

Engineering Notebook

999A

Team Number

Rising Phoenix

Team Name

Capital Robotics Club

9/1/2022

Start Date

00/00/0000

End Date

1

of 1

Book #

v1.0.8.29.22



Resources

students.vex.com

Engineering Resources, Information on Notebooks, Videos, VEX Library, Teams Resources, and Scholarships



mentors.vex.com

Team and Mentor Resources, Mentor Professional Development, VEX Mentor Community and more



teams.vex.com

A Collection of Resources for Teams Provided by the REC Foundation



library.vex.com

Information on Building, Documentation, Troubleshooting, Coding, and other Educational Resources



About the REC Foundation

The REC Foundation's global mission is to provide educators with hands-on, student-led competition programs and educational resources to prepare future innovators for a diverse and inclusive STEM workforce. We see a future where all students design and innovate as part of a team, experience failure, persevere, and emerge confident in their ability to meet global challenges.

engineering.vex.com

notebooking.vex.com

coding.vex.com

[Judging Rubric for Notebooks](#)

vex.com

roboticseducation.org

[VRC 2022-2023 Game - Rules & Game Video](#)

Send suggestions and comments about these Digital Notebooks and Digital Parts to notebooks@vex.com

Table of Contents

Page	Linked Project Slides	Date
4	<u>About Our Team</u>	
6	<u>Drivetrain</u>	
10	<u>Shooter</u>	
21	<u>Loader/Ramp</u>	
28	<u>Dispenser and Touch-down Mechanisms</u>	
38	<u>Strategy</u>	
43	<u>Gear Ratio</u>	
46	<u>Programming</u>	
57	<u>Parts List</u>	

About Our Team

Who Are We?

Sammy: I'm Sammy. I am one of the robot drivers. Some of my favorite activities are drawing and swimming.

Zoe:

Vaishali:

Vaishnavi:

Kristen: Hi! I'm Kristen. I am one of the team drivers. Outside of robotics, I also enjoy art and dance.

Drivetrain

The H Drive

Our drivetrain is currently a H Drivetrain consisting of 5 wheels, 3 motors, and a 2x20 side length. There are two motors for each side (Left & right) which are connected with 5 gears on each side. The

Drivetrain Picture:

shift wheel is placed in the middle of the drivetrain horizontally to help the robot move from side to side for convenience.

Project H-Drive

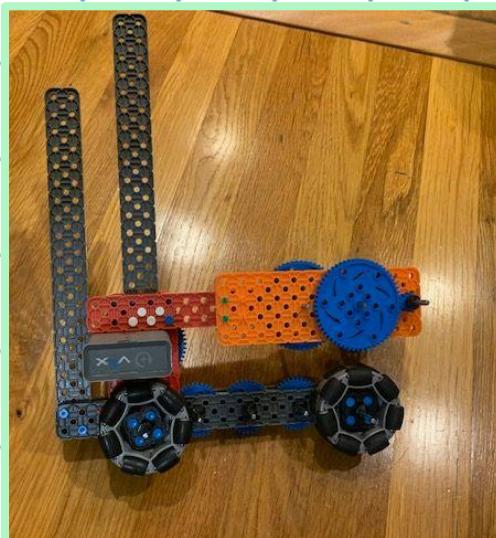
H-Drive continued...

Goal:

- Make our drivetrain a quality H-Drive to use in competitions

Problem:

A problem we encountered was to be careful, meaning we had to be very alert in order to put each small or big piece in the right position for the body of the robot to fit.



Project H-Drive Continued

Gears Instead of Chains?



usually, we will have two wheels on each side of the drivetrain that are connected with a chain. But, last year we discovered some problems: the wheels wouldn't drive straight. The wheels were very hard to control. So, after some observing, we learned that most other robots used gears to connect the two wheels (as you can see in the photo above). This year, we are testing to see if gears will be more reliable for us, so that it will be easier to drive and program the robot. Our hypothesis is that the gears will be more stable than the chain.

Project Why we switched to gears for drivetrain

Shooter

Our Goal!

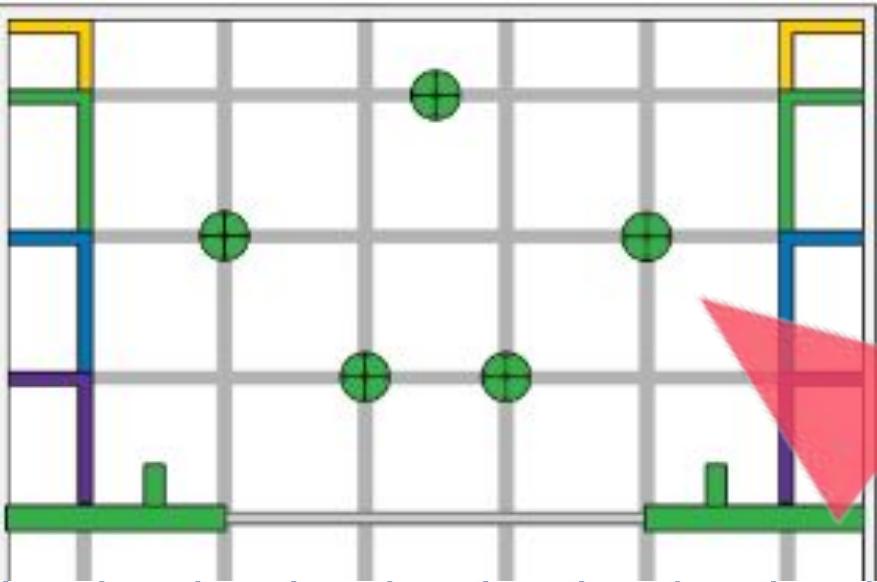
When we started, we knew we wanted a shooter that was efficient; it should be...

- easy to load
- easy to shoot
- shoot all pucks quickly.

We concluded that having a double shooter would help us meet these goals. This is because it's more efficient than one, and anything more than 2 would be too wide to fit between the green stoppers in the scoring zone.

To ensure the shooter is easy to load, we decided that there needed to be a wall around the shooter, so that the pucks didn't fall out. There needed to be a "ceiling" above it too to direct where the pucks will fall. We needed the pucks to fall flat into the shooters.

To figure out how many pucks go into each shooter, we did some math! There are 45 pucks, and most of them would stay on the ramp when there isn't space. We predicted that each shooter would hold 8-10 pucks.

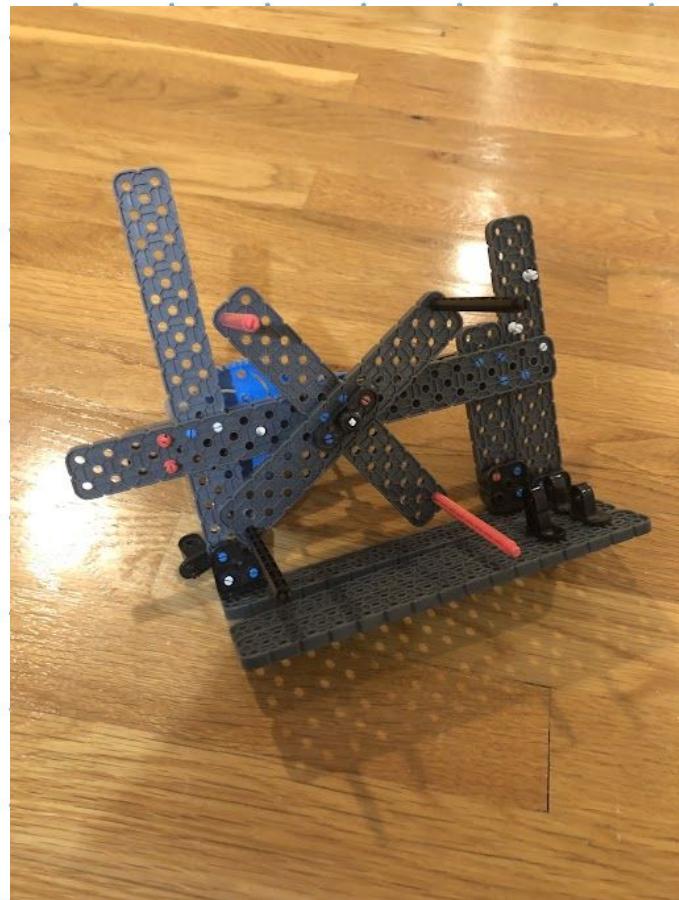


Project Shooter Goals

The Ferris Wheel

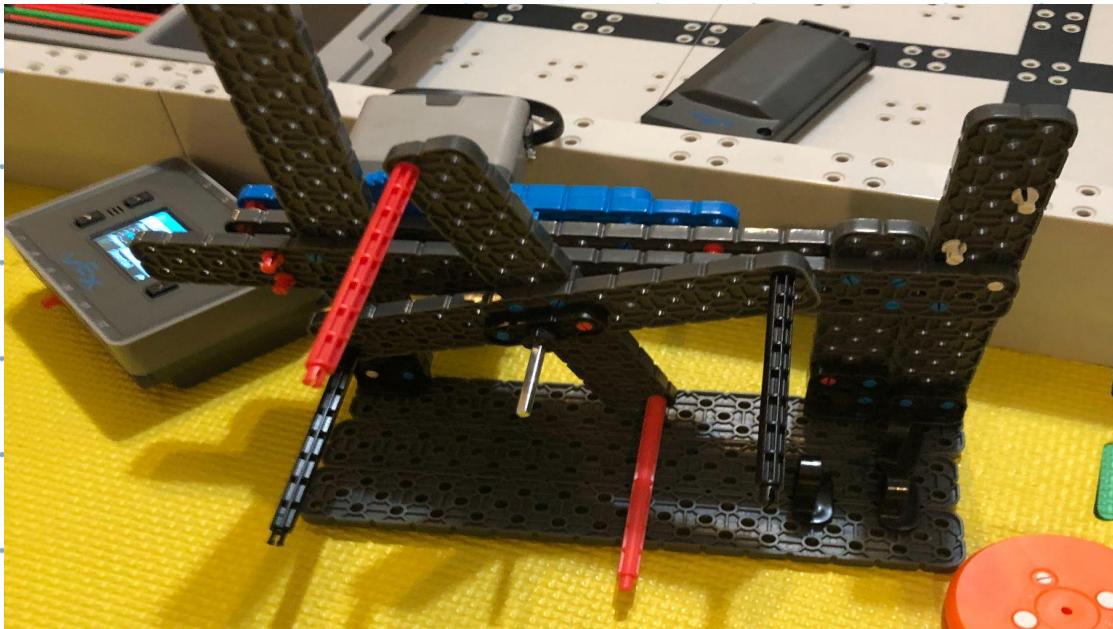
Design

During some of our first meets, we began to brainstorm some ideas for shooters. This design, that we named "the ferris wheel", was one of them. A motor is attached to a gear (gear ratio was 5:1), which is attached to the section that looks like a cross. The standoffs connected to the ends of the cross will hit the pucks. While the motor spins, the standoffs will also spin. The pucks get...



Project Beginning to design shooters

Ferris Wheel Continued



...knocked out
when the
standoffs spin.
Since there are
four standoffs,
four pucks will be
shot very quickly.

Why this design didn't work:

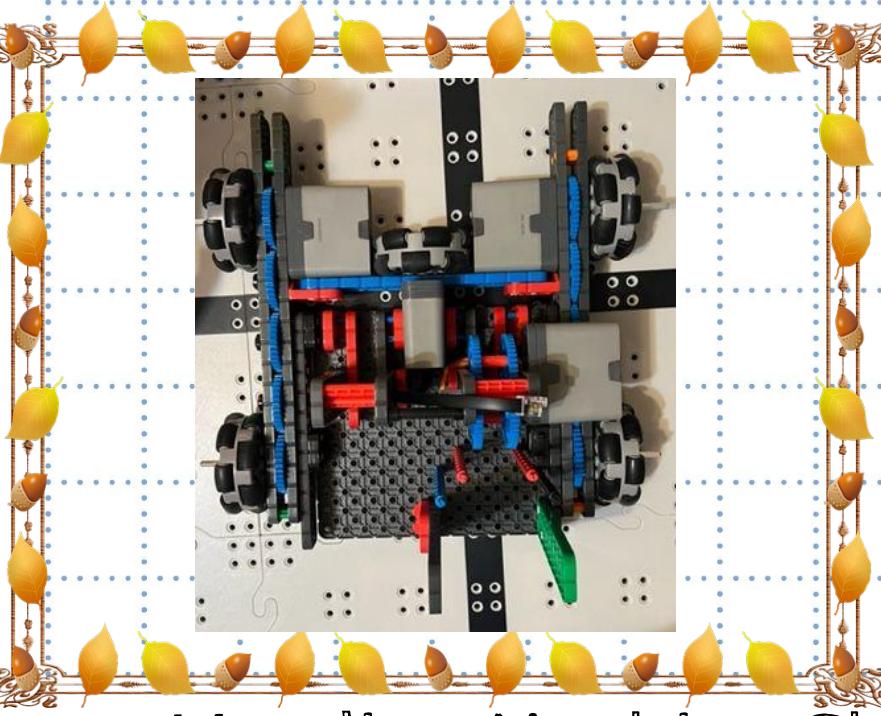
- It was difficult to find a way to load pucks into the mechanism.
- The standoffs didn't have enough power to push the pucks far enough.

The vending machine shooter:

The vending machine shooter is a simple mechanism that mainly consists of a single 2x beam, that repetitively goes back and forth shooting one puck at a time. There are two slots for two piles of pucks (Preferably 8 pucks each) and one shooter for each pile. The 2x beam would shoot the puck closest to the bottom until the stack is gone.

Problem:

A problem we came across was that we had to retract the 2x shooting beam in order for the pucks to be evenly stacked on top of each other. As tested with the beam underneath, the first puck would be slanted and would cause the pucks to be slanted; which would cause them to slide out of their position.

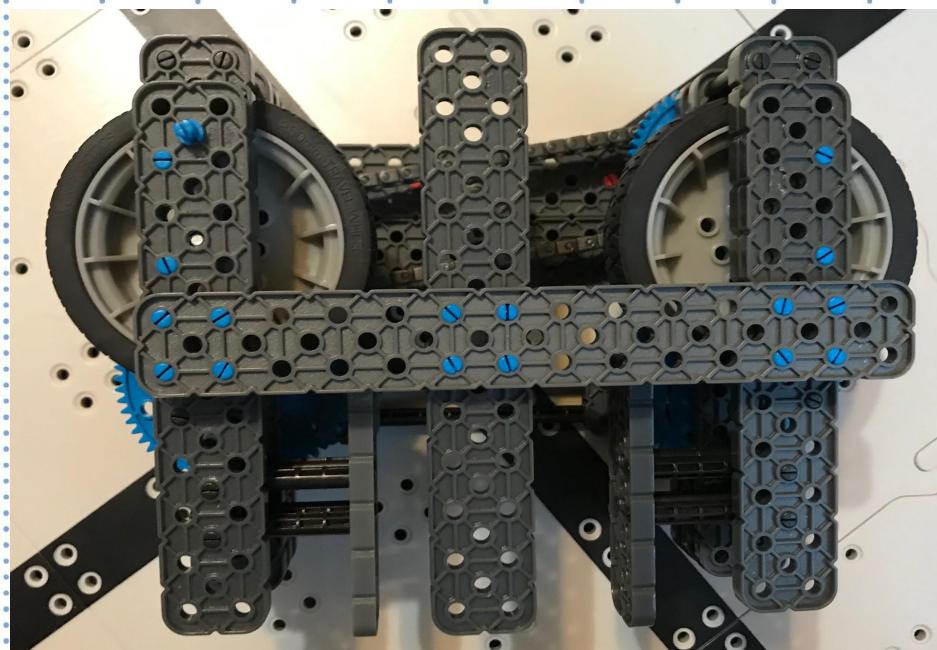
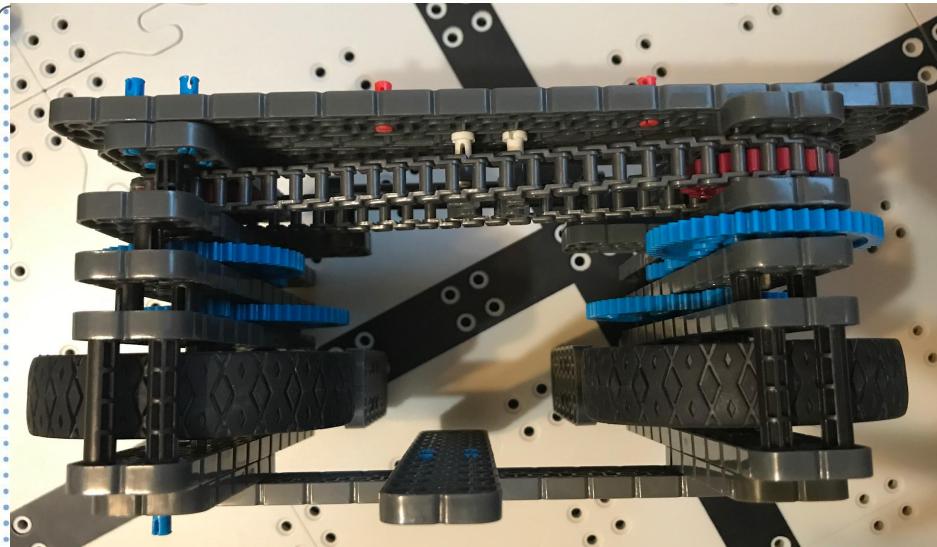


Solution:

The solution we decided to use was a rubber band retracting system which would stretch the beam back allowing more space for pucks to stack on top of each other.

