Team 99909A Rising Phoenix

2022-23 Spin Up Season

Engineering Notebook

Meet our team:

Matthew

Daniel

Pranav

David

Cynthia

Cristina

Meet our robot Fawkes:

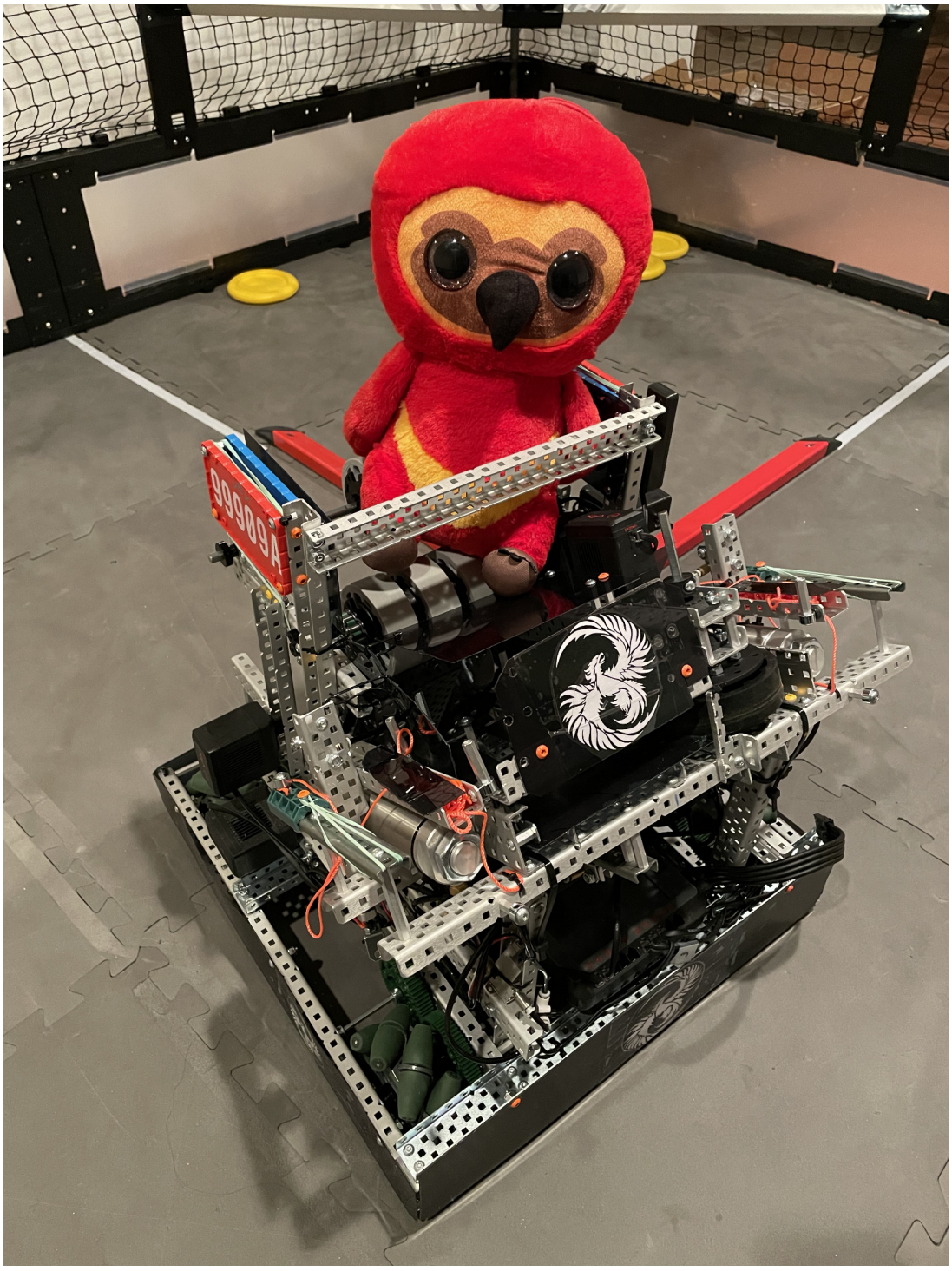


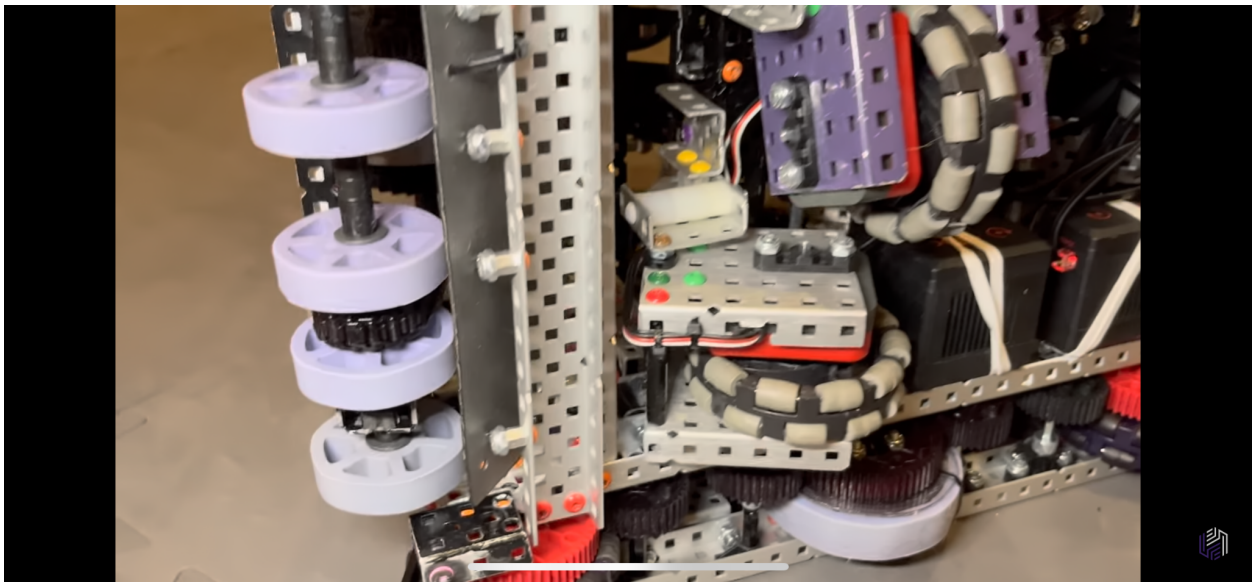
Table of Contents

Journal Entries

8/13

Primary objectives:

- Design and build Mecanum Wheel based drivetrain
    - Things to consider - need to make space for tracking wheels used in the odometry

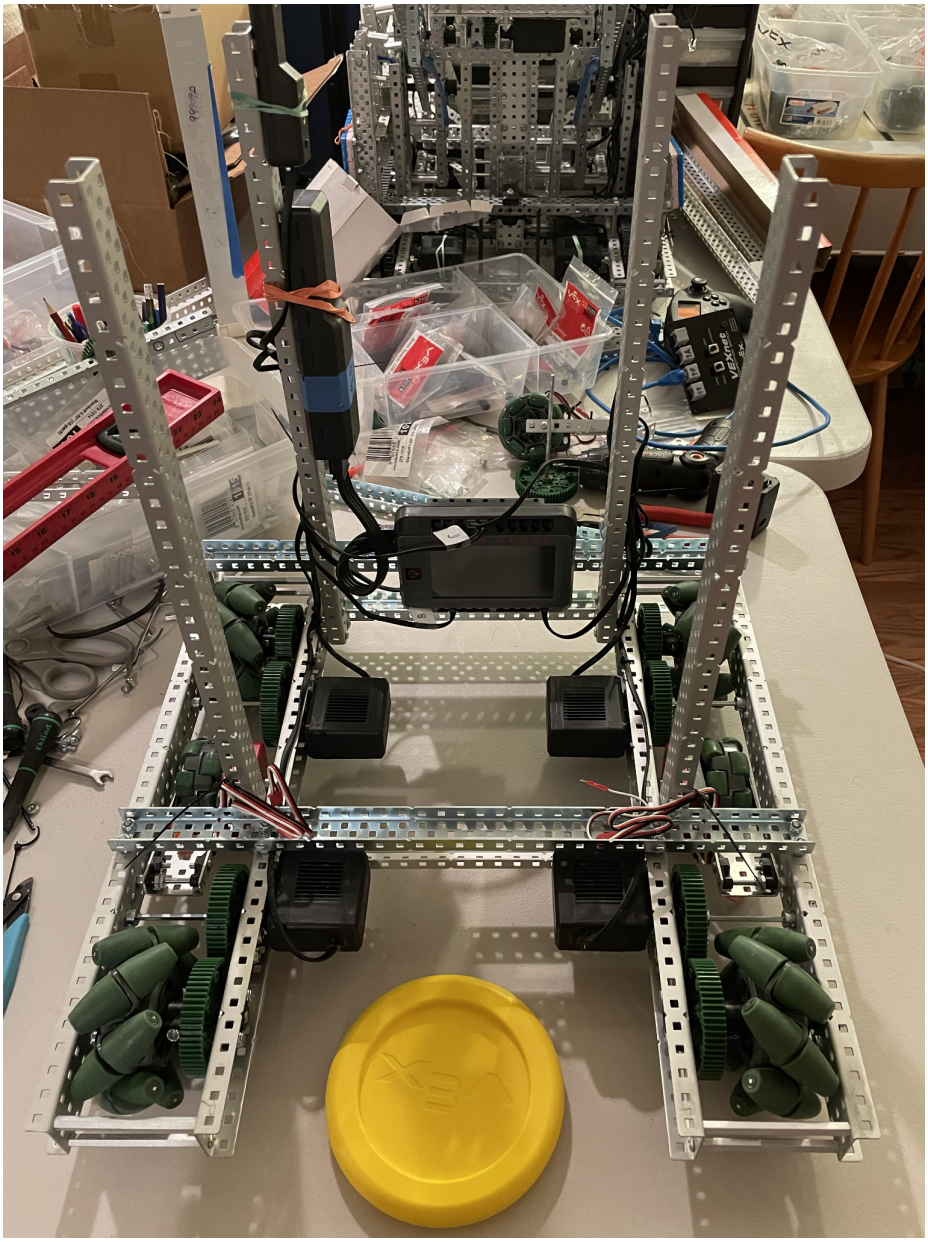


- Design and build a flywheel based disc shooting mechanism

8/20

Primary objectives:

