Robotics Engineering Notebook

REC Foundation

VEX

team name: Rising Phoenix

team number: 999A

season: 2021-2022

start date:

end date:

book number: of:

10
9
8
7
6
5
4
3
2
1 inch

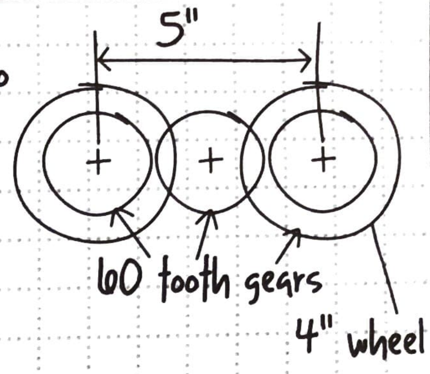
RECF Mission Statement

The Robotics Education & Competition (REC) Foundation seeks to increase student interest and involvement in science, technology, engineering and mathematics (STEM) by engaging students in hands-on, sustainable and affordable curriculum-based robotics engineering programs across the U.S. and internationally. The REC Foundation develops partnerships with K-12 education, higher education, government, industry and the non-profit community to achieve this work.

Suggestions for Getting Started with an Engineering Notebook

- Make an entry every day and initial it.
- Use every page.
- Print all entries in permanent ink.
- Do not use markers that can bleed through the paper.
- All entries are sequentially numbered from page to page.
- Do not remove pages from the bound notebook for any reason.
- Provide a brief statement of the objectives for the session.
- Use a single line to cross out a mistake in an entry.
- Document all research and cite your sources.
- Label all pictures, sketches and calculations.
- Use a glue stick or tape to permanently attach any inserted items.
- Clearly indicate the date before or after each entry on a page.
- Mark off all excess space on a page with an X and initial it.
- Never erase or remove anything from the engineering notebook
- Do not use White Out.
- Show all work for formulas and conversions.
- Entries should be clear and complete so that someone else can follow and understand your design process.
- Document all testing and code debugging.
- Sign and date each page.
- When the notebook is full, archive it and start a new one.
- Store the notebook in a safe place.
- Include outlines for oral presentations on the project upon its completion.
- Study some sample engineering notebooks at http://www.vex.com/vrcteams for inspiration.
- Photocopies of engineering notebooks can be used to support presentations such as the VEX IQ STEM Research Project.

9/21 – Our current drive train is 1 to 1 with (2) 393 motors and 4" wheels. In order to go faster, we will need to change the gear ratio of the drive train. There are several ways to accomplish this mechanically, so I need to do some calculations first to determine the speed of the current drive train. – DR



Circumference = Diameter $\times \pi$ = 4" x 3.14 = 12.56"

The 393 motor has two speeds, low and high. Using the motor information we got off www.vexrobotics.com site, we calculated the speed of our robot – DR

At 7.2V Low Speed = 100 RPM
High Speed = 160 RPM
– DR

Low Speed

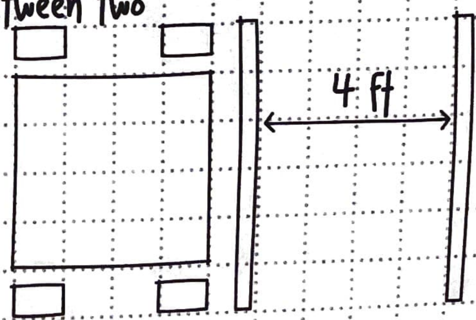
$$100 \frac{rev}{\cancel{min}} \times \frac{1}{60} \frac{\cancel{min}}{sec} = 1.667 \frac{\cancel{rev}}{sec} \times 12.56 \frac{inches}{\cancel{rev}} = 20.93 \frac{inches}{sec}$$

High Speed

$$160 \frac{\cancel{rev}}{\cancel{min}} \times \frac{1}{60} \frac{\cancel{min}}{sec} \times 12.56 \frac{inches}{\cancel{rev}} = 33.49 \frac{inches}{sec}$$

– DR

9/22 – To test these calculations, we ran our robot between two lines that were 4 ft apart and timed how long it took.



In Low Speed $20.93 \frac{in.}{sec} = 48 \frac{in.}{X}$

$$X = \frac{48}{20.93} = 2.29 \text{ sec}$$

– DR

– DR

Journal Entry

9/21 – Nancy and I have been driving the Clawbot around the field and I think it needs to be faster. Today we worked on some basic calculations before we changed about our robot. My teacher does not like it when I tear apart my robot right before a competition. I am not sure why, we are just trying to make it better! – DR

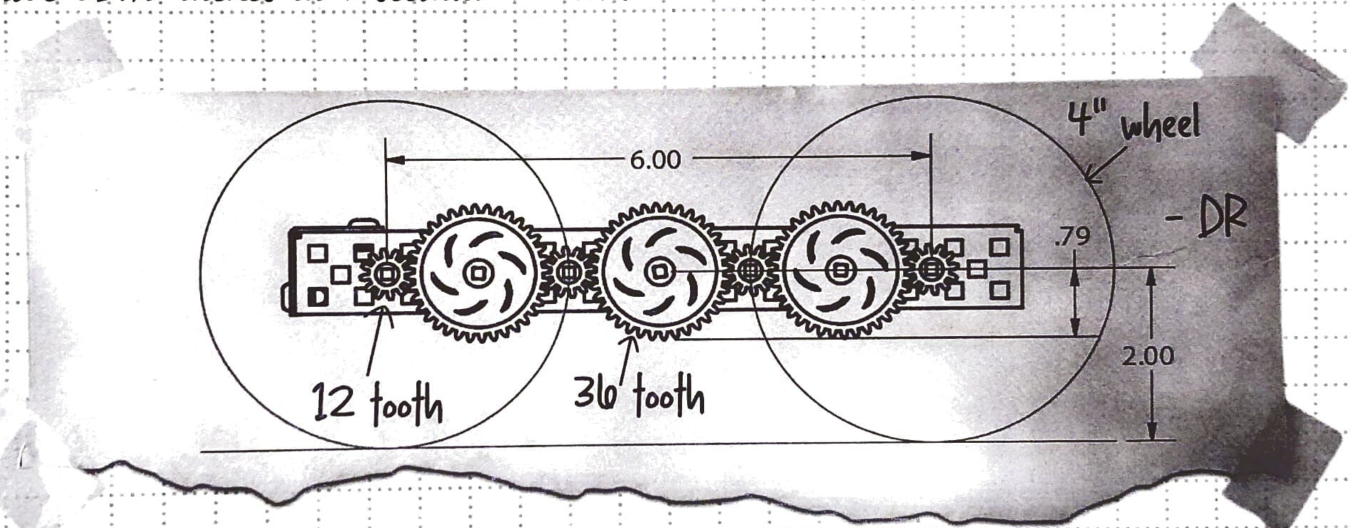
project Drive Train Modification designed by: Dan RECF witnessed by: Nancy RECF

date: 9/21/13

9/22 (continued) - Our estimate was 2.29 seconds and the actual time was 2.1 seconds. The difference is probably due to the higher battery voltage. The VEX site indicated the RPMs were based on a 7.2V battery. We measured the battery voltage and it was 8.1 volts. - DR

9/23 - We swapped the gears inside the motors and repeated the time test. The robot was significantly faster, but turning was not as good. I think we should try using the 4" Omni wheels for the front two wheels. - DR

9/24 - Last night I was trying different gear ratios in CAD, but the wheel spacing was giving me a little trouble. I finally got a 3 to 1 ratio that would work using the 12 and 36 tooth gears. This new ratio increases our speed by 3 times and ~~increased~~ increases our ground clearance by 0.5". In theory, the robot should move 62.79 inches in 1 second. - NR



- DR

9/25 - Today we modified the robot's drive train to the 3 to 1 ratio. We had to move the wheels out 1/2" on either side. This will help with our stability. - DR

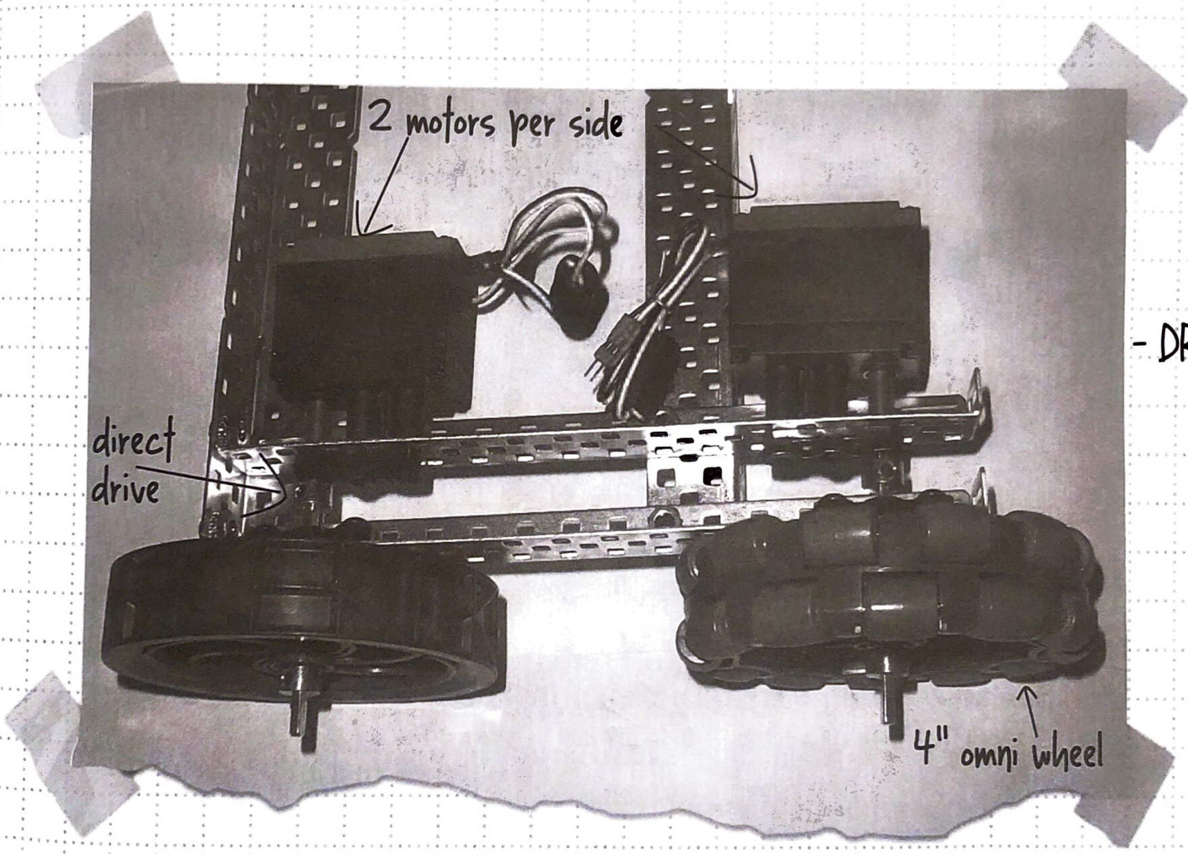
- DR

Journal Entry
9/23 - Yesterday we timed the actual speed of our robot and identified potential ways to increase the speed of our robot. Last night, Nancy came up with a few other ideas on how to change the speed of our robot. She did some CAD work in Autodesk Inventor. I think she is on the right track. - DR

project: Drive Train Modification designed by: Dan RECF witnessed by: Nancy RECF date: 9/25/13

9/26 - The 3 to 1 gearing did not work. Nancy and I discussed our options and we think if we use (4) 393 motors in high speed, we will have the best of both worlds. We will be faster and stronger. Using 4 motors will allow us to direct drive the wheels and make the design even simpler. With no gears, we will also have even better ground clearance. - DR

9/27 - Here is a picture of our new drive train. It seems so simple now. It works pretty well, especially with the omni wheels. The increased speed will definitely give us an advantage at the competition. - NR



- DR

- DR

Journal Entry
9/26 - OK, 3 to 1 was a little too fast. It did not move as fast as we calculated, probably because of the ramp up time of the motors. The robot was very fast, but also very weak and it did not turn well at all. I think we need to find another option. - NR

project Drive Train Modification designed by: Dan RECF witnessed by: Nancy RECF

date: 9/27/13

Team Photo

Team Profile

Sammy - My full name is Samantha but people call me Sammy. I do swimming and my favorite color is black. My favorite food is frozen watermelon. I am a driver.

Mindy - I enjoy playing basketball and hanging out with my friends and family. I also do dance and have a dog named Jay.

Kristen - I enjoy dancing, gymnastics, and art. I love spending time with my family, and three pet birds: Paris, Mochi, and Yoshi.

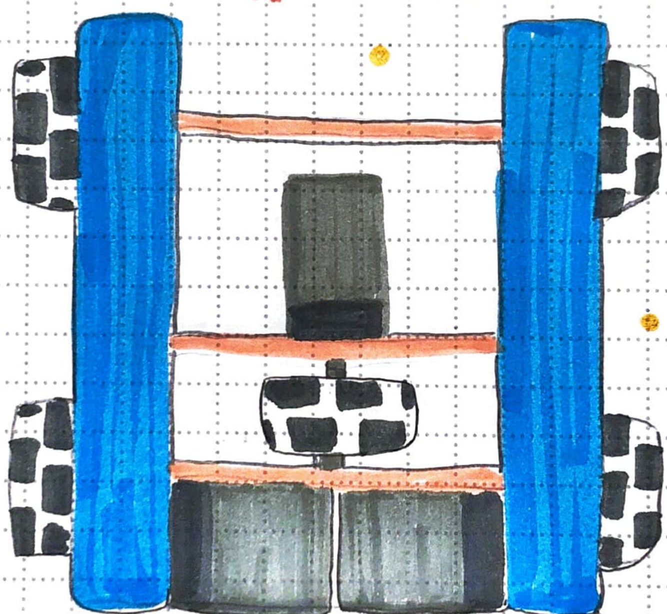
My name is Zoe Pak, and I enjoy playing basketball, piano and being with my friends and family.

Helena - I love art, playing soccer, playing music, and am kind of obsessed with Hamilton... ☺

Renee - I like drawing, crafting, and playing piano. I also like hanging out with my robotics friends!

My Projects

DRIVE



TRAIN

project

designed by: Kristen

witnessed by:

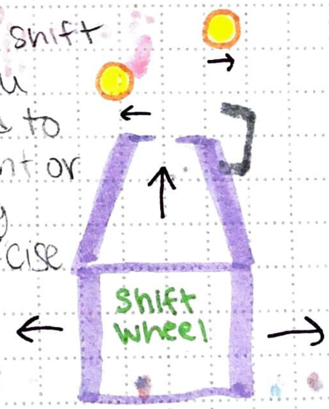
date:

H-DRIVE

During past years, we have used the h-drive for our robot. This season, the h-drive will be crucial. This is because we will need to have a shift wheel to aim at the basket, or pick up balls. The shift wheel allows the robot to travel right and left without turning. We want the robot to be precise.

Without a shift wheel you will need to turn to grab the balls.

With a shift wheel you only need to shift right or left, being more precise



GOALS:

- We want a controllable h-drive so that it will be easier to drive.
- We want a swift h-drive so that it will travel across the large board fast to collect balls. But drivetrain can't be too fast that it's uncontrolable.

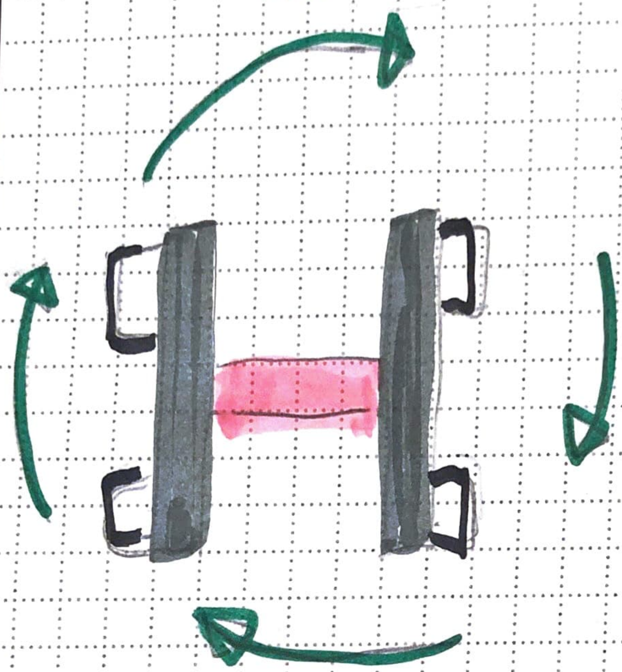
project

designed by: Kristin

witnessed by:

date:

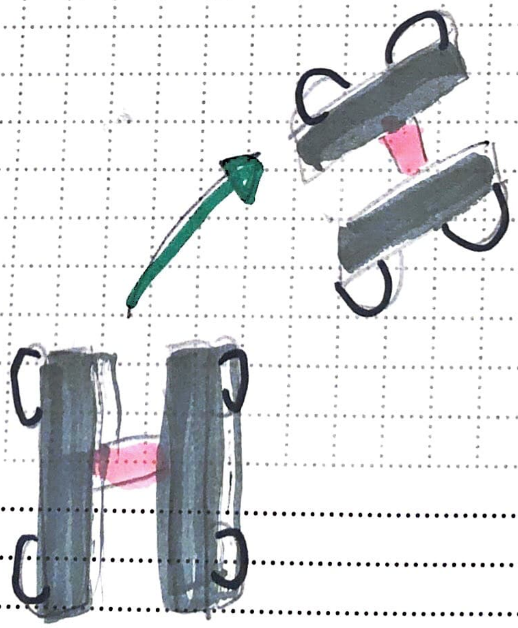
BASE



Our drivetrain can't drive straight. we do everything right but it always goes to the right. This a problem because then we can't latch onto the bar.

PROBLEMS

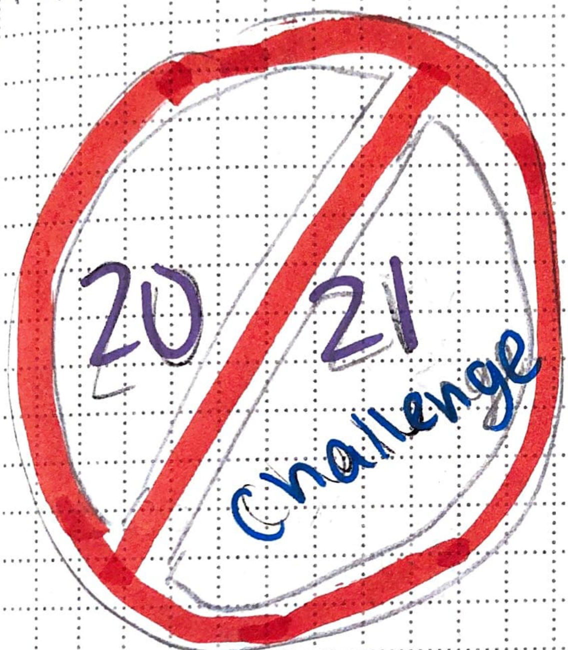
when we want to just turn, one of the wheels has a problem. Everything seems fine but it just won't move how we want.



project designed by: witnessed by:

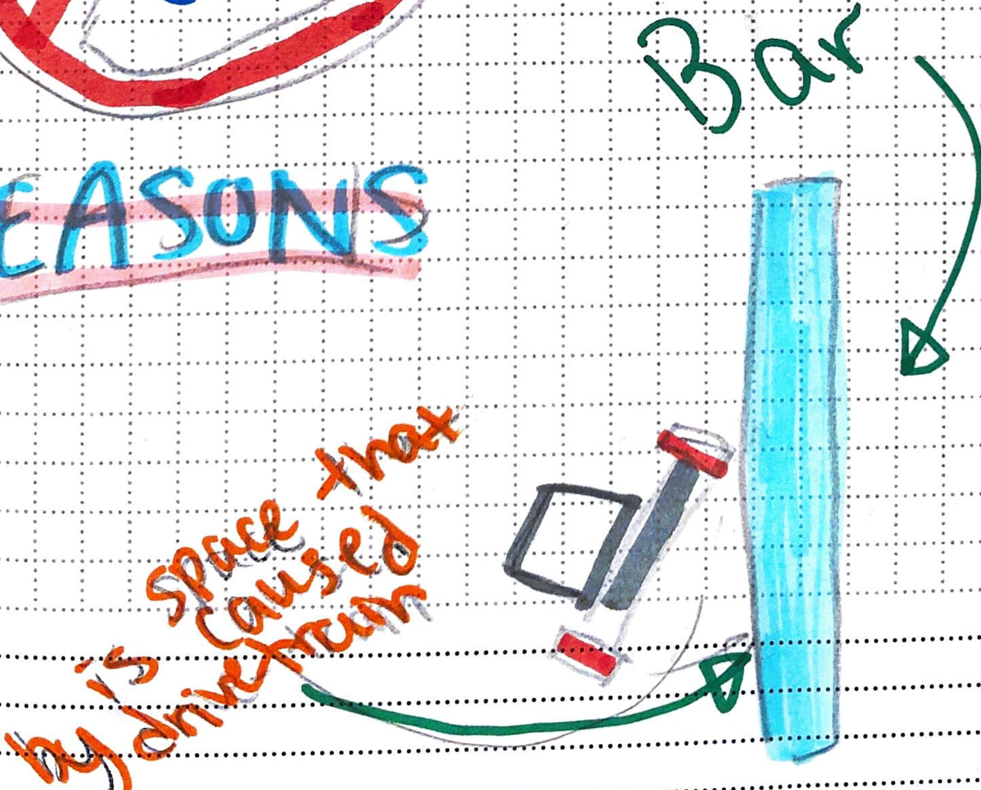
date:

We tried this out and this will be a big problem if we can't fix it.



FLOPS

REASONS



IS THE SHIFT WH NECESSAY

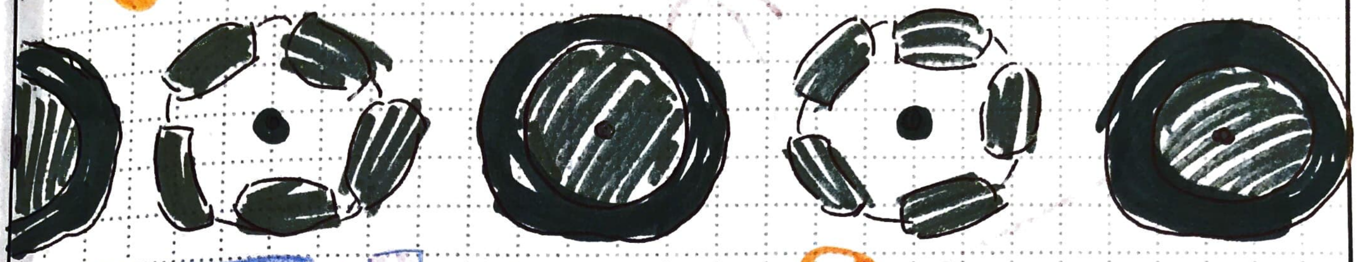
Yes!

The shift wheel helps micro-adjust your robot. You can be more precise!

why do you need to be precise

In order to shoot the balls into the basket, you need to be aligned with it correctly

It is important to earn more POINTS!



EEL

Y

The shift wheel uses one motor, which could be used for something else.

No!

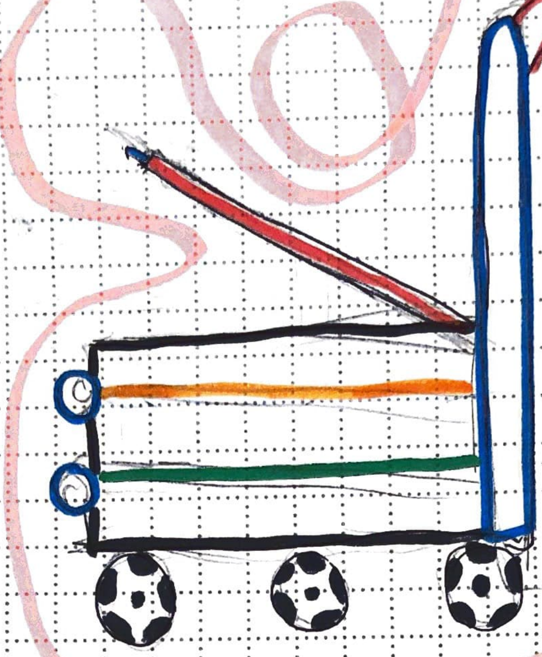
Why?

Since there is a limit of how many motors we can use, (six) we must use them wisely.

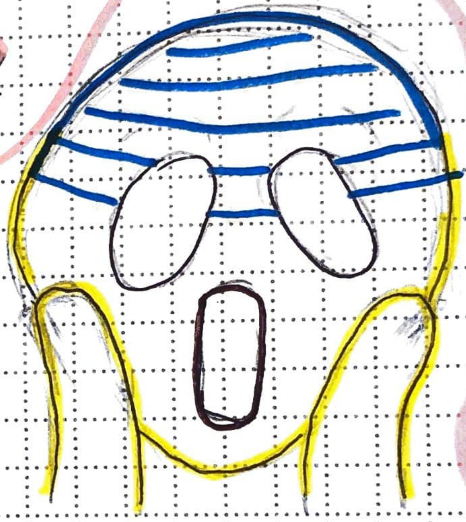
NOTE: if you have extra motors, you can add a shift wheel.

Some things that are more important than micro-adjusting are: The claw, the arm, the roller, and the other wheels.

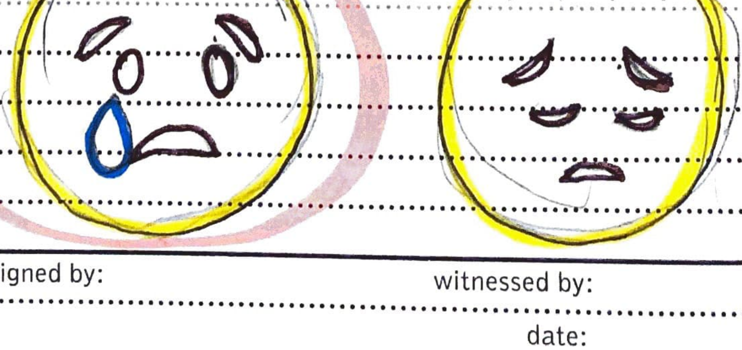
Brainstorm



BIG PROBLEM



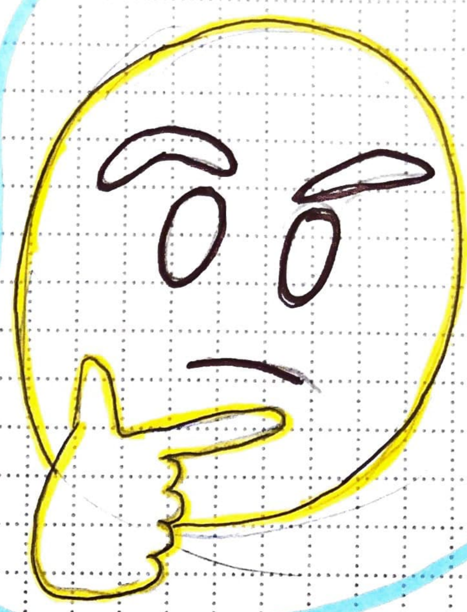
In the beginning, there was no rule against handling balls off the ground. So, we made a robot that could do anything we wanted perfectly. It was within the size limit and followed the rules. But suddenly... some-thing came up that completely destroyed our spirits.



project designed by: witnessed by:

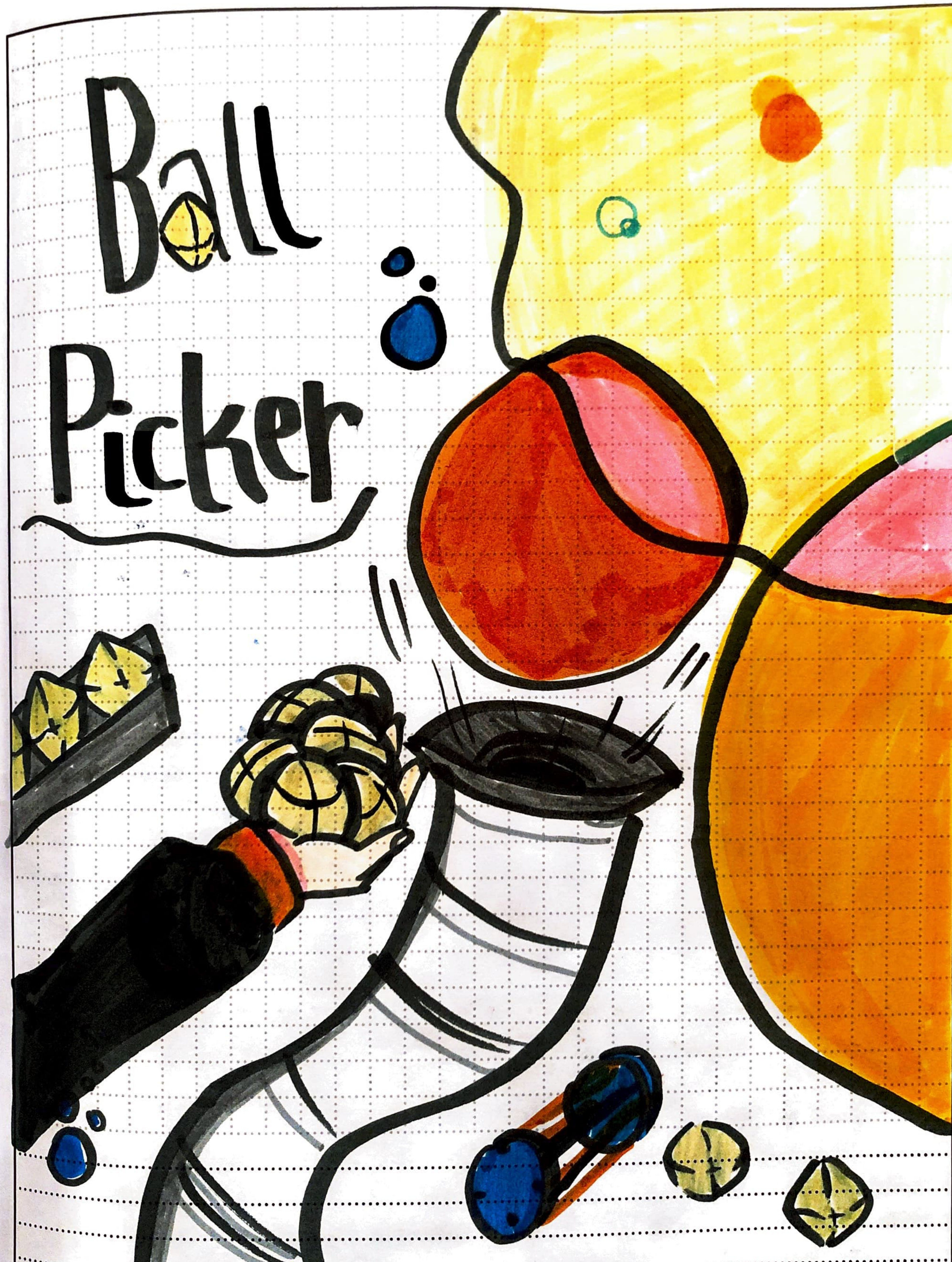
date:

WHAT DID WE DO?



We got together and went straight to work. We thought of new methods we could use, how we would connect the balls and robot, what we could do to get the balls in the basket, etc. We were very efficient and tested different strategies. We also went over the engineering process and used that to help make our new robot.

1. Goal
2. Strategy
3. Pick a Strategy
4. Proof of Concept
5. Build



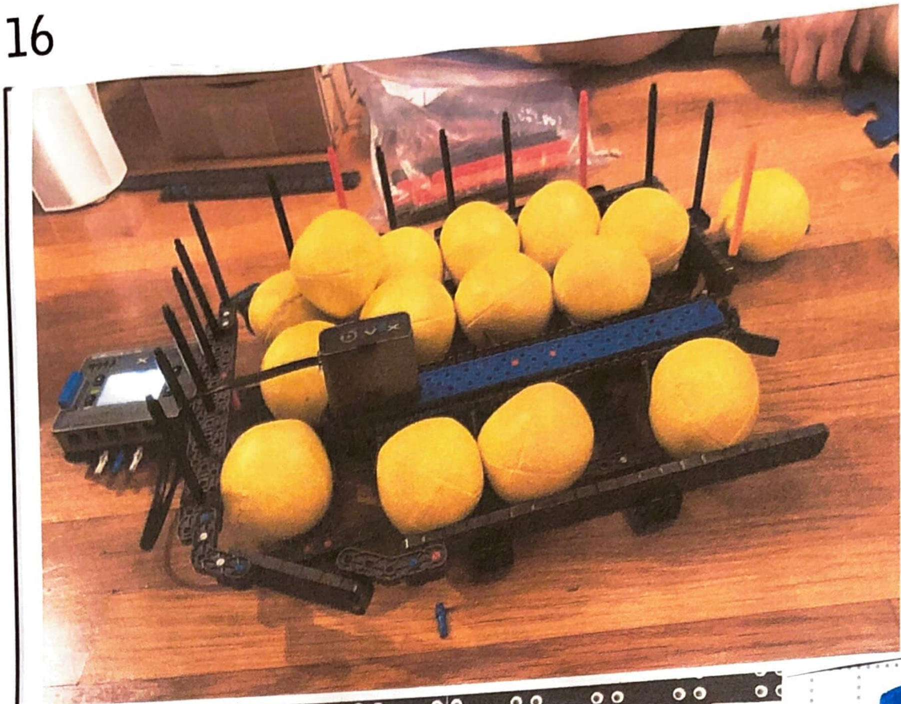
project

designed by: Renee

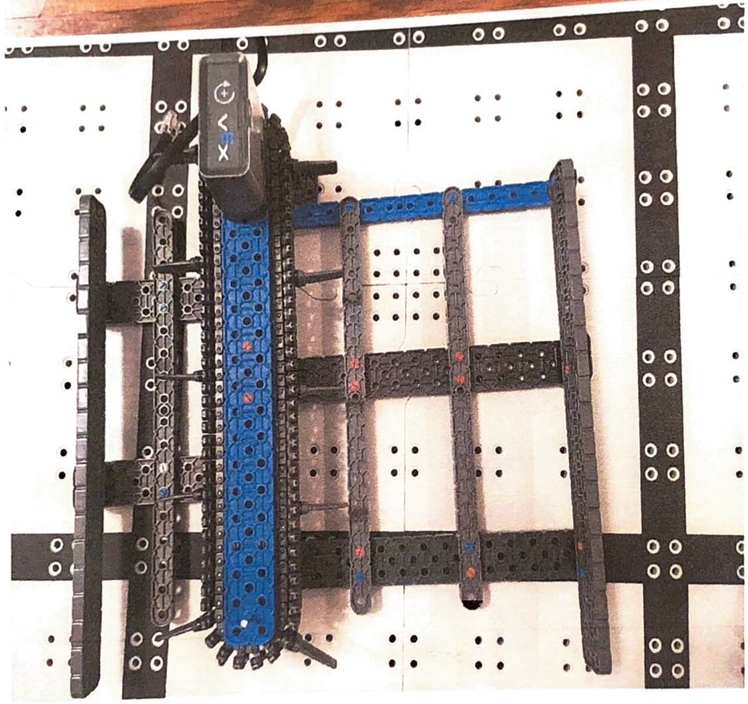
witnessed by:

date: July 10

Ball Picker



FIRST BALL PICKER DESIGN



PROS

- Many balls able to be collected.
- Fast & can use without precision.
- Easy depositing system, just turn the roller the other way.
- Balls cannot fall out, due to the bars on the side of the ball picker.

