99909A

Rising Phoenix

Capital Robotics Club

12/15/2025

Our Team!

Team 99909A Photo:

Left to Right: Zoe, Sammy, Vaishali, Vaishnavi

Get to know 99909A!

Sammy: team co-captain, programmer, and driver, currently a Junior at TJHSST

Zoe: team manager and driver, currently a Junior at Langley High School

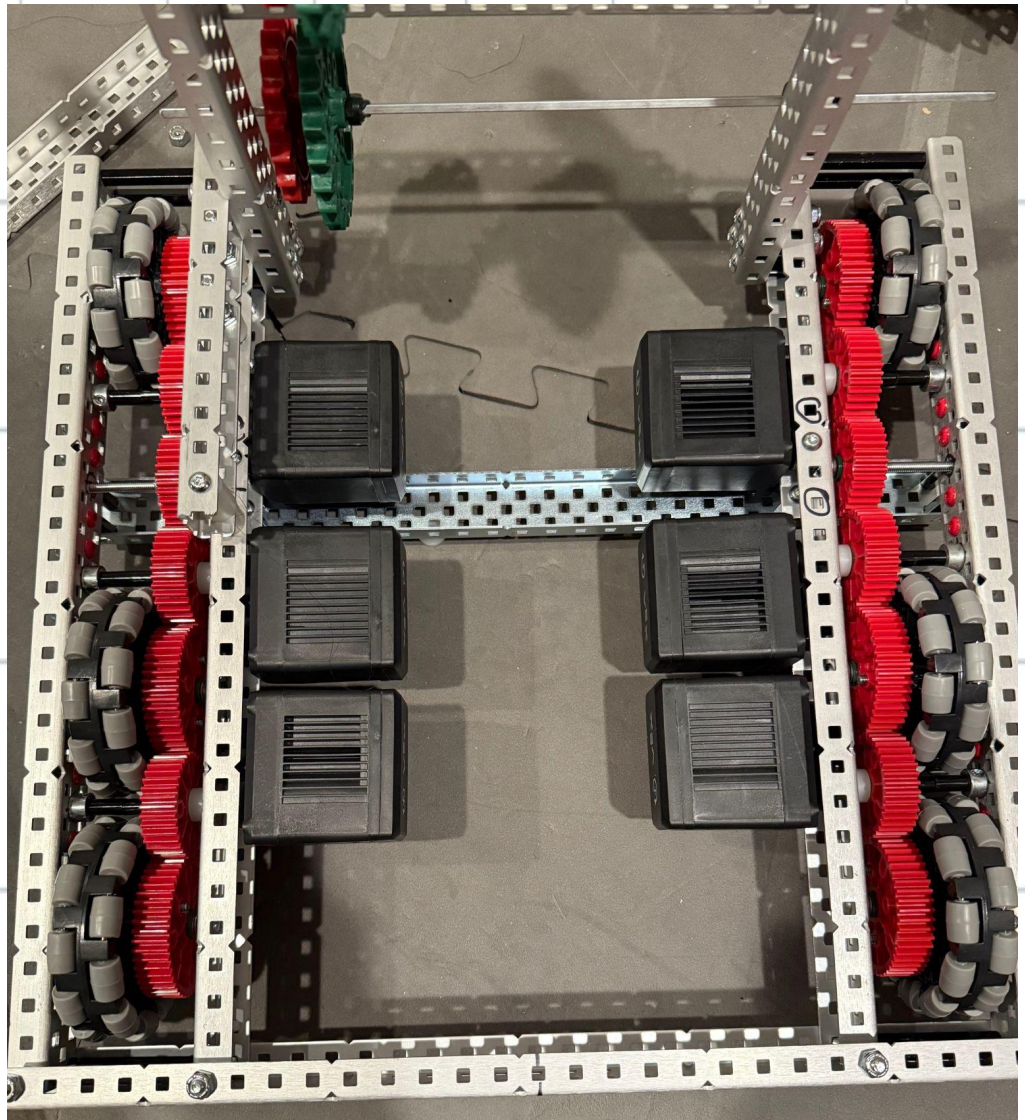
Vaishali: team liaison, currently a Junior at the Madeira School

Vaishnavi: robot builder, currently a Junior at the Madeira School

Drivetrain

6-Motor Drivetrain for Speed

- Our six-motor drivetrain is specifically designed for increased speed, aligning the fast-paced demands of this year's game.
- By dedicating many motors to the drivetrain, we gain a strong edge in acceleration, allowing us to reach scoring zones before our competitors and capitalize on every opportunity.



Technical Advantages

- With boosted motor power, our drivetrain produces additional torque, making movement across the field not just faster but smoother, giving us a real edge in agility.
- Our enhanced control lets us maneuver precisely in high-traffic areas, helping us avoid getting pinned or blocked during critical plays.

6-Motor Drivetrain for Speed PT.2

- This setup boosts agility by giving us more control over rapid direction changes and allowing us to respond effectively to in-game situations, always staying fluid and responsive.
- Speed is critical to our strategy because it allows us to cover more ground quickly, maximizing our offensive play possibilities.

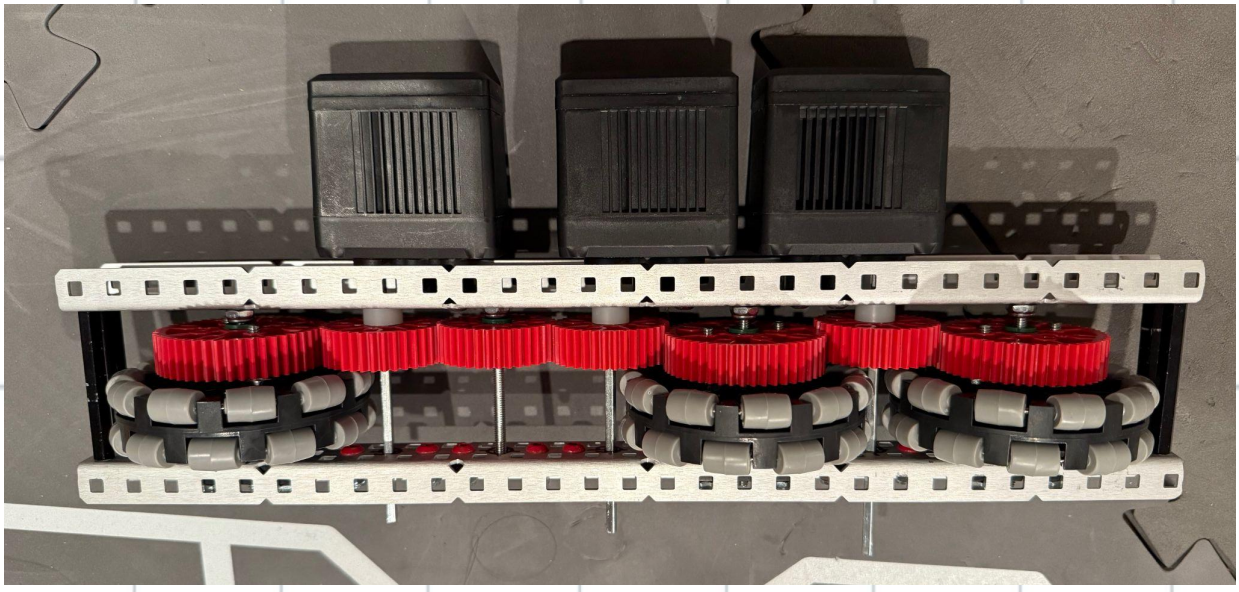
Technical Advantages

- Reduced response time keeps us nimble; split-second decisions are crucial for sidestepping obstacles and smoothly navigating around other robots.
- The combination of speed and agility makes our robot highly adaptable, letting us switch seamlessly between offensive and defensive roles as the game demands.

Gear-linked wheels

In the past, we had used chain linked wheels. All the drivetrain wheels are linked through a set of chains. The issue with that setup is:

- Need 2 sets of chain to connect 3 wheels on each shoe (front-middle and moddle-back). This takes up a lot of spaces and makes the drivetrain show very bulky.
- There is always some slack in the chain so not all the wheels will move at the same time. This can cause issue with autonomous programs.



Strategy considered

- To maximize drivetrain performance, we assigned six motors, which reduced available power for other subsystems.
- This choice called for careful trade-offs in weight and power distribution, with adjustments to ensure the robot remains stable at high speeds.

Strategy considered Pt.2

- Focusing on speed led us to streamline certain mechanisms, trading a bit of versatility for pure offensive efficiency.
- These sacrifices were intentional: prioritizing scoring potential over defensive features aligns directly with our strategy for this season, giving us a competitive edge where it matters most.

Final drivetrain setup

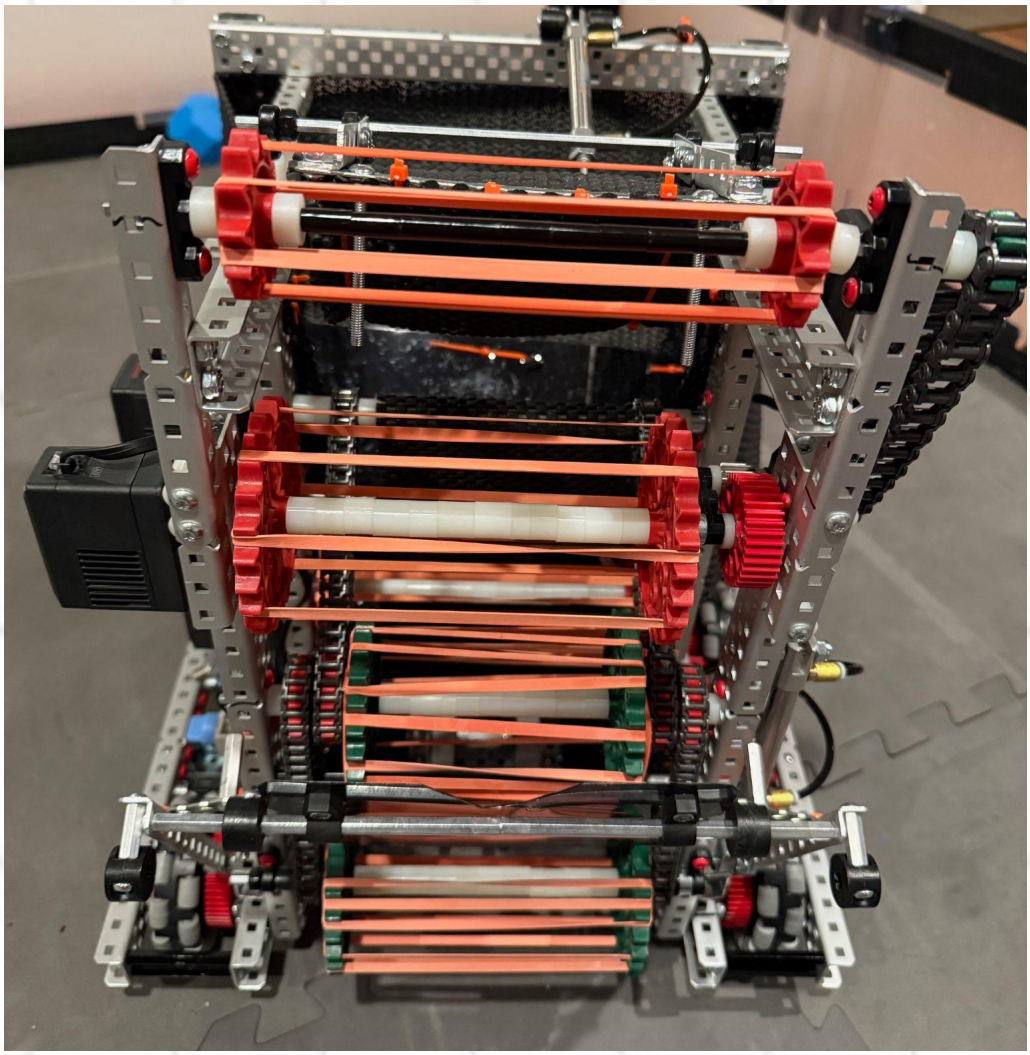
- 3 motors with 600 rpm cartridges (blue)
- Motor are attached to 36-tooth gear
- Omni-wheels are attached to 48-tooth gears
- With the combination of 600 RPM cartridge and 36:48 gear ratio, we are achieving 450 rpm driving speed.