Project Vending Machine Shooter

The Flywheel Shooter



Overview:

The flywheel shooter is composed mainly of two wheels. The wheels are spun and disks can be shot out of it. Since there is a 6 motor limit, then the flywheel is extremely clunky and complicated.

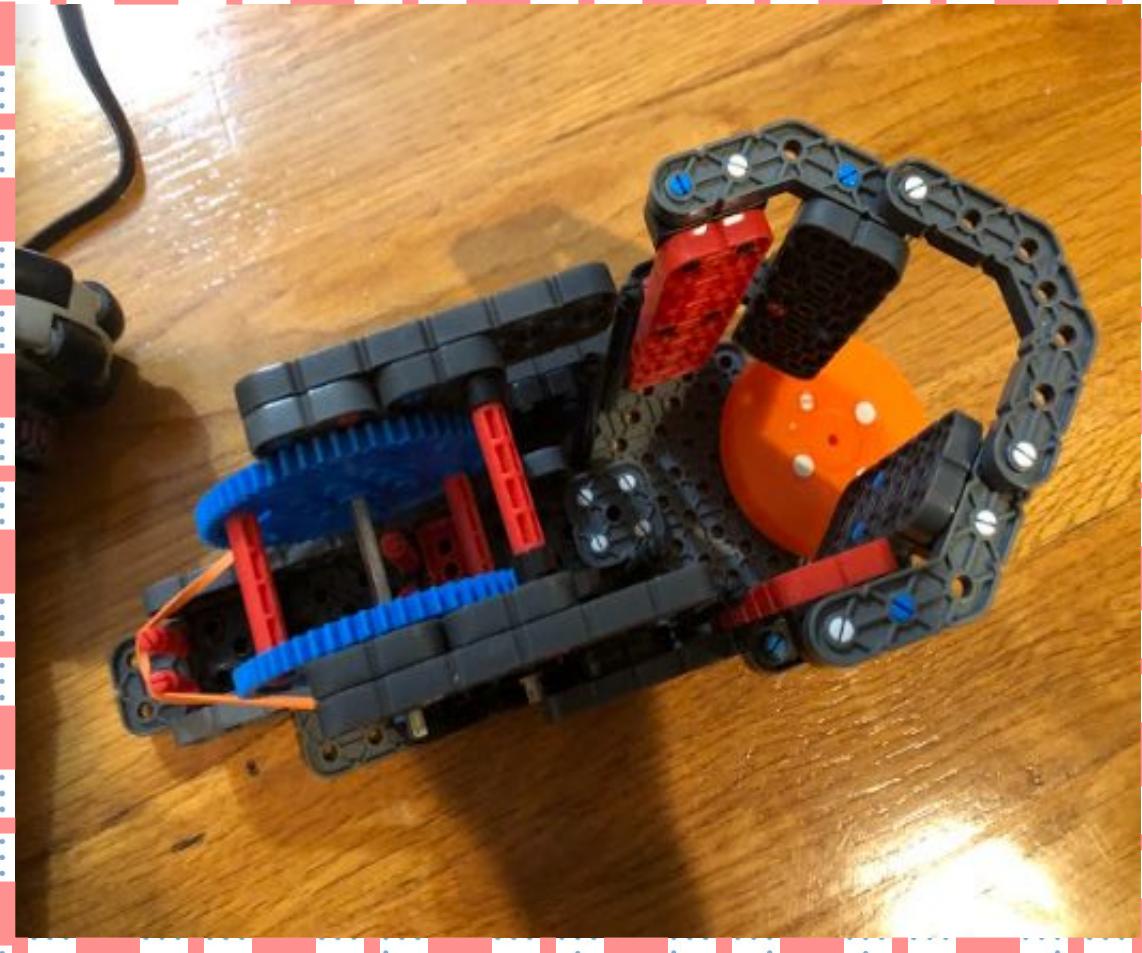
Why not?

The reason we decided to not go with the flywheel is that it took up to much space. There was also a smaller, simpler design we had.

Pro's and Con's

The main pro to this design is that it is extremely efficient at shooting out disks. The main con is that it is very large and takes up a lot of space on the robot.

Project The Flywheel Shooter



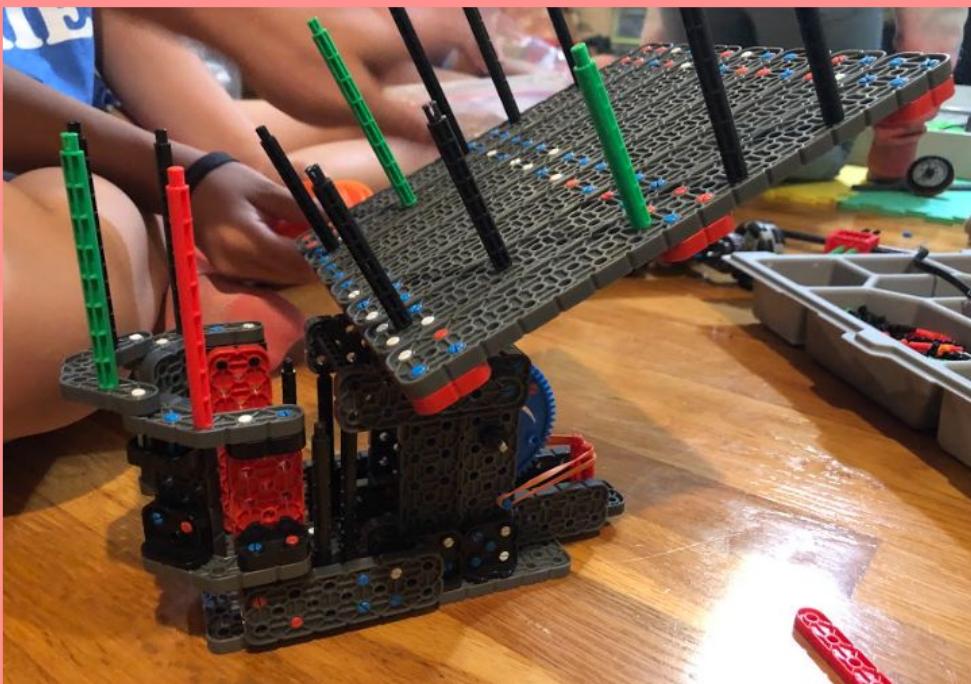
New Design!

This mechanism has a gear that pulls back and releases a bar. The bar then hits the pucks and propels it really far. The gear is attached to a motor, and it turns really fast.

Pros:

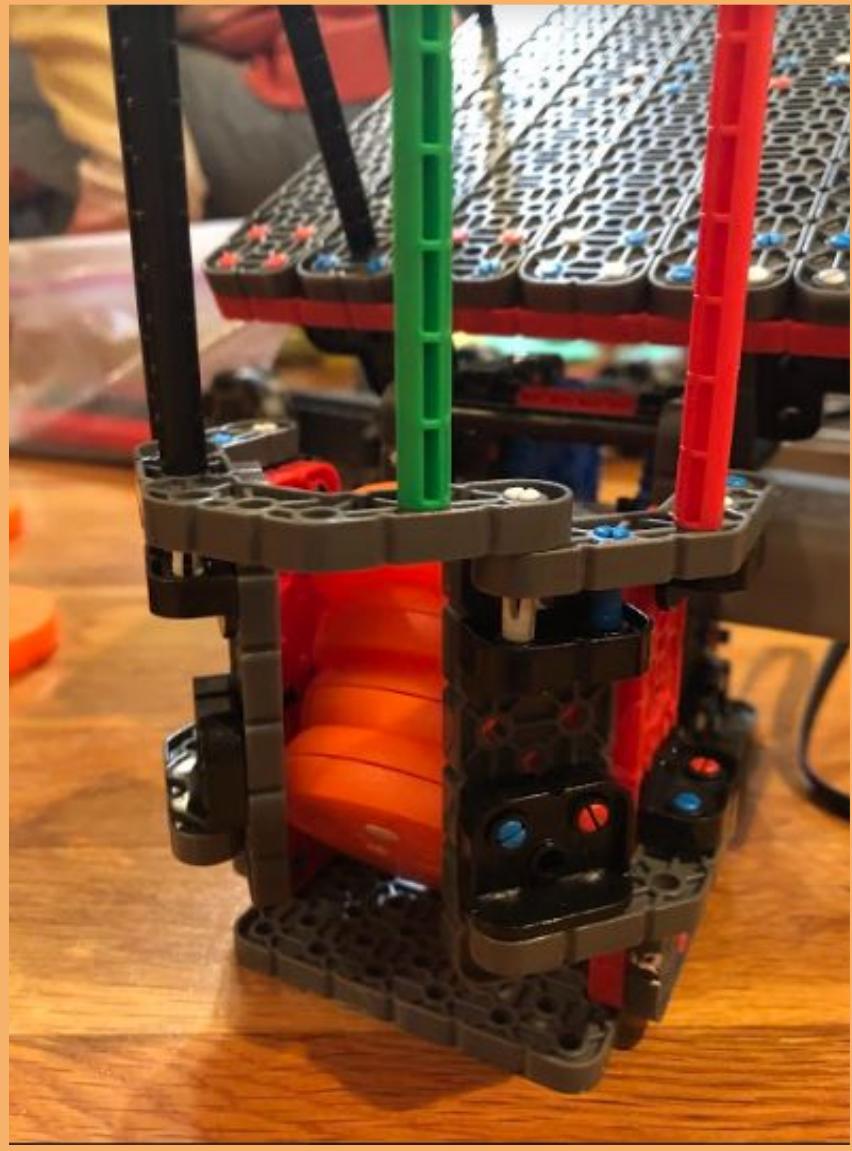
- Shoots pucks out really fast-very efficient
- compact design-and easily be changed to a double shooter.

Project Shooter Design



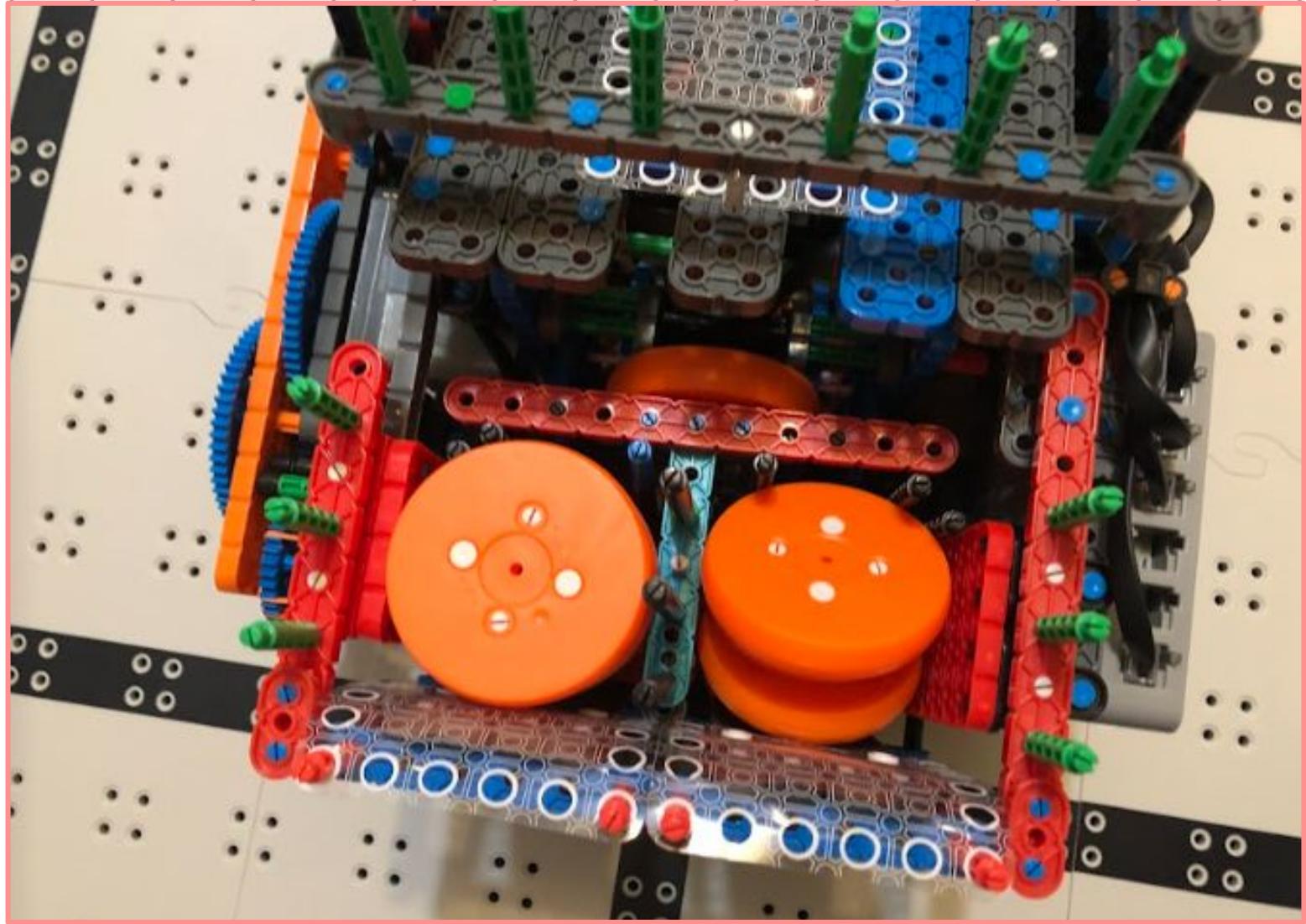
After trying the new design, and learning that it was successful, we attached the ramp we designed last week onto it. The ramp is attached so that pucks can easily slide into the shooter.

When we rolled the pucks in, we start discovering new problems. As you can see, the pucks wouldn't fall into the shooter flat, because the bar that is supposed to hit the pucks is rested right under the landing spot. This would mess up the whole shooting process, because the bar wouldn't be able to hit them as far. We needed to figure out a way to move the shooter bar out of the way.



