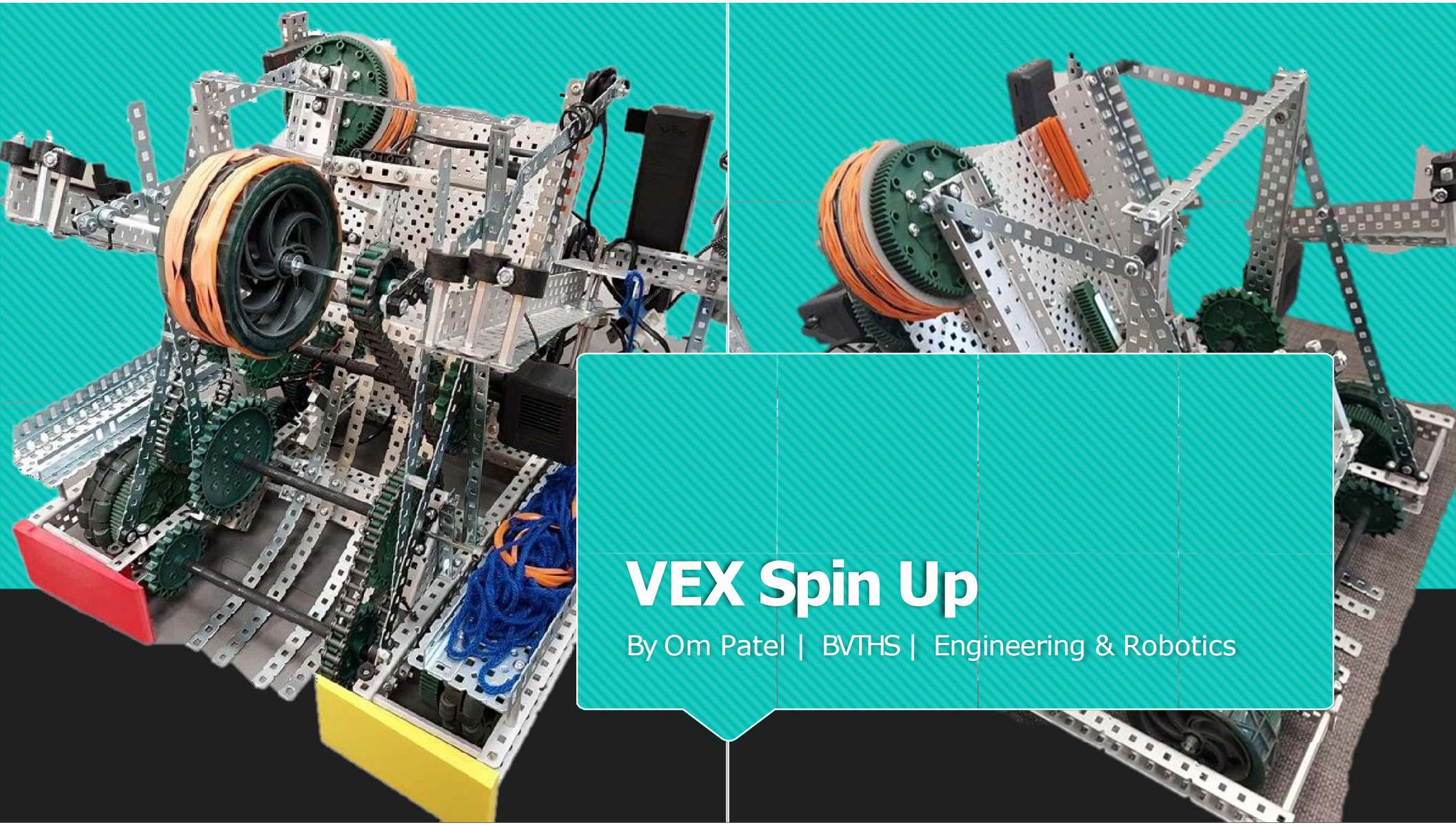


VEX Spin Up

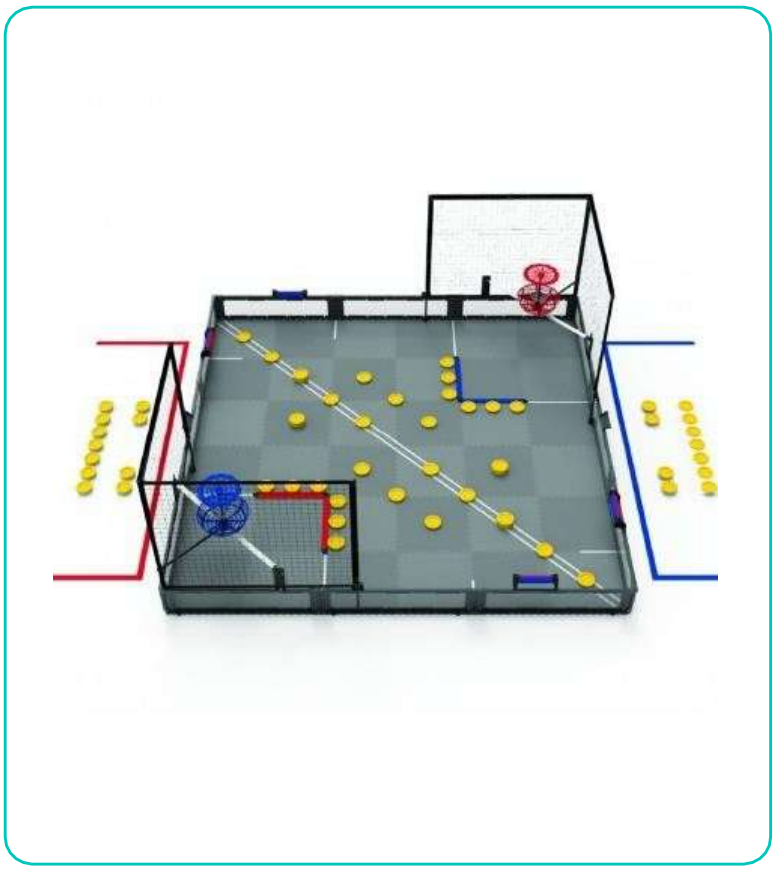
By Om Patel | BVTHS | Engineering & Robotics

Table of Contents

- Goal - 1
- Tasks - 2
- Drive Train - 3
- Intake - 4
- Flywheel - 5
- Indexer - 6
- Expansion - 7
- Roller - 8
- Coding - 9-10
- Sensors - 11-12
- Functions – 13-16
- Modeling – 17

