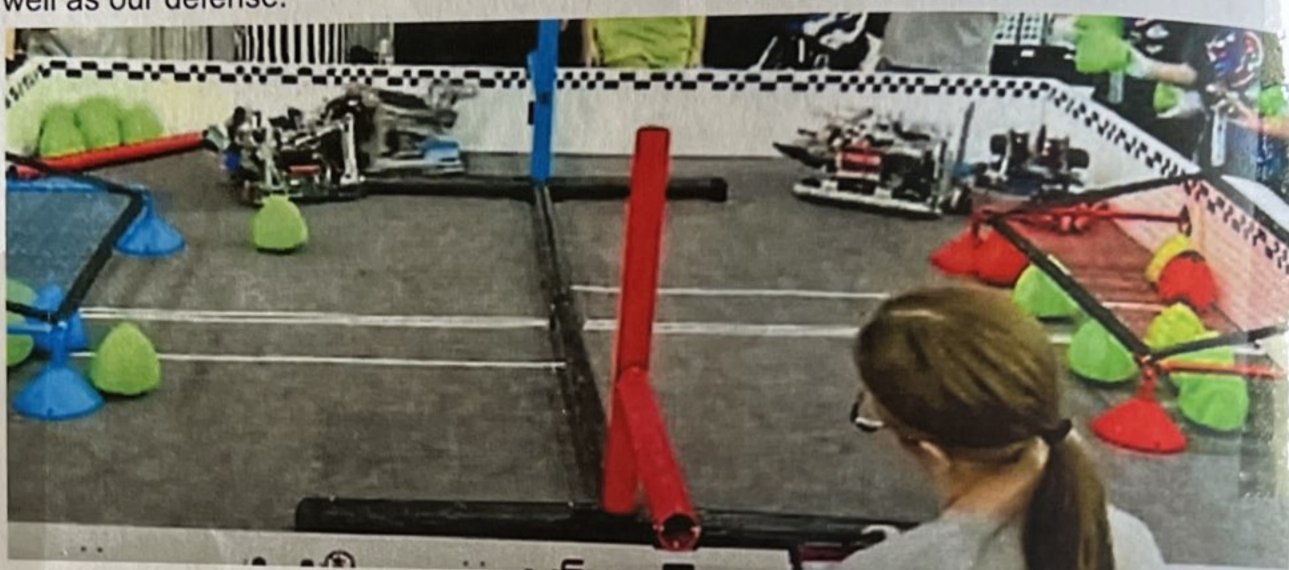
Qualifier #53	Feb 9th at 2:44 PM	**63303V**	42652B	79	31110C	3796F	74

In this match we won autonomous and win point, having a great match. Our bowling was great until we got entangled, however our alliance partner got off a hang allowing us to win the match.



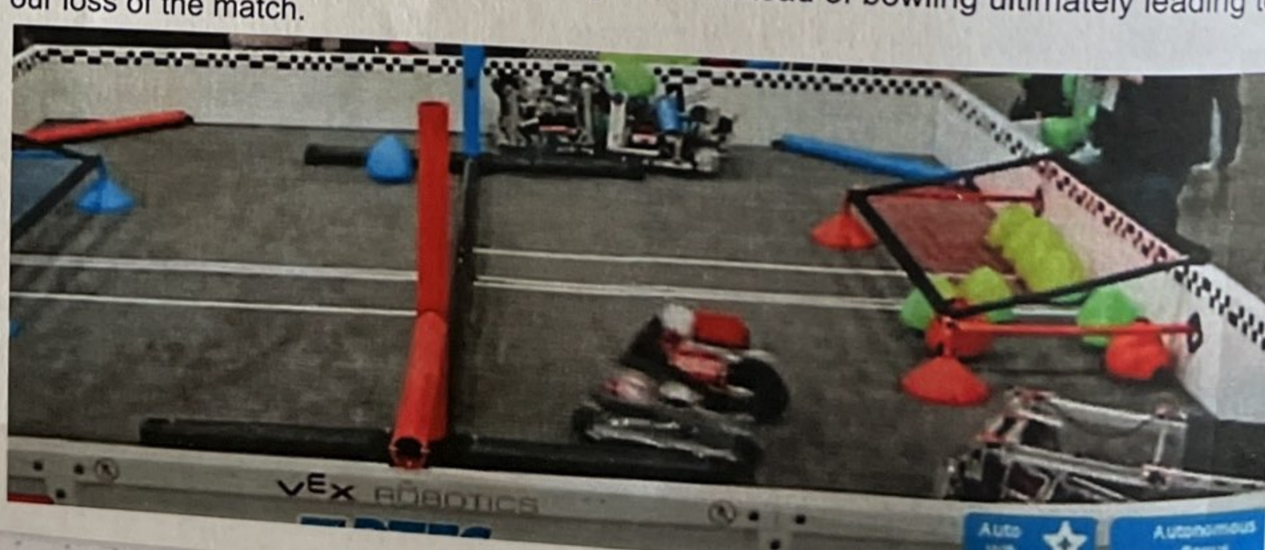
Qualifier #72	Feb 9th at 3:55 PM	**63303V**	24816R	71	18000B	16859E	48

In this match our autonomous did not work at all as we later found out that a drivetrai motor disconnected. Our wing also fell off in this match causing us even more issues, ultimately we ended up winning, but we had many issues as our bowling was limited a well as our defense.



Qualifier #85	Feb 9th at 4:46 PM	42652C	**63303V**	54	44252N	1691A	70

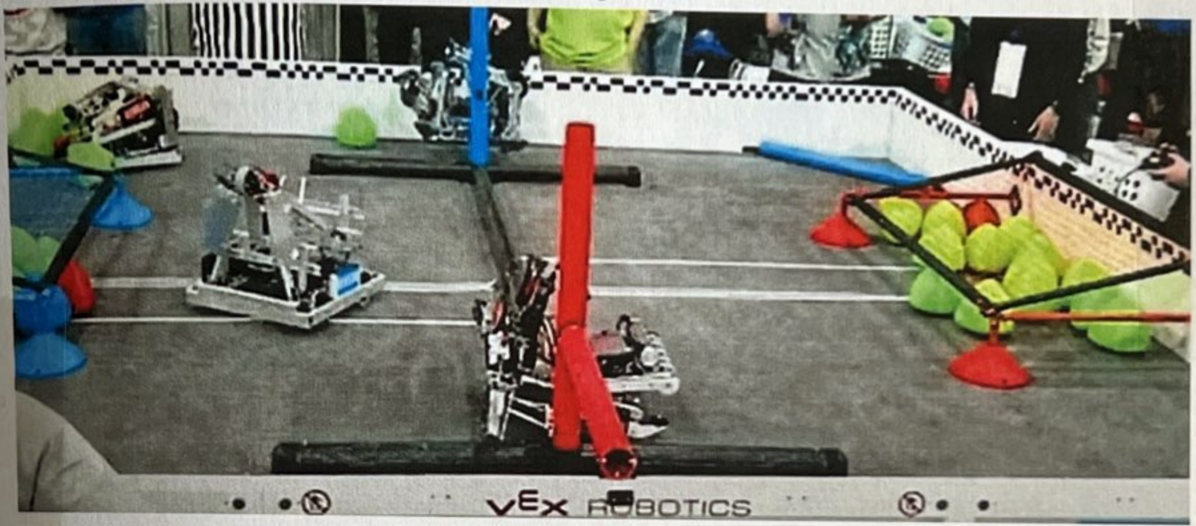
In this match our autonomous did not work at all as we later found out that a drivetrain motor disconnected. After not realizing this again we still tried to drive as normal. However, we spent too much time blocking bowls instead of bowling ultimately leading to our loss of the match.



Project Bristol Match Analyses (Continued)
Date 2/12/2024 Name Eli Fritts

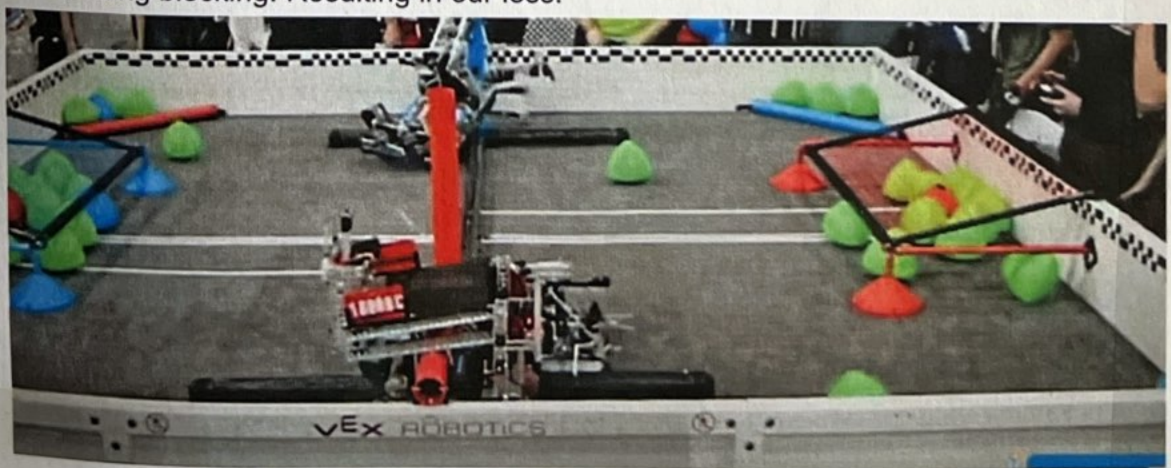
Qualifier #111	Feb 10th at 9:41 AM	42652A	8349G	103	48953B	**63303V**	76

In this match we got the win point however we lost autonomous. Our bowling was great this match, however we should have been better at traversing the field. Secondly, we got a DQ for illegally match loading, also resulting in our loss.



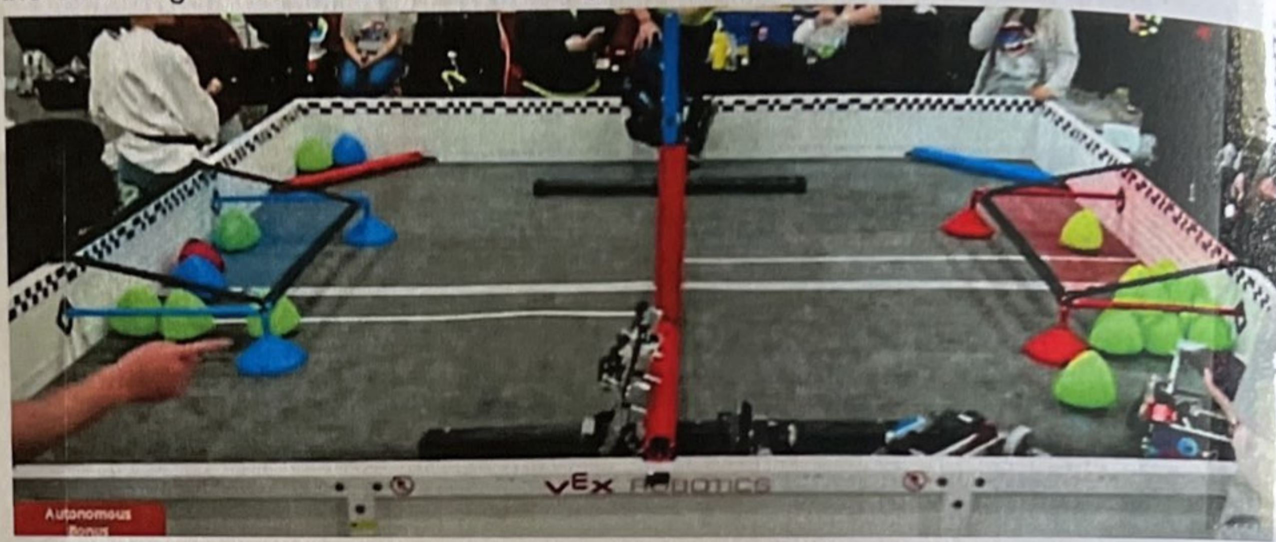
Qualifier #127	Feb 10th at 10:50 AM	96504C	18000C	97	**63303V**	9623S	89

In this match we lost the win point and the autonomous. Soon after, our bowling went very well, however, we were not able to get off the ground to hang and win the match due to hang blocking. Resulting in our loss.



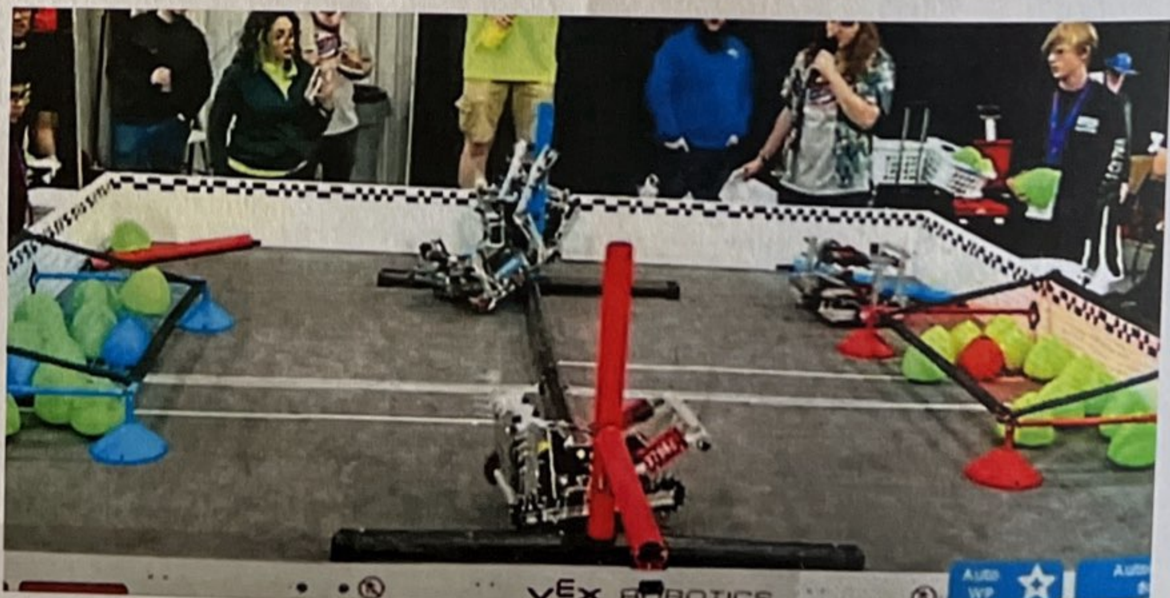
Qualifier #146	Feb 10th at 12:04 PM	19589A	**63303V**	71	8926W	48953F	47

In this match we lost autonomous and the win point as we ran our far side autonomous not working to its full potential. However, our bowling was really good that match. But at the end we got descored and could not get our hang last second.



Qualifier #159	Feb 10th at 2:09 PM	3796E	48953C	80	**63303V**	9909Y	103

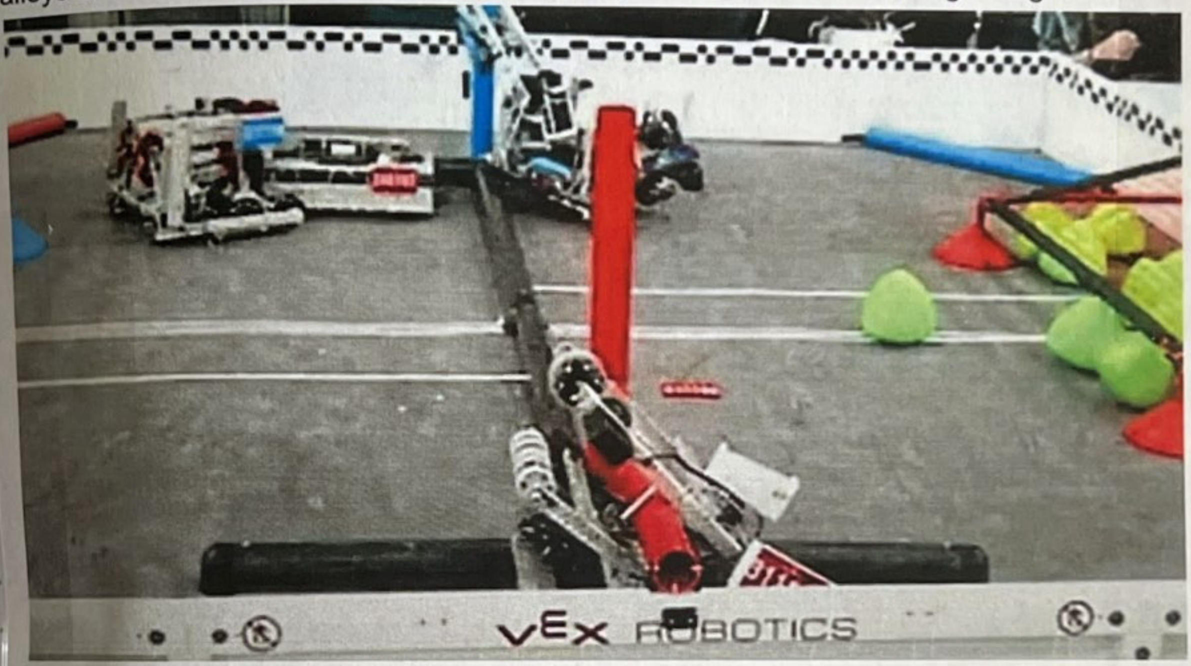
In this match, our win point and autonomous worked perfectly securing us the win point and bonus. However, instead of bowling, we tried a different strategy, starving, not matchloading, and using the triballs on the field. This worked very well, and we were very happy with the match, but we messed up on our backward hang and just fell short of it.



Project Bristol Match Analyses (Continued) Name Eli Fritts
Date 2/12/2024

Qualifier #168	Feb 10th at 2:38 PM	31110B	24816T	90	57711N	**63303V**	85

In this match everything worked perfectly in autonomous, however, our communication in this match was poor. While we tried to bowl this match, we should have used both alleys and communicated much better. We did pull off a last minute hang though.



R16 #2-1	Feb 10th at 4:18 PM	24816T	**63303V**	95	18000H	50810E	120

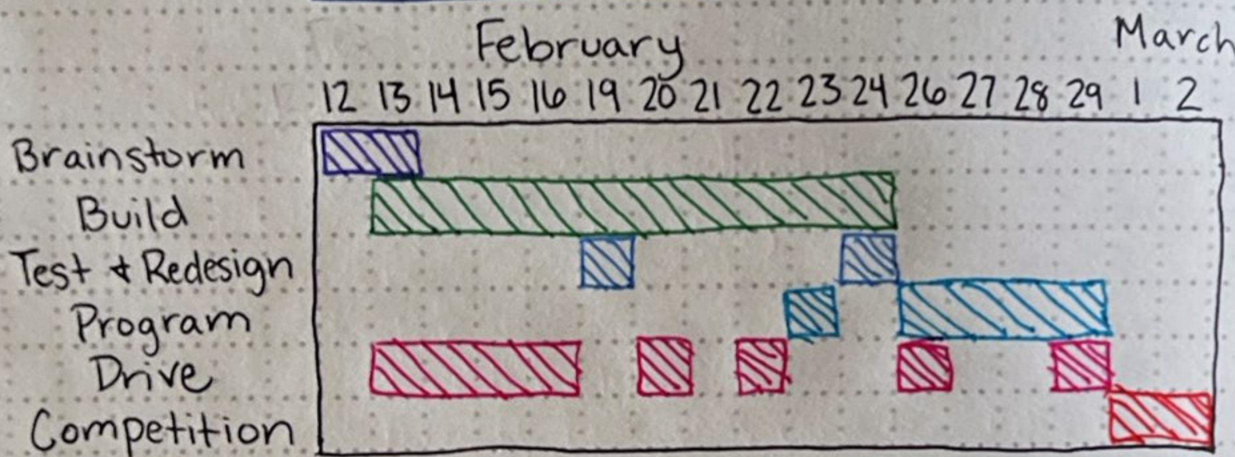
In our last match, our autonomous worked great. However, we were against two descoring robots, forcing us to play a very careful game, which ultimately slowed us down, allowing the other team to get a small upper hand. The bowling was great, however it could not be better due to the careful gameplay we had to play. Speed could fix this in the future.

Iteration 5

Timeline:

Goals for today → Plan out our time and make a Gantt chart to keep us on track and track our progress

Gantt Chart:



Above is our Gantt Chart for the next couple weeks leading up to States.

One thing that we did not include, that we hope to work in, is a scrimmage with several teams to help us prepare for States.

Goals:

Goals for today → Decide what our team wants to focus on before and at STATES
→ Divide up responsibilities for each goal

Robot Goals:

- Fits under goal
- Has open intake for easy matchloading
- Has consisent kicker
- Has double horizontal wings
- Has fast drivetrain with all omnis

Skills Goals:

- 170 point driver
- 160 point programmer
- Be in the TOP 30 Percent

States Goals:

- Win a Judged Award
- Make it to the Semifinals
- Have efficient scouting methods and charts to improve alliances

VP

As a team we discussed the responsibilities of each of these goals so we are aware of our jobs and roles. We will also be helping each other out in any way that we can.

Robot Design Options:

Goals for today → Brainstorm robot design options including rebuilds

Design Option One: 6 inch Versitile Robot

Specifics:

- 7 motor base
  - ↳ 6-11 watt plus 2-5.5 watt motors
  - → 600 rpm direct gearing, 48:48
- 1 motor intake
- 2 motor kicker
  - ↳ 2-11 watt motors
  - → taken off base, only used for skills

Purpose: This robot will have interchangable mechanisms to help during skills and matches. The robot will be six (6) inches tall to allow us to drive under the goals to descore. The interchangeable motors will help us have a fast drive base during matches while having a kicker for skills. Without a kicker during matches, we will use the robot to bowl the matchloads.

Iteration 5

Testing

(Continued)

Design Option Two: 6 inch Robot (Traditional)

Specifics:
- 6 motor base
  - ↳ 6-11 watt motors
  - → geared down to 450 rpm
- 1 motor intake
  - ↳ 5.5 watt motor
- 2 motor kicker
  - ↳ 11 watt plus 5.5 watt motor
  - → 200 rpm geared 12:36:60 for 40 rpm
  - → double slip gear
- 2 sets of wings, pneumatic

Purpose: This robot is similar to option one but with a slower base so we don't have to rearrange motors for skills runs. The robot will still be six (6) inches tall to allow it to go under the goal and descore triballs.

Design Option Three: Current Robot

Specifics:
- Make minor updates to current robot

Purpose: This robot functions well at driving, matchloading, elevating, and such. Only making minor improvements to this design will save us lots of time and give us more time to program and work on driver strategies.

Select Approach:

Goals for today ⟶ Choose a robot design that best meets the requirements and current Strategies of the game so we can begin building

We created a decision matrix of what aspects mattered the most on our robot. Below is the matrix and winning design.

Ideas	Speed	Size	Versitility	Skills	Time to Build	Total
6 inch (1)	5	5	3	5	3	21
6 inch (2)	4	5	5	5	4	23
Current	2	2	3	5	5	17

* Speed: how fast is this robot at driving, elevating, and scoring
Size: how big is the robot and easy to manuever around
Versitility: can the design switch strategies efficiently
Skills: how well does the robot perform in skills run
Time to Build: how long will it take to build the robot

Selected Design: 6 inch (2)

Explanation: We decided to go with the 2nd 6 inch Robot design because it will be simpler and faster to build and won't be as stressful during competitions since we won't have to switch motors on the subsystems.
This robot will let us use the kicker in matches and skills while having a fast base so it is more versitile.

We will begin building the new design, but keep the current robot assembled as much as possible.

* Break down of each criteria aspect we looked at. A rank of 0 (zero) means bad, and a rank of 5 (five) is good.

Utilizing Old Robot:

Goals for today → Find a use for the old robot

→ Find jobs and ways for Owen and Harris to practice

Due to limited time before States and our decision to build a new robot, we will be keeping the current robot assembled for the following reasons.

1.) Owen can practice different skills strategies with the current robot while Eli is building.

2.) Harris can use the current robot to practice PID and programming functions as well as mapping strategies.

3.) In the chance that Eli can not finishing building a new robot, we will still have the old robot with autons and that Owen is familiar with to take with us to States instead.