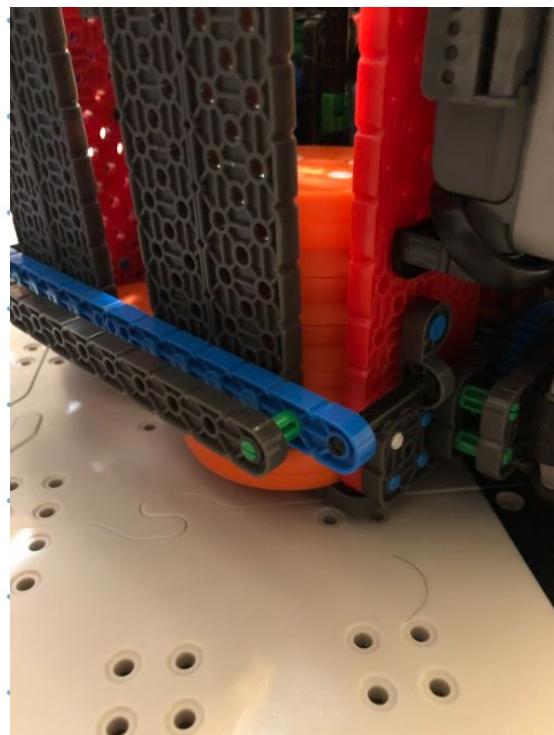
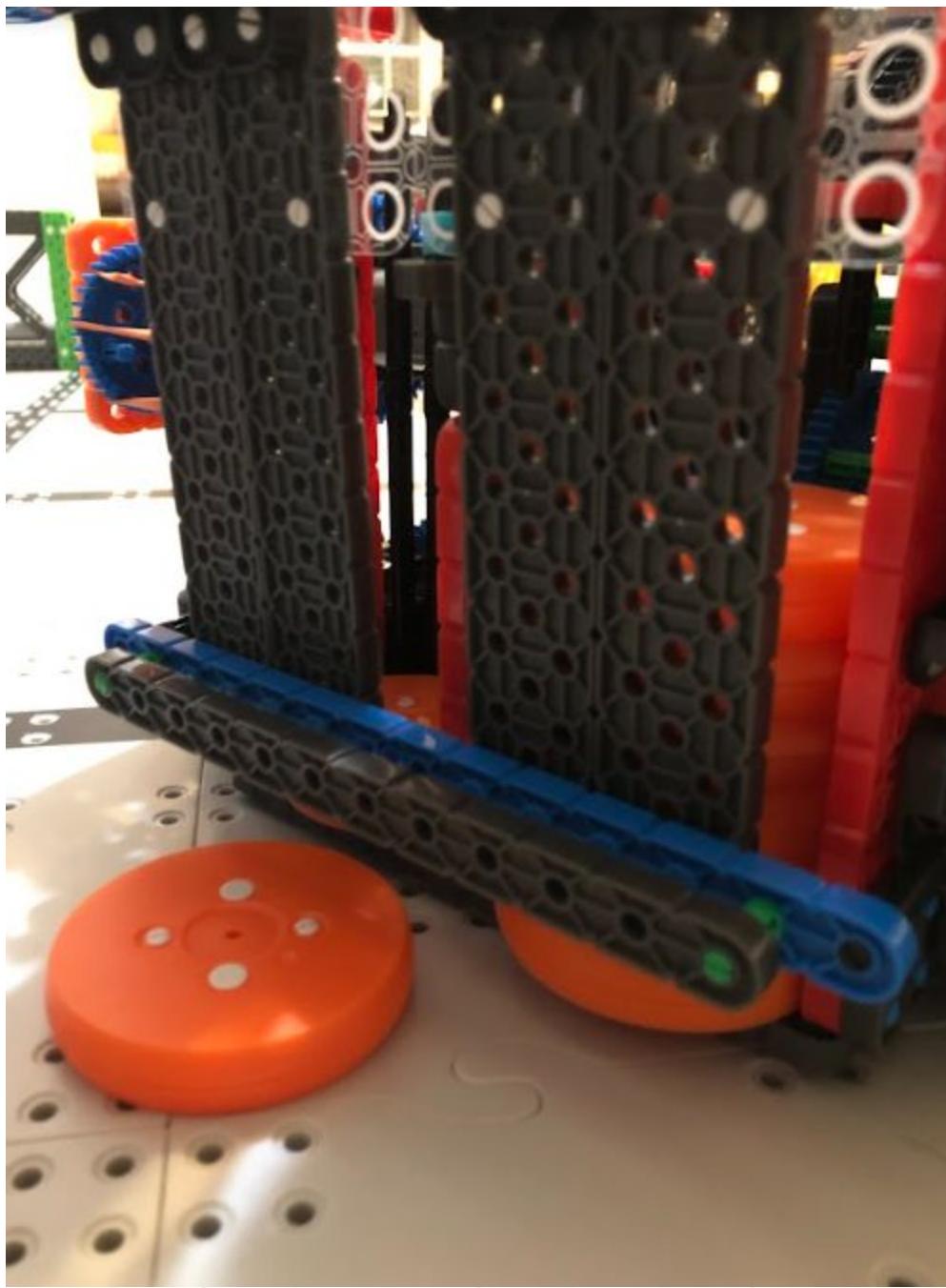
Project Shooter Design Continued

Problems We Encountered in Our Design: Part 1



The first problem that we noticed with our design is that the pucks don't fall into the shooter correctly. As you can see, in the right chamber the pucks fall in sideways, and not completely flat. When the pucks are sideways, the shooter isn't able to reach them to shoot, meaning we can't score any points. Sometimes, the pucks will also fall into the inside of the robot (as shown above). This is a problem because it clogs up the shooter, and there is no way of getting it out. Also, the middle section that divide the two chambers are not very sturdy. We plan to fix all of these problems so that we are more likely to reach our goal.

Project Problems with the current shooter



Problems We Encountered in Our Design: Part 2

Project Other Problems We Encountered

Name Kristen

Date 11/25

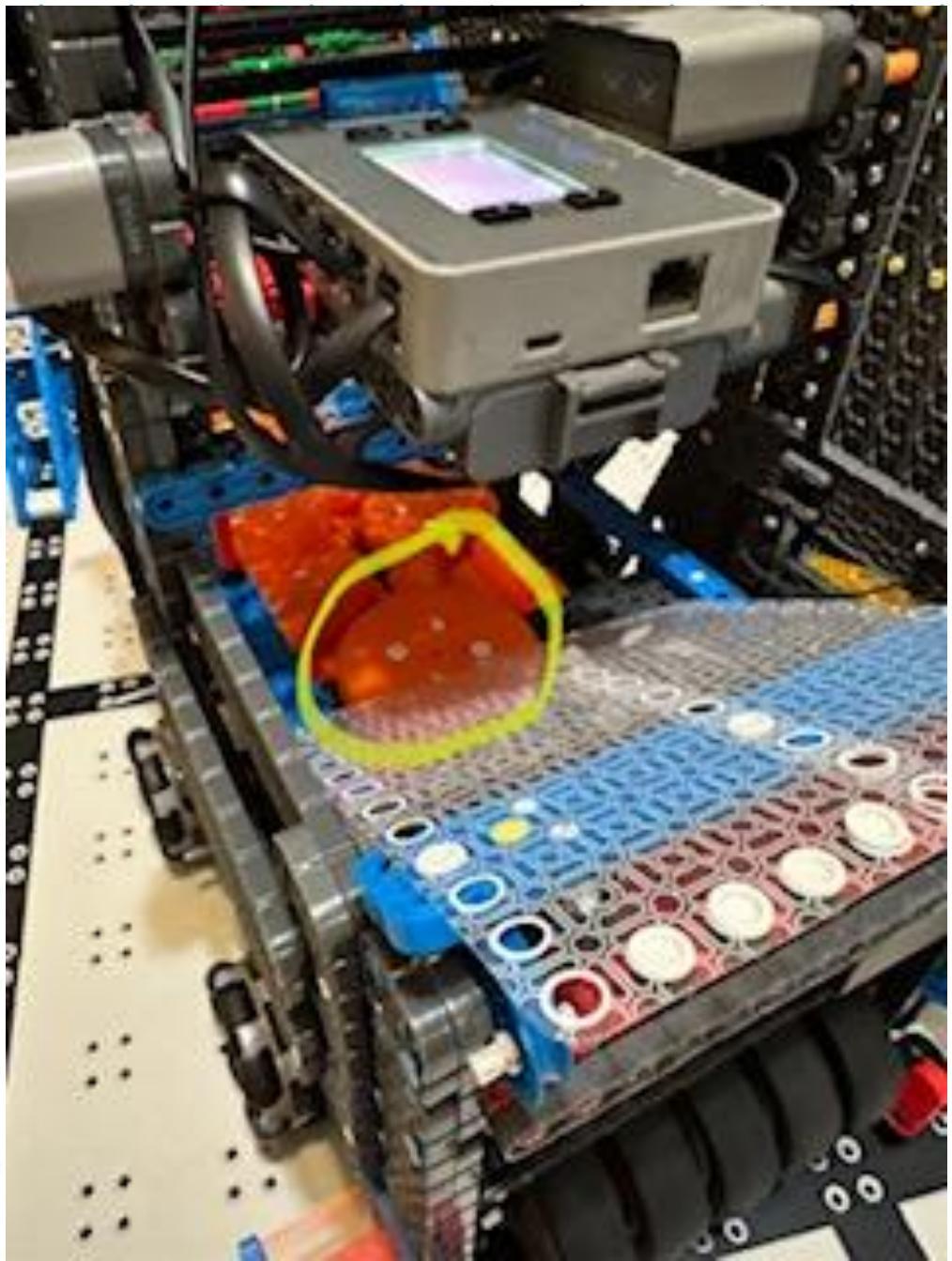
Page

19

Problems We Encountered

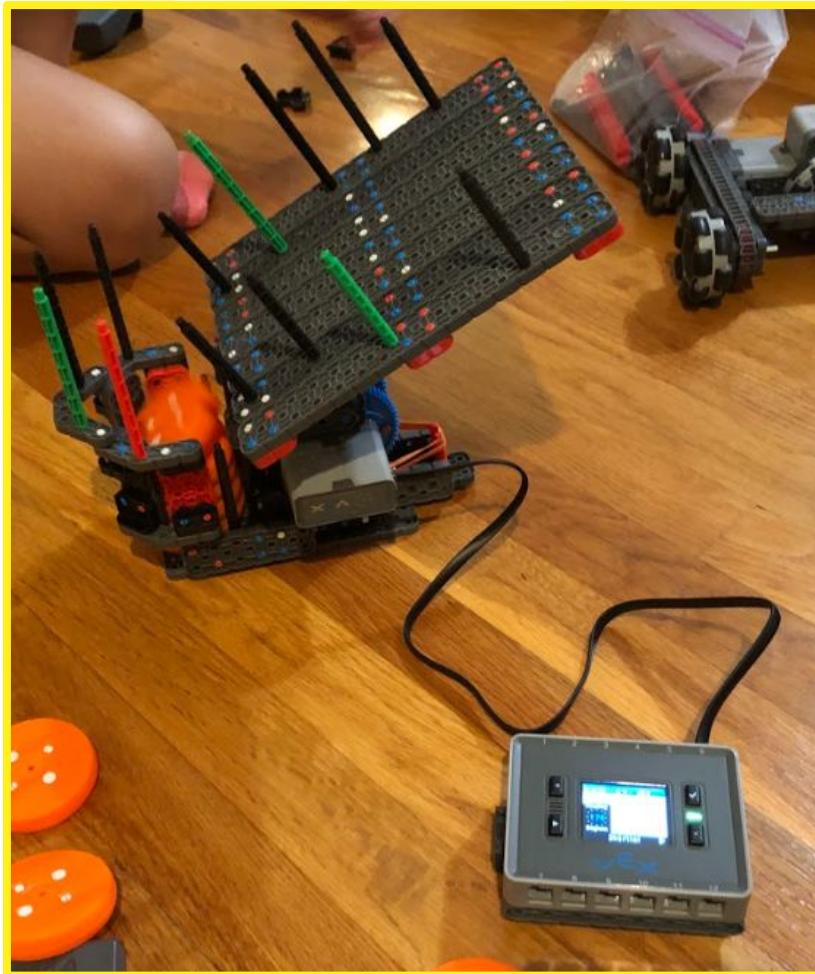
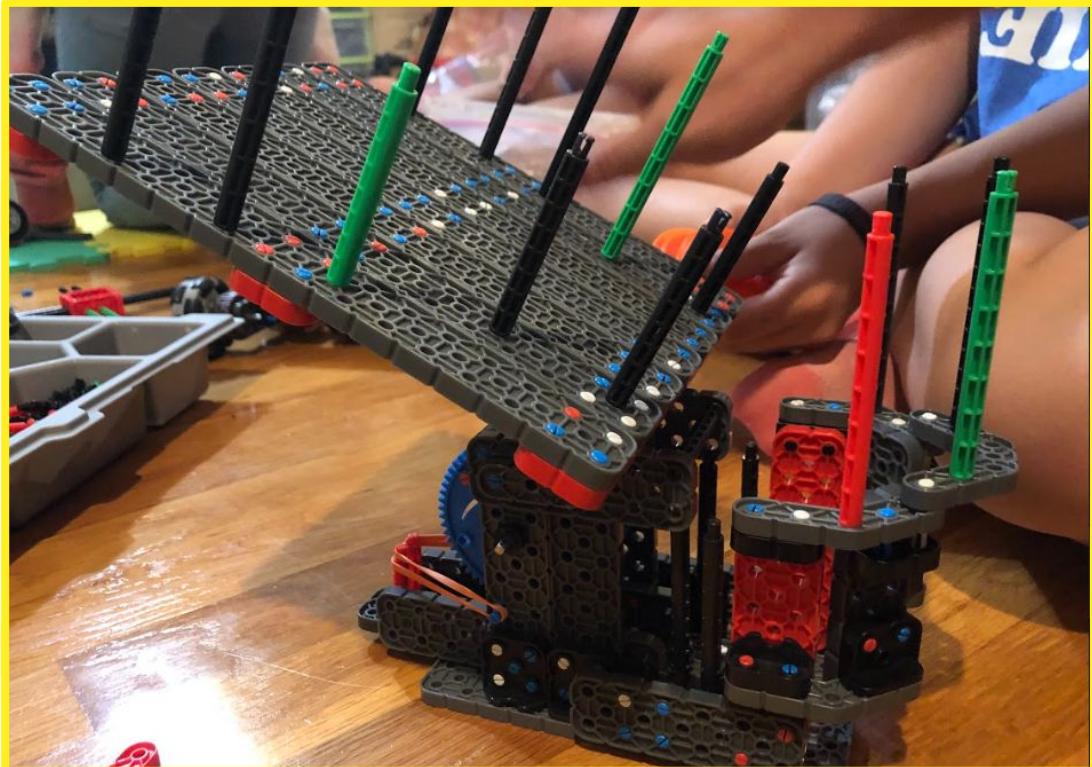
Part 3:

To the left, this photo shows an orange puck getting stuck on top of a plastic piece where it was not supposed to be. This caused the shooter to get jammed and the ramp could not move to shoot.



Loader/Ramp

FIRST RAMP DESIGN



This design was made up of many 2x bars. The stand offs were added to lead the pucks into the shooter. We were planning to have the pucks drop onto the ramp, then roll into the shooter. There were a few problems with this design:

- It was heavy/bulky
- The pucks usually didn't roll into the shooter properly ([see page 17](#)).
- The angle of the ramp wasn't adjustable

Project First Ramp Design

Conveyor Belt Design

This design picked up pucks and then transported them to the ramp by using pulleys that looped around.

This designs goal was to efficiently and easily pick up multiple pucks at a time to deliver to the ramp.

Although, there were a few problems with the design.



PROBLEMS

- It was hard to build around the design to protect the pucks at the sides (from falling off)
- Large
- Hard time picking up pucks

Project Conveyor Belt Design

Conveyor Belt Design Pt.2

The primary explanation we decided against implementing the design were related to some technical issues with the edges as well as its overall bulk and shape. There was nowhere to set it up that would function well. The pucks wouldn't be able to enter if we blocked off the sides.

