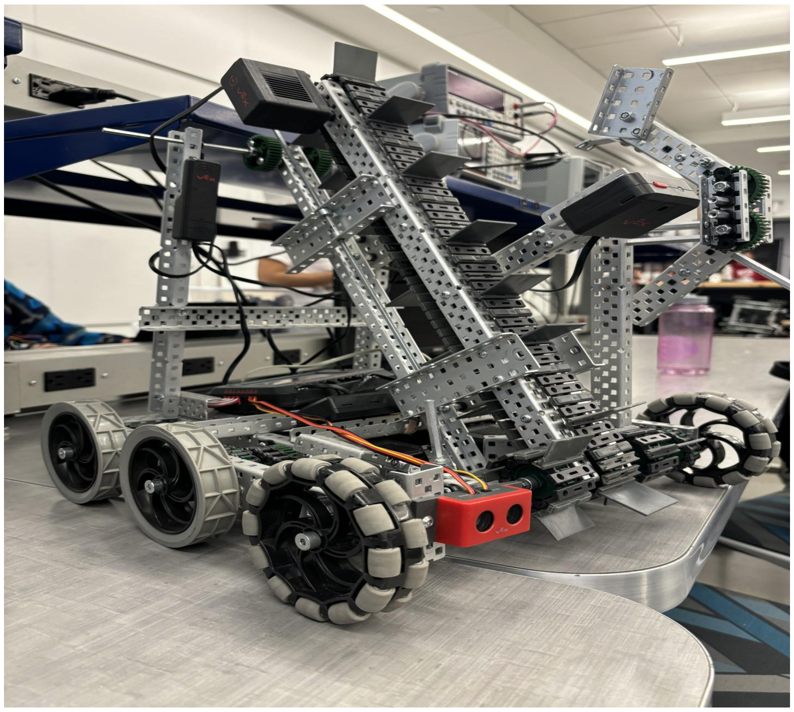
RBE 1001 B23, Introduction to Robotics
Final Project Report



"Black Magic"

Team 10

Member	Signature	Lab Contribution %
Theron Boozer	[illegible]	33.34
Benjamin Synder	[illegible]	33.33
Tanner Young	[illegible]	33.33

Member	Signature	Project Contribution %
Theron Boozer	[illegible]	35
Benjamin Synder	[illegible]	25
Tanner Young	[illegible]	40

Table of Contents

Table of Contents........i
List of Figures........ii
Section 1- Introduction:........1
Section 2- Preliminary Discussion:........3
Section 3- Problem Statement:........5
  Design Specifications:........5
Section 4- Preliminary Designs:........6
Section 5- Selection of Final Design:........11
Section 6- Iterative Design Analysis:........15
  Mechanical Analysis Double Conveyor Belt:........15
    Collector Analysis:........16
  Mechanical Analysis Flywheel:........17
    Shooter Analysis:........18
Section 7- Final Design Analysis:........21
  Mechanical Analysis:........21
    Drive Train Horizontal Tractive Effort:........22
    Drive Train Ramp Tractive Effort:........24
    Drive Train Horizontal Speed:........27
    Drive Train Ramp Speed:........28
    Robot Stability Front:........30
    Robot Stability Back:........31
    Intake Analysis:........31
    Conveyor Belt Analysis:........33
    Lift Transmission/ Torque Requirement:........35
    Lift Operation Speed:........37
    Lift Kinematics:........38
  Electrical Engineering Analysis:........38
    Sensor selection, Location, Utility:........39
    Maximum Current Draw (all systems):........43
  Programming Analysis:........44
Section 8- Summary/evaluation:........45
Section 9- Appendix:........48

List of Figures

Figure 1: Field Layout.......... 1
Figure 2: Ball Placement..........2
Figure 4: Competition Scoring.......... 3
Figure 3: Demo Day Scoring.......... 3
Figure 5: Claw Bot Design Sketch..........6
Figure 6: Bucket Bot Design Sketch..........7
Figure 7: Conveyor Belt Design Sketch..........8
Figure 8: Flywheel Design Sketch.......... 9
Figure 9: Bucket Bot Prototype..........11
Figure 10: Conveyor Belt Prototype..........13
Figure 11: Top and right view of Conveyor 1 FBD.......... 16
Figure 12: Flywheel Prototype.......... 18
Figure 13: Flywheel FBD.......... 19
Figure 14: Driving On A Horizontal FBD..........22
Figure 15: Wheel FBD.......... 22
Figure 16: Driving on a Ramp FBD..........24
Figure 17: Vex V5 motor Torque, Current, and Power Graph..........27
Figure 18: Robot Stability Front FBD..........30
Figure 19: Intake System FBD..........32
Figure 20: Arm FBD..........32
Figure 21: Conveyor Belt FBD..........34
Figure 22: Conveyor Belt Arm.......... 35
Figure 23: Robot Crunches..........36
Figure 24: Inertial Sensor note figure in text needs to change.......... 39
Figure 25: Line Tracker Sensors..........40
Figure 26: Ultrasonic Range Finder Sensor.......... 41
Figure 27: Vision Sensor.......... 42
Figure 28: Vex V5 motor Torque, Current, and Power Graph..........4**3**

Section 1- Introduction:

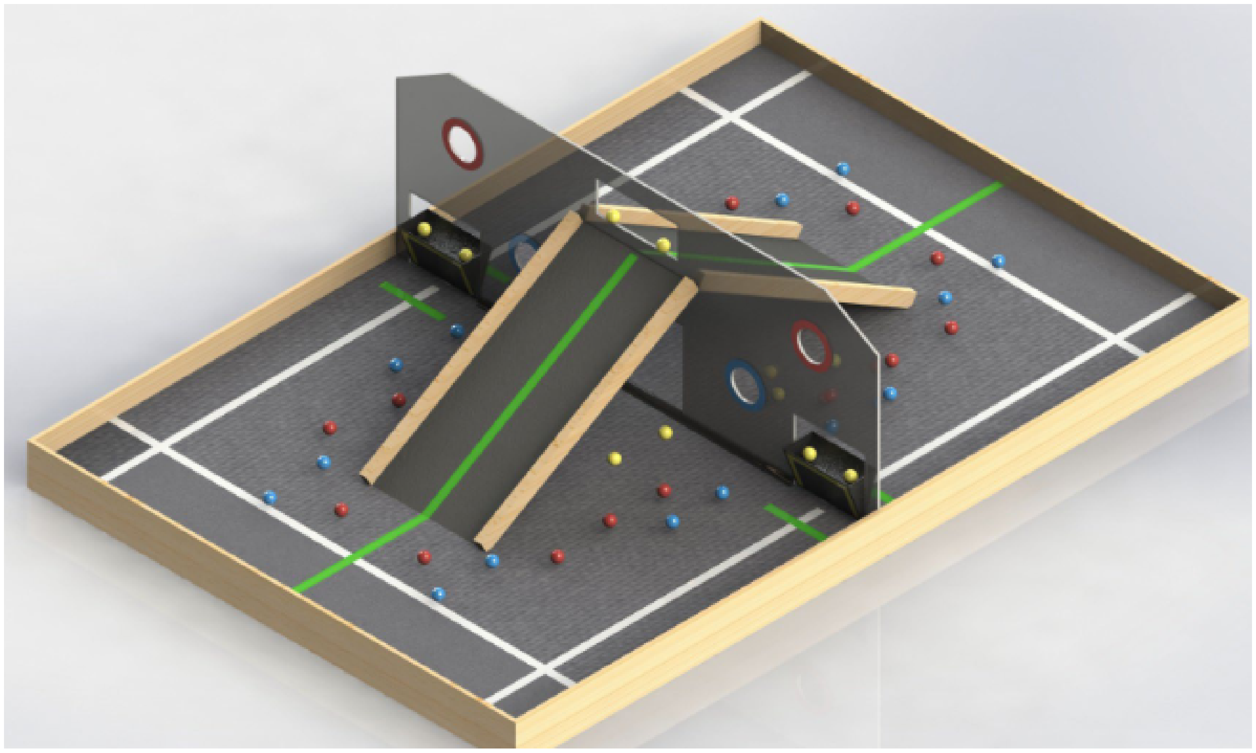


Figure 1: Field Layout

Our team undertook the design challenge of creating an autonomous and telescopically capable vehicular robot for the B23 Final Project. The project involves a coalition of robots working together to defeat another team in the "not in my yard" game. In this timed game, robots aim to place as many red and blue balls as possible on the opponent's side of the field. Various openings, buckets, and holes facilitate the movement of balls between sides. To add complexity and strategy, teams also have the objective of collecting yellow balls and earning points by ending matches with yellow balls on their alliance's side. The game unfolds in two main phases: a one-minute autonomous period where robots follow preset commands and sensor inputs, and a two-minute tele-operated period where teams use a Vex remote control to operate their robots.

Points are accumulated at the end of both periods with balls being counted cumulatively at the end of both rounds. A unique element is the requirement for teams to switch controllers or face penalties after the first minute. In the last 30 seconds of the game, teams can score additional points by crossing over to the opponent's side, playing a combination of offense and defense. Climbing the 30-degree ramp and ending matches parked at the top also offer opportunities for scoring. Fields are set up at the beginning of every match as pictured below in Figure 2. The competition allows teams to play six matches with two points awarded for a win and one for a draw. Tie breakers for ranking will be determined by total points scored during rounds. The highest scoring team will receive a 25% decrement bonus to final project scores.

During Demo Day, teams aim to pass balls through designated holes similar to the competition, with a key difference: balls are counted as they pass through the holes rather than at the end. Each team has a ten-minute slot to showcase both autonomous and driver-controlled sections. The points earned vary depending on the specific hole through which balls are passed, with red balls through the upper red hole holding higher value than those through the lower blue hole.

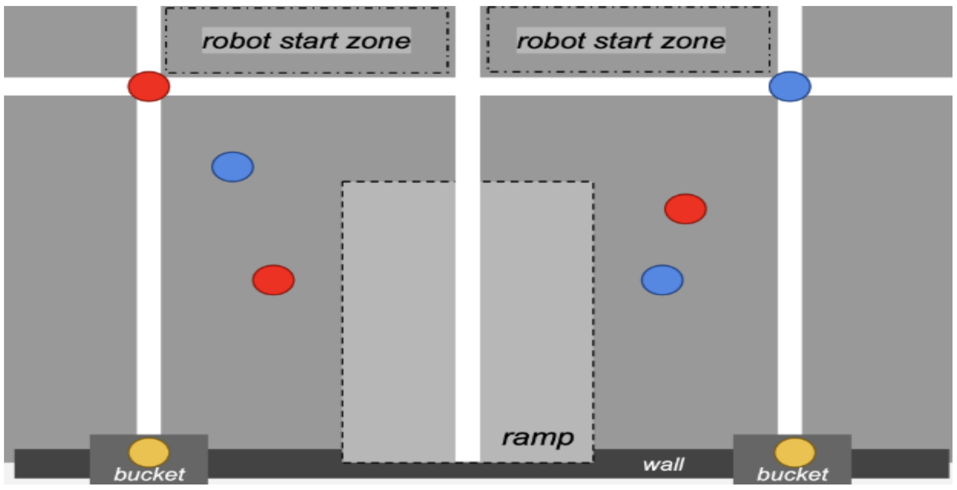
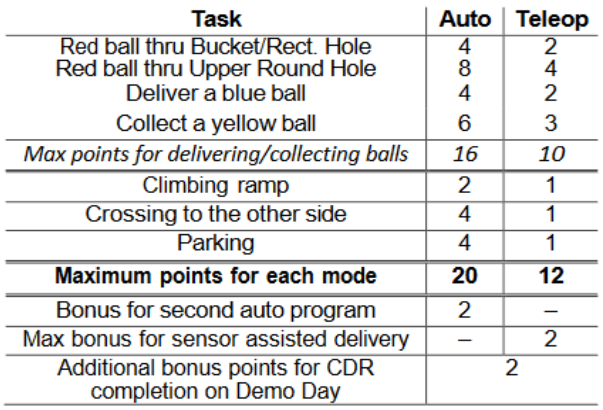
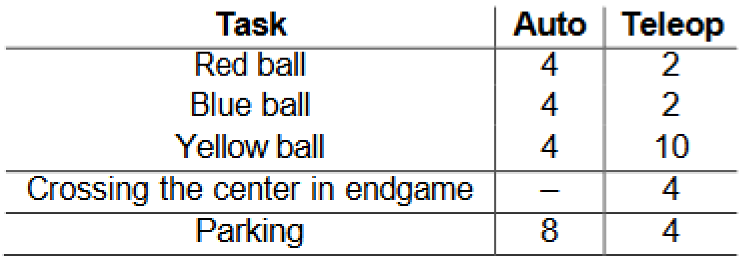


Figure 2: Ball Placement

Section 2- Preliminary Discussion:



Task	Auto	Teleop
Red ball thru Bucket/Rect. Hole	4	2
Red ball thru Upper Round Hole	8	4
Deliver a blue ball	4	2
Collect a yellow ball	6	3
Max points for delivering/collecting balls	*16*	*10*
Climbing ramp	2	1
Crossing to the other side	4	1
Parking	4	1
Maximum points for each mode	**20**	**12**
Bonus for second auto program	2	–
Max bonus for sensor assisted delivery	–	2
Additional bonus points for CDR completion on Demo Day	2	



Task	Auto	Teleop
Red ball	4	2
Blue ball	4	2
Yellow ball	4	10
Crossing the center in endgame	–	4
Parking	8	4

Figure 4: Competition Scoring

Figure 3: Demo Day Scoring

As our team embarked on the initial stages of brainstorming for the competition, we recognized the pivotal role of the scoring rubrics in shaping our strategy, as illustrated in Figure 3 and Figure 4. For Demo Day, red balls passing through the upper hole accumulate 8 points, while blue balls through the lower hole are valued at 4 points. Interestingly, during the actual competition, both red and blue balls carry the same point value. Consequently, our unanimous decision was to construct a robot optimized for efficiently and swiftly feeding blue balls through the lower hole, deemed the most strategic approach.

With this decision in mind, our focus shifted to elements that could enhance our performance and differentiate us from other teams. A six-wheel, four-motor-powered drive train emerged as a key component, providing superior traction on the ramp, preventing bottoming out during crossovers, enhancing speed, and offering the ability to overpower opponents in head-to-head pushing matches. Incorporating two omni wheels at the front facilitated smoother turns, while four rubber-banded traction wheels ensure optimal traction.