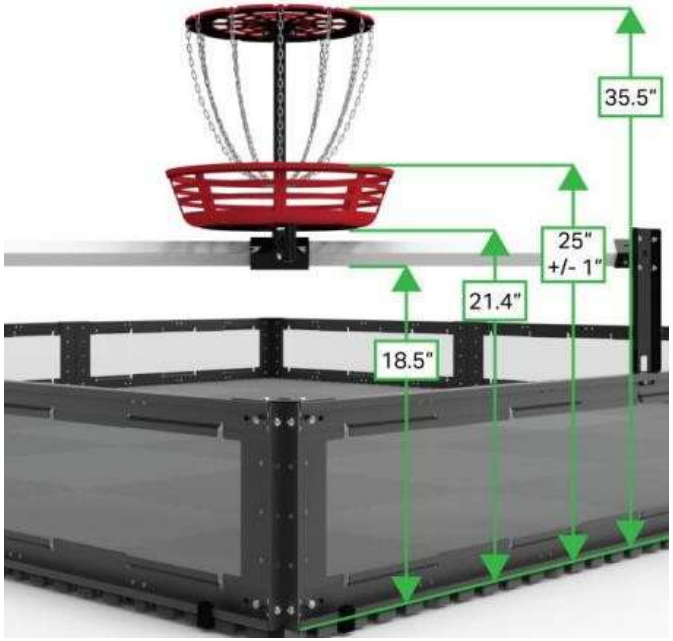
Goal

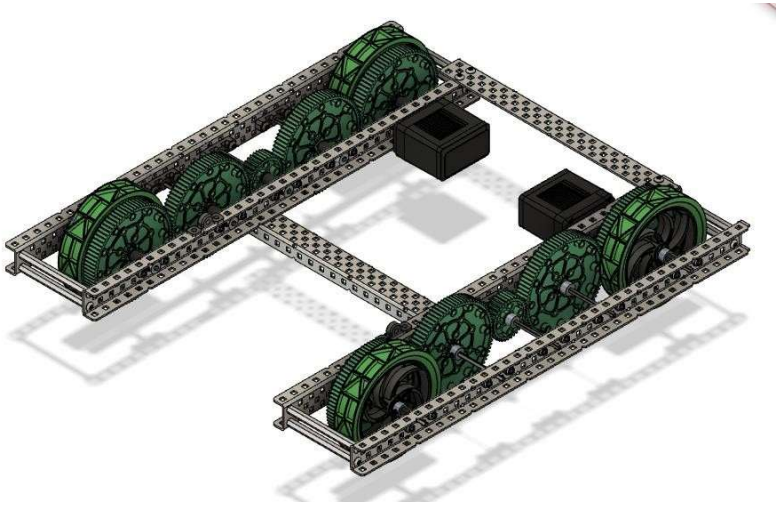
- In this project my partner and I were challenged to build a robot to achieve certain tasks
- This led us to learn and practice time management, critical thinking, coding, and the assembly of robots

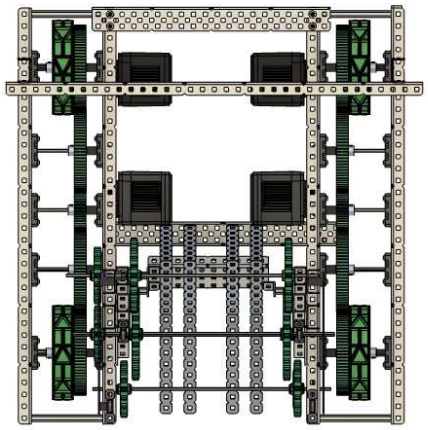


Tasks

- In VEX's Spin Up, the main objective is to shoot discs into a goal that is elevated above the ground that is similar to disc golf (5pts/disc)
- This game also has side objectives such as spinning rollers to your assigned team color (10pts/roller)
- One other quirk of this game is that in the last ten seconds, you must try and cover as many tiles as possible in order to gain more points (3pts/tile)
- One way to negate points is by missing a goal and letting it fall into the lower goal.(-1pt)