CONS

- Very big & heavy, no way to attach to arm.
- Balls get stuck while entering, due to the constant moving of the roller and balls pushing against each other.
- Excessive bulkiness, with this on the arm it wouldn't be able to move or get up the high bar.
- If adding it, many other lighter designs might be blocked off by it's size.

project

designed by: Renee

witnessed by:

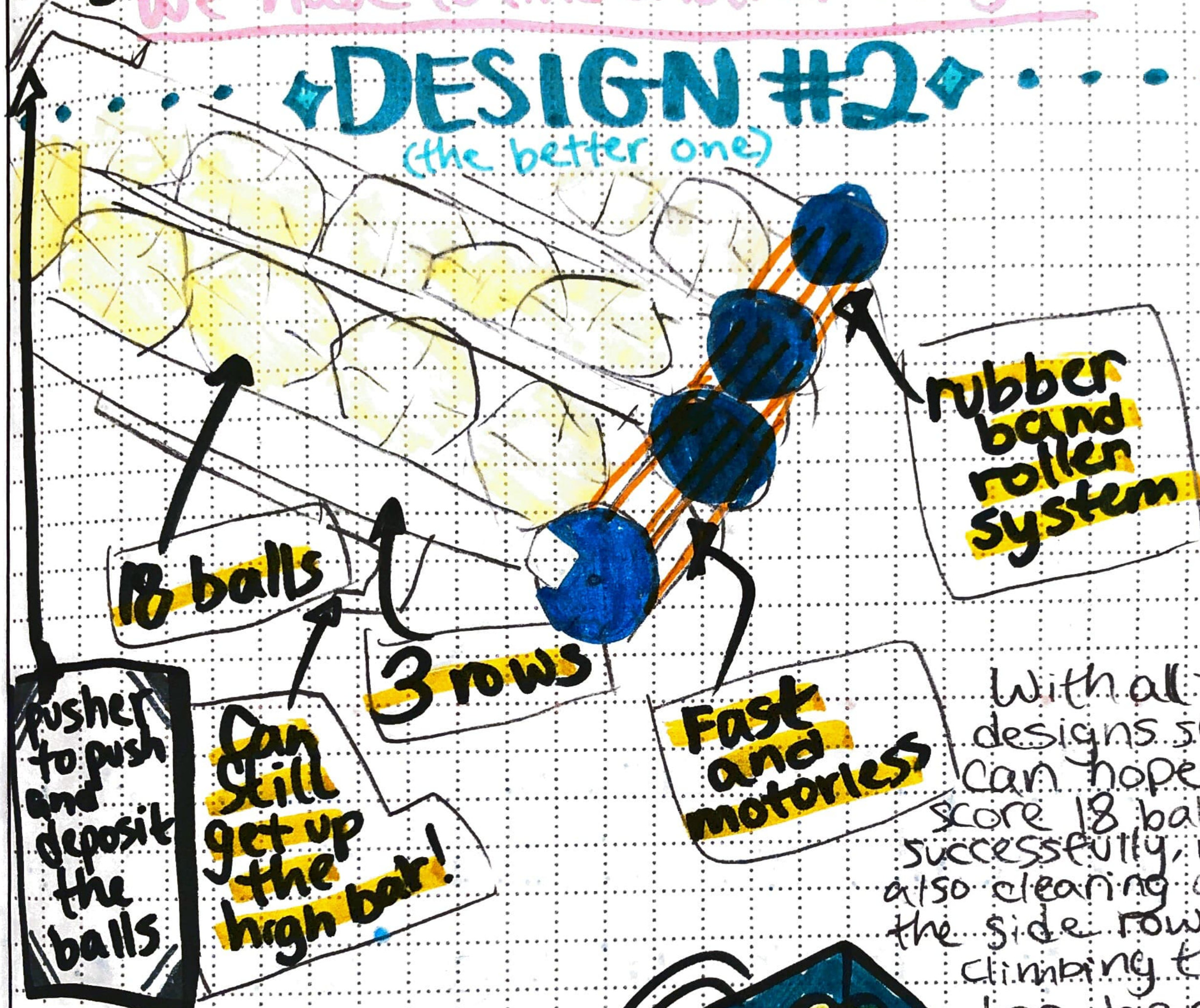
date: July 10

As you can see, the cons overrule the pros. :(

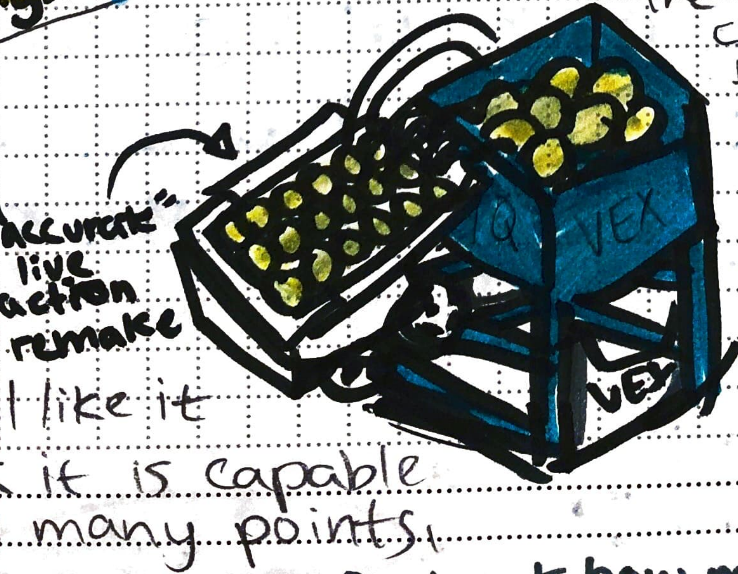
We have to find another design!

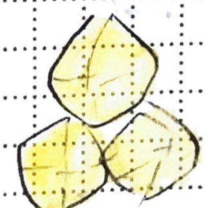
DESIGN #2

(the better one)



With all these designs set, we can hopefully score 18 balls successfully, while also clearing out the side rows & climbing the top bar!





Though the design may cost us a motor and is a bit heavy, I like it and think it is capable of scoring many points.

STILL NEED TO DO: find out how much time it takes to pick up all the balls, and most importantly, BUILD THE DESIGN!!!!

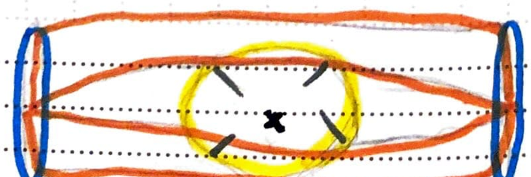
Ball picker

Roller

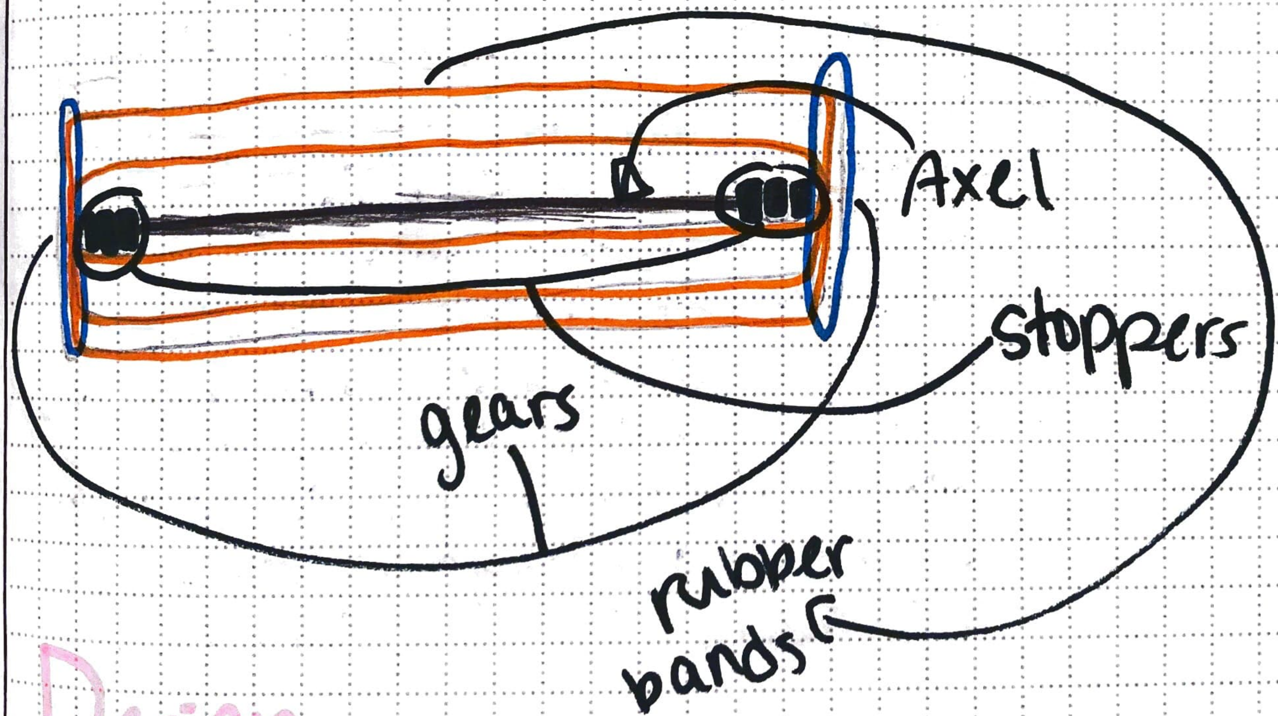
That is the ball picker for robot

This ball picker rotates and pulls in the balls. It's similar to a vac-uum. You move around and the roller spins and the ball's go under and pop into the bucket.

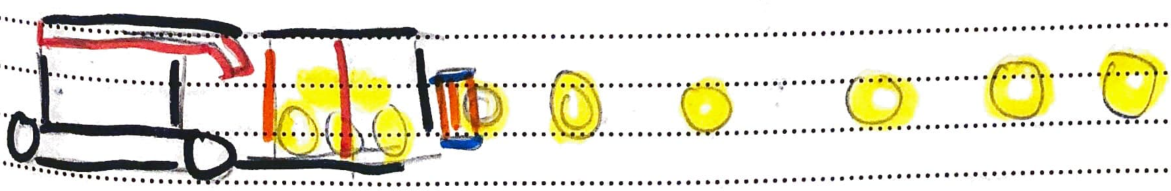
·DRAWING·



Pros	Cons
· easier to control · lightweight · no need to be precaise when pick-ing up	· only one at a time · may be a tiny bit slower instead of just picking up

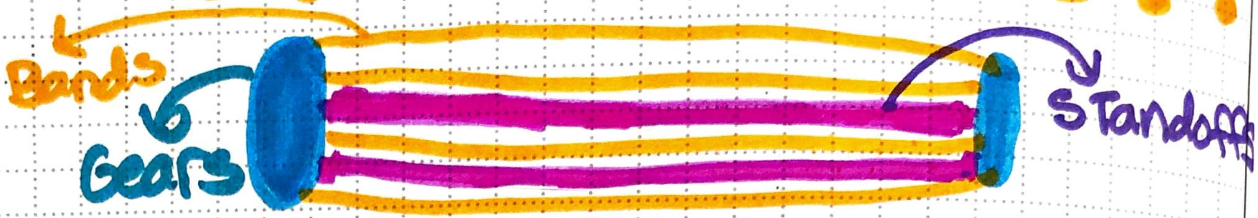


Design



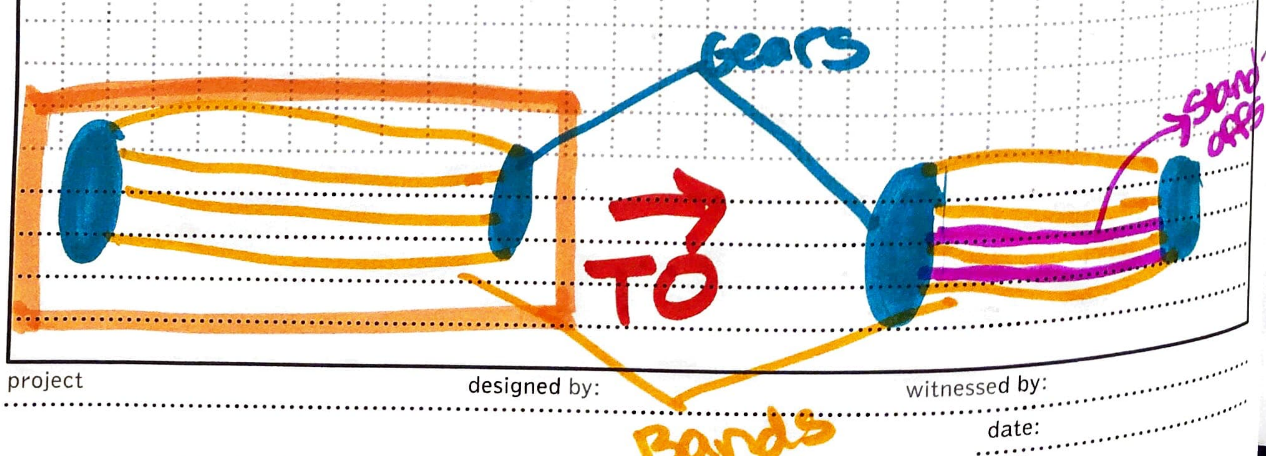
project
designed by:
witnessed by:
date:

BALL PICKER 2.0



PROBLEM EXPLANATION:

This roller may be a very simple design, although there was an obstacle. We tried to build this roller with metal axles to keep the gears apart. Even though we used the weakest rubberbands, the force brought the gears closer together. This is not good because the roller eventually gets too small for even one ball.



project

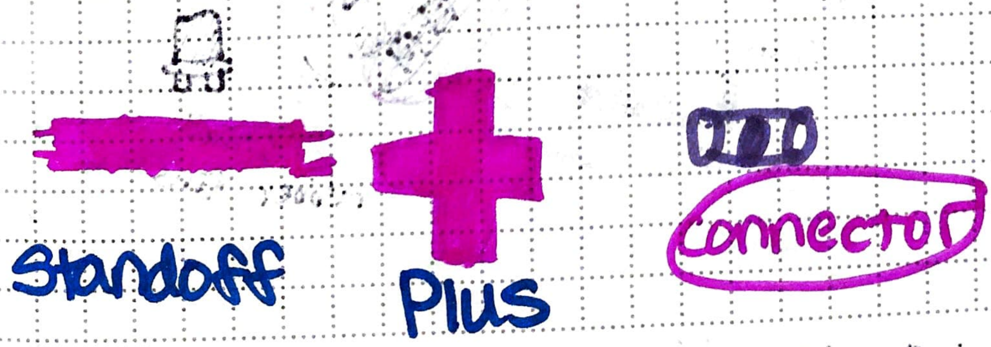
designed by:

witnessed by:

date:

Solution EXPLANATION

This problem took two attempts to be solved. The first was to add more stoppers on the insides of the gears. This slowed down the pulling together problem, but not enough. This is how we came up with a solution to connect standoffs together. With the standoffs, (we made two) we connected them to the center of the gears. This prevented from the pulling gears.



Ball Picker

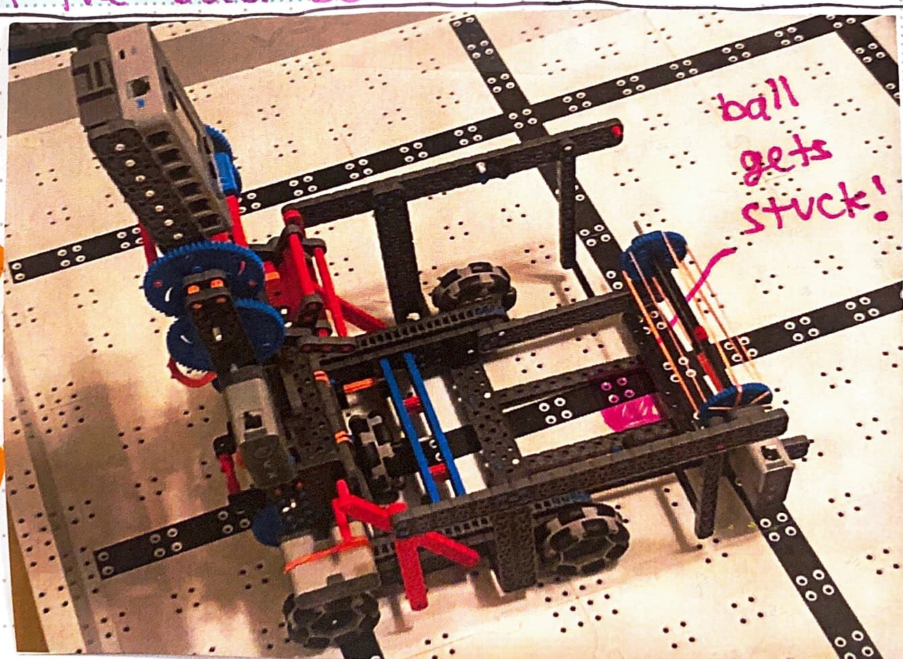
ROLLER: FIN

problem

As we mentioned before, the balls always get stuck in a groove behind the roller. At a certain number of balls, the rollers gets stuck and can't pick more balls. This makes it very unpredictable & inconsistent.

we need the balls to fly up when collected, and pile at the back so more balls can be collected.

Lift for Pic



Lift for Pick

Our idea was to add small rubber fins to re-direct the balls up ward (to solve the initial problem and to be able to collect more balls)
We ALSO expanded the roller a little wider so balls wouldnt get caught between the gear.

project | designed by: Helena | witnessed by: | date:

Ball picker

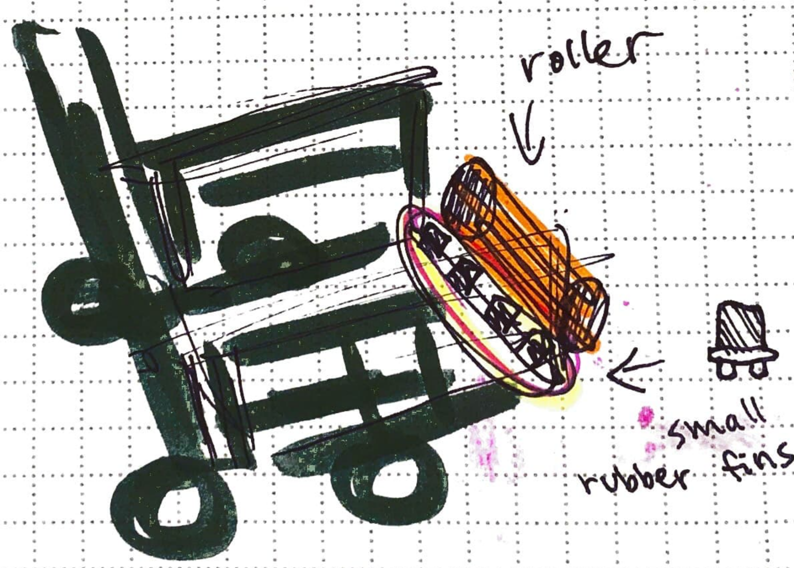
R… N

prob…

As we m… …ack
in a gra… …umber of
balls, th… …ore balls
This mak… …ent.

we nee… …and
pile a… …llected.

Lift for Pick



Lift for Pick

Our idea was to add small rubber fins to re-direct the balls up ward (to solve the initial problem and to be able to collect more balls)
We ALSO expanded the roller a little wider so balls wouldnt get caught between the gear.

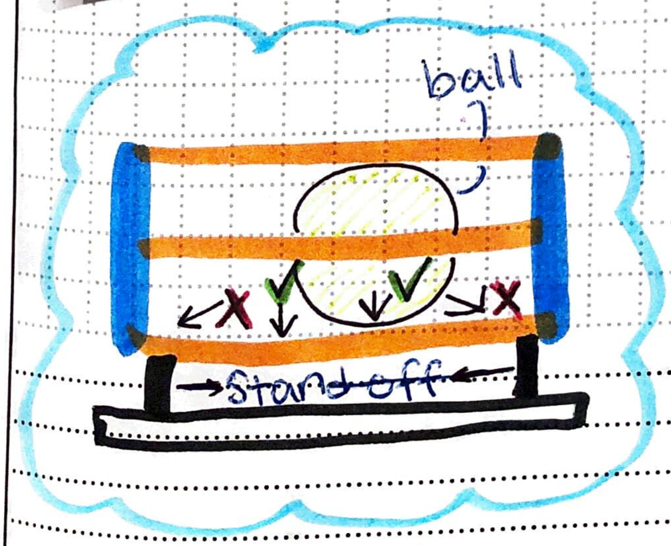
project
designed by: Helena
witnessed by:
date:

ROLLER : STAND OFF

Problem Another problem we ran into with the roller was the balls getting caught in the gears. This would clog up the roller. We needed to find a way to make sure the balls don't go near the gears.

SOUTION

Our idea was to add a stand off under the gear to keep the balls away from there.
When we tested this idea, we ajusted where the stand off needed to be.
Sometimes the stand off would not be enough to block the ball. So we will keep monitoring the roller.

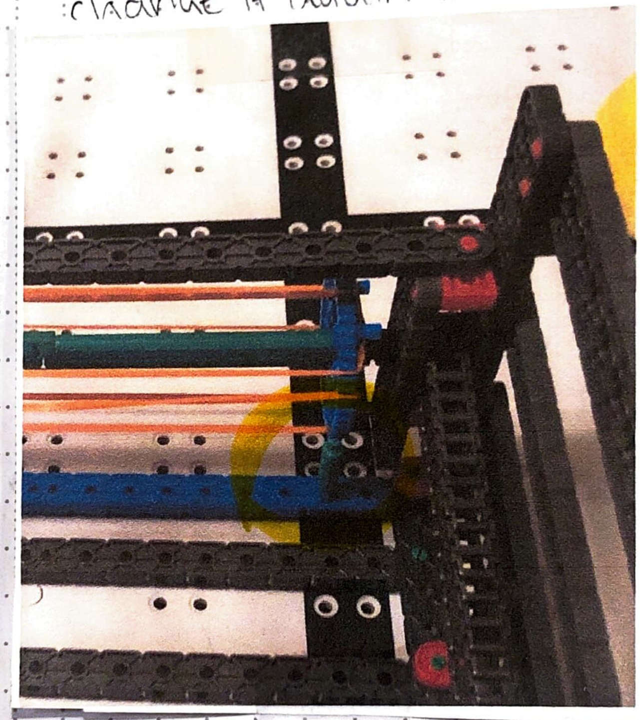


project designed by: Kristen witnessed by: date:

Ball picker

OUR OTHER IDEAS

We have learned that it "take many trials to find the "perfect" design. Even after choosing the third idea, we might change it again in the future!



First Idea: We tried to cover up the gear with a bar/beam. But there was no where to attach it, so we knew there must have been an easier way.

Second Idea: We tred to put a bar/beam horizontally under the gear, but this didn't fully block the balls.

Third Idea! This was the idea we have now. It is to put a stand-off under the gear. Vertically. Although it is sometimes un-reliable, it works the best out of these three ideas.

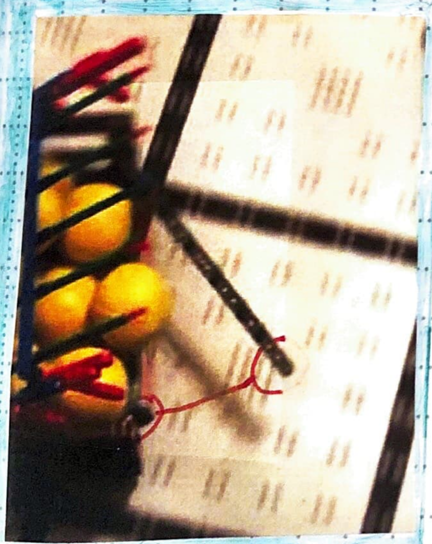
project

designed by: Kristen

witnessed by:

date:

Standoff Bar



Problem

When we were driving the robot and collecting balls, the standoff bar snapped off. This happended many times throughout the driving session. Since the bar was used for containing the balls, we had to fix the problem.

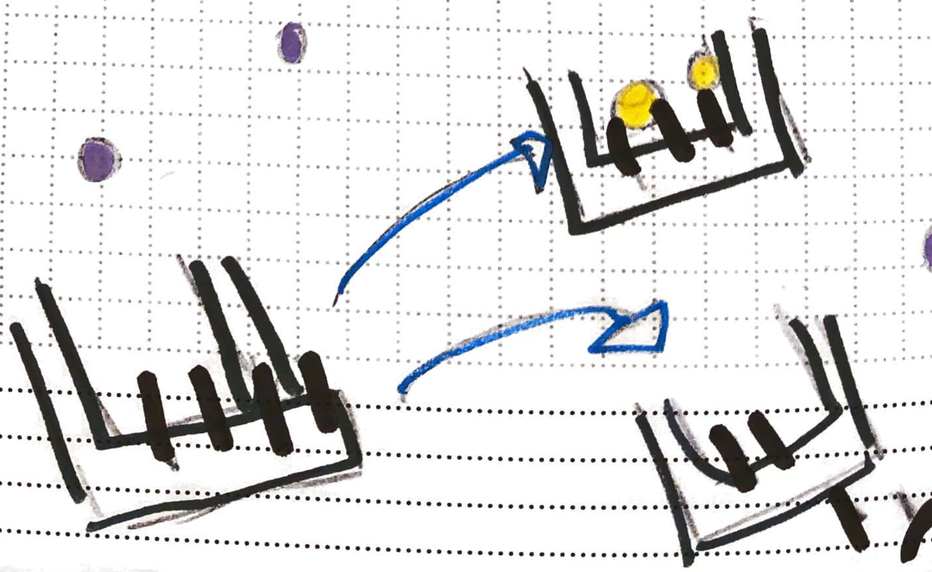
Solution

The solution was to reinforce the pegs connecting the bar to the robot. It worked relativetly well as the bar didn't swing off anymore.

Ball picker

STAND-OFFS

The balls were getting stuck between the standoffs and they kept breaking.



project

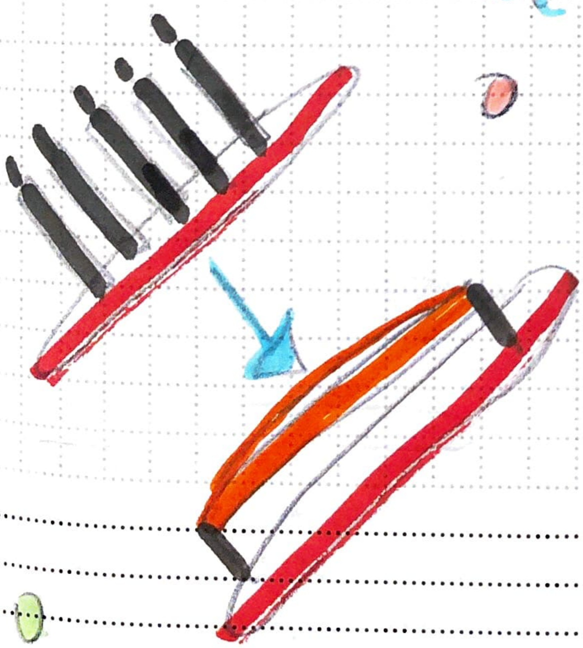
designed by:

witnessed by:

date:

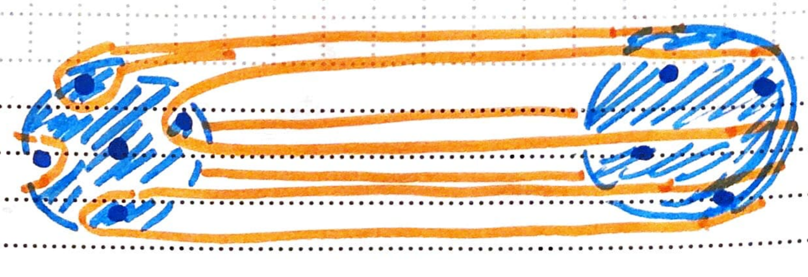
SOLUTION

In order to fix this problem, we replaced the standoffs with rubber bands. The problem with this was, we couldn't have the arm go all the way down.



WHAT WE DID AFTER THE CHANGE IN RULES

We decided that keeping the rubber band roller we've been working on will work, even after the new rules. We might need to change the size off the roller though. We think since the way we collect the balls won't change, the roller will work!



project

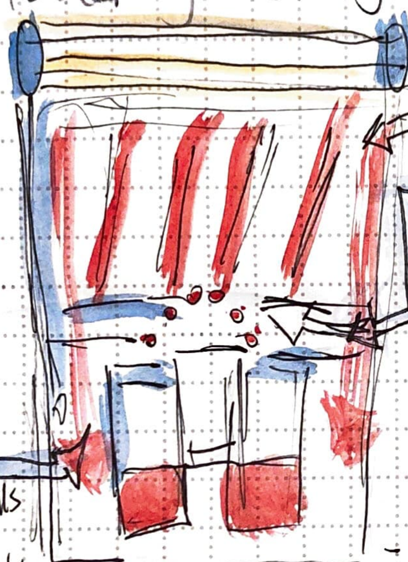
designed by: Kristen

witnessed by:

date:

Roller problems

We were having some trouble getting the balls from the top of the cage to the ~~loading~~ flinging shooters.



Angled downwards so balls can slide through easier

The arrow is supposed to help the balls go down, but instead the balls often ~~got~~ stuck on the first two.

Many balls got stuck here, or kept on pushing on eachother mid-load, resulting in a butchered, short fling.

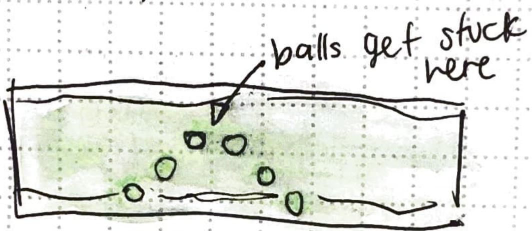
We thought of many ideas, but either they didn't work or they required extra motors!! :((

Wait...... then what?

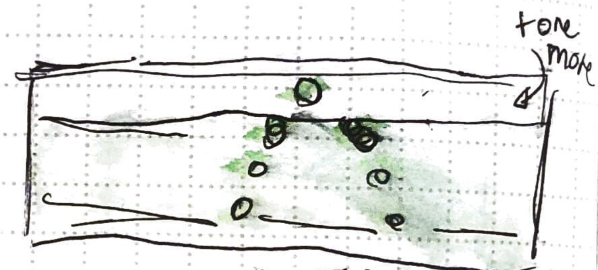
{Solutions}

DON'T FEAR, SOLUTIONS are here!

- The arrow problem actually had a really simple fix!



OLD
(isn't steep enough, more space to get stuck)



NEW
(steeper, less room to get stuck)

- The second problem was quite hard to find a solution. We tried Kristen's fancy idea of using the angle of the cage and flingers to let the first ball through and not the second one, but I thought of something else.

{MINI ROLLERS!!}

The rubber bands stopped the second ball from interfering, and let the balls in when you commanded them.

There was only one problem:
it needed a MOTOR!

Pauses & MOTORS

VEX IQ

MOTOR substitute:

We came up with an awesome idea: chains!

If we hooked the big roller's axle and gear with the small roller's axle & gear, they could run together, letting the balls in while also loading the flinger!

Pausing: A driver's precise skill

Instead of holding the buttons to constantly shoot and load, because of our mini roller and cage design, you must wait for one ball to come in, shoot, and then let in another ball. If we held the shoot and roller spin button, the balls could get next to eachother and interfere with the strength of the throw, or the ball might be halfway on and the process would get butcherd.

THE WAITING ROOM

(Belongs in brainstorming section but ran out ~~of~~ space)

This robot needs a name!

I thought about it for a long time, but an alligator is too basic, and the "head" cage is too big to be another animal. We needed something of Historical value, and I heard someone say the cage was like a waiting room for balls.

So, I thought something very far-fetched, and researched about it. It may not be the final name, but it has potential.