Next, we turned our attention to maximizing the efficiency of the sensors at our disposal. The Vex Vision sensor emerged as a valuable tool for adaptable code execution on the fly. By programming it to identify red and blue balls, the sensor could guide the robot to collect them. Line sensors positioned two inches beneath the robot could efficiently locate balls at the start of the autonomous period. The ultrasonic range finder would be employed to determine the optimal distance from the wall for depositing balls.

Having considered point values, drive train, and sensor ideas, we were ready to strategize for the autonomous and teleop periods. The primary goal for the autonomous phase was twofold: first, collect balls on the line using line sensors, find the correct distance using the ultrasonic range finder, and deposit the balls through the hole. The second part involved utilizing the vision sensor to rove, locate stray balls deposited by opponents, collect them, and return them to the opponent's side.

In the teleop period, our strategy focused on outpacing opponents by depositing balls in the corner zone for efficient collection and redeposit on the opponent's side. A robot designed for speed would enable us to overwhelm opponents, especially in the final 30 seconds, where the vision sensor could be used to hunt for yellow balls on the opponent's side.

Following individual brainstorming sessions, we collectively identified four main competition designs. The claw bot, bucket-based bot, conveyor belt-based bot, and flywheel design each presented distinct advantages and challenges. Our ultimate design goal prioritized reliability to consistently score points based on skill rather than luck. Adaptability to diverse opponent strategies and the ability to deposit the maximum number of balls in the shortest time emerged as key elements for an effective winning strategy.

Section 3- Problem Statement:

Design and construct a robot to pick up, transport, and deliver balls to the opposite side of the field through varying methods.

Design Specifications:

High Priority:

- Scoring through the blue/ red hole
- Fit in the size constraint of 15.25” x 15.25” x 18.75”
- Use the vision sensor
- Climb the ramp

Medium Priority:

- Collect yellow balls
- Weigh as close to ten pounds without going over as possible
- Use the line tracker sensor
- Park on the ramp

Low Priority:

- Crossing underneath the ramp
- Construct a hook for the bucket without the use of a motor
- Use the ultrasonic range sensor
- Use other sensors not required to avoid a point deduction

Section 4- Preliminary Designs:

With a comprehensive understanding of the game rules, our initial brainstorming discussions, and a careful evaluation of the problem statement and our objectives, we were poised to commence the design phase. The four main designs we have identified are the clawbot, bucket design, conveyor belt, and the flywheel. Each of these designs pose both a list of advantages and disadvantages to our design goal. All designs need to meet the high priority requirements and should attempt to meet the medium and low requirements as well. All designs will be constrained to using a six wheel four motor drive train. This will allow for both speed and power at the price of weight and motor consumption in our final design.

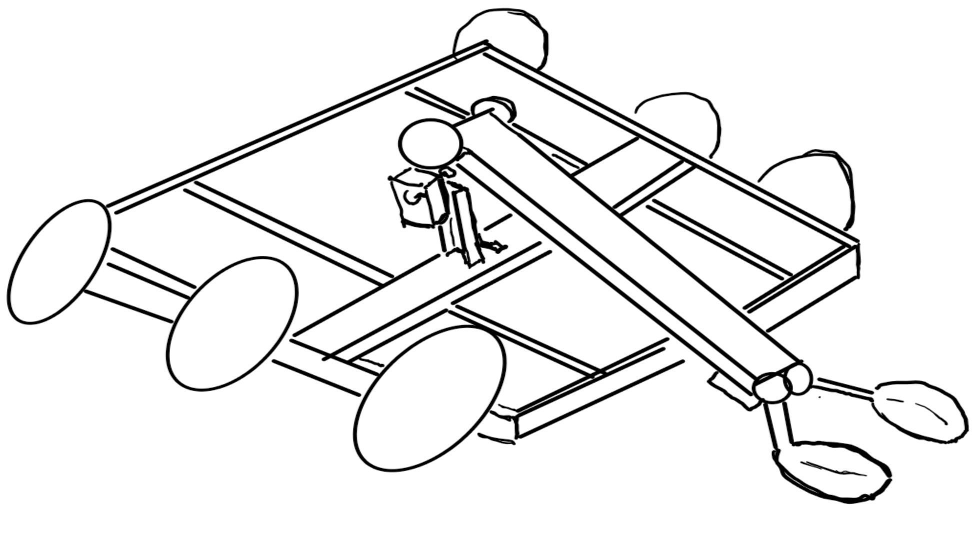


Figure 5: Claw Bot Design Sketch

Our initial concept involves a modification of the claw bot initially designed in the first week of class. Adapting the wheelbase to accommodate six wheels, along with integrating the necessary motors and gearboxes, is a key aspect of this redesign. Various claw options were considered, including utilizing the basic claw provided by Vex, implementing a clamping mechanism with rubber bands and Vex C-bars, or designing and 3D printing a custom claw for optimal ball gripping. After careful consideration, the 3D printed claw emerged as the most effective choice, tailored specifically for the task. Another critical decision involves the design of the arm. Two versions were contemplated: one featuring an up-and-down motion suitable for the blue hole, and the other incorporating a four-bar design to lift the load towards the red hole. The four-bar design demands more time, resources, calculations, and adds weight, making the vertically movable arm the more pragmatic choice for this iteration. The claw bot offers advantages in terms of being lightweight, swift, and relatively easy to construct, allowing more time for coding and troubleshooting. However, its limitation lies in the ability to hold only a single ball at a time. While suitable for Demo Day, this poses a significant challenge for the competition. The design must efficiently transport individual balls in a shorter timeframe than an opponent's bot, which may collect and deposit multiple balls simultaneously.Thus, our primary objective is to successfully feed blue balls through the lower hole individually.

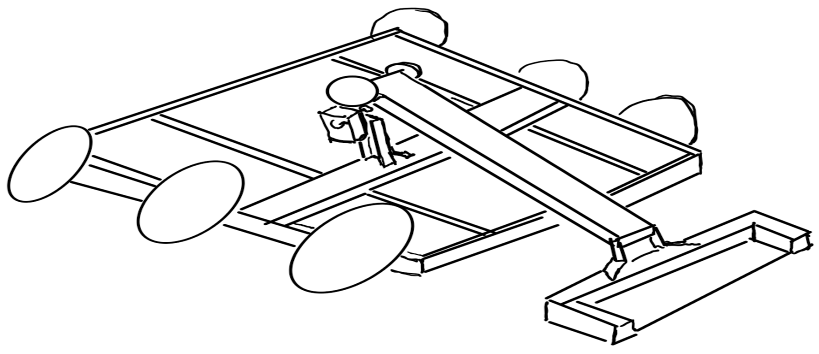


Figure 6: Bucket Bot Design Sketch

In our subsequent concept, we explore the integration of a ground-hugging bucket that hovers just above the surface. Inspired by the maneuverability of skid steer loaders commonly found on farms, this design enables the robot to approach balls and effortlessly collect them through collisions. The emphasis here is on swift collection without the need for precise alignment or stopping movement during the retrieval process. Similar to the claw bot, we have the option of incorporating either a straightforward raising and lowering arm or a four-bar mechanism into this design. Given our priority on speed, the more efficient choice is a raising and lowering arm. In this configuration, one motor is dedicated to lifting the arm, while another motor is utilized to tip the bucket. However, challenges emerge when encountering hard stops, as there is a risk of balls rolling back out of the bucket. Additionally, careful consideration is required to ensure that if the bucket size exceeds that of the hole, all collected balls can be effectively funneled into the designated opening. Thus, our primary objective is to successfully feed five blue balls through the lower hole.

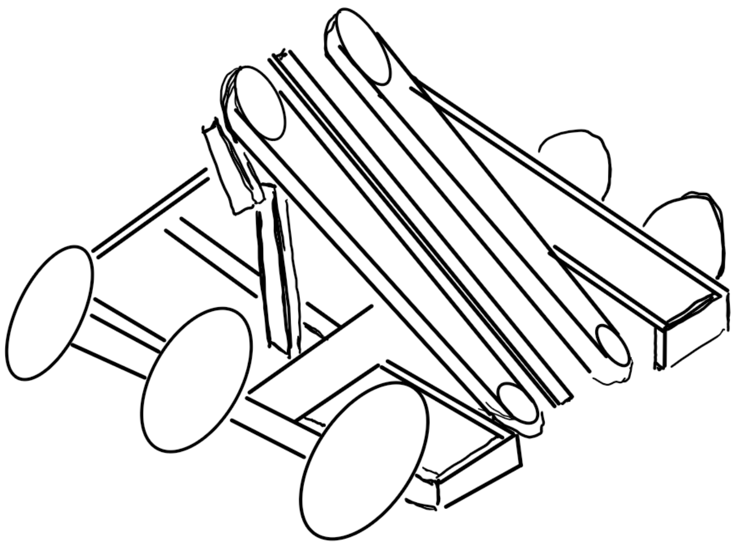


Figure 7: Conveyor Belt Design Sketch

Our next concept revolves around utilizing Vex tank treads to craft a conveyor belt system for effective ball collection. This design involves both an intake system and the conveyor belt itself, with two proposed ideas for its implementation. The first idea entails a single conveyor belt with a track underneath that propels the balls upward. On the other hand, the second concept incorporates two conveyor belts mounted on the sides to collect the balls laterally. While the first idea is lighter, it presents challenges in keeping the balls adequately contained on the belt. In contrast, the second mechanism is weightier but offers enhanced support, aligning with our design priorities where weight is considered only a medium priority. Consequently, the second idea is deemed more effective, with Vex flaps ensuring the balls remain securely in place. However, potential issues may arise, such as limiting the feeding process solely through the blue hole. Weight concerns also come into play due to the bulkiness of the mechanism. Another crucial consideration involves developing a controlled system to release the balls at will, independent of the conveyor belt's operation. Thus, our primary objective is to successfully feed five blue balls through the lower hole.

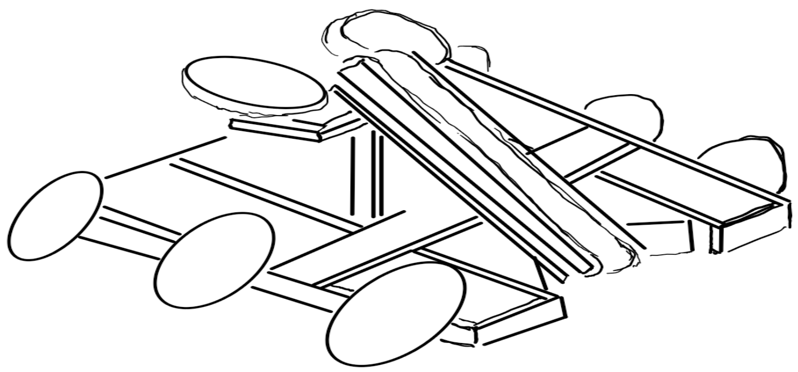


Figure 8: Flywheel Design Sketch

The last concept involves a synergistic combination of a conveyor belt and a flywheel to tackle our designated task. This intricate design incorporates an intake system, a conveyor belt, and a flywheel, presenting the intricate challenge of seamlessly designing, constructing, and coding within the designated time frame. One inherent complexity lies in the demands imposed by flywheels, requiring ample power and precise ball entry angles for optimal functionality. In this design, a single conveyor belt proves more efficient than employing two, strategically chosen to adhere to weight constraints. The flywheel is strategically positioned at the end of the conveyor belt, addressing concerns about balls being inadvertently expelled from the system as the conveyor belt operates. The mechanism operates by rapidly spinning a wheel at high RPM, achieved through a carefully calibrated high-speed gear ratio. To effectively transmit the speed of the flywheel to the ball in a brief timeframe, compression between the ball and the wheel becomes pivotal. This compression enhances the normal force on the ball, consequently increasing friction and facilitating the swift deposit of red balls through the upper hole. Undoubtedly, this design presents a formidable challenge of launching five red balls through the upper hole. However, the potential rewards are substantial if the design is executed with precision and finesse.

Section 5- Selection of Final Design:

Having outlined our preliminary designs, the next crucial step is to identify two concepts for the prototyping phase. One design that we can confidently exclude from consideration is the claw bot design. Its limitation lies in its ability to transport only a single ball at a time, whereas other designs can handle the maximum quantity. To compensate for this shortfall, the claw bot would need to operate five times faster than the alternative concepts, a challenging and impractical goal. Consequently, we find this design unfeasible for advancing to the prototyping phase. Similarly, the flywheel design is not proceeding to the prototyping phase. This design necessitates the integration of both a conveyor belt and a flywheel within a constrained time frame. Although the potential reward for successfully executing this combination is high, the associated risk of failure is significant. As a result, we have decided to focus on prototyping the bucket and conveyor belt designs, which present more practical and achievable prospects.

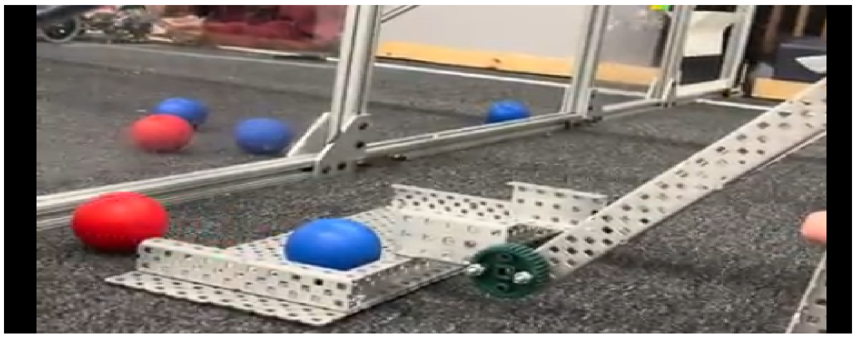


Figure 9: Bucket Bot Prototype

This design introduced some unforeseen challenges. The bucket could not tip with a motor due to the motor being attached to the bucket. Initially, attempting to rotate the bucket using a motor posed a dilemma, as the motor, being attached to the bucket, counteracted its rotation. To address this, we needed to reposition the motor responsible for rotating the bucket to the robot itself, not directly on the bucket. However, this solution introduced a new complication: how to rotate the bucket when it's attached to a moving arm. In our prototype, we devised a solution using slightly loosened screws and nuts on the gears. Striking the right balance was crucial; if too loose, the stability of the bucket was compromised, and if too tight, the bucket wouldn't tip. We leveraged gravity to facilitate the tipping motion when the bucket was lifted into the air, freeing up a motor. Another challenge surfaced when executing a simple lift command for the bucket – a single motor lacked sufficient torque to overcome gravity. This led to the incorporation of a second free motor to solve this issue. Upon evaluation, it's evident that this design requires refinement. Issues such as abrupt stopping and ineffective feeding of all five balls through the hole need addressing. The prototyping phase has been instrumental in identifying and understanding these challenges, emphasizing the indispensable role of hands-on experimentation in the design process.