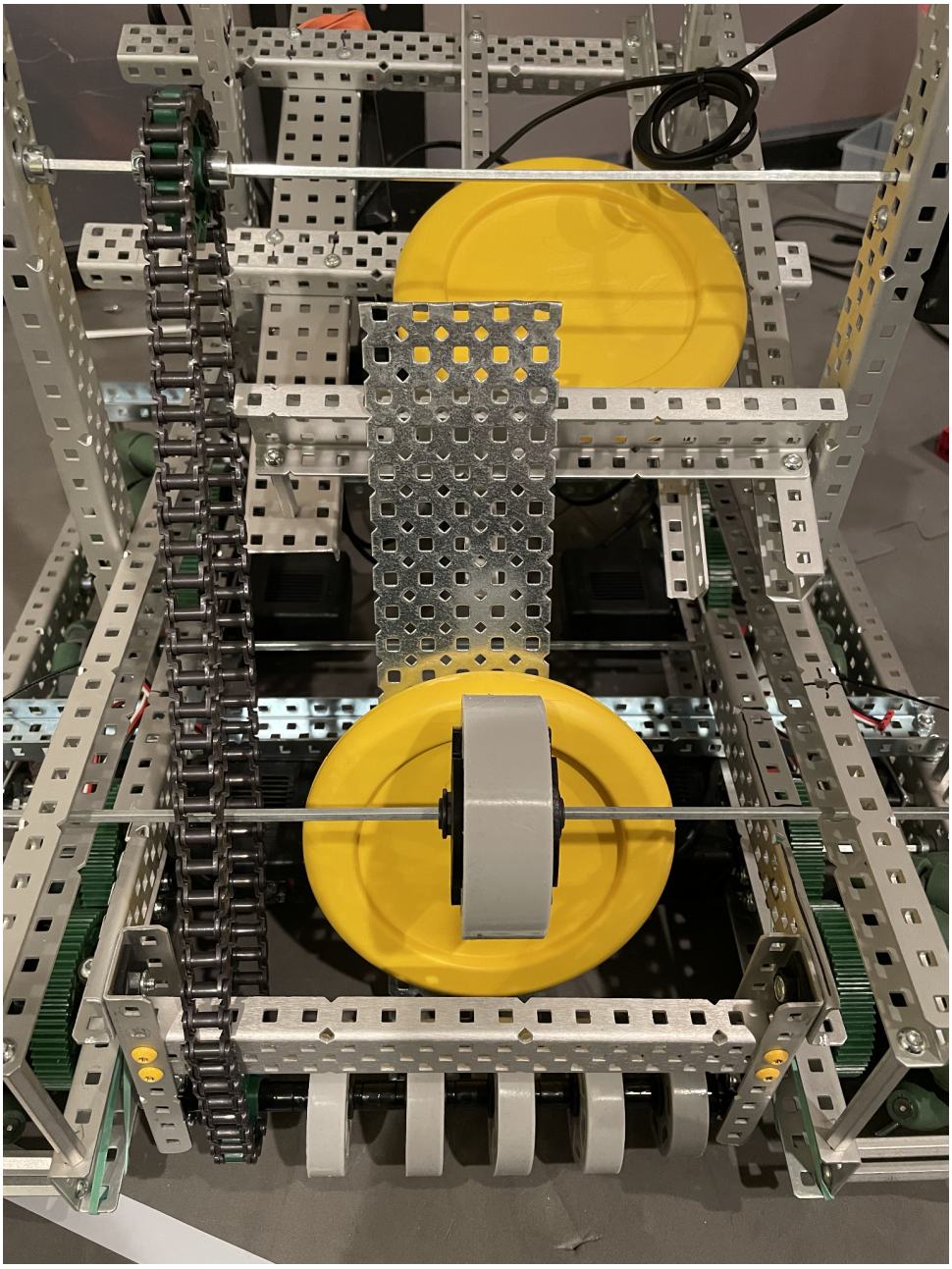
- Progress made on drivetrain



8/27

Primary objectives:

- Brainstorm ideas on disc intake and shooting mechanism

- 

9/3

Primary objectives:

- Design and build shooting mechanism

We are making progress on the flex wheel based disc shooting mechanism. We used blue 6:1 Cartridge (600 RPM) for the pair of spinning motors. That doesn't seem to provide enough momentum for discs to fly out far.
We added 84-12 gear pair to gear up the spinning speed even faster. Now our spinning flex wheels can go an impressive 4200 RPM.

9/10

Primary objectives:

- Testing shooting mechanism

We tested our disc shooter today. Everything appears to be working wonderfully. The discs do fly out quite a distance. We measured the furthest distance that a disc can go (with the right upward angle) is around 12 feet. That will cover the entire length of the field.

9/17

Primary objectives:

- Continue testing shooting mechanism

Our shooter testing hit some snag. After a few shots, the pair of shooter motors heat up really fast. We have to constantly cool them using compressed air. This won't work in tournaments. We need to trouble-shoot and find solutions.

9/24

Primary objectives:

- Install pneumatic piston for loading discs
    - Piston is installed on a rail underneath the shooter's disc magazine.
    - When activated, the piston will push a disc into the flywheel to be launched.

We figured out the issue with shooting flex wheels. We are using a pair of high-strength 84-12 tooth gears to gear up the RPM of the flex wheels. As it turned out, the 2 gears are too snug together with no space between them at all. This will cause a great deal of friction when the wheels are spinning at high speed therefore putting a lot of stress on the spinning motors.
After some research, we discovered that regular 84 - 12 tooth gears are slightly smaller than the high-strength cousin, therefore provide just the right spacing between. We switched off the high-strength gears with regular gears. The motor overheating issue went away. We are able to make continuous shooting without overheating the motors.

10/1

Primary objectives:

- Disc collection mechanism

The biggest issue we are facing is how to get the discs laying flat on the floor to be picked up by the flex wheels. There must be some ramp where discs can climb.

10/8

Primary objectives:

- Disc collection mechanism continues

We are having a hard time to consistently pick up discs from the floor. If we push the discs a little hard with our hands, the discs will go up the ramp with no issue. But robot on its own is having trouble picking up the discs.

10/15

Primary objectives:

- Disc collection mechanism continues

We are still having the same issue with the disc collection mechanism. We tried a few things, like using different size flex wheels or putting more tension on the rubber band

that pushes the flex wheels on the yellow discs. So far, none of the solutions seem to work out.

10/22

Primary objectives:

- Finished shooter installation on robot.

Ideas for robot:

- Adjust the shooter's angle and launch direction
    - Currently, discs come out curved to the left because the flywheel is pulling it leftwards as the discs launch.
    - Plan to install a device to guide the discs further as it is launched.

10/29

Primary objectives:

- Testing shooter mechanism on the robot.
- Troubleshooting disc collection mechanism

We have successfully tested the shooter mechanism by itself. Now that the system is installed onto the robot, we want to make sure it still works flawlessly. As it turned out, it does. The discs could travel up to 12 feet.

But we are still having issues with the disc collection mechanism.

11/5

Primary objectives:

- Fix disc collection mechanism

We finally figured out the problem with the disc collection mechanism. The rotation speed of the disc collecting flex wheels is too slow (200 RPM). Discs don't get enough momentum to go all the way up on the ramp. After switching to 600 RPM, all the issues with disc collection mechanism went away. Hooray!

11/12

Primary objectives:

- Adjusting the shooter
    - Making sure that the shooter shoots straight, not curved.
- Design a rope launcher to deploy ropes that cover tiles at the end of the game for one point per tile.

Ideas for robot:

- One button to turn shooter flywheel on/off
- Last second timer to launch ropes as a fail-safe in case drivers forgets.

We want to test out a new approach that will allow us to shoot the discs at 2 different distances (close and far). But using 2 different buttons that trigger different shooter motor spinning speeds, we might just be able to accomplish that.

11/19

Primary objectives:

- Program and test the dual speed disc shooter

We programmed 2 separate shooter motor speeds (8000 and 6500). We tested the shooting discs at 2 different distances (up close to the bucket and all way on the other side of the field). Both tests went successfully.

11/26

Primary objectives:

- Build Rope shooting mechanism

We have 2 competing designs. We will have a shootout competition to test out which design is superior.

12/3

Primary objectives:

- Practicing driving

- Programming autonomous routine

12/10

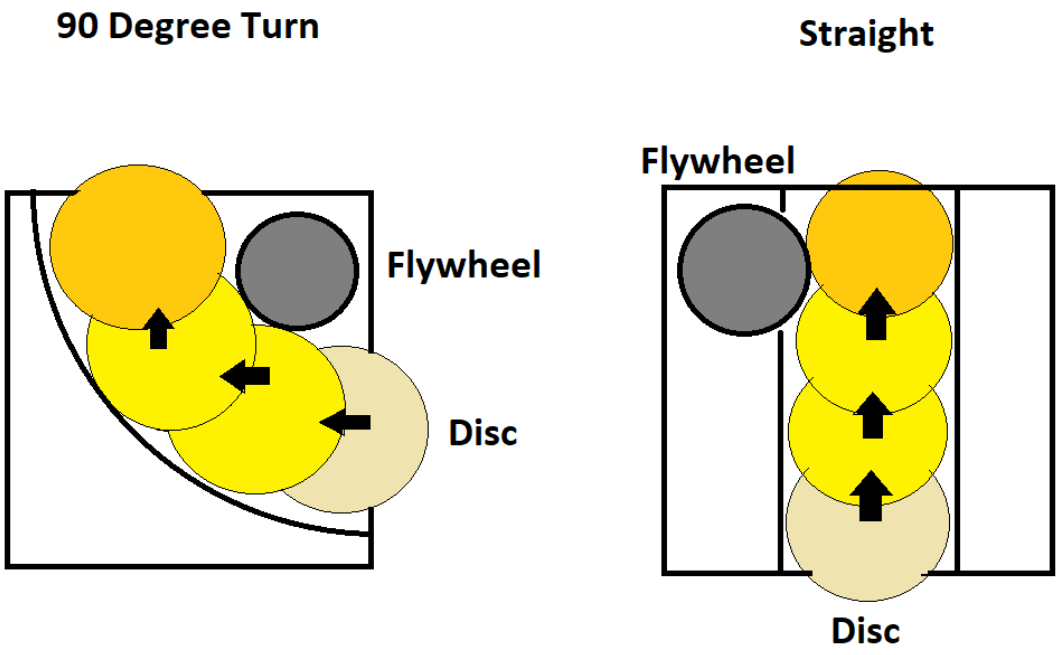
Primary objectives:

- We have an autonomous routine that will put 2 discs in the basket and spring the roller.