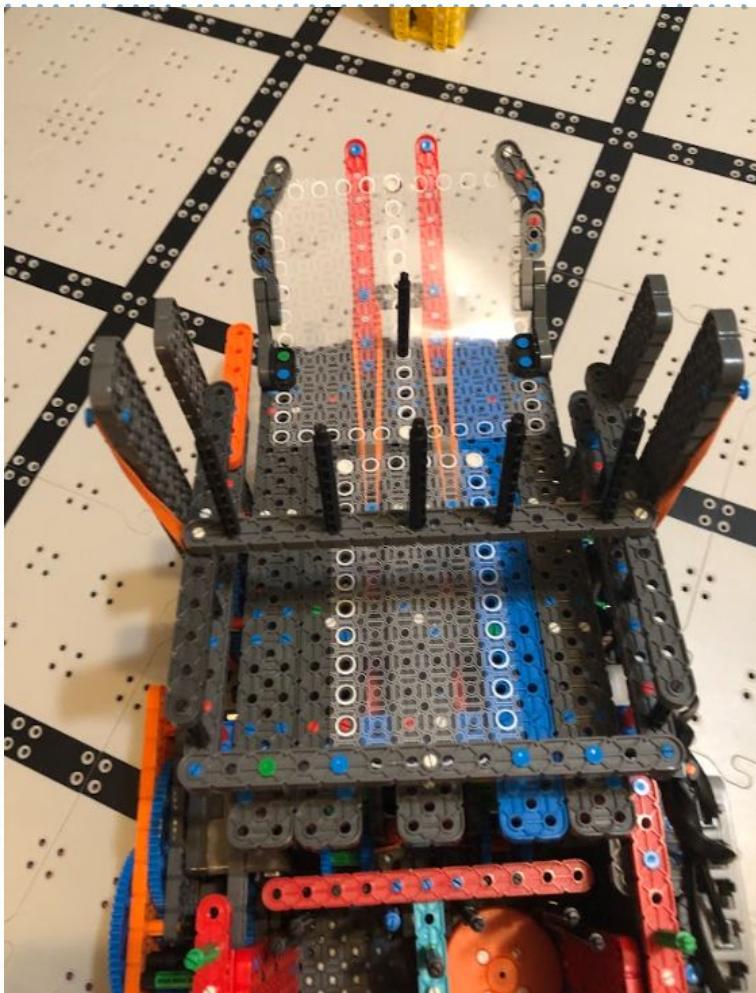


PROBLEMS

- Hard time fitting the pucks into the conveyor belt
- Difficulty adjusting the sides of the conveyor belt, and managing their sizes
- A problem occurred when looking at the sides of the conveyor belt, the pucks fell off at the side.
- There was no room for it; non-efficient.
- slow

Project Conveyor Belt Design

Sturdier Ramp Idea

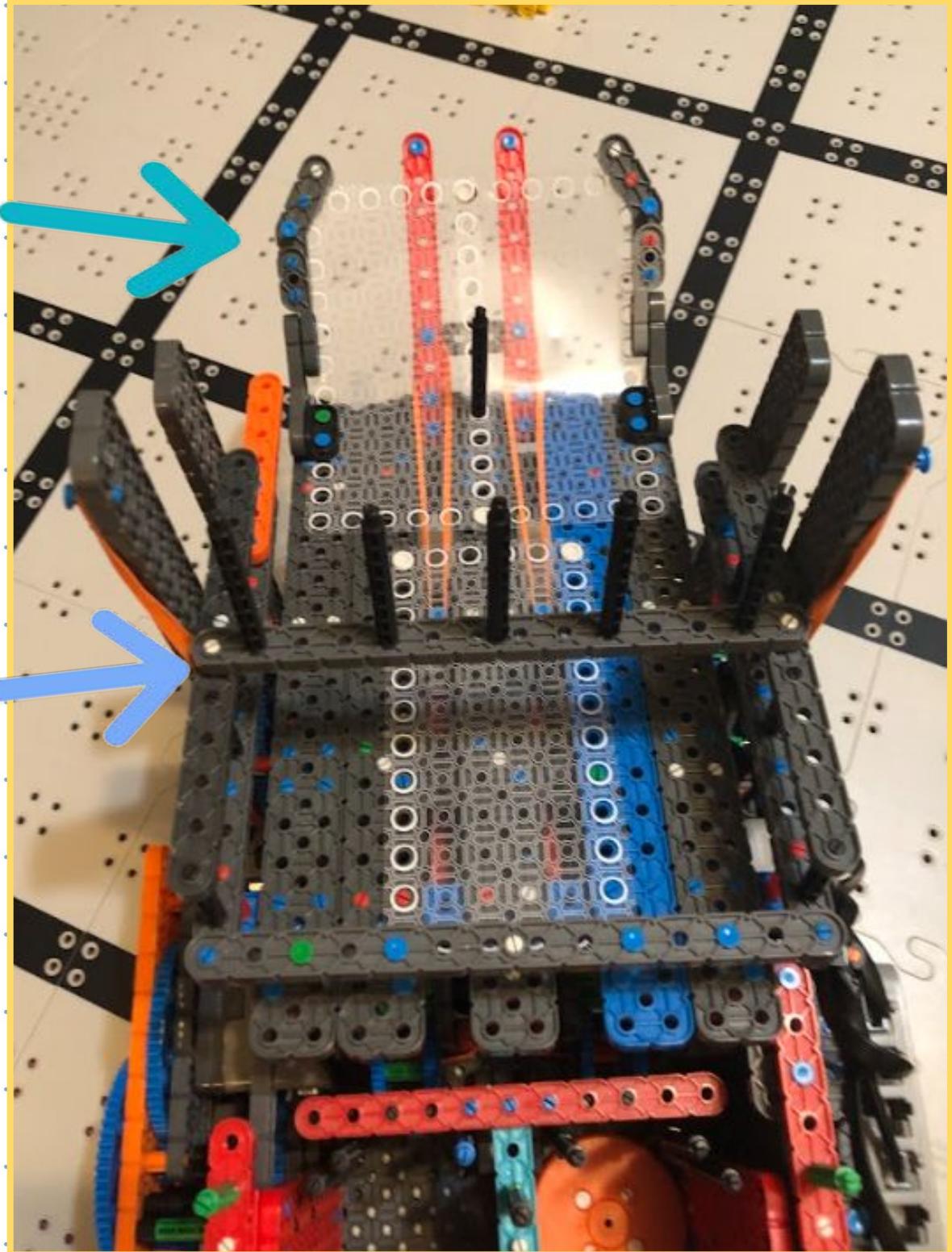


Before, our ramp design was not ideal, because it was flimsy. This design is sturdier than the first ramp we made. The beams are supported with more pieces, so that it can't be moved easily. Previously, the ramp would wobble from side to side, but now the ramp is stationary and more consistent. This ramp will be better than the old one because pieces won't easily fall off and the pucks will not roll off the ramp.

Project Sturdier Ramp Idea

Keeping the Pucks Flat:

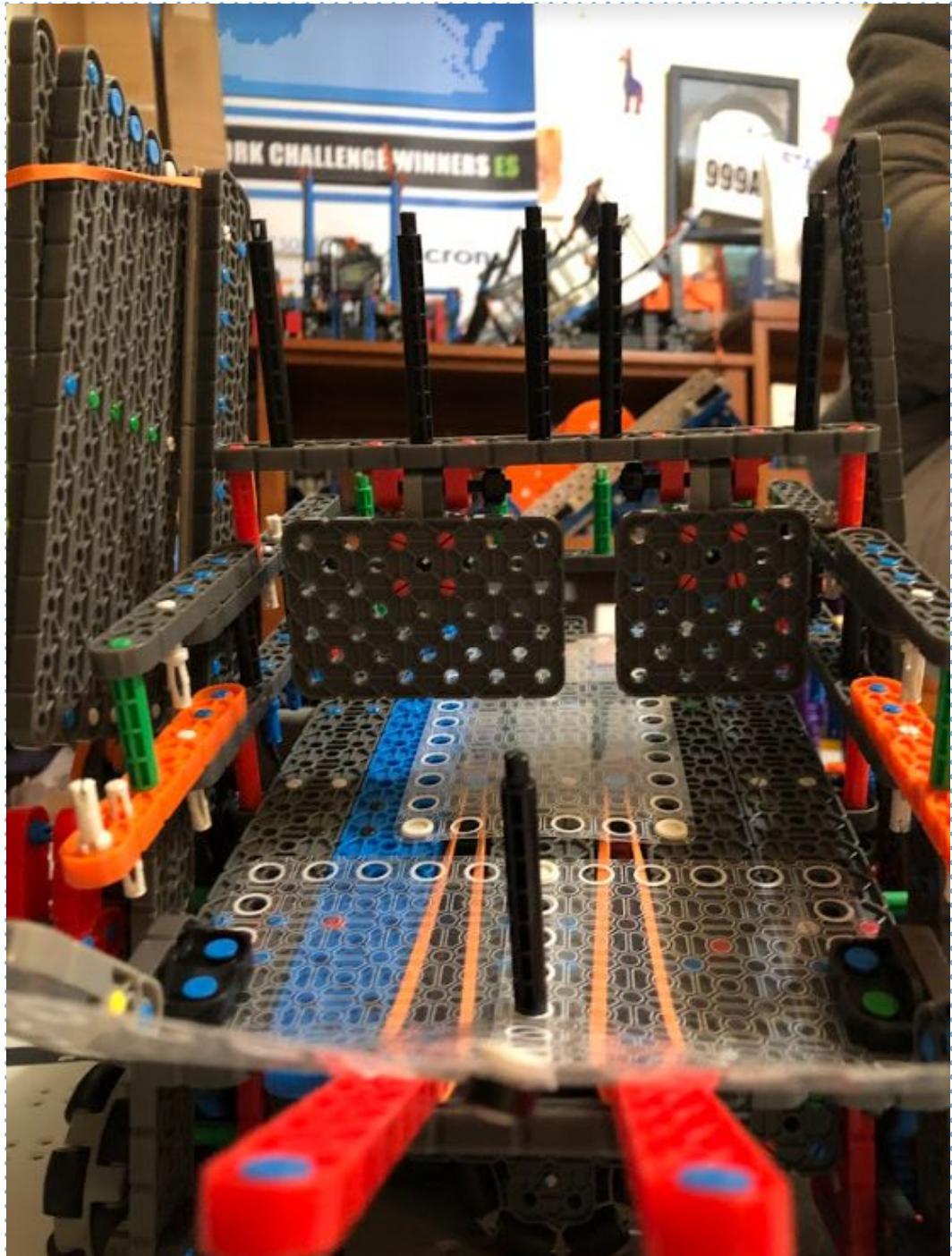
When the pucks roll down the ramp, sometimes they would fall into the shooter section vertical. So, we added a stopper bar to keep them horizontal. That way, the pucks would land flat and it would be easier to shoot them.



Project Adding a Bar to the Ramp

Some more new additions:

1. Last week we added something to keep the pucks flat. But, we soon learned that that wasn't enough. So, we added another section that works like a flap. This section also slows down the speed of the pucks. When they are loading into the shooter, they don't all rush into it at once.
2. This little addition was to help the ramp hook onto the yellow dispenser. We designed it so that the pucks will roll directly onto the ramp.

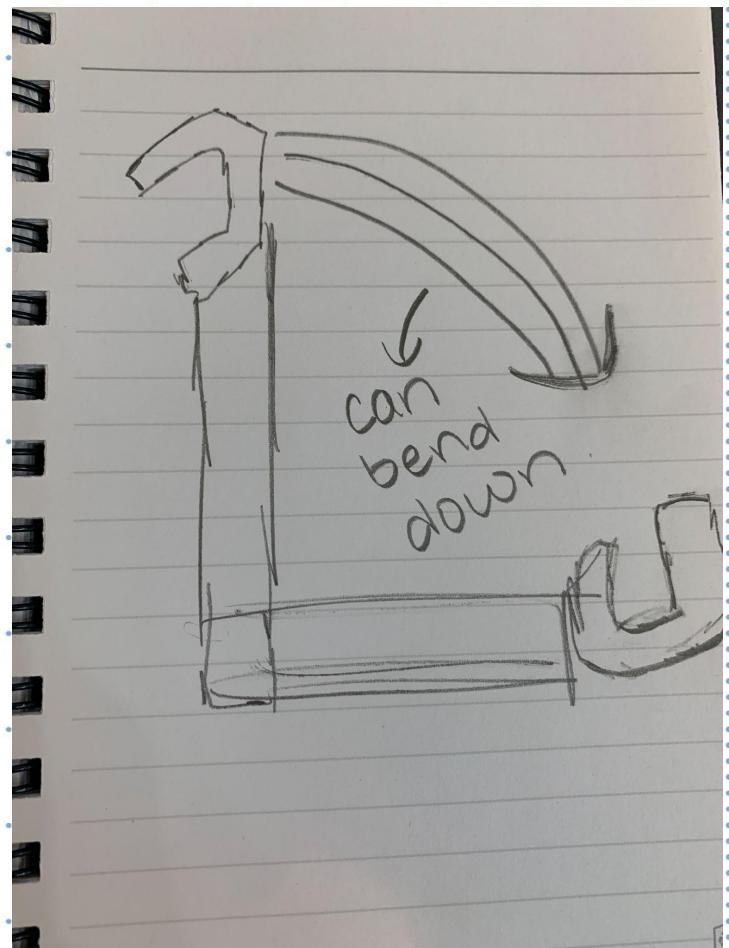


Project Additions to the ramp

Dispenser and Touch-down Mechanisms

HOOK for Yellow Dropper/Blue dispenser

To the left, is a quick sketch from the last robotics meet describing a hook that can release the yellow and blue dispensers. This hook is effective because it allows us to use one motor for two dispensers.



Problem:

A problem we encountered was that the hook was very weak and not as reliable as we thought, which made us innovate our idea to make it stronger/lightweight at the same time.

Project Yellow/blue dispenser hook

Hook for Yellow Dropper



This hook design is made to drop the pucks from the yellow dropper. We plan to put it near the bucket. The robot drives behind the yellow dropper, then goes forward to hook onto it. The hook will have enough force to pull back the dropper so that pucks can fall into the robot. But the hook's rubber band tension is designed so that it can easily slide out of the dropper's edge.

Project Designing a hook for yellow dropper

2nd Hook for Yellow

Dropper

Here is another hook we designed for the dropper, but instead of a hook, there is a wheel. We tried using a wheel so that it can easily roll into place. The problem was that since it rolled forward and back, the wheel didn't have the force to pull out the dropper. We tried using new pieces that only turn one way, but it was hard to attach the pieces. Overall, this design doesn't work yet.



Wheel

Project

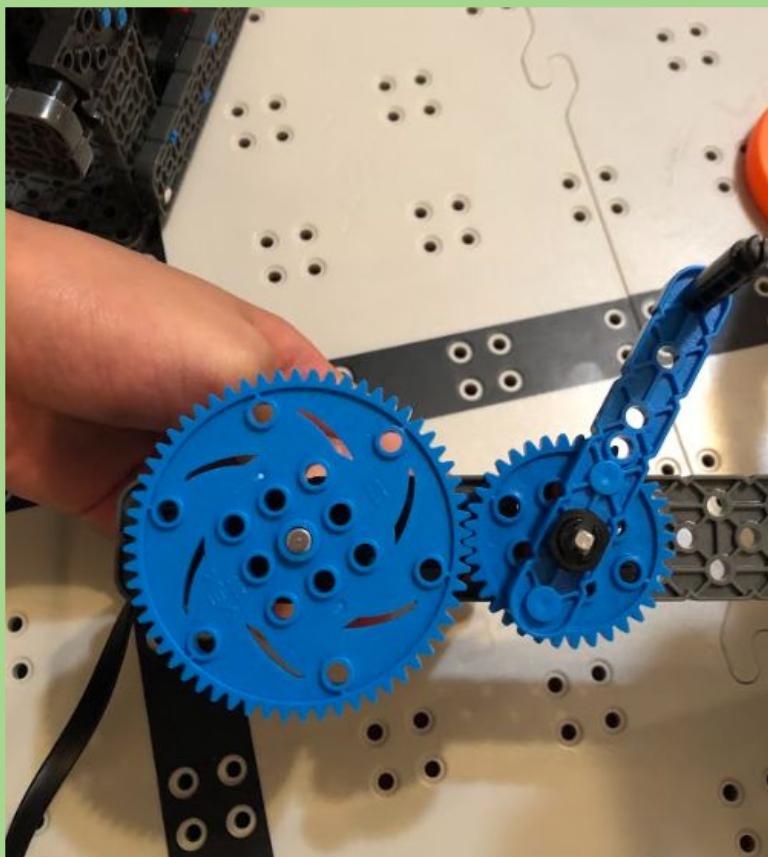
Designing a hook for yellow dropper

Name Kristen

Date September 2022

Page

31



Hook for Purple Dispenser pt 1

This hook is a standoff attached to a gear. The gear ratio of the mechanism is 3:1; making it faster than without a gear ratio. The standoff is placed in between the holes of the dispenser, and spins to allow the pucks to fall.