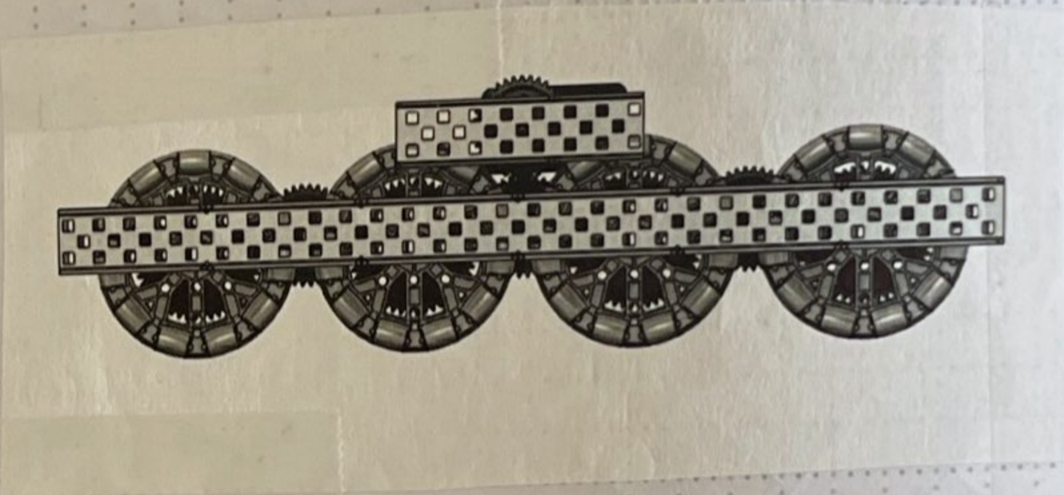
VP

Initial CAD Overview of Base:

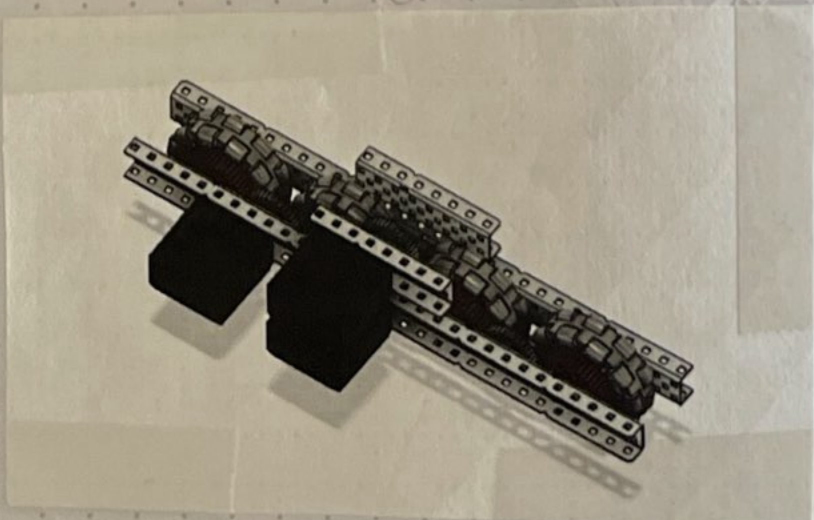
Goals for today → Create a rough CAD sketch of the base to help work out the spacing and design

Below is the CAD of our new base design that Eli created. It has several different angles and includes the c-channels, gears, wheels, and motors. When we go to build the base we will add the bearing flats, spacers, and screws.

Outside, Side View:



Inside, Above View:



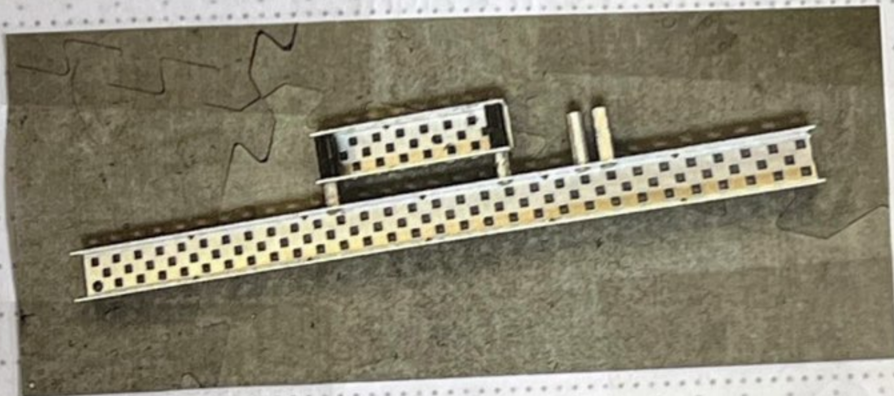
Iteration 5

Testing

Building Base Structure:

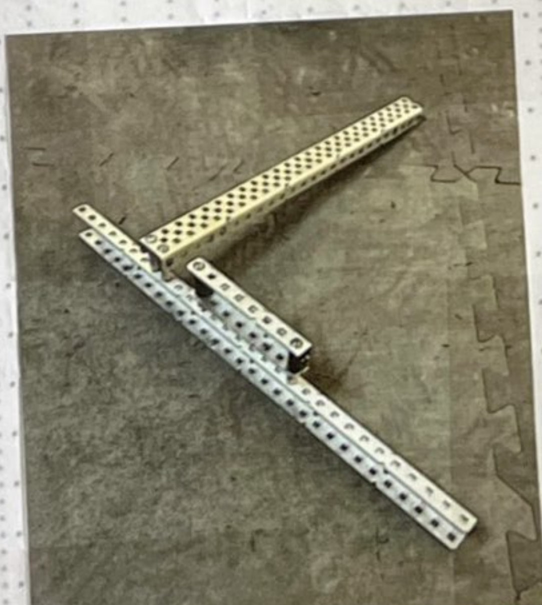
Goals for today → Gather parts and begin building the base

Below are the steps we took with images of what we were able to complete on the base structure today.



① Gather/cut two (2) eight (8) hole long aluminum c-channels and two (2) thirty (30) hole long aluminum c-channels.

② Attach an 8 long c-channel to the top of a 30 long c-channel twelve (12) holes from the back using two (2) 0.500 inch ~~spacers~~ standoffs with spacers in the 8 long c-channel for bracing. Use two (2) 0.250 inch screws and two (2) 1.000 inch screws.
Repeat for a second time on other two (2) c-channel.

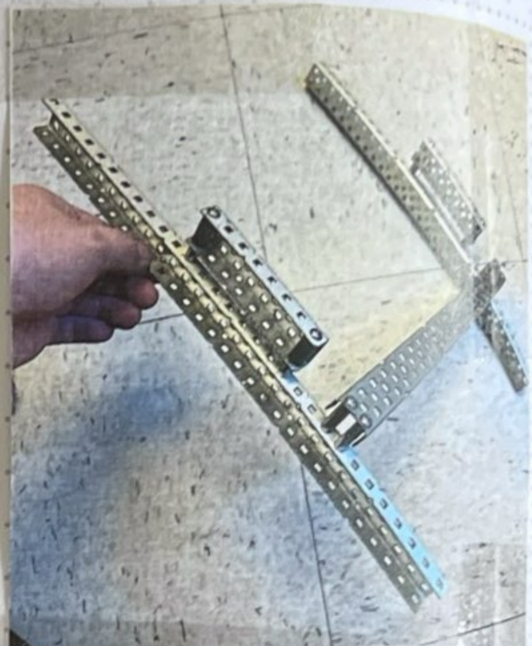


③ Add two (2) 1.000 inch standoffs to each c-channel two (2) holes over from the 8 long c-channel on the 30 long c-channel for a total of four (4) standoffs. Use four (4) 0.250 inch screws.
Depicted above.

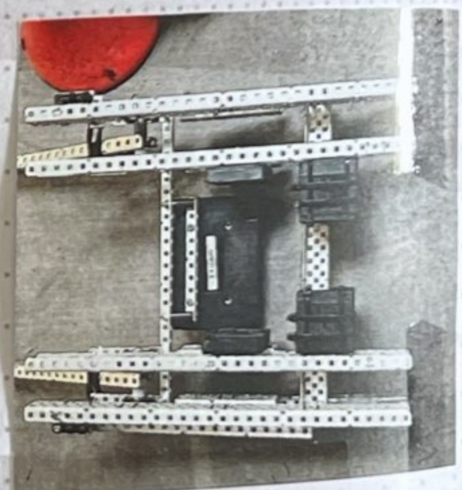
④ Gather/cut a twenty-five (25) hole long aluminum c-channel and attach to standoffs from step 3 using two (2) 0.250 inch screws.

(Continued)

⑤ Repeat step 4 for the other thirty (30) long C-channel on the other end of the twenty-five (25) long C-channel mirrored. Image of complete Step 5 to the the right.

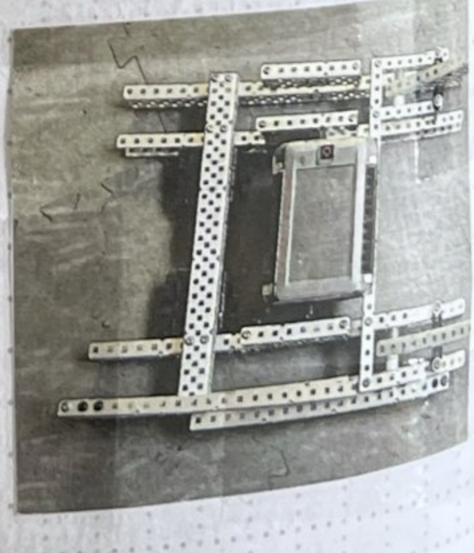


⑥ Gather/cut a twenty-five (25) hole long aluminum 1by1 L-channel. Attach to the base using four (4) screws and nuts ten (10) holes back from the front of the base.



⑦ Attach a ten (10) hole long aluminum 1by1 L-channel to the metal from step 6 using two (2) 1.000 inch screws and three (3) 0.375 inch spacers each for a total of six (6) and two (2) nylock nuts.

⑧ Attach ~~the~~ Brain to the ten (10) long 1by1 L-channel using two (2) 0.250 inch screws. There are two pictures to the right of the brain attached to the base structure from above and below.



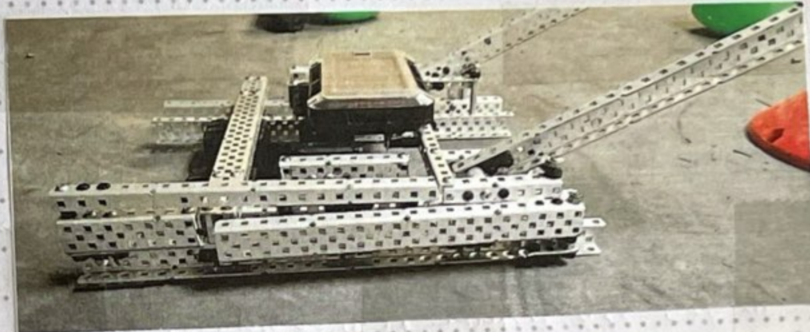
VP

Testing

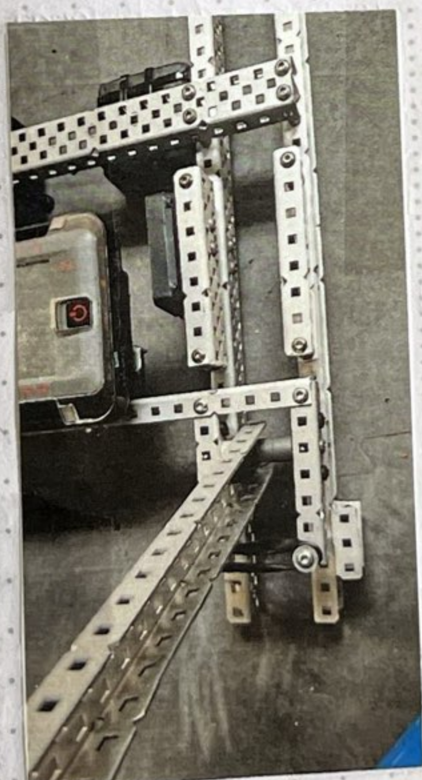
Building Intake Structure:

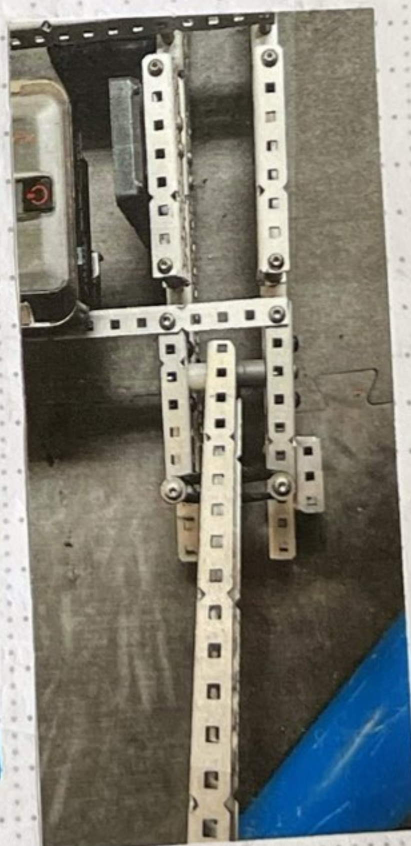
Goals for today → Get the structure of the intake built and attached to the base

Since we have to wait on an order that includes bearing flats, screws, and spacers to finish the base, we decided not to waste time and begin building the intake structure.
Below are the steps we took to build and attach the structure.



1. Gather/cut two (2) twenty-four (24) hole long aluminum c-channels.
   Gather/cut four (4) seven (7) hole long aluminum 1by1 L-channels.

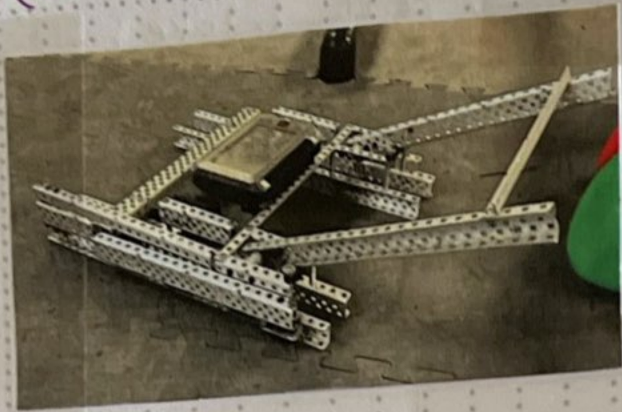




2. Attach the L-channels parallel to the base c-channels underneath the L-channel holding the brain.
   Attach a bearing flat to the inside of each L-channel one hole from the back with the screws on the outside.
   Attach a bearing flat to both of the c-channels at the very ends in the inside using two (2) screws each with the last hole empty.

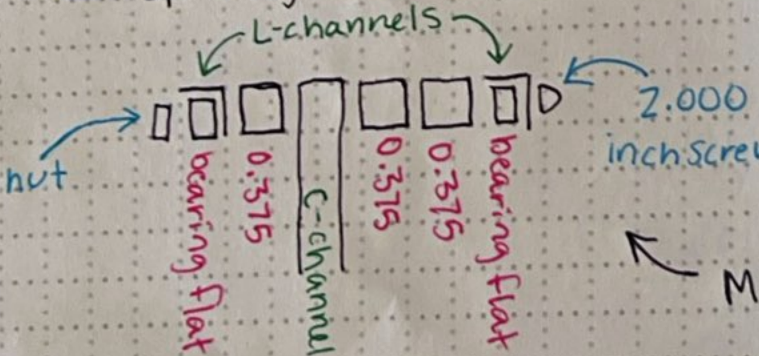
Above are pictures of these steps and future steps.

(Continued)



③ Attach a 0.5 inch standoff, teflon washer, and two(2) 0.250 inch screws to the front of each L-channel for a total of 4. Connect each side with a rubberband to let the intake rest against while also being able to lower under the goal.

④ Add the below diagramed spacing between the L-channels.



Mirror for Left Side

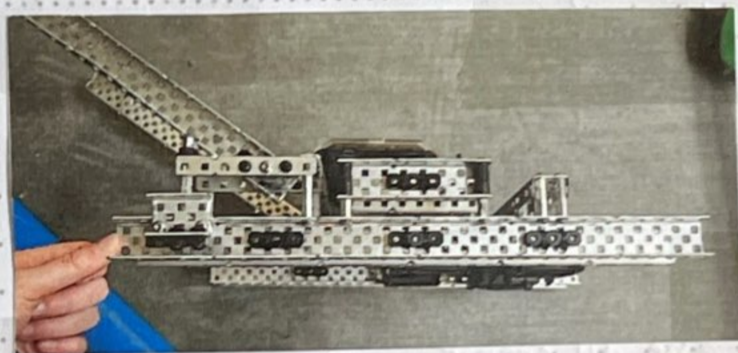
This screw pivots up and down to allow the intakes to slid under the goals and pick up triballs.
Above are pictures of the intake structure.

Testing

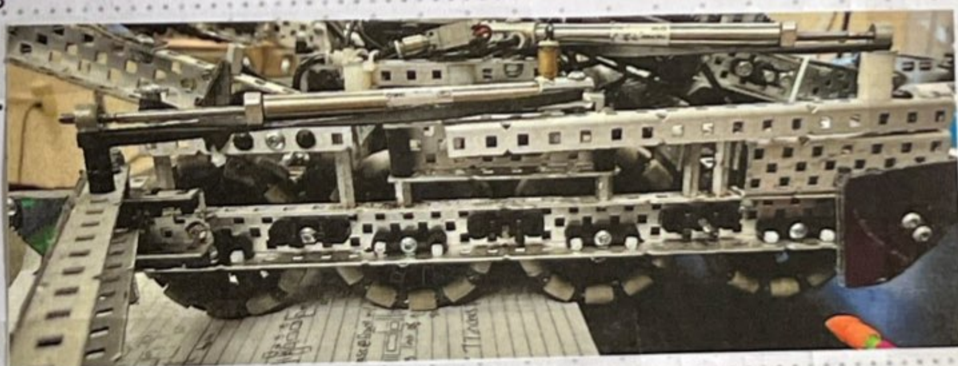
Adding Wheels to Base:

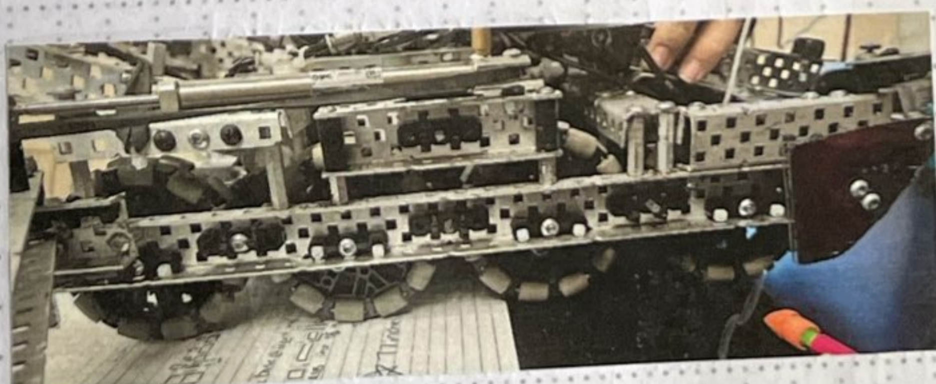
Goals for today → Get the wheels, gears, spacers, and bearing flats added to the base

We got the parts we needed to finish the base so our plan today is to finish the base. Below are the steps we took as well as diagrams and pictures to show everything.

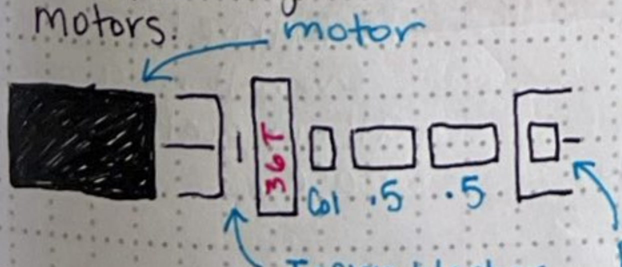


① First we added the bearing flats to the outside c-channel in the depicted locations to the right. There were eight (8) bearing flats on each side for a total of sixteen (16). The bearing flats are attached with two (2) zipties each.





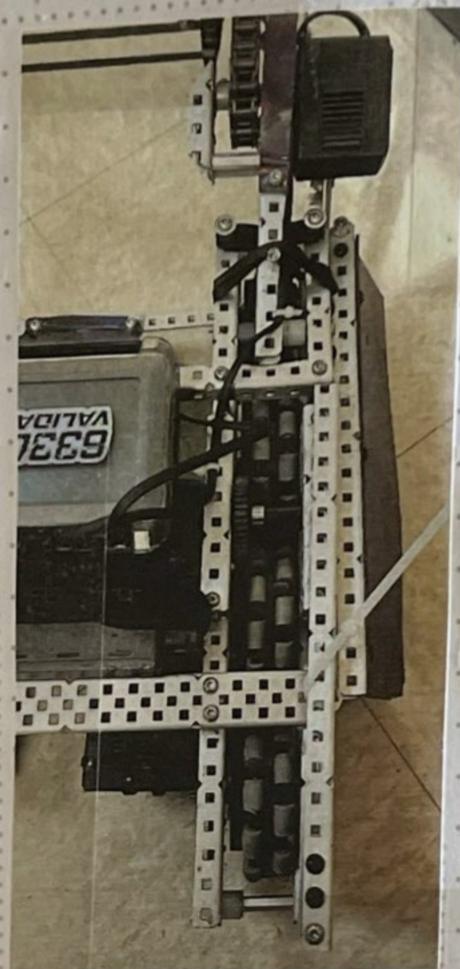
② Next we added the 36 tooth gears to the base with the spacing below. Six (6) of these spacings go in motors while two (2) don't go in motors.

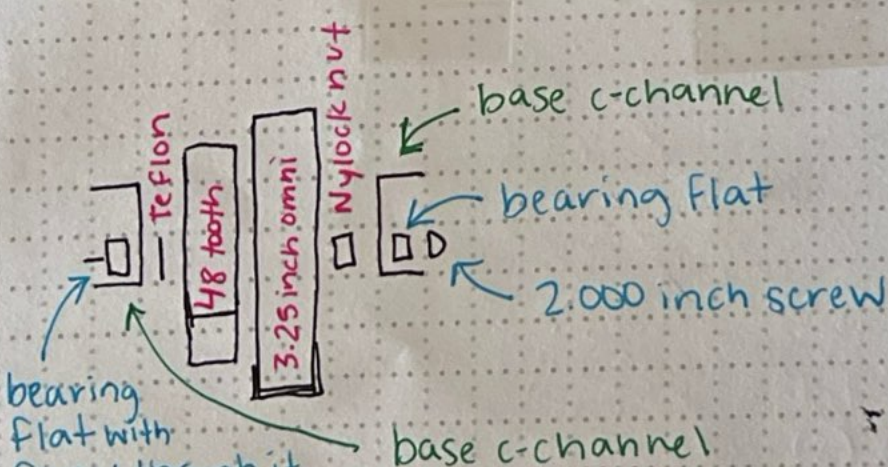


The one without a motor uses a screw instead of an axel. One of the motors/gears is stacked on top of another.

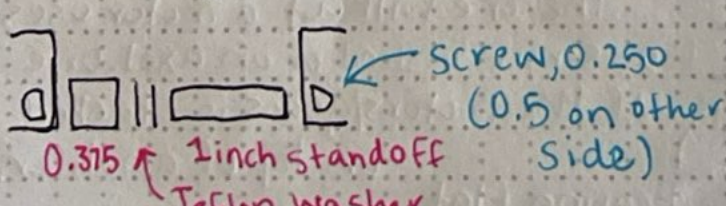
(Continued)

③ Next we added the wheels and spacing between the gears on the base. Below is the diagram. We repeated 4 times each side for a total of 8 wheels. The wheels are ~~are~~ all omnis to help with turning. To the right are pictures from above.





④ We strengthened the base by adding standoffs to both ends.



Results:

Once this base was done and tested, we found that it can travel 77 inches per second and has a 90 rpm boost compared to the last drive base.

Overall we completed all our goals for the base.

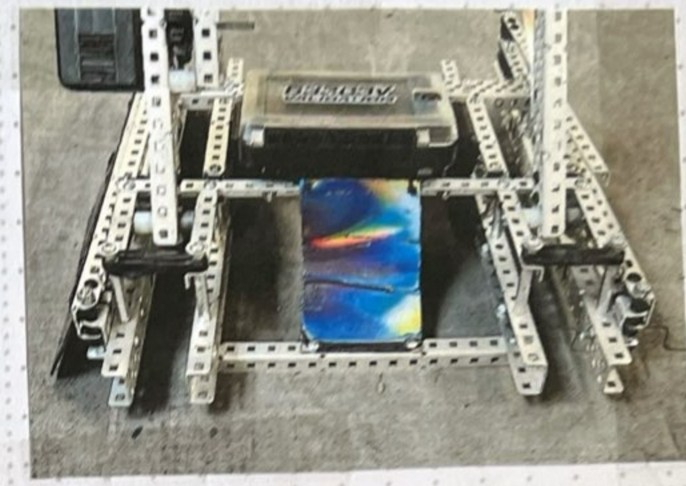
Testing

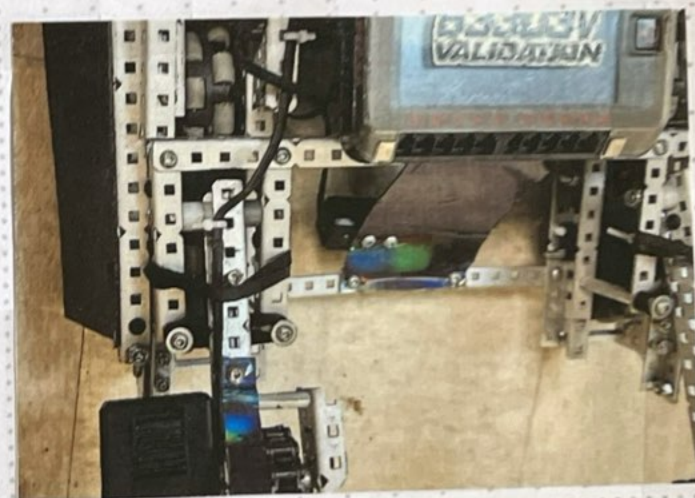
Building Ramp:

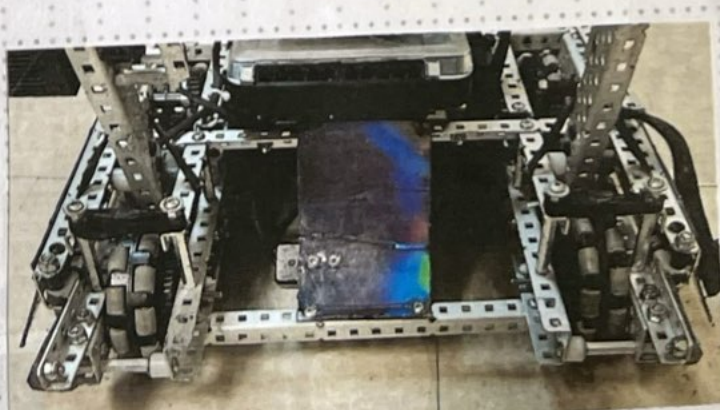
Goals for today → Attach a ramp under the intakes to help hold the triballs in

To the right are images of the ramp built and attached to the robot. Below is also a parts list.