Shooter Mechanism

Based on what we saw from other robots, we found out that the flywheel would be the most efficient way to launch discs into the baskets. It was expected that we would be using flywheels even without looking at other robots, so we order the wheels three months prior because they took a long time to ship.

We first worked on the place where the wheel would travel to go to the flywheels. Our first idea was to make a **90 degree curve**, which had a plastic piece as a wall. After some more thinking, however, we realized that this was unnecessary because **we didn't need this change of direction** and could just load the discs in the **same direction**.

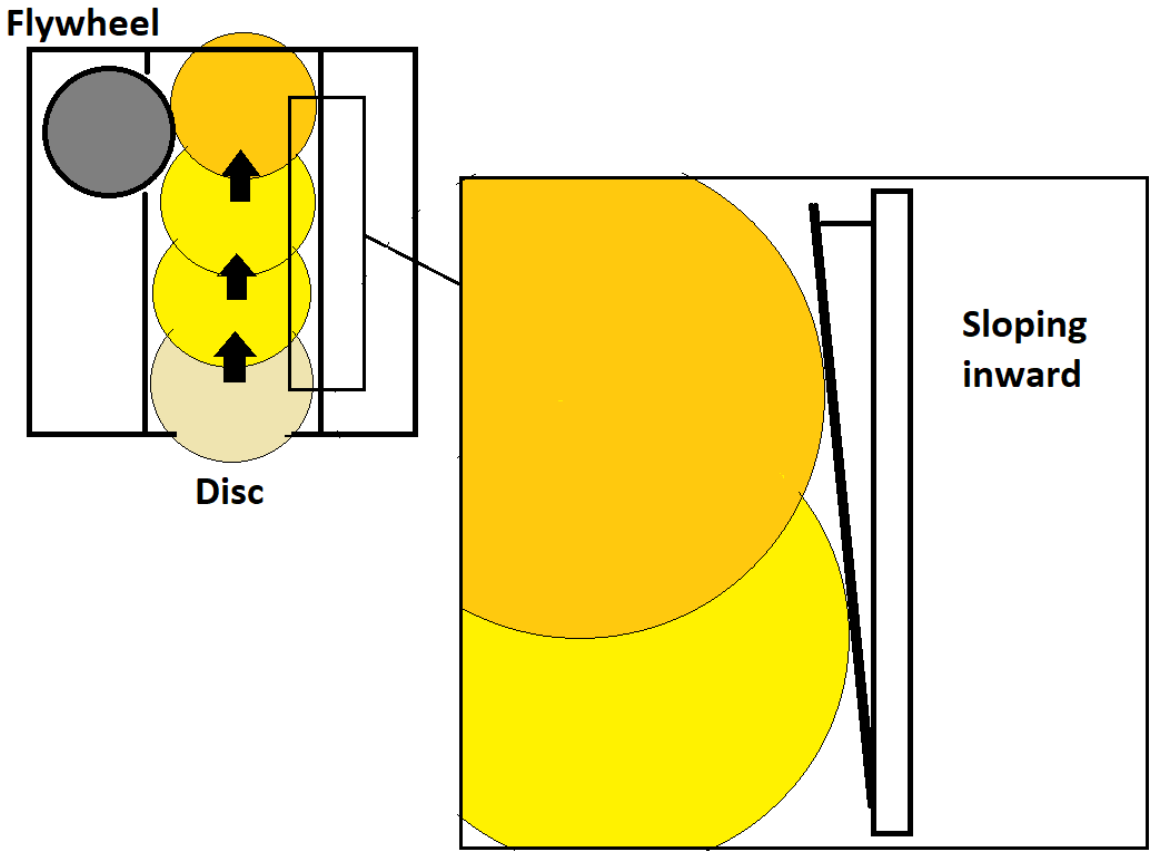


When making our new disc launcher, we decided to use aluminum instead of plastic. This was for **2 main reasons**:

- Aluminum was more sturdy
- Aluminum was easier to work with

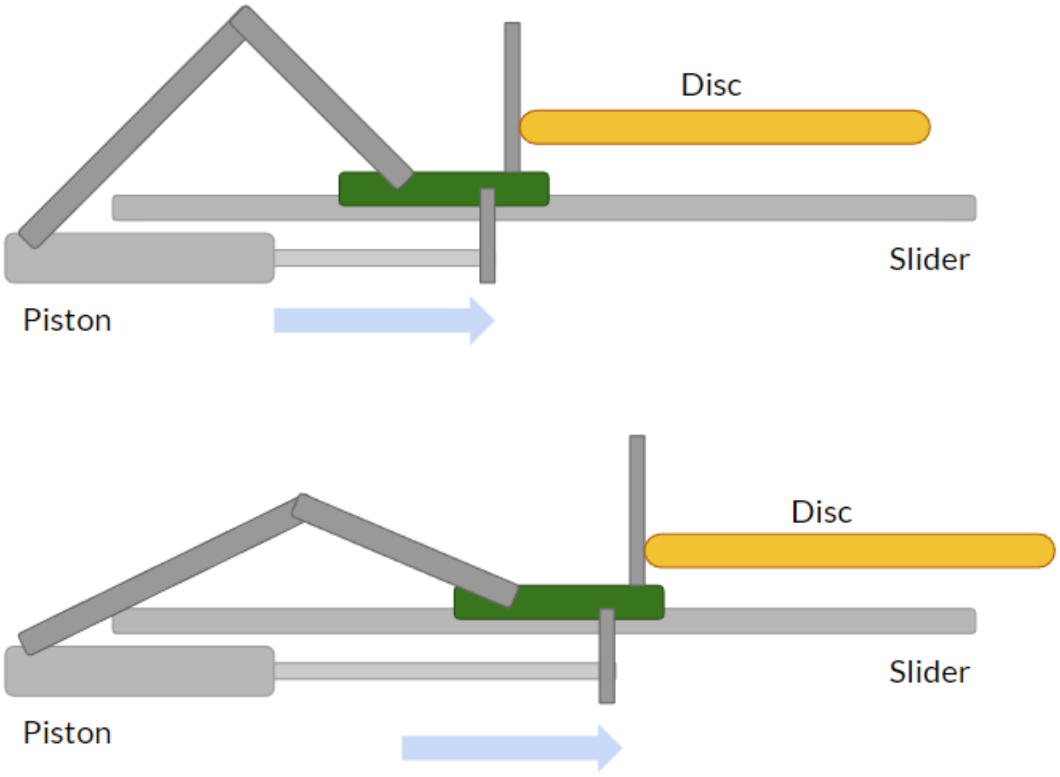
When we worked with plastic for the previous design, it was extremely work-intensive as we had to cut the plastic and measure it to the exact inch, where with aluminum we didn't have to cut as much and overall would just take less time and perform better.

Our first design consisted of a straight channel with the flywheels on the left and the guide on the right. We actually implemented a little slope on the right side so as the disc was pushed closer to the flywheels, it would kind of funnel and fit right in where it needed to be.



With this new design, we also tried using pneumatics. Previously, we did not use pneumatics and this put us at a major disadvantage as we could not have more mechanisms that would potentially help us. So, after seeing many other robots using pneumatics, we also decided to give it a try. We used pistons to push the disc up to where the flywheels would be for **2 main reasons**:

- Pistons were smaller than motors and therefore easier to implement
- We did not have to use 1 of the 8 motors we could use



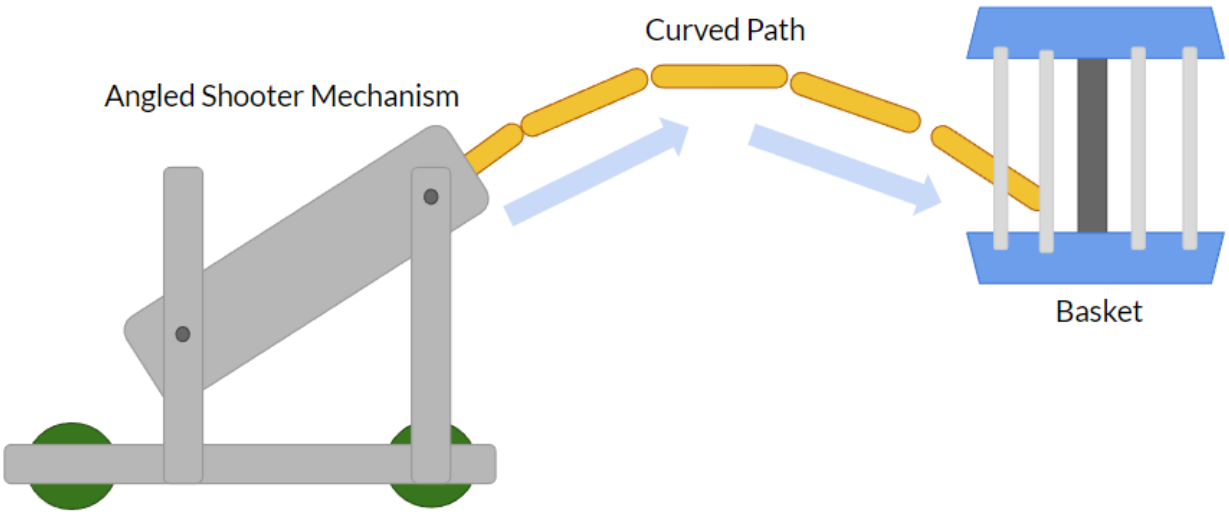
Finally, we had our first prototype of our fully-functioning shooting mechanism. It would push the discs as expected and it shot them much farther than we had anticipated, which was good news because this could be fixed with only some programming. However, we did run into another issue, which was that the discs were being launched at an angle. This meant it would be difficult to line up the robot to shoot in programming and driver

control. This was an easy fix as we just put in a piece past the place where the disc shot so when it shot, it would hit against those pieces to line up and shoot straight.

When mounting the shooting mechanism on the robot, it was clear that we had to mount it at an angle for **2 reasons**:

- So we could be able to score in the basket from a close distance
- We would not have to have as much power in the flywheel due to the parabolic path of the disc

  We left room for adjustment if we decided that we needed to change it later.