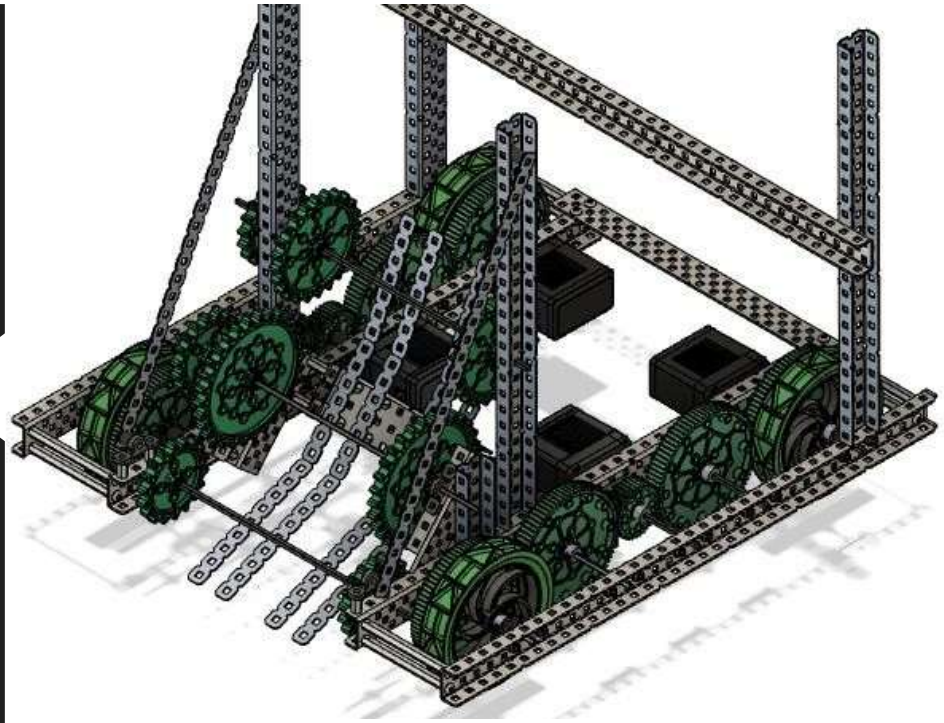


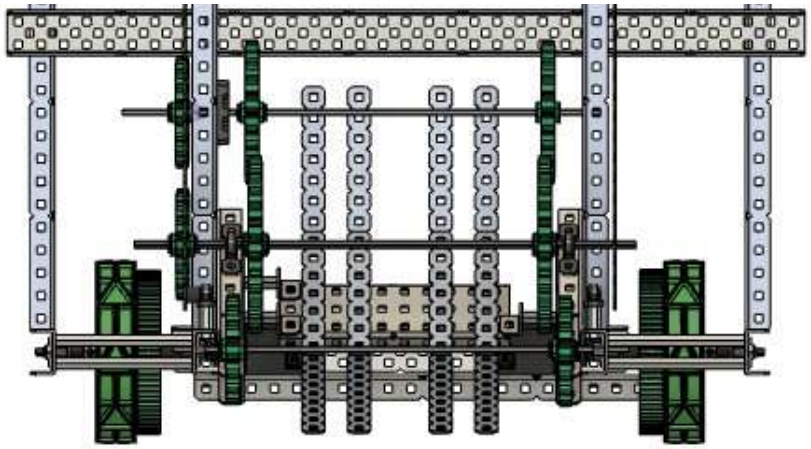
Drive Train

- One of the most important parts in a robot is its drive train, as it the main way it moves
- While making a drive train you must think about speed, mobility, and sturdiness
- Due to this we decided to use a tank drive with 2 motors, 2 traction wheels in the back, and 2 omni wheels in the front

Intake

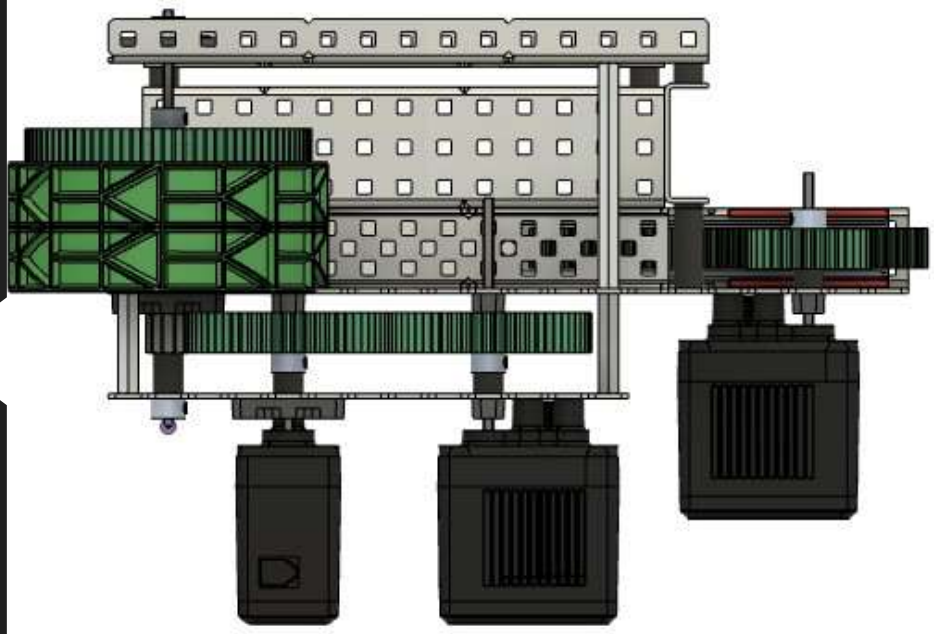
- The intake allows the robot to carry the discs
- While designing the intake we used rubber bands on gears as it provided for a flexible and grippy surface that allows the discs to slide onto and around the metal tracks
- Design Considerations: Flexibility, Torque, Speed, Grip, Consistency