Masque of the Red Death

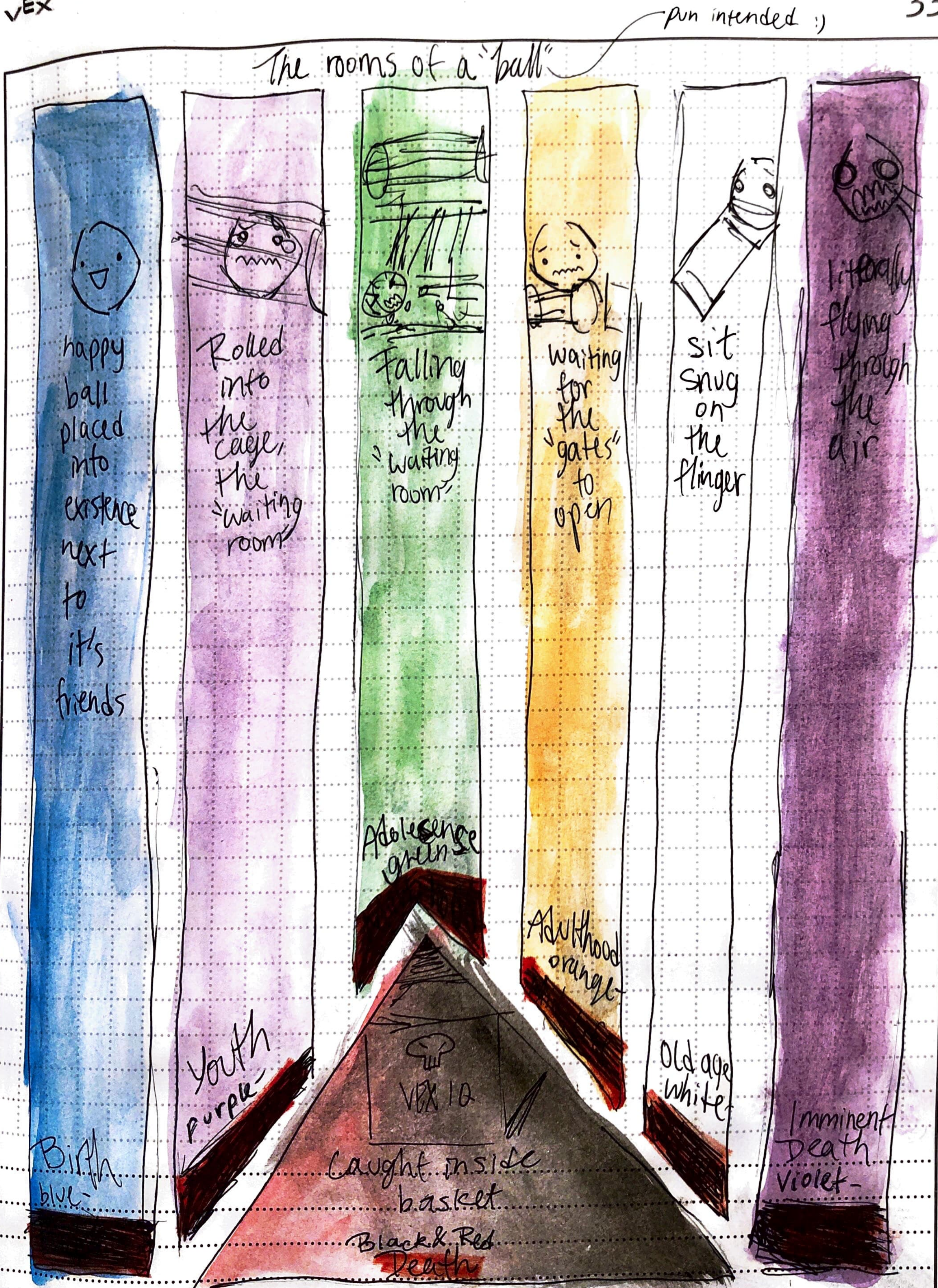
by Edgar Allen POE

The story is about Prince Prospero holding a ball in his utmost secure palace, trying to hide from the Red Death, which is a plague that kills people. His Hideaway place has 7 color-coded rooms, the last with an ebony clock that nobody can stop to cheer up the mood. The clock represents the fate of Death, and the rooms represent the stages of life, eastmost colored blue for birth and westmost Black and red for death. In the end, Prince prospero is inevitably dragged to the last room, and dies from the plague's touch.

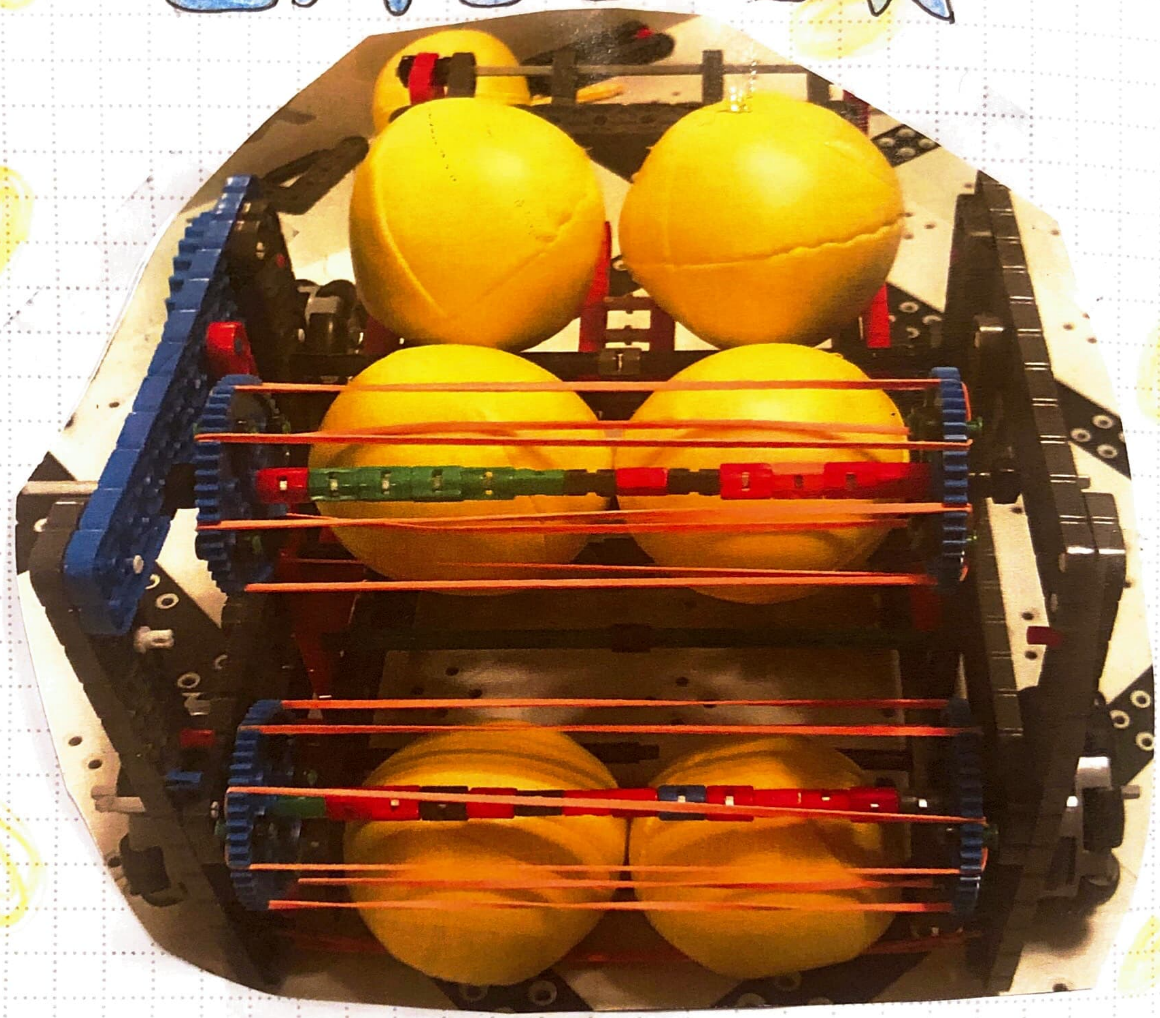
I think the stages of the ball's life could represent the inevitable death or, inevitably getting thrown in the basket. What do you think? "Rooms" →

project designed by: Renee witnessed by:

date:


pun intended :)
The rooms of a "ball"
happy ball placed into existence next to it's friends
Birth
blue-
Rolled into the cage, the "waiting room"
Youth
purple-
Falling through the "waiting room"
Adolesence
green-
waiting for the "gates" to open
Adulthood
orange-
sit snug on the flinger
Old age
white-
literally flying through the air
Imminent Death
Violet-
VEX IQ
Caught inside basket
Black & Red
Death

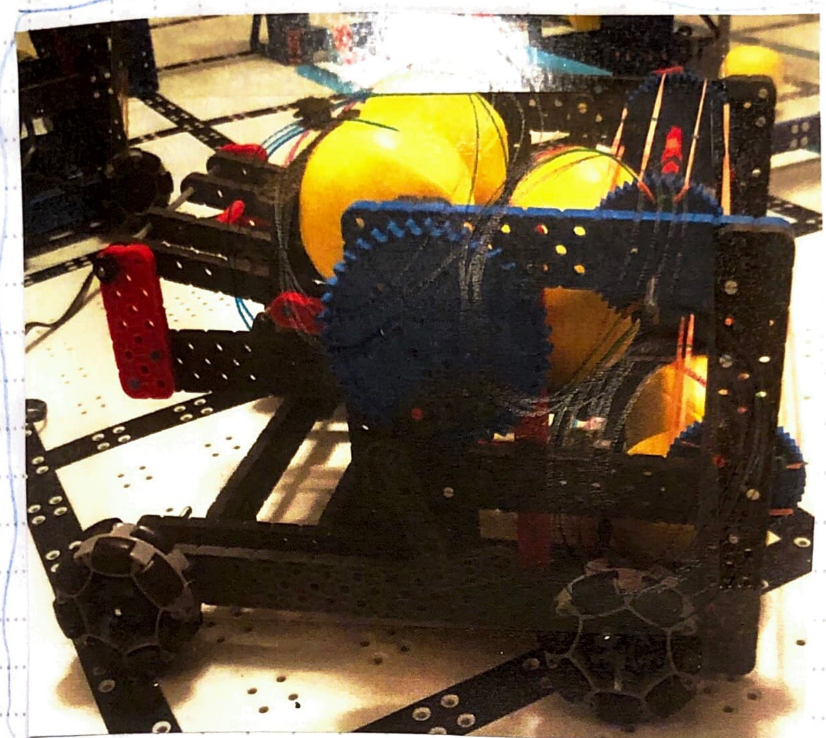
THE ROLL LADDER



This is a design to deposit and pick up the balls. It picks up 2 balls and as the rollers rotate, the balls move up until they are loaded. Then, we can shoot.

project designed by: witnessed by:

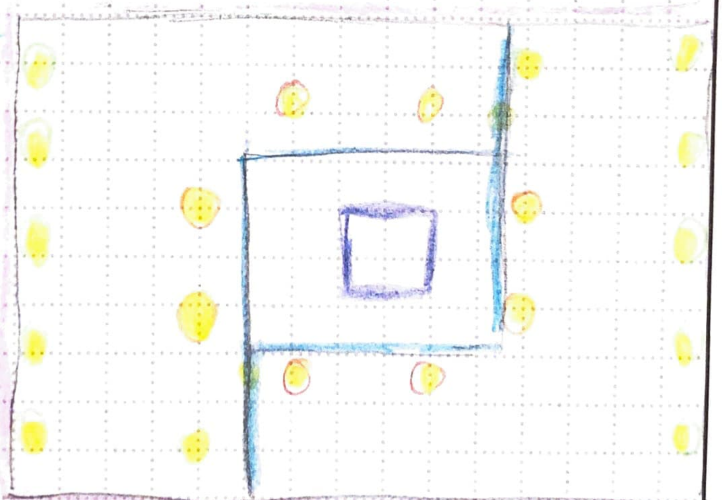
date:



SIDE VIEW

This is a side view of the design. You can see the gears running this design here.

STRATEGY



So, the design picks up 2 balls at a time. The balls outlined in green are easy to pick up. The ones outlined is orange are harder to pick up. The red balls are extremely hard to pick up as there is a strong chance it could be pushed inside the goal. We are also thinking of stopping every 6 balls to go and shoot. This will be pretty effecien

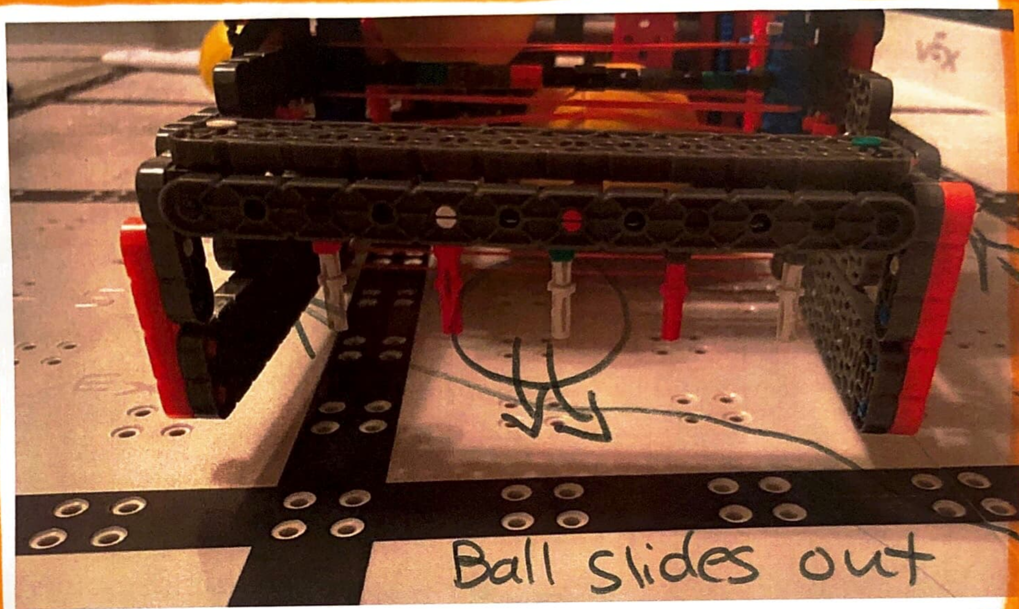
project

designed by:

witnessed by:

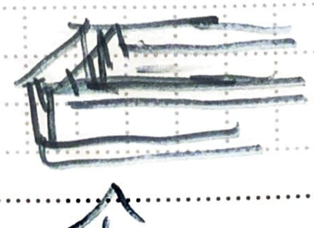
date:

Ball slides thru teeth



To solve the ball sliding out of wait room problem, we reinforced connectio to the main part of the robot

Before



After

project

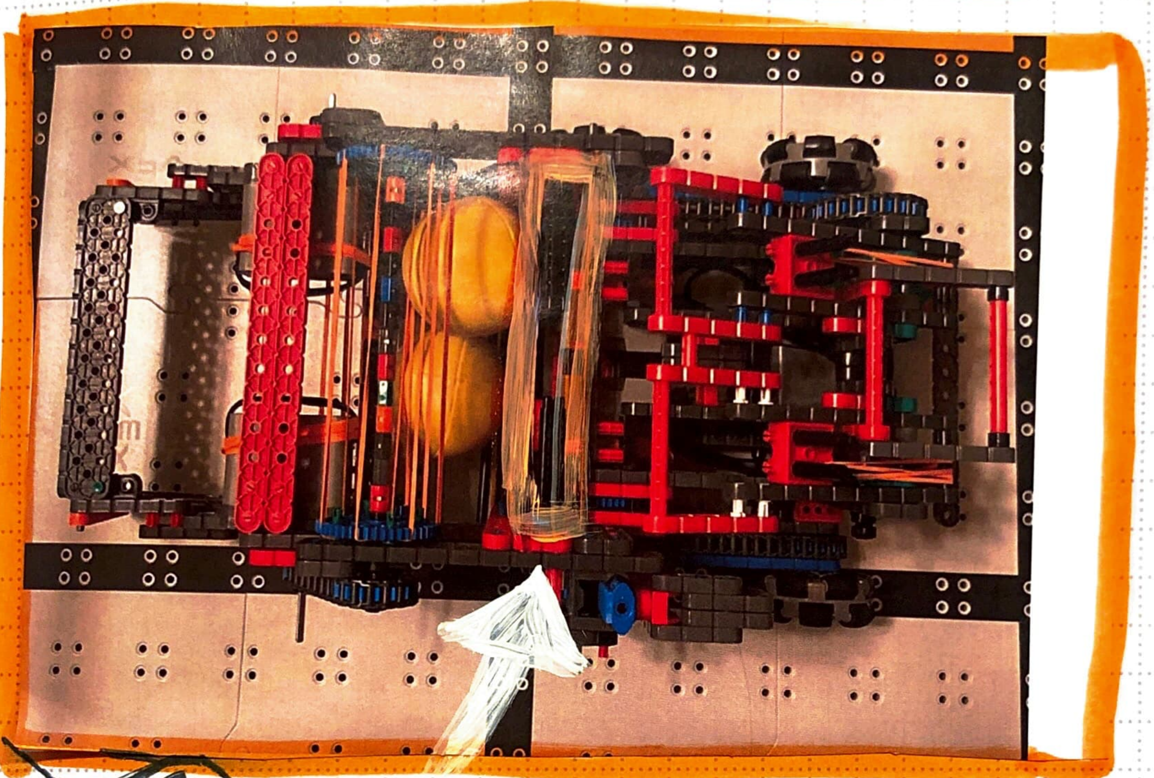
designed by: Helena

witnessed by:

date:

Ball picker

Stuck on roller ladder:



~~Nested between standoffs~~

Fits snugly :)

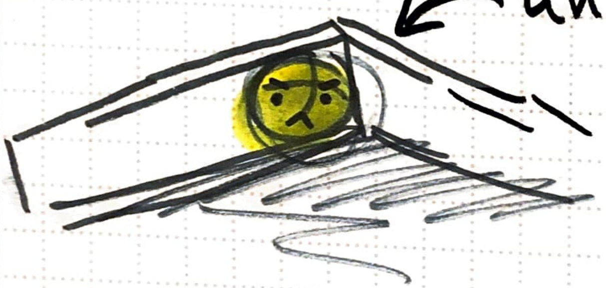
when this happens, we are unable to spin the balls onto the "launching pad" aka catapult

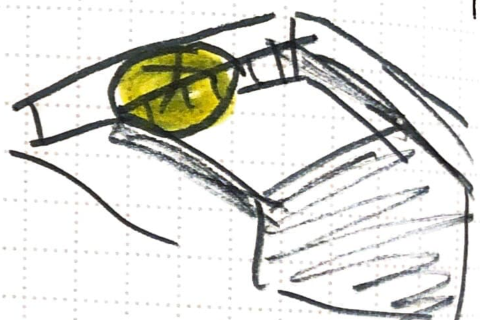
Ball picker

UNSOLVED (For now)
UNIMPLEMENTED

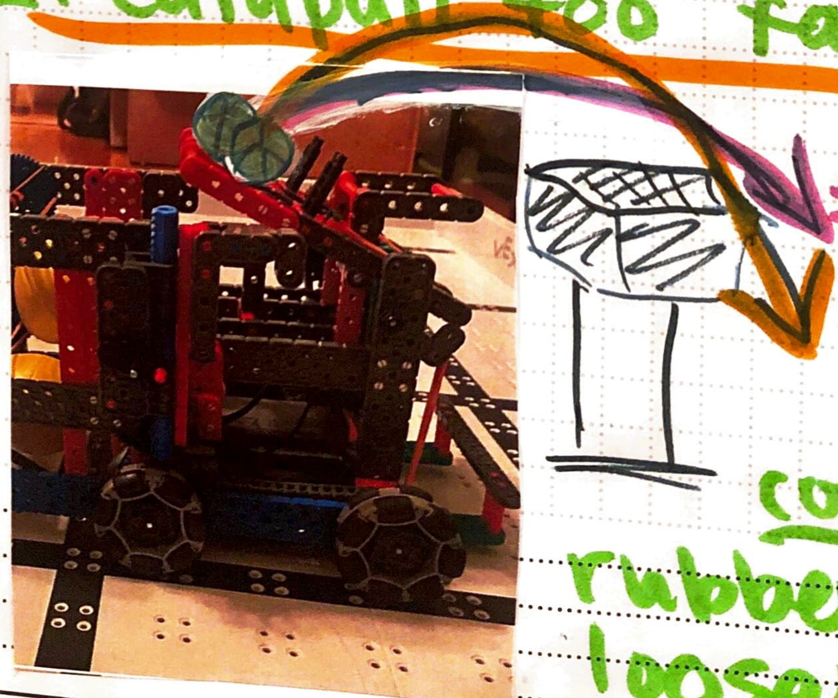
1. Stuck against wall

unable to be picked up





2. Catapult too far/high



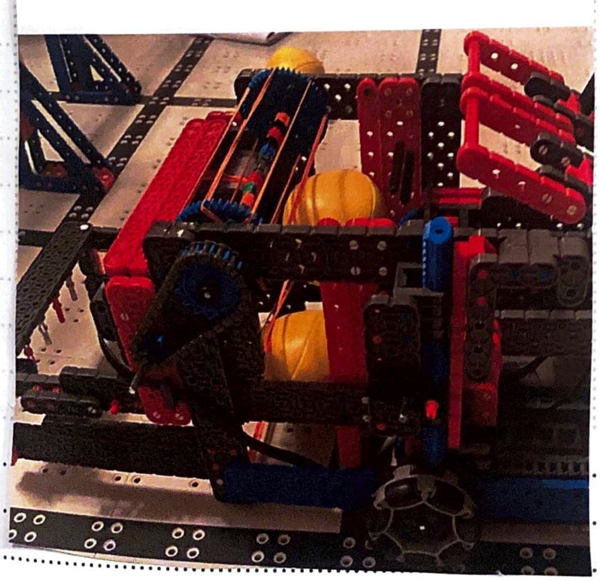
could fix w/ rubber band loosening

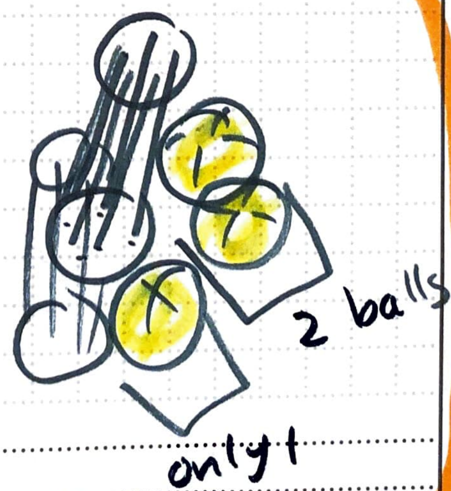
Shooting buttons tedious

Could solve w/ prgm.

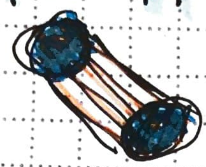
If we could program one button to be in charge of picking + placing in waiting room.

Uneven ball placement



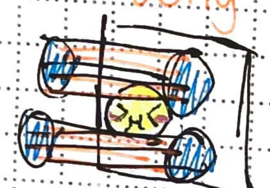


BALL ROLLER PROBLEMS



Many balls got stuck or needed extra time to get rolled up.

Why is this the case?



Well, the balls are squishy. The wall distance from the roller is very narrow, so the ball gets squished.

When the balls get squished, they get more compact, and scrape on the wall. The balls need more power in order to get up, therefore the roller either stops or moves extremely slow.

Solution:

Make the distance wider, obviously!

No more stuck!

project

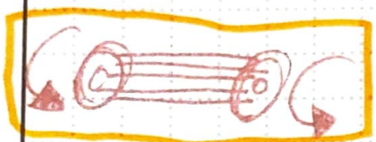
designed by: Helena

witnessed by:

date:

FLING

HOW IT WORKS...

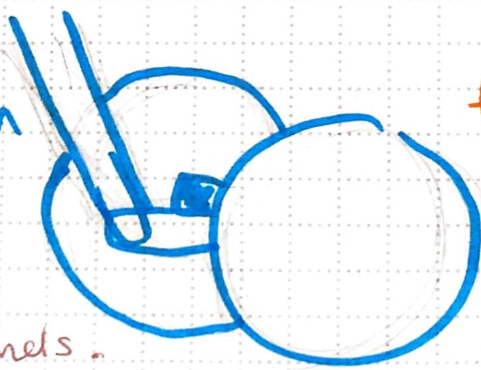


First, the Fling collects balls with the "vacuum", which uses rubber bands to catch on to balls.

second, a pair of hooks/claws attached to the arm lifts the balls and launches them into the High goal.

FLING MECHANISM...

AKA the Choo-Choo Train



This is when the robot flings

The arm is put under pressure with rubber bands. so it naturally wants to go backwards.

But when the Motor turns the gears, the pegs on the gears prevent the arm from flinging. However, when it reaches the point where the bar is horizontal, the arm is no longer stopped ~~froe~~ the peg, and flings!

designed by: Helena !! witnessed by:

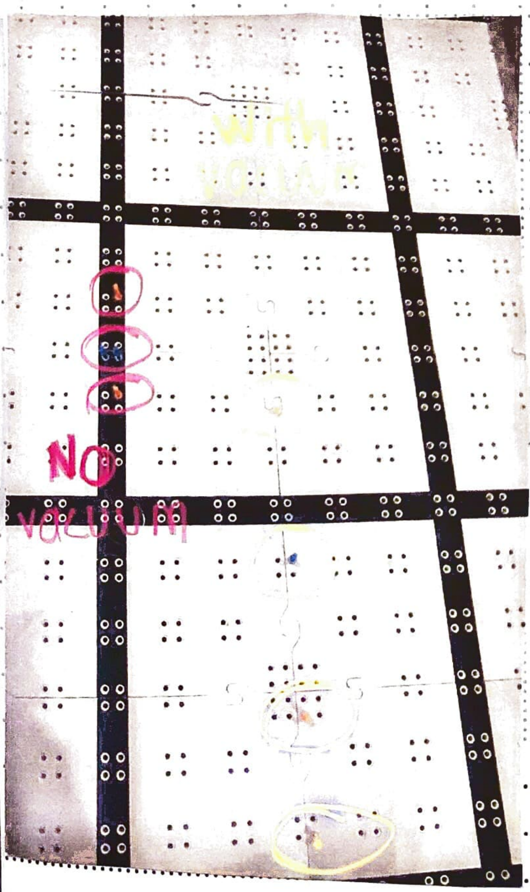
date: 7/1/21

Problems & Solutions

1. Inconsistency !!

OBSERVATION: Every fling, the ball brushes the vacuum rubber band.

I conducted an experiment testing the fling landing spots with and without the vacuum!

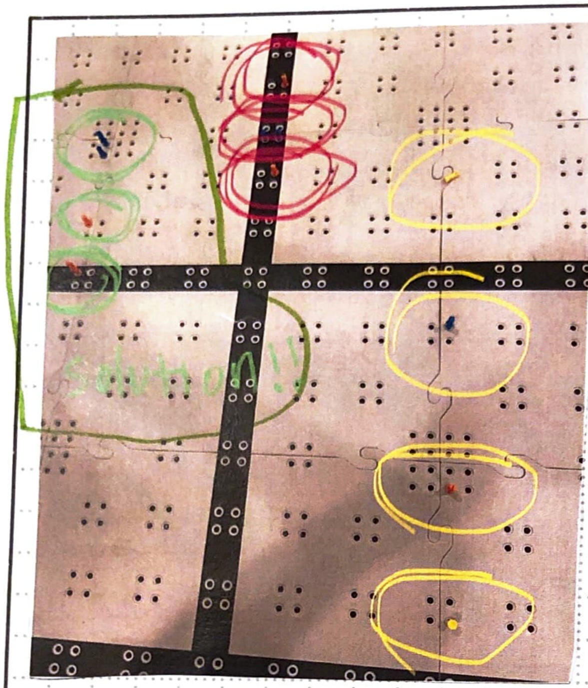


The tests without the vacuum were overall more consistent AND farther.

As a solution, I replaced the vacuum support beams (1×12) into 1×13 beams.

CONTINUED...

CONT.



as you can see, the solution solved the problem!!

2. Stuck under low bar

In order to access all balls on the field, we need to pass under the low bar. The Fling could just fit if it was shorter by 1 mm!!!

Since there was a stopper bar on the top, and it had no use, (even though the instructions said to attach it) I took it off.

The solution worked!!!

project

designed by: Helena

witnessed by:

date: 7/1/21

3. SUPER SLOW

this arm is geared for POWER instead of SPEED.

a solution would be to change the gear ratio (lower it so it's faster)

— so..... slow....

.... seriously!!

4. can it hold 2 at once?

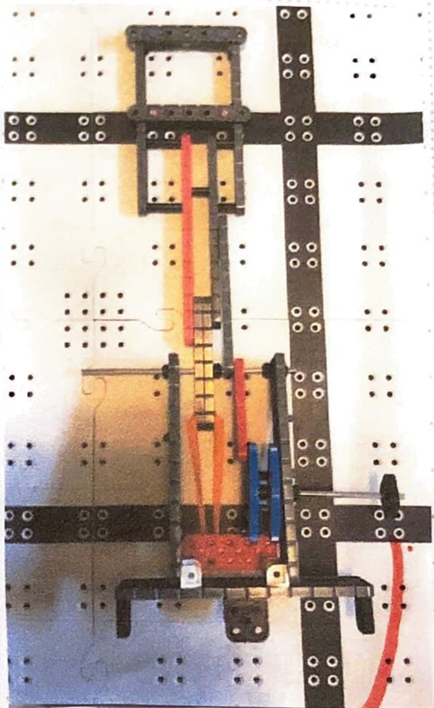
yes... it can, but there was some trouble with picking up balls. they often got stuck on the expanded claw: →

instead of

it's

FLING VARIATION #1

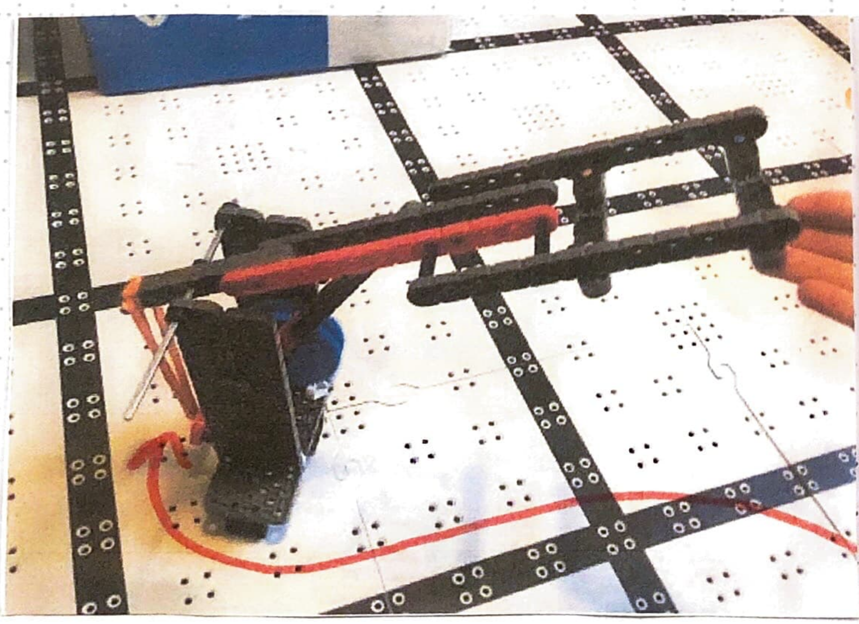
This fling arm shoots one ball at a time. It uses one rubber band's elastic-ness for power. Although there are many, good reasons to use this fling arm, there are cons of this arm too.



Cons:

- not reliable to always shot correctly.
- a little top heavy. If we wanted to add more spaces for more balls shot at once, the top would be to heavy.
- hard to connect to robot.

turning this axle turns the gear, and brings the arm down

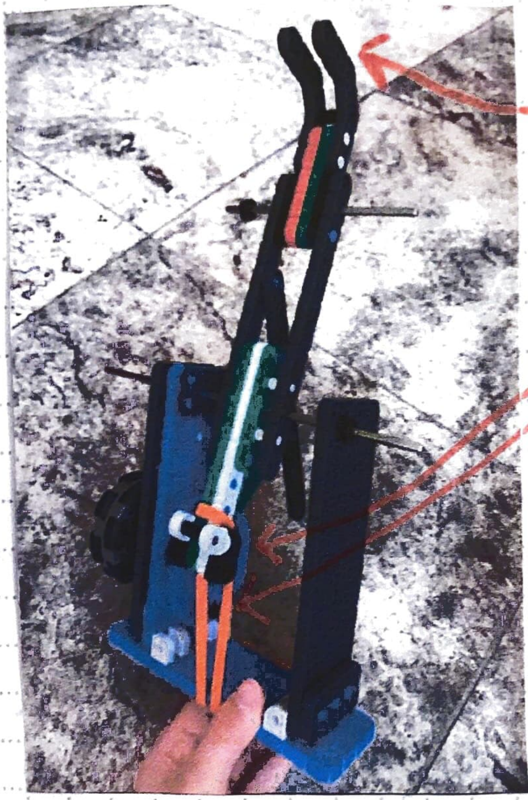


space for one ball

rubber band/ like a spring

continued

FLING VARIATION #2



This fling arm holds one ball here

Fling variation 2 also uses rubber bands and gears to power the arm.

fling variations 1 and 2 were both based off of one idea, then modified for different purposes.