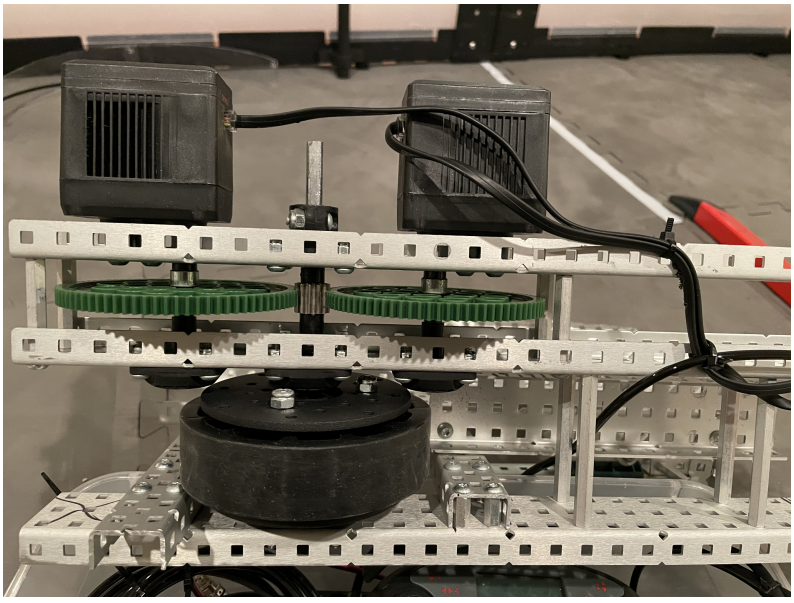
Flywheels

Flywheels are an essential part to our robot; this would be the way we scored the discs. This was a new experience for most of us because in previous years, we had not used flywheels to launch objects, only to collect them. So, we decided that our main objective would be to make it as **fast as possible.**

To do this, we would have to make the fastest gear ratio, and to achieve this, we had to change the motor cartridge to the fastest one which is 600(?). Next, we attached the biggest gears we had on the motors, and connected both of those to the smallest gear. That smallest gear would be connected to the flywheel. Because the biggest gear was being spun by the motor, the smallest gear would spin extremely fast.

After finishing this, we ran the thing and noticed a problem. The flywheels made a noise and after closer inspection, we realized that this was the result of the gears being too close together. In other words, the gears were too cramped. After minutes of thinking, we found a simple solution to this. We found another gear that was slightly smaller than the ones we were using currently so this would free up some space, and surely enough, after putting them in, it ran smoothly.

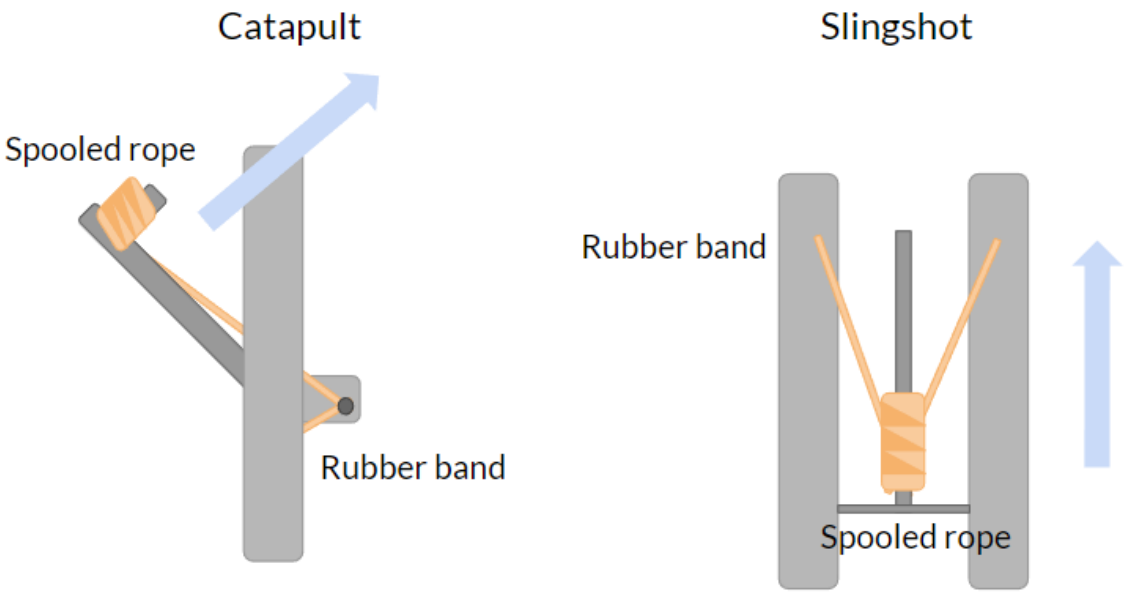
We put this in the shooting mechanism and another problem came up: **the flywheels slowed down too much**. This was a problem we had expected, but decided not to deal with it until we saw that it was a problem. This was easily fixed by adding weighted plates on the flywheels. This would make it so the flywheels spun at a greater force and would be less affected by any interacting object because of the weight of the flywheels.



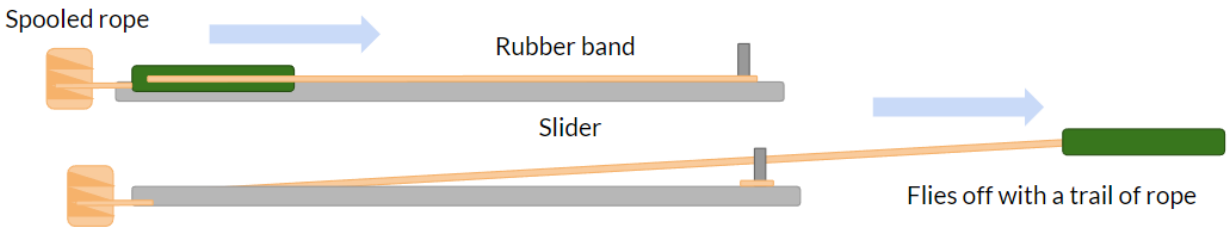
Rope Launching Mechanism

A crucial and simple mechanism that we decided to implement was a rope launching mechanism. This is used by many robots to cover as many tiles as possible, because during the last 10 second, the robots can extend as big as they want horizontally, and each tile touched is a certain number of points.

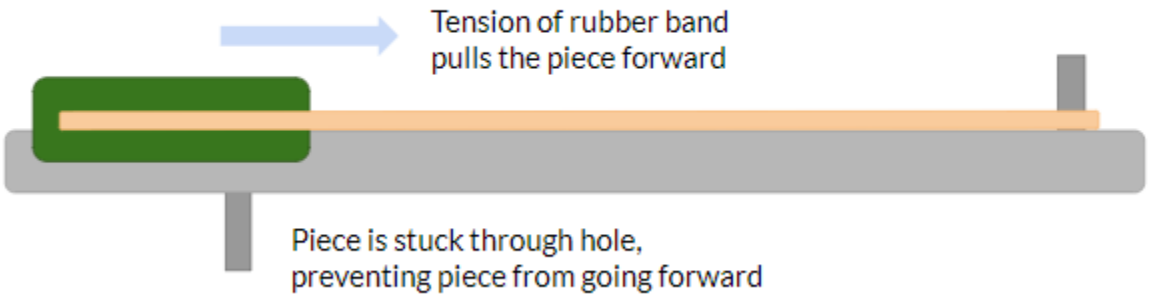
At first, we had two prototype designs, one was a catapult kind of design and the other was a slingshot design:



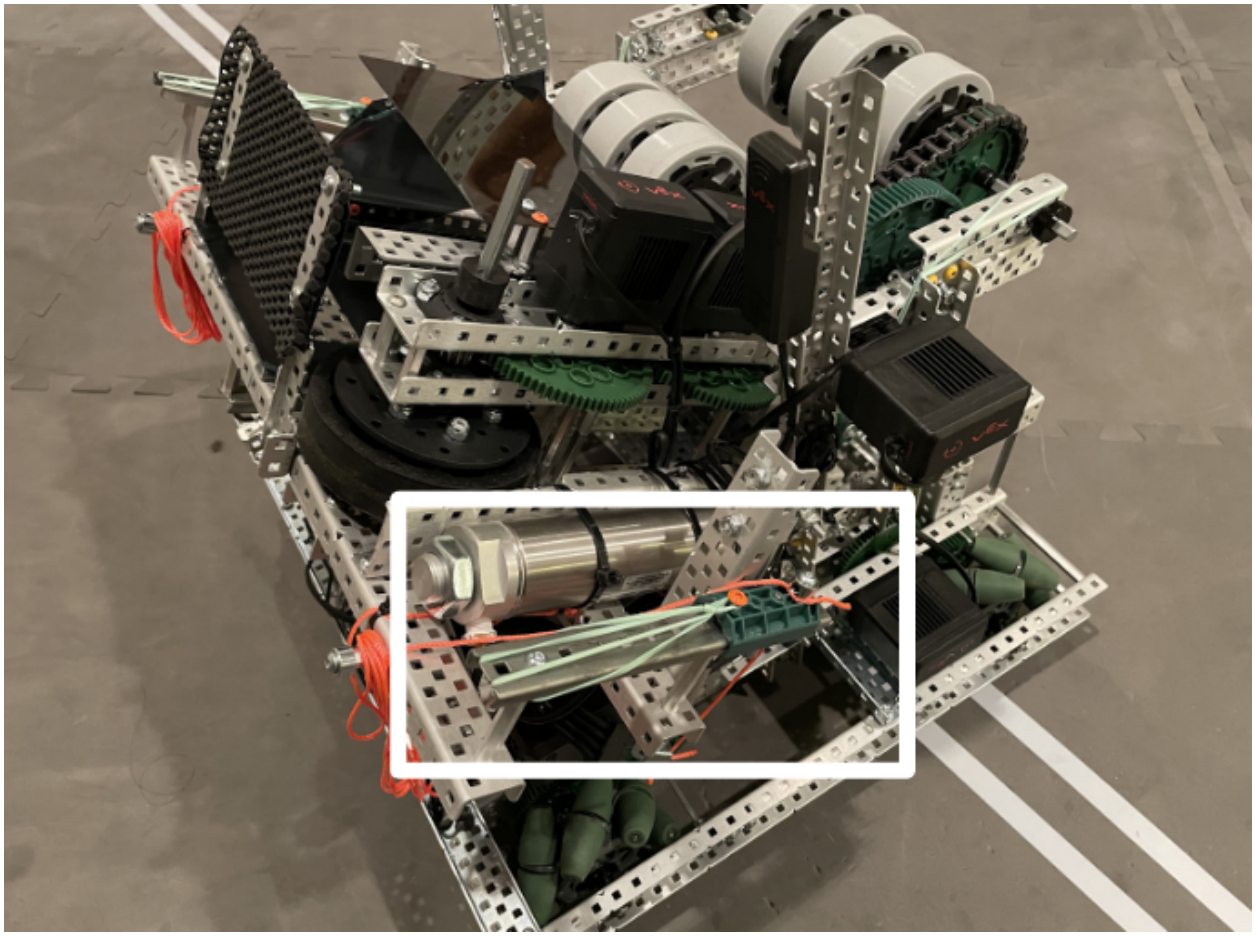
However, after testing, we saw that both of these designs could not launch the rope far enough and it just was not consistent enough. It was also extremely hard to set up. So, we decided to adopt the idea from our disc shooter trigger and used the sliding piece with a rubber band on it. We did this because we thought it would be easier to set up and slide easier.