Block Collector

Problems Regarding the Block collector

Block Persistence and accuracy:

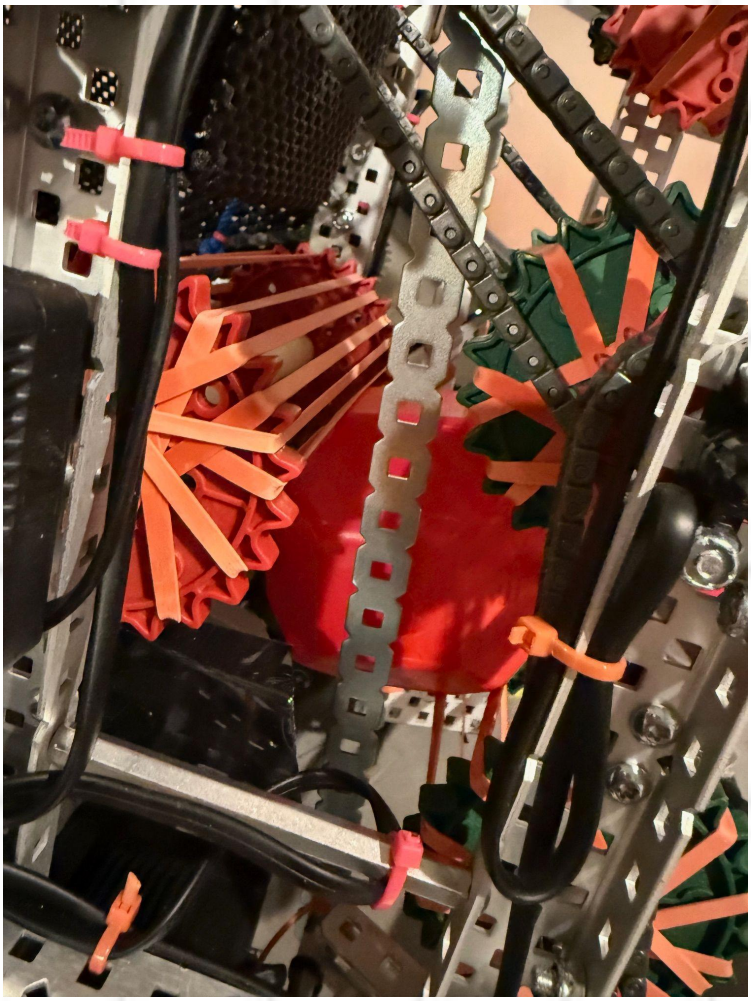
The first and foremost problem that caught our attention while testing the block collector included it not being able to go up the rollers fully and at a 100% accuracy rate.



Problems Regarding the Block collector

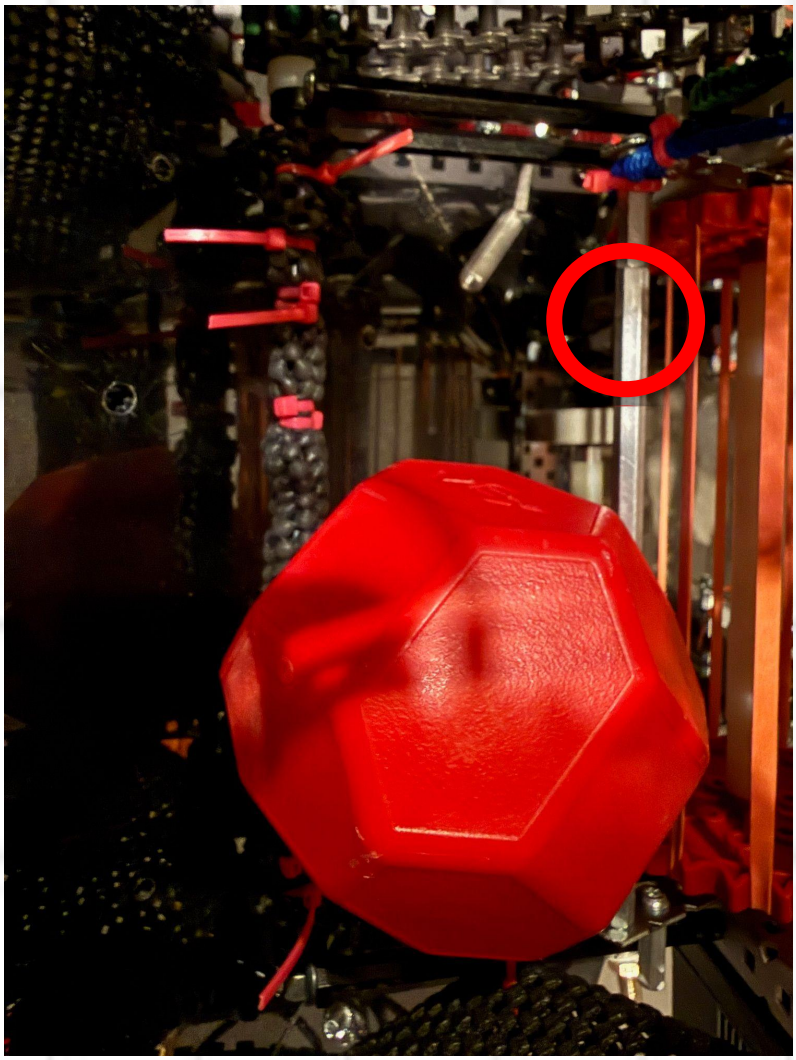
Block Persistence and Accuracy:

If the robot were to be driven with this "stutter-10-second-problem", we would be missing another 50 seconds in the entire match, which could cost us multiple points throughout the competition.



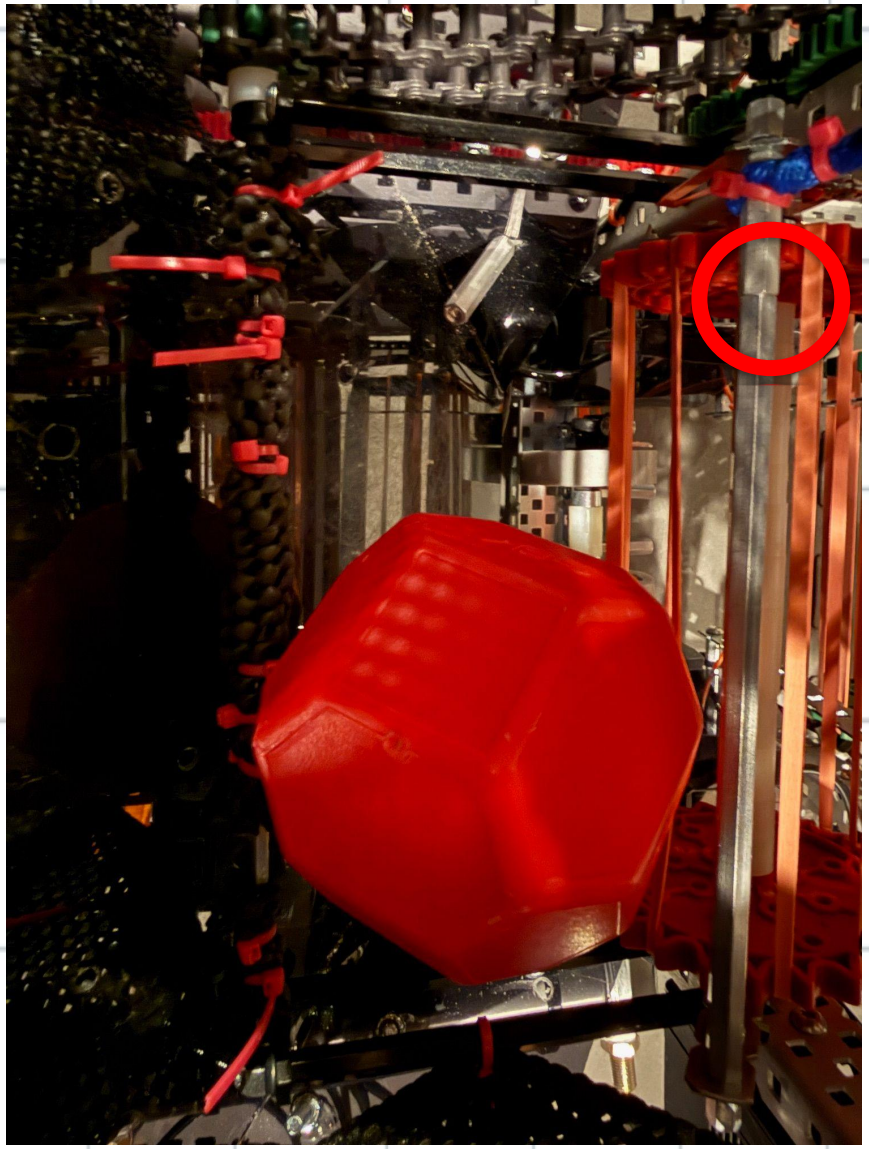
Solutions to Problem #1:

After analyzing the problem by filming what was happening in slow motion via iphone, we noticed that the block got stuck because the bin motor is not spinning to lift it up.
So we installed a stop bar in the bin such that blocks in the bin would rest on it, not touching the roller. As such, we can spin the bin roller that will help moving the block up through the channel.