The L-channel is screwed in under the base four (4) holes back. The Lexan is curved and ziptied at the top while screwed in at the bottom. Then the radio was screwed into the Lexan underneath the ~~radio~~ ramp.







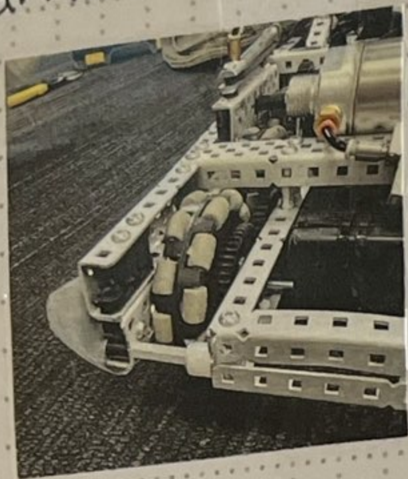
Ramp Parts List:

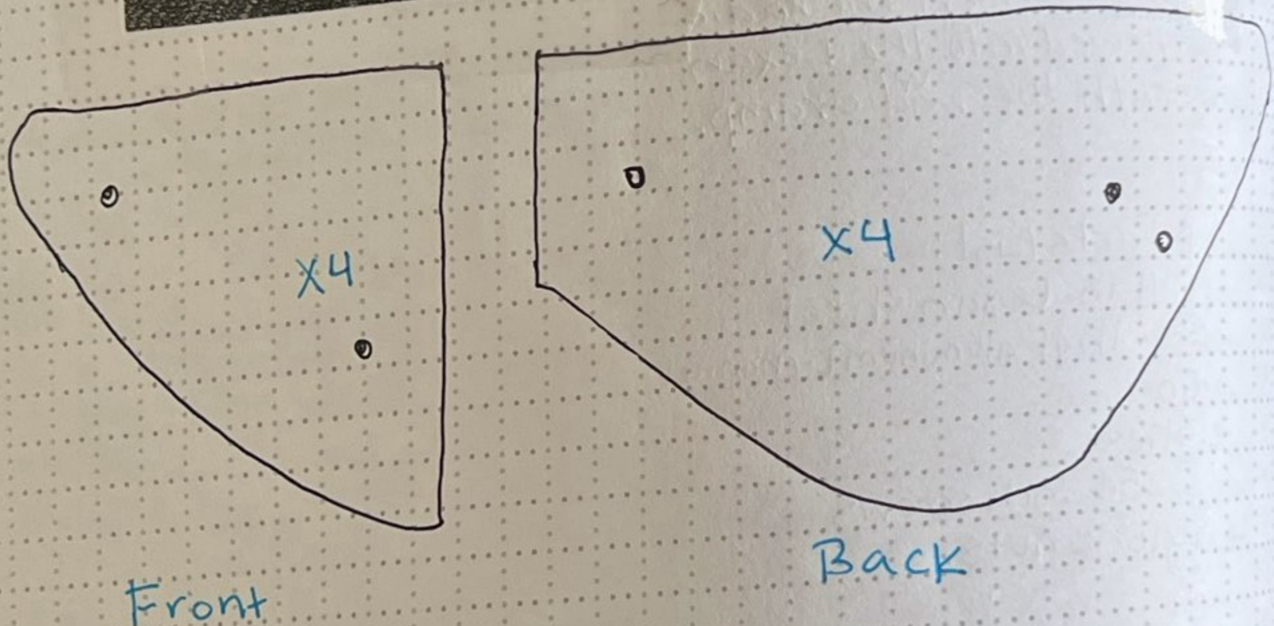
- 3 by 4 in Lexan sheet
- 16 long 1by1 aluminum L-channel
- Radio
- 2 zipties
- 6-0.250 inch screws
- 4 nylock nuts.

VP

Adding Sleds to Base:

Goals for today → Add Sleds to the Base so it can drive over the barrier





Above are images and templates of the front and back Lexan sleds. We put two(2) on each side to help strengthen them while being rammed constantly during matches.

Sleds on the front and back allows for faster movement over the barrier so we can go either direction and save time.

VEX Tennessee State Championship: 63303V Schedule

TENNESSEE STATE CHAMPIONSHIP

2/19/2024:
Harris: Work on programming for previous robot
Owen: Run practice skills runs with a goal of 200.
Eli: Get majority of the kicker finished as well as the wings duplicated.
Vanessa: Work on the notebook.

2/20/2024:
Harris: Work on programming for previous robot
Owen: Run practice skills runs with a goal of 200.
Eli: Finish kicker, and start plus finish intake
Vanessa: Work on the notebook.

2/21/2024:
Harris: Work on programming for previous robot
Eli: Work on battery placement and wings.
Vanessa: Work on the notebook.

2/22/2024:
Harris: Work on programming for previous robot
Owen: Run practice skills runs with a goal of 200.
Eli: Finish drive base and start wiring
Vanessa: Work on the notebook.

2/23/2024:
Harris: Work on programming for previous robot
Owen: Run practice skills runs with a goal of 200.
Eli: Finish wiring and work on finishing wings.
Vanessa: Work on the notebook.

2/24/2024:
Harris: Work on programming for previous robot
Owen: Run practice skills runs with a goal of 200.
Eli: Work on pneumatic systems and hang
Vanessa: Work on the notebook.

2/26/2024:
Harris: Work on programming for previous robot
Owen: Run practice skills runs with a goal of 200.
Eli: Finish pneumatic systems and hang
Vanessa: Work on the notebook.

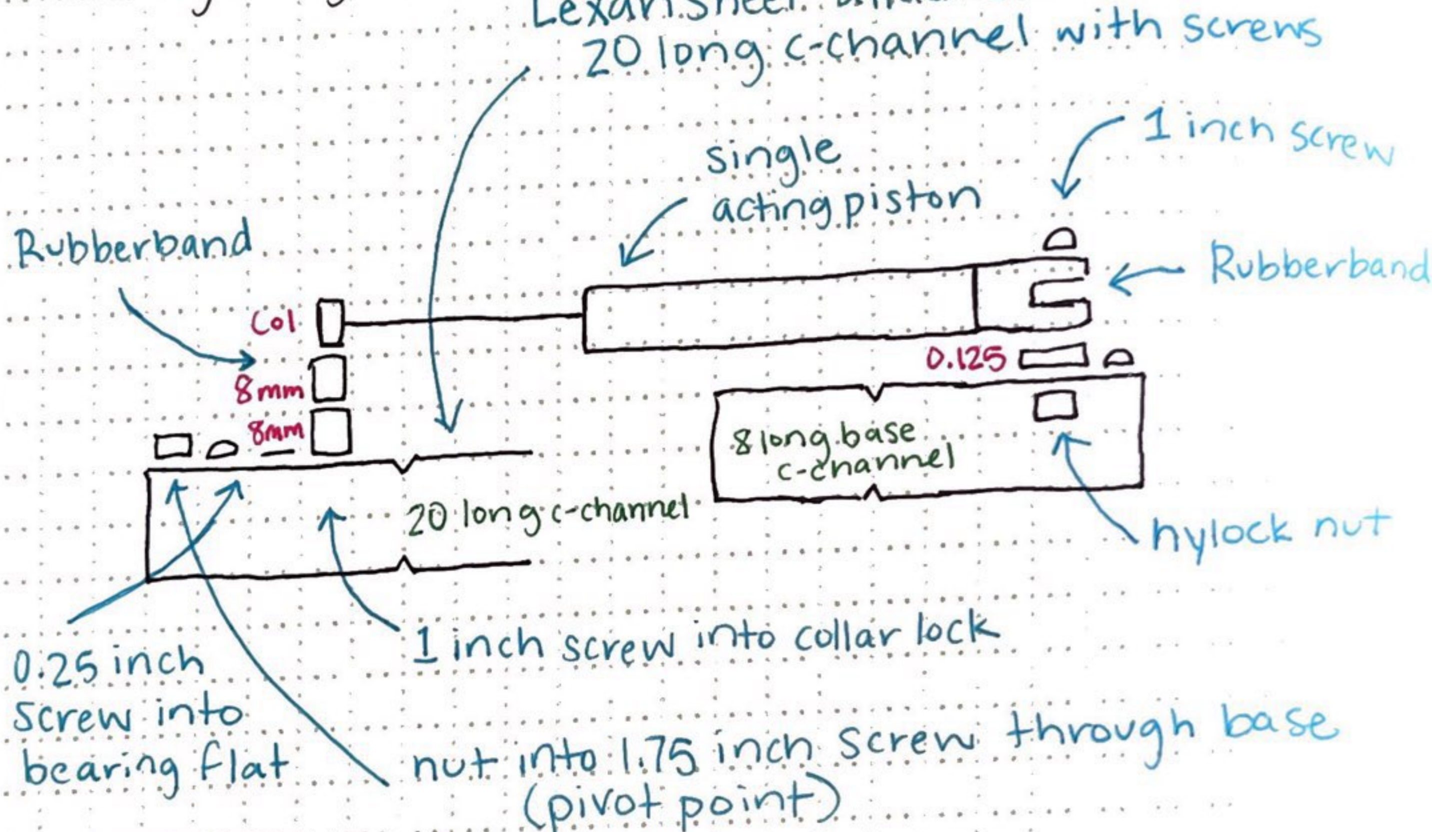
2/27-29/2024:
Harris: Work on programming new robot
Owen: Practice driving new robot

Testing

Building Front Wings:

Goals for today → Get the front wings built and attached to the robot.

Below is the diagram for how we built these horizontal folding wings. Mirror and repeat.



Parts List:

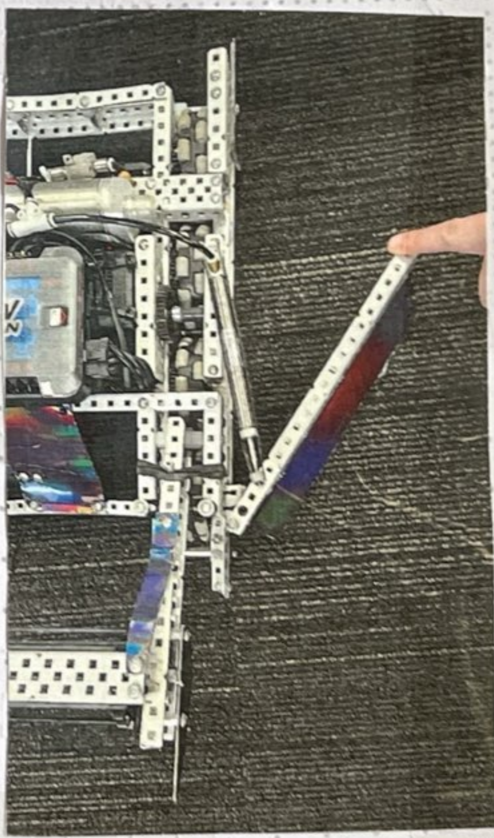
- 2-20 long aluminum c-channels
- 2-9.5 by 3 inch Lexan sheets
- 2-single acting pistons
- 2-0.125 inch spacers
- 4-8 mm spacers
- 8-nylock nuts
- 2-Collar locks
- 2-0.250 inch screws
- 4-1.000 inch screws
- 2-1.750 inch screws
- 2-rubberbands

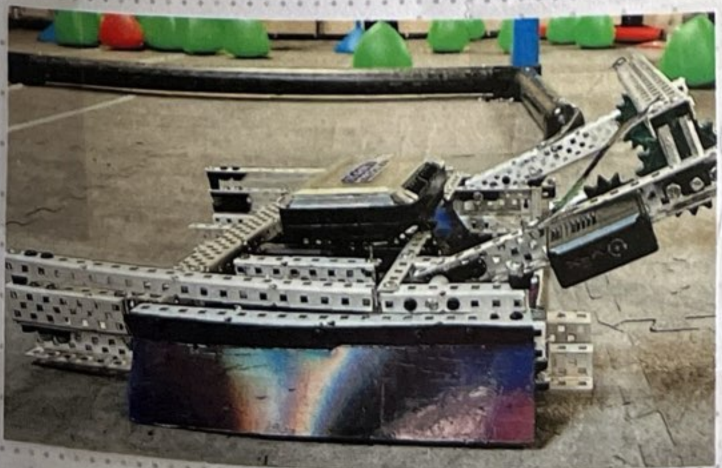
To the left is a complete parts list to build both wings and attach them to the robot.

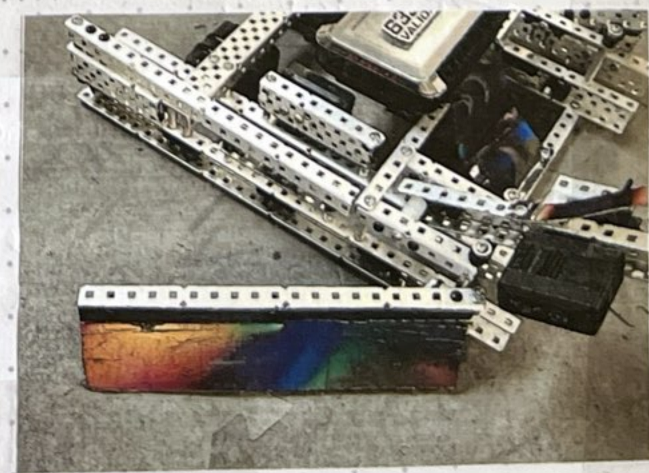
Testing

(Continued)

Below are images of the complete wings from several angles attached to the sides of the base.
We tested them and they work well to score triballs, push triballs over the barrier, and they fold when we hit a wall to prevent getting stuck.



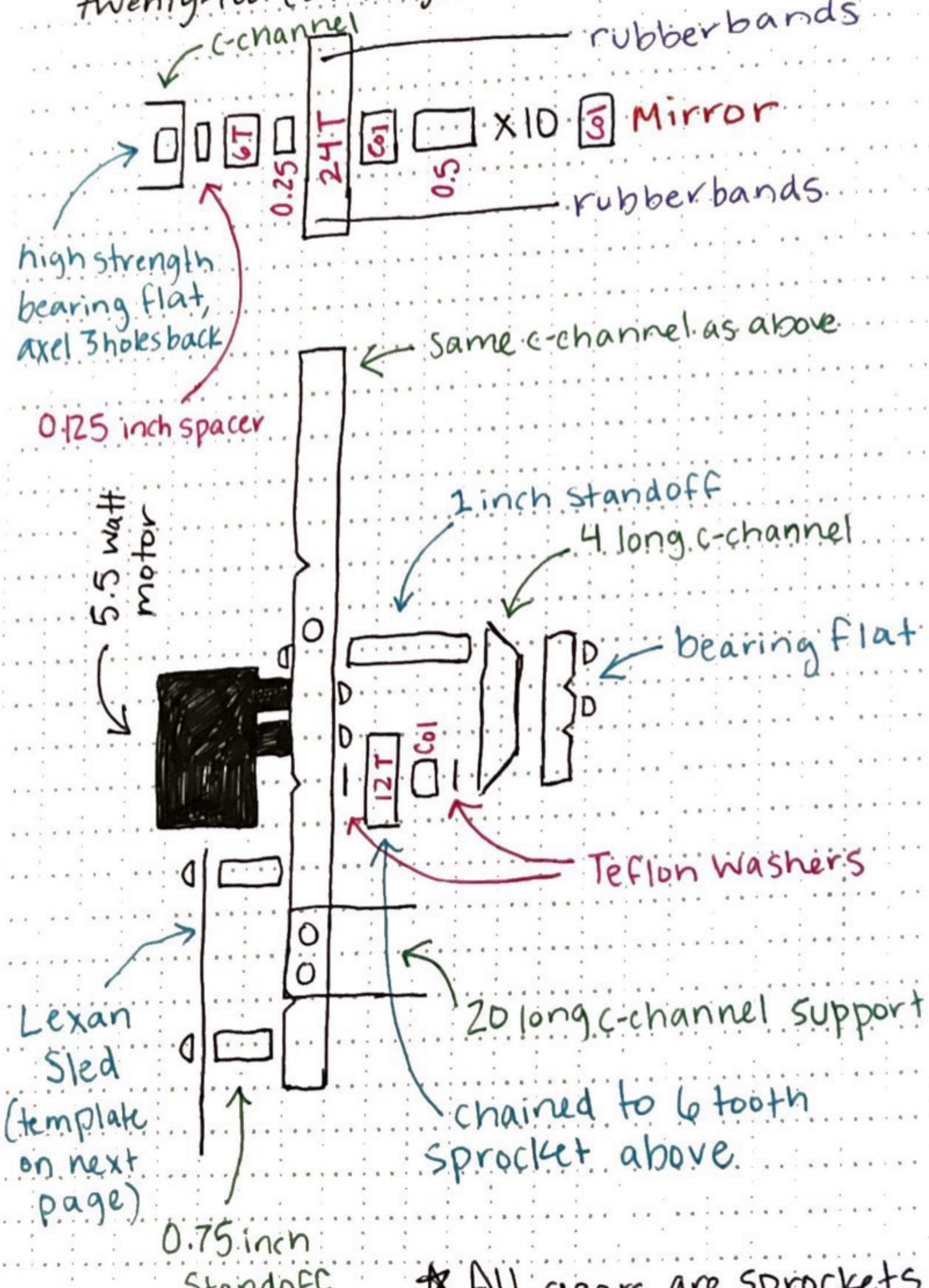




Finishing the Intake:

Goals for today → Get the gears, spacing, and motor attached to the intakes so they are working

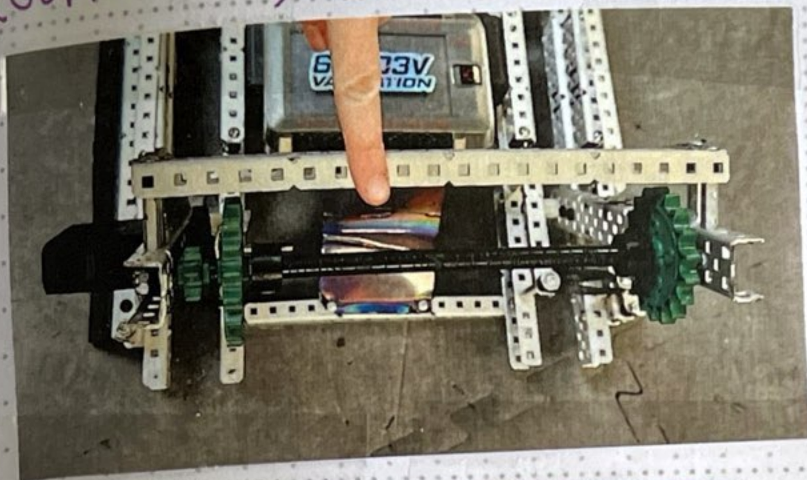
Below are the diagrams for finishing the intake. We cut the twenty-four (24) long c-channels down to seventeen (17) to begin.

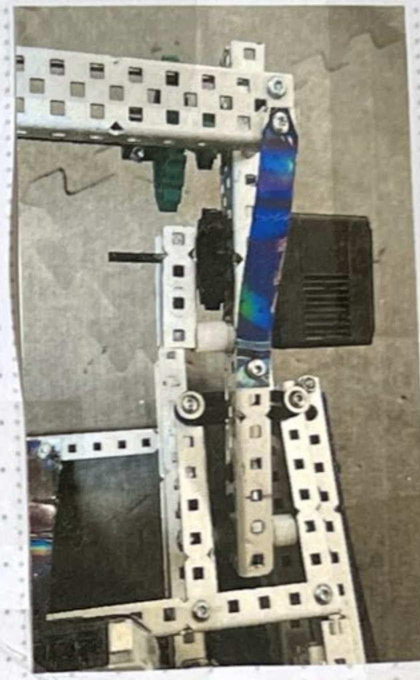


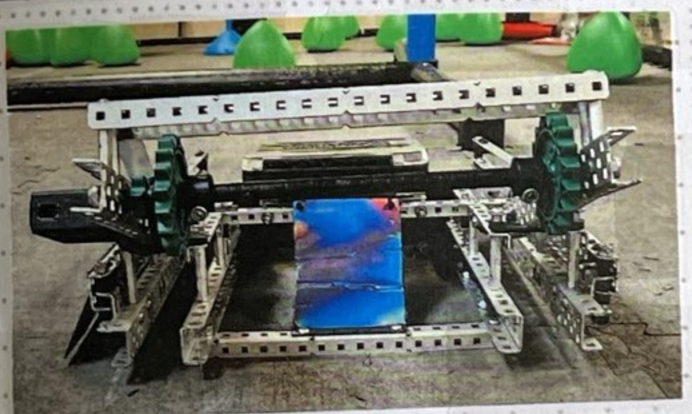
☆ All gears are sprockets above.