PRO	CON
• Not top heavy • has great range	• can only fit one ball, but can add more spaces • not always reliable

project

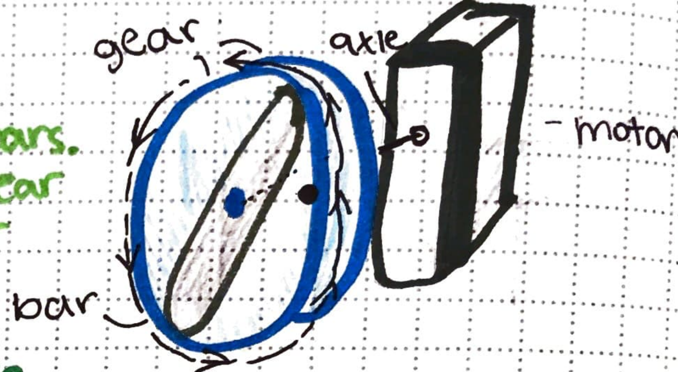
designed by: Kristen

witnessed by:

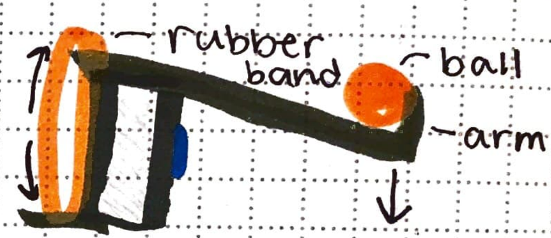
date:

MORE IN-DEPTH DESCRIPTION OF THE GEAR

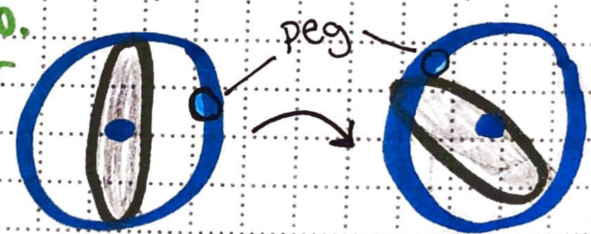
When the motor turns the axle, it also turns the gears. There is a bar on the gear that turns with it. The bar moves a little free-ly.



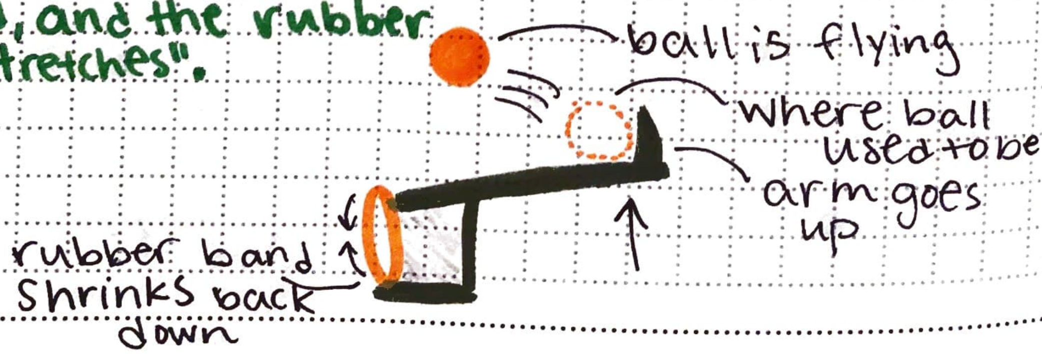
The rubber band acts like a spring. The arm that carries the ball is brought down, adding potential energy.



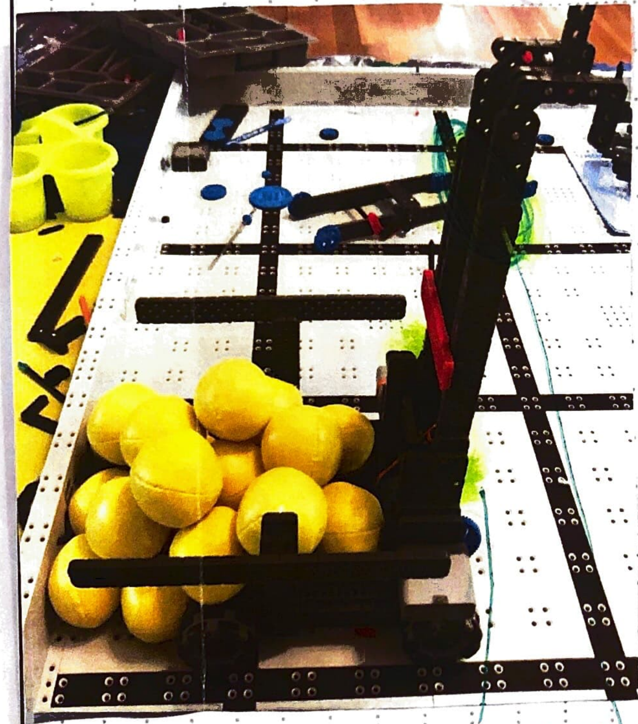
There is a peg on the gear too. The peg slides against the bar on the gear, until it goes over it.



The energy is released, the ball is launched, and the rubber band "un-stretches".



LEVER



This design flips itself over the bar and deposits the balls that way.

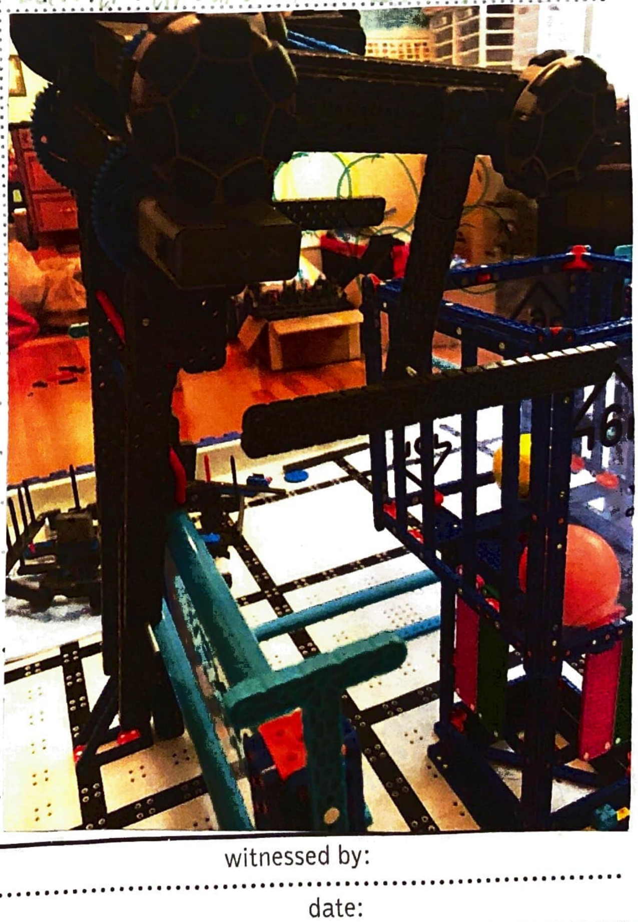
In the picture to the left, you can see where the balls are collected.

We still have to add a barrier around the balls.

This design can also fold over it self.

The benefits is that it can go under the low bar.

The picture above also shows that this design can hold up to 20 balls.



Pros

- it collect many balls
- the robot can fold in on itself and can go under the bars
- the robot can deposit all of the balls in one big swoosh
- can quickly adapt to many different strategies.

Cons

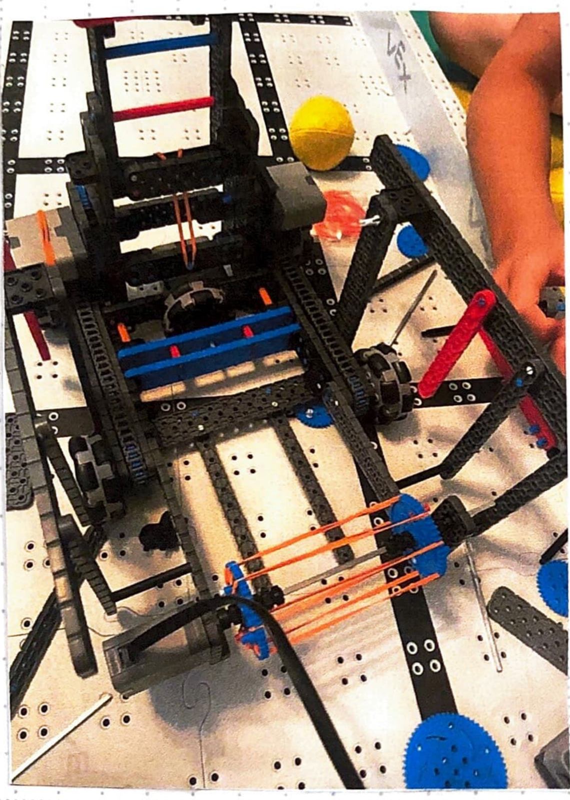
- it is very slow
- to flip the entire robot over the bar will take a lot of power and might exceed the motor limit
- might take two motors to flip the robot
- could result in no shift wheel drivetrain

Overall this design is very flexible, can hold many balls, go under the low bar, and can deposit smoothly. The downsides is that the gear ratio will be very slow and it could take 2 motors to flip resulting in no shift wheel.

project designed by: witnessed by:

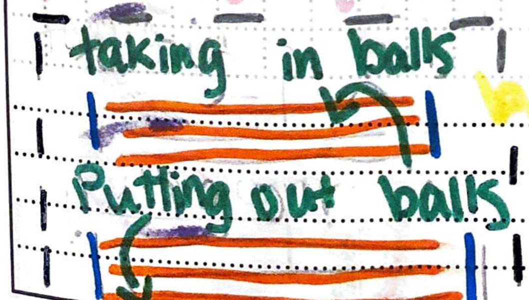
date:

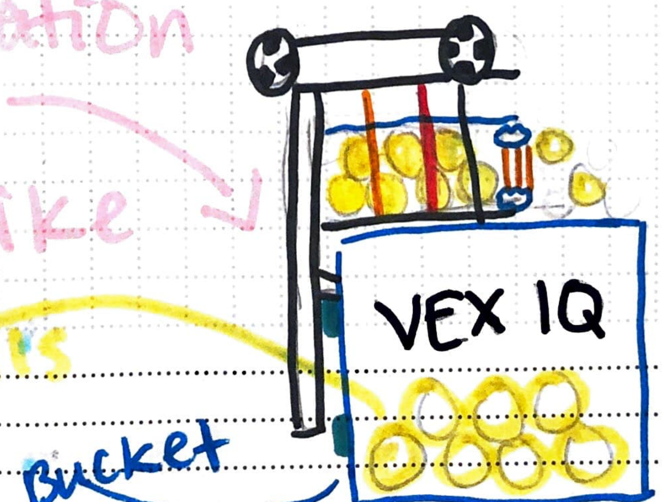
Bucket Idea



The picture to the left is a product of our work so far. After this picture was taken, we added on a few beams to the bottom for when we flip it over. That way, the balls won't fall out.

Rough representation of what it would look like

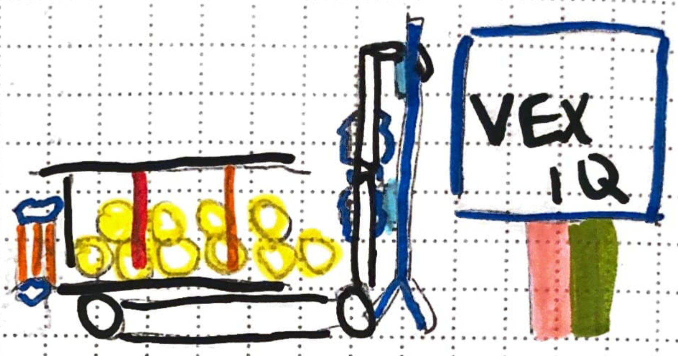




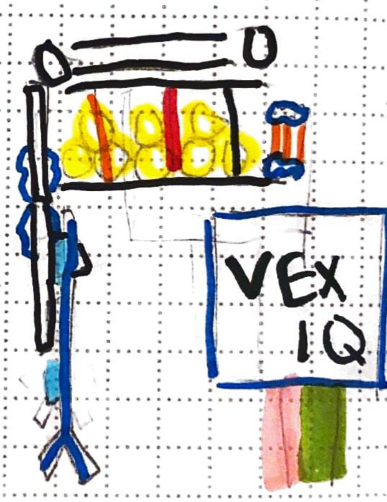
FLIPPING

1.

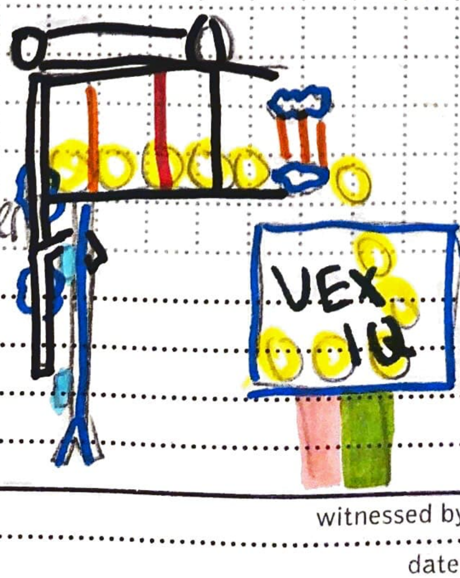
Once you have gained all the balls you want, we will latch onto the high bar.



2. Then, you use the flip mechanism and get the bucket in position.



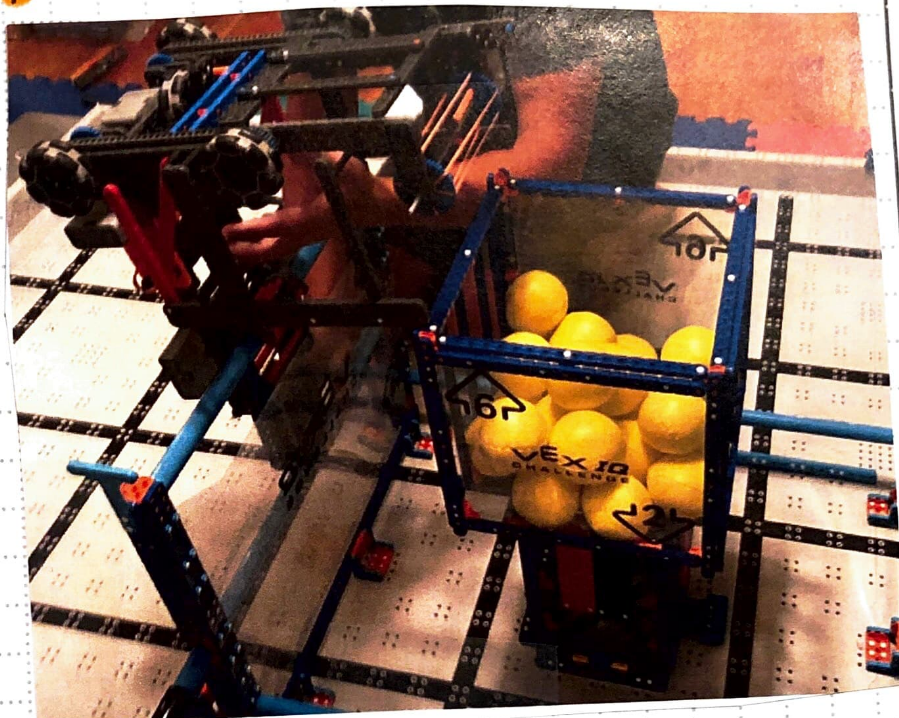
After that, you use the roll/Ball picker and reverse it so that the balls come out.



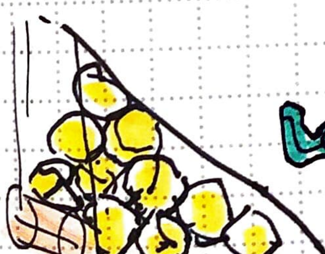
3.

Ramp Idea

In order to deposit the balls, we need an angled ramp to direct the balls into the basket.

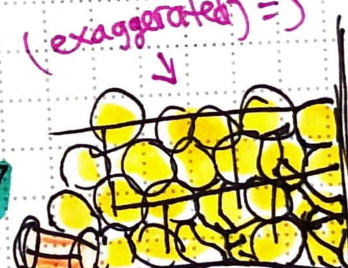


If we make a triangular basket with a ramp like this...



It will contain less balls than if we did

(exaggerated) =)



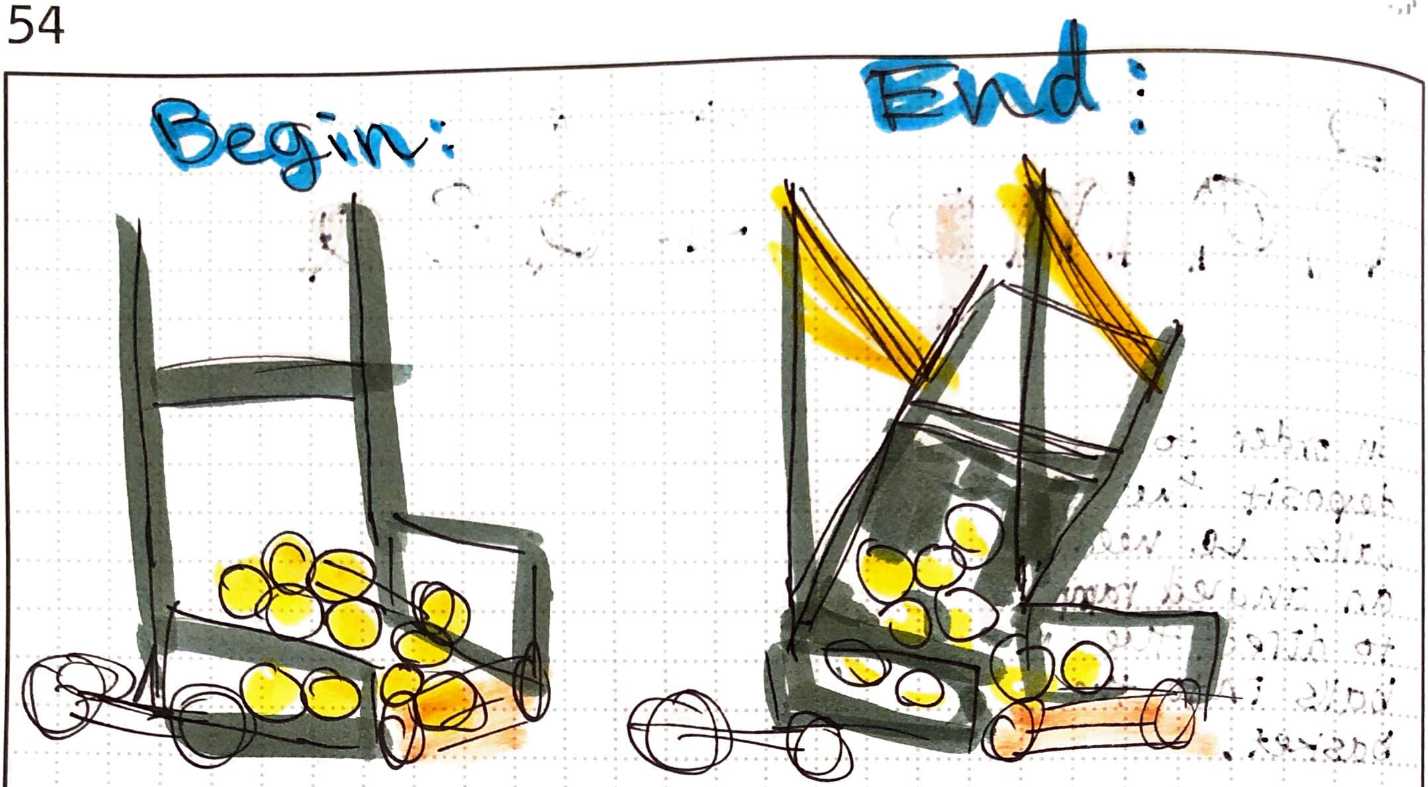
We are thinking about making the ramp retractable and connected by rubber bands. Its like a drawbridge.

project

designed by: Helena

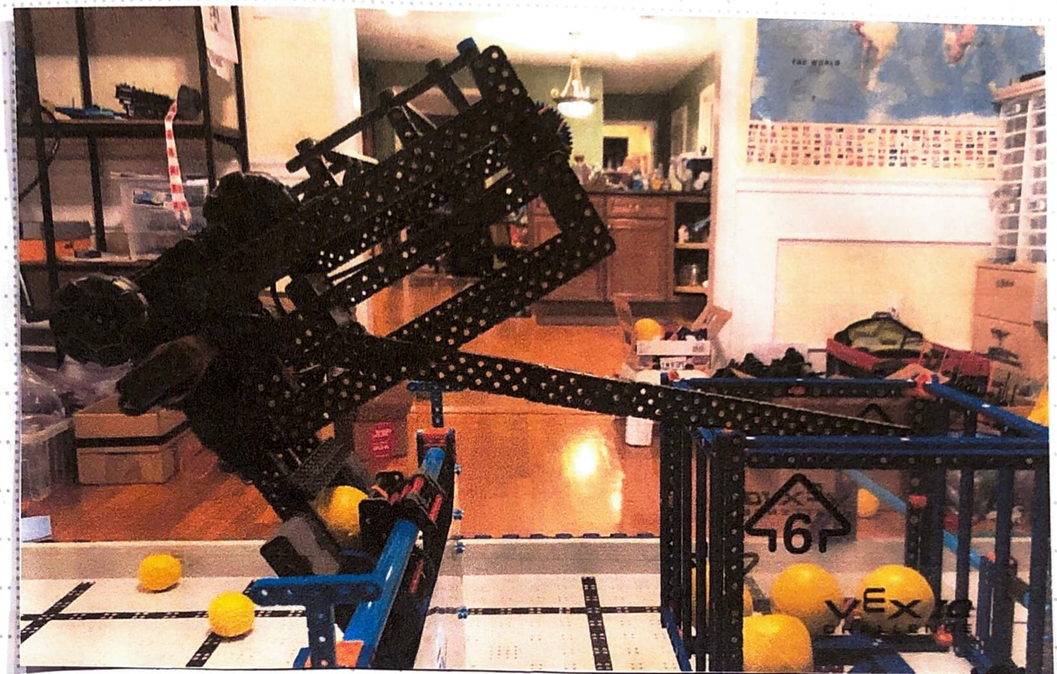
witnessed by:

date:


Begin:
End:


VEX IQ
CHALLENGE

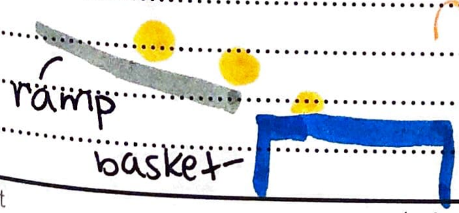
Ramp Continued



We added markings to see how long the ramp needed to be. We learned that the ramp didn't need to completely touch the basket goal. This is because the balls will roll into the basket and not get stuck.

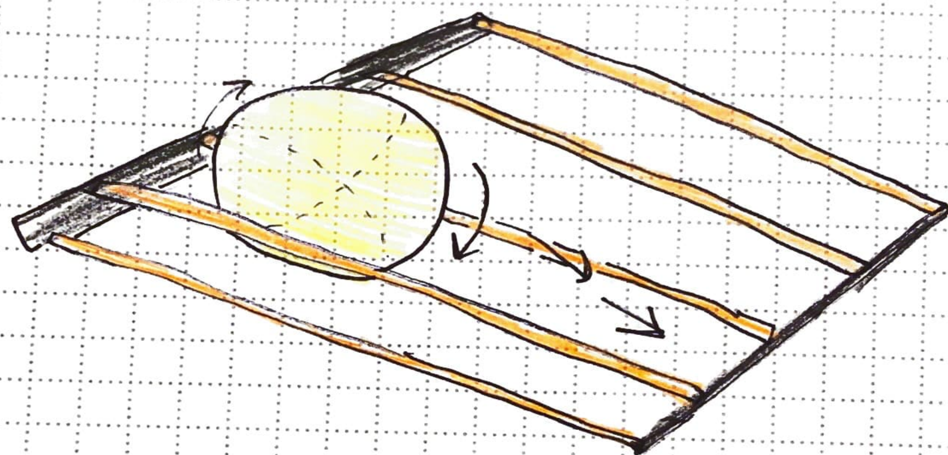
We figured out how much slope we needed for the balls to garentee to roll. Since these balls are not smooth plastic, they needed more slope.

Since the triangular shape of the basket holds less balls than a rectangle, we had to think about a way to make it fit our goal of 18 balls.



BALL LAUNCHER DEPOSITOR continued on pg.

Ball Picker



LIFT TO SEE MORE

We wanted to make sure this cage could have the balls run smoothly while they roll into the goal/basket. Stand-offs seemed to work best when we were testing. Stand-offs are also lighter than beams, it was the best option. We attached many stand-offs together to get our desired length.*

When this robot is flipped over (to deposite balls), the ramp will not touch the basket. The balls will naturally fall into the basket even with a small gap in between the stand-offs and basket.

*Stand offs are attached to a 1.20 bar.

project

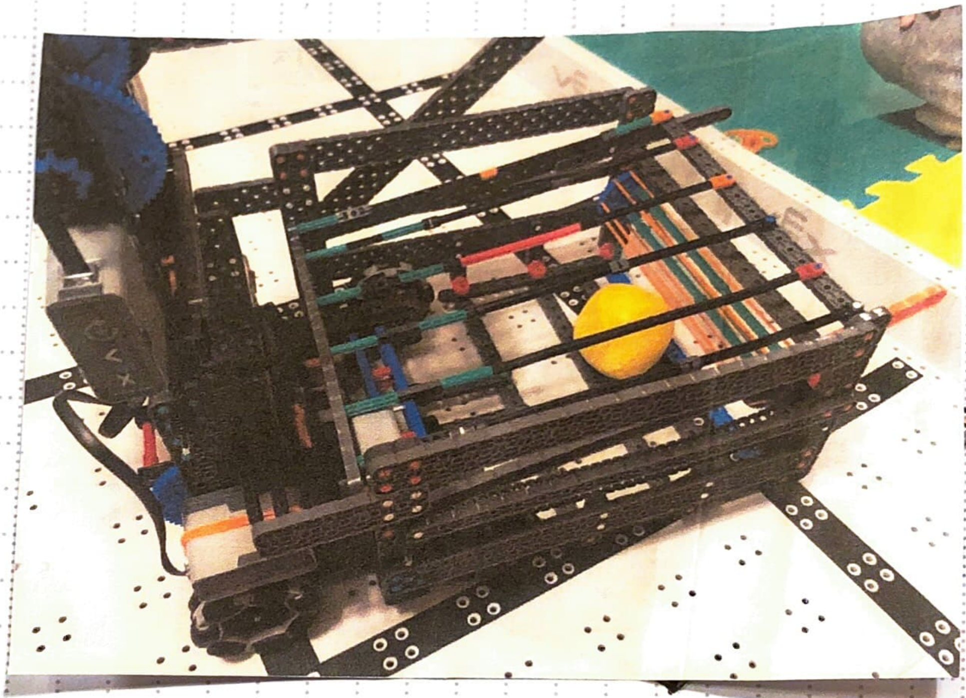
designed by: Kristen

witnessed by:

date:

Ball Picker

ramp/cage idea



LIFT TO SEE MORE

We wanted to make sure this cage could have the balls run smoothly while they roll into the goal/basket. Stand-offs seemed to work best when we were testing. Stand-offs are also lighter than beams, it was the best option. We attached many stand-offs together to get our desired length. *

When this robot is flipped over (to deposite balls), the ramp will not touch the basket. The balls will naturally fall into the basket even with a small gap in between the stand-offs and basket.

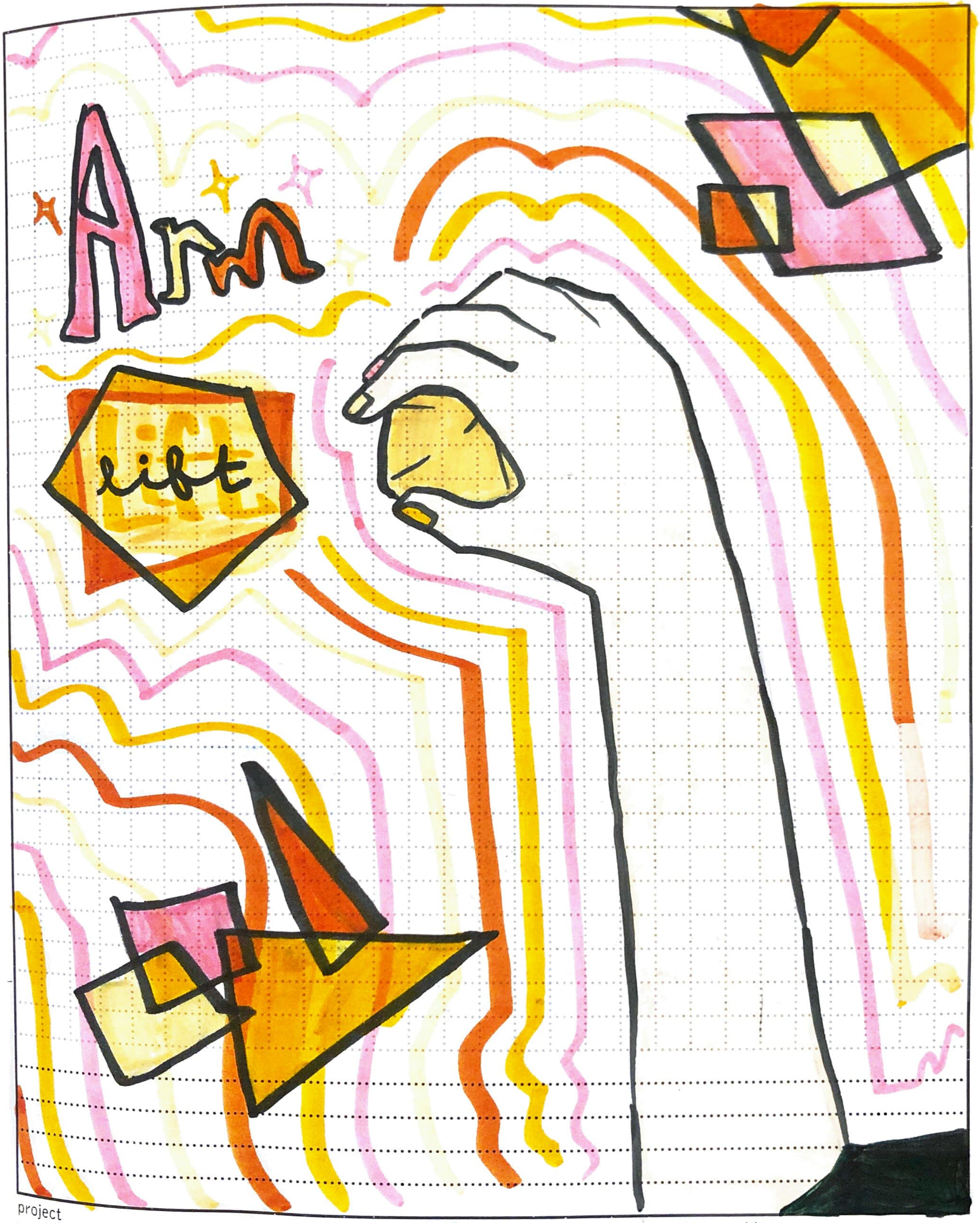
* stand offs are attached to a 1-20 bar.

project

designed by: Kristen

witnessed by:

date:



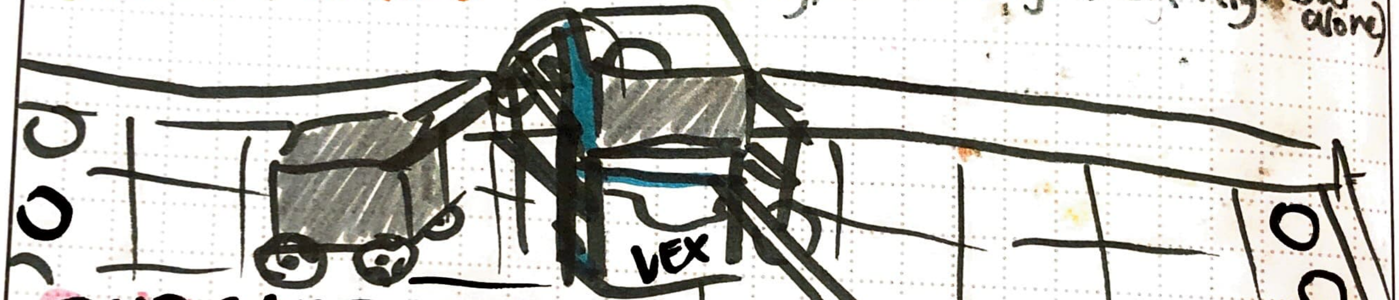
project
designed by: Renee
witnessed by:
date: July 10

THE (genius) TOP BAR METHOD

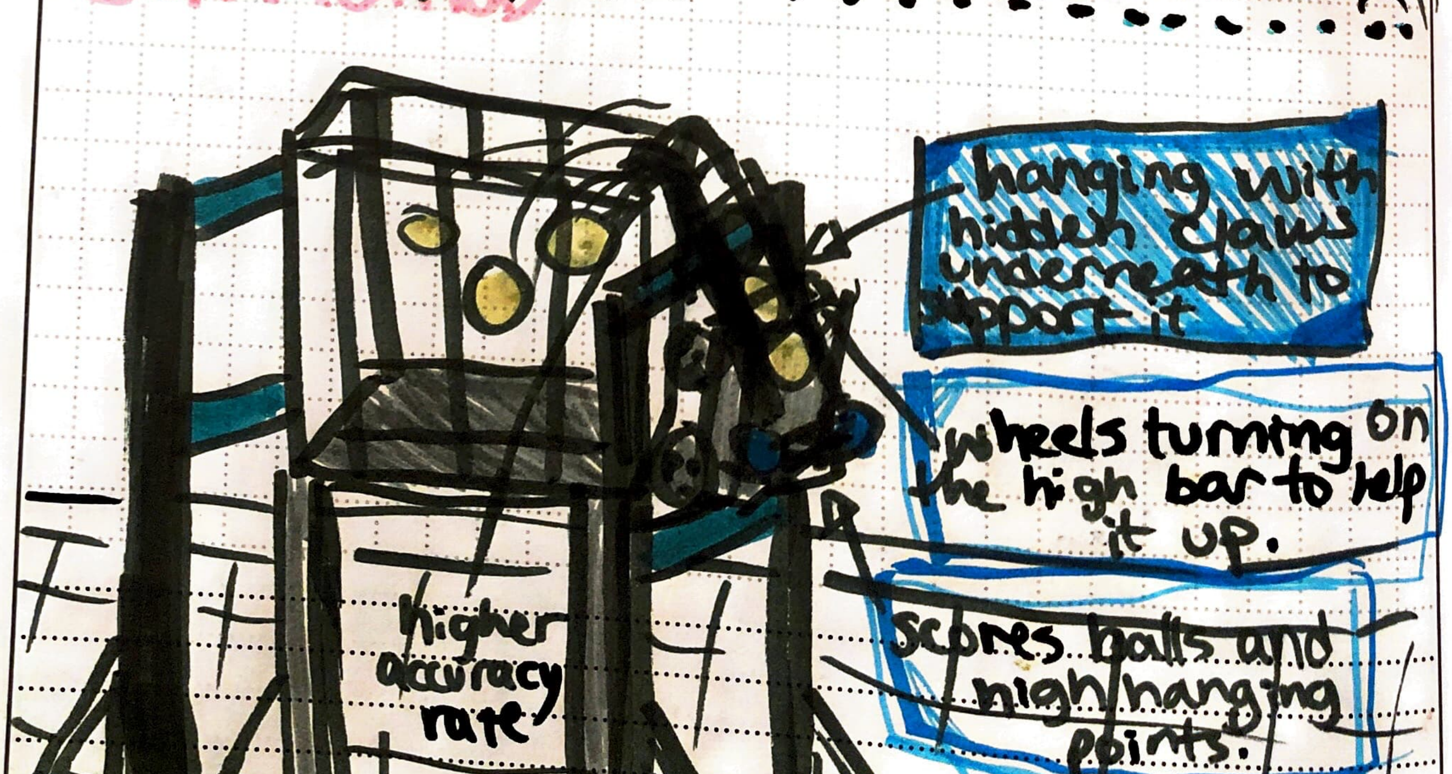
INSTEAD OF FLINGING THE BALLS FROM THE GROUND INTO THE BASKET, OUR TEAM INVENTED A NEW WAY!