When we powered and launched it by hand, it worked perfectly. It shot across the entire board consistently. Now we had to find a way to activate it without using our hands. So, we once again used pistons to do this. First, we had to add a stopper piece that would prevent the sliding piece from shooting forward and would be pushed out by the piston to activate it.



Next, we had to implement piston to push it out. We added the piston directly next to the launcher mechanism to keep it simple.



Pneumatics Powered Disc Shooter Trigger

This is the first year that our team is using a pneumatics system on our robot. There is a learning curve to climb.

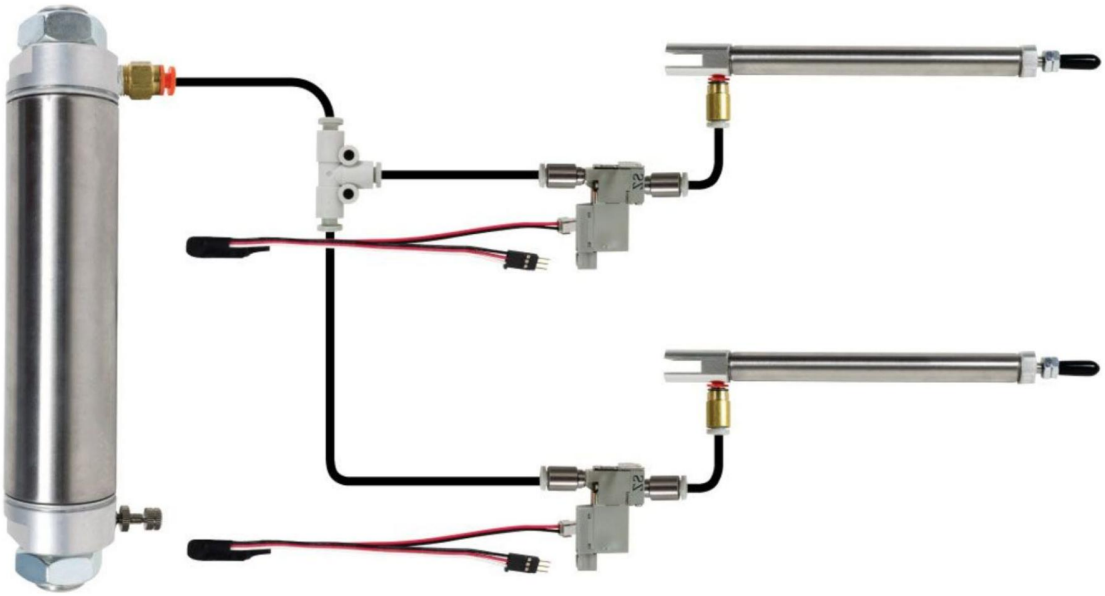
https://www.vexrobotics.com/pneumatics.html

https://kb.vex.com/hc/en-us/articles/4404197704212-Getting-Started-with-Pneumatics-with-the-V5-System

https://www.youtube.com/watch?v=0UqyWNspXHc

https://www.youtube.com/watch?v=7hYjRFp9h7o

VRC pneumatics system provides additional movement mechanisms beyond the allowed 8 motors limit and can greatly enhance the capability of our robot.



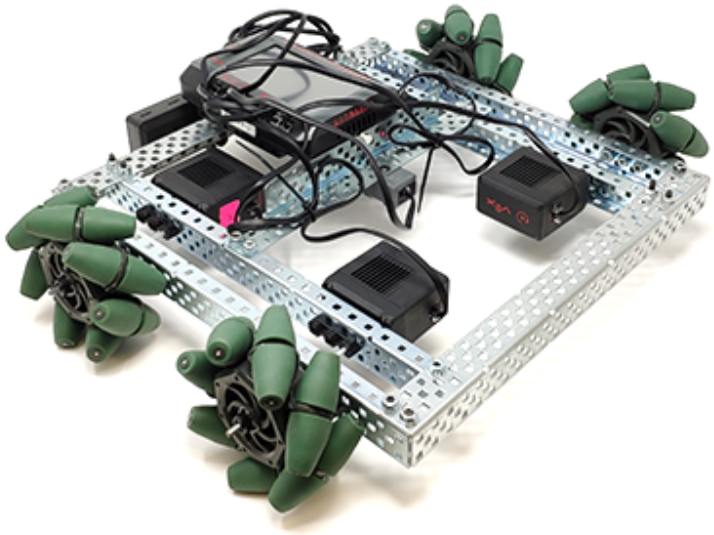
We went with the above single action pneumatics as the disc shooter trigger mechanism. We connected the air reservoir to the volve and to the cylinders following the above schematics and this tutorial

https://kb.vex.com/hc/en-us/articles/4404197704212-Getting-Started-with-Pneumatics-with-the-V5-System

After connecting the volve to the 3-port port on V5 brain, we were able to trigger the action of the cylinders using a programmed controller button.

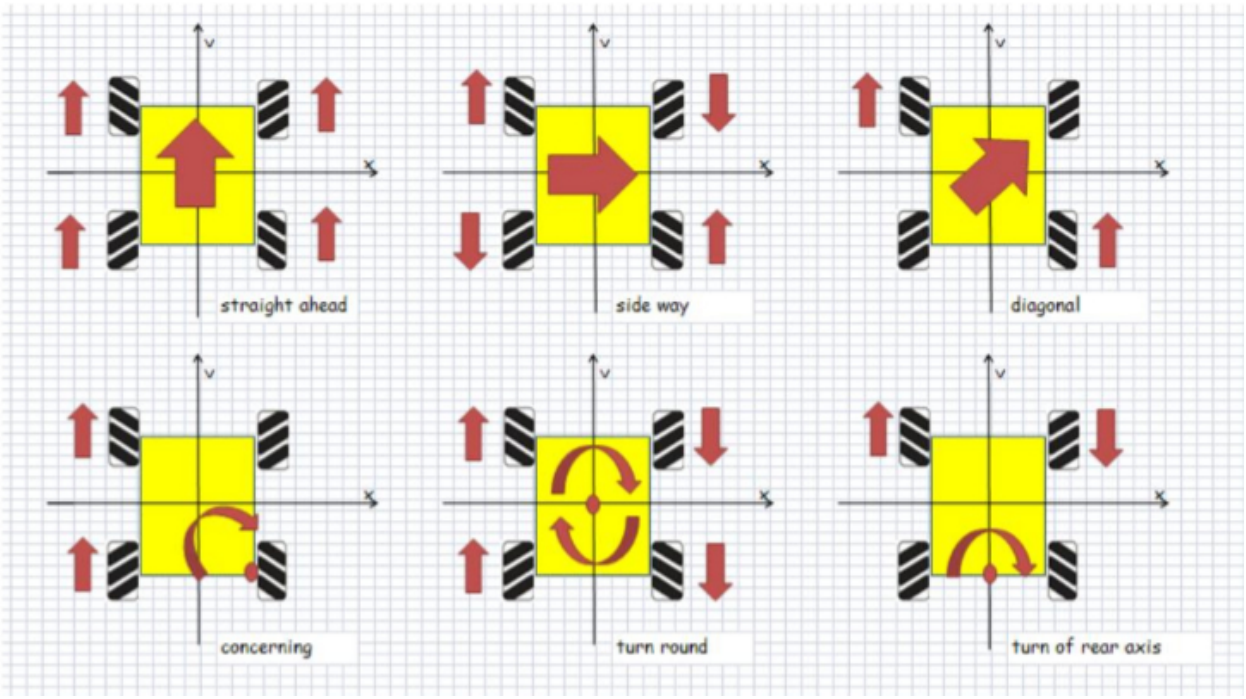
The next step is to figure out how to transform the cylinder action to the disc movement. Initially, we just connect the cylinder to a standoff that is attached to a glider. This mechanism works. But it takes too much space. We need to figure out how to make the trigger mechanism more compact.

Mecanum Wheel based Drivetrain

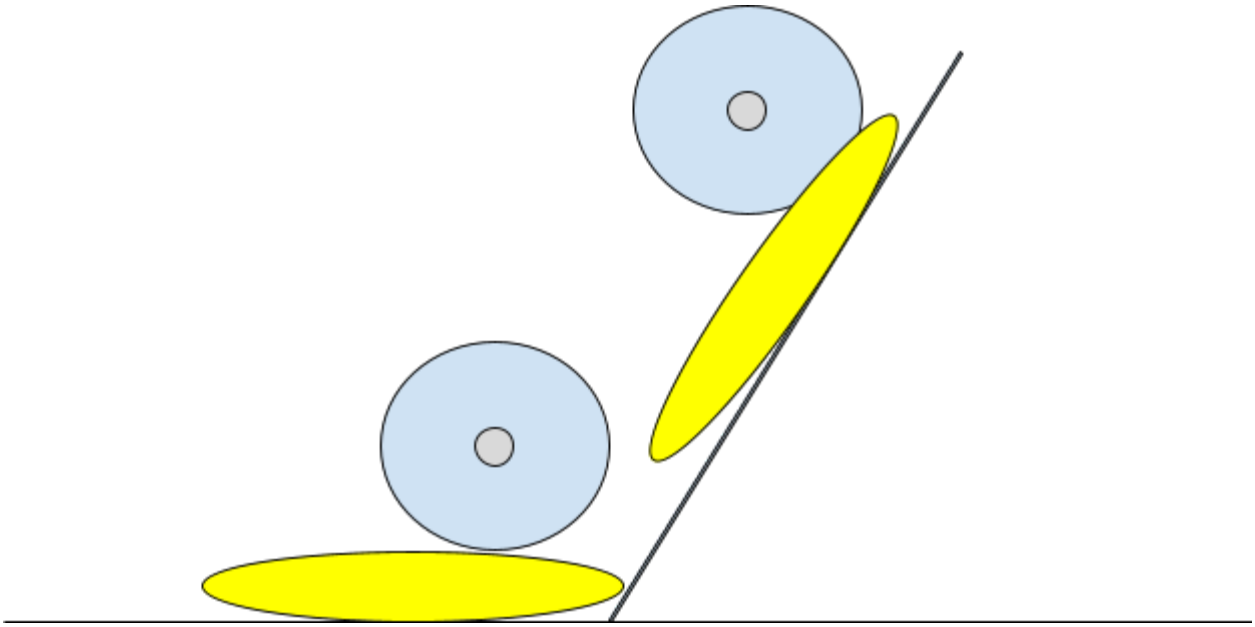


Considering the fact that we will need a lot of lateral movements to collect and shoot the discs, we decided to go with a mecanum wheels based drivetrain that looks like above. In a way, it's easier to create a mecanum wheel drivetrain because each wheel needs to ride on its own motor. There is not need to chain them together through chain or gears. However, it's more complex to program such drivetrain since the combination of wheels together create a variety of movement that we need for the robot.