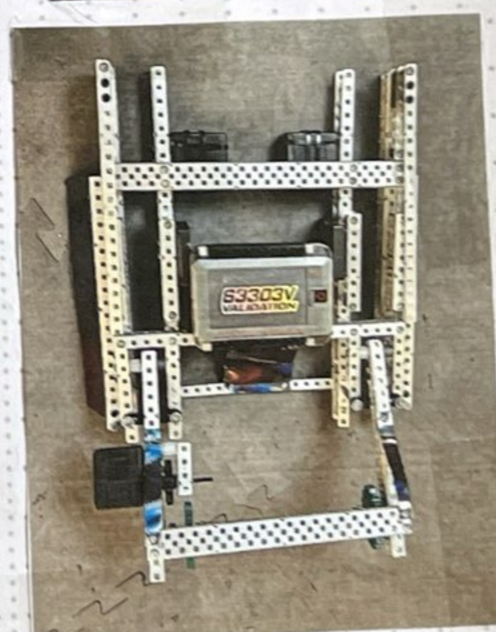
(Continued)

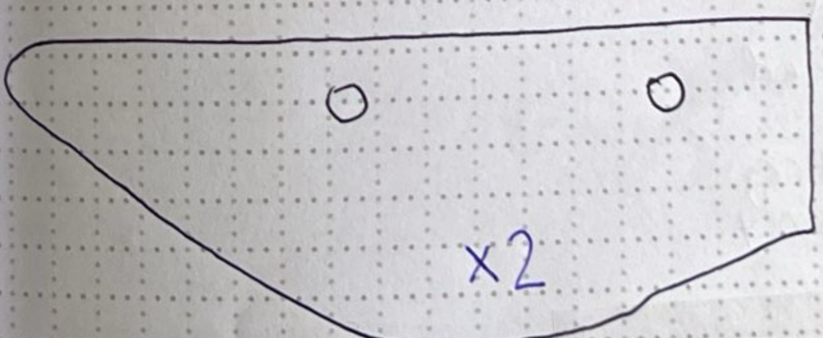
Testing





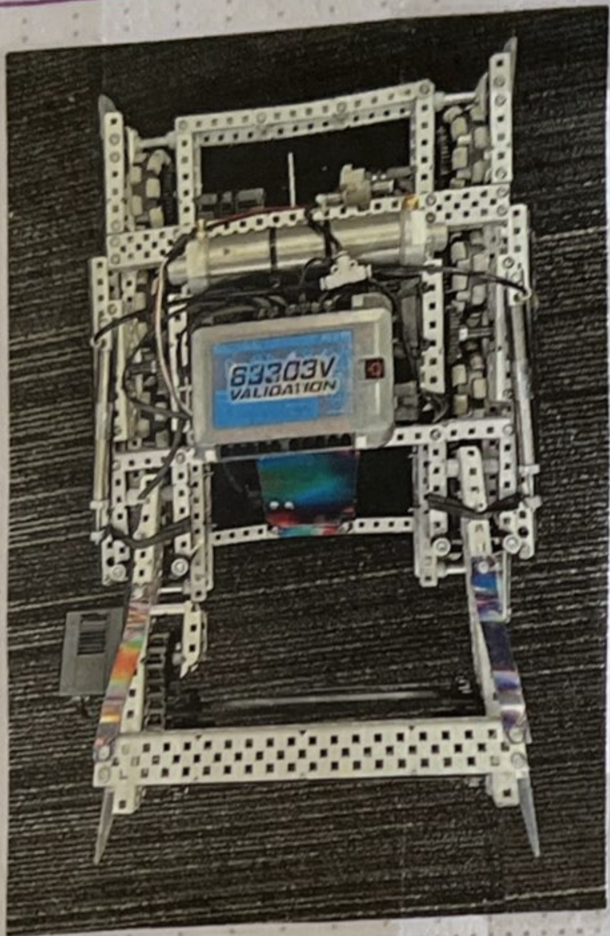


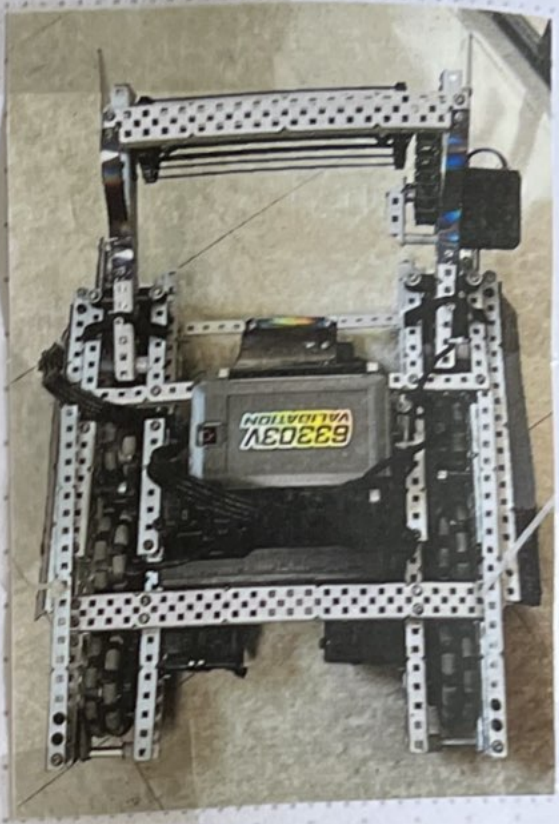


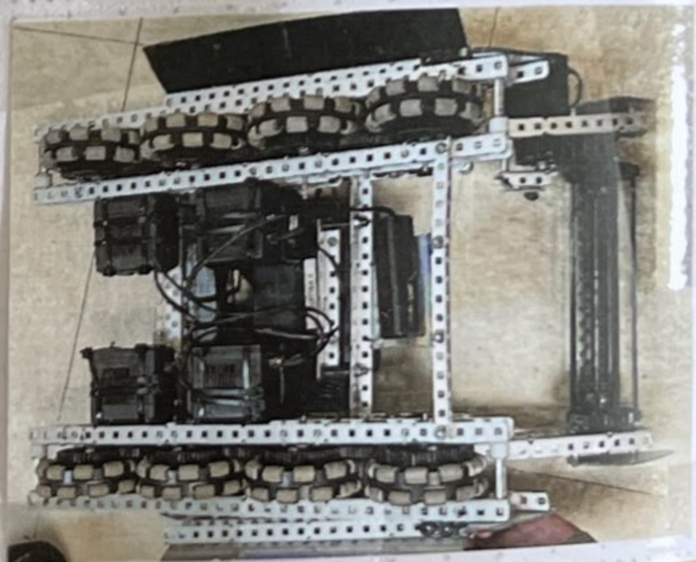


Above is a template of the sleds with pictures of the intake being built. We also added two(2) Lexan strips from the 20 c-channel to past the motor to help it under the goal.

Finished Intake Images:

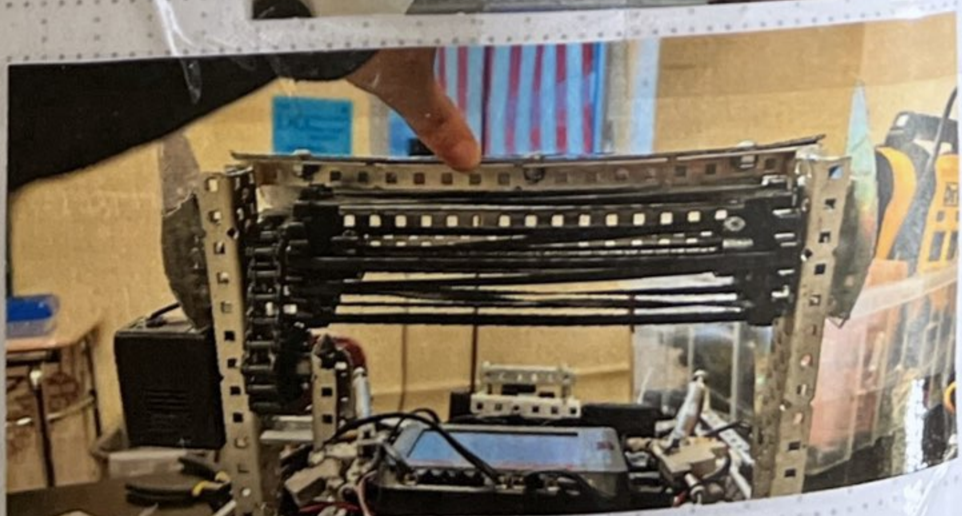






Above and to the right are images from several angles of the finished intake.
Next we will be testing the function and efficiency.

VP



Testing

Testing the Intakes:

Goals for today → Test the Intakes function
→ Make Adjustments as needed

Testing
The intakes hold triballs well unless the intakes bounce upward since they pivot.

Problem/Challenge
The Intakes have nothing to hold them downward.

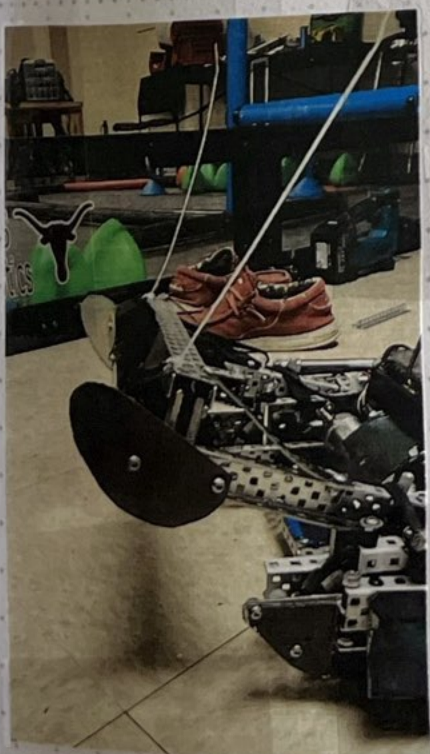
Brainstorming
1. Tighten the pivots
2. Rubberband it down
3. Place a stopper on top

Select Approach
2
Rubberband it down to keep pressure on triballs.

Implement
Ziptie rubberband to L-channels, on intake, on top of intake.

Test Again
The Intake nolonger bounces up, but the sleds get stuck on the goal.

Problem/Challenge
Intake Sleds get caught on the goal.



Brainstorming
1. Reshape sleds
2. Remove sleds

Select Approach
1
Reshape to be shorter.

Implement
New Sleds to the left.

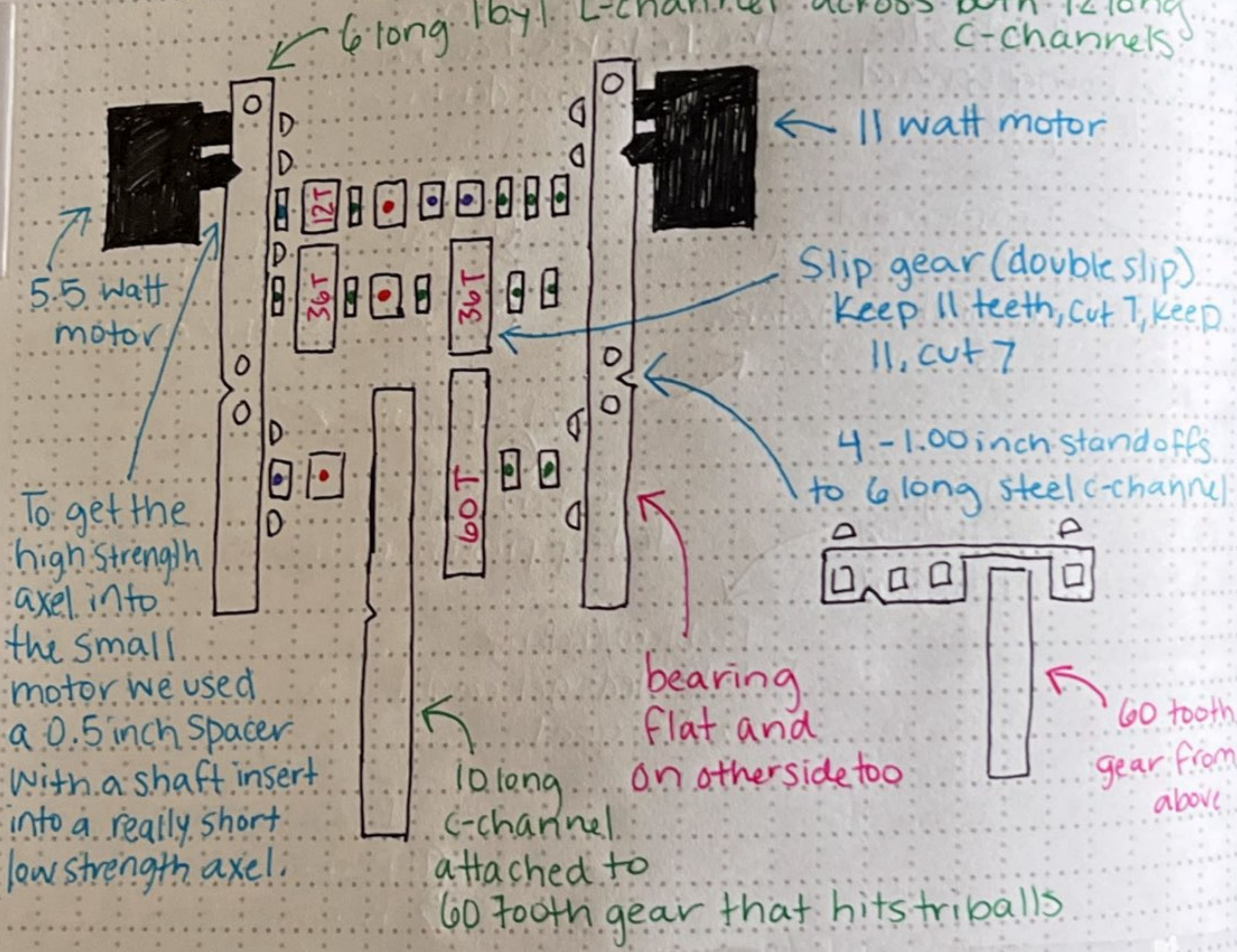
Test Again
Works very well and doesn't get caught.

Project Testing the Intakes
Date 2/22/2024
Name Vanessa Perkins

Building the kicker:

Goals for today → Build the kicker off of the robot to attach later

Below is a diagram for how to build ~~k~~ kicker.

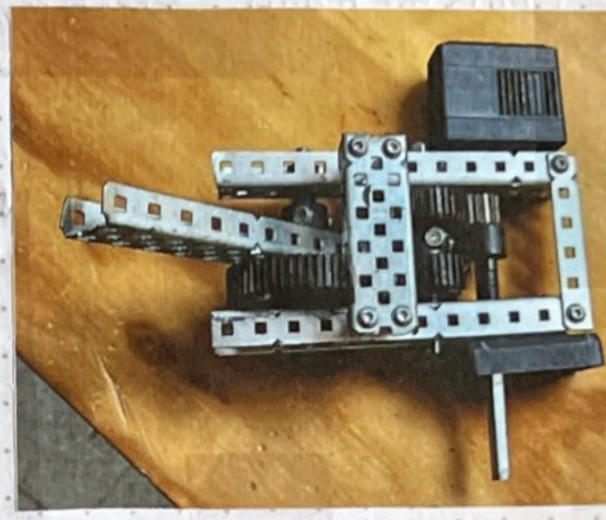


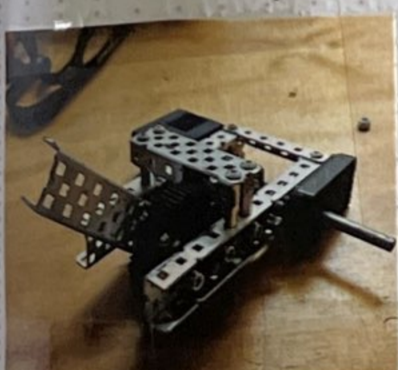
Spacing Parts List:

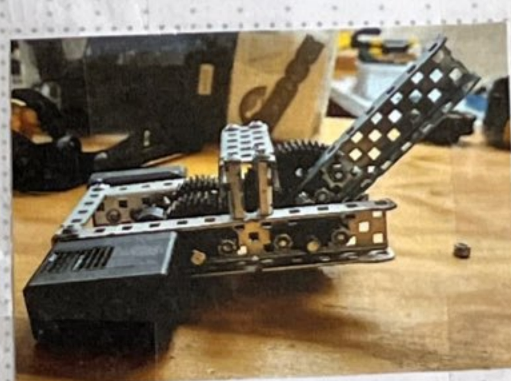
- 0.500 inch high strength spacer •
- 3 - high strength collar locks •
- 11 - 0.125 inch high strength spacers •
- 3 - 0.250 inch high strength spacers •

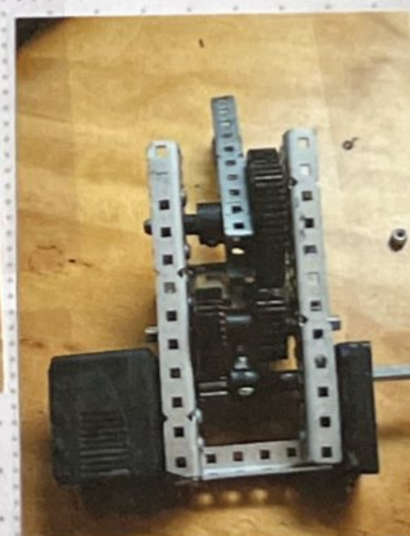
* Colored dots correspond to diagram above

(continued)









Above are images of the kicker completely built minus the cartridge and motor cover because we needed to trim the high strength axel.

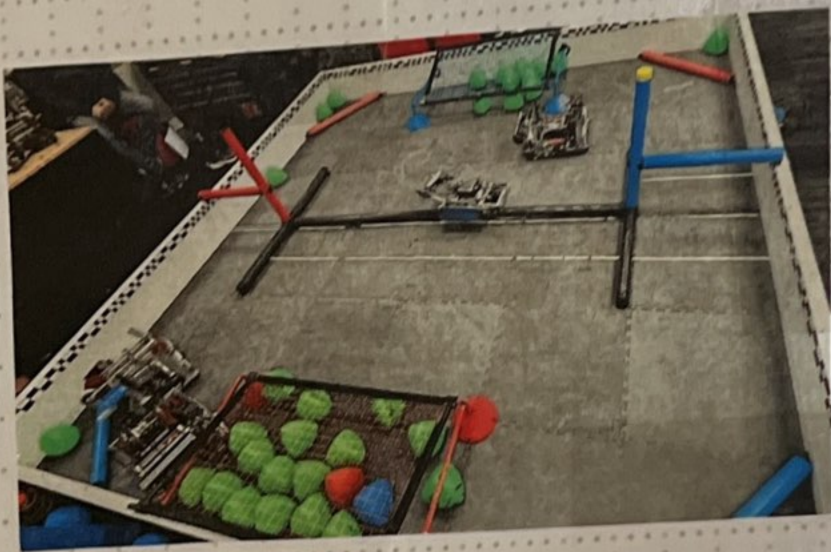
VP

Project Building the Kicker (continued) Name Vanessa Perkins
Date 2/23/2024

AGSI Scrimmage:

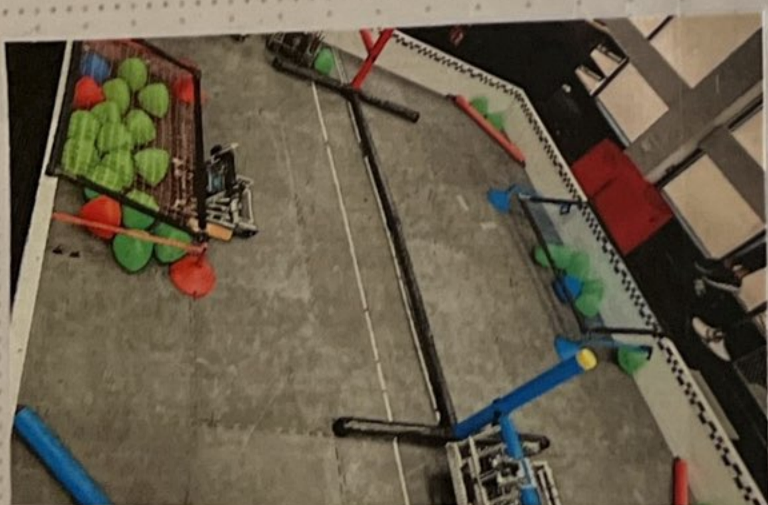
Goals for today → Test our new robot
→ Strength alliances with other teams

We took our old robot and new 6 inch robot to a Scrimmage with the 57711 teams. Our goal was to drive both and decide which we would take to States with us. Below are some notes from our matches together with 57711X and some middle school teams. There are also pictures from the end of each match we ran.



Match 1:

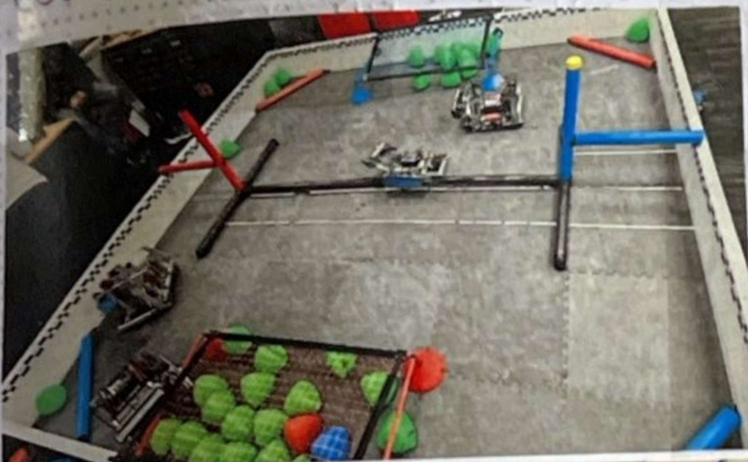
- Allianced with Rocky, 57711X
- Bowled triballs too much so the defensive team stopped us and scored them all which we can fix by pushing the matchloads in the corners to prevent the other team from stealing them.



Match 2:

- Same Alliance
- Showed that our old robot is very slow and not as efficient at intaking and driving over the barrier.

(Continued)



Match 3:
- Same Alliance
- Continued to affirm that the new robot is better.

Match 4:
- Same Alliance
- the robot has several improvements that can be made including covering the intakes to prevent them from being entangled as happened in this match.

Conclusion:
This scrimmage helped affirm our decision to take the new six (6) inch robot to States as well as find the needed improvements and work on match Strategies.
We were thankful for AGSI hosting the scrimmage and our opportunity to strengthen alliances.

VP

Project AGSI Scrimmage (Continued) Name Vanessa Perkins
Date 2/24/2024

Redesign Front Ramp:

Goals for today → Improve the Front Ramp after the AGSI Scrimmage

Problem Challenge: The Ramp got bent very easily at the Scrimmage and was took weak to withstand defence.

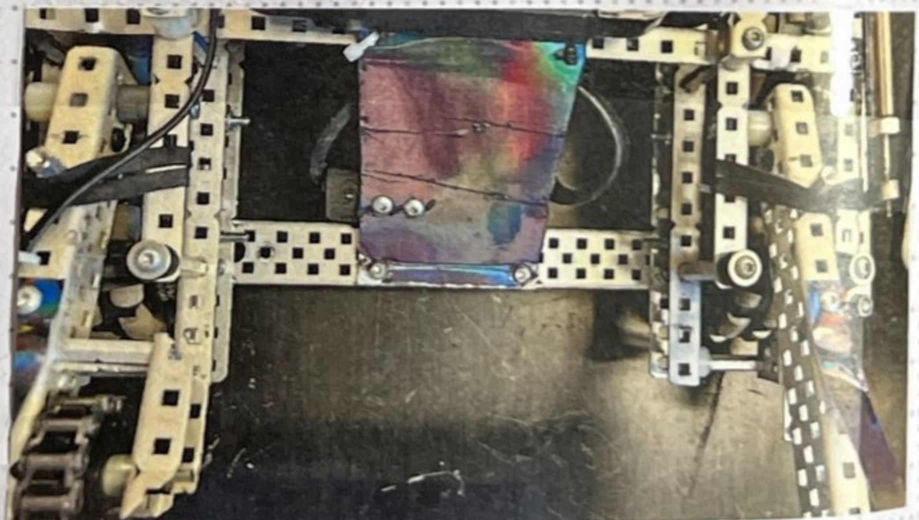
Brainstorming: Options including adding two(2) 1by1 L-channels, using steel, or using an aluminum c-channel.

Select Approach: We chose to go with a c-channel because it is the strongest while still light weight option.

Build/Implement Solution: Below are images of the new sixteen(16) hole long c-channel attached the same way as the old ramp.

Test: This ramp is much stronger and sturdier to withstand play.

VP

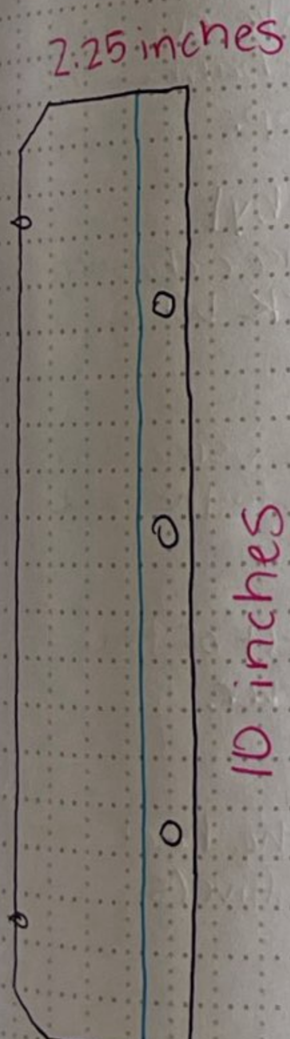


Improving Intake:

Goals for today → Analyze our scrimmage performance and improve intakes as needed

The Scrimmage showed us that our intake needed better protection to strengthen it and prevent our rubberbands from being entangled and make bowling off the front easier. We made the following changes to improve these aspects.

1.) Attach twenty (20) long 1by1 L-channel to top of intake with screws in the front.



2.) Cut out Lexan Sheet as diagramed to the Left with a 45° degree bend on the blue line.
Ziptie to the top c-channel as depicted.
Use 0.500 inch spacers and screws to attach bottom of the Lexan to the newly added 1by1 L-channel.

When tested, this set of improvements worked very well.

Redesign Back Base Bracing:

Goals for today → Strengthen the bracing on the back of the base

The base previously only had a 1by1 L-channel across the back to connect the base however the scrimmage bent it up.
We replaced this L-channel with better bracing by following the below steps.

1.) Replace L-channel with a sixteen(16) hole long aluminum 1x3 c-channel using screws.

2.) Round the corners of the c-channel. Make sure bottom of c-channel is level with base c-channels and there are two(2) 8mm spacers on top.

3.) Gather/cut two(2) six(6) hole long 1by1 L-channels. Attach at angles to base c-channels on both sides 2 holes in from the back using 0.500 inch spacers and screws.

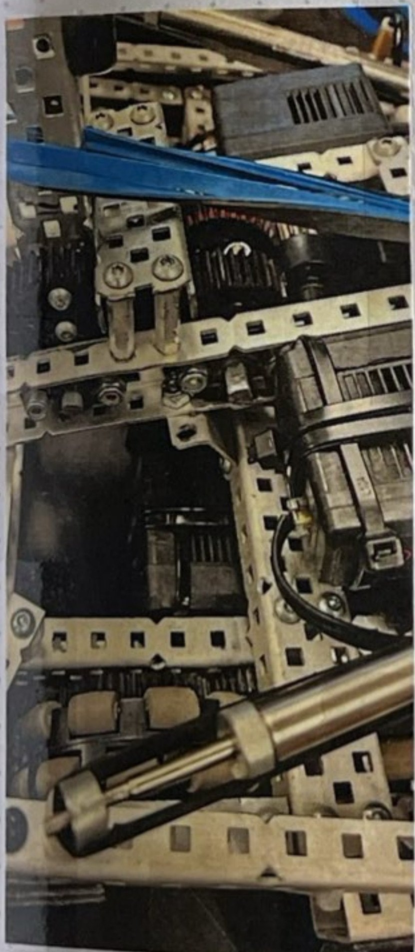
4.) Attach angled L-channels to outside using two(2) collarlocks and couplers attached to the back Lexan sleds through holes drilled in.