Problems:

- Might be hard to aim into the holes.
- Since it's on the side of the dispenser, the pucks might not fall into the robot's bucket.

Ideas for Improvement:

- Somehow making it be on both sides of the robot only using one motor to be easier for the drivers



Project Hook for Purple Dispenser

Hook for Purple Dispenser

pt 2



The Idea:

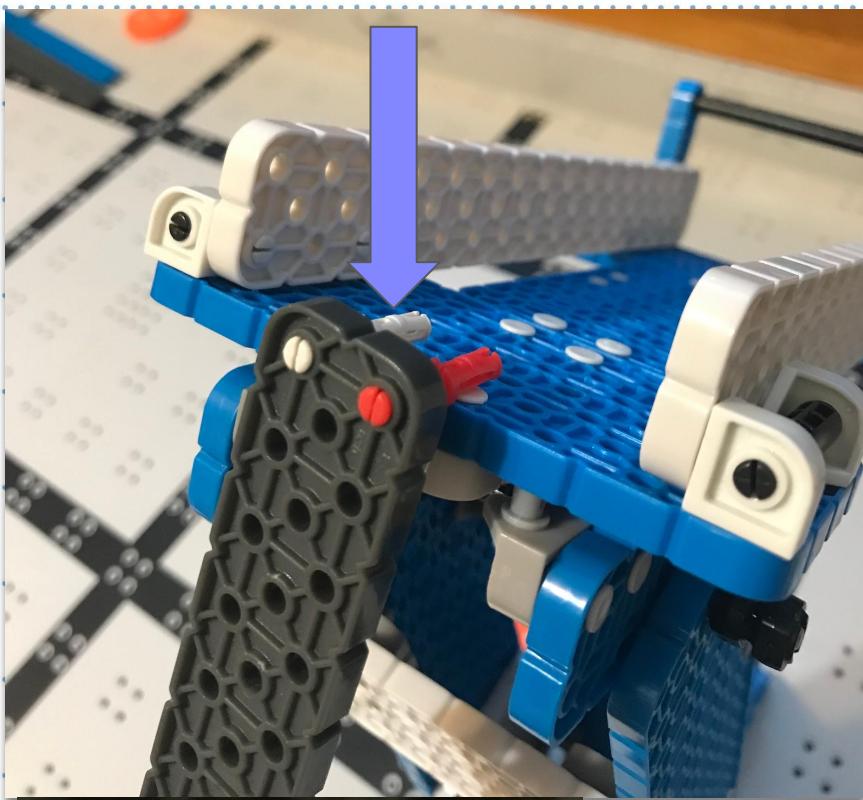
- Using our knowledge from last year, we designed a rubber band roller that turns from a motor. The grip from the rubber bands will turn the dispenser.

Problems:

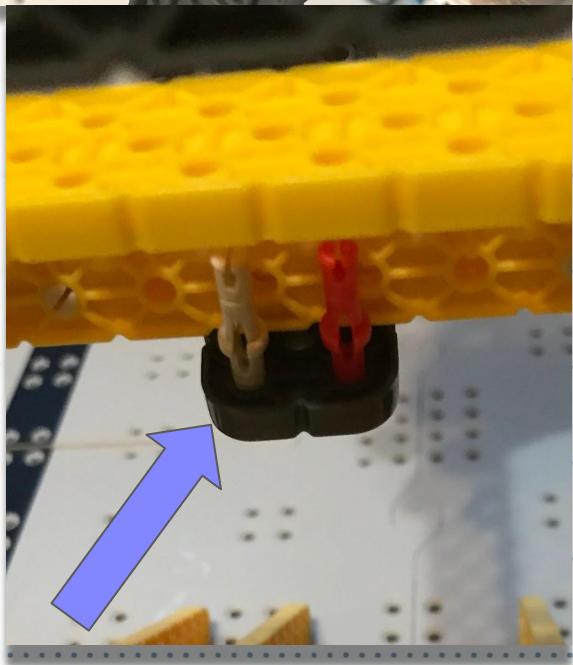
- Was a very slow design. We couldn't gear it up because the rubber bands on the gear were getting the gears stuck, and making it slower.
- May be hard to attach to robot.

Project Hook for Purple Dispenser

The Double Hook



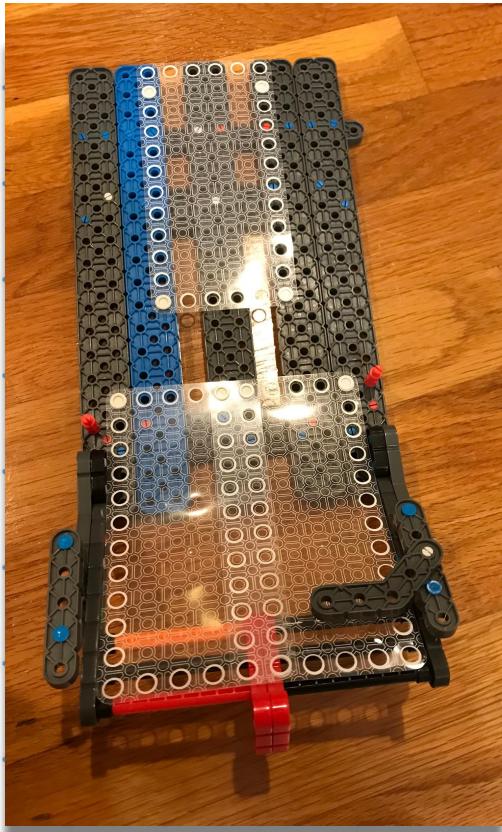
The idea for this hook is very simple. In the pictures to the left, there is a beam with pegs sticking out of the end.



The way the design works is the hook pushes down for the blue dispenser. On the yellow dispenser, it goes under, moves up, and then the robot drives back. By driving back, the yellow dispenser will release the disks.

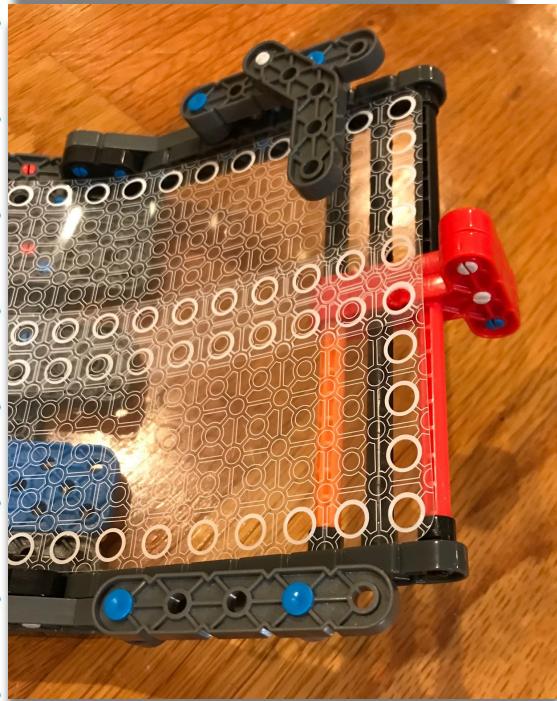
Project Hook for Yellow and Blue Dispenser

The Double Hook 2.0



This design works just like the previous double hook. The only difference is that it is attached to the disk ramp and it is hooked on both sides. This allows the hook to hook down onto the blue disk dispenser and also hook the other way onto the yellow disk dispenser.

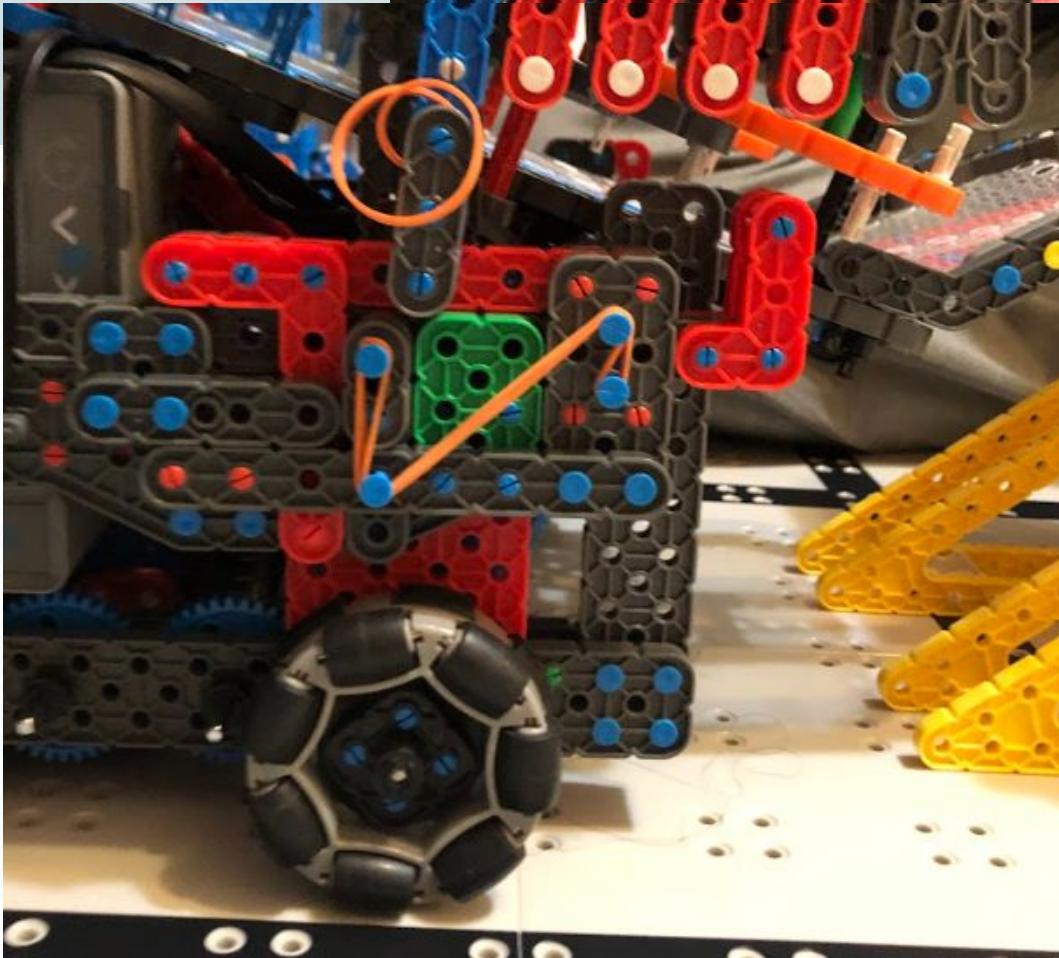
Pros



This design is extremely simple and effective. It also takes up very little space and can be easily incorporated into the robot.

Project Hook for Yellow and Blue Dispenser 2.0

Working on the Touchdown



When the small red "L" piece comes into contact with the wall of the divider, the mechanism will shoot out onto the green section of the board. This mechanism will only be used at the end of each run. During the designing process of this mechanism, we realized that it wouldn't be very good if something were to accidentally hit the "L" piece, and release the long arm. So, we added extra locks (kind of like baby locks) that make sure they the arm goes out at the wrong time.

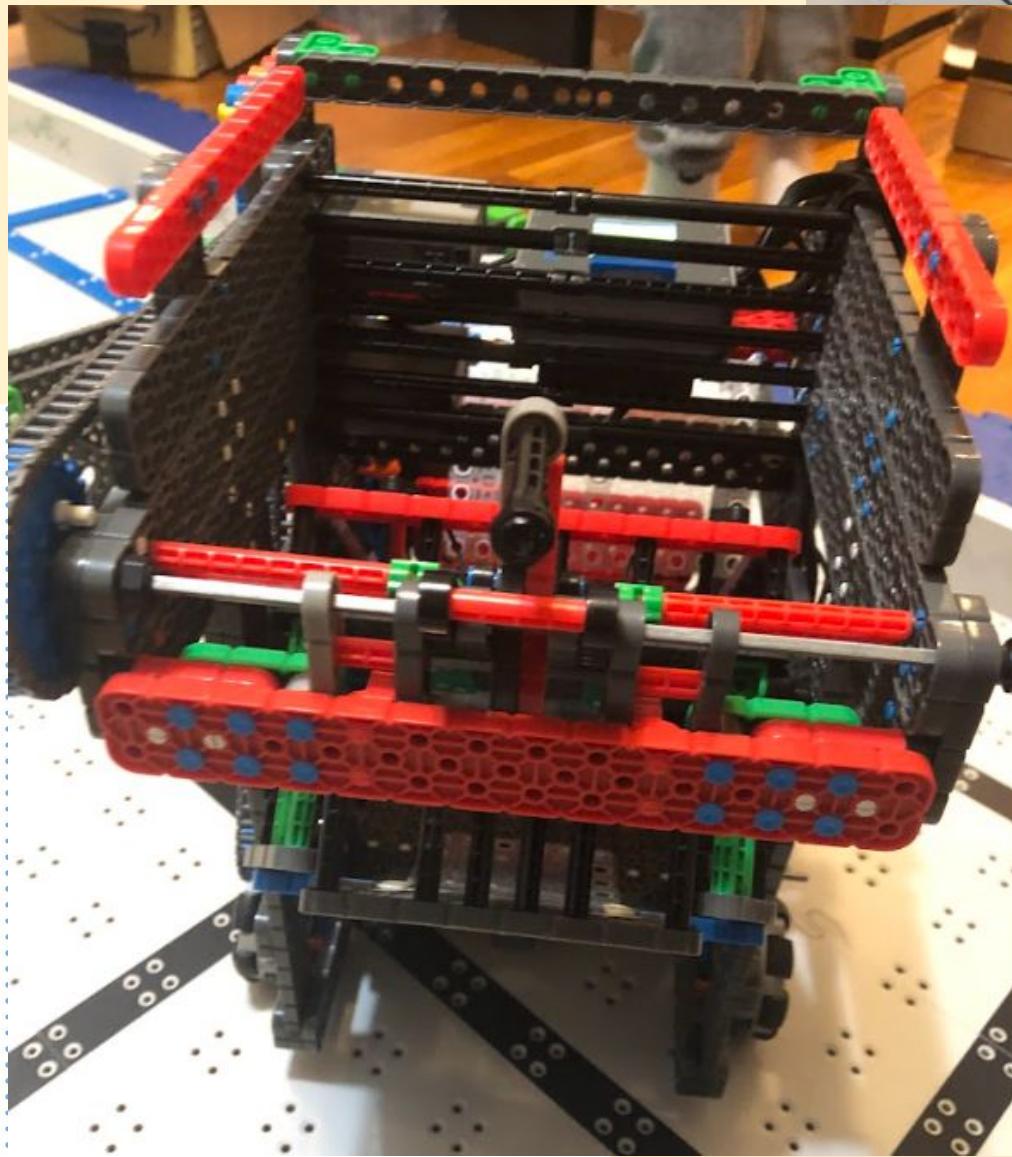
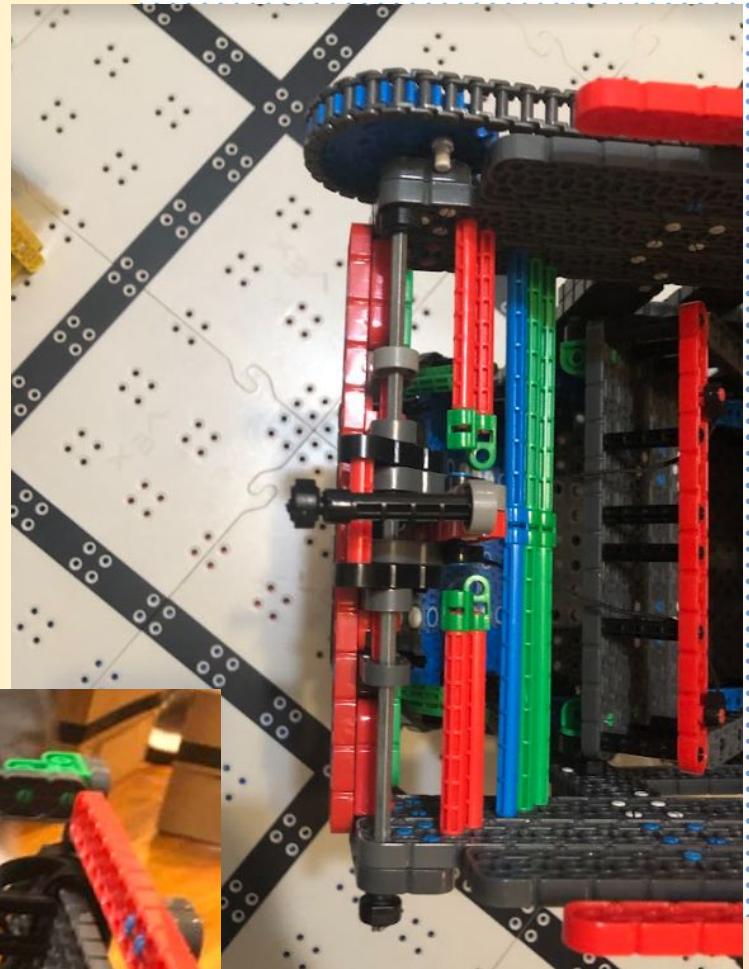
This is how the lock works:

1. Hit "L" piece
2. Moves pieces that are connected to it out of the way
3. Leaves a gap the releases the arm
4. Arm goes out with the tension of the rubber band.
5. (arm is not retractable)

Project Working on the Touchdown Mechanism

A BETTER HOOK DESIGN!