First, we collect all the balls, until it is time to hang on the bars. Then, we climb up to the top bar, scoring 10 points. Then, we rapid-fire the balls into the basket with 100% accuracy!

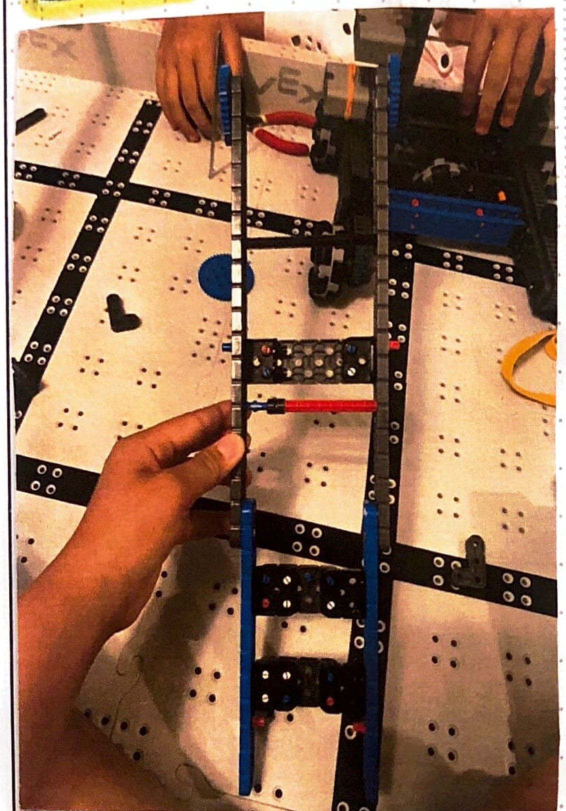
FLING METHOD (less accuracy, cannot get on high bar alone)



OUR METHOD

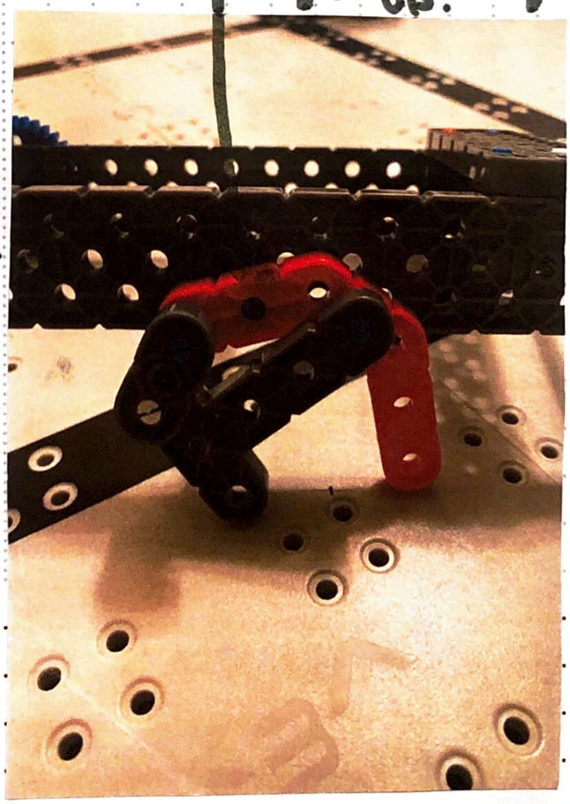


IN DEPTH EXPLANATION



Arm that will be tripled to form 18 balls, while still maintaining enough agility to climb the high bar.

hook that multitasks as both needed positions to pull the robot up.



Currently, we are still brainstorming ways to make the high hanging process easier, but Helena has already shown us proof that the design is actually doable. :)

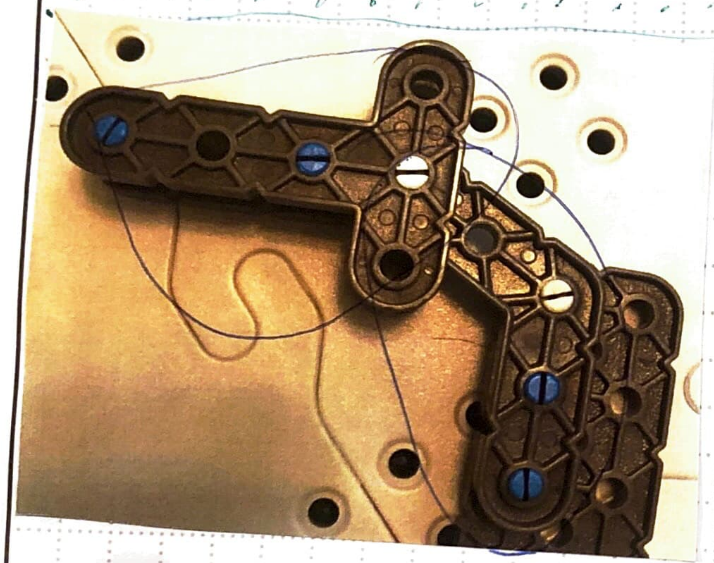
project

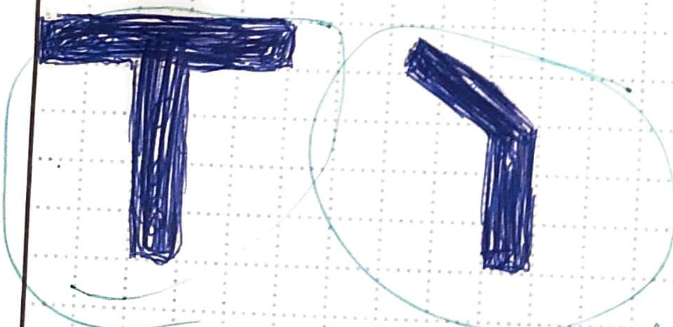
designed by: Renee

witnessed by:

date: July 10

THE NEW CLAW DESIGN





These 2 pieces are used to create this claw design

It is extremely simple and effective. How it works is you move and push it against the bar and it is stuck.

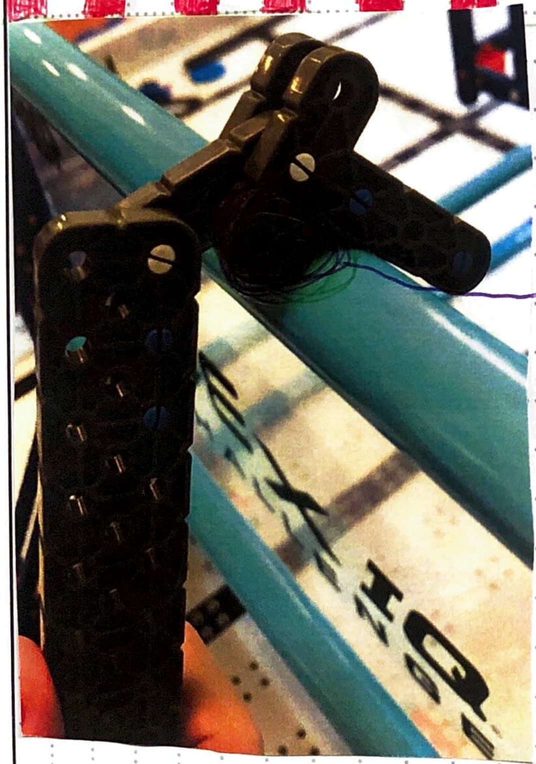
project

designed by:

witnessed by:

date:

HOW IT WORKS



So how it works is that the bar pushes against the inside of the T-shaped piece. When it pushes against it fully, the contraption cannot be pulled down.

This is a very simple design that works wonders. Since it is so simple, if it breaks then we can very quickly put it back together. The problem is we only have one chance to push it against the bar.

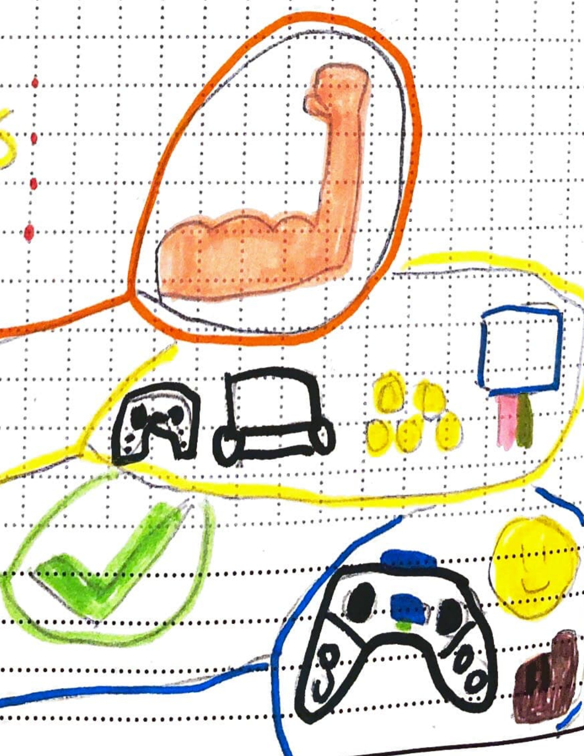
Lift Robot

This new arm is extremely strong and many functions. First, it will have a hook attached to it. We currently have not put the hook on yet, but we have the hook built. Once it is hooked on, the arm will lift the whole entire robot.

Arm Qualities:

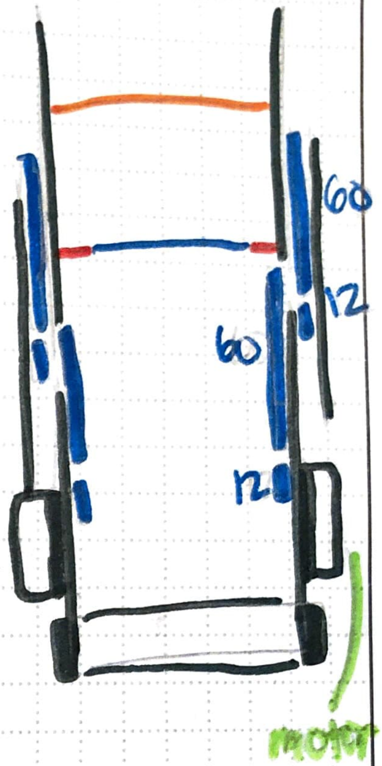
It is:

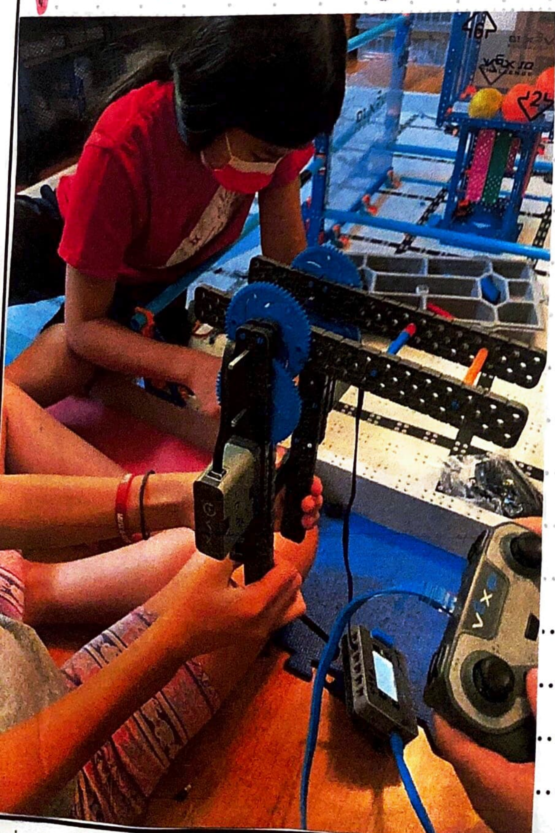
- Strong
- functional
- easy
- Driver friendly



EXPLANATION

As you can see in my rough drawing to the right, there are two motors. This way, the arms may be uneven at times which is a downside. But otherwise, the arm is well rounded.





We have a compound gear. The gear ratio is 25:1. The two gears used are 60 toothed, and 12 toothed. 60/12=5 There are two so you square it. $5^2=25$ therefore, it is 25:1.

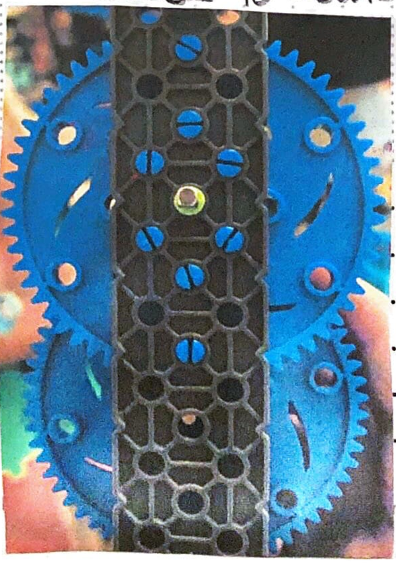
CLIMB MECHANISM!

Hello! As we are currently trying to have our robot climb the high bar, we have ran into a problem. Eventually, we have found the core problem and successfully solved it as a team.

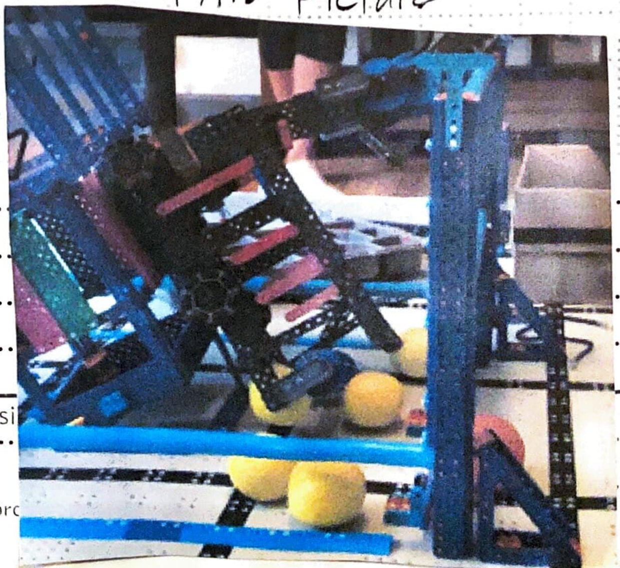
PROBLEM: The problem is that whenever our robot tries to pull itself up the high bar, we hear unusual clicking sounds. This is because the pegs inbetween the arm and robot body are twisting. When these plastic pegs twist, they make an unusual clicking sound. Since the arm is insecure, the robot can tople over all the way. This way we cannot score points.

SOLUTION: Since we video taped this failure on an iPhone, we looked at the playback. We noticed 2/4 of the pegs fell out. Then, we used four additional pegs to secure the arm fully.

8 pegs to secure!



FAIL Picture

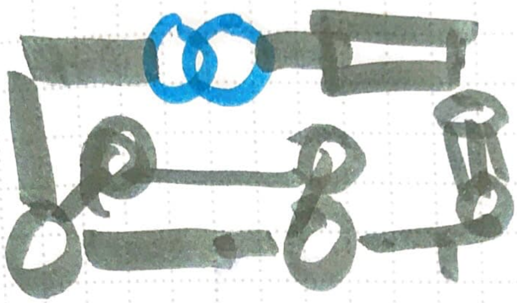


proj desi

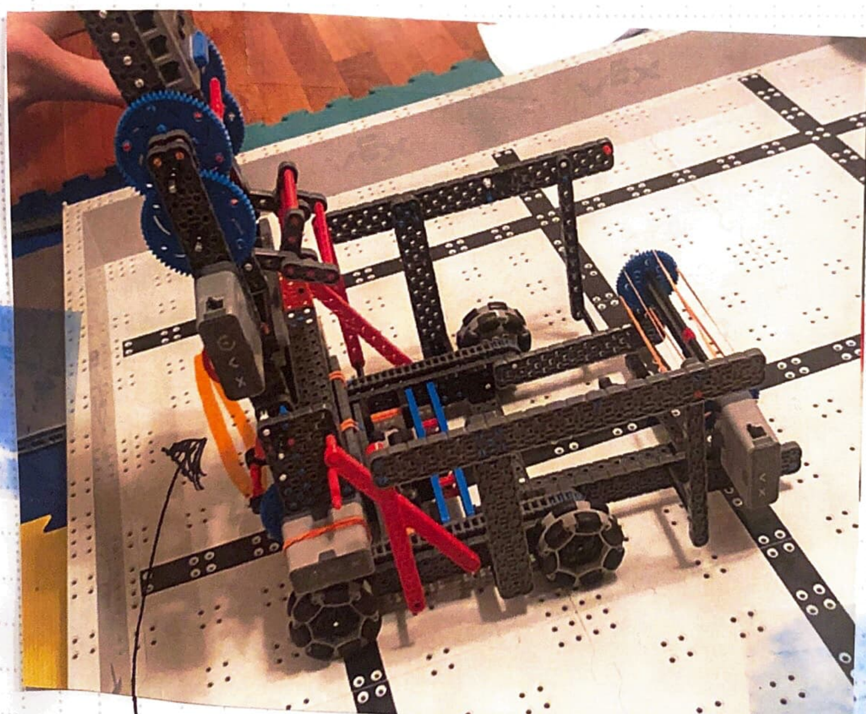
PRO n is the pr

LOCKING ARM

we need our robot to fit under the low bar, so our arm starts out lowered.

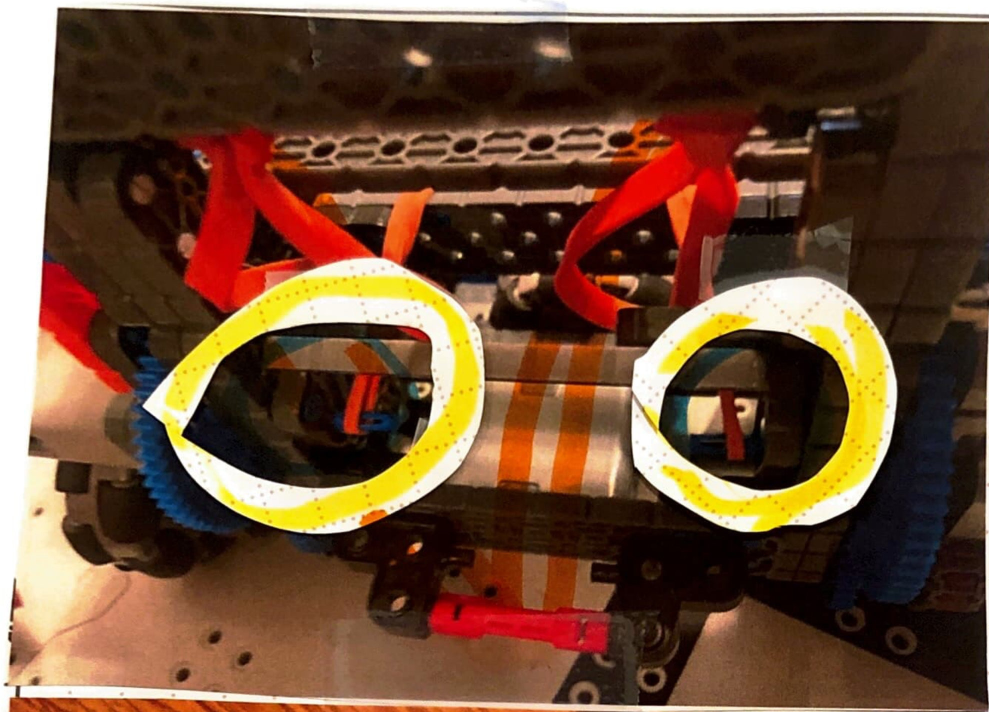


But we need a motorless mechanism to lock the arm into place when we need to raise it up.



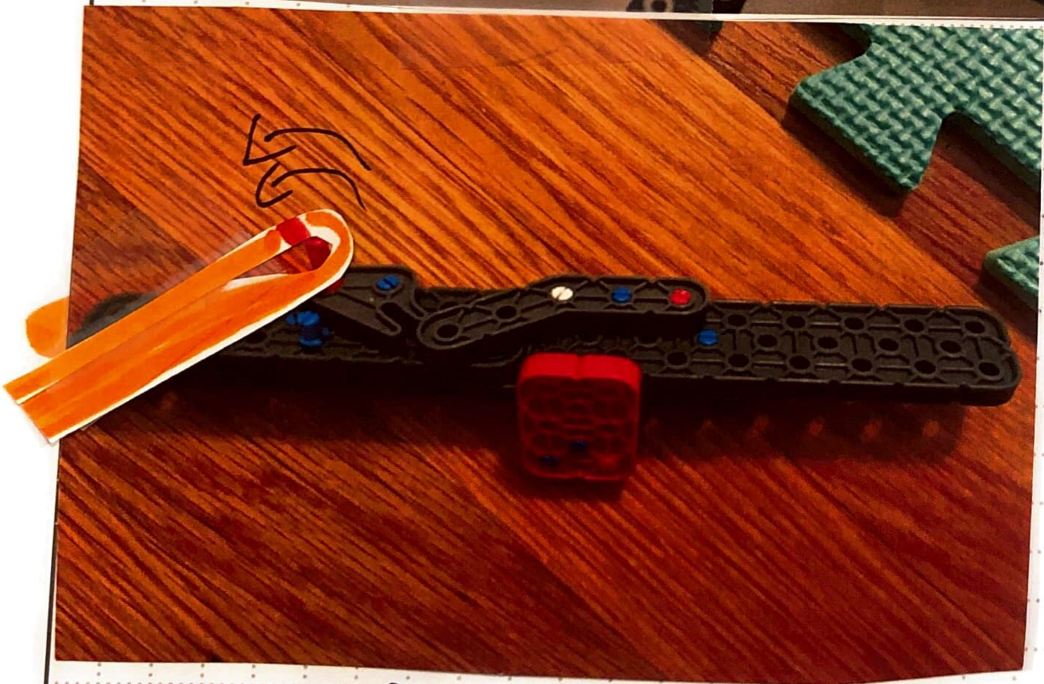
rubber bands spring load the arm (which will lift the arm without motors) But to keep it in place, we need a locking mechanism.

CONTINUED...



As circled to the left, 2 spring loaded blockers are locking the arm in place

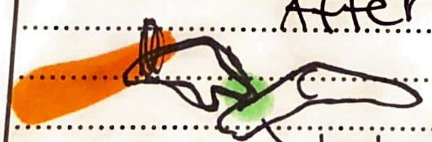
Below is a cleaner example



The peg is held back with a rubber band. when the angle piece is pulled down, it triggers the elbow piece, which snaps and locks in place once the angle piece is past the notch.

Before

After



locked in place

additions

we will have to add a mechanism that holds the arm down at the start and snaps it up later.

project

designed by: Helena

witnessed by:

date:

final flip

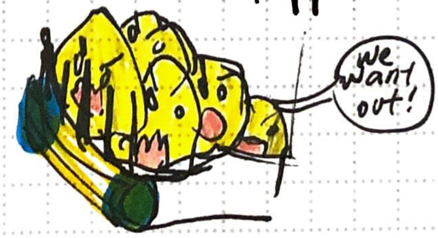
We tested out the flipping mechanism *roller!, and there were 2 immediate reactions:

successful

- We were able to flip over successfully, a very hard task seeing as the cage had 22 balls in it!
- Some of the balls were successfully shaken into the pit (although some were stuck or shaken into the pit below)
- We were able to collect all 22 balls, without exceeding the size limit

unsuccessful

- We weren't able to get all the balls in the ramp had to be at a very certain angle in order to deposit every ball without getting stuck, and even so there was a blue beam stopping balls from going further without momentum.
- When we collected 22 balls, the bar next to the roller stopping the balls from coming out snapped. :(

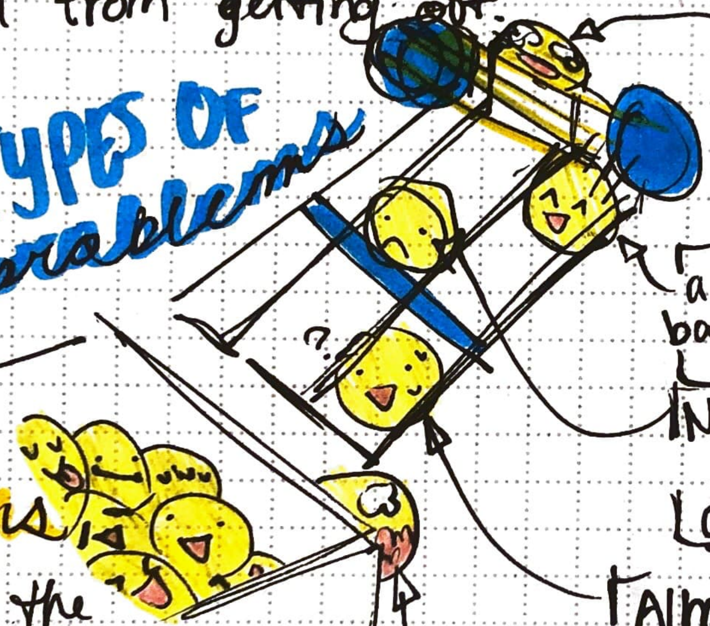


continued

SOLUTIONS
solutions

First we had to find a way to solve the ramp problem, starting with the easiest way. The blue beam was stopping every ball without enough momentum from getting out.

4 TYPES OF problems



1. stuck on roller. you can easily turn the roller, but it might turn into 2, 3, 4, or 5.

2. all the way in the back, but has momentum. can turn into 4 or 5.

3. No momentum, stuck on blue beam. won't go no matter how many shakes

4. Almost there, but the ramp isn't steep enough. You might be able to shake it in

5. Basically already there.. until there is a gap between the cage and the pit. Normally the driver's mistake.

solutions

1. Simply roll the roller, and if it gets stuck shake up & down until it gets in.

VEX IQ

2. We removed the blue beam's pegs, & replaced them 1 peg down so bals could glide through.
4. Mostly just shake, since there is no other way.
5. The result of the drivers hesitation, the ramp not set perfectly, and sometimes shaking. We used pegs to stop the gears from going lower than that point.

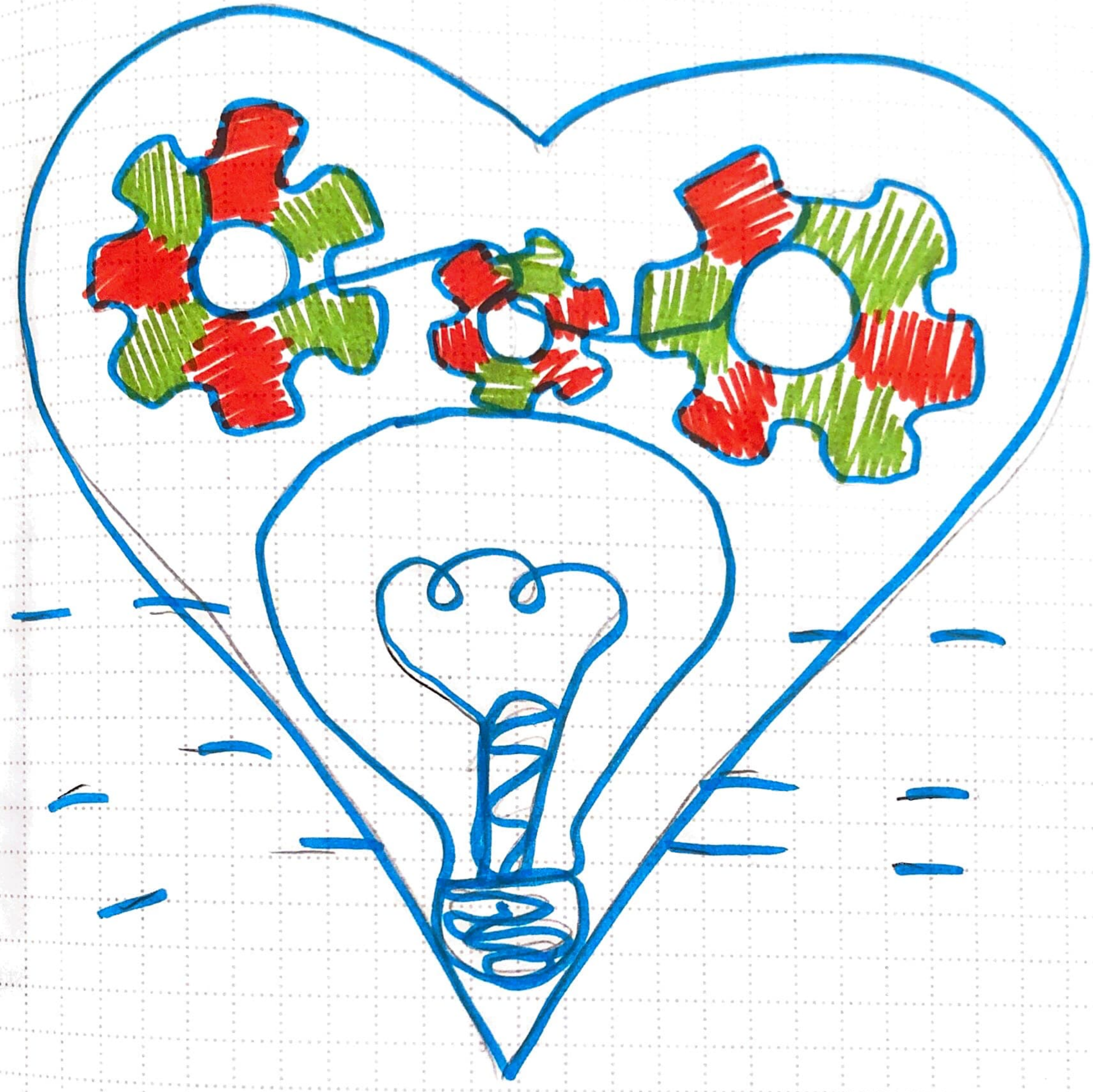
project

designed by: Renee

witnessed by:

date:

VEX Chemical Engineers combine their knowledge of both chemistry and engineering to solve problems like how to produce pharmaceuticals or how to reduce the pollution levels in the environment.



BRAINSTORM

project

designed by: Renee

witnessed by:

date:

Process of Brainstorming

STAGE #1:

What is Our Goal?

STAGE: #2

How Do You MEET your goal?

STAGE #3:

Which strat-egy is The BEST?

STAGE #4:

How To Plan your Robot Building Process?

STAGE #5:

Details needed To Build?

project

designed by:

witnessed by:

date:

STAGE #1:

First, we ask ourselves what our overall goal may be. for our team, it is to hang on the high bar and dump as many balls possible, into the high goal.

STAGE #2:

Next, we create multiple strategies in order to reach our goal. In our case, we have four strategies. A shooting, dumping, catapult, or conveyor belt robot.

STAGE #3:

This step seems pretty simple, although it requires some thinking out of our four strategies, we have to pick one strategy to focus on. As a team, we have decided to focus on a dumping robot because we thought it could help score the most points.

STAGE #4:

Then, we need proof of concept. To us, proof of concept is building just the right amount of the robot. Then, test if it works. If it does, we may proceed to the next step. If it does not, we try to find the core problem and go from there.

(NOTE: Video taping test No.1 is recommended when testing.)

STAGE 5:

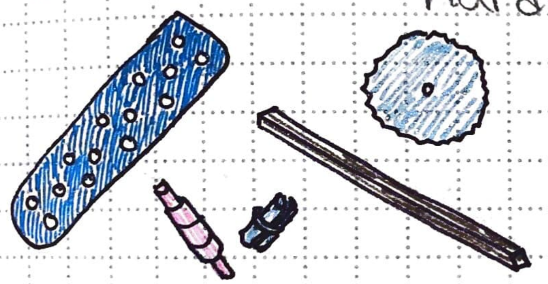
Finally, think and start building the small details and micro-adjustments. For example, if a stopper is missing on the arm or drive-train, place one there. If needed, double check your robot to solve all minor adjustments.