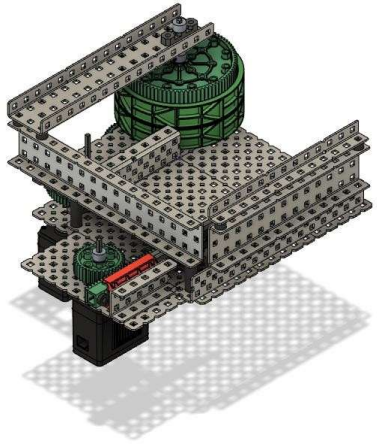


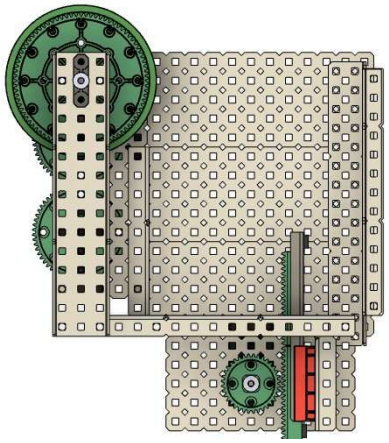


Flywheel

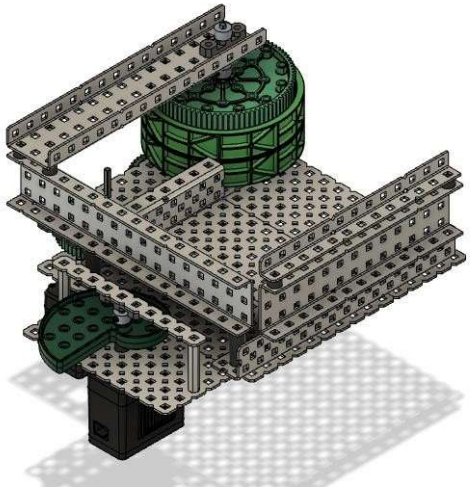
- The Flywheel is a wheel that is spinning at such high speeds it can easily shoot out objects at far distances with lots of power.
- Due to the need for compression, grip, and surface area on the disc we used rubber bands
- We used 2 (6:1) motors on a 5:1 gear ratio as it allowed for more lots of speeds and enough torque for our needs
- Design Considerations: Speed, Torque, Size, Compatibility, Stability



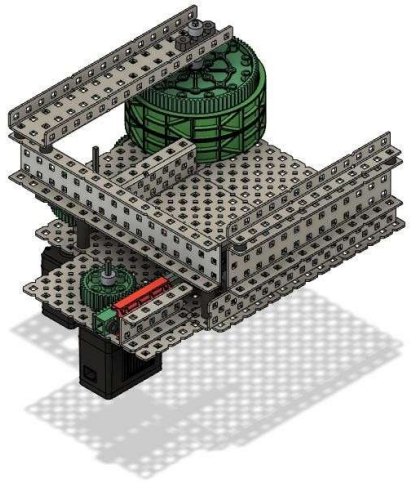




Before

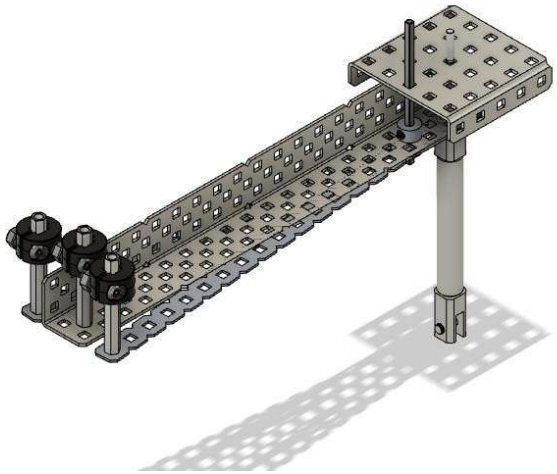


After



Indexer

- The indexer allows the disc to be pushed into the flywheel and makes our more consistent and reliable.
- Design Considerations: Size, Speed, and Compatibility