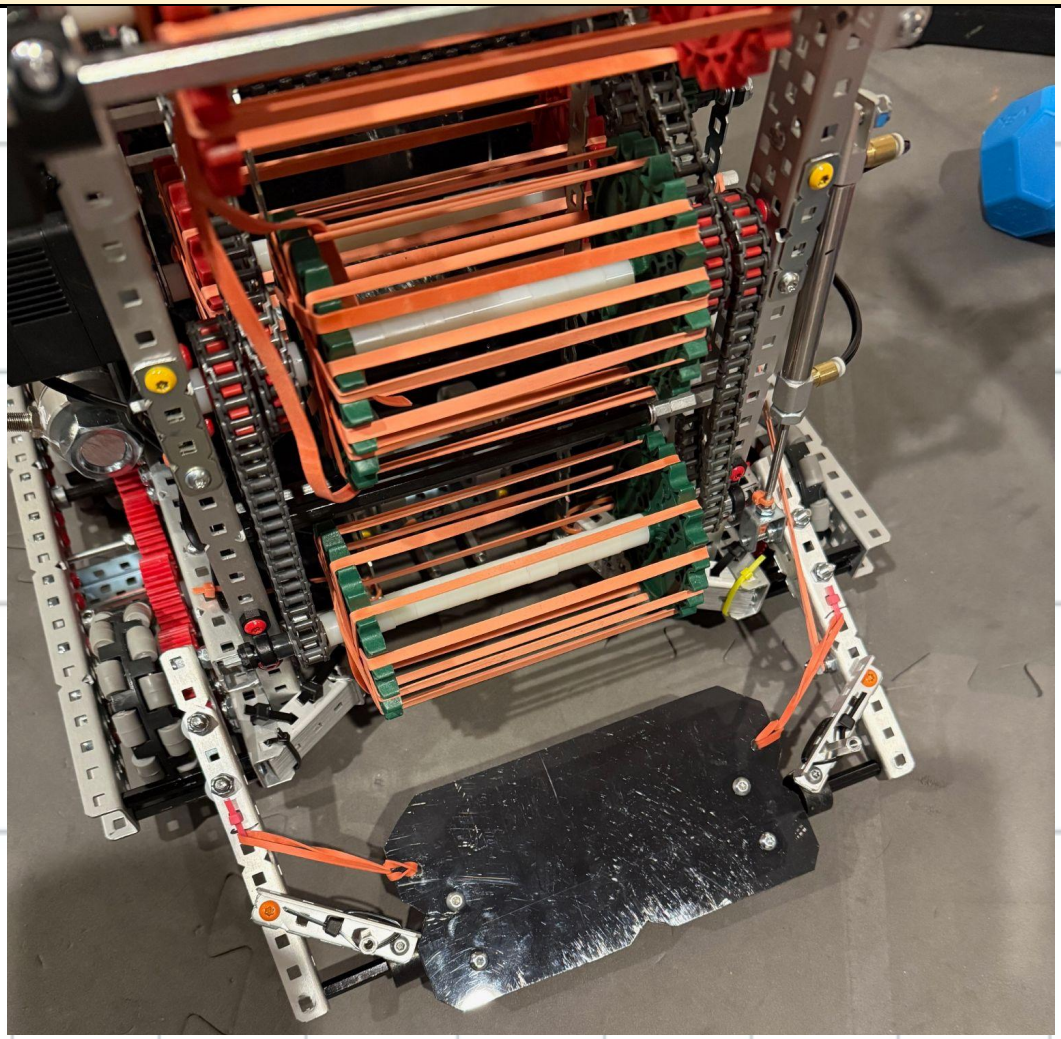
Solutions to Problem #1:

When we are done with collecting blocks and ready deposit the blocks in the goals, a mechanism attached to the top flap will pull the stopper up, allowing the blocks to make contact with the roller rubber bands.



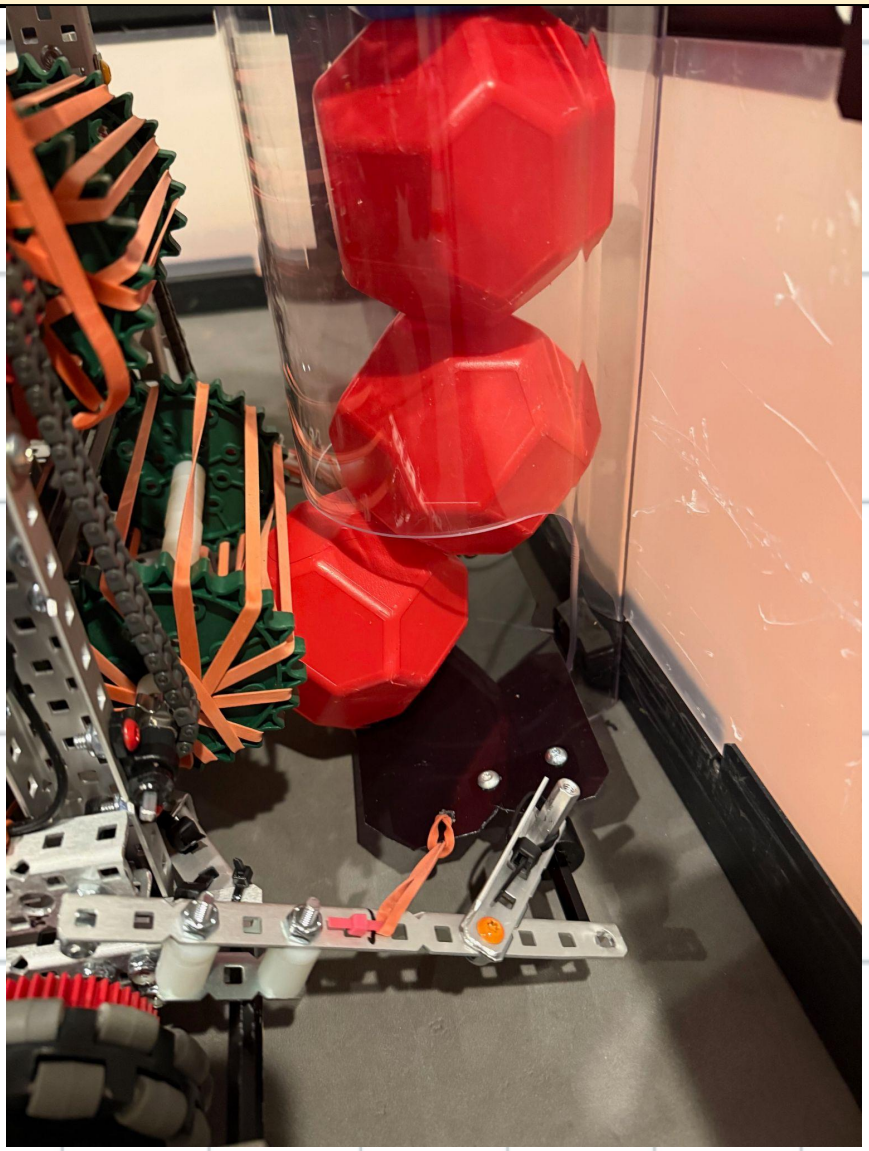
Loader Ramp

The loader tube holds 6 blocks of mixed alliance colors. It's very tricky to get the blocks out of the loader consistently. After trial and error, we decided on this ramp design



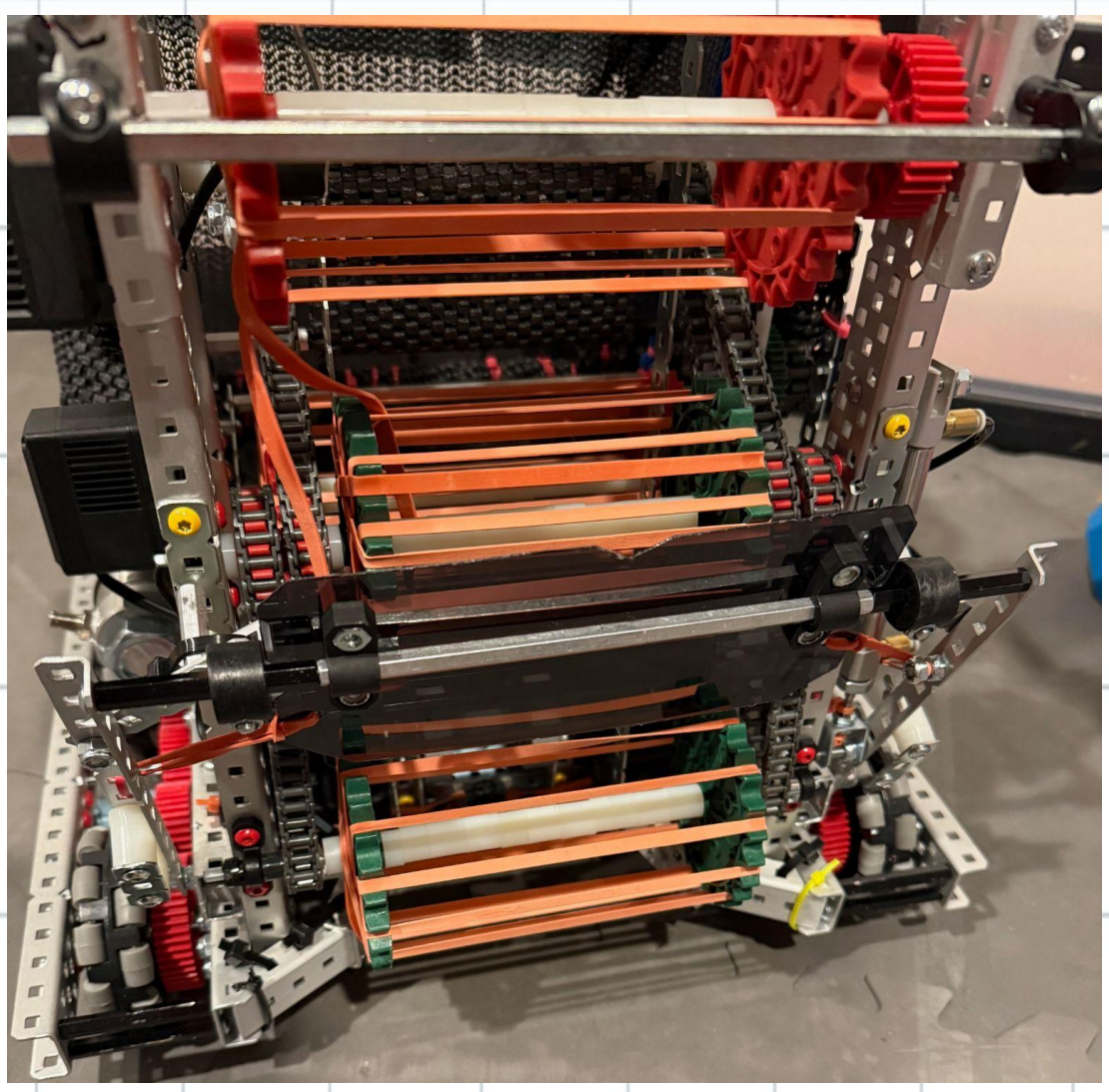
Loader Ramp

When extended and driven against the back of the tube, the ramp would tilt up a bit and forming a downslope ramp that allows blocks to roll down one by one.