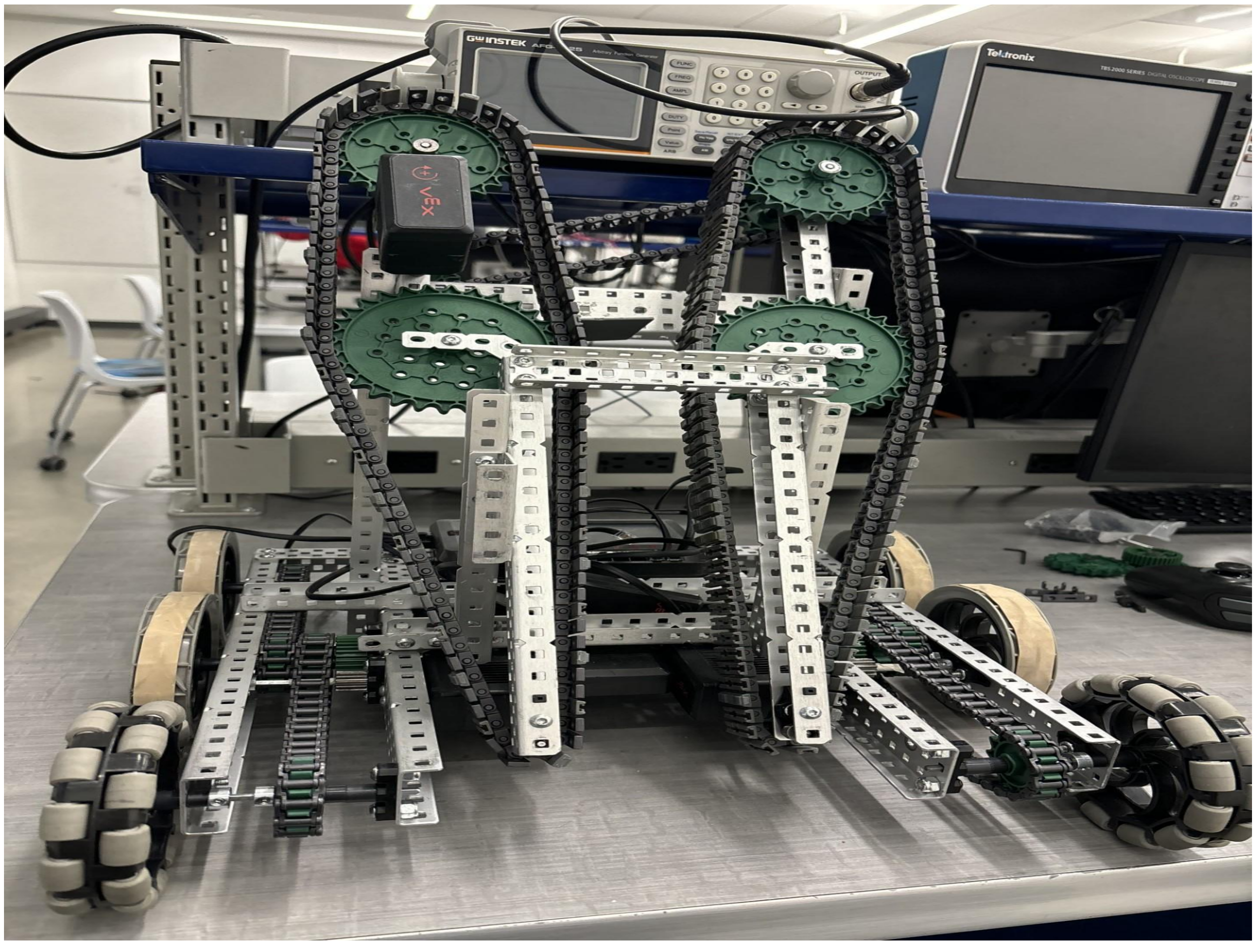


Figure 10: Conveyor Belt Prototype

Our second design, while promising, brought to light unexpected challenges that were not initially considered. The design envisions using one motor for the intake and another for both conveyor belts. However, a complication arises as both conveyor belts need to move in the same direction, necessitating the inclusion of an idler to alter the direction. Unfortunately, this additional stage introduces an efficiency loss, impacting the overall performance. Moreover, the conveyor belt's speed falls short of our desired specifications. The weight constraint is also a concern, with the dual conveyor belts and the requisite frame supports potentially exceeding the ten-pound limit. The current prototype is missing an intake, but is able to grab balls without the feeder. In the current iteration, there is no efficient mechanism in place to prevent the continuous running of the conveyor belt without losing balls out the back. Despite these challenges, it still demonstrates a respectable success rate in feeding balls through the designated hole. Addressing these issues will be crucial in refining the design for optimal functionality.

Upon thoughtful consideration of both prototypes, the conveyor belt design emerges as the clear frontrunner. While the bucket prototype successfully translates ideas into a tangible prototype, it presents a multitude of challenges that require extensive problem-solving and refinement—more so than the conveyor belt counterpart. In contrast, the conveyor belt prototype manages to achieve the project goal, albeit with room for improvement in terms of efficiency and effectiveness. Identifying the existing issues with this design has instilled confidence within our group that these challenges can be addressed and resolved in the refinement process. With a focus on enhancing the conveyor belt design, we are poised to move forward into the next phase of our project—the Final Design analysis. This decision reflects our commitment to continuous improvement and the pursuit of an optimal solution for our envisioned robotic system.

Section 6- Iterative Design Analysis:

Upon opting for the double conveyor belt design, our team initiated a comprehensive phase dedicated to the essential calculations essential for analyzing our chosen design. These calculations served as a foundational step in comprehending the capabilities of our robot and pinpointing potential errors in our design. The insights gained from this analysis played a pivotal role in refining the programming for both the autonomous and teleoperated periods, enhancing the overall performance of our robot. It's crucial to note that the outcomes of this analysis did not directly translate into our final design. Instead, this process served as a valuable learning experience, enabling us to identify and rectify mistakes, implement improvements for subsequent designs, and foster team growth through the iterative design process. While the analysis of the double conveyor belt and flywheel is detailed in this section, our ultimate and more comprehensive design analysis is presented in Section 7.

Mechanical Analysis Double Conveyor Belt:

Initially conceived to address the challenge of ball loss on the sides of a conveyor belt, our design fell short of the standards required for our robot to be truly competitive in the upcoming competition. Although this design had the potential to meet the criteria for the Critical Design Review, our team collectively opted for a more ambitious approach. Motivated by the desire to push our limits, we chose to revisit our list of preliminary designs, recognizing the importance of elevating our robot's capabilities to ensure a stronger performance in the competitive arena.

Collector Analysis:

The collector analysis of the double conveyor belt proved to us that the design was not efficient enough to collect balls effectively. The mechanism struggled with ball collection, had no way to keep balls in the collector if we wanted to run the belt, and ran too slowly.

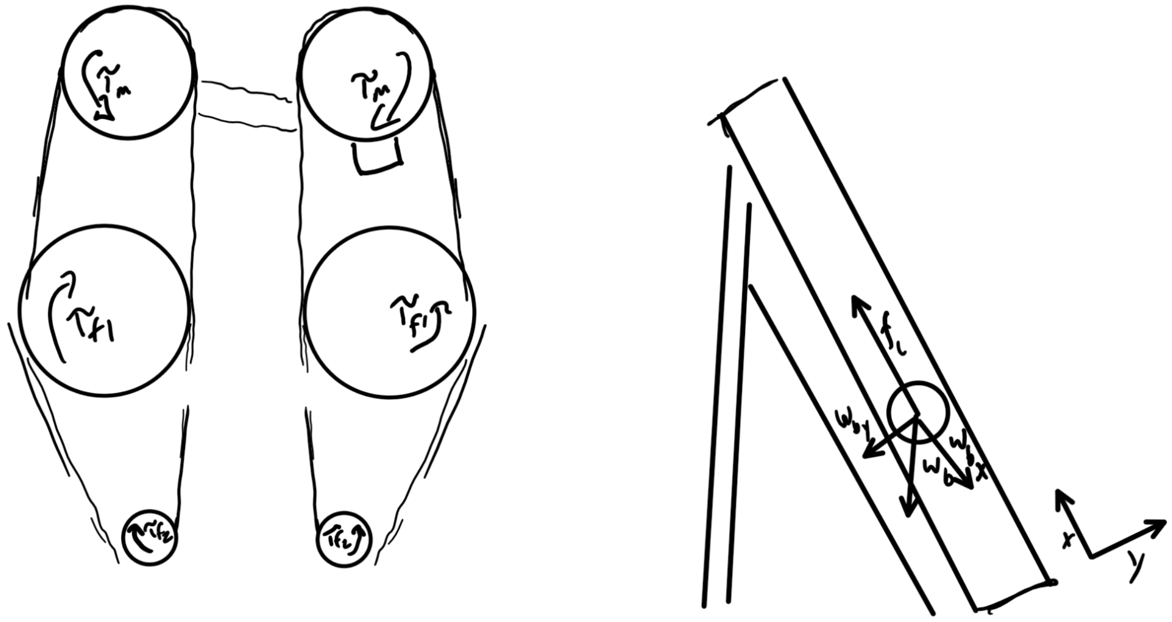


Figure 11: Top and right view of Conveyor 1 FBD

Equations:

$$W_{bx} = W_b \cdot \sin(\theta)$$
$$\tau_{motor} \geq \sum \tau_{friction} + W_{bx} \cdot r$$
$$W_{bx} = W_b \cdot \sin(\theta)$$

Known Variables:

$$\sum \tau_{friction} = 2.25\ in \cdot lbs$$

$$W_b = 0.0625\ lbs$$

$$r = 1\ in$$
$$\theta = 60$$

Solution:

$$\tau_{motor} = 2.25\ (in \cdot lbs)\ +\ 0.0625\ (lbs)\ \cdot\ \sin(60°)\ \cdot\ 1\ in\ \cdot\ 5\ balls = 2.52$$

Mechanical Analysis Flywheel:

Opting to dismantle the double flywheel, our initial step involved preserving the integrity of our six-wheel, four-motor base. This robust robot base was strategically designed to facilitate the seamless integration of various mechanisms, ensuring the essential balance of traction, speed, and power required for a competition-ready robot. Our team's strategic approach began with a thorough reassessment of the preliminary designs outlined in Section 4. Having explored physical models of the bucket and conveyor belt, we found ourselves at a crossroads between a simplified design and a more intricate, potentially challenging one. Following a unanimous decision by the team, we embraced the ambitious challenge of pursuing the flywheel design as our primary course of action.

Shooter Analysis:

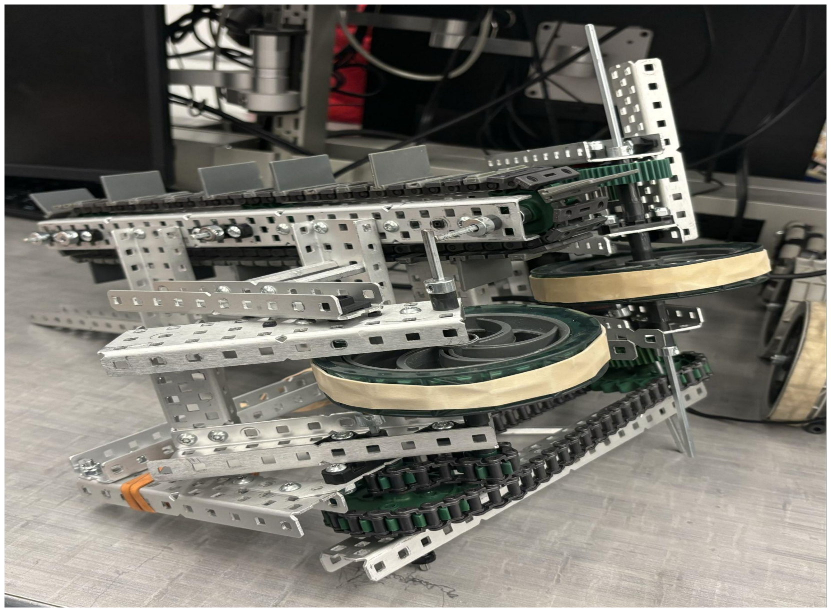


Figure 12: Flywheel Prototype

Rather than conducting a comprehensive analysis of the flywheel mounted to the base bot, we opted to construct the model illustrated in Figure X. While this model demonstrated success in launching balls fed through the new conveyor belt, a critical challenge emerged – the wheels failed to reach a sufficiently high RPM to propel the balls a significant distance, compromising the reliability of the system. The primary issue identified was a lack of adequate 'squish' in both the ball and the wheels, impeding their effectiveness. A major drawback was also identified in the mechanism's power source – a single motor driving five stages of operation. Each stage introduced an additional layer of complexity, introducing factors of error into the system that, in turn, reduced the final output torque and overall wheel speed essential for launching the ball effectively. Recognizing the time constraints, our team made the strategic decision to explore alternative designs. However, not all was lost, as the conveyor belt designed to feed the flywheel proved to be a valuable component that could potentially be repurposed in a new attempt at a single conveyor belt-designed robot.