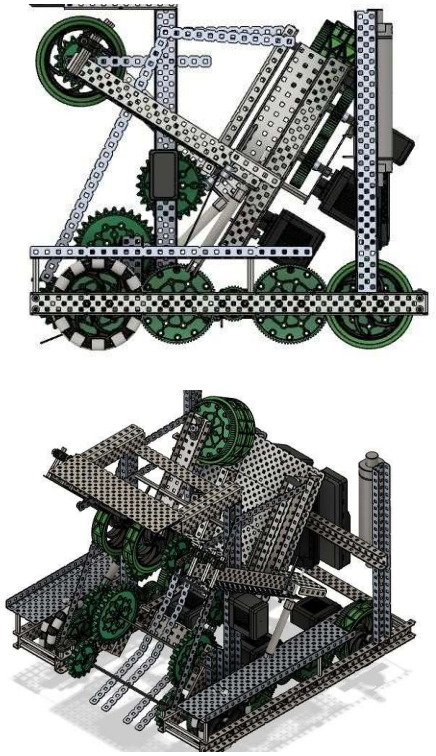


Expansion

- Inspiration: Rubber Band Handguns
- Using a double acting pneumatic cylinder, we were able to recreate the act of shooting a rubber band and with this we could then attach string to one end of it and expand over the whole field
- While designing this we tried to make sure that it was simple and thin so we could attach it to the side of our robot

Roller

- The roller is used to spin the dual-colored rollers attached to the fields.
- The two wheels and rubber bands allow for more flexion and grip permitting the roller to easily spin while the mechanism spins
- We attached the roller to the same motor as the intake system as we didn't want to reach our motor limit and we didn't have enough space to mount another one



Coding

- The autonomous section teaches allows us to practice and further our coding abilities.
- Using functions and simple drive commands we were able to create a path that shot up 5 discs, spun 2 rollers, and expanded across the field.

```
def when_started1():
    global myVariable, count1, count2, count3, heightGoal,
    drivetrain.set_drive_velocity(30, PERCENT)
    # Pick Up Disc 1
    IntakeRoller.set_velocity(100, PERCENT)
    IntakeRoller.spin(FORWARD)
    drivetrain.drive_for(FORWARD, 22, INCHES, wait=True)
    wait(2.5, SECONDS)
    IntakeRoller.stop()
    # Align with Middle Diggonal
    drivetrain.drive_for(REVERSE, 20, INCHES, wait=True)
    drivetrain.turn(RIGHT)
    wait(1.125, SECONDS)
    drivetrain.stop()
    # Pick Up Disc 2
    IntakeRoller.spin(FORWARD)
    drivetrain.drive_for(FORWARD, 22, INCHES, wait=True)
    wait(0.25, SECONDS)
    IntakeRoller.stop()
    drivetrain.drive_for(REVERSE, 6, INCHES, wait=True)
    # Align With Roller1
    drivetrain.turn(RIGHT)
    wait(3.025, SECONDS)
    drivetrain.stop()
    drivetrain.drive_for(FORWARD, 21, INCHES, wait=True)
```

```
# Align With Roller2
drivetrain.drive_for(REVERSE, 15, INCHES, wait=True)
drivetrain.turn(LEFT)
wait(1.125, SECONDS)
drivetrain.stop()
drivetrain.drive_for(FORWARD, 13, INCHES, wait=True)
# To Goal
drivetrain.drive_for(REVERSE, 8, INCHES, wait=True)
drivetrain.turn(RIGHT)
wait(1.2, SECONDS)
drivetrain.stop()
drivetrain.drive_for(REVERSE, 55, INCHES, wait=True)
# Pick Up Disc 3,4,5
drivetrain.drive_for(FORWARD, 62, INCHES, wait=True)
drivetrain.turn(RIGHT)
wait(1.5375, SECONDS)
drivetrain.stop()
IntakeRoller.spin(FORWARD)
drivetrain.drive_for(FORWARD, 34, INCHES, wait=True)
wait(0.5, SECONDS)
IntakeRoller.stop()
```

```
# To Goal
drivetrain.turn(LEFT)
wait(1.5375, SECONDS)
drivetrain.stop()
drivetrain.drive_for(REVERSE, 35, INCHES, wait=True)
drivetrain.turn(RIGHT)
wait(0.5125, SECONDS)
drivetrain.stop()
wait(1, SECONDS)
drivetrain.drive_for(FORWARD, 24, INCHES, wait=True)
drivetrain.turn(LEFT)
wait(2.8625, SECONDS)
drivetrain.stop()
drivetrain.set_drive_velocity(50, PERCENT)
drivetrain.drive_for(REVERSE, 65, INCHES, wait=True)
drivetrain.stop()
# Expansion System
Expansion1.set(True)
Expansion2.set(True)
```