We took help from this VEX Forum post and learned to program forward, backward, turn left, turn right, stride left and stride right.
https://www.vexforum.com/t/vexcode-mecanum-wheel-program/68482


straight ahead
side way
diagonal
concerning
turn round
turn of rear axis

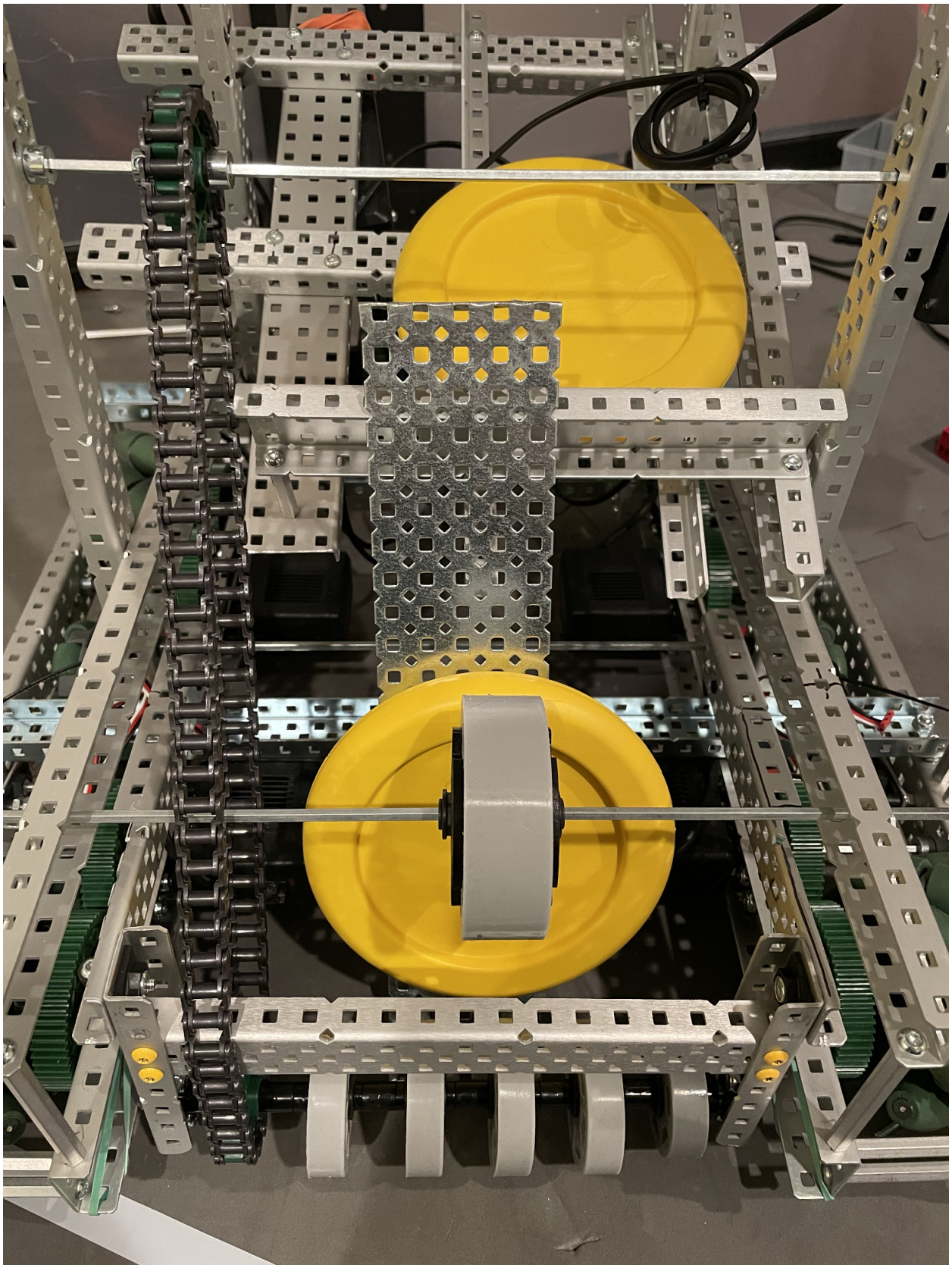
Disc Collecting Mechanism

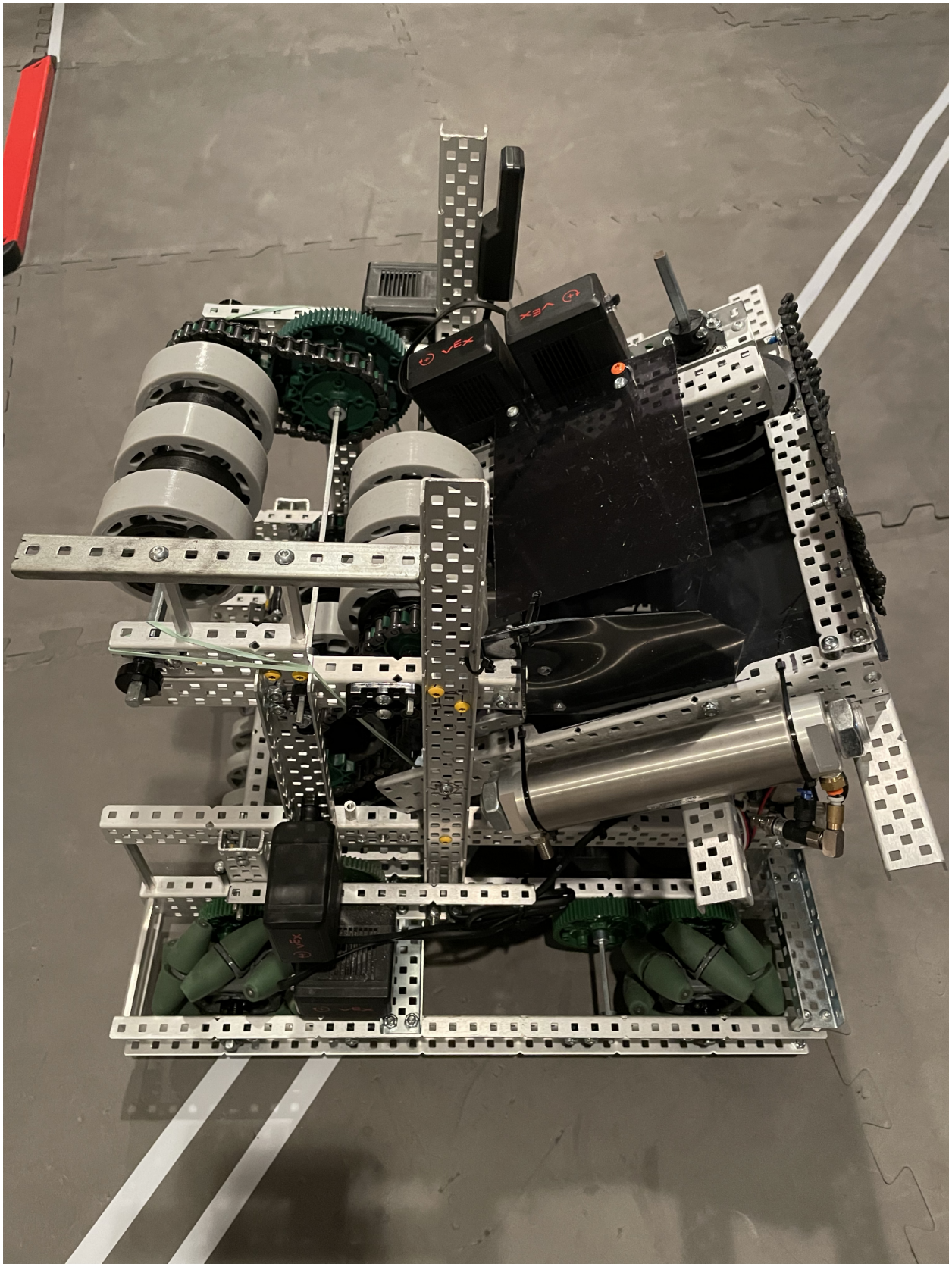


Our initial design of the disc collection system is quite simple. There is a ramp in the back with a pair of flex wheels spinning at high speed (200 RPM). The flex wheels will pick up the yellow discs from the floor and force them to move along the ramp to go upwards. The next pair of flex wheels will continue pushing the yellow discs further up until they reach the shooter disc holding area.

There are several challenges that we are facing as we are building the mechanism.