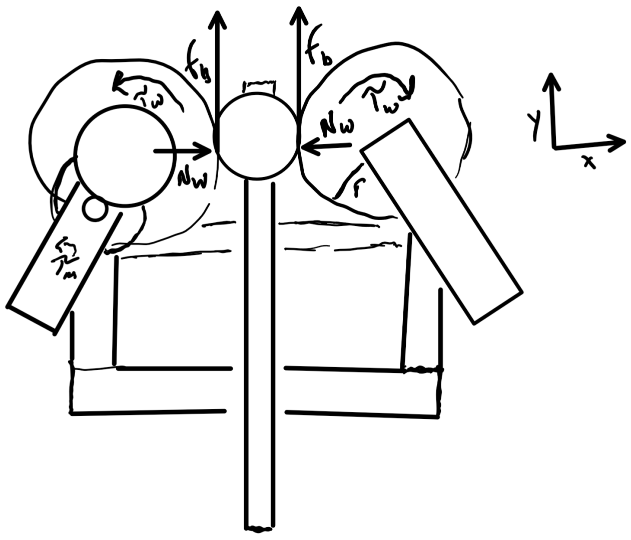


Figure 13: Flywheel FBD

Equations:

$$V_{\text{tang}ential} = \omega_{wheel} \cdot r$$
$$\omega_{wheel} = \omega_{motor} \cdot e$$
$$J = Ft = mV$$
$$t = \frac{d}{V_{\text{tang}ential}}$$

Known Variables:

$$e = 25$$
$$r = 2\ in$$
$$\omega_{motor} = 200\ rpm$$
$$m = 0.0625\ lbs$$
$$d = 0.125\ in$$

Solution:

$$V_{\text{tang}ential} = 200\ (rpm) \cdot \frac{2\pi\ (rad)}{revolution} \cdot \frac{min}{60\ (sec)} \cdot 9 \cdot 2\ (in) = 376.99\ \frac{in}{sec}$$
$$t = \frac{0.125\ in}{376.99\ \frac{in}{sec}} = 0.0003\ sec$$

$$F = \frac{0.0625\ (lbs) \ \cdot\ 376.99 \left(\frac{in}{sec}\right)}{0.0003\ (sec)} = 78,539.58\ lbs$$

This was never going to work, we were too ambitious with the speed ratio.

Section 7- Final Design Analysis:

The evolution leading to our final design was a substantial journey, marked by continuous improvements and iterative refinements that ultimately manifested in our exceptional end product. The success of this bot is indebted to the invaluable lessons learned from previous versions, where innovative ideas were explored, and constructive insights were gained to inform and enhance subsequent designs. When measured against the problem statement outlined in Section 3, our team successfully achieved the goals we set out to address. Notably, the transition from a double conveyor belt design, capable of grabbing the ball from either side, to a streamlined single conveyor belt positioned above the ball resulted in a notable reduction in weight for our robot. Addressing another challenge in our final design, we revamped the intake for the conveyor belt. Unlike previous iterations requiring precise movements, the new design seamlessly grabs the ball if it's within the feeder's vicinity, enhancing efficiency. The standout improvement in our final design, setting it apart from its predecessors, is the transformative ability of the entire conveyor belt to function as an arm. This feature allows us to feed balls through both the blue and red hoops, introducing a new level of flexibility into our design. The culmination of these enhancements represents the collective effort and dedication invested in the robot's creation, serving as a testament to our team's prowess in crafting a competition-ready robot.

Mechanical Analysis:

The mechanical analysis of the final designed robot allows us to know what our design is capable of accomplishing. It also allows us to know what our output torques are and the factors of safety we have before it reaches the point where 50% of the current limited torque is used.

Drive Train Horizontal Tractive Effort:

Below is the horizontal tractive effort of the Conveyor Belt I design. Horizontal tractive effort is the required force to generate motion between the robot and the horizontal surface of the field. Knowing this force will allow us to know if we are able to drive our robot on a horizontal surface and with what factor of safety.

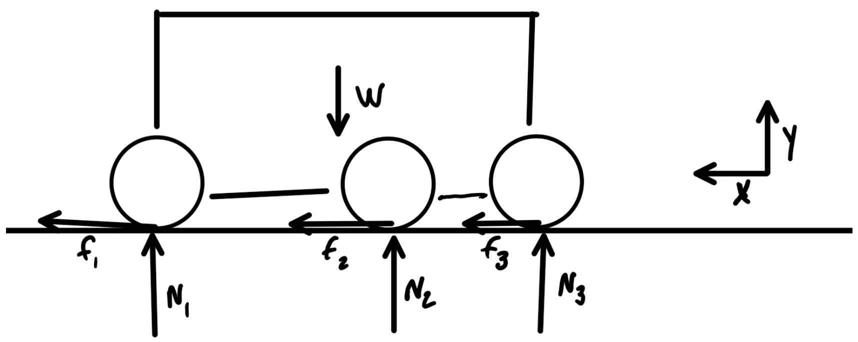


Figure 14: Driving On A Horizontal FBD

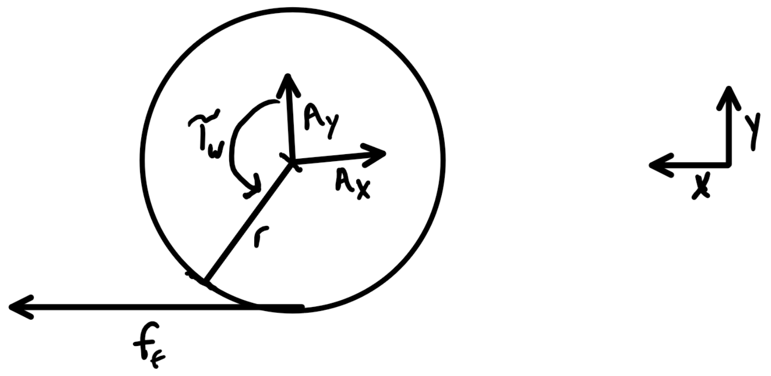


Figure 15: Wheel FBD

Equations:

$$\tau_{wheel} \geq r_{wheel} \cdot \sum f_{friction}$$

$$\sum f_{friction} = \mu \cdot \sum f_{normal}$$

$$\tau_{wheel} = \frac{\tau_{motor}}{e} \cdot \eta$$

$$e = \frac{N_{drives}}{N_{driven}}$$

$$\eta = \left(N_{gears} - 1\right) \cdot 0.9$$

$$\sum f_{normals} = W$$

Known Variables:

$N_{drives} = 12\ (teeth)$

$N_{driven} = 36\ (teeth)$

$N_{gears} = \#\ gears\ between\ (inclusive)\ driver\ and\ driven = 2$

$W = 11.36\ lbs$

$\tau_{motor} = 4.25\ in \cdot lbs\ \cdot 4_{motors} = 17\ in \cdot lbs$

$\mu = 1.1$

Solving:

Required effort:

$Ef_{normal} = 11.36\ lbs$

$Ef_{friction} = 11.36\ (lbs)\ \cdot\ 1.1 = 12.50\ lbs$

$t_{wheel} \geq 2\ (in)\ \cdot\ 12.5\ (lbs) = 25.0\ (in \cdot lbs)$

Actual Maximum Effort:

$$e = \frac{12\ (teeth)}{36\ (teeth)} = \frac{1}{3}$$

$$\eta = (2\ (gears) - 1) \cdot 0.9 = 0.9$$

$$\tau_{wheel} = \frac{17\ (in \cdot lbs)}{\frac{1}{3}} \cdot 0.9 = 45.90\ in \cdot lbs$$

Factor of Safety:

$$F_{safety} = \frac{Actual}{Necessary} = \frac{45.9\ in \cdot lbs}{25.0\ in \cdot lbs} = 1.83$$

Drive Train Ramp Tractive Effort:

Below is the angled tractive effort of the Conveyor Belt I design. Angled tractive effort is the required force to generate motion between the robot and the angled surface of the field. Knowing this force will allow us to know if we are able to drive our robot on the ramp surface and with what factor of safety.

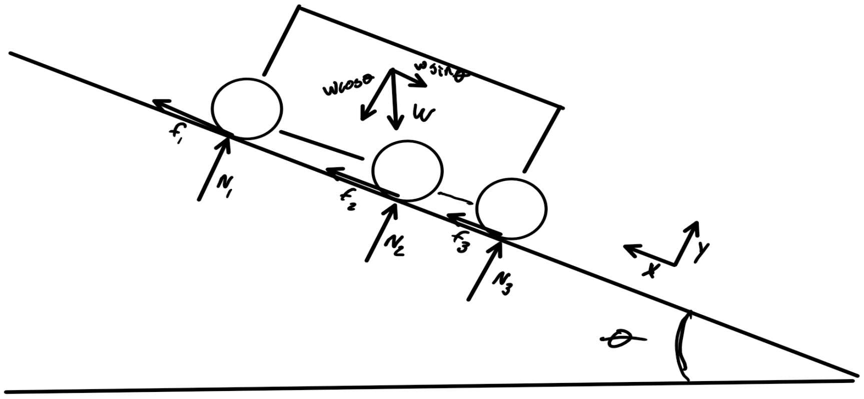


Figure 16: Driving on a Ramp FBD

Equations:

$$\tau_{wheel} \geq r_{wheel} \cdot \sum f_{friction}$$

$$\sum f_{friction} = \mu \cdot \sum f_{normal}$$

$$\tau_{wheel} = \frac{\tau_{motor}}{e} \cdot \eta$$

$$e = \frac{N_{drives}}{N_{driven}}$$

$$\eta = \left(N_{gears} - 1\right) \cdot 0.9$$

$$\sum f_{normals} = W\cos(\theta)$$

$$\sum F_x = 0 = \sum f_{friction} - W\sin(\theta)$$
$$\sum F_y = 0 = \sum f_{normal} - W\cos(\theta)$$

Known Variables:

$$N_{drives} = 12\ (teeth)$$
$$N_{driven} = 36\ (teeth)$$

$$N_{gears} = \#\ gears\ between\ (inclusive)\ driver\ and\ driven = 2$$

$$W = 11.36\ lbs$$

$$\tau_{motor} = 4.25\ in \cdot lbs\ \cdot 4_{motors} = 17\ in \cdot lbs$$

$$\mu = 1.1$$

$$\theta = 30°$$

Solving:

Required Mu:

$$W\sin(\theta) = \sum f_{friction} = \mu \cdot \sum f_{normal}$$

$$W\cos(\theta) = \sum f_{normal}$$

$$W\sin(\theta) = W\cos(\theta) \cdot \mu$$
$$\tan(\theta) = \mu = 0.577$$

Required effort:

$$\sum f_{normal} = 11.36(lbs) \cdot \cos(30^\circ) = 9.84\ lbs$$

$$\sum f_{friction} = 9.84(lbs) \cdot 1.1 = 10.82\ lbs$$

$$\tau_{wheel} \geq 2\ (in) \cdot 10.82\ (lbs) = 21.64\ in \cdot lbs$$

Actual Maximum Effort:

$$e = \frac{12\ (teeth)}{36\ (teeth)} = \frac{1}{3}$$

$$\eta = (2\ (gears) - 1) \cdot 0.9 = 0.9$$

$$\tau_{wheel} = \frac{17\ (in \cdot lbs)}{\frac{1}{3}} \cdot 0.9 = 45.90\ in \cdot lbs$$

Factor of Safety:

$$F_{safety} = \frac{Actual}{Necessary} = \frac{45.90\ (in \cdot lbs)}{21.64\ (in \cdot lbs)} = 2.12$$

Drive Train Horizontal Speed:

Drive train horizontal speed is the amount of inches per second our robot travels. To use max RPM, actual torque on the motor must remain below 0.4 Nm or about 3.5 in*lbs as shown in the figure below:

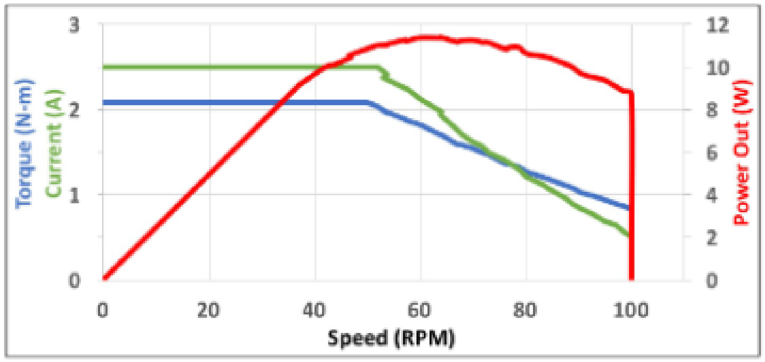


Figure 17: Vex V5 motor Torque, Current, and Power Graph

$$\tau_{motor} = \frac{\tau_{wheel} \cdot e}{motor\ count}$$

$$\tau_{motor} = \frac{25.0\ (in \cdot lbs) \cdot \frac{1}{3}}{4} = 2.08\ in \cdot lbs$$

Good to use RPM = 200

Equations:

$$V_{robot} = \left(\frac{2\pi\ rad}{rotation}\right) r\ (in) \cdot \omega_{wheel} \left(\frac{rotations}{min}\right) \cdot \left(\frac{min}{60\ sec}\right)$$

$$\omega_{wheel} = \omega_{motor} \cdot e$$

$$e = \frac{N_{drives}}{N_{driven}}$$

Known Variables:

$$\omega_{motor} = 200\ (rpm)$$

$$N_{drives} = 12\ (teeth)$$

$$N_{driven} = 36\ (teeth)$$

Solving:

$$e = \frac{12\ teeth}{36\ teeth} = \frac{1}{3}$$

$$\omega_{wheel} = \frac{200\ rotations}{minute} \cdot \frac{1}{3} = 66.67\ rpm$$

$$V_{robot} = 2\pi(2\ in) \cdot (66.67\ rpm) \cdot \left(\frac{minutes}{60\ seconds}\right) = 13.96\left(\frac{in}{sec}\right)$$

Drive Train Ramp Speed:

Drive train ramp speed is the amount of inches per second our robot travels going up a ramp. To use max RPM, actual torque on the motor must remain below 0.4 Nm or about 3.5 in*lbs. Speed on the horizontal and the ramp are the same.

$$\tau_{motor} = \frac{\tau_{wheel} \cdot e}{motor\ count}$$

$$\tau_{motor} = \frac{21.64\ (in \cdot lbs) \cdot \frac{1}{3}}{4} = 1.80\ in \cdot lbs$$

Good to use RPM = 200

Equations:

$$V_{robot} = \left(\frac{2\pi\ rad}{rotation}\right) r\ (in)\ \cdot\ \omega_{wheel}\left(\frac{rotations}{min}\right) \cdot \left(\frac{min}{60\ sec}\right)$$

$$\omega_{wheel} = \omega_{motor}\ \cdot\ e$$

$$e = \frac{N_{drives}}{N_{driven}}$$

Known Variables:

$$\omega_{motor} = 200\ (\ rpm)$$

$$N_{drives} = 12\ (\ teeth)$$

$$N_{driven} = 36\ (\ teeth)$$

Solving:

$$e = \frac{12\ teeth}{36\ teeth} = \frac{1}{3}$$

$$\omega_{wheel} = \frac{200\ rotations}{minute} \cdot \frac{1}{3} = 66.67\ rpm$$

$$V_{robot} = 2\pi(\ 2\ in) \cdot (\ 66.67\ rpm) \cdot \left(\frac{minutes}{60\ seconds}\right) = 13.96\left(\frac{in}{sec}\right)$$

Robot Stability Front:

The robot stability is the angle the robot tips at when driving forward.