Coding

Sensors (Vision Camera)

- The vision sensor allows the user todetect the color and size of an object
- With this ability we were able to code a program that could align the flywheel of our robot to the goal of our team color
- We also used a second vision sensor to make sure we spin the roller until it is spun to our team color

Sensors (Distance Sensor)

- The distance sensor detects the distance and velocity of objects
- With this sensor we could use it to make a safety net that checked if all three disc are loaded into the flywheel before we shoot them for our autonomous code

```
def onevent_alignRightBlue_0():
    global myVariable, count1, count2, count3, heightGoal, message1, shootDisc,
    # allows for the robot to not turn too fast
    drivetrain.set_turn_velocity(25, PERCENT)
    count3 = 0
    # will always take a picture of what the camera sees until the color is det
    while not count3 == 1:
        vexcode_vision_14_objects = vision_14.take_snapshot(vision_14__REDBOX)
        if vexcode_vision_14_objects and len(vexcode_vision_14_objects) > 0:
            count3 = 1
            drivetrain.stop()
            break
        else:
            drivetrain.turn(RIGHT)
        wait(5, MSEC)
```

```
def onevent_alignLeftBlue_0():
    global myVariable, count1, count2, count3, heightGoal, messa
    # allows for the robot to not turn too fast
    drivetrain.set_turn_velocity(25, PERCENT)
    count3 = 0
    # will always take a picture of what the camera sees until t
    while not count3 == 1:
        vexcode_vision_14_objects = vision_14.take_snapshot(visi
        if vexcode_vision_14_objects and len(vexcode_vision_14_o
            count3 = 1
            drivetrain.stop()
            break
        else:
            drivetrain.turn(LEFT)
        wait(5, MSEC)
```

Functions (Alignment)

- Using a function, we could call a command that aligns the robot until the flywheel is aligned with the assigned team goal using the vision sensor.
- This can be done to the left or right for either the red or blue team.

```
def onevent_alignLeftRed_0():
    global myVariable, count1, count2, count3, heightGoal, message1, shootDisc, redR
    # allows for the robot to not turn too fast
    drivetrain.set_turn_velocity(25, PERCENT)
    count3 = 0
    # will always take a picture of what the camera sees until the color is detected
    while not count3 == 1:
        vexcode_vision_14_objects = vision_14.take_snapshot(vision_14__REDBOX)
        if vexcode_vision_14_objects and len(vexcode_vision_14_objects) > 0:
            count3 = 1
            drivetrain.stop()
            break
        else:
            drivetrain.turn(LEFT)
        wait(5, MSEC)
```

```
def onevent_alignRightRed_0():
    global myVariable, count1, count2, count3, heightGoal, message1, shootDisc, redR
    # allows for the robot to not turn too fast
    drivetrain.set_turn_velocity(25, PERCENT)
    count3 = 0
    # will always take a picture of what the camera sees until the color is detected
    while not count3 == 1:
        vexcode_vision_14_objects = vision_14.take_snapshot(vision_14__REDBOX)
        if vexcode_vision_14_objects and len(vexcode_vision_14_objects) > 0:
            count3 = 1
            drivetrain.stop()
            break
        else:
            drivetrain.turn(RIGHT)
        wait(5, MSEC)
```