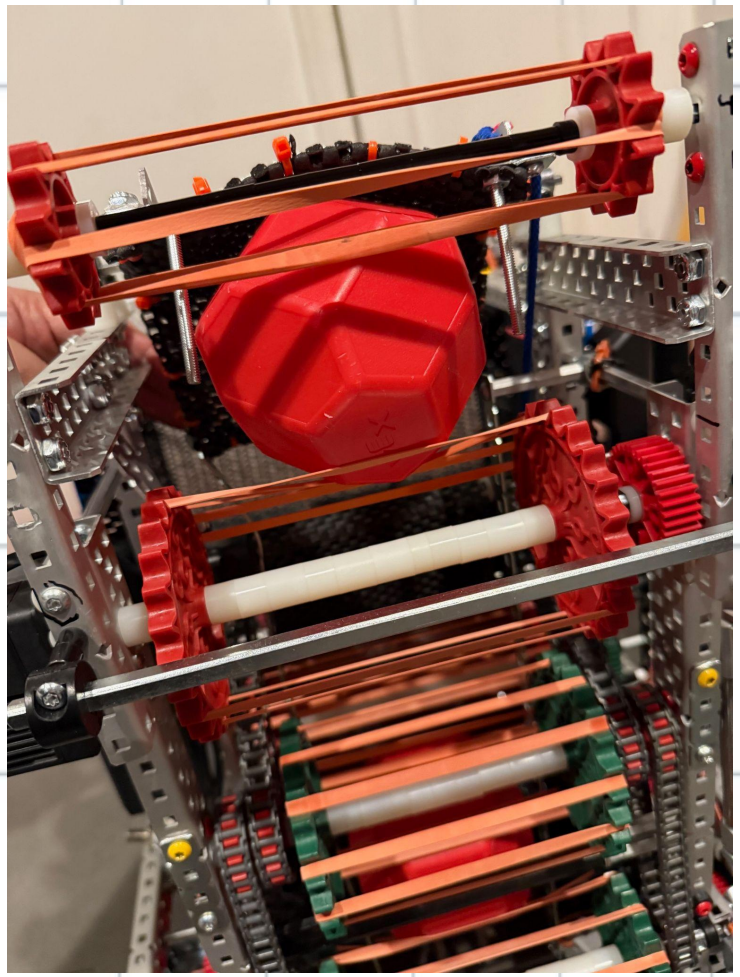


Loader Ramp

When not in use, the ramp is controlled by a piston and would retract and make space for depositing blocks into goals.

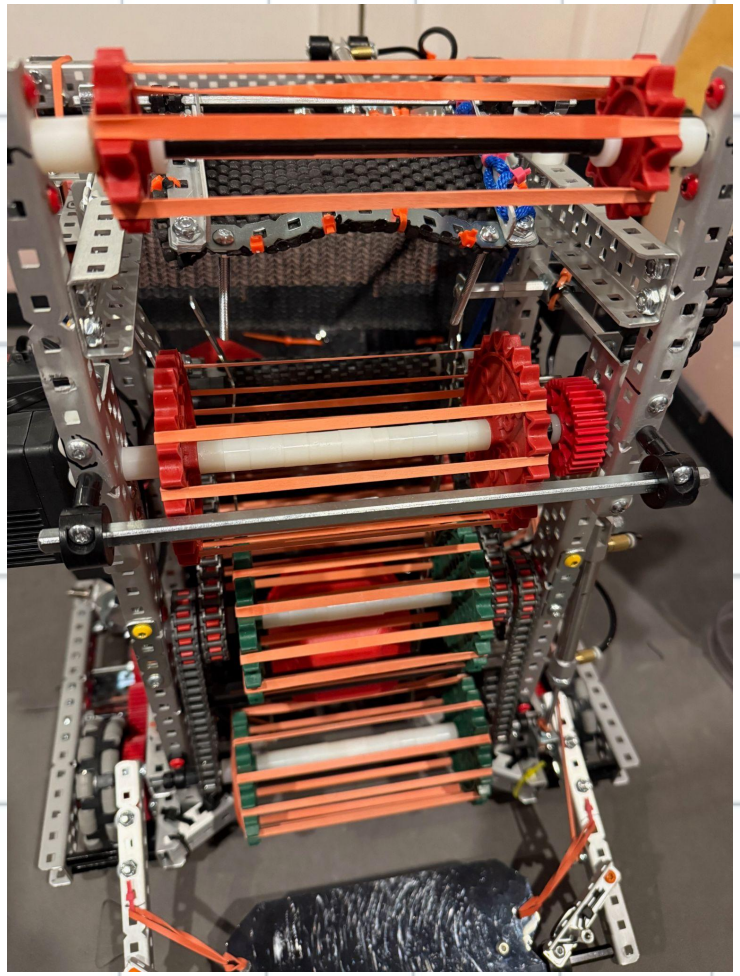


High Goals



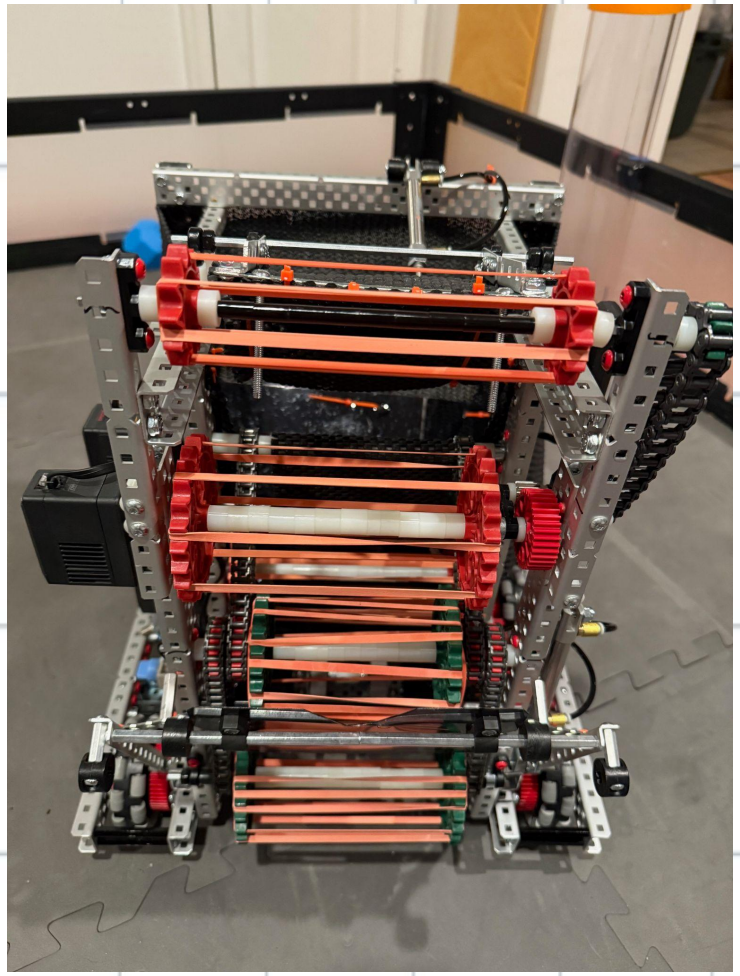
The blocks can be deposited into high goals by a pair of rollers on the top of the robot. A swinging gate would tilt up to allow blocks to move between the pair of top rollers and squeezed out onto high goals.

High Goals



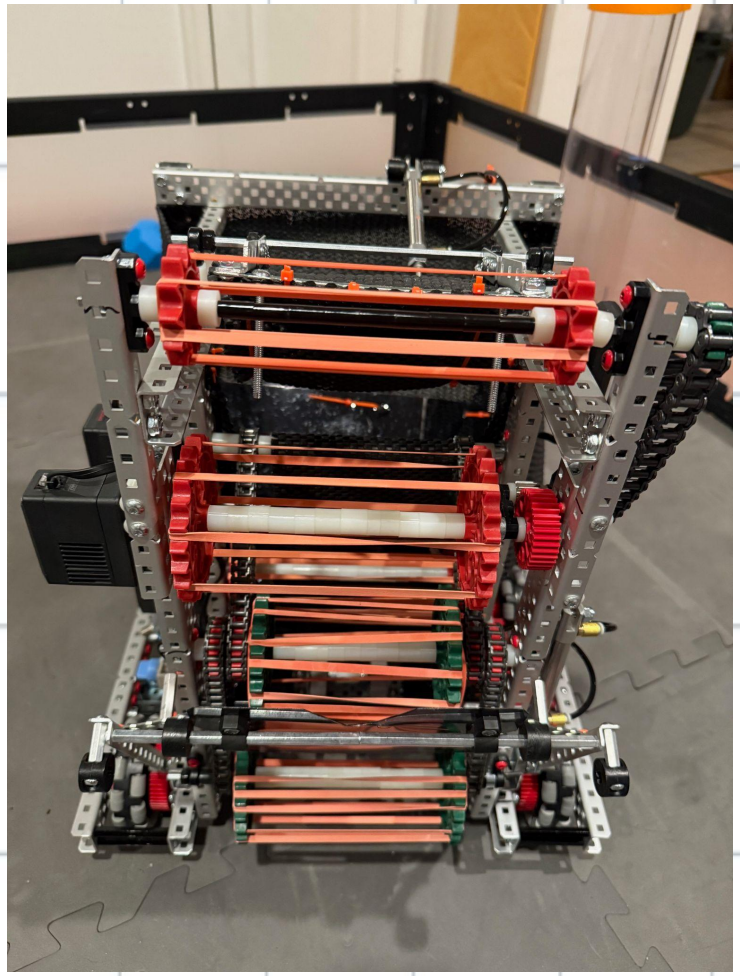
After finishing depositing the blocks, the swinging gate would close and allow the blocks be deposited into the bin. The swinging gate is controlled by a piston attached to the back of the bin.

Mid Goal



To deposit blocks into the mid-level goal, we reverse the top roller. All the blocks passing through the passage will be forced out of the middle pair of rollers onto the mid-level goals.