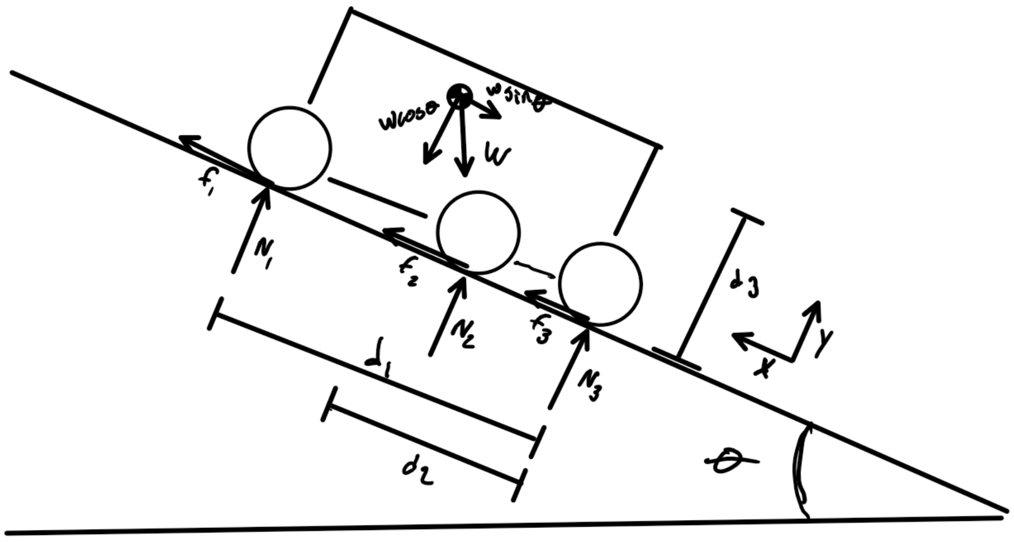


Figure 18: Robot Stability Front FBD

Equations:

$$\tan(\phi) = \frac{d_2}{d_3}$$

$$\phi \geq \theta$$

Known Variables:

$d_2 = 2.75\ in$

$d_3 = 5\ in$

$\theta = 30°$

Solution:

$$arc\tan\left(\frac{2.75}{5}\right) = \phi = 28.8°$$

$$28.8° < 30°$$

Unfortunately, the robot cannot go up the ramp forwards.

Robot Stability Back:

The robot stability is the angle the robot tips at when driving backwards.

Equations:

$$\tan(\phi) = \frac{\left(d_1 - d_2\right)}{d_3}$$

$$\phi \geq \theta$$

Known Variables:

$d_1 = 12.5\ in$

$d_2 = 2.75\ in$

$d_3 = 5\ in$

$\theta = 30°$

Solution:

$$arc\tan\left(\frac{12.5 - 2.75}{5}\right) = 62.85°$$

$$62.85° > 30°$$

Fortunately, the robot can go up the ramp backwards.

Intake Analysis:

To facilitate the transition of balls from the field to the conveyor, they must pass through the intake system. This integral component comprises the lower section of the conveyor belt, flanked by two sprockets of identical size enveloped in tank treads and equipped with flaps. The entire mechanism is linked to the conveyor belt through a centrally positioned axle, ensuring a smooth and efficient transfer of balls from the field onto the conveyor system.

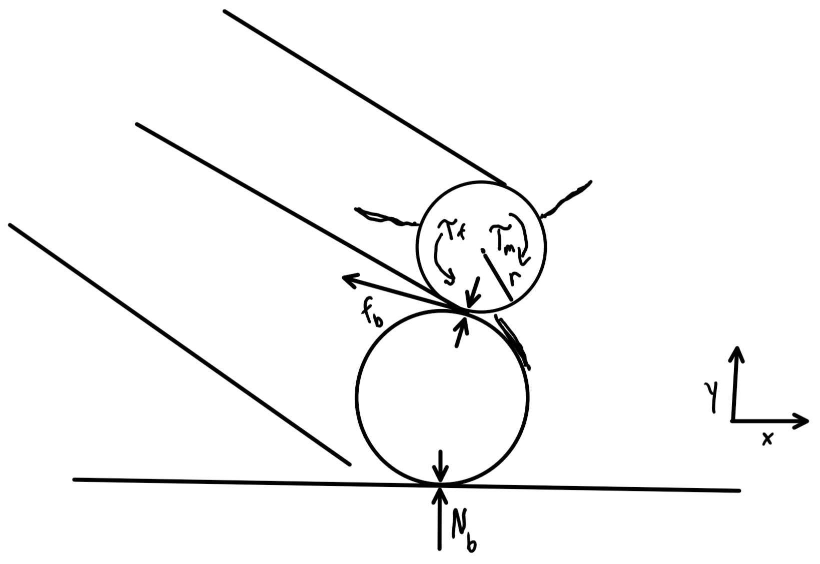


Figure 19: Intake System FBD

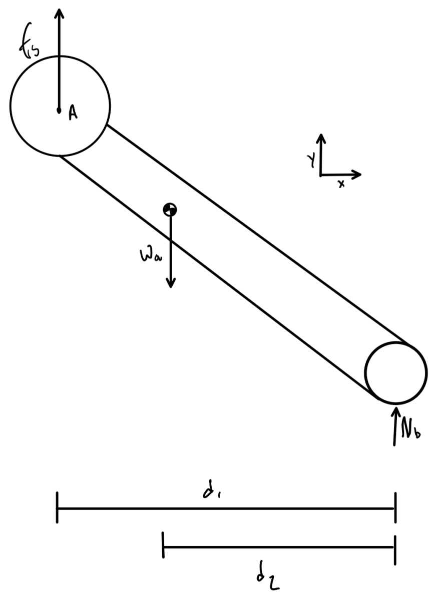


Figure 20: Arm FBD

Equations:

$$r_{outer} = r_{inner} + 0.125$$

$$\tau_{friction} = f_{friction} \cdot r_{inner}$$

$\tau_{motor} = \tau_{friction} + f_{ball} \cdot r_{outer}$

$f_{ball} = \mu \cdot N_{ball}$

$\sum_{M_A} = 0 = N_b \cdot d_1 - W_a \cdot \left(d_1 - d_2\right)$

Known Variables:

$f_{friction} = 0.25\ lbs$

$r_{inner} = 0.5\ in$

$\mu = 0.9$

$d_1 = 9\ in$

$d_2 = 7.125\ in$

$W_a = 2.14\ lbs$

Solution:

$$N_b = \frac{2.14\ (lbs) \cdot (9\ (in) - 7.125(in))}{9\ (in)} = 0.45\ lbs$$

$f_b = 0.45\ (lbs) \cdot 0.9 = 0.41\ lbs$

$\tau_{motor} = 0.25\ (lbs)\ \cdot\ 0.5\ (in)\ +\ 0.41\ (lbs)\ \cdot\ 0.6\ (in)\ = 0.37\ in \cdot lbs$

$0.37\ in \cdot lbs\ <\ 4.25\ in \cdot lbs\ (\ motor\ at\ 50\%\ )$

Conveyor Belt Analysis:

The conveyor belt is the main feature of the final design. This mechanism is able to move balls up it using the conveyor belt powered by a vex v5 motor with tank treads and flaps.

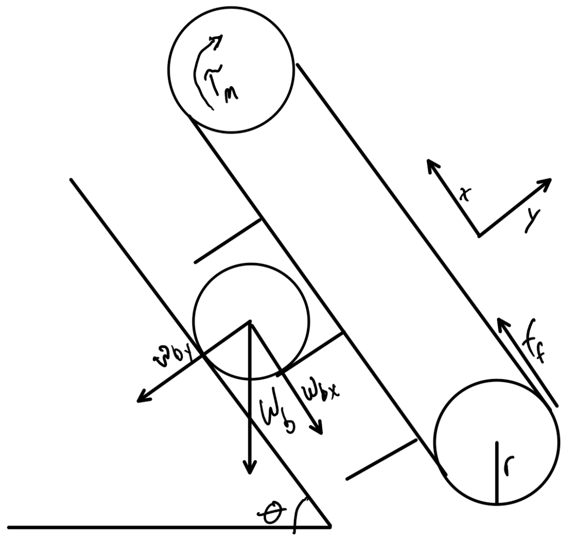


Figure 21: Conveyor Belt FBD

Equations:

$$\tau_{motor} \geq f_{friction} \cdot r_{belt} + W_{bx} \cdot r_{belt}$$

$$W_{bx} = W_b \cdot \sin(\theta)$$

Known Variables:

$W_b = 1\ oz = 0.0625\ lbs$

$f_{friction} = 0.25\ lbs$

$r_{belt} = 0.5\ in$

$\theta = 60°$

Solution:

$$\tau_{motor} \geq 0.25\ (lbs) \cdot 0.5\ (in) + 0.5\ (in) \cdot 0.0625\ (lbs) \cdot \sin(60°) \cdot 5\ (balls) = 0.26\ (in \cdot lbs)$$

$$0.26\ in \cdot lbs \leq 4.25\ in \cdot lbs\ (50\%\ stall\ torque)$$

Lift Transmission/ Torque Requirement:

The conveyor belt has the unique ability of being able to lift the belt similar to an arm. This allows us to feed both red and blue balls depending on what the situation requires. This system is powered by a torqued up vex 5 smart motor.

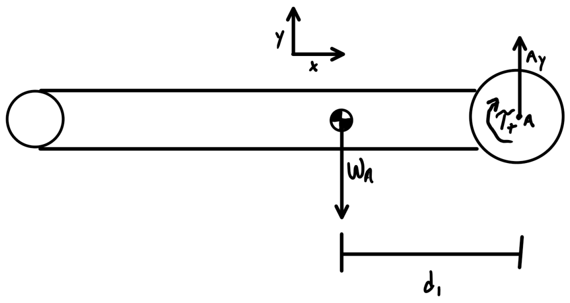


Figure 22: Conveyor Belt Arm

Equation:

$$\sum_{M_A} = 0 = W_a \cdot d_1 - \tau_{transmission}$$

$$\tau_{transmission} = \frac{\tau_{motor}}{e} \cdot \eta$$

$$\eta = 0.9^{n_{stages}}$$

Known Variables:

$W_a = 2.14\ lbs$

$d_1 = 3.75\ in$

$\tau_{motor} = 4.25\ in \cdot lbs$

$n_{stages} = 2$

Solution:

$$\tau_{transmission} = 2.14\ (lbs) \cdot 3.75\ (in) = 8.03\ in \cdot lbs$$

$$\eta = 0.9^2 = 0.81$$

$$e = \frac{4.25\ (in \cdot lbs)}{8.03\ (in \cdot lbs)} \cdot 0.81 = 0.43 \approx \frac{3}{7}$$

However, we want a decent factor of safety as our main mechanism and the possibility of using the arm to right ourselves were we to flip our robot, so let's do the math for lifting our entire robot. Let's go crazy and assume our robot is doing a curl with the arm on the edge of a cliff and the chassis hanging off of it. Ignoring the obvious tipping problem, could we get ourselves up?

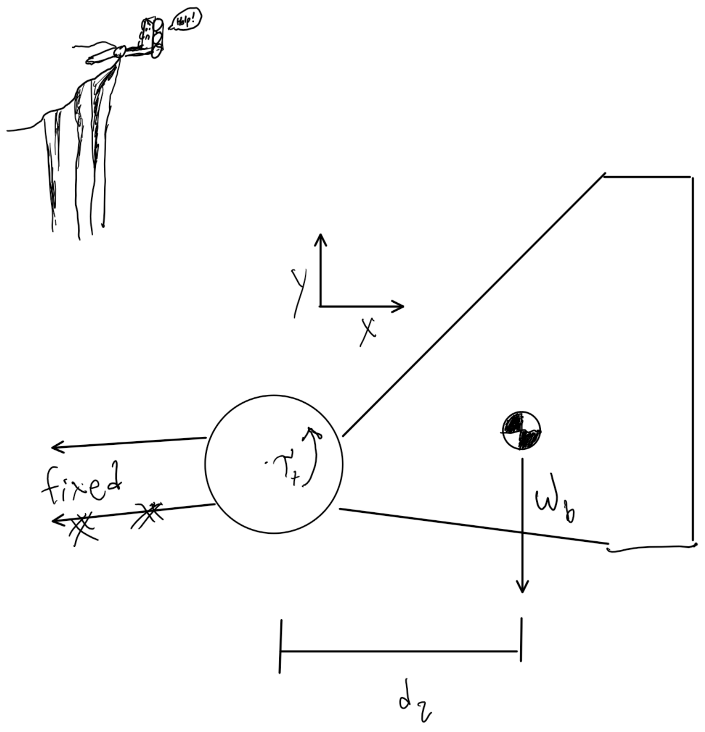


Figure 23: Robot Crunches

Equations:

$$\tau_{transmission} = W_b \cdot d_2$$
$$W_b = W - W_a$$

Known Variables:

$$W = 11.36\ lbs$$
$$W_a = 2.14\ lbs$$
$$d_2 = 12\ in$$

Solution:

$$\tau_{transmission} = (11.36\ (lbs) - 2.14\ (lbs))\ \cdot\ 12\ (in) = 110.64\ in \cdot lbs$$

So using the same eta (still aiming for two stages) and motor torque:

$$e = \frac{4.25\ in \cdot lbs}{110.64\ in \cdot lbs} \cdot 0.81 = 0.03 \approx \frac{1}{30}$$

So for a bit of safety and for nice gearing, we chose a 1:50 gear ratio.