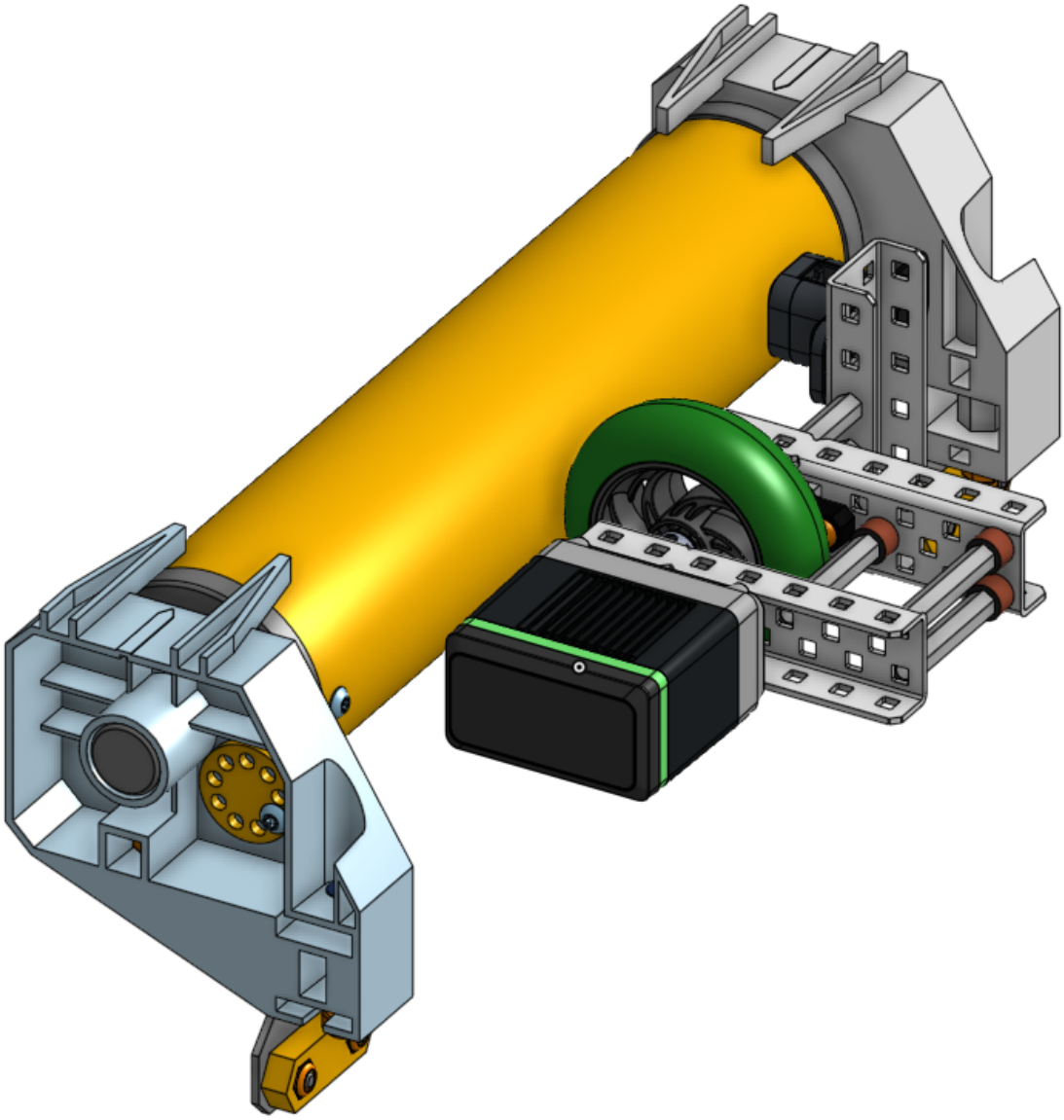
- The discs don't alway get picked up by the spinning flex wheels. Sometimes, they just kinda of vibrate on the floor without going upward onto the ramp
- The discs have a hard time landing in the shooter disc holding area. They always seem to get stacked onto each other in a weird manner.
- Sometimes, discs don't go all the way up the ramp. They will stay in the middle and rattle.




VEX
VEX

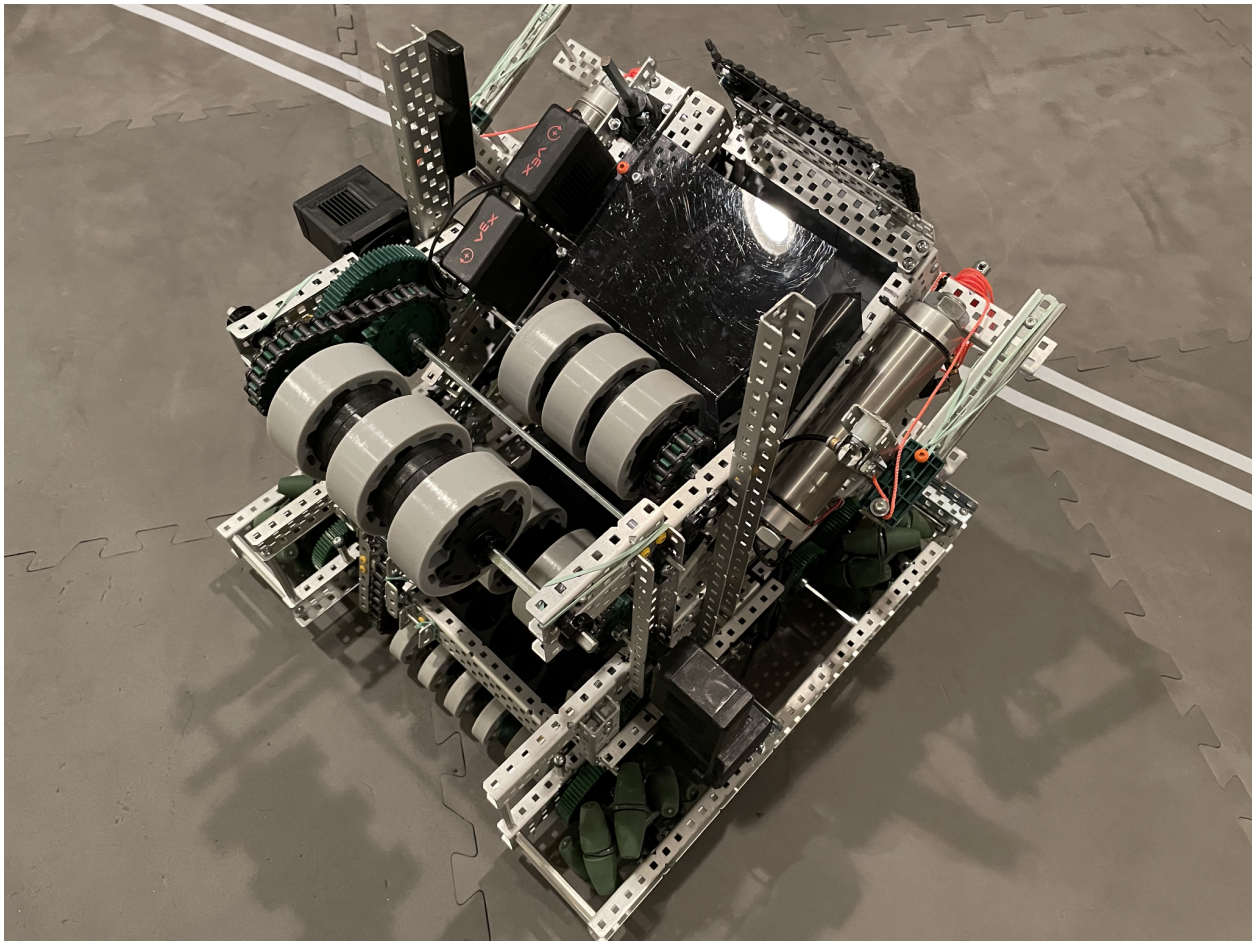
After much trouble shooting, we discovered that the collection flex wheels RPM is too low (200 RPM). After we switched to a high RPM (600 RPM), all the above problems went away. As it turned out, the discs don’t have enough momentum to go all the way up the ramp. A faster flex wheels solved the problem.

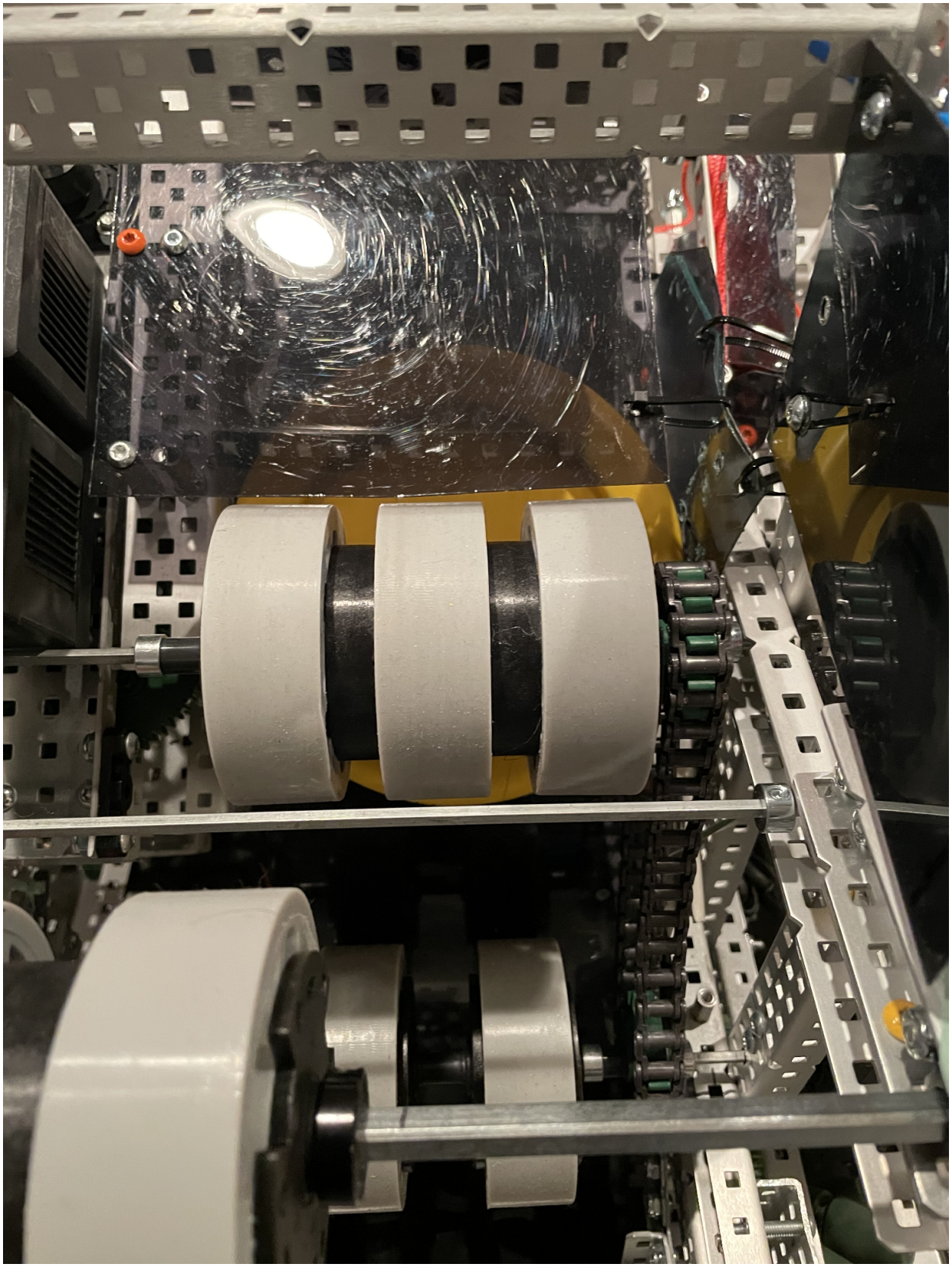
Roller Spinner



The design of the roller spinner is rather straight-forward. Basically, we already have spinning flex wheels that we use to collect discs from the floor. We just need to chain those flex wheels to a wheel that is at the right height to spin the roller.