Low Goal

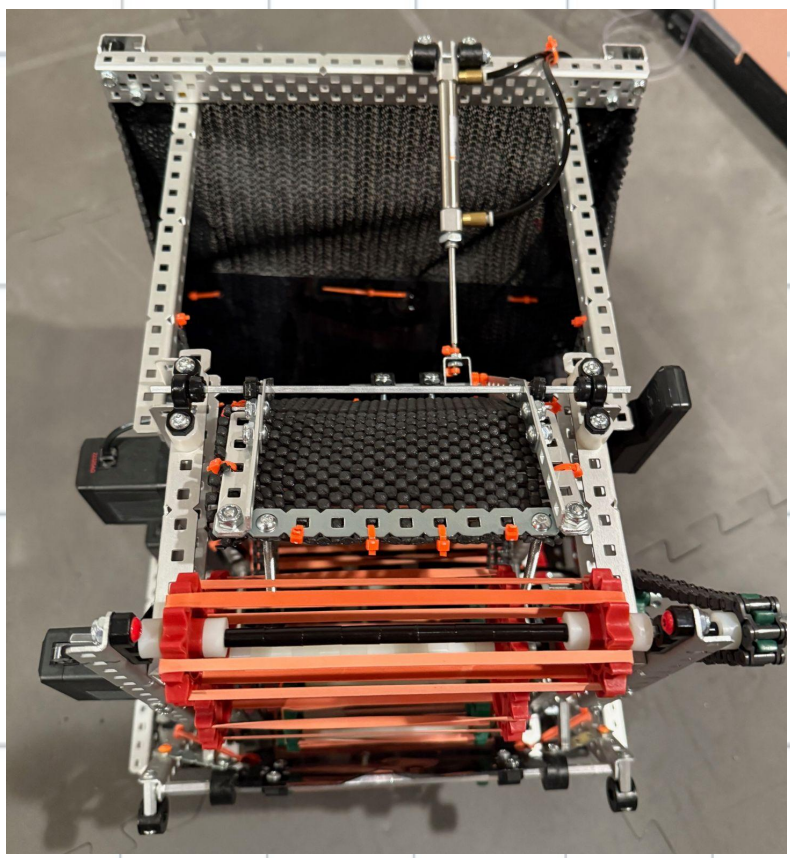


To deposit blocks into the low-level goal, we reverse the collector roller. Now all the blocks coming out of the bin would be pushed out from the bottom onto the low-level goal.

Swinging Gate

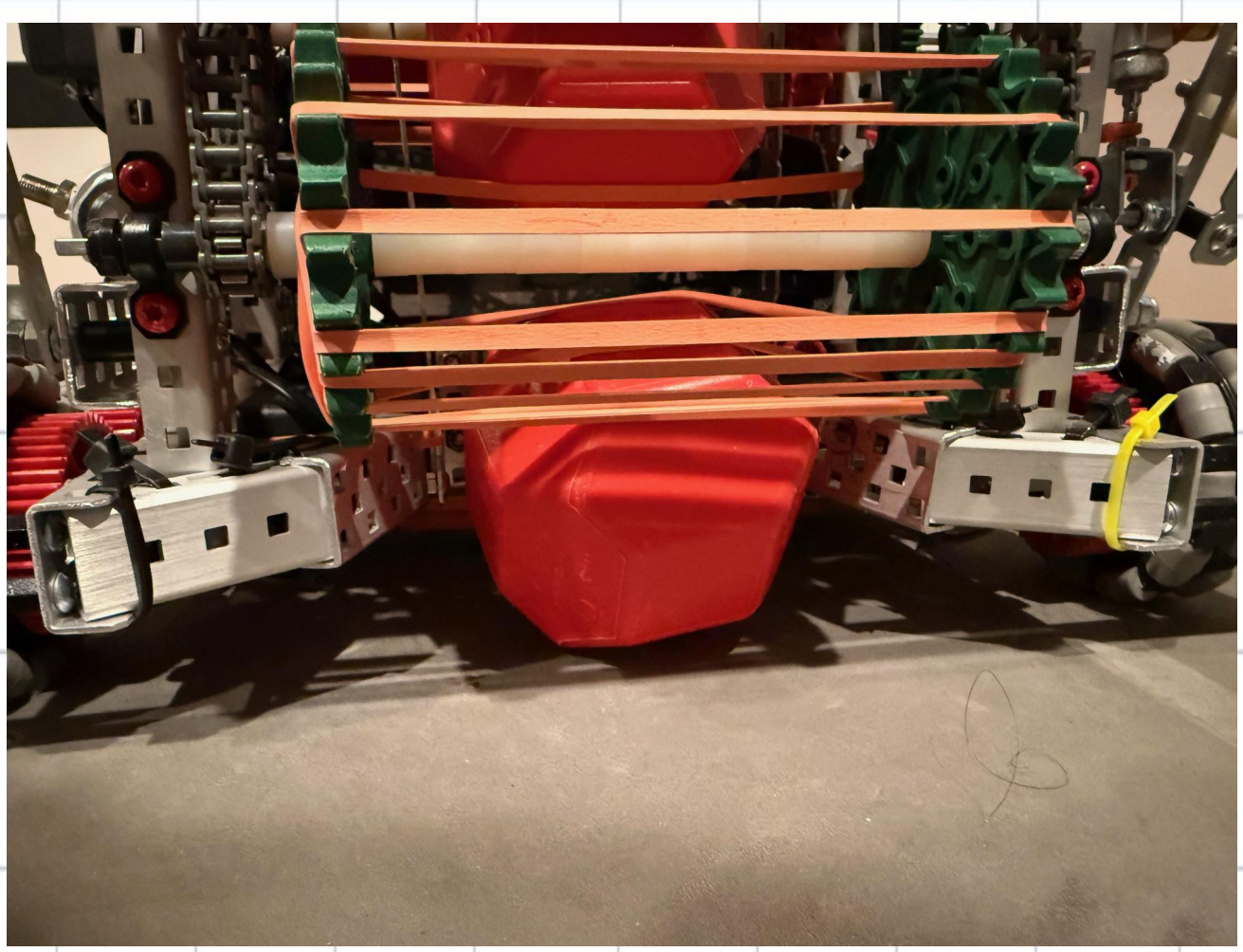
A challenge we have run into however, is that the blocks sometime get stuck between the gate and edge of the bin

we have to adjust the gate size and shape to achieve the perfect operations between open (allow blocks to go out to high goal) and close (alllow blocks to get into bin)



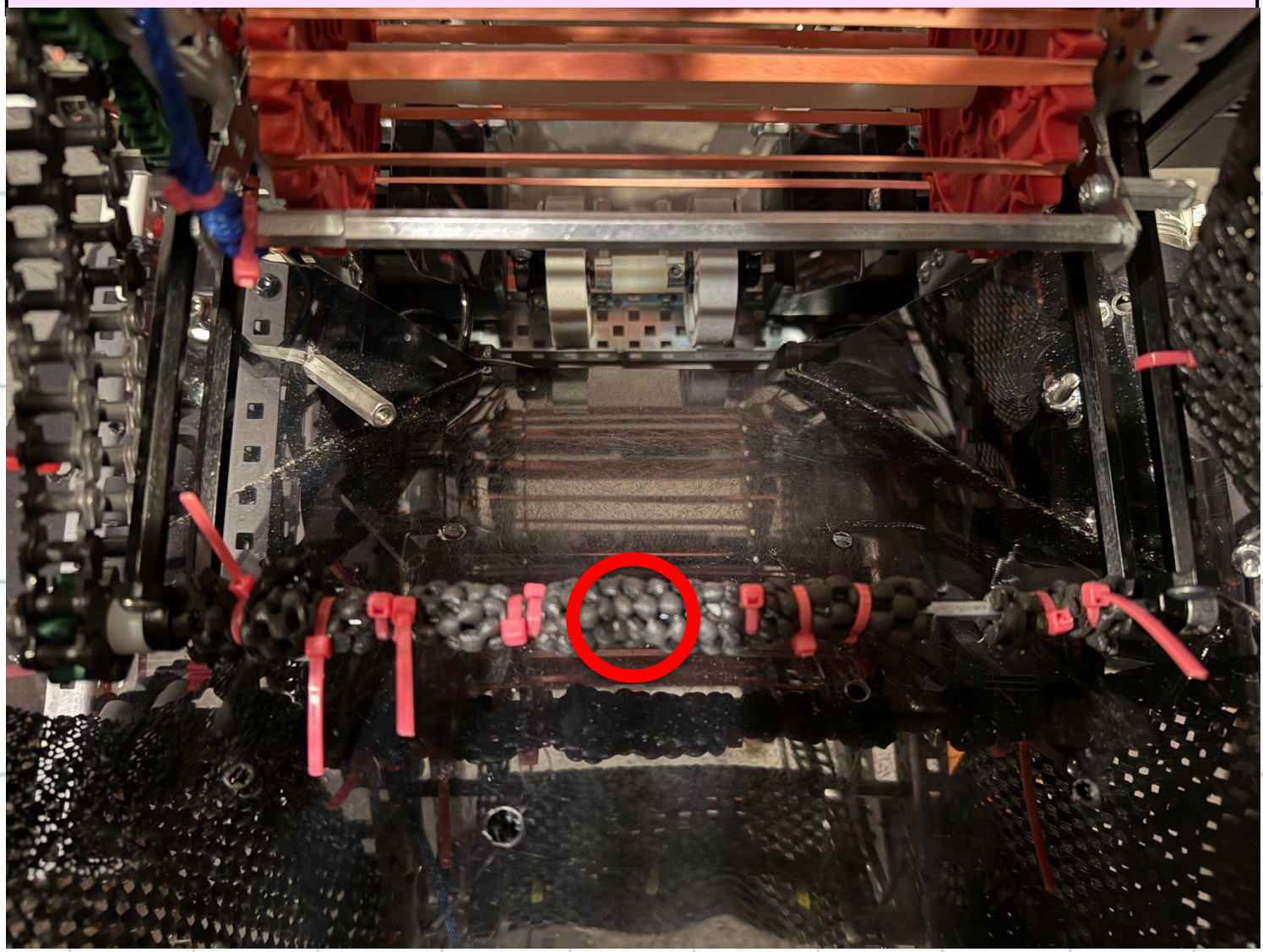
Front Funnel

Another challenge we have run into is that the 2 blocks sometime get stuck in the intake. We built a funnel that will only allow 1 block through at a time.