This hook design is much lighter than our previous one. Last time, we encountered the problem of the hook not be sturdy enough—it would sometimes be hard to control.



So, we made this design to save space, and make it more reliable. Also, it was easier to access the pucks on the blue dispensers due to the angle it and direction it moves.

Project Better Hook Design

Strategy

Second Generation!

This year, we decided to use the new second generation pieces to build our robot. When looking through them, we saw many familiar pieces, but also an abundance of new ones. We are excited to test out the new pieces and figure out how they work. Also, the new brain, controller, and batteries were really cool too. We were also pleasantly surprised to hear about the new coding platform we will be using this year. In the past few years, our team has been using Robotc, but this year, we will use a variation of the c++ language on VEXcode.

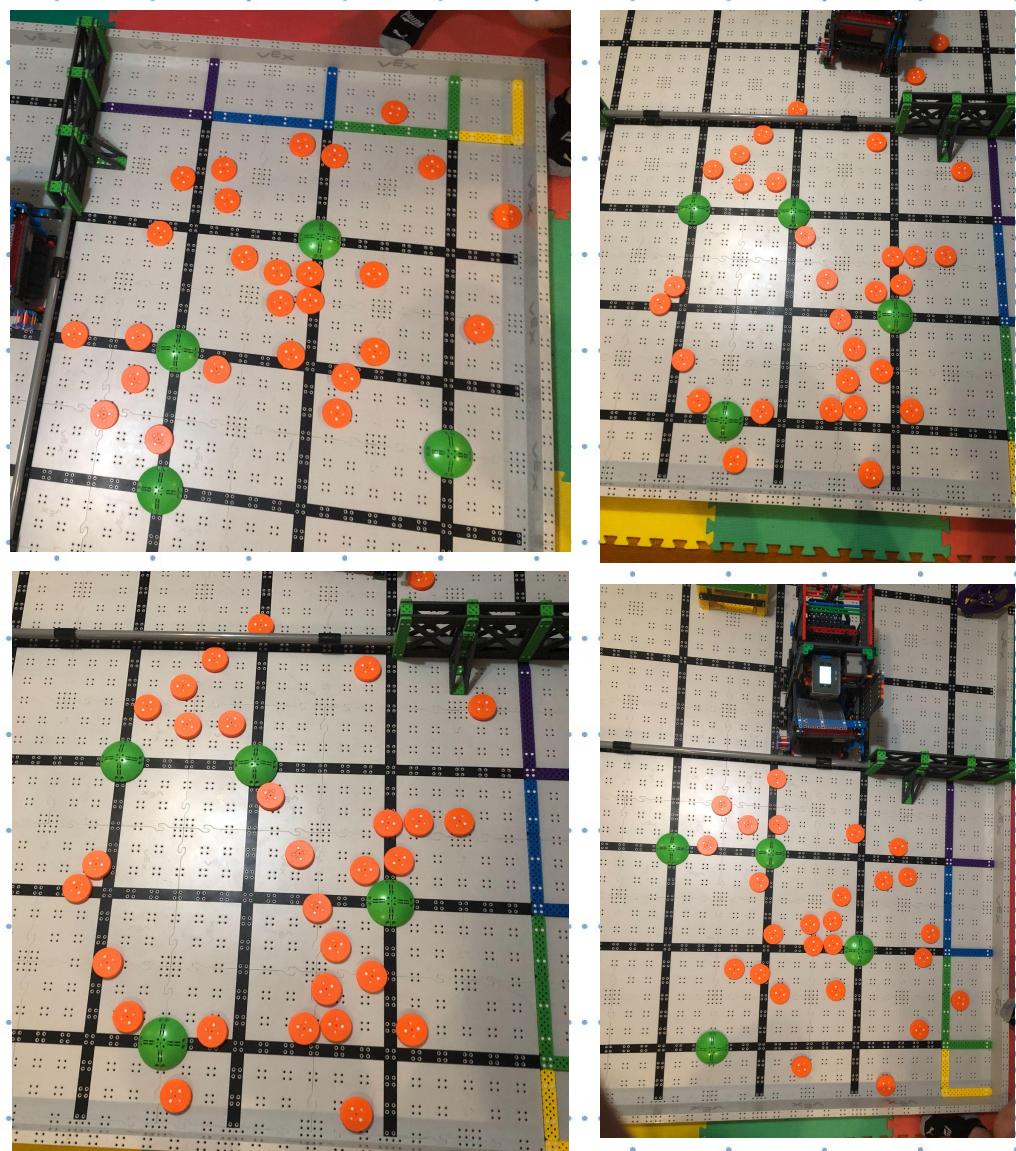


update: While driving our new robot, we noticed that the new batteries last much longer than the old ones. We don't have to charge them as often!

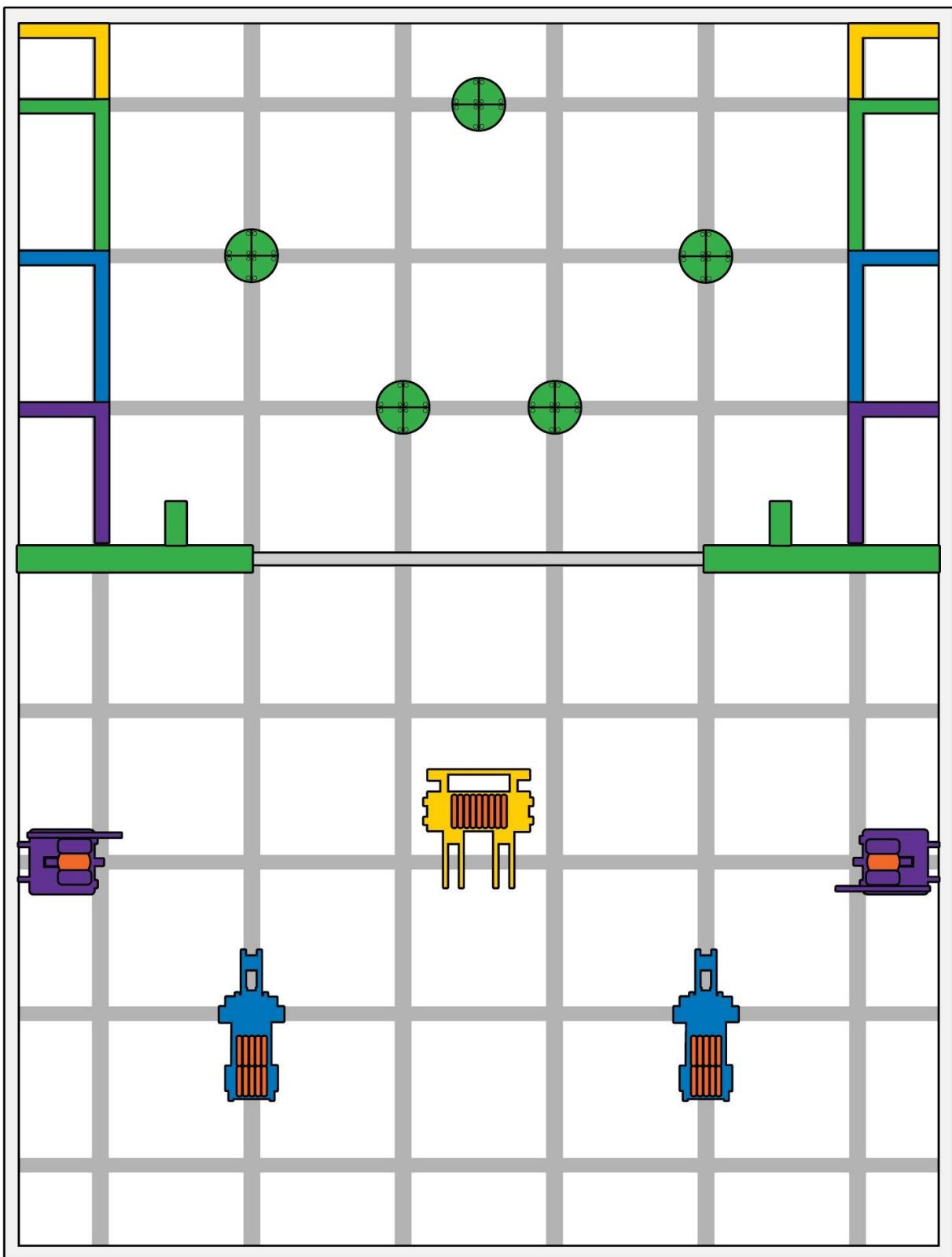
Project **Trying out gen 2 pieces**

After building the prototype of the robot, we started testing it out. Here are the results we got after our first few test runs!

As you can probably tell, the outcomes of the runs were never consistent. Sometimes we would hit the green stoppers on the board, making the pucks super closed together, while other times the pucks were super spread apart. This could be because of the lack of practice, so we plan to keep working hard. Our goal is to have most of the pucks in the green/4-point section.



Project First tests runs

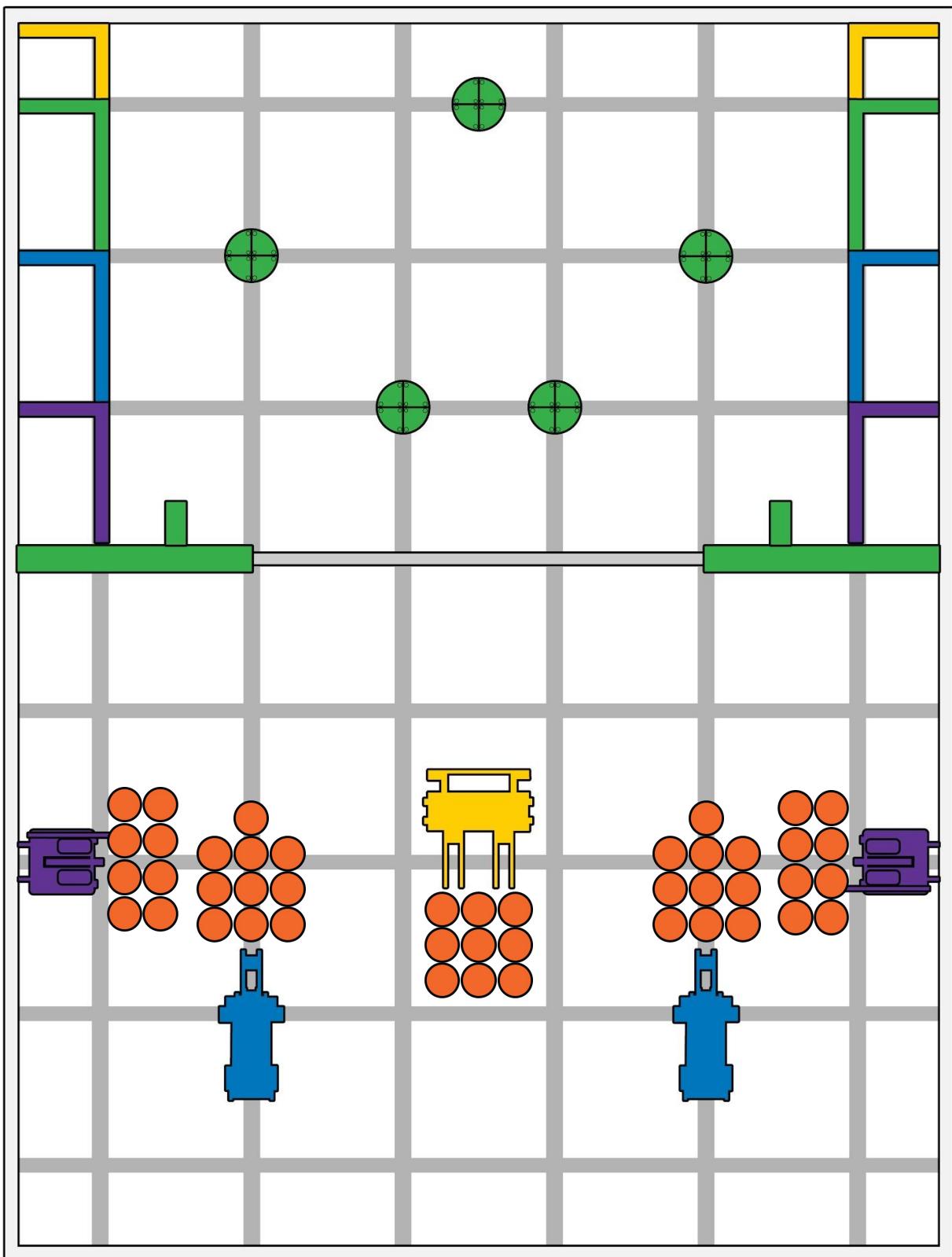


Project.....

Name.....

Date.....