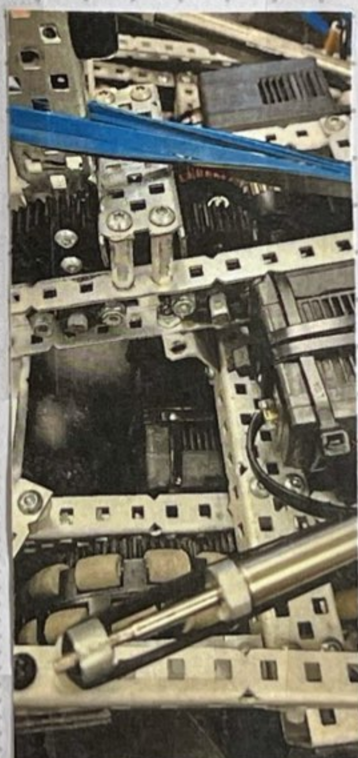
5.) Cut out 9th and 10th top holes from left of the 1x3 c-channel to prepare for the kicker. Bend upward slightly.

6.) Attach a Lexan rectangle sheet to the back c-channel and L-channels using five(5) screws and nylock nuts.

Attaching Kicker:

Goals for today → Find a way to attach the kicker to the robot





To the left are images of how we connected the kicker.

It needed to be in the back of the robot and angle upward so we placed it behind the brain.

The end of the kicker was screwed into the bent upward ends of the 1X3 c-channel from step 5 on page 106, book 3.

The center of the kicker was braced using the above depicted angle brackets and screws.

VP

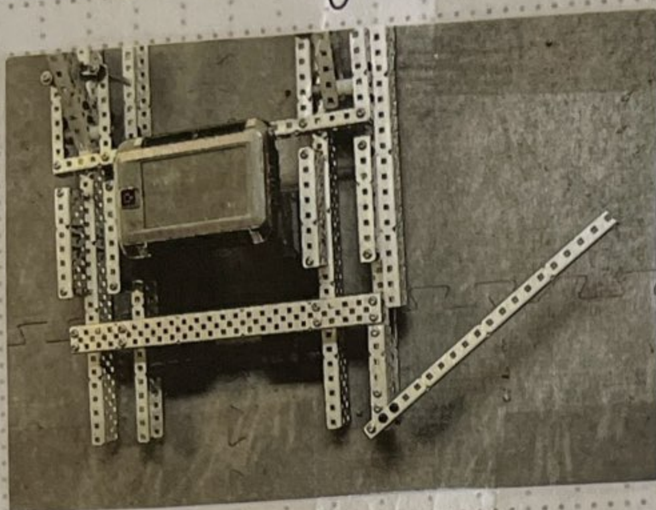
Project Attaching Kicker

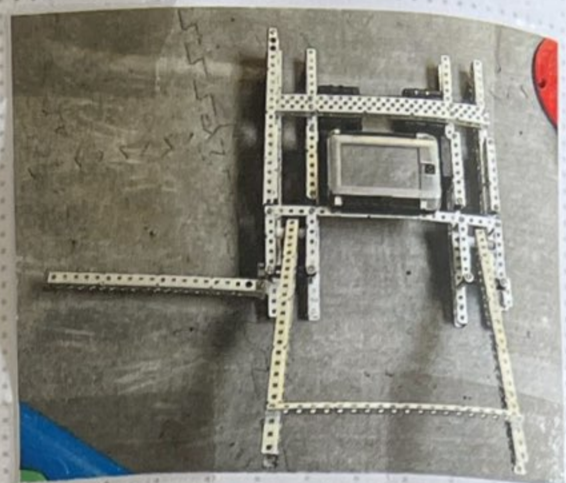
Date 2/27/2024

Name Vanessa Perkins

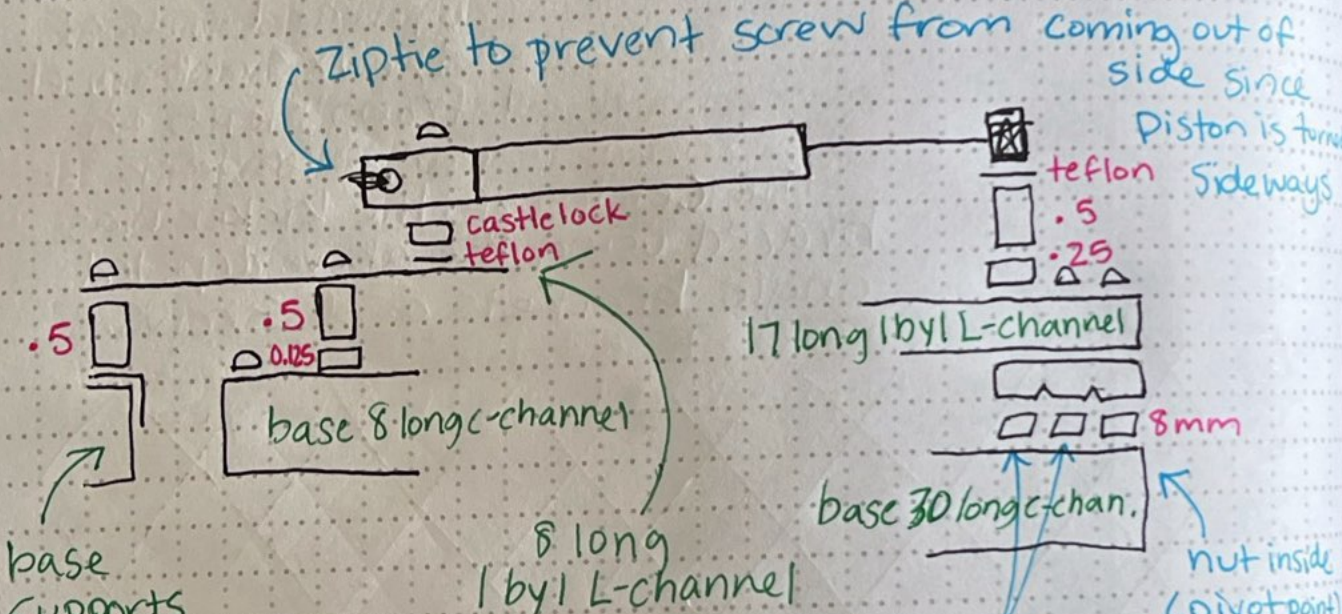
Building the Back Wings:

Goals for today → Get the back wings built and running





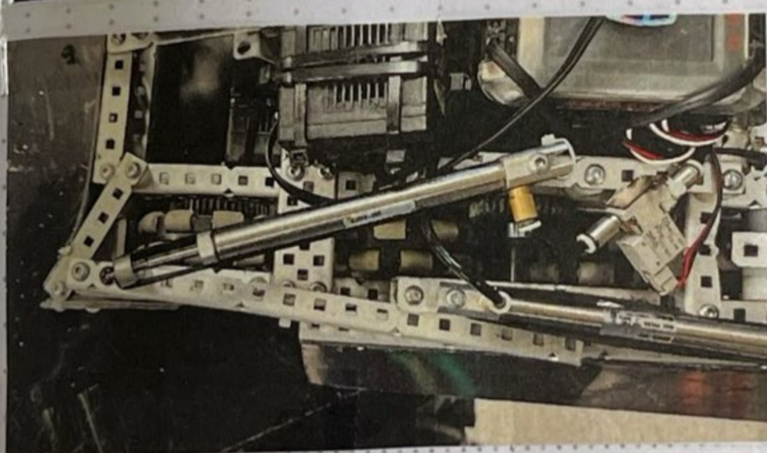
Above are images of the wings beign begun to be built with the diagram of how to build them below.



☆ Collar lock attached to piston and coupled to spacers and wings.

Mirror and Repeat

(continued)



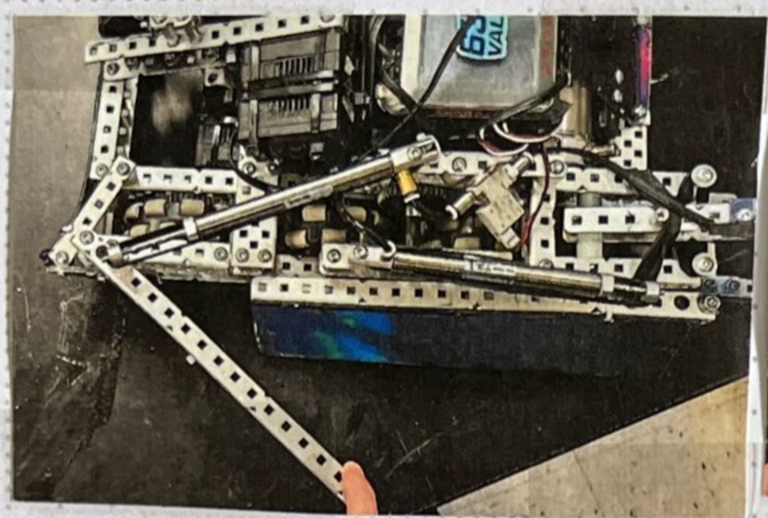
To the left are more images of the back wings and a parts list is written below that goes with the diagram on the previous page.

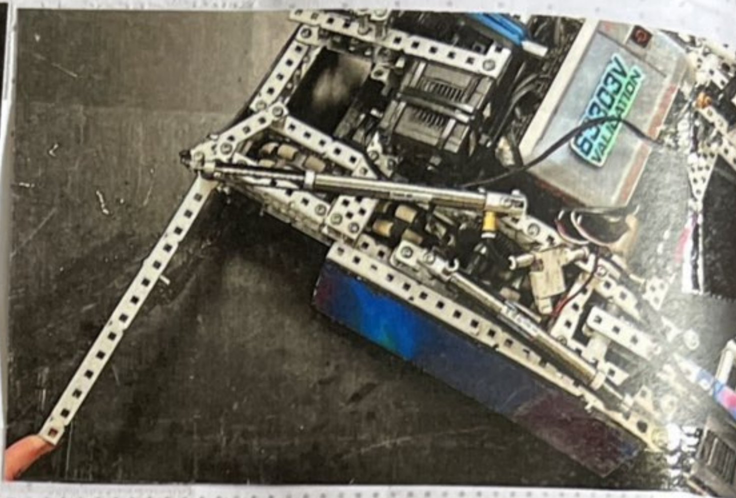
VP

Parts List:

- 2 single acting pistons
- 6-1.000 inch screws
- 2-8 long aluminum 1by1 L-channels
- 6-0.500 inch spacers
- 2-0.125 inch spacers
- 2-0.250 inch spacers
- 4 Teflon washers
- 2 Castle locks
- 2 collar locks
- 2 zipties
- 2 bearing flats
- 2-8mm spacers

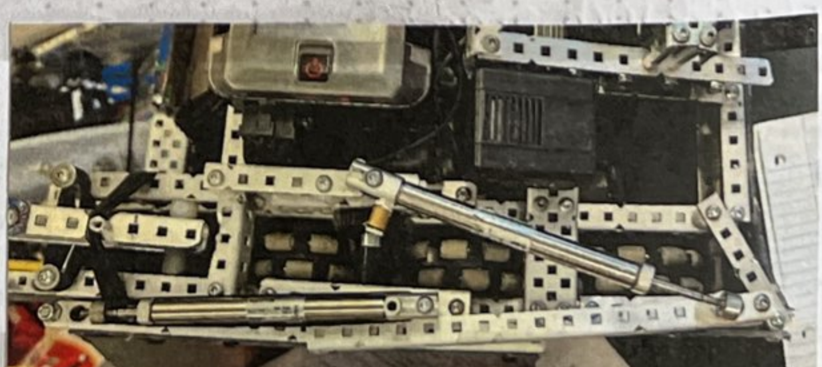
Finished Back Wings Images:





This page has pictures of the finished back wings from several different angles and positions.

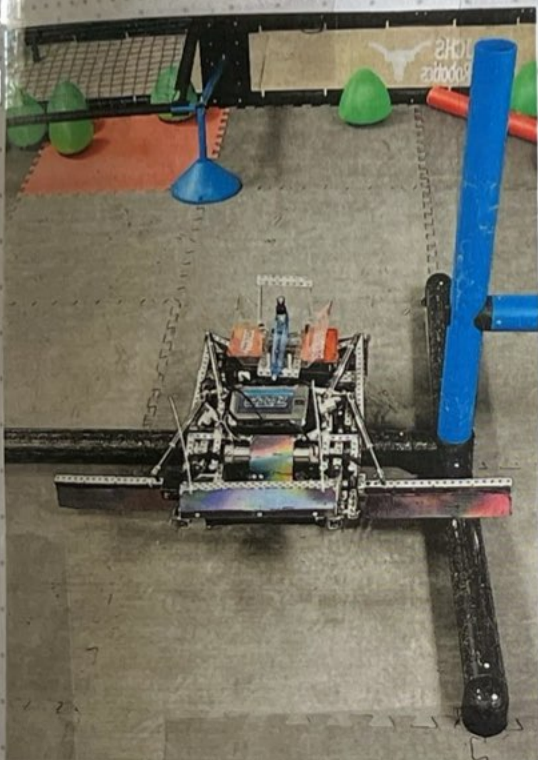
The purpose of the back wings is to descore the triballs from the corners during autons.

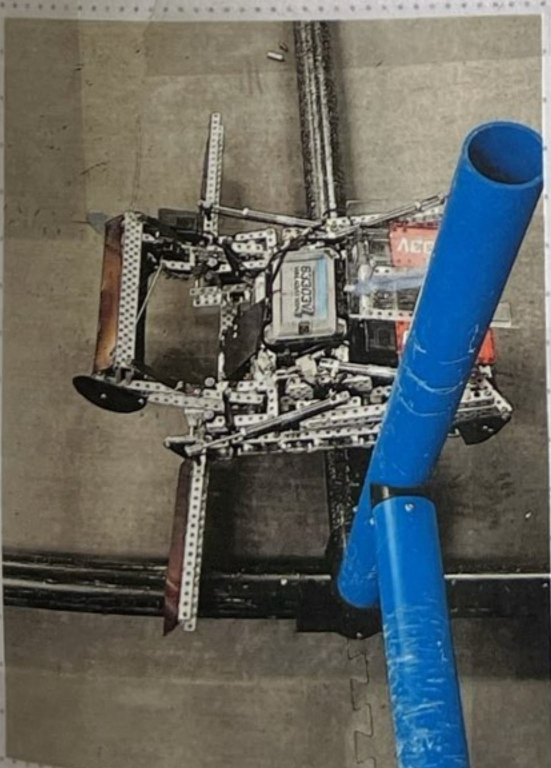


Project Finished Back Wings Images Name Vanessa Perkins
Date 2/27/2024

Elevation:

Goals for today → Find a way to elevate our robot





We want to find a way for our robot to elevate while staying in our six(6) inch dimensions and be done in the short time we have remaining.
We brainstormed for a while but couldn't come up with any reasonable ideas so we decided to ~~wi~~ VP test a barrier hang with our wings.

Test

The wings Lexan skirt was too low for us to hang.

Solution

We shortened the front wing skirts by half an inch on each side. This was done by cutting the Lexan straight across.

Test Again

Now it works when we balance on the barrier to elevate.
To the left are pictures of it working on the field.

VP

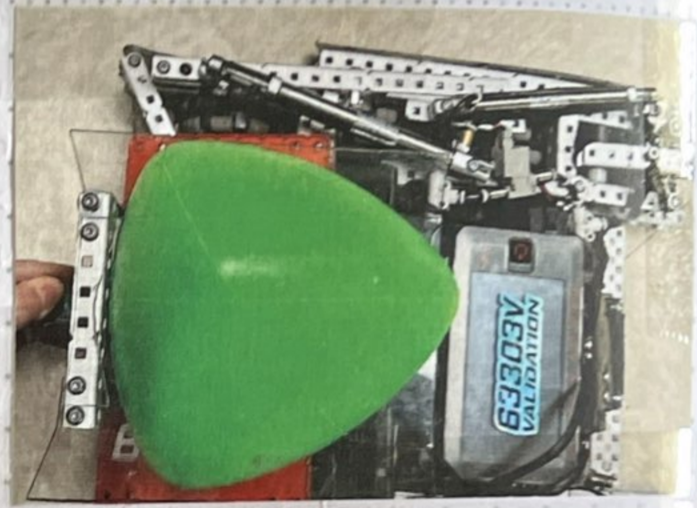
Skills Triball Tray:

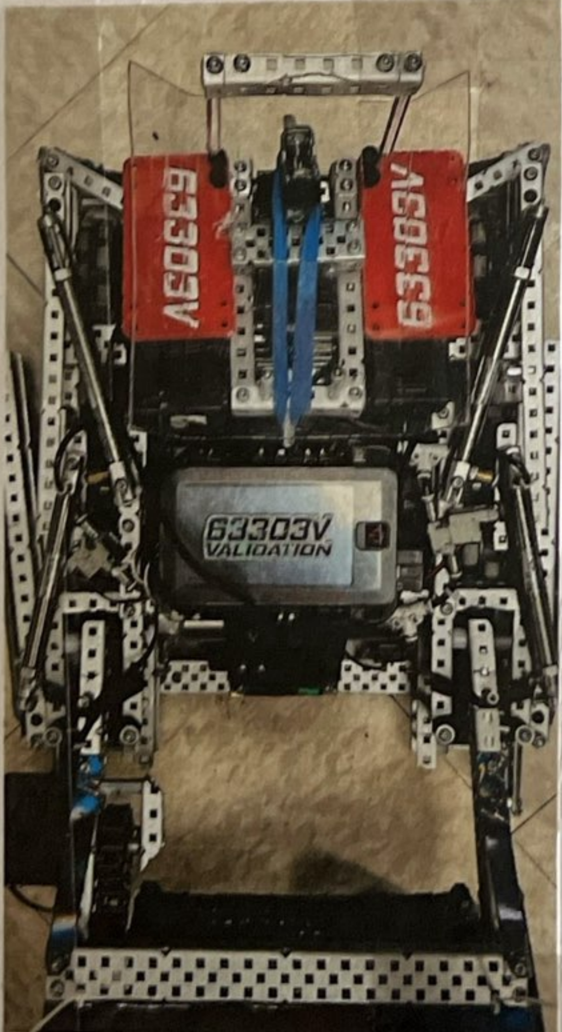
Goals for today → Design a detachable tray to use during skills to align triballs with the kicker.

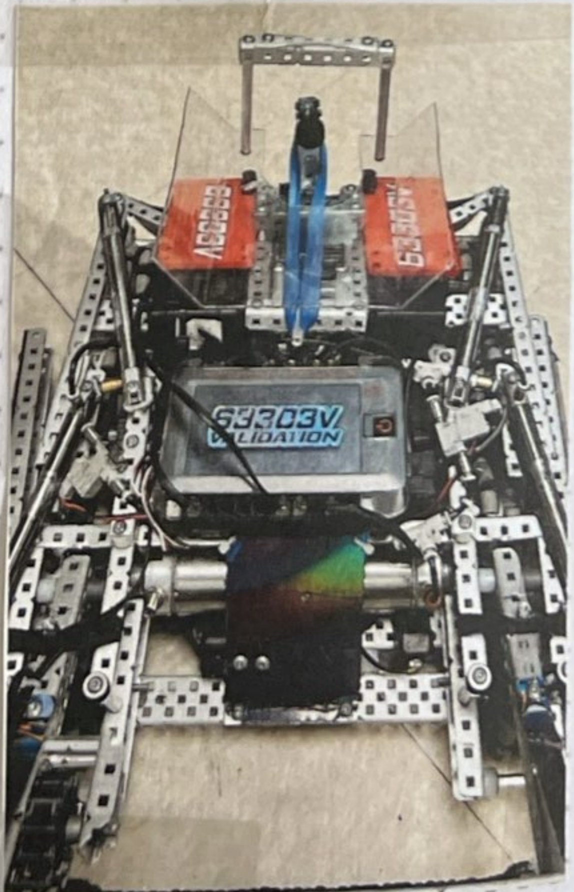
Below are images of the tray we attached to the kicker using two(2) 1.000 inch standoffs. The tray is an 8in square with the edges bent up and an arch cut out of the back for the kicker mechanism.

On the back we used 2.00 inch standoffs to attach an eight(8) long 1 by 1 L-channel with a Lexan sheet with the curve of the triball that it rests against. →

It was tested and it works very well for skills.







Driving and Programming:

Goals for today → Owen practice driving and elevating
→ Harris work on autons

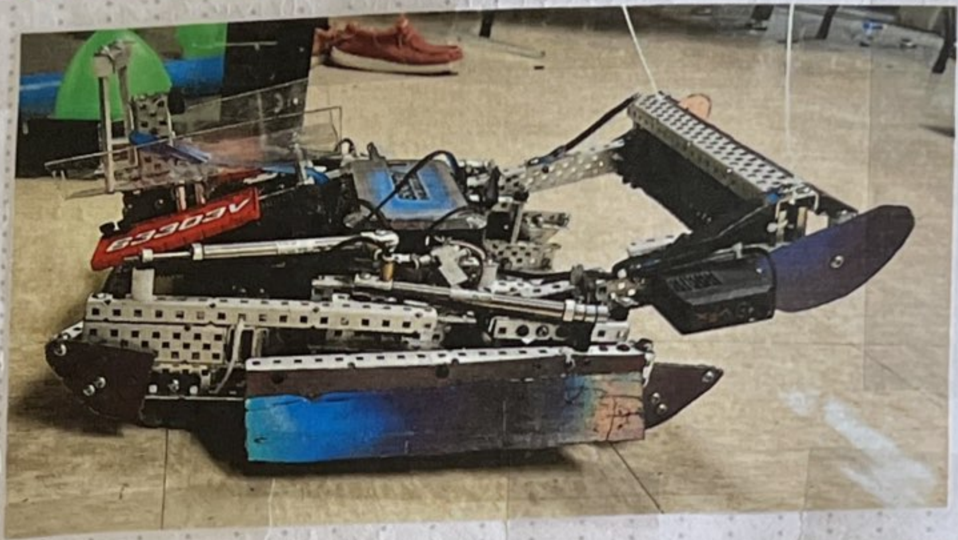
Today we worked on running driver and programming skills to get in final practice before States.

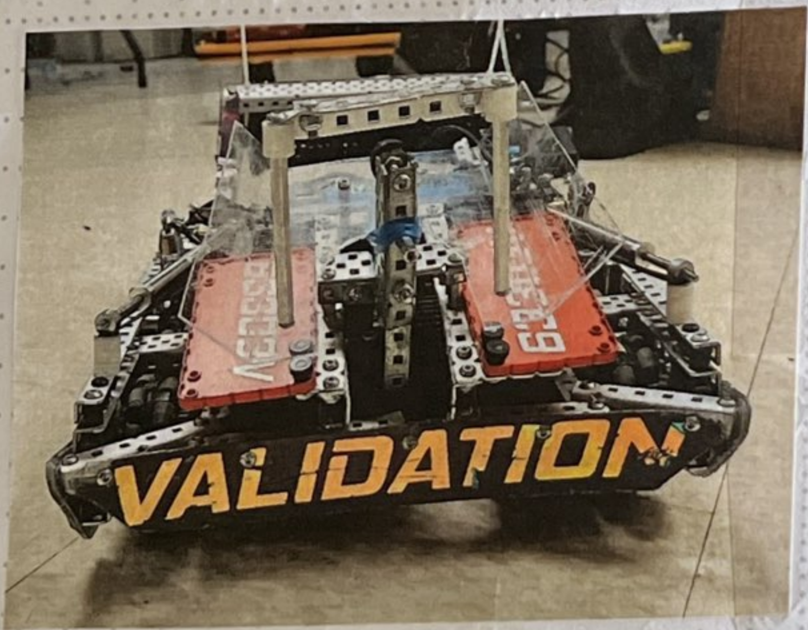
The picture to the left shows our field lowered to help us prepare better for States.

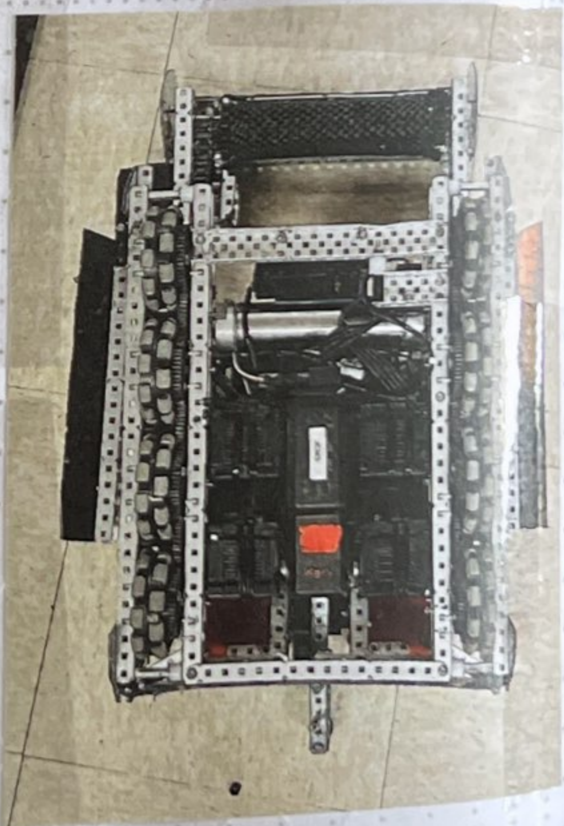
We also plan to take a field with us to practice more and update autons as needed at States.