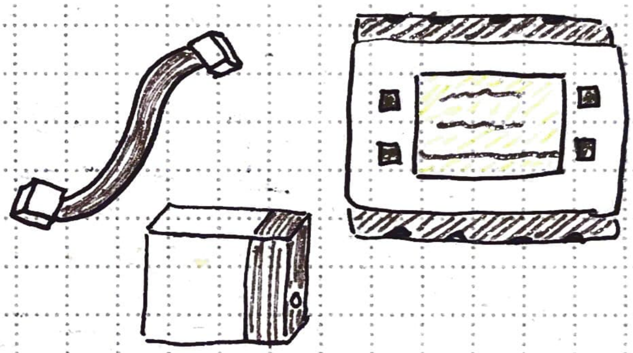
COMPONENTS OF ROBOTS

HARDWARE:

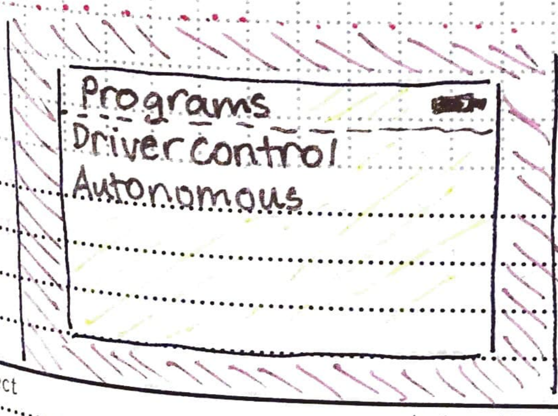
mechanical: you can touch the hardware!

electrical: The electrical part of hardware are the parts that use electricity from the battery. Ex. brain, motor, wire





The mechanical part of hardware are the pieces that don't use electricity. It is what the robot is made up of. Ex. Axle, bars, pegs.



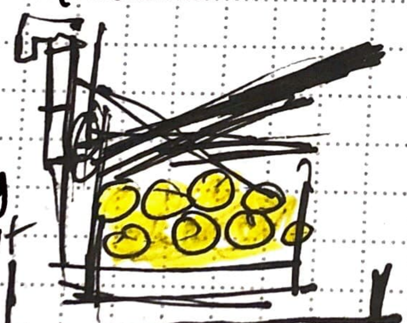
SOFTWARE:

The software of a robot is not touchable. It is like the programs in the brain and the programs you to put into the robot.
Ex. A program I wrote that collects balls.

Measuring

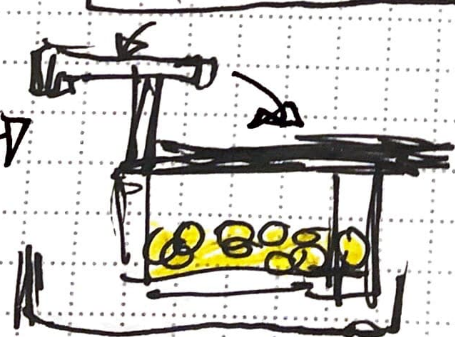
Although we had some solutions to problems, some of them could have exceeded the height limit.
We had to make sure they didn't very precisely, because we just nearly made it in.

The size limit is counted in holes, because we want to be extra precise. It is 38 holes. We woud have exceeded that if it weren't for something we didn't even have to do.



does not exceed

our robot fully sprawled out



exceeds

However, we will never have our robot in that position. We will always either have the ramp up and the claw horizontal, or the ramp horizontal to cross the low bar and the claw flat on top.

project

designed by: Renee

witnessed by:

date:

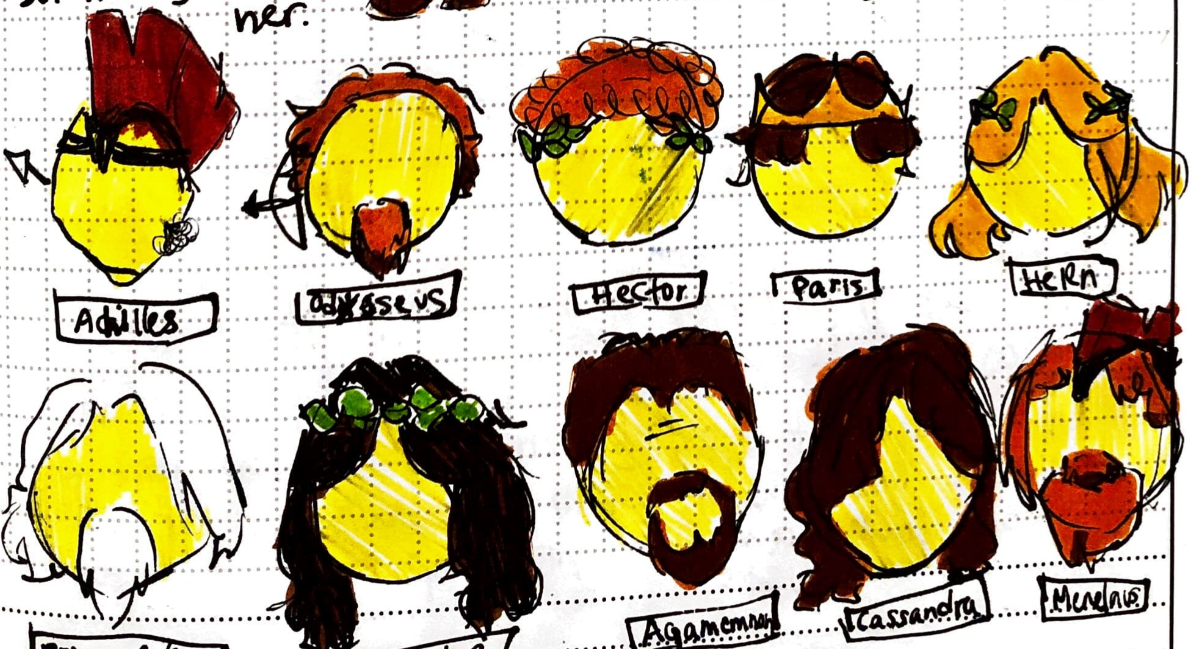
the Trojan Horse

We decided to name our robot the Trojan horse, since it holds many "warriors" inside it, and looks like a horse. It's mouth even latches onto the high beam!

Our balls will be our warriors, and our enemies. We must infiltrate Troy at the dead of night and set fire to all the sleeping trojans [redacted].

fun fact:

Apollo cursed Cassandra to never be able to have anyone believe in her oracles, as god of oracles. She predicted the Trojan horse wasn't a gift, but a trick, but nobody believed her.



THE TROJAN WARRIORS

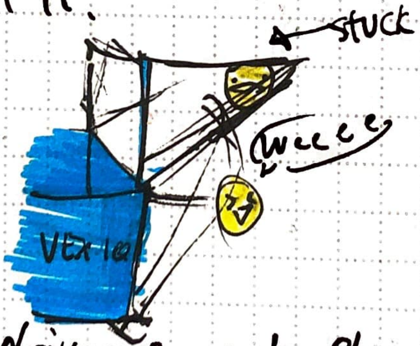
project:
designed by: Rhea
witnessed by:
date:

the DEPOSIT PROBLEM

On the page before the brainstorming title, I wrote down some problems & solutions, but didn't elaborate that much in detail. Plus, we had a couple other problems, too.

PROBLEMS (again)

1: When we flipped the robot over, the robots either go stuck on a space in the ramp, or fell out through it.



2: When the driver goes to flip it to the right position, it is extremely hard to get it perfectly in the right spot. Either it is too high up and stops suddenly, dropping the balls out, or it goes too low and might drop more out if it adjusts.

project

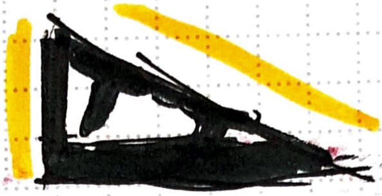
designed by: Renee

witnessed by:

date:

SOLUTIONS solutions (yet again)

1:



we securely blocked off the sides of the ramp with t-pieces, so that the balls had no chance of escaping through.

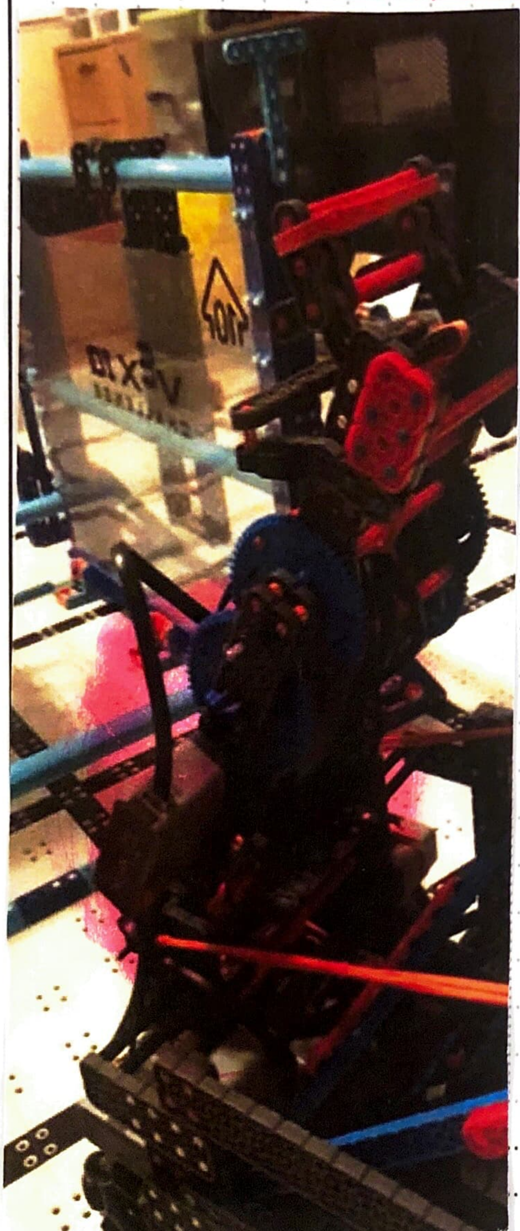
2: This seems like it would have to alter a lot of things, like how far the ramp goes, or making tweaks to the actual ramp. But that might exceed the size limit, and there is an easier option anyways.

STOP THE GEAR!

If we just put the robot in the perfect depositing position, and add a peg where we want the gear to stop, it'll go into that position on its own! And we'll never need to use it again, so we have all the right to do that.

project designed by: Renee witnessed by: date:

Arm Snapper



What is the Arm Snapper?

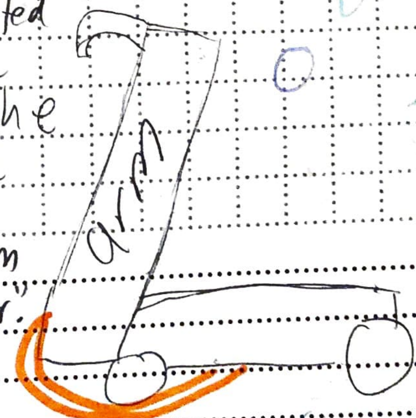
What this mechanism does is it allows the arm to spring up without the help of a motor. Since the motor limit is 6, we had to figure out how to save motors as much as we could. To snap the arm into place, all you have to do is drive back and forth quickly. The momentum would snap the arm in place.

Problem

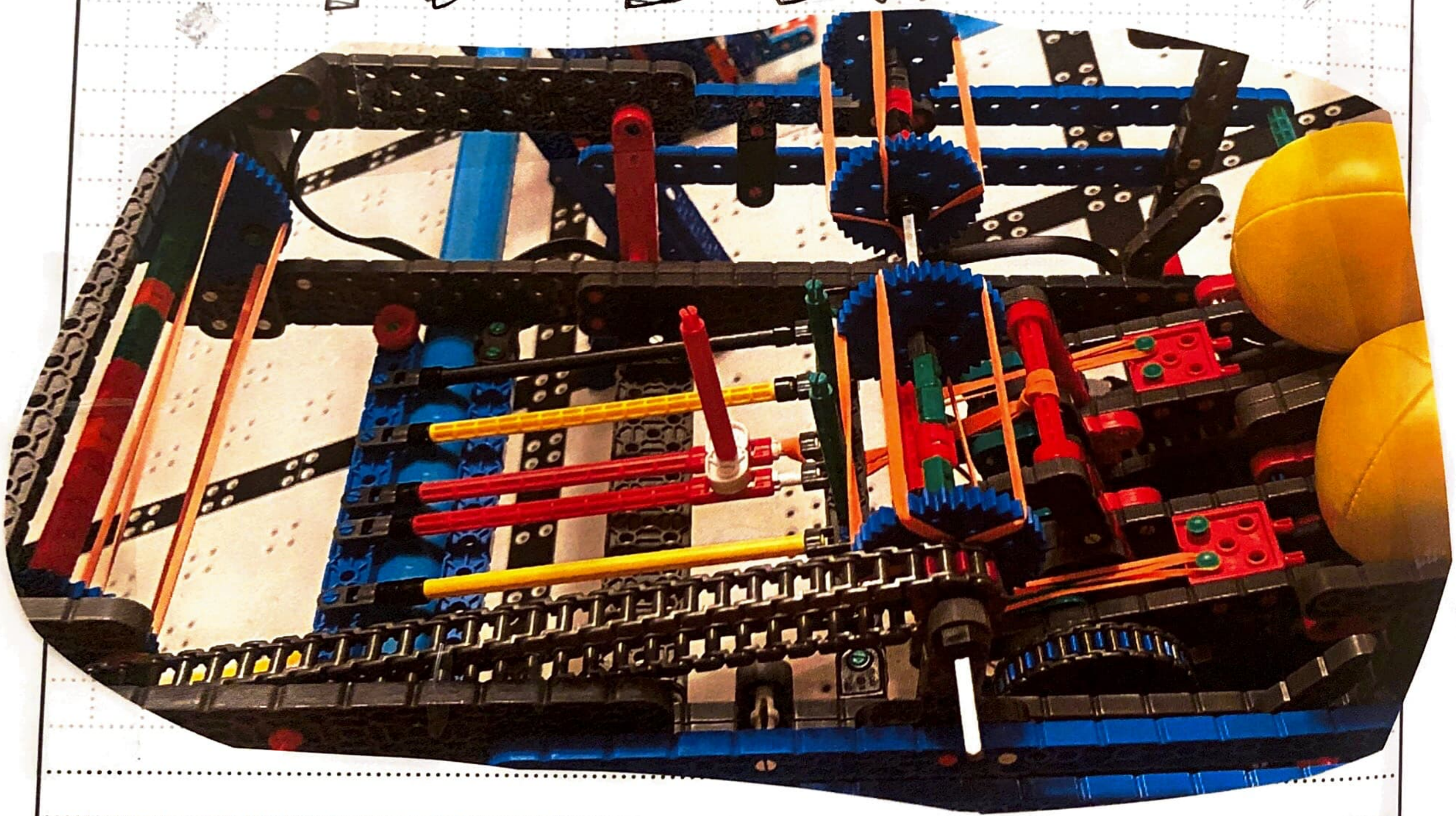
The problem was that there was a 6 motor limit for the robot. Since drive train took 3 motors, we had 3 motors left. 1 would go to the arm, one for the ball picker and 1 for flipping the robot over the high bar. Adding this all together we have 6 motors. Our problem was that we needed a way to put our arm up after it had gone under the low bar.

Solution

Our solution was to use rubber bands. The rubber bands would be connected to the arm and drive train of the robot lifting the arm up. After the rubber bands did their job, the driver would only have to create enough momentum to snap the arm in place, hence the name, "Arm Snapper".



THE Wobble Problem



This problem is where our ball collector and shooter are unstable.

project

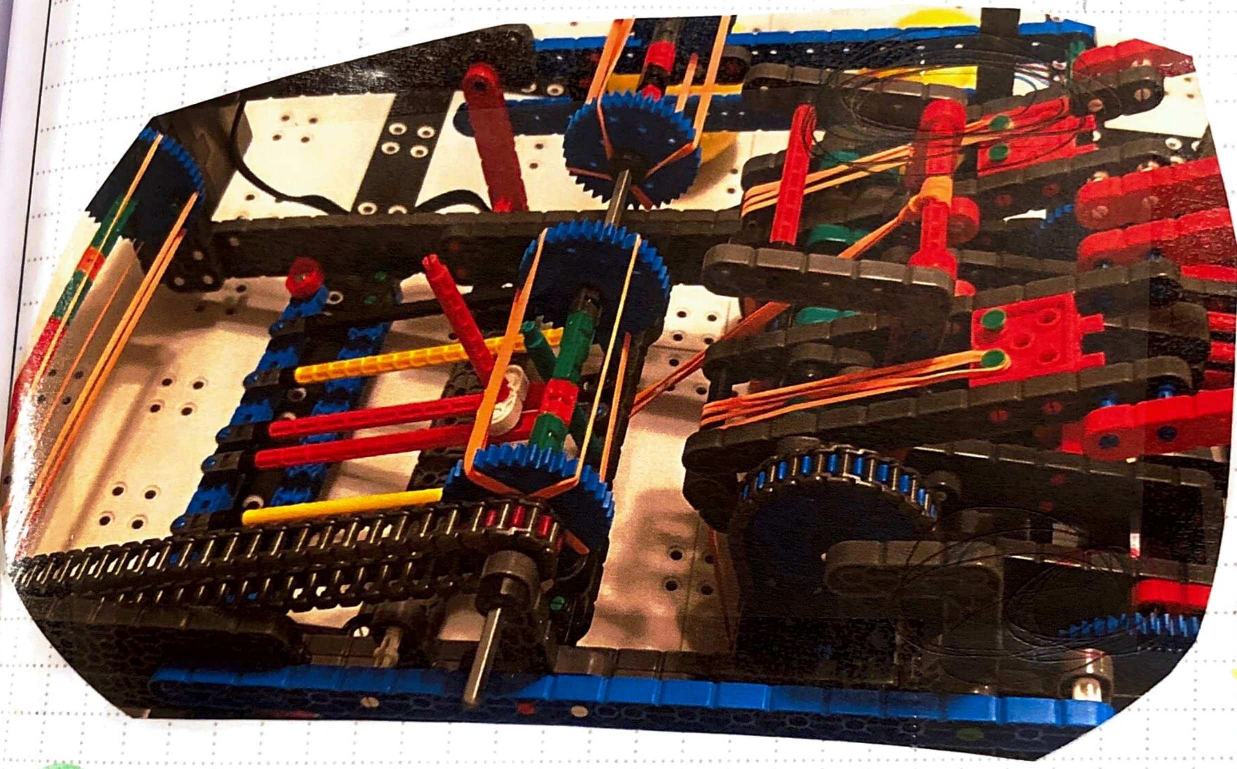
designed by:

witnessed by:

date:

Problem:

The problem is that the ball pickup and deposit mechanism is to unsteady. It wobbles around a lot and could be a major problem.



Solutions:

Some possible solutions would to strengthen the point of contact the ball mechanism has with the drive train. We could add more pegs, standoffs, etc. We could also lighten the mechanism and take some weight off of it. This could also help but I feel as if strengthing the connection would be more beneficial.

project
designed by:
witnessed by:
date:

For this robot meet, our team focused on brainstorming problems and solutions for our robot, Trojan Horse

The paper glued below is a raw copy of the notes I took last week.

NOTES

pro[...] signed by: Zoe witnessed by:

date:

NOTES

1) Robot doesn't drive straight.
2) Claw won't fully grasp high bar
3) Sometimes robot motors doesn't have enough strength to lift and "flip".

1) ~~Design Prob~~

1) Program / Ball gets stuck on wheel
2) Design Problem
3) X
4) increase speed of drivetrain to grasp onto high bar.

pro

ssed by:

date:

Pl

NOTES

5) Balls get stuck between chain ... MEH

Most balls may still go in the basket

6) Two bars supporting back of brain — used as a measurement to line up arm

7) Bu

signed by: zoe

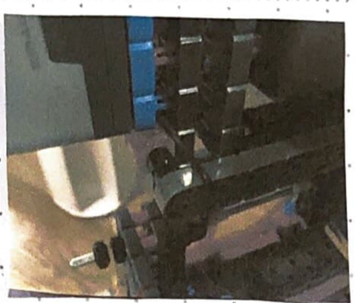
Mindy (for credit so Sammy doesn't
Steal my ideas)

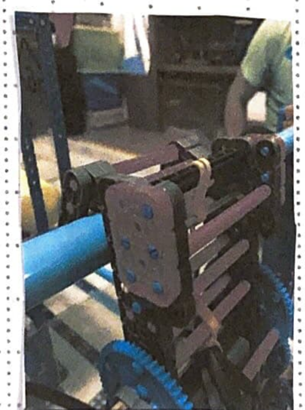
medium Problems:
- the bar with standoffs came off
- had trouble latching onto the bar
- couldn't drive straight.
- took a few tries to get the arm up
- took a while for the robot to go up

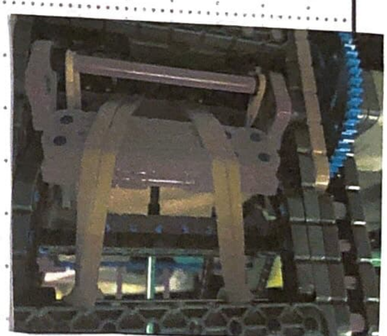
not driver error also programming is impossible
1. max speed is not enough
2. can't go straight affects the activation of claw

medium - ball gets stuck in front

RAW NOTES!



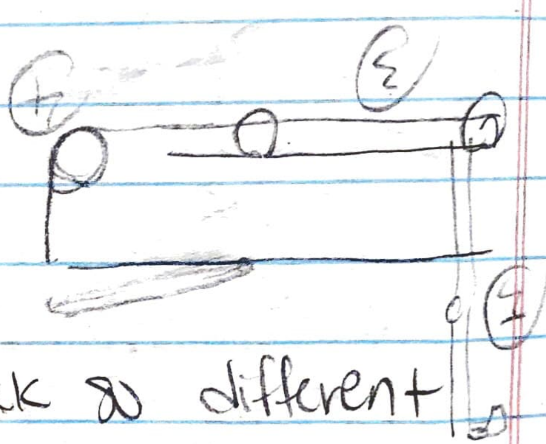




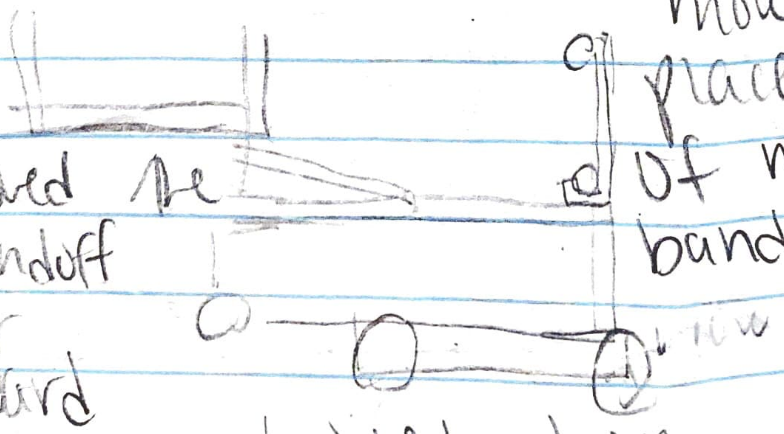
RAW NOTES!

- Back stopper bar comes off
1. can't see how far back the arm goes
2. then it can't latch on

- The arm doesn't lock



move bar back so different angle.



moved placement of rubber band down

moved the standoff bar forward

moved drive-train wheels closer together and guards on wheels

NEW RULE?

UH OH!

We have a new rule added to the game manual! On (8/31) This rule states we are only allowed to have (2) balls when hanging on the high bar!!

Ohhh...NO

Our hard work!

★ Gasp

Oh my...

Do we have to start over?

DUN DUN DUNNN!

How we Feel!

We feel like all our hard work has gone down the drain! Our robot can collect and dump balls, although if we want to dump our goals amount of balls, we are violating this rule!!

project

designed by:

witnessed by:

date:

combined wit

<G19> Ball control is limited while Hanging. *Robots* which are not contacting the *Floor* may control a maximum of two (2) *Balls*.

In the context of this rule, "control" implies that the *Robot* is manipulating the *Balls*, not simply touching them. For example, if a *Ball* moves with the *Robot* either vertically or while turning, then the *Robot* is "controlling" the *Ball*. Other synonyms for "control" could be "hold", "possess", "support", "lift", or "carry".

There is no restriction on the number of *Balls* a *Robot* may control at any other point during the *Match*. However, a *Robot* which is controlling more than two (2) *Balls* must remove any excess *Balls* from their *Robot* before leaving contact with the *Floor*.

Minor, momentary, or incidental violations of this rule that do not affect the Match will result in a warning. Score affecting offenses will result in a *Disqualification*. *Teams* that receive multiple warnings may also receive a *Disqualification* at the *Head Referee's* discretion.

witnessed by:

date:

project

d

GAME CHANGING!

unfortunately, due to the infuriating rule change,,,

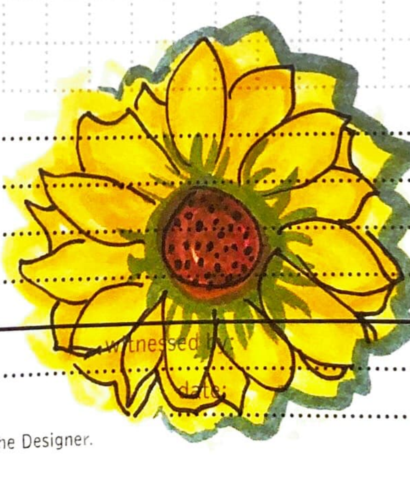
we are forced to start the process over :(sorry Achilles!

However, this isn't THE END OF THE WORLD...

we decided to turn to our old idea back in our brainstorming stage:

THE DOUBLE CATAPULT!

combined with our fish hook reel idea for the High Hang.



Proof of Concept

project

date:

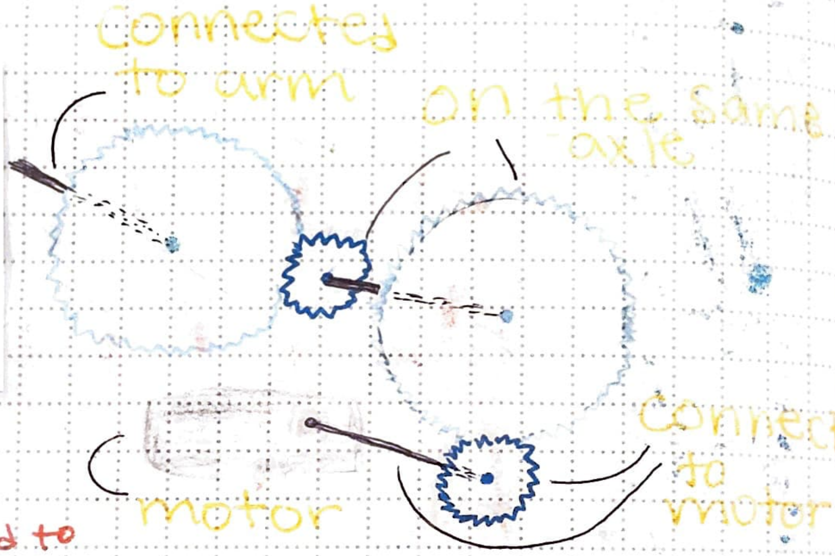
Helena


Power UP!
GEAR RATIO

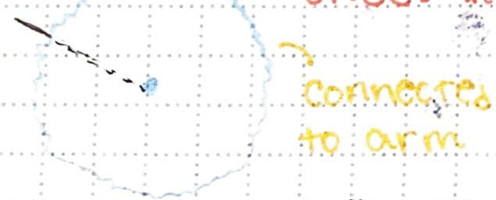
COMPOUND

The ratio for our ~~robot~~ robot is currently 9:1. We are using a compound ratio.

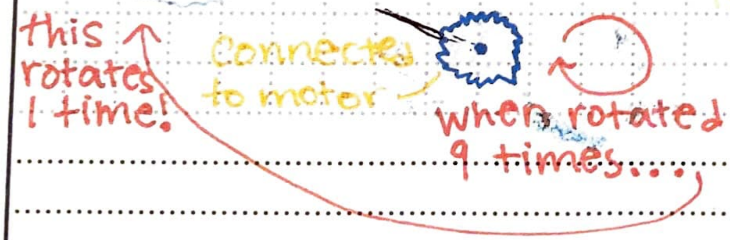
HOW IT WORKS



The motor turns the 12 tooth gear attached to it. When that gear turns three times, the 36 teeth gear turns once. This is because 12 fits in 36 3 times. When the 36 teeth turns once, the 12 teeth gear on it's axle also turns once. This is because they are on the same axle. Like before, when that 12 teeth gear turns 3 times, the gear connected to the arm turns once.



If you multiply them (3/1 • 3/1), you will get $\boxed{9:1}$



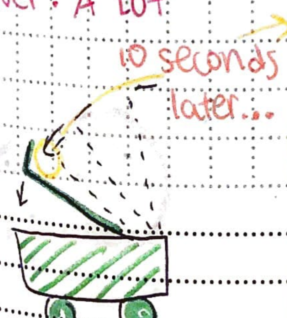
project designed by: Kristen witnessed by:

date:

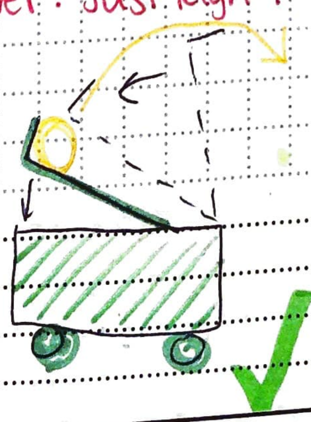
WHAT GEAR RATIO SHOULD THE FLING USE?

We discussed that to make sure the gear ratio will make enough power to flings two balls into the basket, but not take too long to do so. The more power you add to the gear ratio, the slower it will become. So to get it perfect, we need to find the "sweet spot"!

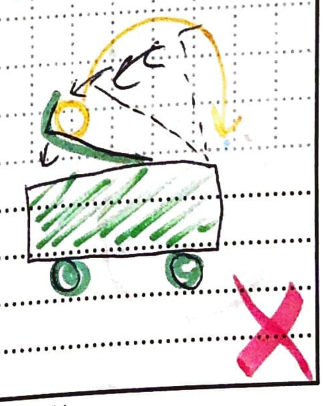
Time: 15 seconds
Power: A Lot



Time: 3 seconds
Power: Just Right!



Time: 1 second
Power: A little



project

designed by: Krita

witnessed by:

date:

What is STEM?

S T E M

Science
Technology
Engineering
Math

We all know what STEM stands for Science, Technology, Engineering, and Math. But, do we really know what they mean? Do we understand the purpose of each? Do we know what their relations between each other really are?