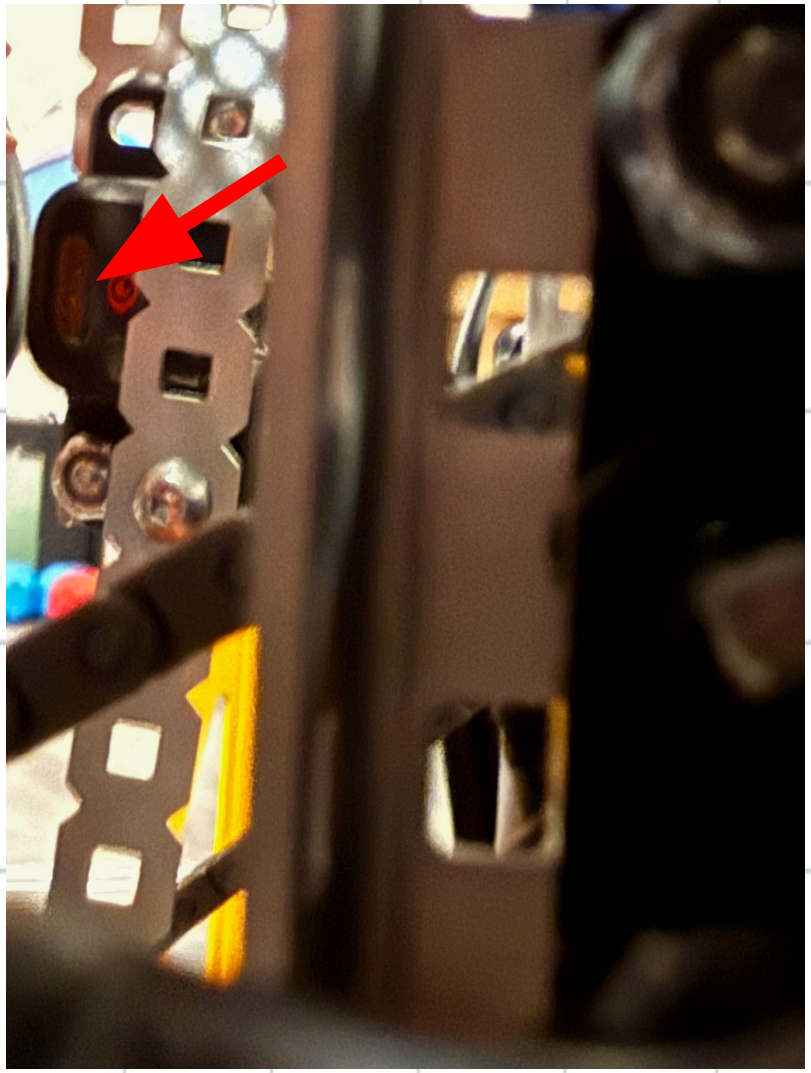
Bin Stir Bar

Sometimes, too many blacks in the bin will cause them to be congested and not coming out. We thought a way to attach a stir bar to the bin so that the blocks would always be stirred while in the bin.



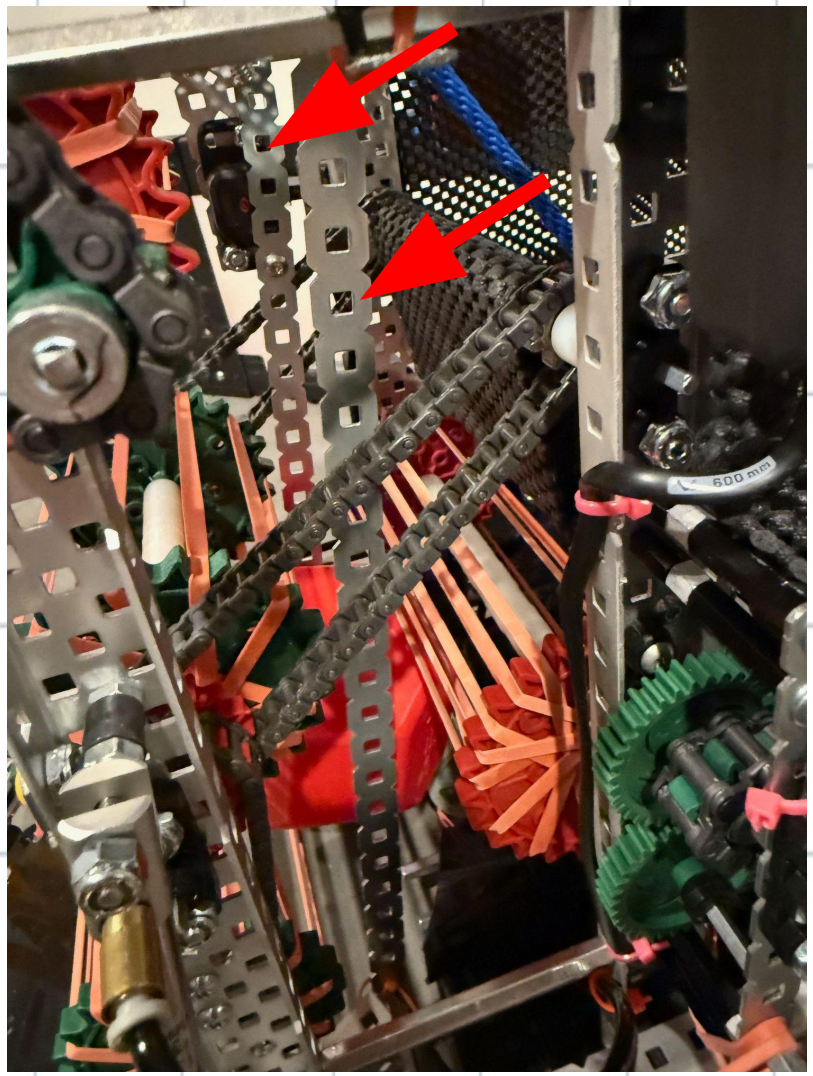
Alliance Block

we use a color sensor to detect the color of blocks in the funnel. The initial preload block set the alliance color. Any block with different color will trigger the top roller to reverse spinning and push out



Block channel Guard

The roller is quite wide. As a result, the blocks can bundle together and clog the channel. We installed a pair of guard to align blocks in the direction of moving. It works quite well. They also help color sensor too.



Driving Strategy

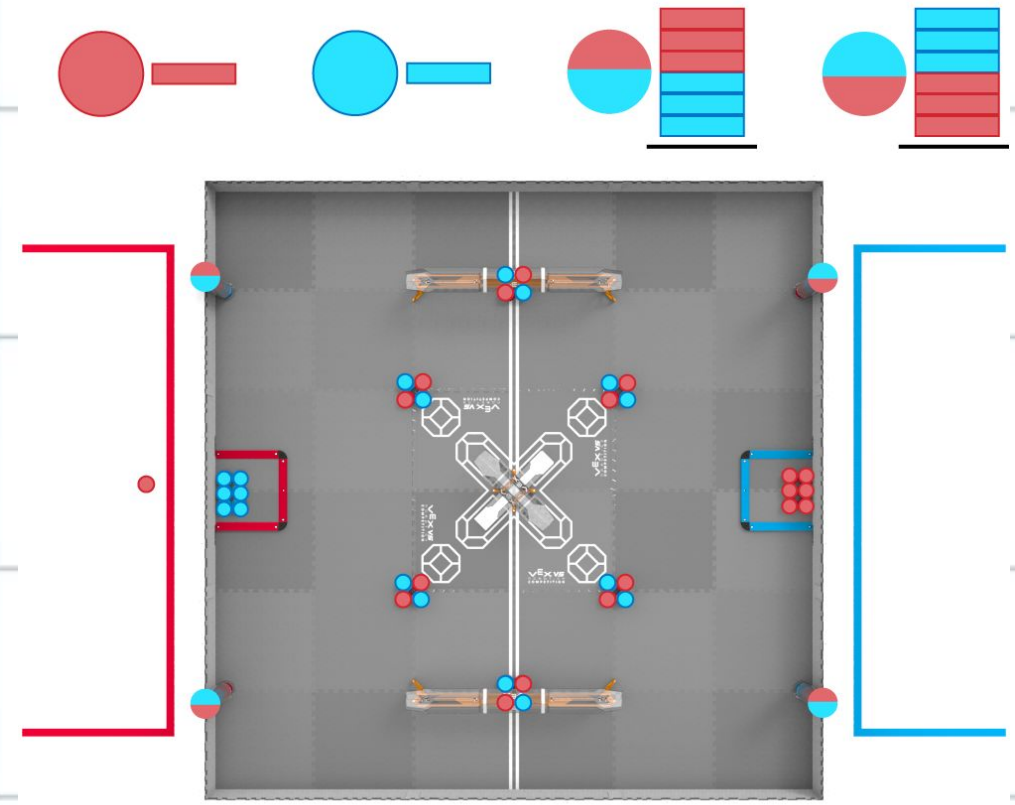
Our Driving Strategy - Part 1 (Skills)

Our driving strategy has to be very concise, time efficient, and able to get as many points as possible in the limited time we have available. Being able to travel with a fast robot is a very efficient factor, however controlling our speed, being able to use it to our advantage, and utilizing while collecting and scoring blocks in the last seconds is our specialty.

Our Driving Strategy - Part 2 (Skills)

Our Attack:

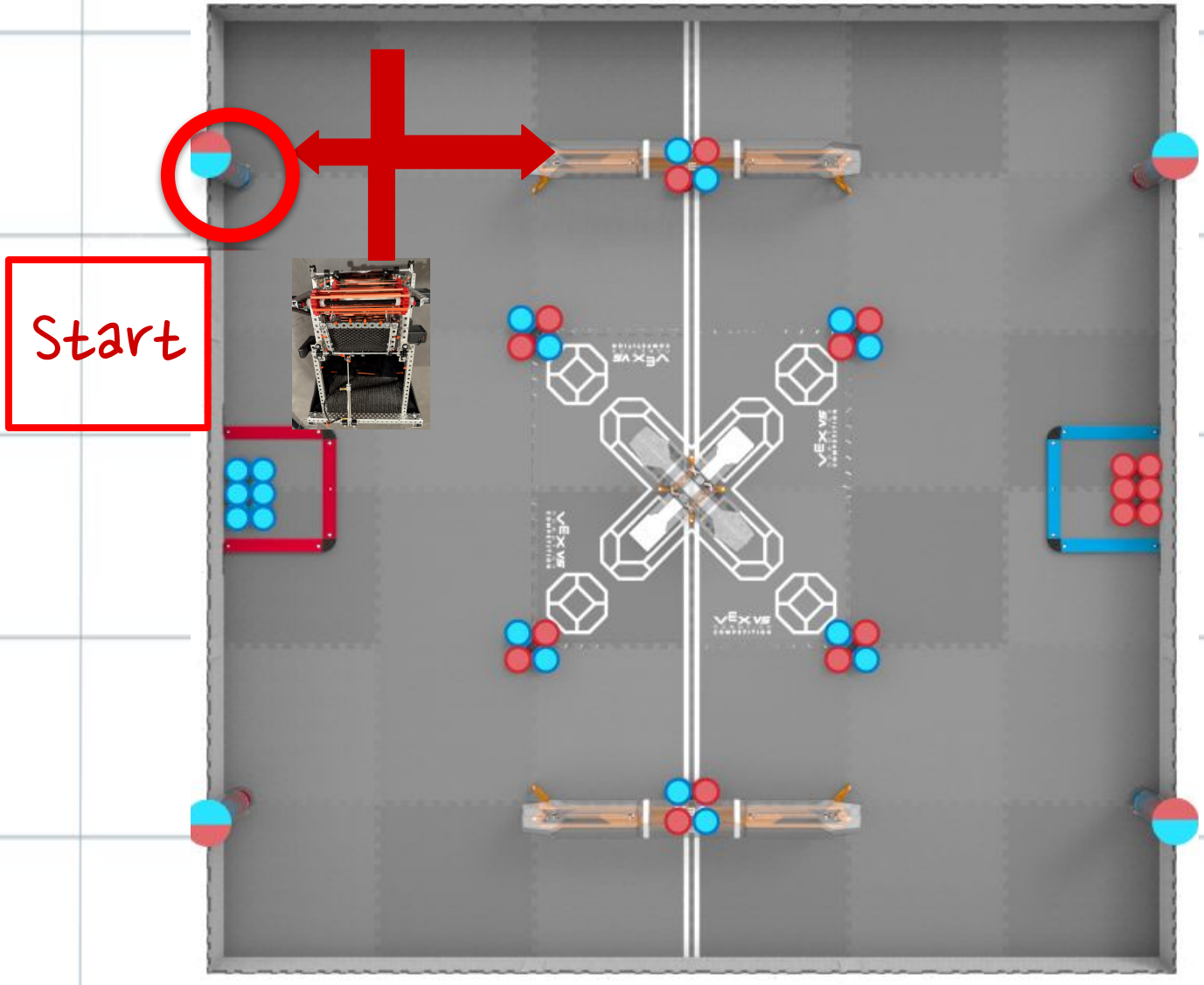
As we tried to approach the board for the first time, we considered doing the entire board at once, however we noticed a pattern in the block setup between the left and right sides of the board.