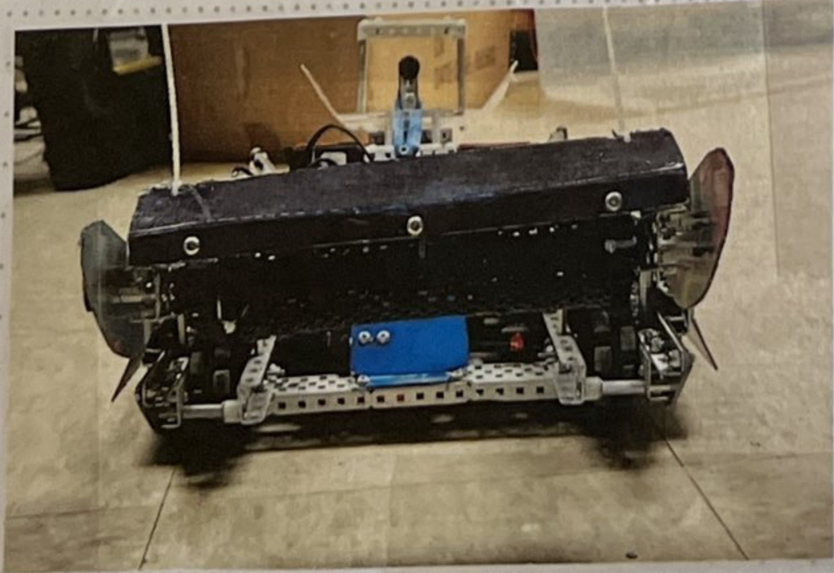
VR

Full Robot Images:





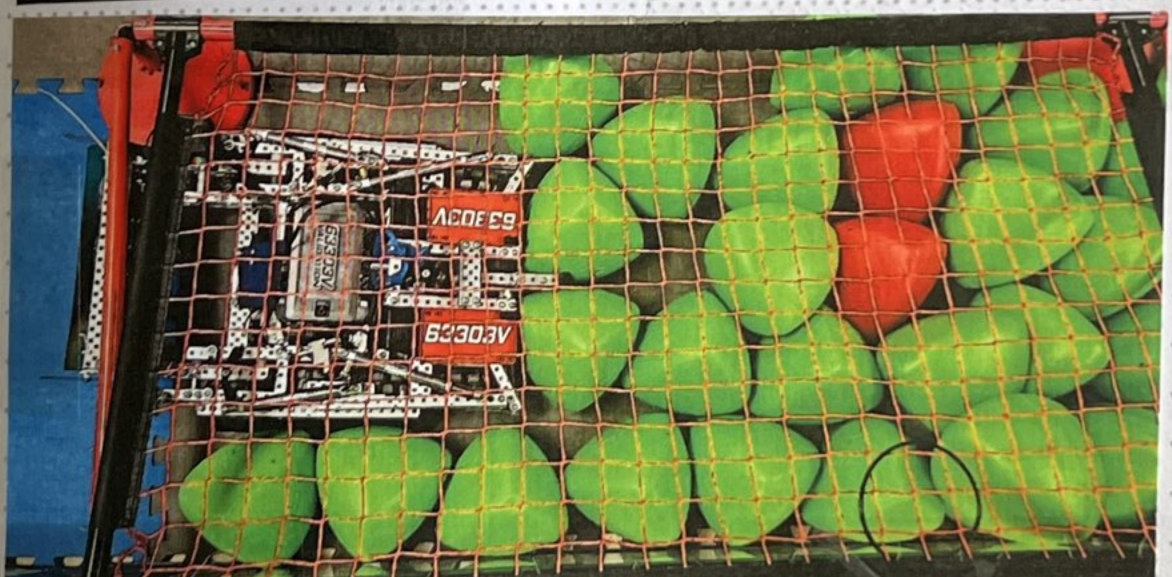


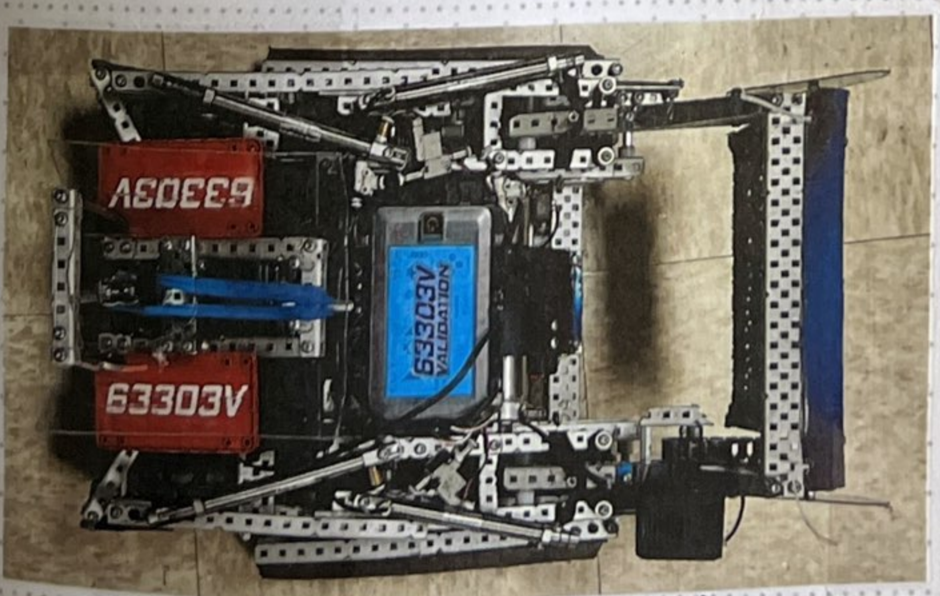


Above are several angles of the full complete robot with more on the next page.

Project Full Robot Images
Name Vanessa Perkins
Date 2/29/2024

(continued)





Above are images of the robot traveling under the goal since it is only six (6) inches tall.

Project Full Robot Images (continued) Name Vanessa Perkins
Date 2/29/2024

AWP Close Side Auton

```
#include "vex.h"

// ---- START VEXCODE CONFIGURED DEVICES ----
// Robot Configuration:
// [Name]               [Type]         [Port(s)]
// Controller1          controller
// Front_Right_Bottom   motor          2
// Intake               motor          11
// Front_Left_Bottom    motor          5
// Back_Right           motor          6
// Top_Right            motor          7
// Top_Left             motor          9
// Back_Left            motor          8
// FrontWings           digital_out    G
// Inertial             inertial       19
// BackWingLeft         digital_out    H
// BackWingRight        digital_out    F
// Kicker               motor_group    16, 18
// ---- END VEXCODE CONFIGURED DEVICES ----

using namespace vex;
competition Competition;

/*---------------------------------------------------------------------------
/
/*                          VEXcode Config
*/
/*
*/
/*  Before you do anything else, start by configuring your motors and
*/
/*  sensors using the V5 port icon in the top right of the screen. Doing
*/
/*  so will update robot-config.cpp and robot-config.h automatically, so
*/
/*  you don't have to. Ensure that your motors are reversed properly. For
*/
/*  the drive, spinning all motors forward should drive the robot forward.
*/
/*---------------------------------------------------------------------------
/

/*---------------------------------------------------------------------------
/
/*                          JAR-Template Config
*/
/*
*/
/*  Where all the magic happens. Follow the instructions below to input
*/
/*  all the physical constants and values for your robot. You should
*/
/*  already have configured your robot manually with the sidebar configurer.
*/
```

```
/*---------------------------------------------------------------------------*
/

Drive chassis(

//Specify your drive setup below. There are seven options:
//ZERO_TRACKER_NO_ODOM, ZERO_TRACKER_ODOM, TANK_ONE_ENCODER,
TANK_ONE_ROTATION, TANK_TWO_ENCODER, TANK_TWO_ROTATION, HOLONOMIC_TWO_ENCODER,
and HOLONOMIC_TWO_ROTATION
//For example, if you are not using odometry, put ZERO_TRACKER_NO_ODOM below:
ZERO_TRACKER_NO_ODOM,

//Add the names of your Drive motors into the motor groups below, separated by
commas, i.e. motor_group(Motor1,Motor2,Motor3).
//You will input whatever motor names you chose when you configured your robot
using the sidebar configurer, they don't have to be "Motor1" and "Motor2".

//Left Motors:
motor_group(Front_Left_Bottom, Top_Left, Back_Left),

//Right Motors:
motor_group(Front_Right_Bottom, Top_Right, Back_Right),

//Specify the PORT NUMBER of your inertial sensor, in PORT format (i.e.
"PORT1", not simply "1"):
PORT19,

//Input your wheel diameter. (4" omnis are actually closer to 4.125"):
3.25,

//External ratio, must be in decimal, in the format of input teeth/output
teeth.
//If your motor has an 84-tooth gear and your wheel has a 60-tooth gear, this
value will be 1.4.
//If the motor drives the wheel directly, this value is 1:
0.75,

//Gyro scale, this is what your gyro reads when you spin the robot 360
degrees.
//For most cases 360 will do fine here, but this scale factor can be very
helpful when precision is necessary.
360,

/*---------------------------------------------------------------------------*
/
/*                              PAUSE!
*/
/*
*/
/*  The rest of the drive constructor is for robots using POSITION TRACKING.
*/
/*  If you are not using position tracking, leave the rest of the values as
*/
/*  they are.
*/
/*---------------------------------------------------------------------------*
/
```

Project AWP Close Side Auton (Continued) Name Harris Perkins

Date 2/29/2024

```
//If you are using ZERO_TRACKER_ODOM, you ONLY need to adjust the FORWARD
TRACKER CENTER DISTANCE.

//FOR HOLONOMIC DRIVES ONLY: Input your drive motors by position. This is only
necessary for holonomic drives, otherwise this section can be left alone.
//LF:       //RF:
PORT1,      -PORT2,

//LB:       //RB:
PORT3,      -PORT4,

//If you are using position tracking, this is the Forward Tracker port (the
tracker which runs parallel to the direction of the chassis).
//If this is a rotation sensor, enter it in "PORT1" format, inputting the port
below.
//If this is an encoder, enter the port as an integer. Triport A will be a
"1", Triport B will be a "2", etc.
3,

//Input the Forward Tracker diameter (reverse it to make the direction
switch):
2.75,

//Input Forward Tracker center distance (a positive distance corresponds to a
tracker on the right side of the robot, negative is left.)
//For a zero tracker tank drive with odom, put the positive distance from the
center of the robot to the right side of the drive.
//This distance is in inches:
-2,

//Input the Sideways Tracker Port, following the same steps as the Forward
Tracker Port:
1,

//Sideways tracker diameter (reverse to make the direction switch):
-2.75,

//Sideways tracker center distance (positive distance is behind the center of
the robot, negative is in front):
5.5

);

int current_auton_selection = 0;
bool auto_started = false;

void pre_auton(void) {
 // Initializing Robot Configuration. DO NOT REMOVE!
 vexcodeInit();
 default_constants();

}

void autonomous(void) {
 auto_started = true;
 Intake.spin(forward, 100, pct);
```

```
chassis.drive_distance(45, 0, 12, 12, 1000, 1000, 3000, 1, 0, 0, 0, 0, 0, 0,
0);
Intake.setStopping(hold);
wait(0.5, sec);
//Intake.stop();
chassis.drive_distance(-10, 45, 12, 12, 1000, 1000, 1000, 1, 0, 0, 0, 0, 0,
0, 0);
Intake.stop();
chassis.turn_to_angle(195);
chassis.drive_distance(43.5, 190, 12, 12, 1000, 1000, 2000, 0.4, 0, 0, 0, 0,
0, 0, 0);
BackWingRight.set(true);
wait(0.5, sec);
chassis.turn_to_angle(95, 12, 1000, 1000, 1000, 0.1, 0, 0, 0);
Intake.spin(reverse, 100, pct);
chassis.drive_distance(25, 90, 12, 12, 1000, 1000, 1000, 1, 0, 0, 0, 0, 0, 0,
0);
BackWingRight.set(false);
chassis.turn_to_angle(80);
chassis.drive_distance(12, 80, 12, 12, 1000, 1000, 1000, 1, 0, 0, 0, 0, 0, 0,
0);


}

/*---------------------------------------------------------------------------*
/
/*
*/
/*                         User Control Task
*/
/*
*/
/*  This task is used to control your robot during the user control phase of
*/
/*  a VEX Competition.
*/
/*
*/
/*  You must modify the code to add your own robot specific commands here.
*/
/*---------------------------------------------------------------------------*
/

void usercontrol(void) {
 // User control code here, inside the loop
 while (1) {
   Front_Left_Bottom.spin(forward, Controller1.Axis3.position(percent) +
   Controller1.Axis1.position(percent), percent); // The code for driving with
the Front Left motor

   Back_Left.spin(forward, Controller1.Axis3.position(percent) +
   Controller1.Axis1.position(percent), percent); // Left Back Drive Code

   Top_Left.spin(forward, Controller1.Axis3.position(percent) +
   Controller1.Axis1.position(percent), percent); // Left Center Drive Code
```

```
Front_Right_Bottom.spin(forward, Controller1.Axis3.position(percent) -
Controller1.Axis1.position(percent), percent); // Right Front Drive Code

Top_Right.spin(forward, Controller1.Axis3.position(percent) -
Controller1.Axis1.position(percent), percent); // Right Center Drive Code

Back_Right.spin(forward, Controller1.Axis3.position(percent) -
Controller1.Axis1.position(percent), percent); // Right Back Drive Code



    if(Controller1.ButtonR1.pressing())
  { Intake.setVelocity(100,percent);
    Intake.spin(vex::directionType::fwd, 100, velocityUnits::pct);
    Intake.setStopping(hold);

    }

else if(Controller1.ButtonR2.pressing())
  { Intake.setVelocity(-100,percent);
    Intake.spin(vex::directionType::fwd, -100, velocityUnits::pct);
    Intake.setStopping(hold);
    }

else if(Controller1.ButtonX.pressing())
   {FrontWings.set(true);
   }

else if(Controller1.ButtonA.pressing())
   {FrontWings.set(false);
   }

else if(Controller1.ButtonLeft.pressing())
   {BackWingLeft.set(true);
   }

else if(Controller1.ButtonUp.pressing())
   {BackWingLeft.set(false);
   }

else if(Controller1.ButtonRight.pressing())
   {BackWingRight.set(true);
   }

else if(Controller1.ButtonDown.pressing())
   {BackWingRight.set(false);
   }

else if(Controller1.ButtonL1.pressing())
   { Kicker.setVelocity(100,percent);
     Kicker.spin(vex::directionType::fwd, 100, velocityUnits::pct);
     Kicker.setStopping(hold);
     }

else if(Controller1.ButtonL2.pressing())
   { Kicker.setVelocity(-100,percent);
     Kicker.spin(vex::directionType::fwd, -100, velocityUnits::pct);
```

```
      Kicker.setStopping(hold);
      }

        else
     {
      Intake.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);
      Kicker.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);
      }

    wait(20, msec); // Sleep the task for a short amount of time to
                    // prevent wasted resources.

  }
}

//
// Main will set up the competition functions and callbacks.
//
int main() {
  // Set up callbacks for autonomous and driver control periods.
  Competition.autonomous(autonomous);
  Competition.drivercontrol(usercontrol);

  // Run the pre-autonomous function.
  pre_auton();

  // Prevent main from exiting with an infinite loop.
  while (true) {
    wait(100, msec);
  }
}
```

Far Side Auton

```
#include "vex.h"

// ---- START VEXCODE CONFIGURED DEVICES ----
// Robot Configuration:
// [Name]               [Type]          [Port(s)]
// Controller1          controller
// Front_Right_Bottom   motor           2
// Intake               motor           11
// Front_Left_Bottom    motor           5
// Back_Right           motor           6
// Top_Right            motor           7
// Top_Left             motor           9
// Back_Left            motor           8
// FrontWings           digital_out     G
// Inertial             inertial        19
// BackWingLeft         digital_out     H
// BackWingRight        digital_out     F
// Kicker               motor_group     16, 18
// ---- END VEXCODE CONFIGURED DEVICES ----

using namespace vex;
competition Competition;

/*---------------------------------------------------------------------------*
/
/*                              VEXcode Config
*/
/*
*/
/*  Before you do anything else, start by configuring your motors and
*/
/*  sensors using the V5 port icon in the top right of the screen. Doing
*/
/*  so will update robot-config.cpp and robot-config.h automatically, so
*/
/*  you don't have to. Ensure that your motors are reversed properly. For
*/
/*  the drive, spinning all motors forward should drive the robot forward.
*/
/*---------------------------------------------------------------------------*
/

/*---------------------------------------------------------------------------*
/
/*                              JAR-Template Config
*/
/*
*/
```

Project Far Side Auton   Name Harris Perkins

Date 2/29/2024

```
/*  Where all the magic happens. Follow the instructions below to input
*/
/*  all the physical constants and values for your robot. You should
*/
/*  already have configured your robot manually with the sidebar configurer.
*/
/*------------------------------------------------------------------------*
/

Drive chassis(

//Specify your drive setup below. There are seven options:
//ZERO_TRACKER_NO_ODOM, ZERO_TRACKER_ODOM, TANK_ONE_ENCODER,
TANK_ONE_ROTATION, TANK_TWO_ENCODER, TANK_TWO_ROTATION, HOLONOMIC_TWO_ENCODER,
and HOLONOMIC_TWO_ROTATION
//For example, if you are not using odometry, put ZERO_TRACKER_NO_ODOM below:
ZERO_TRACKER_NO_ODOM,

//Add the names of your Drive motors into the motor groups below, separated by
commas, i.e. motor_group(Motor1,Motor2,Motor3).
//You will input whatever motor names you chose when you configured your robot
using the sidebar configurer, they don't have to be "Motor1" and "Motor2".

//Left Motors:
motor_group(Front_Left_Bottom, Top_Left, Back_Left),

//Right Motors:
motor_group(Front_Right_Bottom, Top_Right, Back_Right),

//Specify the PORT NUMBER of your inertial sensor, in PORT format (i.e.
"PORT1", not simply "1"):
PORT19,

//Input your wheel diameter. (4" omnis are actually closer to 4.125"):
3.25,

//External ratio, must be in decimal, in the format of input teeth/output
teeth.
//If your motor has an 84-tooth gear and your wheel has a 60-tooth gear, this
value will be 1.4.
//If the motor drives the wheel directly, this value is 1:
0.75,

//Gyro scale, this is what your gyro reads when you spin the robot 360
degrees.
//For most cases 360 will do fine here, but this scale factor can be very
helpful when precision is necessary.
360,
```

```
/*---------------------------------------------------------------------------*
/
/*                                PAUSE!
*/
/*
*/
/*  The rest of the drive constructor is for robots using POSITION TRACKING.
*/
/*  If you are not using position tracking, leave the rest of the values as
*/
/*  they are.
*/
/*---------------------------------------------------------------------------*
/

//If you are using ZERO_TRACKER_ODOM, you ONLY need to adjust the FORWARD
TRACKER CENTER DISTANCE.

//FOR HOLONOMIC DRIVES ONLY: Input your drive motors by position. This is only
necessary for holonomic drives, otherwise this section can be left alone.
//LF:      //RF:
PORT1,     -PORT2,

//LB:      //RB:
PORT3,     -PORT4,

//If you are using position tracking, this is the Forward Tracker port (the
tracker which runs parallel to the direction of the chassis).
//If this is a rotation sensor, enter it in "PORT1" format, inputting the port
below.
//If this is an encoder, enter the port as an integer. Triport A will be a
"1", Triport B will be a "2", etc.
3,

//Input the Forward Tracker diameter (reverse it to make the direction
switch):
2.75,

//Input Forward Tracker center distance (a positive distance corresponds to a
tracker on the right side of the robot, negative is left.)
//For a zero tracker tank drive with odom, put the positive distance from the
center of the robot to the right side of the drive.
//This distance is in inches:
-2,

//Input the Sideways Tracker Port, following the same steps as the Forward
Tracker Port:
1,
```

```
//Sideways tracker diameter (reverse to make the direction switch):
-2.75,

//Sideways tracker center distance (positive distance is behind the center of the robot, negative is in front):
5.5

);

int current_auton_selection = 0;
bool auto_started = false;

void pre_auton(void) {
  // Initializing Robot Configuration. DO NOT REMOVE!
  vexcodeInit();
  default_constants();


}

void autonomous(void) {
  auto_started = true;
  FrontWings.set(true);
  wait(0.4, sec);
  FrontWings.set(false);
  Intake.spin(fwd, 100, pct);
  chassis.drive_distance(43, 0, 12, 12, 1000, 1000, 1000, 5, 0, 0, 0, 0, 0, 0, 0);
  wait(0.2, sec);
  Intake.setStopping(hold);
  Intake.stop();
  chassis.drive_distance(-45);
  chassis.turn_to_angle(35, 12, 1000, 1000, 1000, 0.06, 0, 0, 0);
  chassis.drive_distance(19, 35, 12, 12, 1000, 1000, 1000, 5, 0, 0, 0, 0, 0, 0, 0);
  wait(1, sec);
  chassis.right_swing_to_angle(15, 12, 1000, 1000, 1000, 0.06, 0, 0, 0);
  Intake.spin(reverse, 100, pct);
  FrontWings.set(true);
  wait(0.3, sec);
  chassis.drive_distance(20, 15, 12, 12, 1000, 1000, 1000, 5, 0, 0, 0, 0, 0, 0, 0);




}
```

```
/*---------------------------------------------------------------------------*
/
/*
*/
/*                              User Control Task
*/
/*
*/
/*  This task is used to control your robot during the user control phase of
*/
/*  a VEX Competition.
*/
/*
*/
/*  You must modify the code to add your own robot specific commands here.
*/
/*---------------------------------------------------------------------------*
/

void usercontrol(void) {
 // User control code here, inside the loop
 while (1) {
   Front_Left_Bottom.spin(forward, Controller1.Axis3.position(percent) +
   Controller1.Axis1.position(percent), percent); // The code for driving with
the Front Left motor

   Back_Left.spin(forward, Controller1.Axis3.position(percent) +
   Controller1.Axis1.position(percent), percent); // Left Back Drive Code

   Top_Left.spin(forward, Controller1.Axis3.position(percent) +
   Controller1.Axis1.position(percent), percent); // Left Center Drive Code
    Front_Right_Bottom.spin(forward, Controller1.Axis3.position(percent) -
   Controller1.Axis1.position(percent), percent); // Right Front Drive Code

   Top_Right.spin(forward, Controller1.Axis3.position(percent) -
   Controller1.Axis1.position(percent), percent); // Right Center Drive Code

   Back_Right.spin(forward, Controller1.Axis3.position(percent) -
   Controller1.Axis1.position(percent), percent); // Right Back Drive Code



      if(Controller1.ButtonR1.pressing())
   { Intake.setVelocity(100,percent);
      Intake.spin(vex::directionType::fwd, 100, velocityUnits::pct);
      Intake.setStopping(hold);

      }
```

```
else if(Controller1.ButtonR2.pressing())
  { Intake.setVelocity(-100,percent);
    Intake.spin(vex::directionType::fwd, -100, VelocityUnits::pct);
    Intake.setStopping(hold);
    }

else if(Controller1.ButtonX.pressing())
  {FrontWings.set(true);
  }

else if(Controller1.ButtonA.pressing())
  {FrontWings.set(false);
  }

else if(Controller1.ButtonUp.pressing())
  {BackWingLeft.set(true);
  }

else if(Controller1.ButtonLeft.pressing())
  {BackWingLeft.set(false);
  }

else if(Controller1.ButtonRight.pressing())
  {BackWingRight.set(true);
  }

else if(Controller1.ButtonDown.pressing())
  {BackWingRight.set(false);
  }

else if(Controller1.ButtonL1.pressing())
  { Kicker.setVelocity(100,percent);
    Kicker.spin(vex::directionType::fwd, 100, velocityUnits::pct);
    Kicker.setStopping(hold);
    }

else if(Controller1.ButtonL2.pressing())
  { Kicker.setVelocity(-100,percent);
    Kicker.spin(vex::directionType::fwd, -100, velocityUnits::pct);
    Kicker.setStopping(hold);
    }


     else
  {
    Intake.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);
    Kicker.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);
    }
```

```
    wait(20, msec); // Sleep the task for a short amount of time to
                    // prevent wasted resources.
  }
}

//
// Main will set up the competition functions and callbacks.
//
int main() {
  // Set up callbacks for autonomous and driver control periods.
  Competition.autonomous(autonomous);
  Competition.drivercontrol(usercontrol);

  // Run the pre-autonomous function.
  pre_auton();

  // Prevent main from exiting with an infinite loop.
  while (true) {
    wait(100, msec);
  }
}
```