Page 41

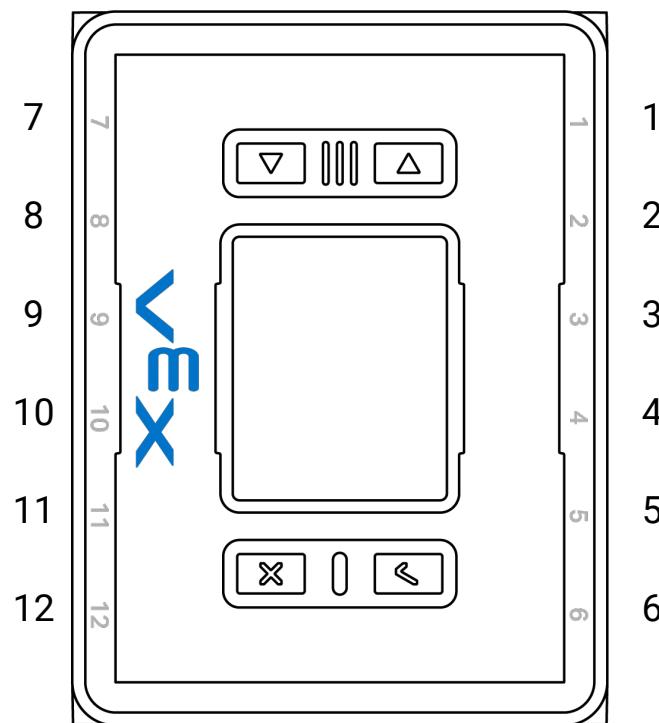
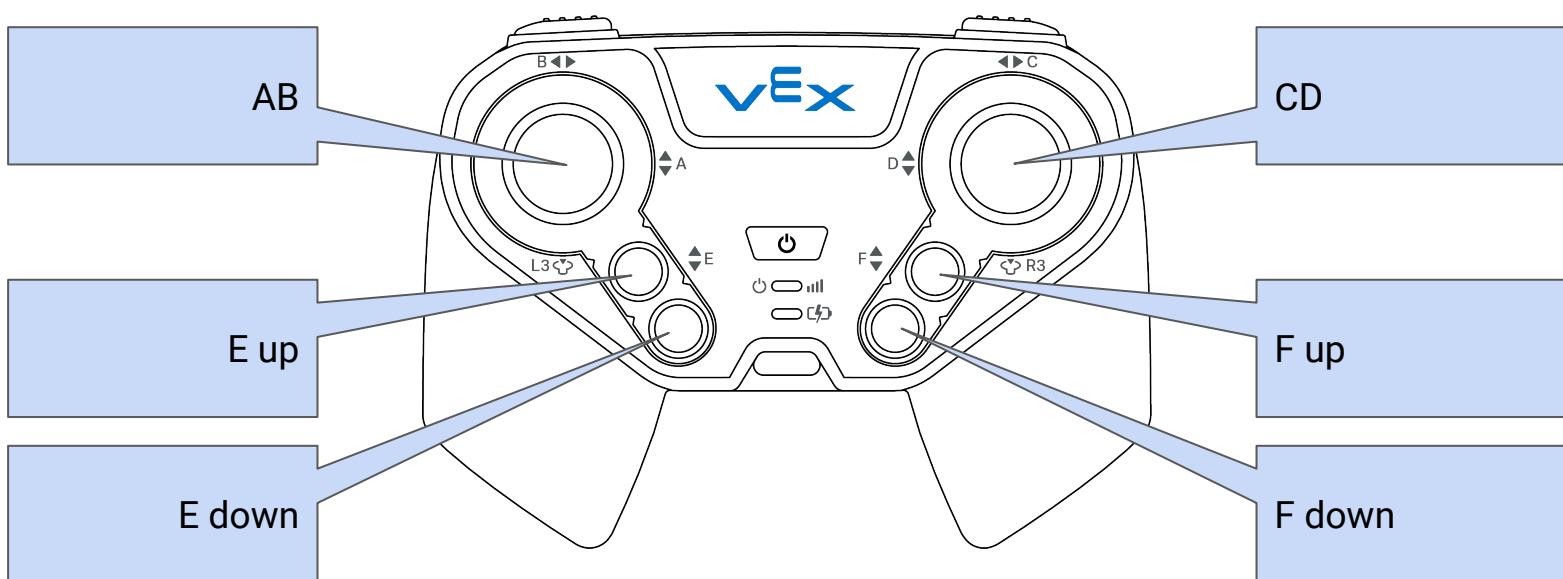
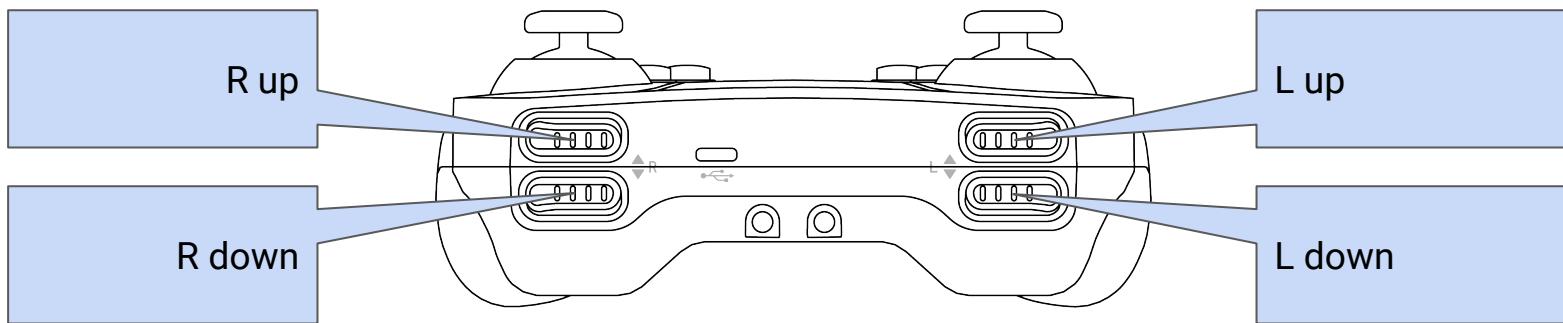


Project

Name

Date

Page 42



Project

Name

Date

Page 43

Gear Ratio

Introduction to Gear Ratio!

Why do we have Gear Ratio?

We have gear ratios to represent a smaller and easier version of our gear combinations. It can change the speed or strength of our pieces so that it is easily adjustable.

Something we learned is that if you want to change the gear to go faster, it will also become slower. Vice versa, if you want to change the gear to be faster, it will become weaker.

What makes it so important?

Gear ratios are important as well as the overall gear idea. If we ever want to make the robot drive faster, we could change the gear ratio. Or if we ever wanted our arm to be stronger, we could change its gear ratio.

Gear Formulas

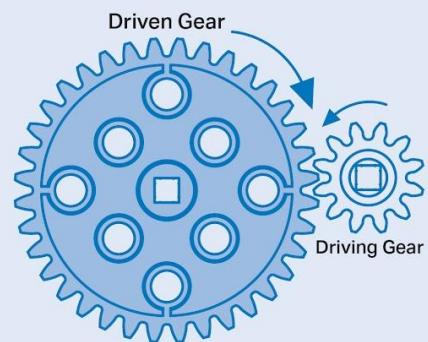
$$\text{Gear Ratio} = \frac{\# \text{ of Driven Gear Teeth (Output)}}{\# \text{ of Driving Gear Teeth (Input)}}$$

Power Transfer is a 1:1 gear ratio where the driving and driven gear have the **same number of teeth**.

Increasing Torque (lowering speed) is a gear ratio where the driving gear has **fewer teeth** than the driven gear.

Increasing Speed (lowering torque) is a gear ratio where the driving gear has **more teeth** than the driven gear.

$$\text{Compound Gear Ratio} = (\text{Gear Ratio 1}) \times (\text{Gear Ratio 2}) \times (\dots)$$



Motion Formulas

$$\text{Average Speed} = \frac{\text{Total Distance}}{\text{Total Time}}$$

Distance is from the axis of rotation

$$\text{Rotational Speed} = \frac{\# \text{ of Turns}}{\text{Time}} = \frac{\text{Degrees}}{\text{Time}}$$

$$\text{Circumference} = \pi \times \text{Diameter}$$

$$\text{Power} = \text{Force} \times \text{Velocity}$$

$$\pi \approx 3.14$$

$$\text{Torque} = \text{Force} \times \text{Distance}$$

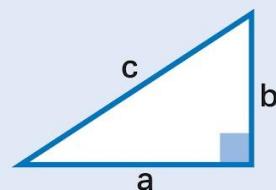
$$\text{Force} = \text{Mass} \times \text{Acceleration}$$

Mathematical Formulas

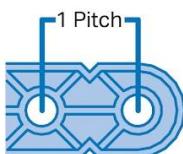
Complimentary angles are angles that sum to 90°

Supplementary angles are angles that sum to 180°

$$\text{Pythagorean Theorem: } c^2 = a^2 + b^2$$



$$1 \text{ Pitch} = 0.5 \text{ in} = 12.7 \text{ mm}$$



Programming

Downloading the IDE

VEXcode Install

VEXcode IQ (Blocks & Text)

- Learn more about VEXcode
- VEXcode IQ Features
- VEXcode IQ Screenshots
- System Requirements

We learned that the new generation of pieces also came with a new IDE/programming system. This is how we learned to download the new IDE.

The Steps:

Online at codelQ.vex.com

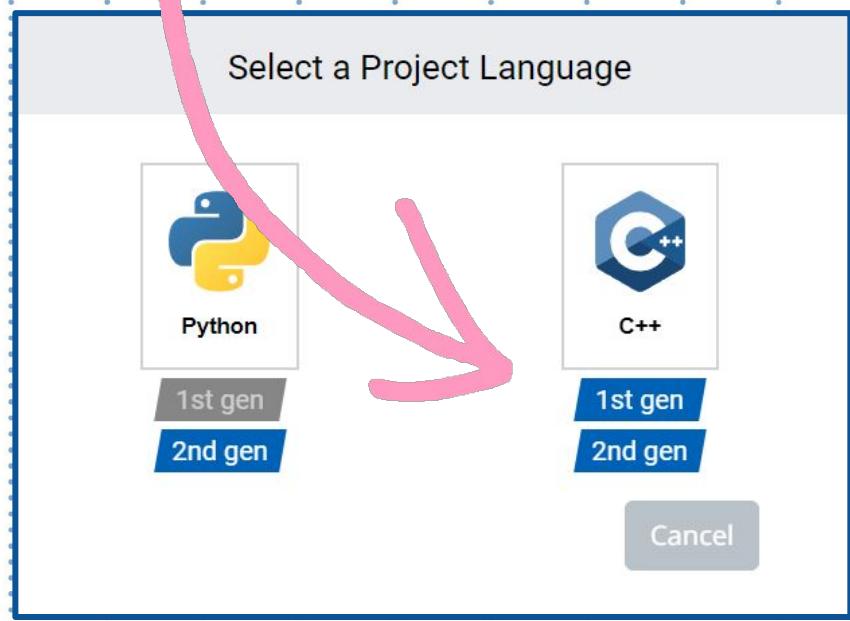
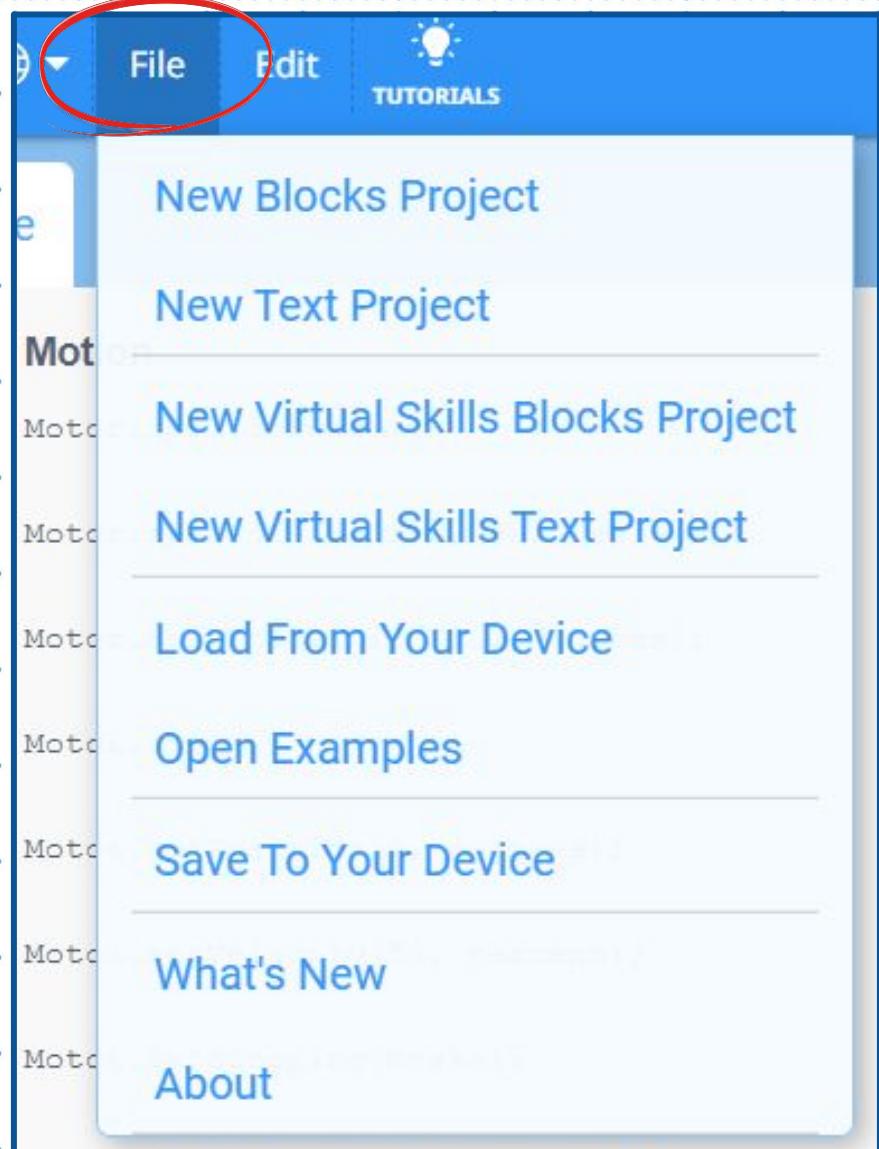
1. First, we went to <https://www.vexrobotics.com/vexcode/install/iq>
2. We could either install the software, or use the online version. The online version is @ codelQ.vex.com
3. After downloading, we could choose whether we want to use blocks or text (we decided on text, because that's what we've been using before).

How we downloaded the VEXcode IDE

4. To start a new project, we click "File", then "New Text Project"

5. We decided to program with C++ this year, because we've already learned a little bit of Python before.

We wanted to try something new!



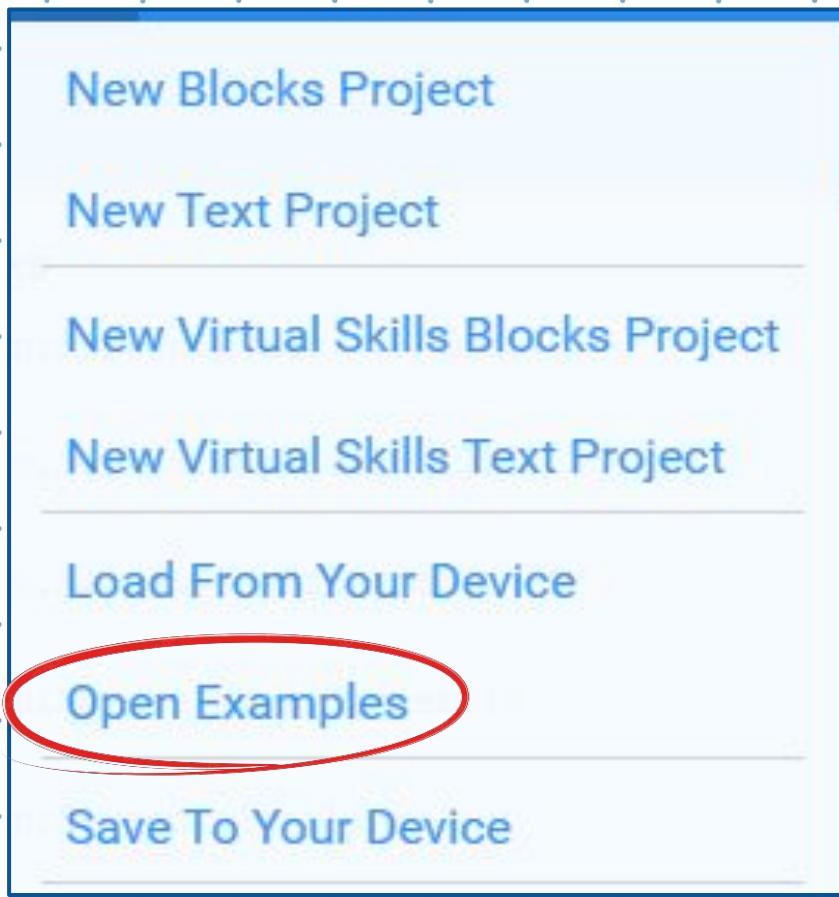
6. congrats! You've just created a new file. Now, practice opening sample programs to see example programs!