Audience View

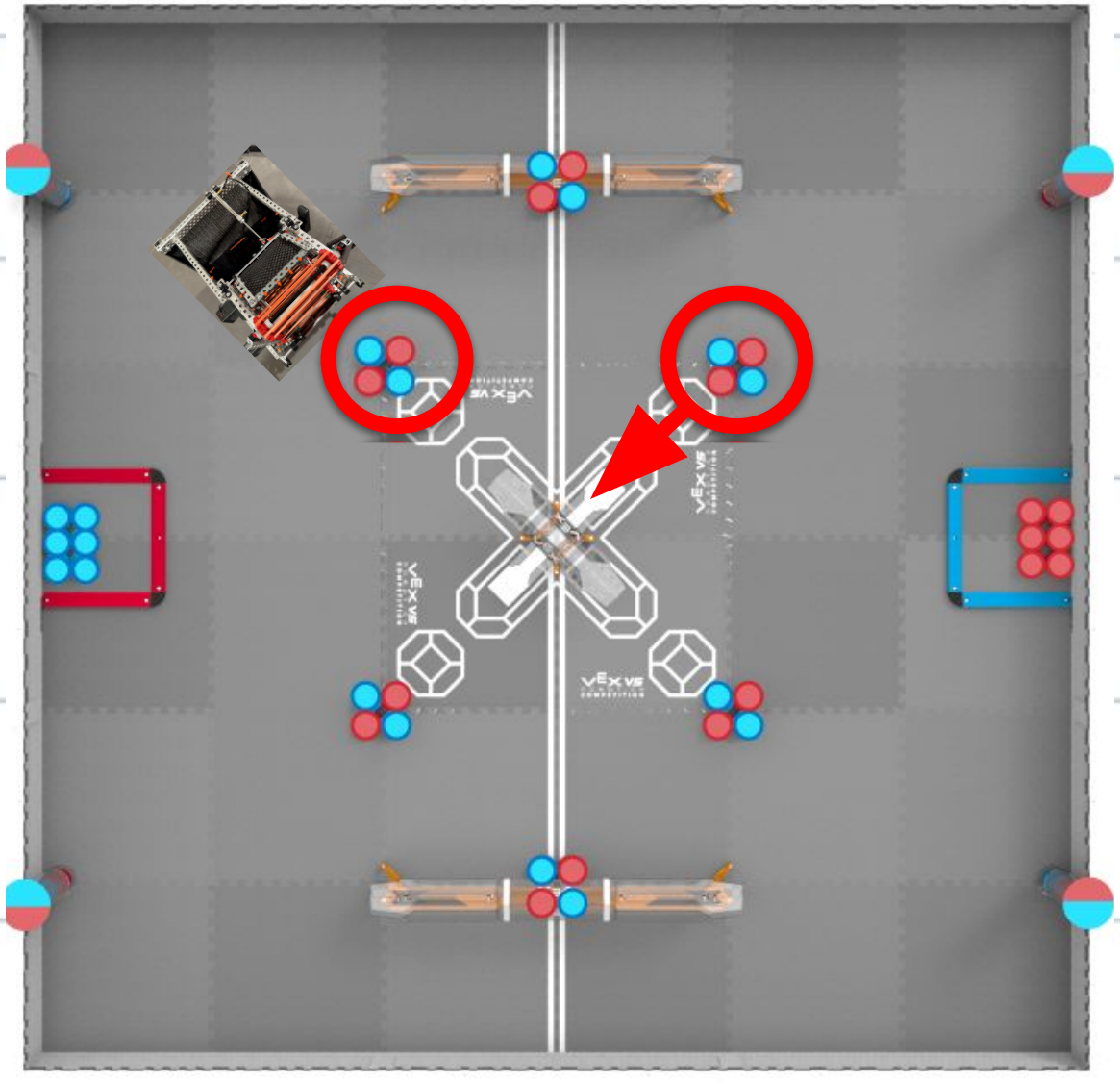
Attack #1

Our first attack consists of splitting the board into two parts, top and bottom. We set up robot starting on top of the red park zone. We go for the top left loader and then turn around to deposit blocks into top high goal



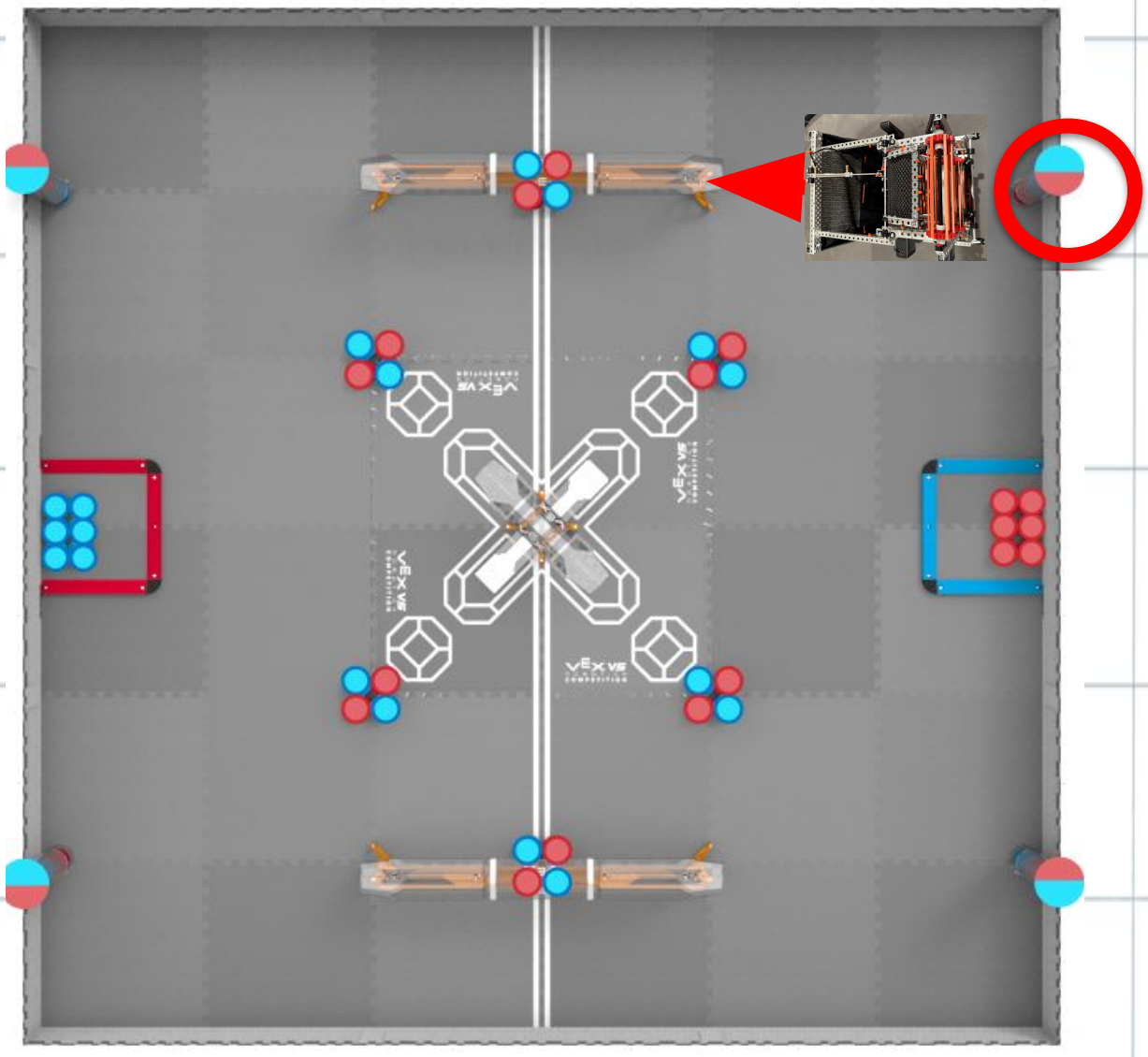
Attack #2

The we turn robot 45 degree to get the 4 blocks on the ground but we don't deposit. we straighten out robot and collect the 4 blocks on the right. We then deposit the 8 blocks onto the bottom goal.