Lift Operation Speed:

The lift operation speed is the max speed the arm can be raised or lowered, Keeping the torque below 3.5 in*lbs to maintain max RPM of 200:

$$3.5\ in \cdot lbs > \frac{110.64\ in \cdot lbs}{0.81} \cdot \frac{1}{50} = 2.73\ in \cdot lbs$$

Equation:

$$\omega_{arm} = \omega_{motor}\ \cdot\ e$$

Known Variables:

$$\omega_{motor} = 200\ rpm$$

$$e = \frac{1}{50}$$

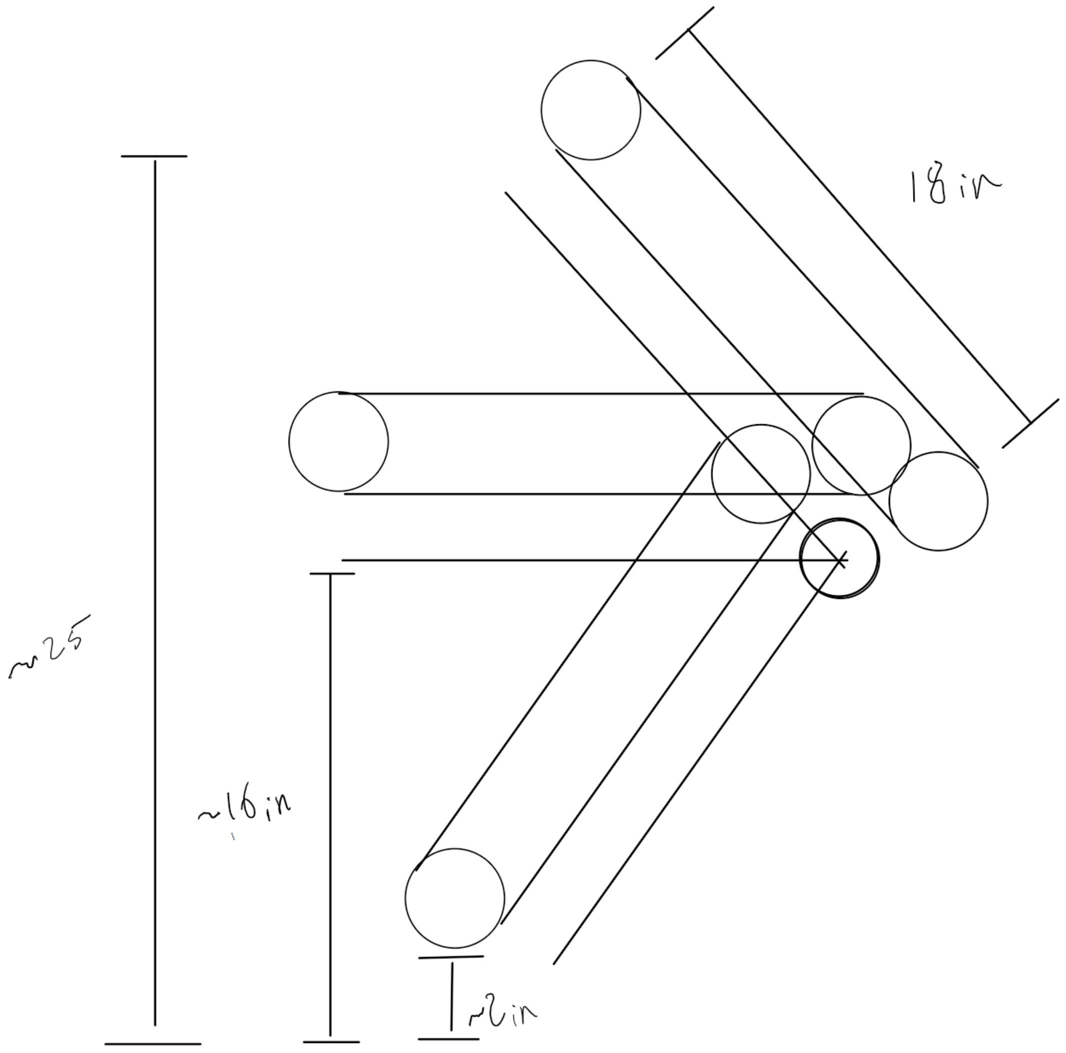
Solution:

$$\omega_{arm} = 200\ rpm\ \cdot \frac{1}{50} = 4\ rpm$$

$$= \frac{4\ rev}{min} \cdot \frac{360°}{rev} \cdot \frac{min}{60\ sec} = \frac{24°}{sec}$$

The ~50 degrees needed to cover for the red hoop will take roughly two seconds.

Lift Kinematics:



Electrical Engineering Analysis:

Below is the analysis for the sensors on the robot as well as the maximum current draw of the motors used in the design. The electrical analysis connects our mechanism to our program.

Sensor selection, Location, Utility:

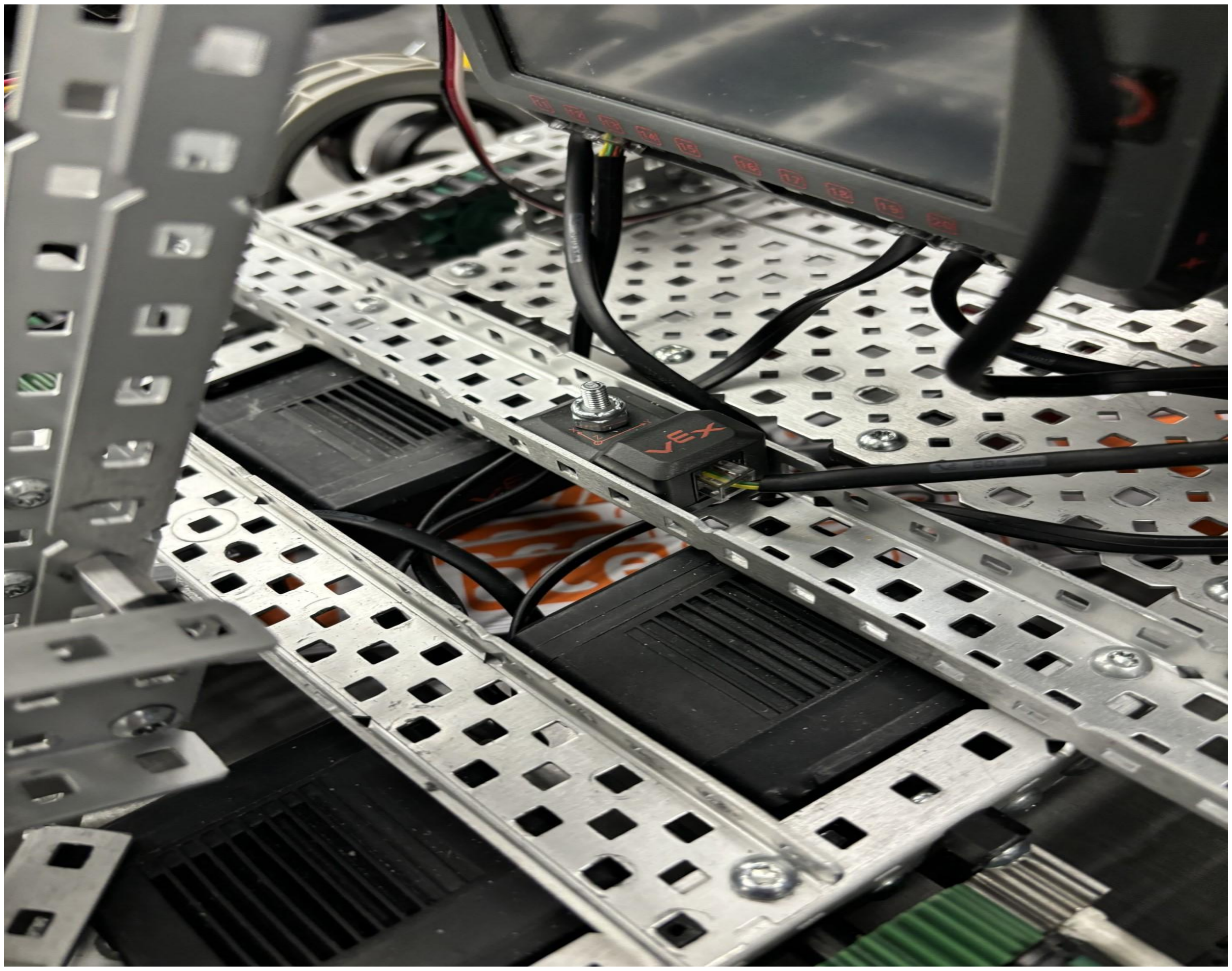


Figure 24: Inertial Sensor note figure in text needs to change

Illustrated in Figure X is the inertial sensor, strategically positioned as close to the robot's turning point as possible. This sensor serves a crucial role in our system by enabling precise angle calculations for turns during the autonomous period, eliminating the reliance on dead reckoning for turn accuracy.

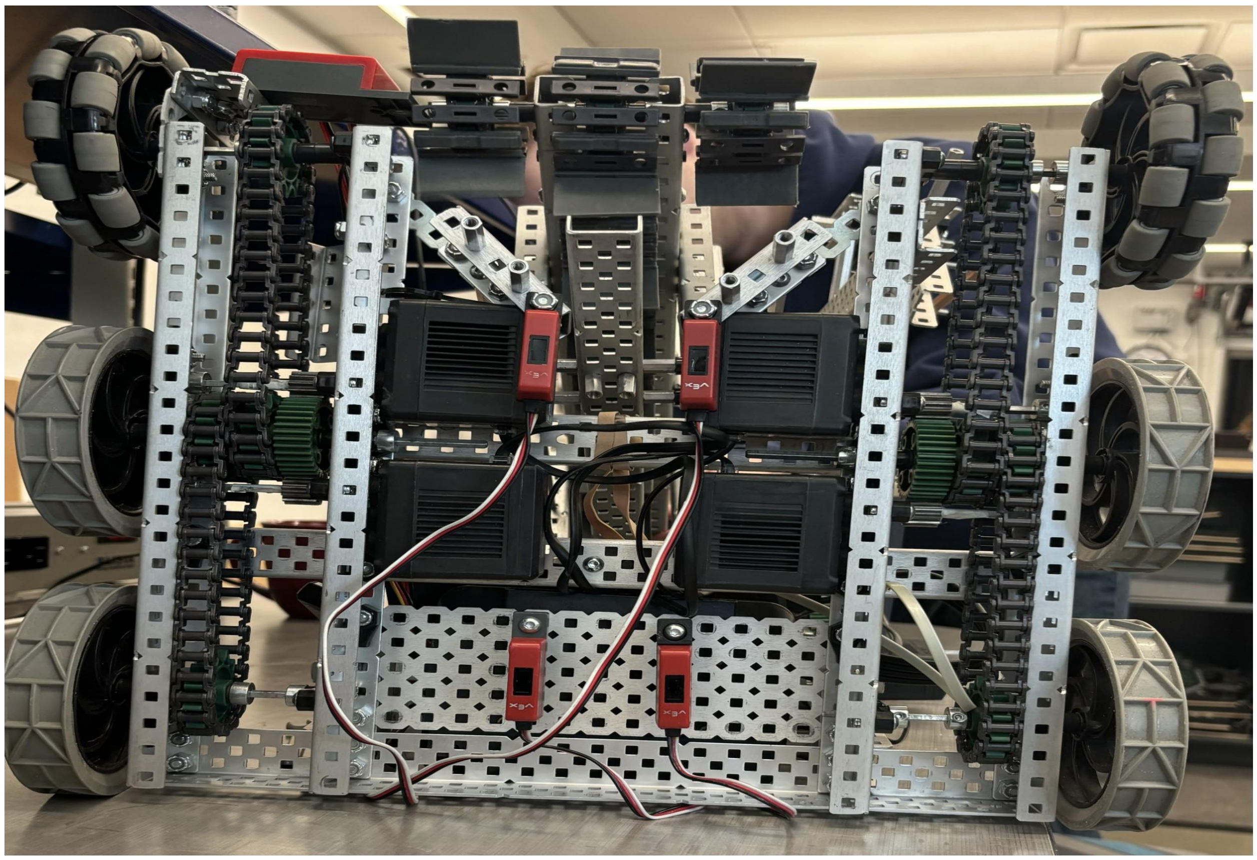


Figure 25: Line Tracker Sensors

Displayed in Figure X are the line tracker sensors, strategically positioned beneath the robot at precisely two inches apart from the center of each sensor. The specific placement at this distance is crucial, as determined during lab 2, where we observed enhanced efficiency in the line tracker program when the sensors bordered the line instead of being directly on top of it. Aligning with the field's two-inch intervals between lines, our chosen placement reinforces the effectiveness of the sensors. In our autonomous function, these line tracker sensors play a pivotal role, guiding the robot to follow the lines accurately and facilitating the collection of balls positioned along these lines. We chose two sets of two to reduce the error of either driving forward or backward.