VP

Skills Auton

```
#include "vex.h"

// ---- START VEXCODE CONFIGURED DEVICES ----
// Robot Configuration:
// [Name]               [Type]          [Port(s)]
// Controller1          controller
// Front_Right_Bottom   motor           2
// Intake               motor           11
// Front_Left_Bottom    motor           5
// Back_Right           motor           6
// Top_Right            motor           7
// Top_Left             motor           9
// Back_Left            motor           8
// FrontWings           digital_out     G
// Inertial             inertial        19
// BackWingLeft         digital_out     H
// BackWingRight        digital_out     F
// Kicker               motor_group     16, 18
// ---- END VEXCODE CONFIGURED DEVICES ----

using namespace vex;
competition Competition;

/*---------------------------------------------------------------------------*
/
/*                          VEXcode Config
*/
/*
*/
/*  Before you do anything else, start by configuring your motors and
*/
/*  sensors using the V5 port icon in the top right of the screen. Doing
*/
/*  so will update robot-config.cpp and robot-config.h automatically, so
*/
/*  you don't have to. Ensure that your motors are reversed properly. For
*/
/*  the drive, spinning all motors forward should drive the robot forward.
*/
/*---------------------------------------------------------------------------*
/

/*---------------------------------------------------------------------------*
/
/*                          JAR-Template Config
*/
/*
*/
```

```
/*  Where all the magic happens. Follow the instructions below to input
*/
/*  all the physical constants and values for your robot. You should
*/
/*  already have configured your robot manually with the sidebar configurer.
*/
/*---------------------------------------------------------------------------*
/

Drive chassis(

//Specify your drive setup below. There are seven options:
//ZERO_TRACKER_NO_ODOM, ZERO_TRACKER_ODOM, TANK_ONE_ENCODER,
TANK_ONE_ROTATION, TANK_TWO_ENCODER, TANK_TWO_ROTATION, HOLONOMIC_TWO_ENCODER,
and HOLONOMIC_TWO_ROTATION
//For example, if you are not using odometry, put ZERO_TRACKER_NO_ODOM below:
ZERO_TRACKER_NO_ODOM,

//Add the names of your Drive motors into the motor groups below, separated by
commas, i.e. motor_group(Motor1,Motor2,Motor3).
//You will input whatever motor names you chose when you configured your robot
using the sidebar configurer, they don't have to be "Motor1" and "Motor2".

//Left Motors:
motor_group(Front_Left_Bottom, Top_Left, Back_Left),

//Right Motors:
motor_group(Front_Right_Bottom, Top_Right, Back_Right),

//Specify the PORT NUMBER of your inertial sensor, in PORT format (i.e.
"PORT1", not simply "1"):
PORT19,

//Input your wheel diameter. (4" omnis are actually closer to 4.125"):
3.25,

//External ratio, must be in decimal, in the format of input teeth/output
teeth.
//If your motor has an 84-tooth gear and your wheel has a 60-tooth gear, this
value will be 1.4.
//If the motor drives the wheel directly, this value is 1:
0.75,

//Gyro scale, this is what your gyro reads when you spin the robot 360
degrees.
//For most cases 360 will do fine here, but this scale factor can be very
helpful when precision is necessary.
360,
```

```
/*---------------------------------------------------------------------------*
/                              PAUSE!
/*
*/
/*
*/  The rest of the drive constructor is for robots using POSITION TRACKING.
/*
*/  If you are not using position tracking, leave the rest of the values as
/*
*/  they are.
/*
*/
/*---------------------------------------------------------------------------*
/

//If you are using ZERO_TRACKER_ODOM, you ONLY need to adjust the FORWARD
TRACKER CENTER DISTANCE.

//FOR HOLONOMIC DRIVES ONLY: Input your drive motors by position. This is only
necessary for holonomic drives, otherwise this section can be left alone.
//LF:        //RF:
PORT1,       -PORT2,

//LB:        //RB:
PORT3,       -PORT4,

//If you are using position tracking, this is the Forward Tracker port (the
tracker which runs parallel to the direction of the chassis).
//If this is a rotation sensor, enter it in "PORT1" format, inputting the port
below.
//If this is an encoder, enter the port as an integer. Triport A will be a
"1", Triport B will be a "2", etc.
3,

//Input the Forward Tracker diameter (reverse it to make the direction
switch):
2.75,

//Input Forward Tracker center distance (a positive distance corresponds to a
tracker on the right side of the robot, negative is left.)
//For a zero tracker tank drive with odom, put the positive distance from the
center of the robot to the right side of the drive.
//This distance is in inches:
-2,

//Input the Sideways Tracker Port, following the same steps as the Forward
Tracker Port:
1,
```

```
//Sideways tracker diameter (reverse to make the direction switch):
-2.75,

//Sideways tracker center distance (positive distance is behind the center of
the robot, negative is in front):
5.5

);

int current_auton_selection = 0;
bool auto_started = false;

void pre_auton(void) {
 // Initializing Robot Configuration. DO NOT REMOVE!
 vexcodeInit();
 default_constants();

}

void autonomous(void) {
 auto_started = true;
 //Intake fall down
 chassis.drive_distance(-9);
 chassis.left_swing_to_angle(-39, 12, 2000, 2000, 2000, 0.1, 0, 0, 0);
 BackWingRight.set(true);
 Kicker.spin(fwd, 100, pct);
 wait(27, sec);
 BackWingRight.set(false);
 Kicker.setStopping(hold);
 Kicker.stop();
 Intake.spin(reverse, 100, pct);
 chassis.turn_to_angle(10);
 chassis.drive_distance(28);
 chassis.turn_to_angle(0);
 chassis.drive_distance(62, 0, 12, 12, 1000, 1000, 5000, 1, 0, 0, 0, 0, 0, 0,
0);
 chassis.turn_to_angle(135);
 chassis.drive_distance(-18);
 chassis.turn_to_angle(90);
 BackWingRight.set(true);
 BackWingLeft.set(true);
 chassis.drive_distance(-12, 90, 12, 12, 1000, 1000, 2000, 5, 0, 0, 0, 0, 0,
0, 0);
 BackWingRight.set(false);
 BackWingLeft.set(false);
 chassis.drive_distance(13);
 chassis.turn_to_angle(195);
 chassis.drive_distance(45);
 chassis.turn_to_angle(320);  //135
```

Project Skills Auton (Continued) Name Harris Perkins
Date 2/29/2024

```
FrontWings.set(true);
//BackWingRight.set(true);
//BackWingLeft.set(true);
chassis.drive_distance(35, 320, 12, 12, 1000, 1000, 2000, 5, 0, 0, 0, 0, 0,
0, 0);
FrontWings.set(false);
//BackWingRight.set(false);
//BackWingLeft.set(false);
chassis.turn_to_angle(320);
chassis.drive_distance(-35);
chassis.turn_to_angle(90);
chassis.drive_distance(-33);
chassis.turn_to_angle(0);  //180
FrontWings.set(true);
//BackWingRight.set(true);
//BackWingLeft.set(true);
chassis.drive_distance(25, 180, 12, 12, 1000, 1000, 1000, 5, 0, 0, 0, 0, 0,
0, 0);
FrontWings.set(false);
//BackWingRight.set(false);
//BackWingLeft.set(false);
chassis.drive_distance(-25);
chassis.turn_to_angle(-90);  //90
chassis.drive_distance(36);
chassis.turn_to_angle(40);  //220
FrontWings.set(true);
//BackWingRight.set(true);
//BackWingLeft.set(true);
chassis.drive_distance(20, 40, 12, 12, 1000, 1000, 1000, 1, 0, 0, 0, 0, 0, 0,
0); //35
chassis.turn_to_angle(0);
chassis.drive_distance(15, 0, 12, 12, 1000, 1000, 1000, 5, 0, 0, 0, 0, 0, 0,
0);
FrontWings.set(false);
//BackWingRight.set(false);
//BackWingLeft.set(false);
chassis.drive_distance(-25);


}

/*---------------------------------------------------------------------------*
/
/*
*/
/*                          User Control Task
*/
/*
*/
```

```
/*  This task is used to control your robot during the user control phase of
*/
/*  a VEX Competition.
*/
/*
*/
/*  You must modify the code to add your own robot specific commands here.
*/
/*----------------------------------------------------------------------*
/

void usercontrol(void) {
 // User control code here, inside the loop
 while (1) {
   Front_Left_Bottom.spin(forward, Controller1.Axis3.position(percent) +
   Controller1.Axis1.position(percent), percent); // The code for driving with
the Front Left motor

   Back_Left.spin(forward, Controller1.Axis3.position(percent) +
   Controller1.Axis1.position(percent), percent); // Left Back Drive Code

   Top_Left.spin(forward, Controller1.Axis3.position(percent) +
   Controller1.Axis1.position(percent), percent); // Left Center Drive Code
    Front_Right_Bottom.spin(forward, Controller1.Axis3.position(percent) -
   Controller1.Axis1.position(percent), percent); // Right Front Drive Code

   Top_Right.spin(forward, Controller1.Axis3.position(percent) -
   Controller1.Axis1.position(percent), percent); // Right Center Drive Code

   Back_Right.spin(forward, Controller1.Axis3.position(percent) -
   Controller1.Axis1.position(percent), percent); // Right Back Drive Code



     if(Controller1.ButtonR1.pressing())
   { Intake.setVelocity(100,percent);
     Intake.spin(vex::directionType::fwd, 100, velocityUnits::pct);
     Intake.setStopping(hold);

     }

else if(Controller1.ButtonR2.pressing())
   { Intake.setVelocity(-100,percent);
     Intake.spin(vex::directionType::fwd, -100, velocityUnits::pct);
     Intake.setStopping(hold);
     }

else if(Controller1.ButtonB.pressing())
   {
```

```
    Intake.spin(reverse, 100, pct);
    Intake.setBrake(hold);
    Back_Left.spin(forward, 100,pct);
    Back_Right.spin(reverse, 100,pct);
    Top_Left.spin(forward, 100,pct);
    Top_Right.spin(reverse, 100,pct);
    Front_Left_Bottom.spin(forward, 100,pct);
    Front_Right_Bottom.spin(reverse, 100,pct);
    wait(.1,seconds);
    Back_Left.spin(reverse, 100,pct);
    Back_Right.spin(reverse, 10,pct);
    Top_Left.spin(reverse, 100,pct);
    Top_Right.spin(reverse, 10,pct);
    Front_Left_Bottom.spin(reverse, 100,pct);
    Front_Right_Bottom.spin(reverse, 10,pct);
    wait(.255,seconds);  //.27
    Back_Left.spin(reverse, 100,pct);
    Back_Right.spin(reverse, 100,pct);
    Top_Left.spin(reverse, 100,pct);
    Top_Right.spin(reverse, 100,pct);
    Front_Left_Bottom.spin(reverse, 100,pct);
    Front_Right_Bottom.spin(reverse, 100,pct);
    wait(.15,seconds);
     Back_Left.stop();
    Back_Right.stop();
    Front_Left_Bottom.stop();
    Front_Right_Bottom.stop();
    Top_Left.stop();
    Top_Right.stop();
    BackWingRight.set(true);
    Intake.stop();
  }

else if(Controller1.ButtonX.pressing())
  {FrontWings.set(true);
  }

else if(Controller1.ButtonA.pressing())
  {FrontWings.set(false);
  }

else if(Controller1.ButtonLeft.pressing())
  {BackWingLeft.set(true);
  }

else if(Controller1.ButtonUp.pressing())
  {BackWingLeft.set(false);
  }
```

Project Skills Auton (continued)	Name Harris Perkins

Date 2/29/2024

```
else if(Controller1.ButtonRight.pressing())
   {BackWingRight.set(true);
   }

else if(Controller1.ButtonDown.pressing())
   {BackWingRight.set(false);
   }

else if(Controller1.ButtonL1.pressing())
   { Kicker.setVelocity(100,percent);
     Kicker.spin(vex::directionType::fwd, 100, velocityUnits::pct);
     Kicker.setStopping(hold);
     }

else if(Controller1.ButtonL2.pressing())
   { Kicker.setVelocity(-100,percent);
     Kicker.spin(vex::directionType::fwd, -100, velocityUnits::pct);
     Kicker.setStopping(hold);
     }


       else
   {
     Intake.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);
     Kicker.spin(vex::directionType::fwd, 0, vex::velocityUnits::pct);
     }

   wait(20, msec); // Sleep the task for a short amount of time to
                   // prevent wasted resources.
 }
}

//
// Main will set up the competition functions and callbacks.
//
int main() {
 // Set up callbacks for autonomous and driver control periods.
 Competition.autonomous(autonomous);
 Competition.drivercontrol(usercontrol);

 // Run the pre-autonomous function.
 pre_auton();

 // Prevent main from exiting with an infinite loop.
 while (true) {
   wait(100, msec);
 }
```

Innovate Award Submission Information Form

Only for events that are offering the Innovate Award: Either this form or an entry with the equivalent information is required to be eligible for the Innovate Award. Digital submissions can additionally use the Engineering Notebook link feature found in the "My Account" dashboard on RobotEvents.com.

Instructions: Please fill out all information, printing clearly. For in-person notebooks, please place this page either inside the front cover of the team's notebook or placed as the **last entry** in the notebook when submitting it for judging. In the case of digital notebooks, a picture of the form can be uploaded and placed either at the beginning of the digital notebook, after the Table of Contents, or entered as the last entry in the notebook. Teams may only submit one aspect of their design to be considered for this award at each event.

Full Team Number: 63303V

Brief Description of the novel aspect of the team's design being submitted:

Our robot includes double horizontal wings which allow us versatility in skills and matches while still maintaining a low profile. From descoring for the AWP to pushing triballs over the barrier and in the goal, the wings also help us elevate making them present in every aspect of the game.

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

- Page 94-95, book 3
- Page 108-110, book 3
- Page 111, book 3

Project: Innovate Award Submission Information Form

Name: Vanessa Perkins & Eli Fritts

Date: 2/29/2024

Competition Analysis:

Goals for today → Review our performance at States

This past weekend we went to states in Brentwood. Our two interviews went really well, but we had some set backs will skills and matches that kept us from qualifying for awards or ranking good for eliminations.

In skills we had to figure out how to matchload on raised fields when we were used to low ones which lowered our skills as well as one programming where a triball got stuck during matchloading which we had to fix by adding a spin on the intake.

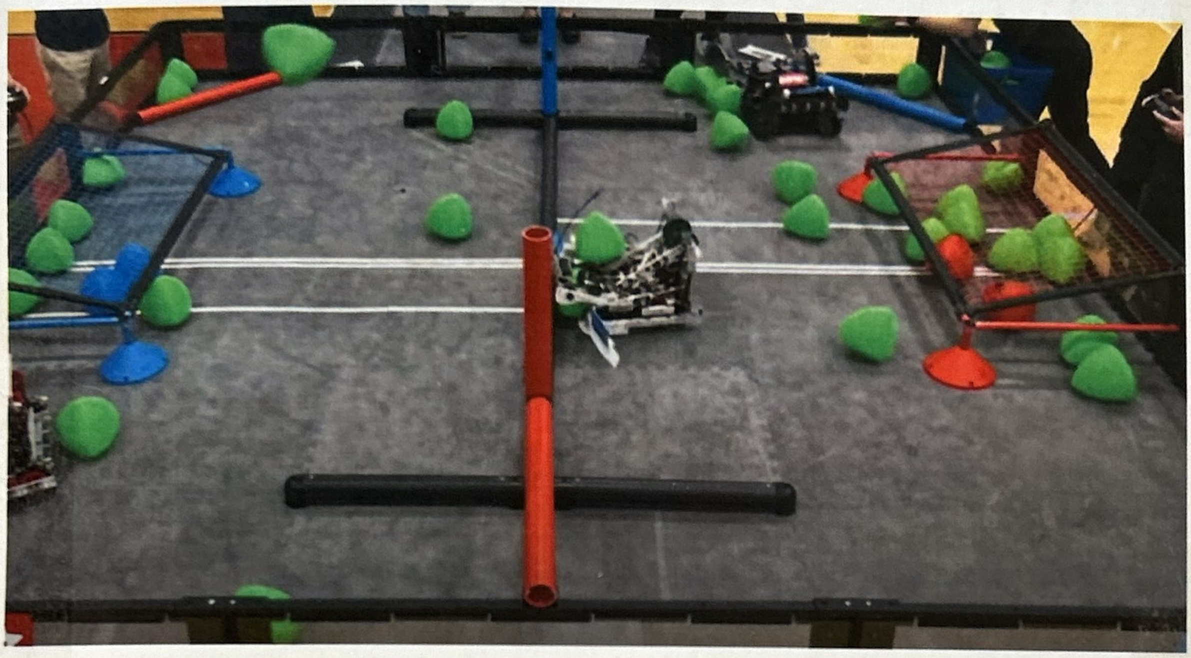
In matches we found that Owen and Eli did not have enough practice with watching for double zoning and controlling the robot enough to descore goals. We got to the Round of 16's but our alliance had trouble following our planned strategy so we lost. On the next several pages are more detailed descriptions of our matches.

Overall, we learned a lot and were thankful for the experience. We have hopes of getting of the wait list to go to WORLDS.

VP

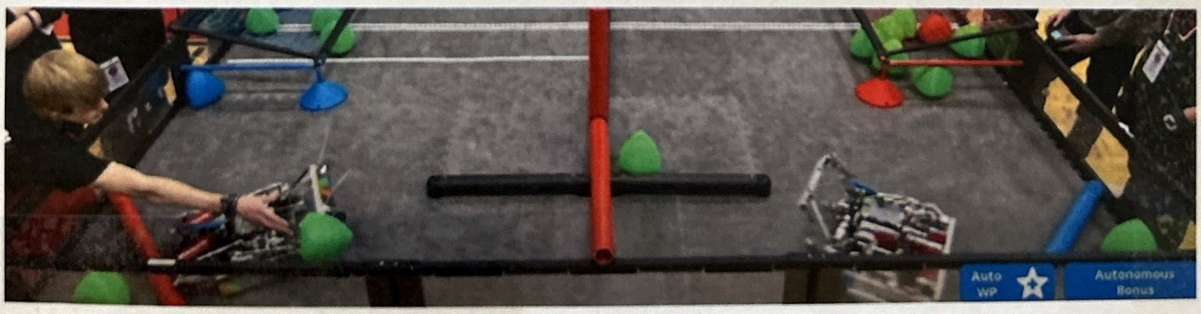
Qualifier #7	Mar 1st at 7:24 PM	69315A	9364A	122	57249D	**63303V**	91

In this match, we had a decent autonomous where we achieved everything except for touching the horizontal bar. This made us miss out on our AWP. Our bowling in this match was poor, as the field was overwhelmed by tribals from matchloading of the opposing alliance, therefore, we should have pushed them back over and did relays as possible. There was also numerous instances of double zoning where we could have descored as well.



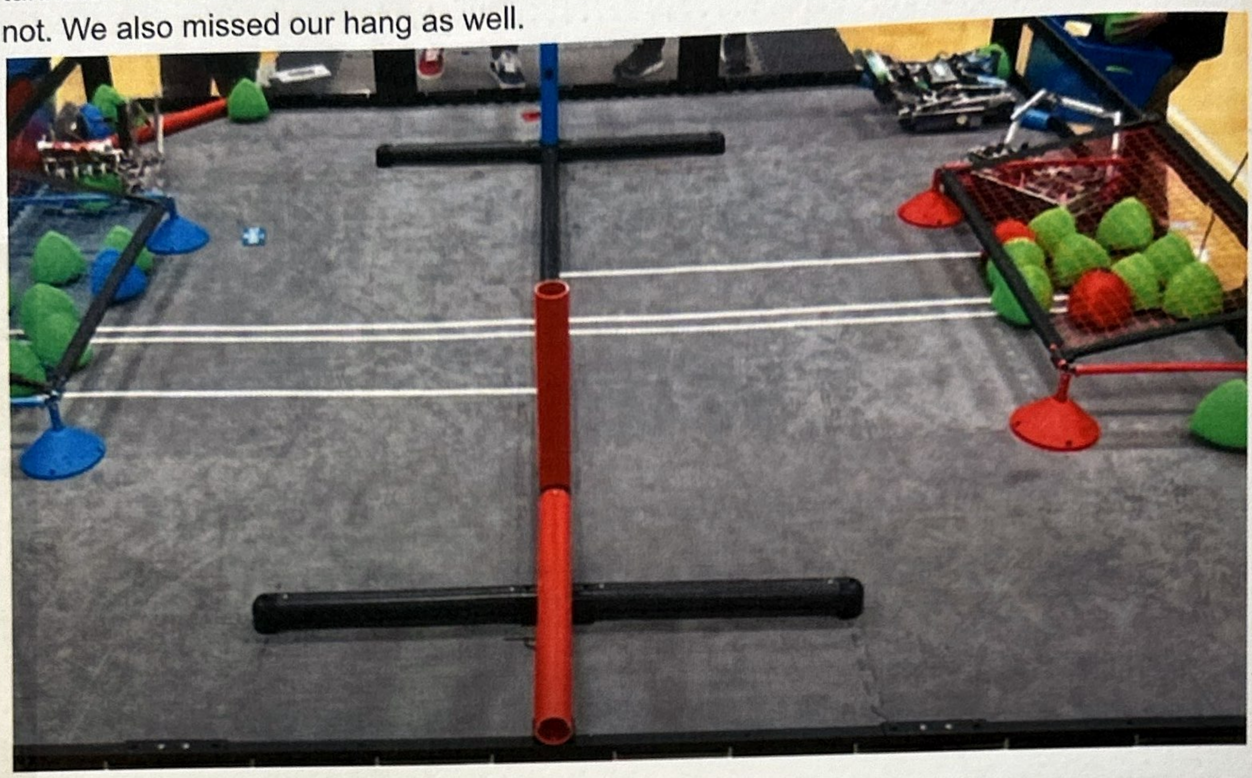
Qualifier #25	Mar 1st at 8:30 PM	38017A	**63303V**	60	991S	22391B	60

In this match, our autonomous did not work at all, every aspect of it failed. Our bowling this match was really bad as well, as Owen was not able to control the robot through the tunnels. There was also a majority of the match where we could have descored and did not.



Qualifier #44	Mar 2nd at 9:44 AM	991J	12876D	65	**63303V**	99905X	59

In this match, our autonomous did not work at all, every aspect of it failed. Our bowling this match was really bad as well, as Owen was not able to control the robot through the tunnels. There was also a majority of the match where we could have descored and did not. We also missed our hang as well.



States Match Analyses continued in Book 4

VP

Project ~~M~~ VP States Match Analyses (Continued) Name Eli Fritts

Date 03/05/2024

Engineering Notebook

63303V
Team Number

Validation
Team Name

Johnson County High School
School

3/5/2024
Start Date

End Date

4 of 4
Book #

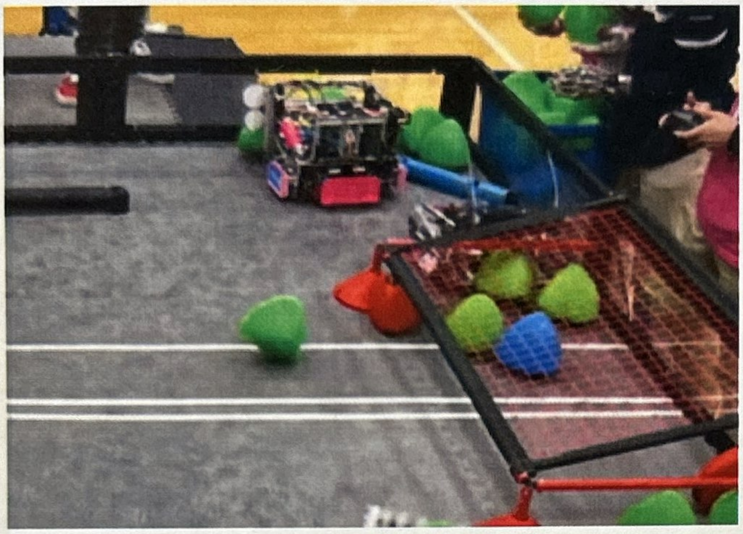
VEX ROBOTICS

REC FOUNDATION

Table of Contents

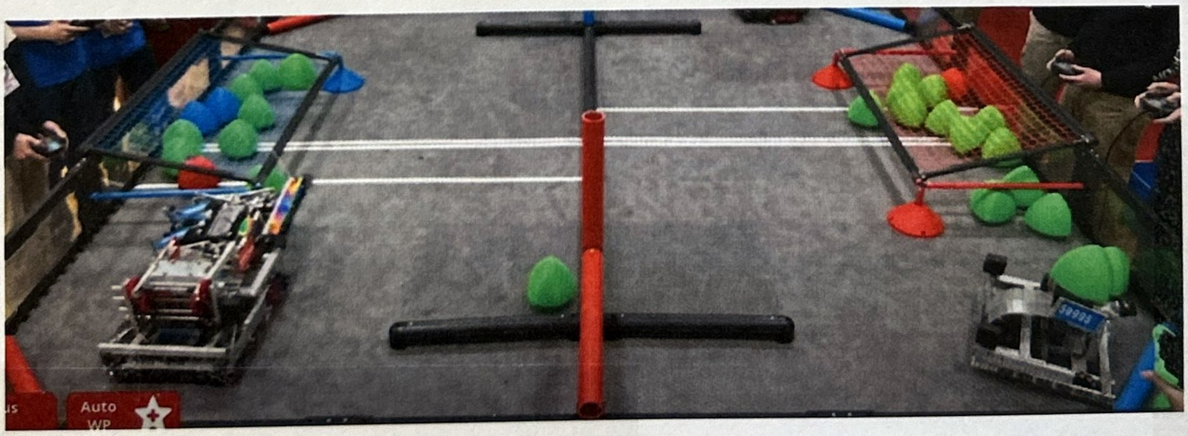
Qualifier #62	Mar 2nd at 10:51 AM	14338A	**63303V**	75	96504A	98709J	49

In this match, we ran our far side autonomous, and it went perfect. However, our alliance partner messed up on their AWP. In this match our bowling was bad again, and we should have descored in several instances.