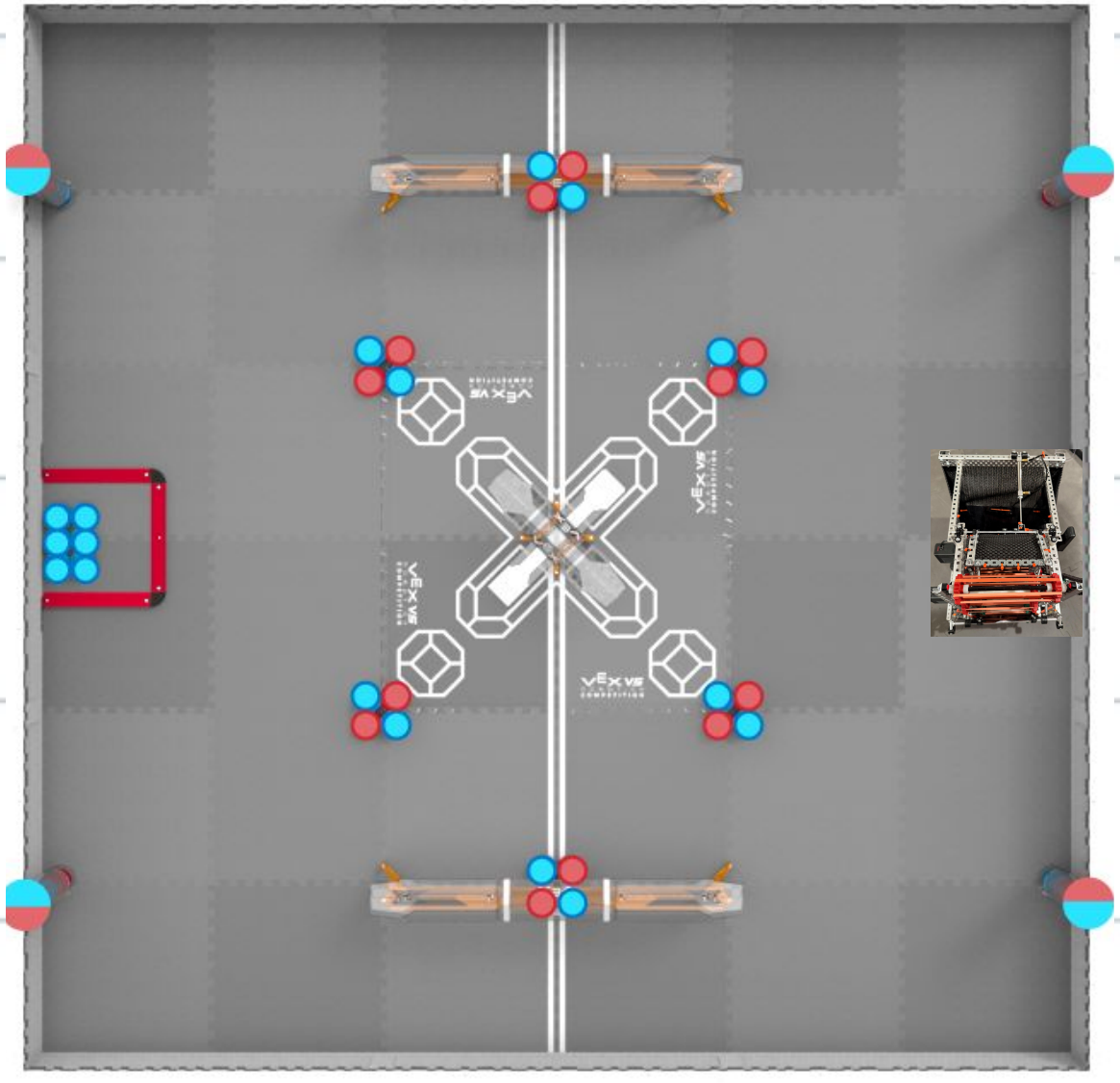
Attack #3

we align our robot on the wall then go and collect blocks from the top right loader. Then we deposit them on the other end of the high goal



Attack #4

Finally, we park the robot on the blue park zone, while pushing out all the blocks there



Programming

Block Collector:

```
#include "main.h"


void CollectorOn(){
FrontRollerMotor.move_voltage(12000); //max voltage 12000
TopRollerMotor.move_voltage(12000); //max voltage 12000
BinRollerMotor.move_voltage(-8000); //max voltage 12000
//pros::delay(20);
toppiston.set_value(false);
}

void CollectorOff(){
pros::delay(20);
FrontRollerMotor.move_voltage(0); //max voltage 12000
TopRollerMotor.move_voltage(0); //max voltage 12000
BinRollerMotor.move_voltage(0); //max voltage 12000
}


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

FrontRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
TopRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);

while (true) {

if (master.get_digital(DIGITAL_R1)) {
CollectorOn();
collector_on = true;
}
else if (master.get_digital(DIGITAL_R2)) {
//CollectorMotorA.move_voltage(-12000); //max voltage 12000
//CollectorMotorB.move_voltage(-12000); //max voltage 12000
CollectorOff();
collector_on = false;
}

//assign ball color
if (OpticalSensor.get_proximity() == 255)
{
hue_value = OpticalSensor.get_hue();

//set alliance ball color
if (flag_set == false)
{
if (hue_value < 120 || hue_value > 330){
//set alliance ball color to red
alliance_ball = (char*)"red";
}
else{
//set alliance ball color to blue
alliance_ball = (char*)"blue";
}
flag_set = true;
current_ball = alliance_ball;
//}
master.print(0, 0, "alliance ball: %s", alliance_ball);
}
//set current ball color
if (hue_value < 120 || hue_value > 330){
//set current ball color to red
current_ball = (char*)"red";
}
else{
//set current ball color to blue
current_ball = (char*)"blue";
}
}

//test ball color
if (collector_on == true) {
if (current_ball == alliance_ball) {
TopRollerMotor.move_voltage(12000); //max voltage 12000
}
else if (current_ball != alliance_ball) {
TopRollerMotor.move_voltage(-12000); //max voltage 12000
pros::delay(200);
TopRollerMotor.move_voltage(12000); //max voltage 12000
current_ball = alliance_ball;
}
}

}

}
```

Drivetrain:

```
//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 LeftControl = -power - turn;
int RightControl = power - turn;

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 LeftControl = -power - turn;
int RightControl = power - turn;

LFMotor.move(LeftControl);
LMMotor.move(LeftControl);
LBMotor.move(LeftControl);

RFMotor.move(RightControl);
RMMotor.move(RightControl);
RBMotor.move(RightControl);


pros::delay(2);

}
}
```

High Goal:

```
#include "main.h"


void HighGoalOn(){
//FrontRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
//TopRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
//BinRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);

FrontRollerMotor.move_voltage(12000); //max voltage 12000
TopRollerMotor.move_voltage(12000); //max voltage 12000
BinRollerMotor.move_voltage(-12000); //max voltage 12000
pros::delay(20);
toppiston.set_value(true);
goal_on = true;
}

void HighGoalOff(){
//FrontRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
//TopRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
//BinRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);

pros::delay(20);
FrontRollerMotor.move_voltage(0); //max voltage 12000
TopRollerMotor.move_voltage(0); //max voltage 12000
BinRollerMotor.move_voltage(0); //max voltage 12000
goal_on = false;
}



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

//FrontRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
//TopRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
//BinRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);

while (true) {

if (master.get_digital(DIGITAL_L1)) {
HighGoalOn();
}
else if (master.get_digital(DIGITAL_L2)) {
//CollectorMotorA.move_voltage(-12000); //max voltage 12000
//CollectorMotorB.move_voltage(12000); //max voltage 12000
HighGoalOff();
}

if (goal_on == true) {
if (BinRollerMotor.get_actual_velocity() == 0.0) { //BinRollerMotor stalls
BinRollerMotor.move_voltage(12000); //max voltage 12000
pros::delay(200);
BinRollerMotor.move_voltage(-12000); //max voltage 12000
}
}

}

}
```

Middle Goal:

```
#include "main.h"


void MidGoalOn(){
//FrontRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
 //TopRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
 //BinRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);

    FrontRollerMotor.move_voltage(12000); //max voltage 12000
     TopRollerMotor.move_voltage(-12000); //max voltage 12000
     BinRollerMotor.move_voltage(-12000); //max voltage 12000
                pros::delay(20);
            toppiston.set_value(true);
                goal_on = true;
                }

void MidGoalOff(){
//FrontRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
 //TopRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
 //BinRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);

                pros::delay(20);
       FrontRollerMotor.move_voltage(0); //max voltage 12000
        TopRollerMotor.move_voltage(0); //max voltage 12000
        BinRollerMotor.move_voltage(0); //max voltage 12000
                goal_on = false;
                }


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

//FrontRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
 //TopRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
 //BinRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);

                while (true) {

        if (master.get_digital(DIGITAL_A)) {
                MidGoalOn();
              //pros::delay(20);
                }
    else if (master.get_digital(DIGITAL_Y)) {
  //CollectorMotorA.move_voltage(-12000); //max voltage 12000
   //CollectorMotorB.move_voltage(12000); //max voltage 12000
                MidGoalOff();
                }

                }

                }
```

Low Goal:

```
#include "main.h"


void LowGoalOn(){
//FrontRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
 //TopRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
 //BinRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);

    FrontRollerMotor.move_voltage(-12000); //max voltage 12000
    //TopRollerMotor.move_voltage(-12000); //max voltage 12000
     BinRollerMotor.move_voltage(-12000); //max voltage 12000
                pros::delay(20);
            toppiston.set_value(true);
                goal_on = true;
                }

void LowGoalOff(){
//FrontRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
 //TopRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
 //BinRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);

                pros::delay(20);
        FrontRollerMotor.move_voltage(0); //max voltage 12000
        //TopRollerMotor.move_voltage(0); //max voltage 12000
         BinRollerMotor.move_voltage(0); //max voltage 12000
                goal_on = false;

                }


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

FrontRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
//TopRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
 BinRollerMotor.set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);

                while (true) {

    if (master.get_digital(DIGITAL_X)) {
                LowGoalOn();
                }
  else if (master.get_digital(DIGITAL_B)) {
                LowGoalOff();
                }
            }
            }
```

Competition Reflections

Parts List

complete List of Parts used:

- Motors: 9 (7 11w + 2 5.5w)
- Piston: 2
- Gas tank: 1
- 2x25 c channel (aluminum): 12
- 3x35 c channel (aluminum):1
- 2x25 c channel (steel):1
- Omni-wheels: 6
- Drive shaft: 12
- various sprockets for rollers
- chains: 100 links
- Rubber bands
- various high strength gears

Outreach

Outreach History

For more than three years, our team led robotics workshops at Burke Library, encouraging a passion for STEM in the community. In order to give kids from all backgrounds access to hands-on learning experiences, we set out to raise awareness of and education about STEM.

Through these workshops, we sparked creativity and gave students the tools to tackle challenges with confidence. By consistently being there for our community, we established a foundation of trust and inspired a wave of excitement about STEM, helping students see new possibilities for their future.

Why We Started The RISE Foundation

We believe STEM, especially robotics, is an unmatched tool for teaching essential hard and soft skills. It's engaging, fun, and hands-on, offering students the chance to apply critical life skills like communication, teamwork, and problem-solving. Robotics serves as a springboard for shaping future leaders and innovators.

Our personal journey has shown us that STEAM knowledge extends far beyond traditional fields like engineering and science. It plays a vital role in industries such as business, law, healthcare, and beyond. With the RISE Foundation, our mission is not just to educate but to equip young people with the resources they need to thrive while fostering a culture of leadership and lifelong learning.

RISE Foundation

The RISE Foundation, managed by our team, uses robotics as a platform to inspire and empower students, but our mission goes far beyond that. We actively engage the community through monthly or bi-weekly STEAM hours and workshops at Herndon Library, providing consistent opportunities for hands-on learning. Additionally, we've partnered with Sully community center to host robotics "camps" and workshops during the summer and winter, broadening our reach and impact. As we continue to grow, we are committed to expanding these initiatives to serve even more communities.

Our Focus Areas:

- Giving learners the leadership, teamwork, communication, problem-solving, and critical thinking skills that are necessary in any line of work.
- Providing underrepresented youth with meaningful access to STEM opportunities and more by facilitating community-based programs that provide regular support.
- Educating and guiding the upcoming generation of leaders, empowering them to carry forward our mission of community support and innovation.

At its core, the RISE Foundation fosters a cycle of growth: today's students become tomorrow's mentors, helping our community continue to grow and thrive.


VEX IQ

Where we've worked