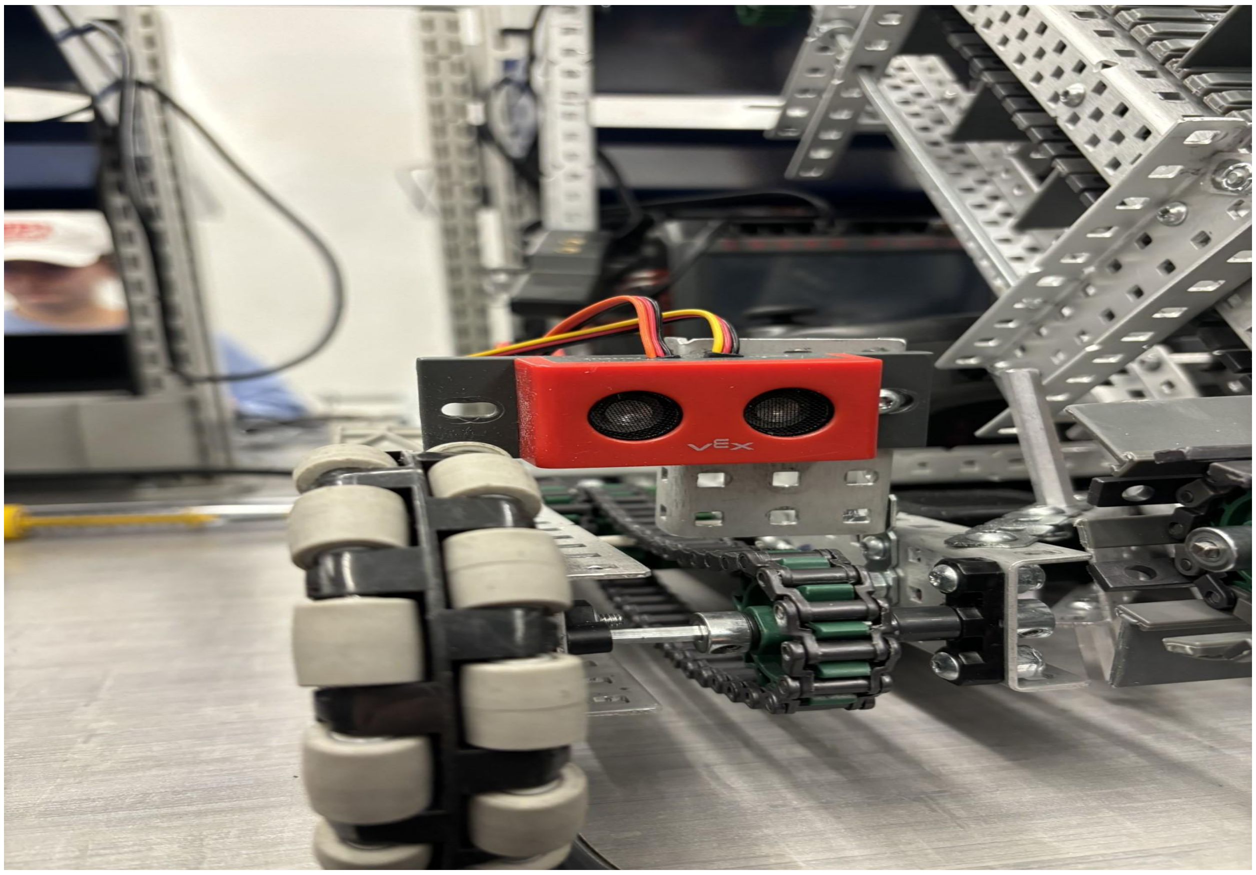


Figure 26: Ultrasonic Range Finder Sensor

In Figure X, the ultrasonic range finder is depicted, strategically mounted to the left of the front right omni wheel, positioned above the gear train. The placement above the right gear train is carefully chosen for optimal functionality. This sensor serves a crucial purpose in our system—it is employed to conclude the line tracker sensor function when the end of the line is reached. This is particularly vital as the line tracker sensor alone lacks the capability to detect the endpoint. Moreover, the sensor is strategically positioned above the top of a ball resting on the field. This placement ensures that the program is not falsely terminated due to the presence of a stray ball instead of the intended wall.

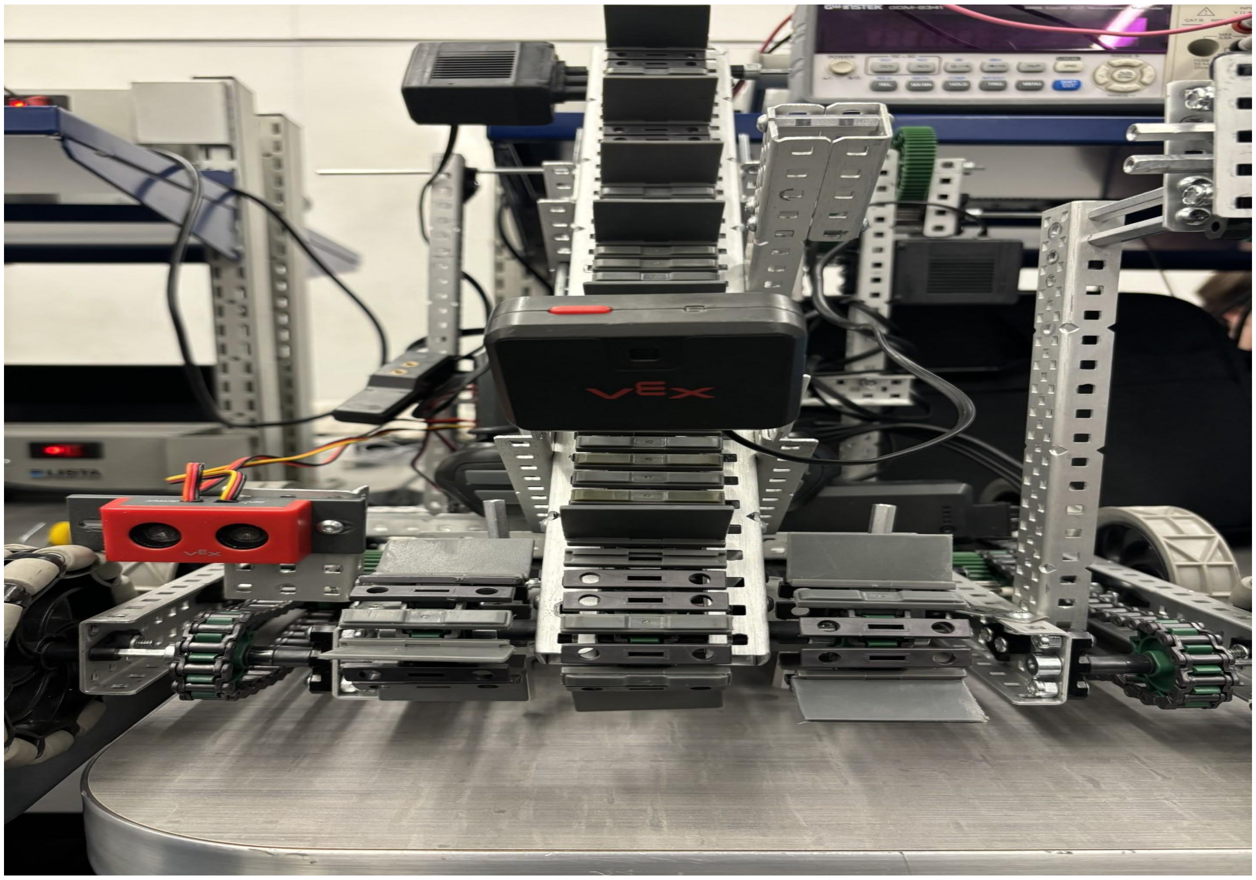


Figure 27: Vision Sensor

Displayed in Figure X is the Vision Sensor, affixed to a mount positioned above the conveyor belt. This sensor is programmed and calibrated to discern red, blue, and yellow balls based on the given command. Upon detecting the chosen color, the sensor directs the robot to rotate strategically in front of the ball, facilitating seamless collection from the intake into the conveyor belt. Designed for versatility, this mode proves invaluable in the autonomous period, enabling the robot to track and locate stray balls with adaptability to different scenarios. The program's utility extends into teleop mode as well, capitalizing on the efficiency of a well-crafted program for more effective ball collection compared to a human driver.

Maximum Current Draw (all systems):

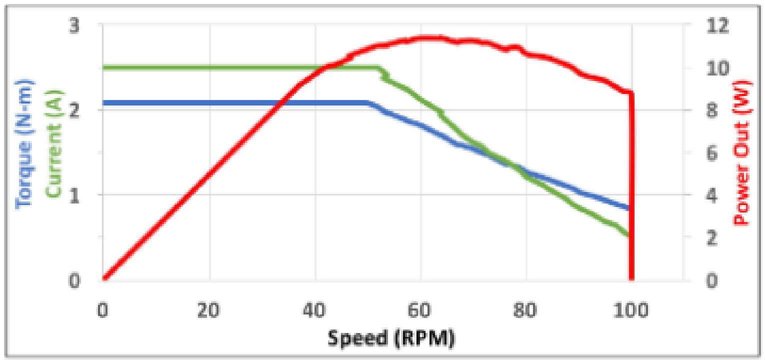


Figure 28: Vex V5 motor Torque, Current, and Power Graph

Similar to speed the current draw of a motor is directly related to the torque through the figure above, so to determine the max current draw we will assume each motor is outputting its max torque at once. Luckily, we've previously calculated all of those.

Drive Motors:

$$\tau_{motor} = \frac{25.0\ (in \cdot lbs) \cdot \frac{1}{3}}{4} = 2.08\ in \cdot lbs$$

Conveyor Motor:

$$\tau_{motor} = 0.25\ (lbs) \cdot 0.5\ (in) + 0.41\ (lbs) \cdot 0.6\ (in) = 0.37\ in \cdot lbs$$

Arm Motor:

$$3.5\ in \cdot lbs > \frac{110.64\ in \cdot lbs}{0.81} \cdot \frac{1}{50} = 2.73\ in \cdot lbs$$

In each of these cases the torque is below 3.5 in*lbs and therefore run at 200 rpm. At 200 rpm the current used is 0.5 amps. With all of our motors using 0.5 amps our total max current is 0.5 * 6 or 3 amps.

Programming Analysis:

The programming of this robot was approached with a heavy emphasis on defined functions and usage of sensors. The main code itself is divided into two main states: the autonomous function and the tele-operational state. We made use of line sensors to assist in driving the robot along straight, predictable paths and as a means of accounting for error introduced in its more nuanced functions. One of those more complicated functions is something we call ball hunt. Using the vision sensor, the robot identifies the closest ball to it and uses the coordinate feedback of the sensor to move the robot into position to intake the ball. While this function does help to account for the random variation in the placement of balls on the field, it also introduces that same randomness to the robot's position, necessitating that we use means such as the line and ultrasonic sensors to help the robot correct its position. The wide array of functions we created allowed for a streamlined writing of both autonomous and tele-op code, allowing us to order the functions like building blocks and quickly adjust variables during testing. We also put focus onto the driving experience of the robot, creating toggle functions for the intake and braking so that drivers wouldn't have to hold down key buttons throughout the match. We also used the ball hunt functions to assist the driver, allowing them to be activated with a button press during the match. This adds the steering generated by the ball hunt functions to the driver's joystick inputs, allowing the driver and the robot to more easily and precisely chase down balls in harmonious cooperation. We also have a version of the code that mirrors the autonomous turning values to allow us to start on either side of the field.

Section 8- Summary/evaluation:

After a thorough assessment of our performance during the Critical Design Review, we are pleased to report that our team executed exceptionally well. Notably, during the autonomous period, we achieved a commendable score of 18 out of 20 by successfully collecting and depositing two red balls while efficiently navigating and ascending the ramp. Our deployment of the vision sensor hunter allowed us to consistently collect red balls from various starting positions. However, our robot encountered a setback during the attempt to park on the ramp, as it struggled to maintain proper traction needed to stay securely positioned against the wall. This issue may stem from factors such as the rubber bands on the traction wheels slipping off, the weight distribution of our robot, or a miscalculation in determining the required distance for ascending the ramp. In the teleoperated phase, our performance remained smooth, showcasing our adeptness at feeding both red and blue balls through the hoops. Despite our achievements, parking the robot at the top of the ramp remained a challenge. Notably, we continued to demonstrate the effectiveness of our vision sensor hunter for programmed assisted ball collection. Taking into account extra credit and deductions for weight considerations, our final score stood at an impressive 31 out of 32. Reflecting on our journey from the initial selection of the double conveyor belt to the exploration of the flywheel, and ultimately arriving at our final design, our team is collectively satisfied with the progress made. We are proud of our accomplishments and look forward to further refining our strategies for the upcoming competition.

In the interim between the CDR and the competition, our team engaged in a thorough analysis of our performance, seeking areas of excellence and opportunities for enhancement. Recognizing the significance of winning the yellow ball during the autonomous period, we decided to implement a mechanical non-motor hook. This innovative addition aimed to provide us with a competitive edge, particularly in situations where other teams found it challenging to secure this crucial point. After evaluating our code utilized in the CDR, we set out to write a new program directing our robot to deploy the mechanical hook, pick up balls from the corner zone, feed them through the red hoop, and then repeat the process. By effectively securing the yellow ball and swiftly depositing 10 balls per minute onto the opponent's side, we aimed to accumulate 44 points during the autonomous phase. Learning from our experience at the CDR, where parking posed a challenge, we made a strategic decision to prioritize ball feeding over attempting to climb during the competition. This adjustment was geared towards maximizing our efficiency and focusing on aspects where we could excel. These targeted improvements were instrumental in better preparing our team for the final competition.

Following the competition, our performance didn't meet our expectations. While we successfully implemented a forty-point autonomous routine during testing, the robot unexpectedly veered off course at the onset of the competition, resulting in errors. We believe these errors were caused from improper field setup due to lines being taped to the field not down the center. Due to our large wheelbase this could have caused our error. Additionally, we faced the challenge of being matched against the winning robot three times. Even though we had a functioning hook, we were unable to use it due to troubles in the auto not reaching the bucket. Despite these setbacks, we found solace in our driver-controlled performance, where we consistently achieved an average of 2-4 successful payloads per round— a result that left our team content with our overall contribution.

Our final robot design stands as a resounding success for our team. Throughout the iterative design process, we meticulously refined our robot three times before arriving at the final configuration. Each team member had the opportunity to showcase their skills in designing, building, programming, driving, and presenting, contributing distinct strengths to our overall performance in the competition. In our most notable autonomous run, we secured an impressive 44 points, while during the human-operated period, we achieved a commendable 30 points in just one minute. The journey involved calculated risks, with some, like our successful redesigns, yielding positive outcomes, and others, such as the hook, presenting challenges. Our accomplishments align closely with the goals we set in section 3, and we take pride in the seamless functionality of our vision hunt feature. This innovative addition significantly enhanced our ability to track and gather balls during both the autonomous and teleoperated periods, emerging as a standout achievement for our team. Moreover, our robot exhibited remarkable versatility by effectively feeding through both the red and blue holes, setting us apart from many other teams. Despite the long hours and inevitable mistakes, this experience has equipped us with valuable skills that we can carry forward into future RBE classes. In reflecting on this competition, we're pleased to acknowledge a resurgence of the team's early camaraderie, capturing the essence of the "black magic" from our rookie years. This event served as an enriching introduction to robotics at WPI, laying a solid foundation for our ongoing journey in the field.