[Learn about Sample](#)

[Programs on the next page!](#)

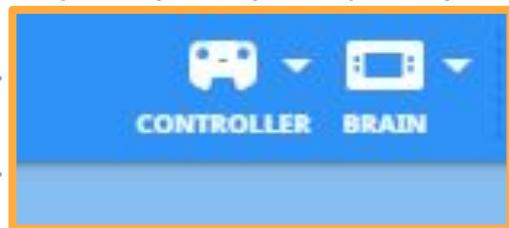
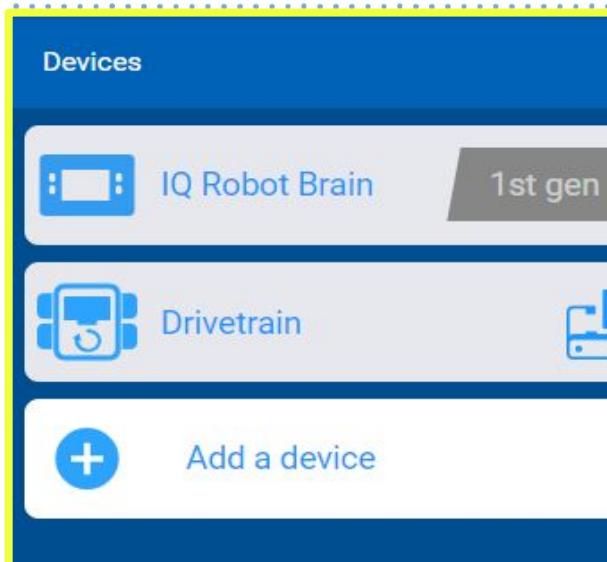
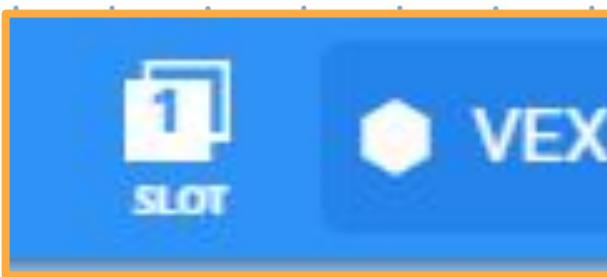
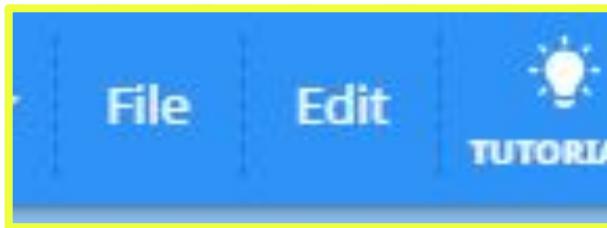
Project More info on the IDE

Using Sample Programs



When we first opened the IDE, we learned that there were sample programs we could practice with. Sample programs will help us learn how to write programs in the new C++ language, and we can learn some basic commands. Looking at sample programs will give us a head start in programming.

What we did to run the program:



1. After opening <https://codeiq.vex.com/> or open your download software, then click "File", then "Open Examples".
2. There are many examples we looked through, including basebot, clawbot, and snapshot. choose one.
3. To run the code, select a slot to place the program to by clicking the "slot" button at the top.
4. Name your motors. Press this icon: at the top right corner. Then select "add a device" Add your motors and sensors.
5. Save the code by going to "File" and "Save to your device", then connect the brain and controller to the computer. Your Program should appear in the slot you selected.

Project What We Did to Run the Program

Figuring Out Github

This is the first year we are using GitHub. GitHub is a platform that allows us share our codes to teammates. It is attached to Git, which is a software that lets us save all copies of our code, and put to code into a repository.

Here is how we downloaded/prepared GitHub and Git:

1. First, we went to github.com, and created an account. Each of us created our own accounts.
2. Then, one of us created a repository, and invited the rest of us to be collaborators.



3. Next, we downloaded Git at <https://git-scm.com/download>
4. We created a new file in GitHub to practice linking Git and Github, and transferring files.



git

These are
some basic

commands we used in Git:

`git init`

`git remote add origin URL_of_git_repository`

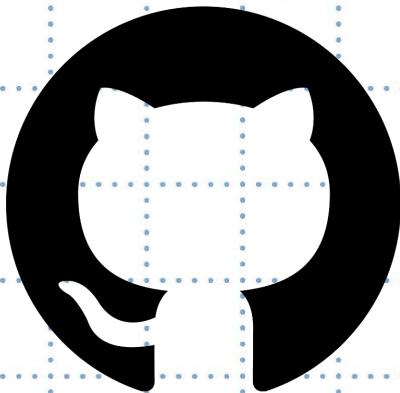
`git pull origin master`

`git add .`

`git commit -m "message"`

`git remote -v`

`git push -u origin master`



GitHub

WHAT DO ALL THE COMMANDS DO?

`git init`

- Turns local directory into a git repository

`git remote add origin URL_of_git_repository` (make sure file explorer is open to the right folder)

- Connecting local repository to a remote repository on github

`git pull origin master`

- Allows to pull all files in remote github repository

`git add .`

- Checks all change into local git repository
- Puts into a staging area
- The “.” means to add all files

`git commit -m "message"`

- Commits the changes

`git remote -v`

- Checks which github repository you are connected to

`git push -u origin master`

- Pushes change into github

The screenshot shows the VEXcode IQ interface. The top menu bar includes 'File', 'Edit', 'TUTORIALS', 'VEXcode Project ...', 'Saved', 'CONTROLLER', 'BRAIN', 'BUILD', 'RUN', 'STOP', 'SHARE', and 'FEEDBACK'. The left sidebar has sections for Looks, Sound, Events, Control, Sensing, Variables, and Functions, each with corresponding icons. The main code editor window displays C++ code:

```
1 // #pragma region VEXcode Generated Robot Configuration...
2
3 // -----
4 // Module:      main.cpp
5 // Author:      {author}
6 // Created:     {date}
7 // Description: IQ project
8
9 //
10
11 // Include the IQ Library
12 #include "iq_cpp.h"
13
14 // Allows for easier use of the VEX Library
15 using namespace vex;
16
17 int main() {
18     // Begin project code
19 }
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
```

The code includes comments explaining the generated configuration region, the inclusion of the IQ library, and the use of the vex namespace. It also contains a main() function placeholder.

On the meet of 10/15/22...

For this week's meet, we went through a sample code in the 'Gen 2' part of the updated code for c++. We went through the first couple lines showing the `#Pragma` sign, strings, and a while loop. While not knowing much knowledge, we tried to connect this code to the one last year in Robot c.

Project

Name

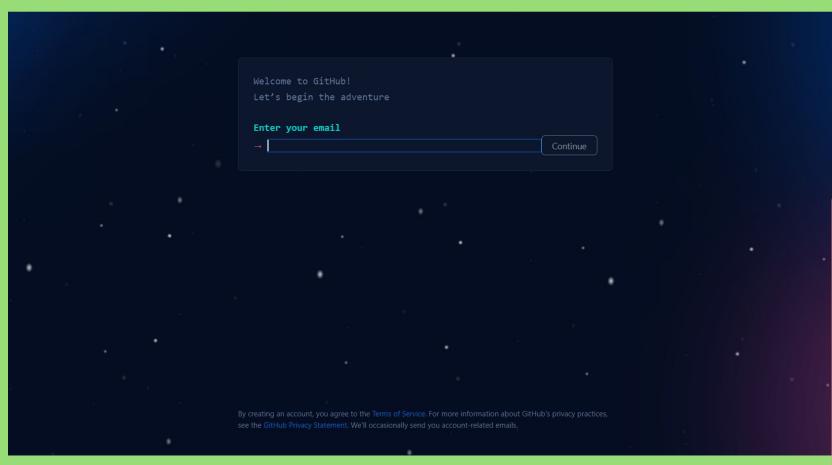
Date

Page

A Tour of Github!

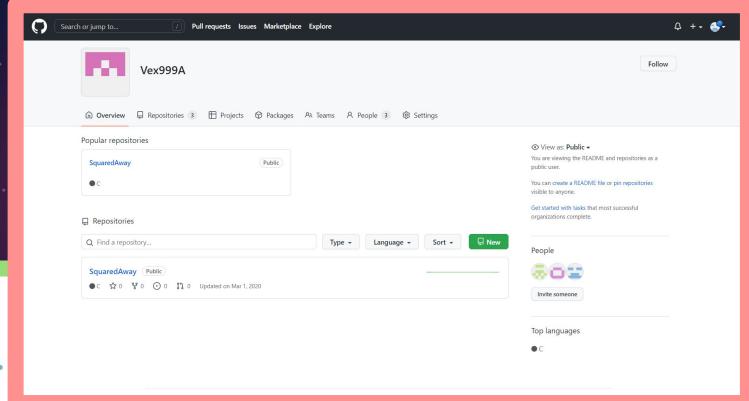
Step 1:

The first step is to make an account for github and finish making your make a fun username!



Step 2:

The second step is to add your account and get added by your team account or coach.



Step 3:

Post your code, and start sharing with your team!



Project USING Github!

Name Zoe Pak

Date 10/31/22

Page 48

Teleops in Programming

Goal:

- Be able to comfortably drive the robot no matter the driver.
- Switch controller buttons using programming

Picture:

The screenshot shows the VEXcode IQ software interface. The code editor contains the following VEXcode project code:

```
1 Brain.Screen.print("VEXcode");
2
3 Looks
4 Brain.Screen.setScreenColor(1, 1);
5 Brain.Screen.clearScreen();
6 Brain.Screen.newLine();
7 Brain.Screen.clearLine();
8 Brain.Screen.drawString("0, 0, 10, 10");
9 Brain.Screen.drawLine("0, 0, 10, 10");
10 Brain.Screen.drawRectangle("0, 0, 10, 10");
11 Brain.Screen.setFont("font001");
12 Brain.Screen.setFontWidth(10);
13 Brain.Screen.setPenColor("red");
14 Brain.Screen.fillRect("0, 0, 10, 10");
15 Brain.buttonLeft.pressed(callback);
16
17 while (controller.ButtonUp.isPressed()) {
18     // wait until buttonUp is released
19     walk(0, 0, 10);
20 }
21
22 if (controller.ButtonDown.isPressed()) {
23     // Spinning the SpinMotor in reverse opens the claw
24     Raspmotor.spin(reverse);
25 }
26
27 while (controller.ButtonDown.isPressed()) {
28     // wait until buttonDown is released
29     walk(0, 0, 10);
30 }
31
32 if (controller.ButtonUp.isPressed()) {
33     // Spinning the SpinMotor in forward opens the claw
34     Spinmotor.spin(forward);
35 }
36
37 while (controller.ButtonUp.isPressed()) {
38     // wait until buttonUp is released
39     walk(0, 0, 10);
40 }
41
42 if (controller.ButtonDown.isPressed()) {
43     // Spinning the SpinMotor in reverse opens the claw
44     Raspmotor.spin(reverse);
45 }
46
47 while (controller.ButtonDown.isPressed()) {
48     // wait until buttonDown is released
49     walk(0, 0, 10);
50 }
51
52 if (controller.ButtonUp.isPressed()) {
53     // Spinning the SpinMotor in forward opens the claw
54     Spinmotor.spin(forward);
55 }
56
57 while (controller.ButtonUp.isPressed()) {
58     // wait until buttonUp is released
59     walk(0, 0, 10);
60 }
61
62 if (controller.ButtonDown.isPressed()) {
63     // Spinning the SpinMotor in reverse opens the claw
64     Raspmotor.spin(reverse);
65 }
66
67 while (controller.ButtonDown.isPressed()) {
68     // wait until buttonDown is released
69     walk(0, 0, 10);
70 }
71
72 if (controller.ButtonUp.isPressed()) {
73     // Spinning the SpinMotor in forward opens the claw
74     Spinmotor.spin(forward);
75 }
76
77 while (controller.ButtonUp.isPressed()) {
78     // wait until buttonUp is released
79     walk(0, 0, 10);
80 }
81
82 if (controller.ButtonDown.isPressed()) {
83     // Spinning the SpinMotor in reverse opens the claw
84     Raspmotor.spin(reverse);
85 }
86
87 while (controller.ButtonDown.isPressed()) {
88     // wait until buttonDown is released
89     walk(0, 0, 10);
90 }
91
92 if (controller.ButtonUp.isPressed()) {
93     // Spinning the SpinMotor in forward opens the claw
94     Spinmotor.spin(forward);
95 }
96
97 while (controller.ButtonUp.isPressed()) {
98     // wait until buttonUp is released
99     walk(0, 0, 10);
100 }
```

Problem:

- The program was not loading properly and the buttons seemed to not be working.

Solution:

- we found out that we had two while loops in a row, but that could not work since the code is followed consecutively.
- we combined the while loops to run correctly in order.

Project Teleops in Programming

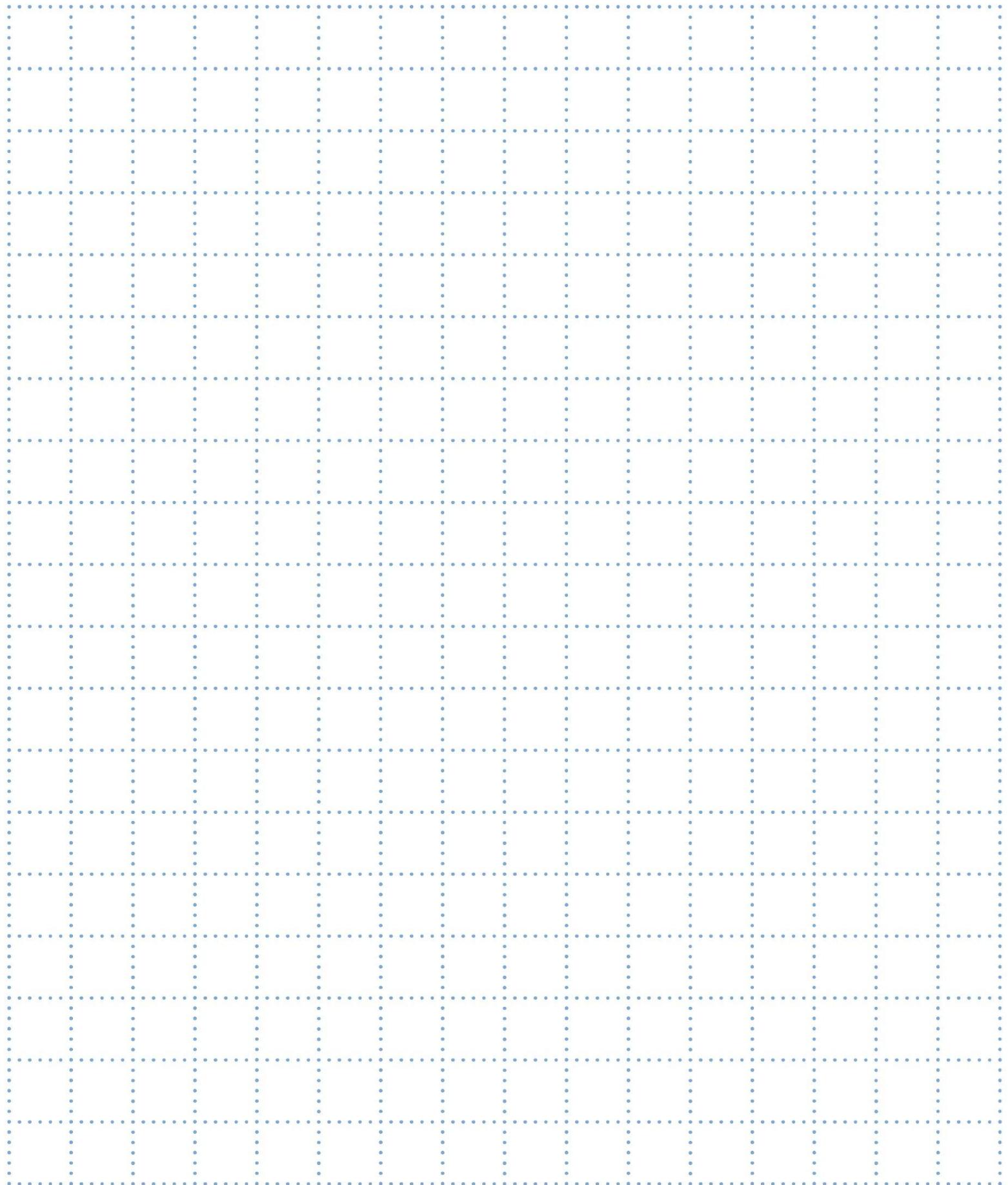
Name Zoe Pak

Date 11/5/22

Page

49

Parts List



Project

Name

Date

Page 59

Sticker slide

Project

Name

Date

Page 60