The only issue here is the rotation speed. In order to pick up discs from the floor effectively, we have to use a 6:1 blue cartridge (600 RPM) to spin the flex wheels fast enough to catch the discs laying flat on the floor. However, such rotation speed turns to be much more than what we need for the roller spinner. In stead, we need to slow down the rotation and gain more torque in order to spin those rollers effectively. The only to achieve this is to gear down. We used 84 - 12 gear pair to reduce the rotation speed to 1/7 of the disc collection wheels. This turn out to be just the right speed for the roller spinner.





Programming

Our team code repository:

https://github.com/VRC99909A/SpinUp/tree/master/src

Collector.cpp

```
#include "main.h"


void CollectorMotor(double distance, int speed){
    //CollectorMotorA.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
    CollectorMotorB.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
    //CollectorMotorA.move_relative(distance, speed);
    CollectorMotorA.move_relative(distance, speed);
}

 void CollectorVel(int power) {
  CollectorMotorA.move_voltage(power);
  CollectorMotorB.move_voltage(power);
}


// thread for all collector controls
void Collector_fn(void* param) {

  //CollectorMotorA.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
  CollectorMotorB.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);

  while (true) {
```

```
    if (master.get_digital(DIGITAL_R1)) {
      CollectorMotorA.move_voltage(12000); //max voltage 12000
      CollectorMotorB.move_voltage(12000); //max voltage 12000
      //pros::delay(20);
    }
    else if (master.get_digital(DIGITAL_R2)) {
      CollectorMotorA.move_voltage(-12000); //max voltage 12000
      CollectorMotorB.move_voltage(-12000); //max voltage 12000
      pros::delay(200);
      CollectorMotorA.move_voltage(0); //max voltage 12000
      CollectorMotorB.move_voltage(0); //max voltage 12000
    }

  }

}
```

DriveTrain.cpp

```
#include "main.h"

// Returns true/false as to whether the drive wheels have reached their position goal set by driveForDistance
bool AtDistanceDriveGoal(int threshold) {
  return (abs((int)(FLMotor.get_position()-FLMotor.get_target_position())) < threshold);
}

//set drive target, but does not wait for them to reach their target
void Drive(double distance, int speed) {
  FLMotor.move_relative(distance, speed);
  BLMotor.move_relative(distance, speed);
  FRMotor.move_relative(distance, speed);
  BRMotor.move_relative(distance, speed);
}

void DriveVel(int power){
  FLMotor.move_voltage(power);
  BLMotor.move_voltage(power);
  FRMotor.move_voltage(power);
  BRMotor.move_voltage(power);
}

void Turn(double distance, int speed) {
  FLMotor.move_relative(distance, speed);
  BLMotor.move_relative(distance, speed);
  FRMotor.move_relative(- distance, speed);
  BRMotor.move_relative(- distance, speed);
}
```

```
void Shift(double distance, int speed) {
  FLMotor.move_relative(distance, speed);
  BLMotor.move_relative(- distance, speed);
  FRMotor.move_relative(- distance, speed);
  BRMotor.move_relative(distance, speed);
}

//thread for all drive train controls
void DriveTrain_fn(void* param) {
  /** - tank drive variables -
  int LeftControl = master.get_analog(ANALOG_LEFT_Y);
  int RightControl = master.get_analog(ANALOG_RIGHT_Y);
  **/

  /** - arcade drive variables -**/
  int power = master.get_analog(ANALOG_RIGHT_Y); //right joystick Y drive straight
  int turn = master.get_analog(ANALOG_RIGHT_X); //right joystick X turn
  int shift = master.get_analog(ANALOG_LEFT_X); //left joystick X shift left/right

  int LeftFrontControl = - power + turn + shift;
  int RightFrontControl = - power - turn - shift;
  int LeftBackControl = - power + turn - shift;
  int RightBackControl = - power - turn + shift;


  while (true){

    /** - tank drive -
    //get joysticks position value and maps them to a variable
    LeftControl = master.get_analog(ANALOG_LEFT_Y);
    RightControl = master.get_analog(ANALOG_RIGHT_Y);
```

```
    FLMotor.move(LeftControl);
    BLMotor.move(LeftControl);
    FRMotor.move(RightControl);
    BRMotor.move(RightControl);
    pros::delay(2);
    **/

    /* arcade drive*/
    int power = master.get_analog(ANALOG_RIGHT_Y); //right joystick Y drive straight
    int turn = master.get_analog(ANALOG_RIGHT_X); //right joystick X turn
    int shift = master.get_analog(ANALOG_LEFT_X); //left joystick X shift left/right

    int LeftFrontControl = - power + turn - shift;
    int RightFrontControl = - power - turn + shift;
    int LeftBackControl = - power + turn + shift;
    int RightBackControl = - power - turn - shift;

    FLMotor.move(LeftFrontControl);
    BLMotor.move(LeftBackControl);
    FRMotor.move(RightFrontControl);
    BRMotor.move(RightBackControl);

    pros::delay(2);

  }
}
```

Shooter.cpp

```
#include "main.h"


void ShooterPiston(){
  piston.set_value(true);
  pros::delay(200);
  piston.set_value(false);
}

void Launcher(){
    cannonA.set_value(true);
    cannonB.set_value(true);
    pros::delay(200);
    cannonA.set_value(false);
    cannonB.set_value(false);
    //pros::delay(20);
}


void ShooterMotor(double distance, int speed){
    ShooterFMotor.move_relative(distance, speed);
    ShooterBMotor.move_relative(distance, speed);
}

void ShooterVel(int power) {
  ShooterFMotor.move_voltage(power);
  ShooterBMotor.move_voltage(power);
}

//thread for all shooter controls
```

```
void Shooter_fn(void* param) {

  ShooterFMotor.set_brake_mode(pros::E_MOTOR_BRAKE_BRAKE);
  ShooterBMotor.set_brake_mode(pros::E_MOTOR_BRAKE_BRAKE);
  int counter_L1 = 0;
  int counter_L2 = 0;
  while
   (true) {

    if (master.get_digital(DIGITAL_L1) && counter_L1 == 0) {
        ShooterFMotor.move_voltage(6500);
        ShooterBMotor.move_voltage(6500);
        pros::delay(1000);
        counter_L1 = 1;
        counter_L2 = 0;
    }
    else if (master.get_digital(DIGITAL_L1) && counter_L1 == 1){
        ShooterFMotor.move_voltage(0);
        ShooterBMotor.move_voltage(0);
        pros::delay(1000);
        counter_L1 = 0;
        counter_L2 = 0;
    }
    else if (master.get_digital(DIGITAL_L2) && counter_L2 == 0) {
        ShooterFMotor.move_voltage(8000);
        ShooterBMotor.move_voltage(8000);
        pros::delay(1000);
        counter_L1 = 0;
        counter_L2 = 1;
    }
    else if (master.get_digital(DIGITAL_L2) && counter_L2 == 1){
```

```
      ShooterFMotor.move_voltage(0);
      ShooterBMotor.move_voltage(0);
      pros::delay(1000);
      counter_L1 = 0;
      counter_L2 = 0;
    }

    if (master.get_digital(DIGITAL_DOWN)) { //Shooter mech
      ShooterPiston();
      //pros::delay(20);
    }

    if (master.get_digital(DIGITAL_A) && master.get_digital(DIGITAL_B)) { //Launcher
mech
      Launcher();
    //pros::delay(20);
  }
  }
}
```

Autons.cpp

```
#include "main.h"

/**Shoot 2 discs and change roller color**/
void LeftOne(){

   ShooterVel(8000);
   pros::delay(5000);
   ShooterPiston();
   pros::delay(1900);
   ShooterPiston();
   pros::delay(1500);
   ShooterVel(0);

   Turn(50, 100);
      do {
         pros::delay(20);
      } while (!AtDistanceDriveGoal(5));

   pros::delay(1000);

   Shift(50, 100); //Can remove shift step once 3rd wheel taken out
      do {
         pros::delay(20);
      } while (!AtDistanceDriveGoal(5));

   pros::delay(500);

   CollectorVel(-8000);
   Drive(50, 50);
   pros::delay(500);
```

```
    CollectorVel(0);
    pros::delay(500);

    Drive(-150, 100);
    CollectorVel(0);
    pros::delay(100);

    Shift(-500, 100);

}

/**Shoot 2 discs and change roller color + shoot 3 discs in middle of field**/
void LeftTwo(){



}

/**comment**/
void RightOne() {
}

/**comment**/
void RightTwo() {
}

/**comment**/
void Shell(){
}
```

```
/**varirables and functions array for auton selector**/
int autonselector = 0;
//define and initialize string array "titles[]" as constant
const char *titles[] = {"LeftOne   ", "LeftTwo   ", "RightOne  ", "RightTwo  ", "Shell     ",
"SkillsOne ", "SkillsTwo "};

//define and initialize an array of function pointers for all auton functions
void (*scripts[])() = {&LeftOne, &LeftTwo, &RightOne, &RightTwo, &Shell,
&SkillsAutonOne, &SkillsAutonTwo};

//define auton script runner function - run the selected auton script through on screen
"autonselector"
void LCDScriptExecute() {scripts[autonselector]();}
```

Parts List

We did it!