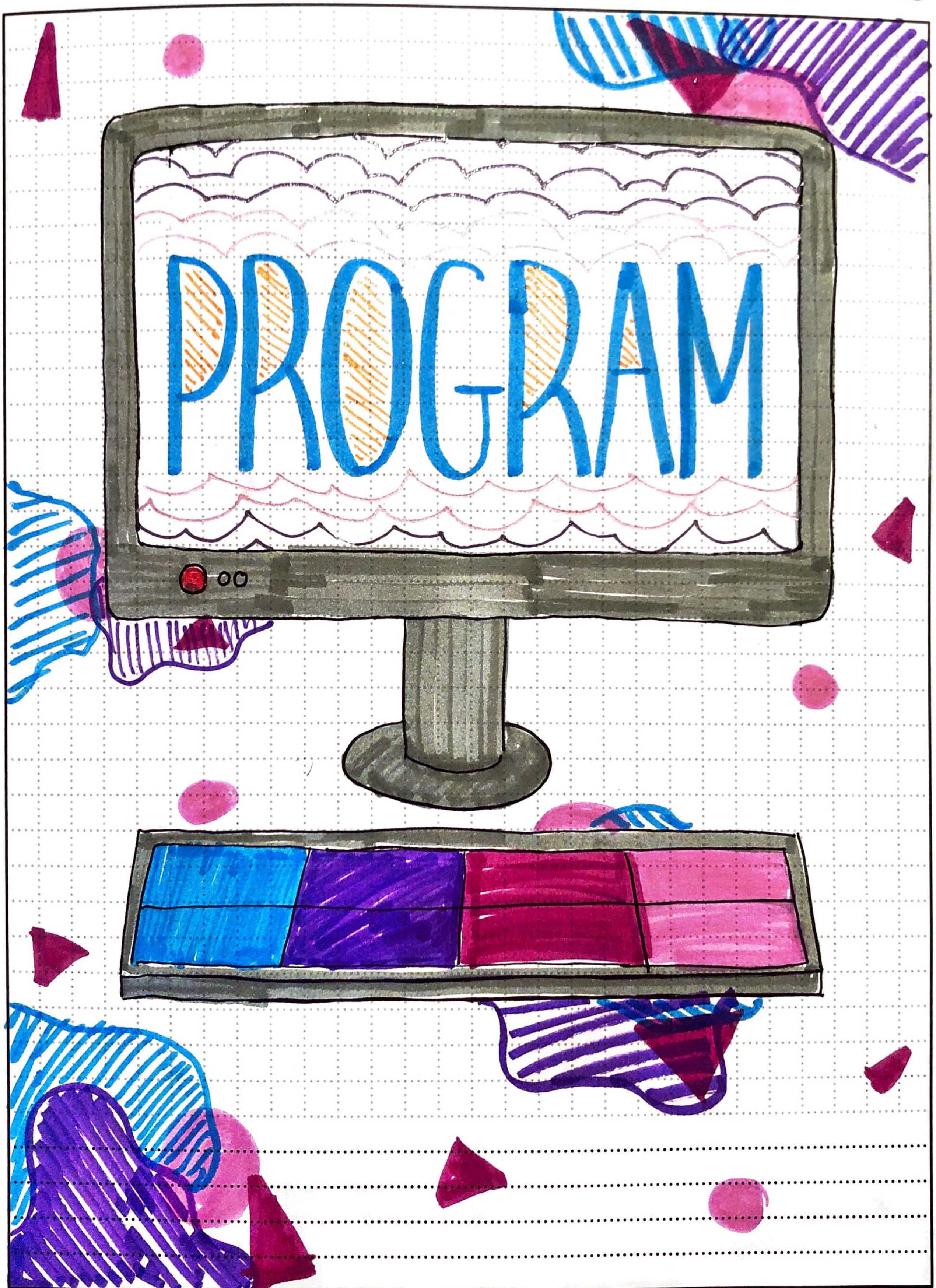
project
designed by:
witnessed by:
date:

Science
- explains why something is a certain way
- laws
- almost also social science

Technology
- applies science into something useful/daily life
- Ex. Computers

Engineering
- putting together the technology to make something that wors
- different kinds

Math
- used in all three (Science, Technology, Engineering)
- used as a tool


PROGRAM

```
#pragma config(Sensor, port3,  BumperL,        sensorVexIQ_Touch)
#pragma config(Sensor, port7,  BumperR,        sensorNone)
#pragma config(Motor,  motor1,          Right,         tmotorVexIQ, PIDControl,
driveRight, encoder)
#pragma config(Motor,  motor4,          Roller,        tmotorVexIQ, PIDControl,
reversed, encoder)
#pragma config(Motor,  motor5,          Shooter,       tmotorVexIQ, PIDControl,
reversed, encoder)
#pragma config(Motor,  motor6,          Left,          tmotorVexIQ, PIDControl,
reversed, driveLeft, encoder)
#pragma config(Motor,  motor10,         BallLift,      tmotorVexIQ, PIDControl,
encoder)
#pragma config(Motor,  motor11,         Shift,         tmotorVexIQ, PIDControl,
encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard
!!*//

//https://www.vexforum.com/t/discussion-on-using-tasks-in-robotc/33025
//https://robocatz.com/functions-motor.htm
//https://github.com/Vex999A/PitchingIn

//shoot function
void shoot() {
        resetMotorEncoder(Shooter);
        setMotorTarget(Shooter, 150, 100);
        waitUntilMotorStop(Shooter);
        while((getBumperValue(BumperL) == 0)&&(getBumperValue(BumperR) == 0))
        {
                setMotorSpeed(Shooter, 100);
                wait1Msec(5);
        }
        setMotorSpeed(Shooter, 0);
        wait1Msec(10);
}

//load function
void load(){
        setServoTarget(BallLift, 370);
        sleep(900);
        setMotorSpeed(Roller, 100);
        setServoTarget(BallLift, 0);
        //setMotorSpeed(Roller, 0);
        wait1Msec(10);
}

/*Task to reset shooter*/
task ShooterTask()
{
        while(true)
        {
```

```
			/*Shooter by Channel R - Up*/
			if(getJoystickValue(BtnRUp) == 1)
			{
				shoot();
			}
		}
	}

	/*Task to load balls to shooter*/
	task LoaderTask()
	{
		while(true)
		{
			/*Ball Loader by Channel R - Down*/
			if(getJoystickValue(BtnRDown) == 1)
			{
				load();
			}
		}
	}

	/*Task to automatically shoot and load balls*/
	task AutoShooterTask()
	{
		while(true)
		{
			/*Activate auto-shooter by Channel F - Up & Down together*/
			if(getJoystickValue(BtnFUp) == 1)
			{
				if(getJoystickValue(BtnFDown) == 1){
					shoot();
					wait1Msec(100);
					load();
				}
			}
		}
	}

	task main()
	{
		/**********************
		*Control Channel Setup *
		**********************
		Channel C/D: JoyStick Drive
		Channel A: Forward/Backward Only - reversed

		Channel L: Shift operation
		Channel RUp: Shooter operation
		Channel RDown: Ball loader operation
		Channel E: Balllift operation
```

```
        Channel F: Roller operation

        Channel FUp + FDown: AutoShooter operation
        */

        /*****************
        *Declare Variables*
        *****************/
        static const int threshold = 10; //set threshold constant for Joystick input
to weed out noise

        static const int forwardThrottle = 1; //set forward throttle constant
        static const int turnThrottleN = 4; //set turn throttle numerator constant
        static const int turnThrottleD = 5; //set turn throttle denominator constant

        static int forwardSpeed; //declare local static variable forwardSpeed
        static int turnSpeed; //declare local static variable turnSpeed

        /***************
        *Initialization*
        ***************/
        setMotorEncoderUnits(encoderDegrees);

        setMotorBrakeMode(Right, motorHold);
        setMotorBrakeMode(Left, motorHold);
        setMotorBrakeMode(BallLift, motorHold);
        setMotorBrakeMode(Shift, motorHold);
        setMotorBrakeMode(Shooter, motorHold);
        setMotorBrakeMode(Roller, motorHold);

        resetMotorEncoder(Right);
        resetMotorEncoder(Left);
        resetMotorEncoder(BallLift);
        resetMotorEncoder(Shift);
        resetMotorEncoder(Shooter);
        resetMotorEncoder(Roller);

        /*Start auxiliary tasks*/
        startTask(ShooterTask);   // start task ShooterTask
        startTask(LoaderTask);   // start task LoaderTask
        startTask(AutoShooterTask);   // start task AutoShooterTask

        /*Reset shooter*/
        shoot();

        /*Start main control infinite loop*/
        while(true)
        {
                /*Drivetrain
                *Right joystick: we use the right joystick (both channel C and D) to
```

```
control both driving and turning,
		*Left joystick: We use the left joystick to do only straight
movements (channel A for straight forward and backward).
		*/
		//displayTextLine(3, "joystick: %d", getJoystickValue(ChD));
		if(getJoystickValue(ChC) > threshold  ||
			getJoystickValue(ChC) < -threshold ||
		getJoystickValue(ChD) > threshold  ||
		getJoystickValue(ChD) < -threshold )
		{
			forwardSpeed = getJoystickValue(ChD)/forwardThrottle; //grab
channel D Joystick value and store in forwardSpeed
			turnSpeed =
(getJoystickValue(ChC)*turnThrottleN)/turnThrottleD; //grab channel C Joystick value
and store in turnSpeed
			setMotorSpeed(Left, forwardSpeed + turnSpeed);
			setMotorSpeed(Right, forwardSpeed - turnSpeed);
			//displayTextLine(3, "forward+turn: %d", forwardSpeed +
turnSpeed);
			//displayTextLine(3, "joystick: %d", getJoystickValue(ChD));
			sleep(10); //pause to allow system to process other tasks
		}
		else if(
			getJoystickValue(ChA) > threshold  ||
		getJoystickValue(ChA) < -threshold )
		{
			forwardSpeed = getJoystickValue(ChA)/forwardThrottle; //grab
channel A Joystick value and store in forwardSpeed
			setMotorSpeed(Left, -forwardSpeed);
			setMotorSpeed(Right, -forwardSpeed);
			sleep(10); //pause to allow system to process other tasks
		}
		else
		{
			setMotorSpeed(Left, 0); //set all drive motors to rest
			setMotorSpeed(Right, 0); //set all drive motors to rest
		}


		/*BallLift by Channel E*/
		if(getJoystickValue(BtnEUp) == 1)
		{
			setMotorSpeed(BallLift, 100);
			sleep(100); //pause to allow system to process other tasks
			setMotorSpeed(BallLift, 0);
		}
		else if(getJoystickValue(BtnEDown) == 1)
		{
			setMotorSpeed(BallLift, -100);
			sleep(100); //pause to allow system to process other tasks
```

```
				setMotorSpeed(BallLift, 0);
			}
			else
			{
				//setMotorSpeed(BallLift, 0);
			}


			/*Shift by Channel L*/
			if(getJoystickValue(BtnLUp) == 1)
			{
				setMotorSpeed(Shift, 100);
				sleep(10); //pause to allow system to process other tasks
			}
			else if(getJoystickValue(BtnLDown) == 1)
			{
				setMotorSpeed(Shift, -100);
				sleep(10); //pause to allow system to process other tasks
			}
			else
			{
				setMotorSpeed(Shift, 0);
			}


			/*Ball Collector
			*Button F_up for continuous rolling of the collector, and F_down for
reversing and stopping the collector.
			*/
			if(getJoystickValue(BtnFUp) == 1)
			{
				setMotorSpeed(Roller, 100);
				sleep(10); //pause to allow system to process other tasks
			}
			else if(getJoystickValue(BtnFDown) == 1)
			{
				setMotorSpeed(Roller, -100);
				sleep(100); //pause to allow system to process other tasks
				setMotorSpeed(Roller, 0);
			}

		}

}
```

```
#pragma config(Sensor, port2,  TouchLED,       sensorVexIQ_LED)
#pragma config(Sensor, port3,  BumperL,        sensorVexIQ_Touch)
#pragma config(Sensor, port7,  BumperR,        sensorNone)
#pragma config(Sensor, port8,  Gyro,           sensorVexIQ_Gyro)
#pragma config(Sensor, port12, Color,          sensorVexIQ_ColorGrayscale)
#pragma config(Motor,  motor1,          Right,         tmotorVexIQ, PIDControl,
driveRight, encoder)
#pragma config(Motor,  motor4,          Roller,        tmotorVexIQ, PIDControl,
reversed, encoder)
#pragma config(Motor,  motor5,          Shooter,       tmotorVexIQ, PIDControl,
reversed, encoder)
#pragma config(Motor,  motor6,          Left,          tmotorVexIQ, PIDControl,
reversed, driveLeft, encoder)
#pragma config(Motor,  motor10,         BallLift,      tmotorVexIQ, PIDControl,
encoder)
#pragma config(Motor,  motor11,         Shift,         tmotorVexIQ, PIDControl,
reversed, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard
!!*//

//https://www.vexforum.com/t/discussion-on-using-tasks-in-robotc/33025
//https://robocatz.com/functions-motor.htm
//https://github.com/Vex999A/PitchingIn

//define static global variables - task triggers
static int shootSwitch = 0;
static int loadSwitch = 0;
static int autoShooterSwitch = 0;

const int threshold = 90; //white ~ 160; black ~ 50 (gray scale)

//shoot function
void shoot() {
        resetMotorEncoder(Shooter);
        setMotorTarget(Shooter, 150, 100);
        waitUntilMotorStop(Shooter);
        while((getBumperValue(BumperL) == 0)&&(getBumperValue(BumperR) == 0))
        {
                setMotorSpeed(Shooter, 100);
                wait1Msec(5);
        }
        setMotorSpeed(Shooter, 0);
        wait1Msec(10);
}

//load function
void load(){
        setServoTarget(BallLift, 370);
        sleep(900);
        setMotorSpeed(Roller, 100);
```

```
        setServoTarget(BallLift, 0);
        //setMotorSpeed(Roller, 0);
        wait1Msec(10);
}


/*Task to reset shooter*/
task ShooterTask()
{
        while(true)
        {
                /*Shooter by variable shootSwitch*/
                if(shootSwitch == 1) //check to see if shootSwitch is turned on
                {
                        shoot(); //shoot
                        shootSwitch = 0; //turn off shooter
                }
        }
}

/*Task to load balls to shooter*/
task LoaderTask()
{
        while(true)
        {
                /*Ball Loader by variable loadSwitch*/
                if(loadSwitch == 1) //check to see if loadSwitch is turned on
                {
                        load(); //load
                        loadSwitch = 0; //turn off loader
                }
        }
}

/*Task for delay load */

task DelayloadTask()
{
        while(true)
        {
                //=2 is delay load
                if(loadSwitch == 2) //check to see if loadSwitch is turned on
                {
                        wait1Msec (1000);
                        load(); //load
                        loadSwitch = 0; //turn off loader
                }
        }
}
```

```
/*Task to automatically shoot and load balls*/
task AutoShooterTask()
{
	while(true)
	{
		/*Activate auto-shooter by variable autoShooterSwitch*/
		if(autoShooterSwitch == 1) //check to see if autoShooterSwitch is
turned on
		{
			shoot(); //shoot
			wait1Msec(100);
			load(); //load
			autoShooterSwitch = 0; //turn off auto shooter
		}
	}
}


/*Task Main*/
task main()
{
	/*initialization steps start*/
	setTouchLEDColor(TouchLED, colorRed); //start initialization steps

	/******************
	*Declare Variables*
	******************/
	//int turning;

	/***************
	*Initialization*
	***************/
	wait1Msec(5000);

	//set motor encoder units
	setMotorEncoderUnits(encoderDegrees);

	//set motor brake mode
	setMotorBrakeMode(BallLift, motorHold);
	setMotorBrakeMode(Shooter, motorHold);

	//reset sensors/encoders
	resetGyro(Gyro); //Reset gyro sensor to 0 reading

	resetMotorEncoder(Right); //Reset Right drive motor to 0 position
	resetMotorEncoder(Left); //Reset Left drive motor to 0 position
	resetMotorEncoder(BallLift); //Reset BallLift motor to 0 position
	resetMotorEncoder(Shift); //Reset Shift drive motor to 0 position
	resetMotorEncoder(Shooter); //Reset Shooter drive motor to 0 position
	resetMotorEncoder(Roller); //Reset Roller drive motor to 0 position
```

```
	//Calibrate gyro sensor
	startGyroCalibration(Gyro, gyroCalibrateSamples1024); // longer cal, works
better
	wait1Msec(100);
	// wait for calibration to finish
	while(getGyroCalibrationFlag(Gyro))
	{
		displayTextLine(1, "Calibrating... %02d");
		wait1Msec(100);
	}

	setTouchLEDColor(TouchLED, colorGreen); //Gyro sensor calibration completed
	//playSound(soundTada);
	eraseDisplay();
	resetGyro(Gyro);
	/*initialization steps complete*/


	/*autonomous program starts here*/
	/*Start auxiliary tasks*/
	startTask(ShooterTask); //start shooter task
	startTask(DelayloadTask); //start delay load task
	startTask(LoaderTask); //start loader task
	startTask(AutoShooterTask); //start auto shooter task

	while(true)
	{
		/*Touch LED sensor to start the autonuous program*/
		waitUntil(getTouchLEDValue(TouchLED) == 1);

		/*Reset shooter*/
		shootSwitch = 1;

		//start roller
		setMotorSpeed(Roller, 100);

		//go straight out
		resetMotorEncoder(Left);
		resetMotorEncoder(Right);
		setMotorTarget(Left, 250, 40);
		setMotorTarget(Right, 250, 40);
		waitUntilMotorStop(Left);
		waitUntilMotorStop(Right);
		wait1Msec(50);

		//turn 90 degree left to point to the West
		while(getGyroDegrees(Gyro) < 68)
		{
```

```
        setMotorSpeed(Left, -40);
        setMotorSpeed(Right, 40);
}
setMotorSpeed(Left, 0);
setMotorSpeed(Right, 0);
wait1Msec(50);

//shift right to capture balls against fence
resetMotorEncoder(Shift);
setMotorTarget(Shift, 270, 100);
waitUntilMotorStop(Shift);
wait1Msec(50);

resetMotorEncoder(Left);
resetMotorEncoder(Right);
setMotorTarget(Left, 720, 50);
setMotorTarget(Right, 720, 50);
waitUntilMotorStop(Left);
waitUntilMotorStop(Right);
wait1Msec(50);

//shift left
resetMotorEncoder(Shift);
setMotorTarget(Shift, -670, 100);
waitUntilMotorStop(Shift);
wait1Msec(50);

//adjust direction to point to the West
while(getGyroDegrees(Gyro) < 80)
{
        setMotorSpeed(Left, -50);
        setMotorSpeed(Right, 50);
}
setMotorSpeed(Left, 0);
setMotorSpeed(Right, 0);
wait1Msec(50);

//go straight and capture the ball under low bar
resetMotorEncoder(Left);
resetMotorEncoder(Right);
//setMotorTarget(Left, 300, 90);
//setMotorTarget(Right, 300, 100);
//waitUntilMotorStop(Left);
//waitUntilMotorStop(Right);
while((getMotorEncoder(Left)+getMotorEncoder(Right))/2 < 300) {
        setMotorSpeed(Left, 90);
        setMotorSpeed(Right, 100);
}
setMotorSpeed(Left, 0);
setMotorSpeed(Right, 0);
```

```
wait1Msec(50);

clearTimer(T4);
while ( getTimerValue(T4) < 200) {
        setMotorSpeed(Shift, -100);
}
setMotorSpeed(Shift, 0);
clearTimer(T4);
wait1Msec(50);
//shift left against the wall
//resetMotorEncoder(Shift);
//setMotorTarget(Shift, -250, 100);
//waitUntilMotorStop(Shift);
//wait1Msec(50);


//go straight to the corral
resetMotorEncoder(Left);
resetMotorEncoder(Right);
setMotorTarget(Left, 500, 90);
setMotorTarget(Right, 500, 100);
waitUntilMotorStop(Left);
waitUntilMotorStop(Right);
wait1Msec(50);

//load ball by triggering load task
loadSwitch = 1; //load once

//shift right
resetMotorEncoder(Shift);
setMotorTarget(Shift, 350, 100);
waitUntilMotorStop(Shift);
wait1Msec(50);

//go straignt to capture 1st ball in the corral
clearTimer(T4);
while ( getTimerValue(T4) < 1000) {
        setMotorSpeed(Left, 100);
        setMotorSpeed(Right, 100);
}
setMotorSpeed(Left, 0);
setMotorSpeed(Right, 0);
clearTimer(T4);

//back up
resetMotorEncoder(Left);
resetMotorEncoder(Right);
setMotorTarget(Left, -100, 80);
setMotorTarget(Right, -100, 80);
waitUntilMotorStop(Left);
```

```
waitUntilMotorStop(Right);
wait1Msec(50);

//turn 90 degree right to point to the North
//displayTextLine(3, "Gyro is: %d", getGyroDegrees(Gyro));
while(getGyroDegrees(Gyro) > 20)
{
        setMotorSpeed(Left, (50));
        setMotorSpeed(Right, -(50));
}
setMotorSpeed(Left, 0);
setMotorSpeed(Right, 0);
wait1Msec(500);

//load ball by triggering load task
//loadSwitch = 1; //load again

//back up
resetMotorEncoder(Left);
resetMotorEncoder(Right);
setMotorTarget(Left, -250, 100);
setMotorTarget(Right, -250, 100);
waitUntilMotorStop(Left);
waitUntilMotorStop(Right);
wait1Msec(50);

//shift left against the wall
resetMotorEncoder(Shift);
setMotorTarget(Shift, -500, 100);
waitUntilMotorStop(Shift);
wait1Msec(50);

loadSwitch = 2;

//go straight and capture the remaining 4 balls in the corral
clearTimer(T4);
while ( getTimerValue(T4) < 4000) {
        setMotorSpeed(Left, 97);
        setMotorSpeed(Right, 100);
}
setMotorSpeed(Left, 0);
setMotorSpeed(Right, 0);
clearTimer(T4);
wait1Msec(50);
//resetMotorEncoder(Left);
//resetMotorEncoder(Right);
//while((getMotorEncoder(Left)+getMotorEncoder(Right))/2 < 1600) {
//      setMotorSpeed(Left, 90);
//      setMotorSpeed(Right, 100);
//}
```

```
		//setMotorSpeed(Left, 0);
		//setMotorSpeed(Right, 0);
		//wait1Msec(50);

		/*//load ball by triggering load task
		loadSwitch = 1; //load again */

		//shift right into shooting position
		resetMotorEncoder(Shift);
		setMotorTarget(Shift, 1800, 100);
		waitUntilMotorStop(Shift);
		wait1Msec(50);

		while(getColorValue(Color) > threshold) // While the Color sensor in 
'grayscale' mode reads a value less than threshold
		{
			setMotorSpeed(Shift, 100);
		}
		wait1Msec(15); //wait 15 ms to go a little further
		setMotorSpeed(Shift, 0);
		wait1Msec(50);

		//align self against the wall
		clearTimer(T4);
		while ( getTimerValue(T4) < 500) {
			setMotorSpeed(Left, 100);
			setMotorSpeed(Right, 100);
		}
		setMotorSpeed(Left, 0);
		setMotorSpeed(Right, 0);
		resetGyro(Gyro); //reset Gyro now when robot is against the wall
		wait1Msec(500);
		clearTimer(T4);
		wait1Msec(50);

		//back up to get into shooting position
		clearTimer(T4);
		while ( getTimerValue(T4) < 400) {
			setMotorSpeed(Left, -100);
			setMotorSpeed(Right, -100);
		}
		setMotorSpeed(Left, 0);
		setMotorSpeed(Right, 0);
		clearTimer(T4);
		wait1Msec(50);

		//shoot and load 1
		shoot();
		wait1Msec(200);
		load();
```

```
wait1Msec(200);

//shoot and load 2
shoot();
wait1Msec(200);
load();
wait1Msec(200);

//shoot and load 3
shoot();
wait1Msec(200);
load();
wait1Msec(200);

//shoot 4
shootSwitch = 1;
wait1Msec(500);




/**
2nd leg starts here
**/



//turn 90 degree right
//resetGyro(Gyro);
while(getGyroDegrees(Gyro) > -72) //was 70
{
        setMotorSpeed(Left, 50);
        setMotorSpeed(Right, -50);
}
setMotorSpeed(Left, 0);
setMotorSpeed(Right, 0);
wait1Msec(50);

//shift left
resetMotorEncoder(Shift);
setMotorTarget(Shift, -450, 100);
waitUntilMotorStop(Shift);
wait1Msec(50);

//go straight and capture the ball under low bar
resetMotorEncoder(Left);
resetMotorEncoder(Right);
//setMotorTarget(Left, 300, 90);
//setMotorTarget(Right, 300, 100);
//waitUntilMotorStop(Left);
```

```
//waitUntilMotorStop(Right);
while((getMotorEncoder(Left)+getMotorEncoder(Right))/2 < 300) {
        setMotorSpeed(Left, 90);
        setMotorSpeed(Right, 100);
}
setMotorSpeed(Left, 0);
setMotorSpeed(Right, 0);
wait1Msec(50);

//shift left
resetMotorEncoder(Shift);
setMotorTarget(Shift, -100, 100);
waitUntilMotorStop(Shift);
wait1Msec(50);


//go straight
resetMotorEncoder(Left);
resetMotorEncoder(Right);
setMotorTarget(Left, 550, 90);
setMotorTarget(Right, 550, 100);
waitUntilMotorStop(Left);
waitUntilMotorStop(Right);
wait1Msec(50);


//shift right
resetMotorEncoder(Shift);
setMotorTarget(Shift, 350, 100);
waitUntilMotorStop(Shift);
wait1Msec(50);

//go straignt to capture 1st ball in the corral
clearTimer(T4);
while ( getTimerValue(T4) < 1000) {
        setMotorSpeed(Left, 100);
        setMotorSpeed(Right, 100);
}
setMotorSpeed(Left, 0);
setMotorSpeed(Right, 0);
clearTimer(T4);
wait1Msec(50);

//back up
resetMotorEncoder(Left);
resetMotorEncoder(Right);
setMotorTarget(Left, -100, 80);
setMotorTarget(Right, -100, 80);
waitUntilMotorStop(Left);
waitUntilMotorStop(Right);
```

```
wait1Msec(50);

//turn 90 degree right
//displayTextLine(3, "Gyro is: %d", getGyroDegrees(Gyro));
while(getGyroDegrees(Gyro) > -160)
{
        setMotorSpeed(Left, (50));
        setMotorSpeed(Right, -(50));
}
setMotorSpeed(Left, 0);
setMotorSpeed(Right, 0);
wait1Msec(50);

//back up
resetMotorEncoder(Left);
resetMotorEncoder(Right);
setMotorTarget(Left, -250, 100);
setMotorTarget(Right, -250, 100);
waitUntilMotorStop(Left);
waitUntilMotorStop(Right);
wait1Msec(50);

//shift left against the wall
resetMotorEncoder(Shift);
setMotorTarget(Shift, -500, 100);
waitUntilMotorStop(Shift);
wait1Msec(50);

//go straight and capture the remaining 4 balls in the corral
clearTimer(T4);
while ( getTimerValue(T4) < 4000) {
        setMotorSpeed(Left, 97);
        setMotorSpeed(Right, 100);
}
setMotorSpeed(Left, 0);
setMotorSpeed(Right, 0);
clearTimer(T4);
wait1Msec(50);

//load ball by triggering load task
loadSwitch = 1; //load once


//shift right into shooting position
resetMotorEncoder(Shift);
setMotorTarget(Shift, 1800, 100);
waitUntilMotorStop(Shift);
wait1Msec(50);

while(getColorValue(Color) > threshold) // While the Color sensor in
```

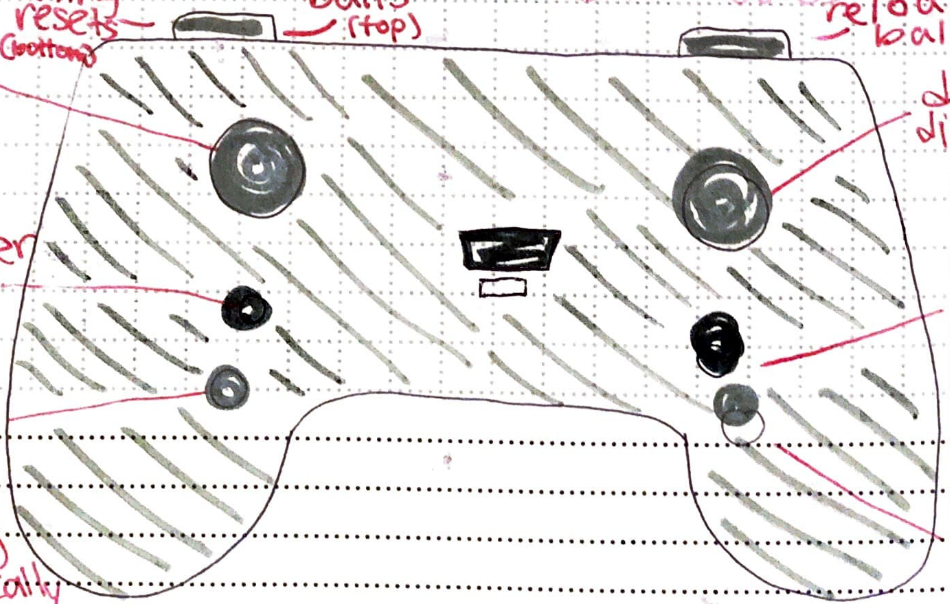
```
'grayscale' mode reads a value less than threshold
                {
                        setMotorSpeed(Shift, 100);
                }
                wait1Msec(15); //wait 15 ms to go a little further
                setMotorSpeed(Shift, 0);
                wait1Msec(50);

                loadSwitch = 1; //load again

                //align self against the wall
                clearTimer(T4);
                while ( getTimerValue(T4) < 500) {
                        setMotorSpeed(Left, 100);
                        setMotorSpeed(Right, 100);
                }
                setMotorSpeed(Left, 0);
                setMotorSpeed(Right, 0);
                resetGyro(Gyro);
                clearTimer(T4);
                wait1Msec(200);

                //back up to get into shooting position
                clearTimer(T4);
                while ( getTimerValue(T4) < 450) {
                        setMotorSpeed(Left, -100);
                        setMotorSpeed(Right, -100);
                }
                setMotorSpeed(Left, 0);
                setMotorSpeed(Right, 0);
                clearTimer(T4);
                wait1Msec(50);


                //shoot and load 1
                shoot();
                wait1Msec(200);
                load();
                wait1Msec(200);

                //shoot and load 2
                shoot();
                wait1Msec(200);
                load();
                wait1Msec(200);

                //shoot 3
                shoot();

                setMotorSpeed(Roller, 0); //stop roller
```

```
		//turn 90 degree left
		resetGyro(Gyro);
		while(getGyroDegrees(Gyro) < 67)
		{
			setMotorSpeed(Left, -50);
			setMotorSpeed(Right, 50);
		}
		setMotorSpeed(Left, 0);
		setMotorSpeed(Right, 0);
		wait1Msec(50);

		//shift right
		resetMotorEncoder(Shift);
		setMotorTarget(Shift, 150, 100);
		waitUntilMotorStop(Shift);
		wait1Msec(50);

		//back up to hang on the low bar
		//resetMotorEncoder(Left);
		//resetMotorEncoder(Right);
		//setMotorTarget(Left, -600, 100);
		//setMotorTarget(Right, -600, 100);
		//waitUntilMotorStop(Left);
		//waitUntilMotorStop(Right);
		//wait1Msec(50);

		clearTimer(T4);
		while ( getTimerValue(T4) < 1800) {
			setMotorSpeed(Left, -100);
			setMotorSpeed(Right, -100);
		}
		setMotorSpeed(Left, 0);
		setMotorSpeed(Right, 0);
		clearTimer(T4);
		wait1Msec(50);



		/*autonomous program ends here*/
	}
}
```

Custom driver control functions

Custom driver control functions let you use programming in driver control, making it easier for the driver. With the click of a button, the robot will complete a comand on its own! So far, we have added 4 of these functions to our program.

1. Since we will need to have the rubber band roller collecting balls at all times, we set it so that it will continuosly roll with a click of a button. It will stop when we press the roll backward button. This function will save time from needing to hold down the button.
2. This function resets the shooter. Since there is a bumper sensor, the shooter will reset all the way down. It is more accurate if the program competes this task because of the bumper sensor. (It needs to ~~released~~ reset for more balls to load.)



project

designed by:

witnessed by:

date:

3. This function automatically reloads balls. It takes four balls (two at a time), uses the loader, and puts them into the flying and the place near the flying. This function saves us time because we won't have to stop the robot to reload, we can just press a button on the controller, let it do its thing, and countinue with the rest of the run.

4. This functions shoots all ball for us (automatically). Once we get into shooting position, we press the button. Since there are already balls on the shooter, it will shoot. Then it will load. The robot will keep shooting and loading until there are no more balls.

These functions greatly help us during driver control!

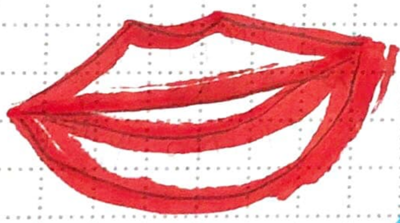
Discussion continued

METHOD.

pen- alternating between capital

pencil- alternating between capital

proof of concept: model first before deciding what you want to do.



K.I.S.S.

Keep
It
Simple
Stupid

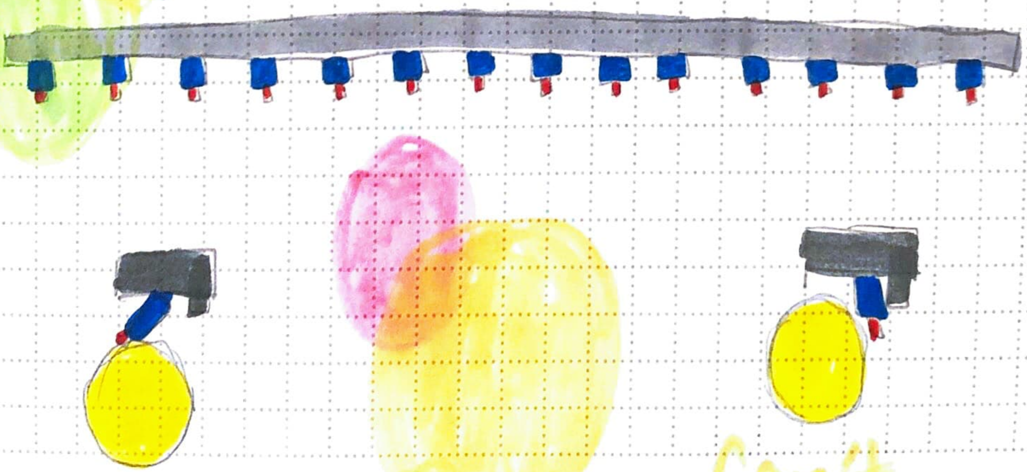
We want to have solutions that are easy and require as little work as possible. We also want it to weigh as less as possible.

project designed by: witnessed by:

date:

Concept - balls can go in but they can't come out

TEETH



Building our robot

- When trying to replicate our robot, we decided that we should build it part by part. Here is our list of how we should build: →

DRIVE TRAIN

- drivetrain is like the base of the robot, we should build the base first
- The drivetrain gear ratio is 3:5 (more strong)

#1

#1

SHOOTER

- Most attached to the base → should be built 2nd

#2

#2

LOADER

- most attached/in contact with the shooter

ROLLER

- not in contacte with any other part

#4

#4

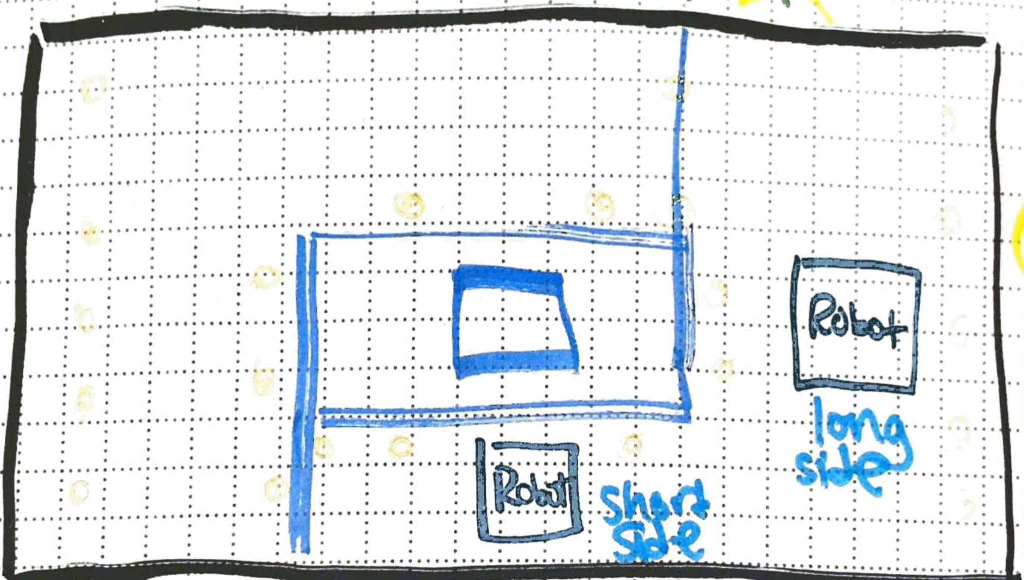
HANGER

with any part

- Smallest

- Doing this order will be easier than trying to build piece by piece
- These parts are the main SUBSYSTEMS of our robot

WHICH SIDE?



There are multiple angles we can shoot balls from, included the long side and/or the short side.