Functions (Alignment)

Functions (Roller)

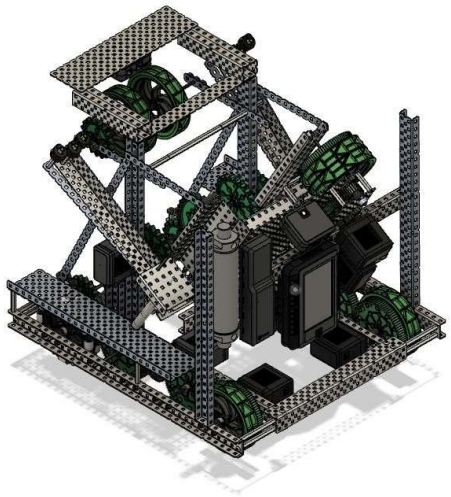
- Using a function, we could call a command that spins the roller until the field roller is our team color (red/blue) using the Vision Sensor

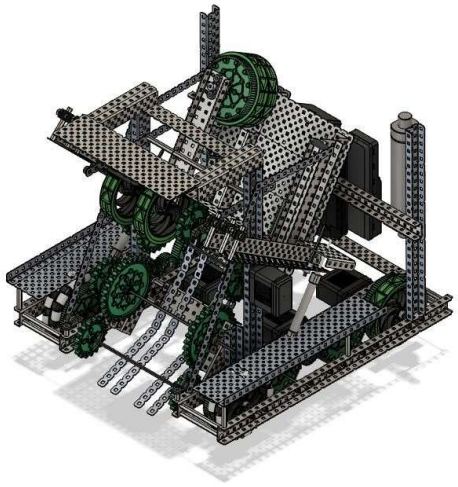
```
def onevent_blueRoller_0():
    global myVariable, count1, count2, count3, heightGoal, messa
    # allows for the roller to not spin too fast
    IntakeRoller.set_velocity(40, PERCENT)
    count2 = 0
    # will always take a picture of what the camera sees until
    while not count2 == 1:
        vexcode_vision_13_objects = vision_13.take_snapshot(vis
        if vexcode_vision_13_objects and len(vexcode_vision_13_
            count1 = 1
        else:
            IntakeRoller.spin_for(FORWARD, 10, DEGREES, wait=Tru
        wait(5, MSEC)


def onevent_redRoller_0():
    global myVariable, count1, count2, count3, heightGoal, messa
    # allows for the roller to not spin too fast
    IntakeRoller.set_velocity(40, PERCENT)
    count2 = 0
    # will always take a picture of what the camera sees until
    while not count2 == 1:
        vexcode_vision_13_objects = vision_13.take_snapshot(vis
        if vexcode_vision_13_objects and len(vexcode_vision_13_
            count1 = 1
        else:
            IntakeRoller.spin_for(FORWARD, 10, DEGREES, wait=Tru
        wait(5, MSEC)
```

Functions (Shoot)

- Using a function, we could call a command that rapidly shoots discs only when the robot is carrying a stack of 3 discs. This is done with the distance sensor.

```
def onevent_shootDisc_0():
    global myVariable, count1, count2, count3, heightGoal, message1, shootDis
    # uses distance sensor to check that there are discs in the flywheel
    if Distance.object_distance(MM) < 50:
        Flywheel.set_velocity(100, PERCENT)
        Indexer.set_max_torque(50, PERCENT)
        # if discs are there they will spin the flywheel and start the loop
        Flywheel.spin(FORWARD)
        wait(2, SECONDS)
        # extends and retracts the indexer 4 times
        for repeat_count in range(4):
            Indexer.spin(FORWARD)
            wait(3, SECONDS)
            Indexer.spin(REVERSE)
            wait(3, SECONDS)
            wait(5, MSEC)
        # stops the flywheel
        Flywheel.stop()
```





Modeling

- This project also allowed me to practice my modeling skills as every part and prototype made for this project was created in Fusion 360.