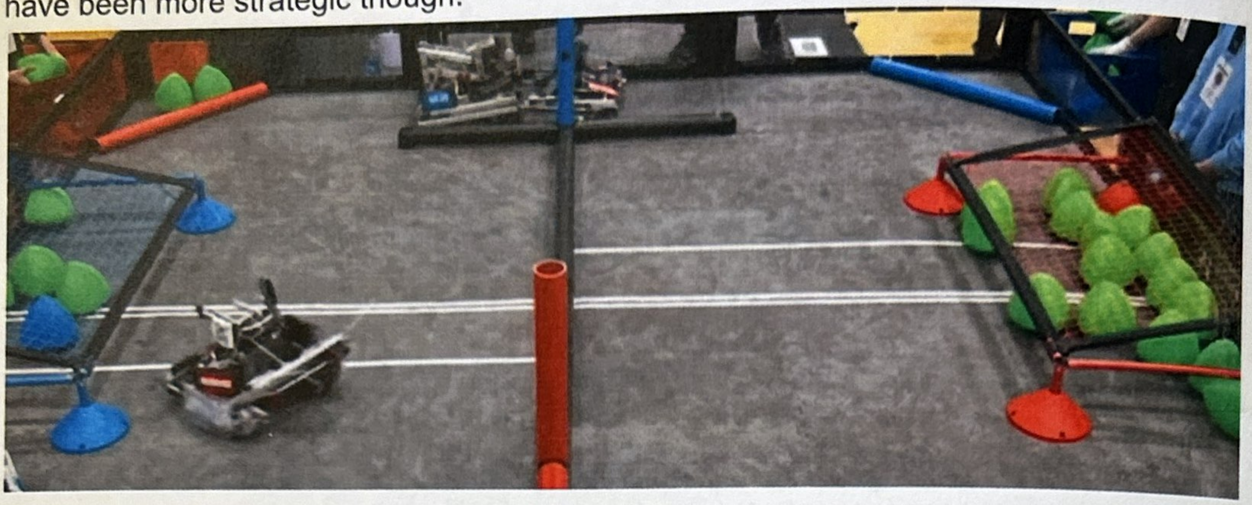
Qualifier #78	Mar 2nd at 11:55 AM	9364C	37334A	105	5999S	**63303V**	56

In this match, we ran our close side autonomous, and it went terrible. In this match our partner shot tribals in 5 increments while we pushed them in. This worked very well, however, we were nearly shut down by the opposing alliance, allowing to not get ahead.



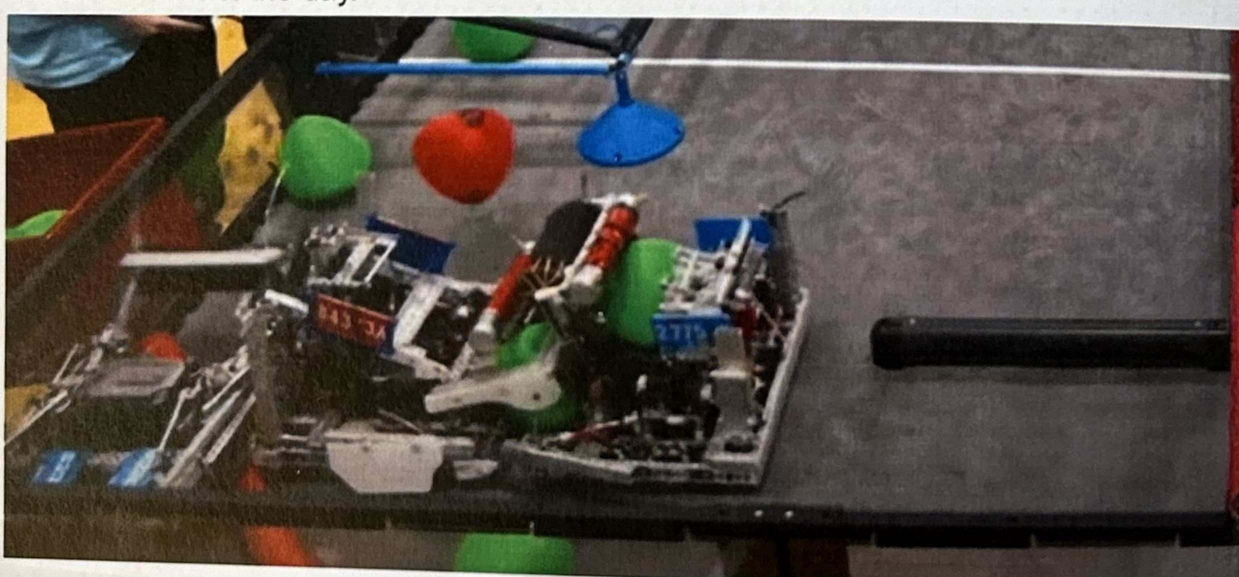
Qualifier #85	Mar 2nd at 12:21 PM	**63303V**	96504C	103	663A	2774A	40

In this match, we ran our far side autonomous, and it went perfect. Our alliance partner bowled, while we played defense and did relays, this went very well. Our defense could have been more strategic though.



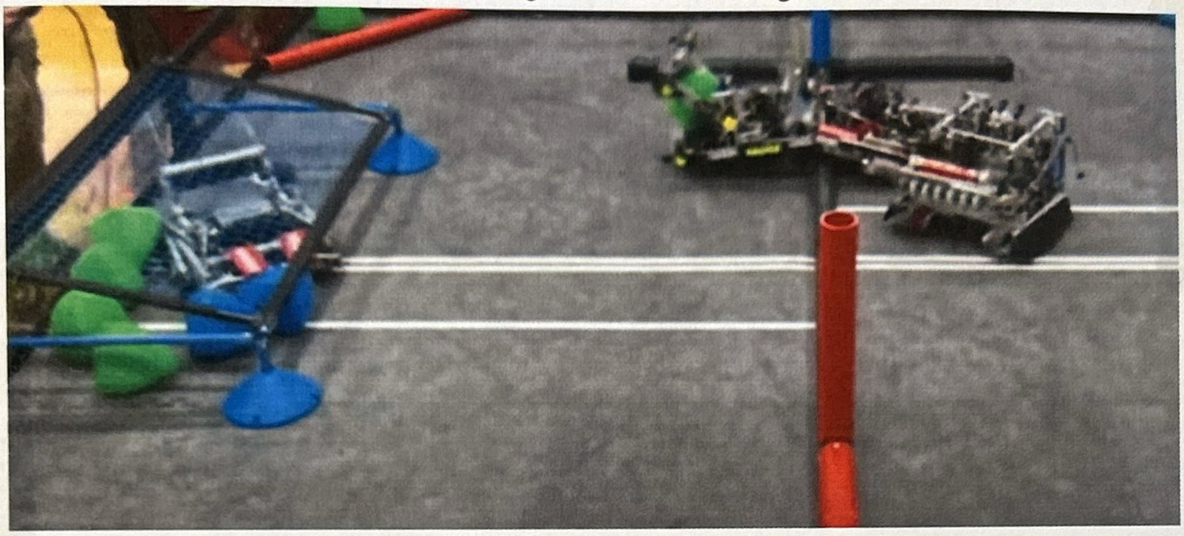
Qualifier #110	Mar 2nd at 3:27 PM	38301B	84373A	55	2775X	**63303V**	85

In this match, our autonomous worked great, as we ran the far side, getting the AWP. Our alliance partner bowled and made relays, while we did defense and relays, this was our best match of the day.



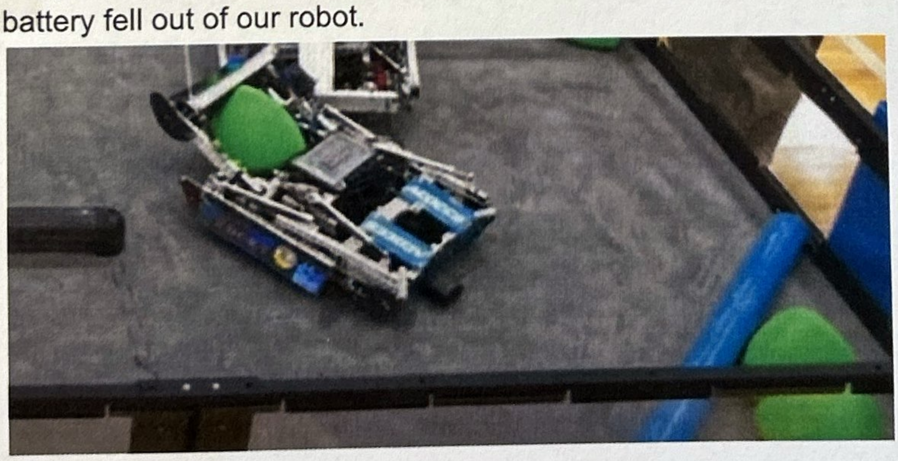
Qualifier #126	Mar 2nd at 4:21 PM	**63303V**	97934S	58	9364D	38401A	75

In this match we did very well, we won autonomous, but missed the win point due to our alliance partner. We had the opportunity to descore several times, however, we did not do it time effectively. Additionally, a hang would have swung the match as well.



Qualifier #137	Mar 2nd at 5:07 PM	98709W	2775V	82	**63303V**	63303A	40

In this match we lost autonomous and the win point. However, with effective defense and relays we were able to make a good comeback. Until, the opposing team hung and our battery fell out of our robot.

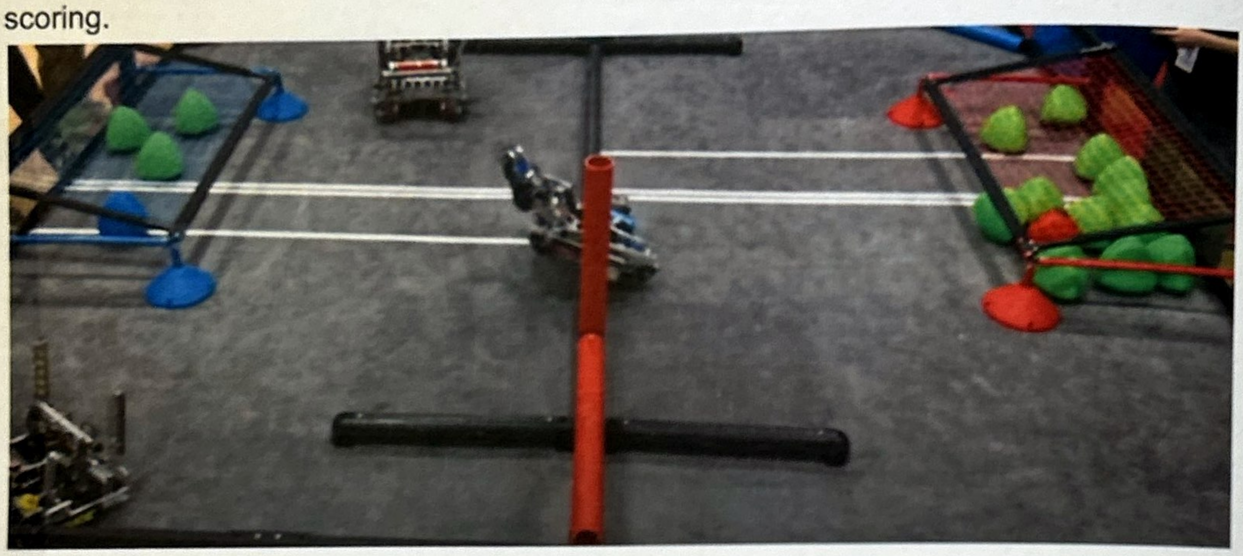


Project States Match Analyses (continued) Name Eli Fritts

Date 3/5/2024

R16 #3-1	Mar 2nd at 7:09 PM	38401A	9364D	125	14338A	**63303V**	30

In this match we lost autonomous, and the match. We had numerous chances to descore and did not and our defense was not effective due to our lack of alliance partner scoring.



VR

Goals:

Goals for today → Plan out what we want to focus on as a team before WORLDS

Below is the list of our goals for the robot and our team that we want to plan around.

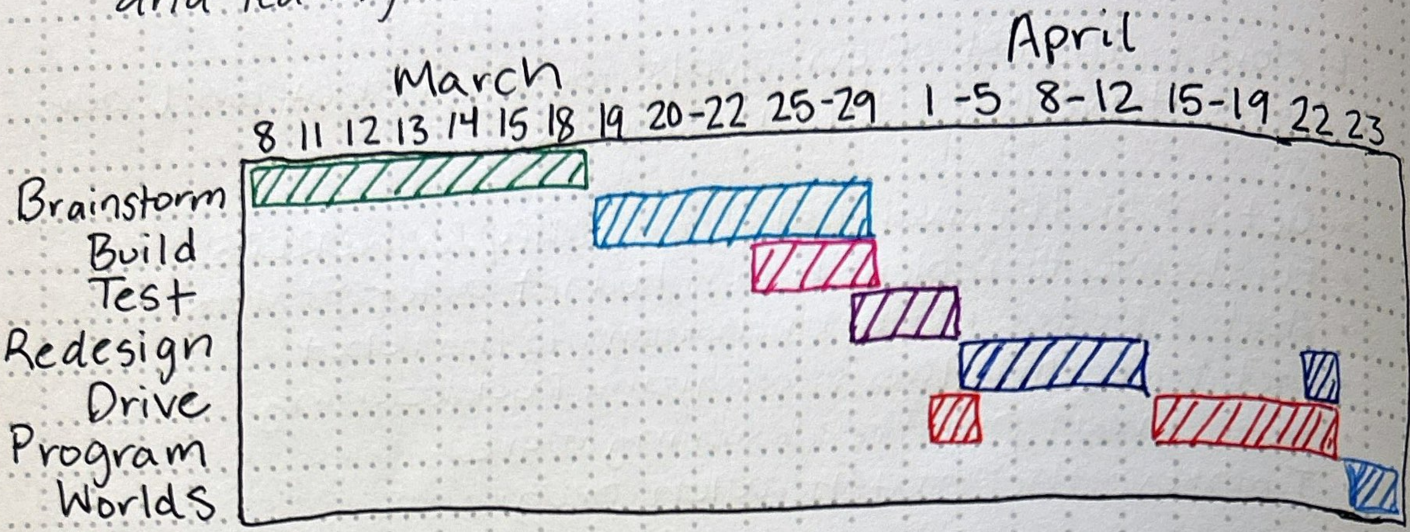
- Get off of the waitlist and qualify for WORLDS
- Fundraise to replenish our budget after States
- Add a better hang mechanism to the robot.
- Get more practice recognizing Doublezoning
- Have Owen practice descoring more
- Improve the match auton programs

These goals will help our team recognize which responsibilities we have to help improve the team.

VP

Timeline:

Goals for today → Plan out our time between now and leaving for WORLDS



Above is a Gantt Chart to help us organize our timeline for March and April. We have to have every ready to leave for Texas on April 23rd.

Our brainstorming stage is longer than normal, because we are actually awaiting acceptance before moving too far forward in preparing for worlds.

VP

Improving Hang Mechanism:

Goals for today → Make a plan for how to improve our hang mechanism

Problem/Challenge

Owen struggled to barrier hang with our wings so we lost many matches.

Brainstorming

① Give Owen more practice.
② Make minor adjustments to the wings.
③ Build a new mechanism.

Select Approach

③

We have enough time, and this option will allow for the biggest improvement.

Build/Implement Solution

- Adjust the base to make room for a new mechanism
- Design and build a hang mechanism that hooks into the top of the elevation bar.

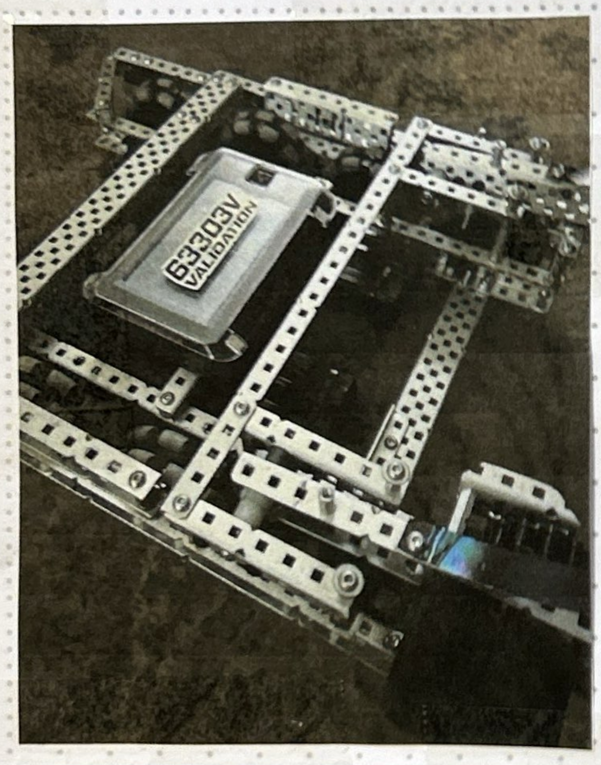
Our next few pages will go through our process of building and implementing this solution into our robot as best as possible before WORLDs notebook submissions are due.

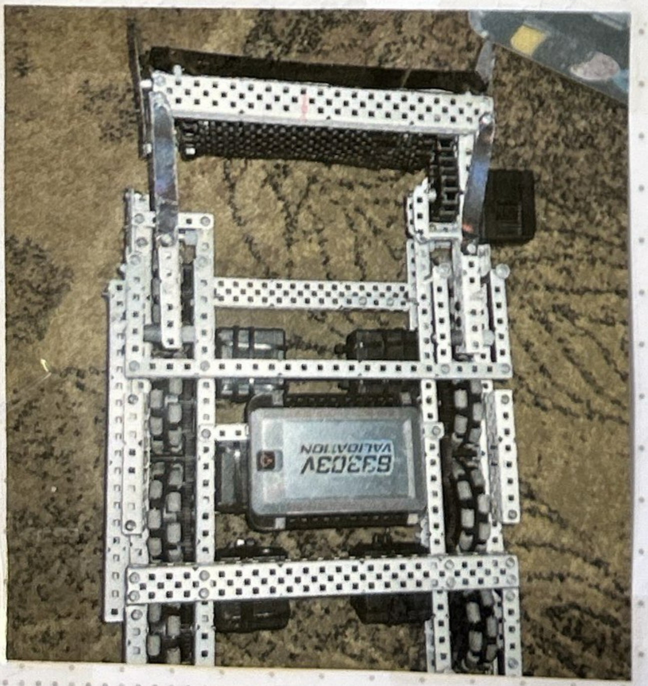
VP

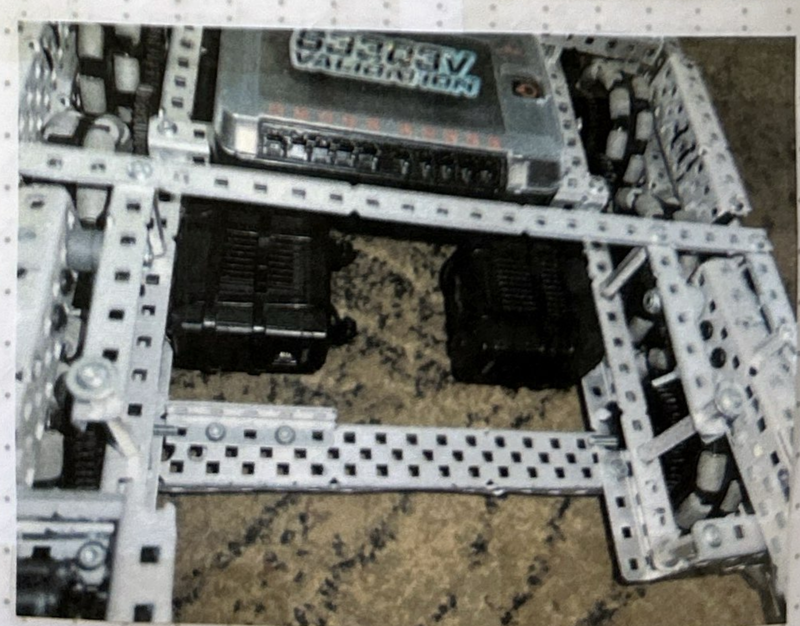
Adjusting Base:

Goals for today → Relocate the base motors and the brain to make more space for a hang mechanism

Previously, the base motors were stacked on top of each other in the middle making the robot taller by roughly two (2) inches making it harder to build on while still being able to descore. To fix this, we moved the motors forward and in row as depicted below. This also allowed us to lower the brain placement.

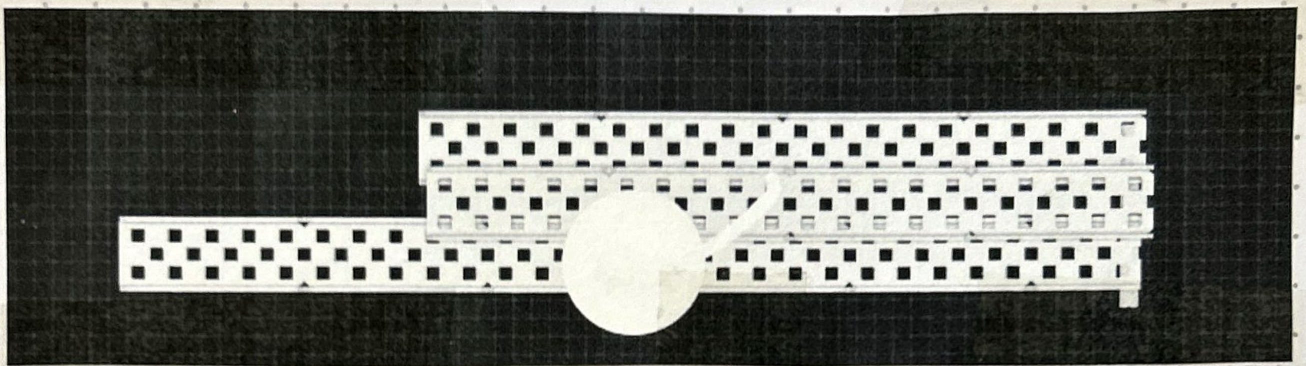


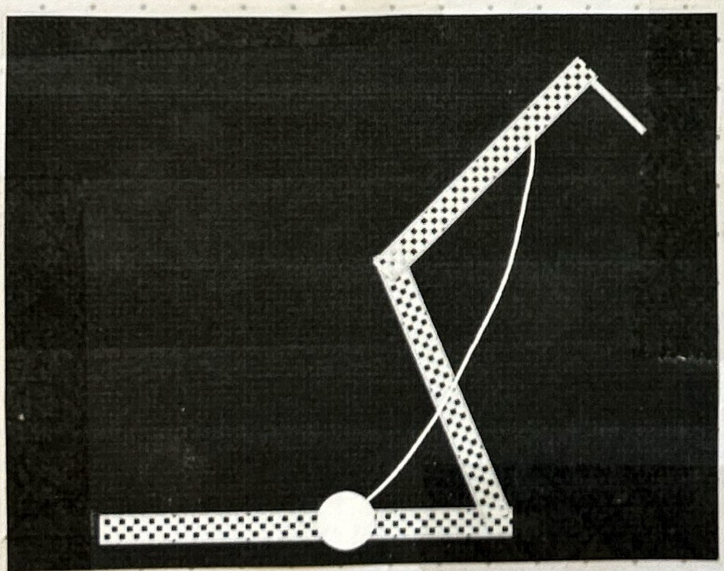




Designing Hang Mechanism:

Goals for today → Create a plan and rough sketch of our low profile hang mechanism design





This design was inspired by team 2029C's Robot Reveal video.

The idea is too have a compact mechanism that folds out tall enough to hook into the elevation bar and compress again to raise the robot.

The top image is the concept folded up made out of several c-channels side by side with the standoff hook tucked in allowing it to go under the goal.

Then, the 2nd image shows it unfolded through the help of pistons. A motor would then pull the string as shown. The white circle represents the pulley and winch system where the string will be wound up using the motor.

Future Plans:

Goals for today → Record what we have left to do before Worlds, but after Notebook Submission deadlines

Below are our next steps to prepare for Worlds.

1. Build the new hang mechanism and test and redesign as needed.
2. Make other minor robot changes such as improve efficiency of the intake.
3. Plan several Scrimmages with Hope Robotics and AGSI.
4. Improve programs.
5. Practice driving and recognizing double zoning.
6. Practice Interviews.

VP