If we shoot from the short side, we only need to back up to the wall (aligning with the middle. But if we want to shoot from the long side we will need to drive forward after backing up

project

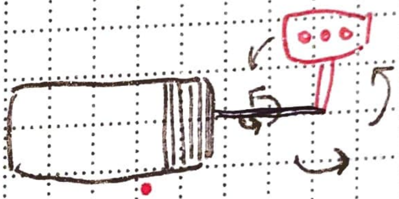
designed by: Kuita

witnessed by:

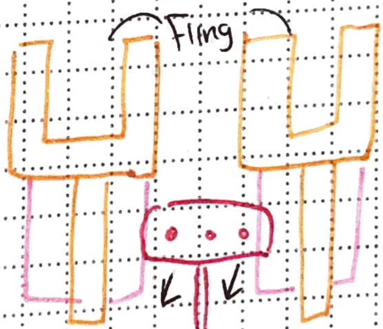
date:

HOW DOES the ball launcher ACTIVATE

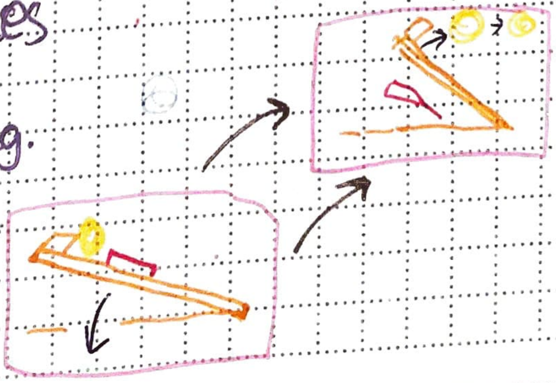
A motor turns an axle, which turns a small piece.



The small piece presses onto the flings.



The piece continues rotating, then releases the fling. This causes the balls to FLY!



project

designed by: Kristen

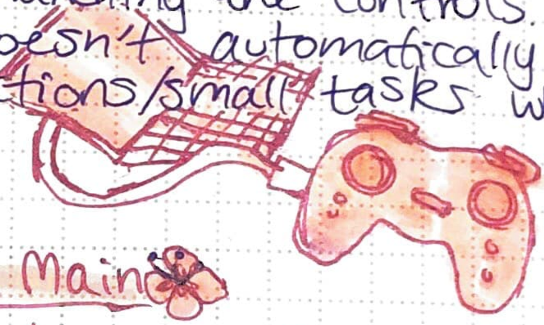
witnessed by:

date:

autonomous VS. driver Control

What are the differences?

Well, to put it simply, you don't have to use a controller in Autonomous. A code will start and run with no more than a touch of a button from the programmer. Driver Control, however, requires a driver to still be present and handling the controls. The code in Driver Control doesn't automatically run, it only runs different functions/small tasks when you press the buttons.



The Task Main

Different programming languages have different "Task Mains", the core of the program that tells the robot what to do. Things outside of the Task Main are things like defining functions, resetting motors, coding LED, bumper, or any type of sensor, and more. The Task Main is where the functions are implemented, or tasks are ordered directly.

Autonomous.

No is most likely

NO		YES	MAYBE
☐	Task Main!	☑	☐
☐	Functions	☐	☑
☑	if, else*	☐	☑
☐	while true:	☐	☑
☐	sleep (blank #)	☑	☐

Driver Control

Yes most likely

No		Yes	Maybe
☐	Task Main!	☑	☐
☐	Functions	☑	☑
☐	If, else*	☑	☐
☐	while true	☑	☐
☐	Sleep (#)	☑	☐

project *Can also be if, then designed by:

witnessed by:

date:

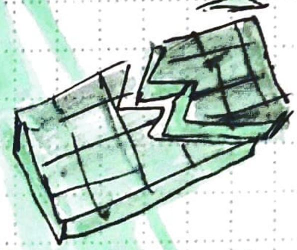
breaking records

So far, the record for skills and driving are 94 and 96.

You would need 14 balls in the bucket, and the driver has 40 secs to collect balls + get into the position, and the autoshooter program will do the rest, then low hang.

OUR RECORD

as of 12/10/21
held by Samantha Li



40-48 ish seconds for half the board

optimal time:

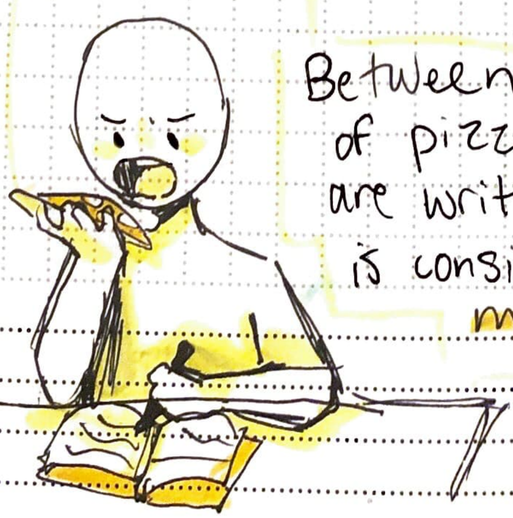
30 secs for half the board AKA 1 Min

→ MULTITASKING

¿is it truly "multi" tasking?

NO!

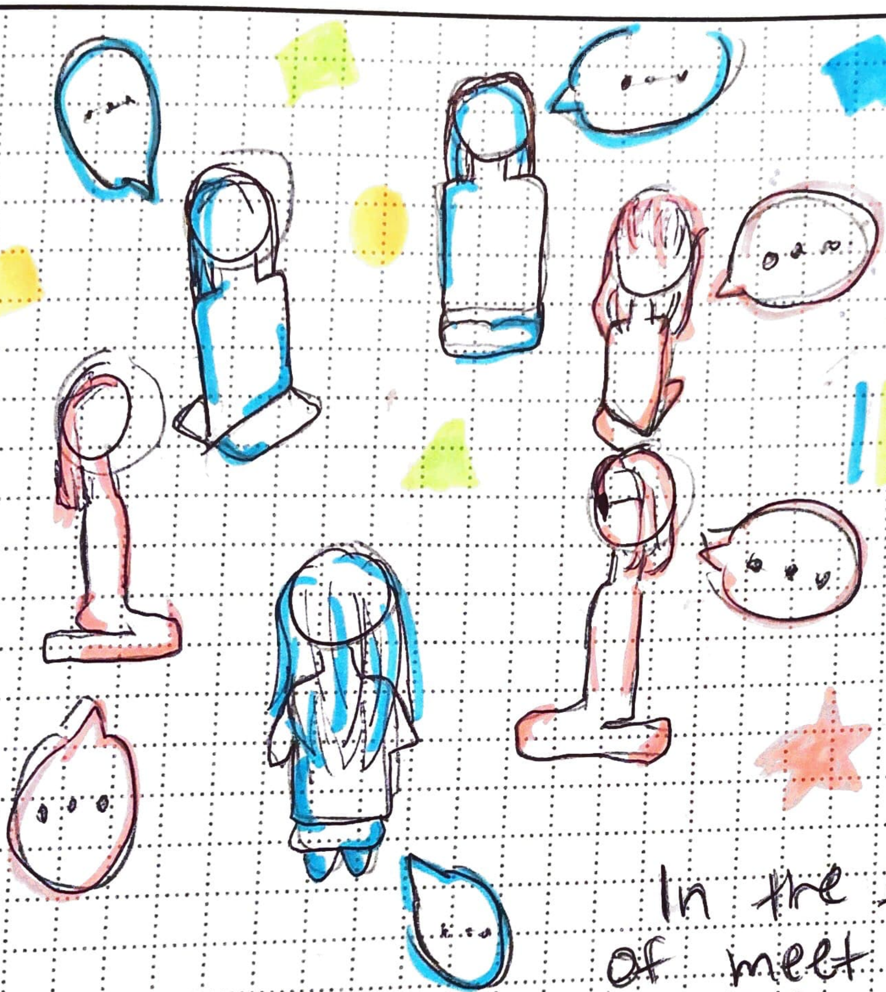
Multitasking is really just doing two tasks seperately, alternating between the two so subconciously and fast, it seems like you are doing the two at the SAME TIME!



Between bites of pizza, you are writing. This is considered multitasking.

Brainstorm

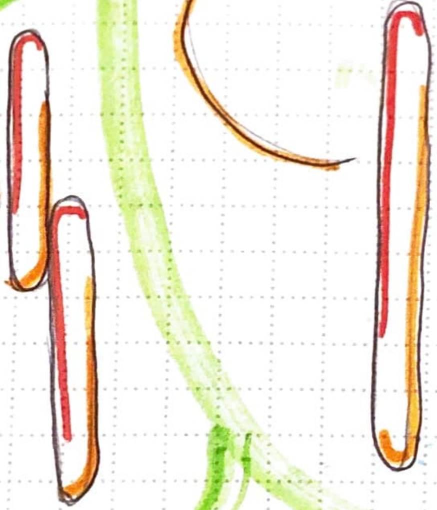
SHARING IDEAS



In the first part of meet this week we discussed some problems, for example, getting the balls into the shooter and shooting accuratelly. we spent some time thinking about it and then ended up sharing our ideas and what we did to solve these problems.

PROBLEM WITH ROBOT?

COMPARE AND CONTRAST



wobbly
less accurate
more space
more pressure

More sturdy
more accurate
less space
no pressure

OLD
- gear ratio 1:3
- motor had to hold position at hardest point
- motor over-heats easily
- lots of pressure

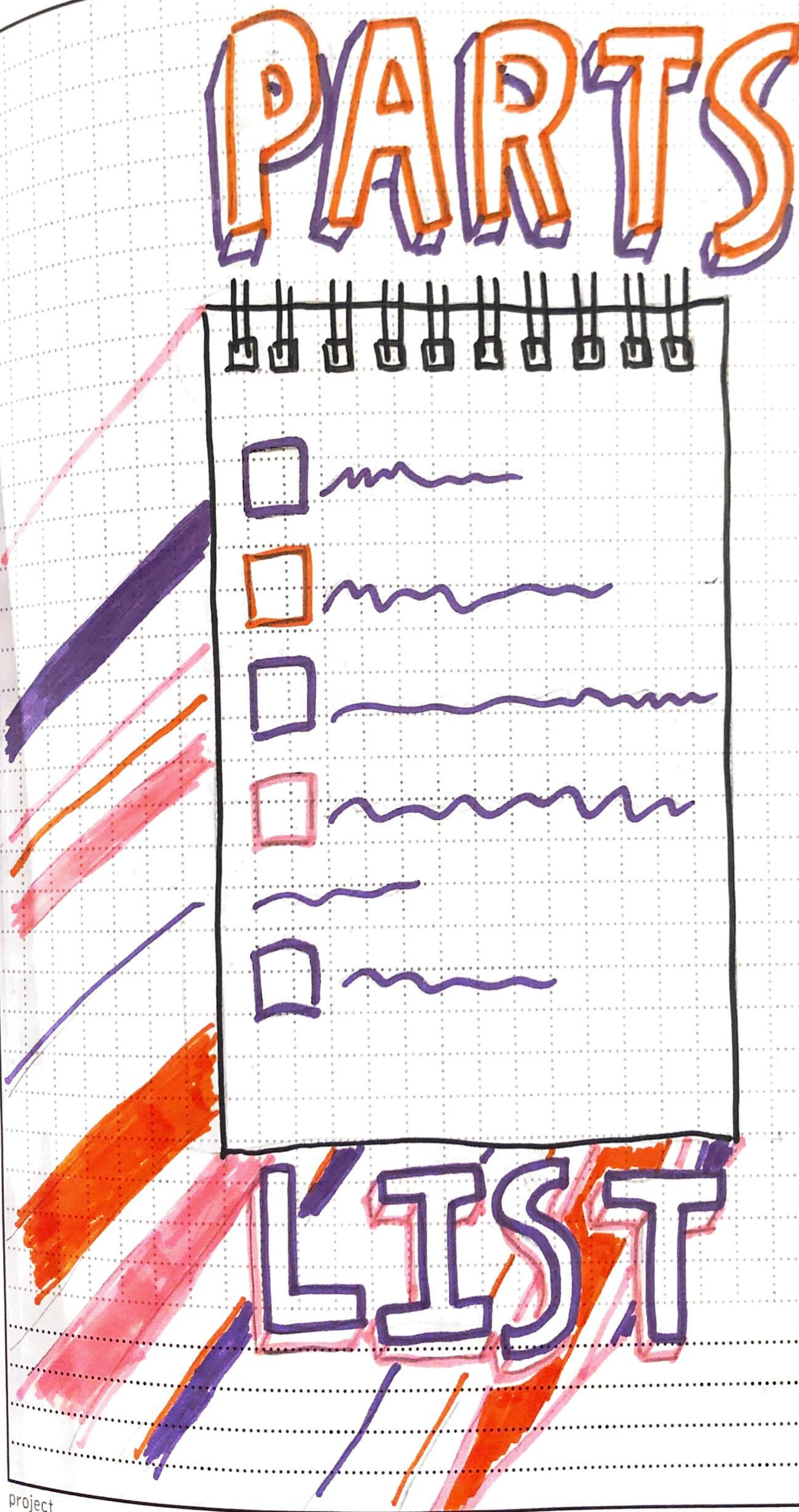
Both
- shoot balls
- holds 2 balls

NEW
- gear ratio 1:1
- more rigid and consistent
- no force required to hold down position
- Stays down on its own

project
designed by:
witnessed by:
date:

- 2021-22 Pitching In robot parts list

  Pegs
- Wheels [6]
- Axles [6]
- Stoppers [20]
- 2x2 axle hole for wheel [4]
- 2x2 [2]
- 16x2 beam
- Rubber bands [12]
- Gears [4]
- Stand-off connectors [13]
- Standoffs [30]
- 6x1 beam
- 9x1 beam
- 4x6 beam
- Conveyor belt gear
- Washers
- Chains
- Angle Pieces
- 10x2
- 20x2
- Brain [1]
- Battery
- 2x5 beam [2]
- Spacers
- 2x12 beam [7]
- 2x14 beam
- 7x1 beam
- L Piece - small [2]
- Elevator Shaft Piece
- 18x2 beams
- 1x13 beam[2]
- 2x2 connectors [6]
- X2 pegs w/ heads [30]



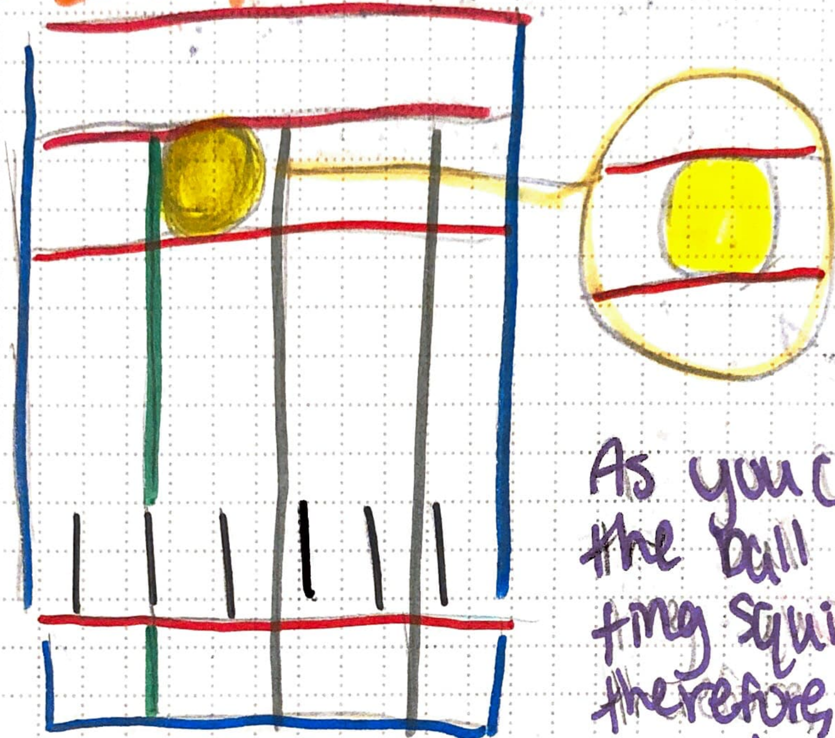
project

designed by:

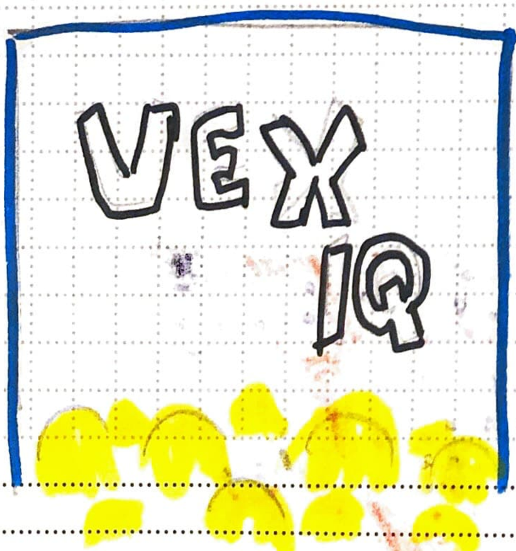
witnessed by:

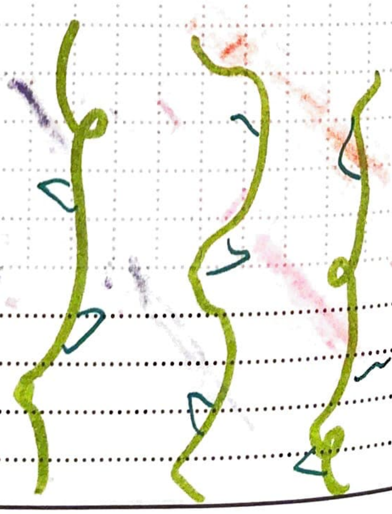
date:

BALL Deposit Cont.



As you can see, the ball is getting squished therefore, preventing the ball to come out into the bucket.





project

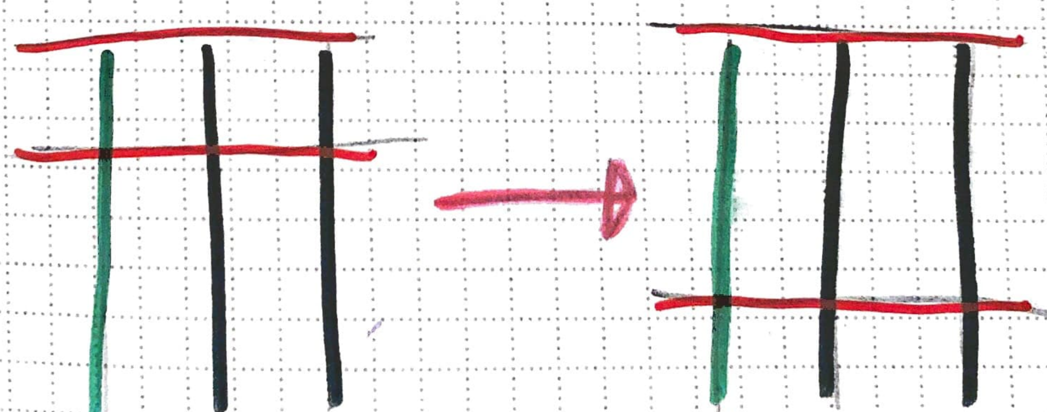
designed by:

witnessed by:

date:

fixing it [the process]

As you can see in the illustration on the other page, it was stuck between 2 bars. So, in order to fix this, we tried bringing the bar closest to us up more towards the middle of the standoffs.

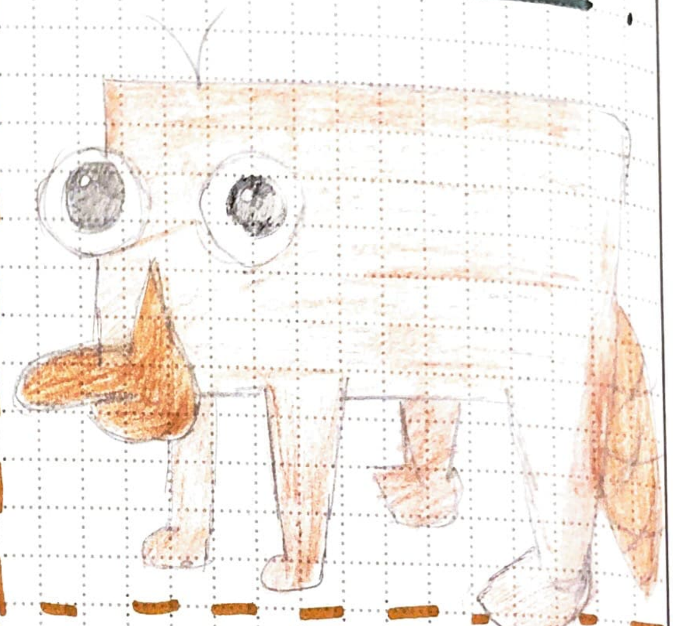


Before

After

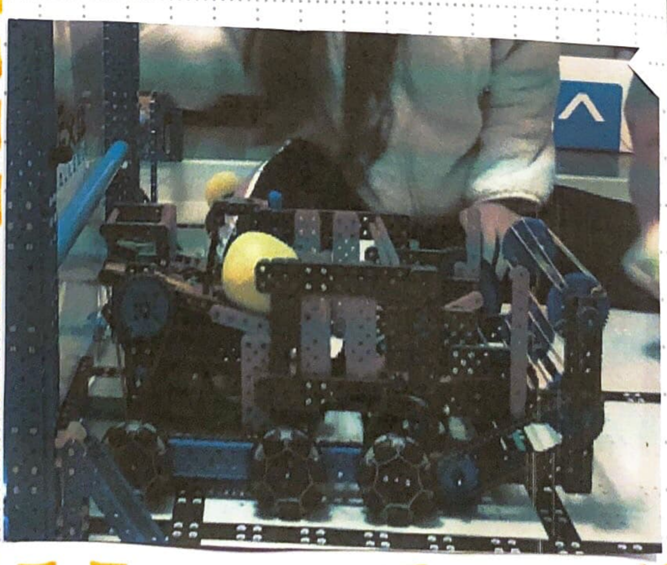
PLATYPUS

Our new and improved robot is called the Platypus. We mainly got the name from the side view of the robot. In total, this robot has five simple steps to successfully score into the high goal.



1) Roll the balls into the waiting basket

2) Lower the ramp while the rollers are rolling



3) Lower the shooter completely

4) Lift the ramp to deposit balls into the shooter

5) SHOOT!

project

designed by: Zoe

witnessed by:

date:

Pros	Cons
• Can shoot balls • Fits size limit • Roller rolls in balls	• Only enough space for 4 balls • Time consuming • Waiting area is too weak to hold more than two balls • Driver's job is too hard

Previous Bot

Pros	Cons
• Can consistently shoot balls • Fits size limit • Sturdy Design	• Little time consuming using all 5 steps • Balls sometimes goes left/right of the high goal.

Platypus

project

designed by: Zoe

witnessed by:

date:

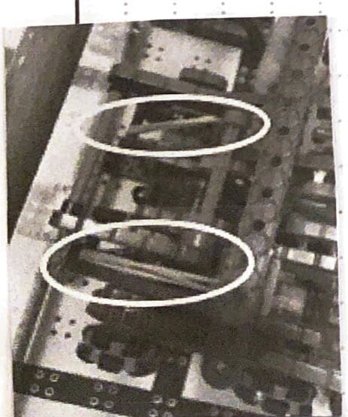
in Short...

This week, our team gathered together to test drive the Platypus. We found problems and solutions along the way.

Observations:

1st

- When rubberbands are put in the center of the shooter, balls go in all directions.
- When moved to the side of the standoffs, balls shoot straight.



2nd

- The second observation is that the left shooter's rubberband power was loose.
- Therefore, we had to adjust it multiple times

designed by: Zoe

witnessed by:

date:

PROBLEM

- After rolling balls in either the right or left shooter doesn't shoot as far, hits the high bar, or both go into the high goal.

- In the competition if we shoot too much, the rubberbands will stretch too much and the shooters will not be as powerful as before.

SOLUTION

- We moved the rubberbands to the side of the standoffs below the shooter to help guide the ball when shooting.

(we are still working on the second solution)

80-20 RULE

What is it?

- Distributing 100% between what is good and what we need to work on.

Why?

- To give us a view of what we need to accomplish

How?

- List what we did which is good
- List what needs improvement
- Turn into percentage!

80%	20%
• Roller • Ramp • Drivetrain	• Transitions for balls • Shooting angle

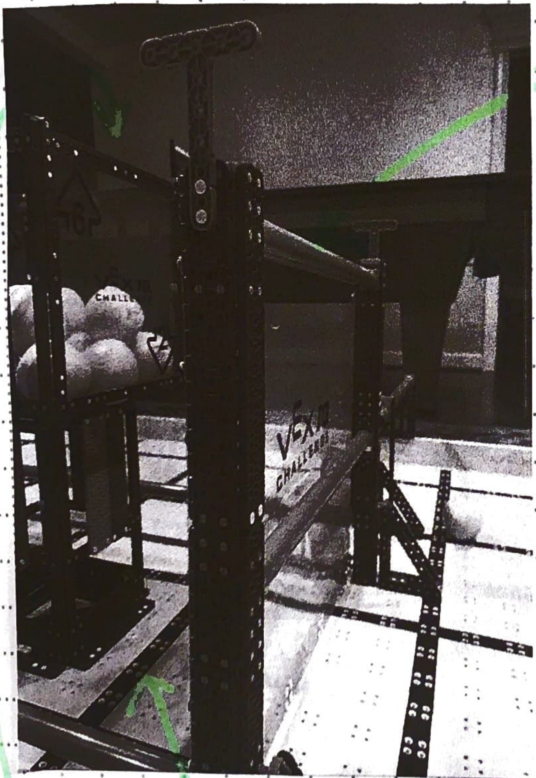
project

designed by: Zoe

witnessed by:

date:

PICTURES & Captions



Hit Bar

Goes in basket

Falls into 2pts Zone

My Notes:

Date. 11-19-21

Balls go in one by one;
- Since two shooters are filled, balls will "overflow"

Driver Control: (Hard Job)
Stop -Go roller to prevent overflow.

Job:
- Drive around
- Collect balls
-

© San-X

My Notes:

Bar hitting (Right)		Bar hitting (Left)		1/2 going into ZA	
Left	Right	Left	right	Left	Right
	III	III		Both	Both
					I
				Success%:	
				III	

Date.

witnessed by:

Zoe

19-21

ilakkuma™

Problem:

- After rollers roll balls in, right ball shooter / left ball shooter

- In competition, have to adjust bands spec

Observation:

- Left shooter rubberband - Too loose meaning there is less power

Solution:

Rubberband	moved from
center to	left / Right
on shooter.	Shooter
has to	be down complet
80% - Go	od
20% -	Path to shooter / stopper for Bands

Online References

name	website
VEX Robotics	www.vexrobotics.com
REC Foundation	www.roboticseducation.org
Robot Events	www.robotevents.com
VEX Robotics Support & Forum	www.vexrobotics.com/support
Project Lead the Way	www.pltw.org
BEST Robotics	www.bestinc.org
VEX Forum	www.vexforum.com
VEX IQ Forum	www.vexiqforum.com
VEX EDR Curriculum	http://curriculum.vex.com
VEX IQ Curriculum	www.vexiq.com/curriculum

Business & Resource Contacts

name	company	email	phone
VEX Support	VEX Robotics	support@vex.com	903-453-0802
ROBOTC Support	CMU Robotics Academy	support@robotc.com	412-687-9067

Goal Setting

goal

steps required to accomplish goal

=

criteria used for assessment

goal

steps required to accomplish goal

=

criteria used for assessment

goal

steps required to accomplish goal

=

criteria used for assessment

goal

steps required to accomplish goal

=

criteria used for assessment

goal

steps required to accomplish goal

=

criteria used for assessment

goal

steps required to accomplish goal

=

criteria used for assessment

Mechanical Motion Control

Velocity = Distance / Time
Acceleration = (Final Velocity – Initial Velocity) / Time
Force = Mass x Acceleration
Circumference = Diameter x Pi (3.14) = Radius x 2Pi (6.28)
Feet Per Minute (FPM) = RPM x Circumference in ft.

Electrical

Ohms = Volts/Amperes (R=V/I)
Amperes = Volts/Ohms (I=V/R)
Volts = Amperes x Ohms (V=IR)

Mechanical Power Transmission

Gear Ratio = (Driving Gear Teeth):(Driven Gear Teeth)
Gear Reduction = Driven Gear Teeth / Driving Gear Teeth
Output Torque = Input Torque x Gear Reduction
Gear Reduction Required = Output Torque / Input Torque
Output Speed = Input Speed / Gear Reduction
Gear Reduction Required = Input Speed / Output Speed
Compound Gear Reduction =
Gear Reduction 1 x Gear Reduction 2 x Gear Reduction 3 x (all other gear reductions)

Speed, Power, and Torque

Speed = Distance / Time
Rotational Speed = Rotational Cycles / Time
Rotational Speed = Degrees / Time
Torque = Force x Distance
Force = Torque / Distance
(Distance is from the axis of rotation)
Power = Force x Velocity

Motors

Key Specifications:

Stall Torque ... Free Speed ... Stall Current ... Free Current

For determining motor characteristics as voltage varies:
New Value = Spec Value x (New Voltage / Spec Voltage)

For determining Motor Current Draw at a given Torque Load:
Current Draw = ((Stall Current – Free Current) / Stall Torque) x Given Torque Load + Free Current

For determining the Torque Load at a given Motor Current Draw:
Torque Load = (Given Motor Current – Free Current) x Stall Torque / (Stall Current – Free Current)

For determining Motor Rotational Speed at a given Torque Load:
Rotational Speed = -(Free Speed / Stall Torque) x Given Torque Load + Free Speed

For determining "Super" Motor Specs from Multiple Motors combined as one, geared to the same Speed:

Free Speed = SAME
Stall Torque = Sum of all Motor Stall Torques
Stall Current = Sum of all Motor Stall Currents
Free Current = Sum of all Motor Free Currents

Commonly Used Engineering Symbols

a	acceleration
A	ampere
A	area
AC	alternating current
Btuh	British thermal unit/hour
BTU	British thermal unit
c	calorie
C	circumference
°C	degree Celsius
Ø	diameter
cc	cubic centimeters
cm	centimeter
cps	cycles per second
Δ	change
δ	deformation
d	diameter
θ	displacement
D	distance
dB	decibel
DC	direct current
deg/s	degrees per second
e	strain or unit elongation
E	effort
E	modulus of elasticity
F	force
°F	degree Fahrenheit
ft	feet
ft-lb	foot-pounds
g	gram
g	acceleration of gravity
gal	gallons
gpm	gallons per minute
GR	gear ratio
H	height
hp	horsepower
Hz	hertz
I	current
I	moment of inertia
in.	inch(with period)
in.-lb	inch pounds
J	joule
K	kelvin
kg	kilogram
km	kilometer
kph	kilometers per hour
kW	kilowatt
L	length
L	liter
lb	pound or pounds
lbf	pound force
lb-ft	pound feet
lpm	liters per minute
m	mass
m	meter
M	moment force
μm	micrometer
MA	mechanical advantage
mg	milligram
mA	milliampere
min	minute
mL	milliliter
mm	millimeter
mph	miles per hour
ms	millisecond
MΩ	mega ohm
N	Newton
N-m	Newton-meters
ns	nanosecond
oz	ounce
‖	parallel
⊥	perpendicular
p	pitch
p	momentum
P	power
p	pressure
pi(π)	(3.1416)
psi	pounds per square inch
r	radius
R	resistance
R	rate

Commonly Used Engineering Symbols (cont.)

R	reaction force
R	R-value
°R	degree Rankine
rpm	revolutions per minute
ρ	density
s	second
s	normal stress
Σ	sum
τ	shear stress
T	temperature
t	thickness
t	time

τ	torque
θ	angle theta
TPI	threads per inch
v	velocity
V	Volt
V	Volume
w	weight
W	watt
W	work
ω	angular velocity
Ω	Ohm
x	horizontal displacement
$\bar{x}$	mean or average
y	vertical displacement
Z	plastic section modulus

fraction	decimal	percent
½	0.50	50.0%
1/3	0.33	33.0%
2/3	0.67	66.7%
1/4	0.25	25.0%
3/4	0.75	75.0%
1/5	0.20	20.0%
2/5	0.40	40.0%
3/5	0.60	60.0%
4/5	0.80	80.0%
1/6	0.17	16.7%
5/6	0.83	83.9%
1/8	0.13	12.5%
3/8	0.38	37.5%
5/8	0.63	62.5%
7/8	0.88	87.5%
1/9	0.11	11.1%
2/9	0.22	22.2%
4/9	0.44	44.4%
5/9	0.56	55.6%
8/9	0.89	88.9%
1/10	0.10	10.0%
3/10	0.30	30.0%
7/10	0.70	70.0%
9/10	0.90	90.0%
1/16	0.06	06.3%
1/32	0.03	03.1%

Conversions

Weight

1 gram = 0.035274 ounces	1 ounce = 28.35 grams
1 gram = 0.0022046 pound	1 pound = 453.59 grams
1 kg = 35.274 ounces	1 pound = 0.454 kilograms
1 kg = 2.2046 pounds	100 pounds = 45.36 kilograms

Length

1 millimeter = 0.03937 inches	1 inch = 25.4 millimeters
1 centimeter = 0.3937 inches	1 inch = 2.54 centimeters
1 meter = 39.37 inches	12 inches = 304.8 millimeters
1 meter = 3.281 feet	1 foot = 30.48 centimeter
1 meter = 1.0936 yards	1 yard = 0.9144 meter
1 kilometer = 0.6214 miles	1 mile = 1.609 kilometers

Area

1 square centimeter = 0.155 square inches
1 square meter = 10.7639 square feet
1 square inch = 6.452 square centimeters
1 square foot = 0.0929 square meter

Many websites like "Convert-me.com" can also be used for conversions.

Volume

1 milliliter = 0.0339 fluid ounces	1 fluid ounce = 29.5735 milliliters
1 liter = 4.2268 cups	1 cup = 0.2366 liters
1 liter = 2.1133 pints	1 pint = 0.4732 liters
1 liter = 1.0567 quarts	1 quart = 0.9464 liters
1 liter = 0.2642 gallons	1 gallon = 3.7854 liters

I ♥ Robots

VEXROBOTICS.COM

ROBOTICSEDUCATION.ORG

REC Foundation

VEX

276-3023-Rev2