Section 9- Appendix:

Main robot code (auto and tele-op):

```
# ------------------------------------------------------------------------#
#
#
#   Module:       Drive_1.py
#
#   Author:       besny
#
#   Created:      12/5/2023, 5:07:35 PM
#
#   Description:  V5 project
#
#
#
#-------------------------------------------------------------------------#

# Library imports
from vex import *

# Brain should be defined by default
brain=Brain()

controller1 = Controller()

left_motor_1 = Motor(Ports.PORT11, 18_1, True)
left_motor_2 = Motor(Ports.PORT12, 18_1, True)
right_motor_1 = Motor(Ports.PORT20, 18_1, False)
right_motor_2 = Motor(Ports.PORT19, 18_1, False)

conveyor_motor = Motor(Ports.PORT1, 18_1, True)
arm_motor = Motor(Ports.PORT10, 18_1, False)

right_light = Line(brain.three_wire_port.b)
left_light = Line(brain.three_wire_port.a)
right_light_2 = Line(brain.three_wire_port.c)
left_light_2 = Line(brain.three_wire_port.d)

ultrasonic = Sonar(brain.three_wire_port.g)
```

```
robo_gyro = Inertial(Ports.PORT3)

SIG_3_YELLOW_BALL = Signature(1, 199, 457, 328, -4123, -3733, -3928,
3.000, 0)
vision_yellow = Vision(Ports.PORT9, 72, SIG_3_YELLOW_BALL)

SIG_3_BLUE_BALL = Signature (2, -3151, -2463, -2806, 11599, 14149, 12874,
2.500, 0)
vision_blue = Vision(Ports.PORT9, 72, SIG_3_BLUE_BALL)

SIG_3_RED_BALL = Signature (3, 8169, 10161, 9166, -793, -393, -594, 2.500,
0)
vision_red = Vision(Ports.PORT9, 72, SIG_3_RED_BALL)

con_off = True
drivable = True
fb_add = 0
lr_add = 0

wheelDiameter = 4.0
wheelCircumference = 3.14 * wheelDiameter
gearRatio = 3.0
wheelTrack = 14.25
degreesPerInch = 360.0 / wheelCircumference

# FUNCTIONS

def turn_angle(angle): # allows the robot to turn a designated number of
degrees in a designated direction
    degrees = angle/ 360 * (wheelTrack * 3.14) *degreesPerInch *gearRatio
    left_motor_1.spin_for(FORWARD, degrees, DEGREES, 100, RPM, False)
    left_motor_2.spin_for(FORWARD, degrees, DEGREES, 100, RPM, False)
    right_motor_1.spin_for(REVERSE, degrees, DEGREES, 100, RPM, False)
    right_motor_2.spin_for(REVERSE, degrees, DEGREES, 100, RPM, True)

def drive_left(speed, direction): # consolidates both left motors into a
single gearbox
    left_motor_1.set_velocity(speed + direction, RPM)
    left_motor_2.set_velocity(speed + direction, RPM)
    left_motor_1.spin(FORWARD)
    left_motor_2.spin(FORWARD)

def drive_left_2(speed, direction): # consolidates both left motors into a
single gearbox
    left_motor_1.set_velocity(speed + direction, RPM)
    left_motor_2.set_velocity(speed + direction, RPM)
    left_motor_1.spin_for(FORWARD, 500, MSEC, 100, RPM, False)
    left_motor_2.spin_for(FORWARD, 500, MSEC, 100, RPM, True)

def drive_right(speed, direction): # consolidates both right motors into a
single gearbox
    right_motor_1.set_velocity(speed - direction, RPM)
    right_motor_2.set_velocity(speed - direction, RPM)
    right_motor_1.spin(FORWARD)
    right_motor_2.spin(FORWARD)

def drive_right_2(speed, direction): # consolidates both right motors into
a single gearbox
    right_motor_1.set_velocity(speed - direction, RPM)
    right_motor_2.set_velocity(speed - direction, RPM)
    right_motor_1.spin_for(FORWARD, 500, MSEC, 100, RPM, False)
    right_motor_2.spin_for(FORWARD, 500, MSEC, 100, RPM, True)

def drive_main(speed, direction): # main drive function of the robot,
drives both gearboxes, accounting for a direction, also brakes
   drive_left(speed, direction)
   drive_right(speed, direction)

def detect_yellow(): # uses the camera to detect the nearest yellow
object, hopefully a ball
    objects = vision_yellow.take_snapshot(SIG_3_YELLOW_BALL)

    if (objects):
        brain.screen.print('x:', vision_yellow.largest_object().centerX, '
y:',  vision_yellow.largest_object().centerY, '   width:',
vision_yellow.largest_object().width)

def hunt_yellow(): # Robot will assist drive function by correcting with
variables determined by the vision sensor
```

```
    global fb_add
    global lr_add
    detect_yellow()
    while vision_yellow.largest_object().centerY -190 <= 0:
        detect_yellow()
        fb_add = vision_yellow.largest_object().centerY - 250
        lr_add = vision_yellow.largest_object().centerX-184
        wait(20)
    fb_add = 0
    lr_add = 0

def detect_blue(): # uses the camera to detect the nearest blue object,
hopefully a ball
    objects = vision_blue.take_snapshot(SIG_3_BLUE_BALL)

    if (objects):
        brain.screen.print('x:', vision_blue.largest_object().centerX, '
y:',  vision_blue.largest_object().centerY, '   width:',
vision_blue.largest_object().width)

def hunt_blue(): # Robot will assist drive function by correcting with
variables determined by the vision sensor
    global fb_add
    global lr_add
    detect_blue()
    while vision_blue.largest_object().centerY -190 <= 0:
        detect_blue()
        fb_add = vision_blue.largest_object().centerY - 250
        lr_add = vision_blue.largest_object().centerX-184
        wait(20)
    fb_add = 0
    lr_add = 0

def detect_red(): # uses the camera to detect the nearest red object,
hopefully a ball
    objects = vision_red.take_snapshot(SIG_3_RED_BALL)

    if (objects):
        brain.screen.print('x:', vision_red.largest_object().centerX, '
y:',  vision_red.largest_object().centerY, '   width:',
vision_red.largest_object().width)

def hunt_red(): # Robot will assist drive function by correcting with
variables determined by the vision sensor
    global fb_add
    global lr_add
    while vision_red.largest_object().centerY -190 <= 0:
        detect_red()
        fb_add = vision_red.largest_object().centerY - 250
        lr_add = vision_red.largest_object().centerX-184
        wait(20)
    fb_add = 0
    lr_add = 0

def hunt_red_2(): # Auto version of the hunt_red function that includes a
function to look for a ball if it doesn't find one and drives within the
function
    global fb_add
    global lr_add
    objects = vision_red.take_snapshot(SIG_3_RED_BALL)
    while objects == None:
        drive_left_2(-100,0)
        wait(500)
        objects = vision_red.take_snapshot(SIG_3_RED_BALL)
        drive_right_2(-100,0)
        wait(500)
        objects = vision_red.take_snapshot(SIG_3_RED_BALL)

    while vision_red.largest_object().centerY -190 <= 0:
        detect_red()
        fb_add = vision_red.largest_object().centerY - 250
        lr_add = vision_red.largest_object().centerX-184
        drive_main(fb_add, lr_add)
        wait(10)
    fb_add = 0
    lr_add = 0
```

```
def light_drive(speed, wall_distance): # uses light sensors to follow a
line at a given speed
    kp=1
    left_light.reflectivity()
    right_light.reflectivity()
    ultrasonic.distance(DistanceUnits.IN)
    wait(500)

    while(ultrasonic.distance(DistanceUnits.IN) > wall_distance):
        brain.screen.print(left_light.reflectivity())
        error = right_light.reflectivity()-left_light.reflectivity()
        drive_main(speed, error*kp)

def light_drive_back(speed, time): # uses light sensors to follow a line
    kp=1.3
    left_light_2.reflectivity()
    right_light_2.reflectivity()
    ultrasonic.distance(DistanceUnits.IN)
    wait(500)
    x=0

    while(x <= time):
        # brain.screen.print(left_light.reflectivity())
        error = right_light_2.reflectivity()-left_light_2.reflectivity()
        drive_main(speed, error*kp)
        wait(1)
        x+=1

def gyro_drive(speed, angle, distance): # uses a gyro to drive in a
designated direction until a given distance away from an object
    angle = angle*math.pi/180
    while(ultrasonic.distance(DistanceUnits.IN) > distance):
        error = math.sin(angle -
robo_gyro.heading(RotationUnits.REV)*2*math.pi)
        drive_main(speed, 60*error)

def gyro_turn(angle): # uses a gyro to turn to face a designated angle
    # angle = angle*math.pi/180
    diff = angle/360 - robo_gyro.heading(RotationUnits.REV)
    while abs(diff) > 0.3:
        brain.screen.print_at(diff, x = 40, y = 40)
        diff = angle/360 - robo_gyro.heading(RotationUnits.REV)
        # error = robo_gyro.heading(RotationUnits.REV)*2*math.pi - angle
        # drive_main(0, 60*error)
        drive_main(0,60)

def R1(): # Function for Conveyor toggle
    global con_off
    if(con_off):
        conveyor_motor.spin(REVERSE, 100, PERCENT)
        wait(20)
        con_off = False
    else:
        conveyor_motor.stop(COAST)
        wait(20)
        con_off = True

def R2(): # Function for Conveyor toggle
    global con_off
    if(con_off):
        conveyor_motor.spin(FORWARD, 100, PERCENT)
        wait(20)
        con_off = False
    else:
        conveyor_motor.stop(COAST)
        wait(20)
        con_off = True

def full_stop(): # Function for braking toggle
    global drivable
    if(drivable):
       wait(20)
       drivable = False
    else:
        wait(20)
        drivable = True

def follow_thru(): # Function to add a forward and backward motion to
intake balls
    drive_main(-100, 0)
    wait(1500)
    drive_main(100, 0)
    wait(1500)

#AUTO CODE

loop = True
while (loop):
    ultrasonic.distance(DistanceUnits.IN)
    robo_gyro.calibrate()
    wait(20)
    robo_gyro.set_heading(180)
    arm_motor.reset_position()
# drive forward
    drive_main(-100,0)
    wait(1900, MSEC)
# turn right
    turn_angle(-90)

    for i in range(0,2):
    # line follow to red ball and collect ball and stop with ultra
        R1()
        light_drive(-175,7)
    # back up and turn a little for better placement to collect from
placing section
        drive_main(200, 0)
        wait(500)
        turn_angle(-30)

    # repeatedly collect placed balls
        follow_thru()
        follow_thru()
        drive_main(-100, 25)
        wait(2000)
        R2()

    # turn toward hoop
        turn_angle(102)
        drive_main(-200, 0)
        wait(1000)
    # raise arm to desired height to deposit red balls
        while arm_motor.position(DEGREES) < 360*6.5:
            arm_motor.spin(FORWARD, 200, RPM, False)
    # start line following to hoop while raising arm
        light_drive(-175, 4)
        left_motor_1.stop(BRAKE)
        left_motor_2.stop(BRAKE)
        right_motor_1.stop(BRAKE)
        right_motor_2.stop(BRAKE)
        arm_motor.stop(BRAKE)
        wait(500)
    # conveyor outtake
        R2()
        wait(2000)
        R2()
    # lower the robot arm back to starting position
        while arm_motor.position(DEGREES) > 0:
            arm_motor.spin(REVERSE, 200, RPM, False)
    # Line follow backwards while lowering the arm
        light_drive_back(200, 2750)
    # turn to get back into position
        turn_angle(-100)
    # drive back a tad
        drive_main(150, 0)
        wait(1000)

# TELE-OP CODE

    loop = False

controller1.buttonR1.pressed(R1) # toggle control for Conveyor
controller1.buttonR2.pressed(R2) # toggle control for Conveyor
controller1.buttonDown.pressed(full_stop) #toggle control for brakes

controller1.buttonY.pressed(hunt_yellow) # manual activation of ball hunt
for a yellow ball
controller1.buttonB.pressed(hunt_blue) # manual activation of ball hunt
for a blue ball
controller1.buttonA.pressed(hunt_red) # manual activation of ball hunt for
a red ball
```

```
while(True):
    fbm = controller1.axis3.position()
    lrm = controller1.axis4.position()
    drive_main((-2*fbm) + fb_add, (1.5*lrm) + lr_add) # function to drive
the robot, adds output variables from the ball hunting functions when they
are active

    if drivable == False: # brakes the robot drive wheels based on the
toggle function
        right_motor_1.stop(BRAKE)
        right_motor_2.stop(BRAKE)
        left_motor_1.stop(BRAKE)
        left_motor_2.stop(BRAKE)

    if(controller1.buttonL1.pressing()): # manual control for arm motor
        arm_motor.spin(FORWARD, 125, RPM)
    elif(controller1.buttonL2.pressing()):
        arm_motor.spin(REVERSE, 125, RPM)
    else:
        arm_motor.stop(COAST)

    wait(20, MSEC)
```

Right side autonomous code:

```
#RIGHT SIDE AUTO CODE

loop = True
while (loop):
    ultrasonic.distance(DistanceUnits.IN)
    robo_gyro.calibrate()
    wait(20)
    robo_gyro.set_heading(180)
    arm_motor.reset_position()
# drive forward
    drive_main(-100,0)
    wait(1900, MSEC)
# turn right
    turn_angle(90)

    for i in range(0,2):
    # line follow to red ball and collect ball and stop with ultra
        R1()
        light_drive(-175,7)
    # back up and turn a little for better placement to collect from
placing section
        drive_main(200, 0)
        wait(500)
        turn_angle(30)

    # repeatedly collect placed balls
        follow_thru()
        follow_thru()
        drive_main(-100, -25)
        wait(2000)
        R2()

    # turn toward hoop
        turn_angle(-102)
        drive_main(-200, 0)
        wait(1000)
    # raise arm to desired height to deposit red balls
        while arm_motor.position(DEGREES) < 360*6.5:
            arm_motor.spin(FORWARD, 200, RPM, False)
    # start line following to hoop while raising arm
        light_drive(-175, 4)
        left_motor_1.stop(BRAKE)
        left_motor_2.stop(BRAKE)
        right_motor_1.stop(BRAKE)
        right_motor_2.stop(BRAKE)
        arm_motor.stop(BRAKE)
        wait(500)
    # conveyor outtake
        R2()
        wait(2000)
        R2()
    # lower the robot arm back to starting position
        while arm_motor.position(DEGREES) > 0:
            arm_motor.spin(REVERSE, 200, RPM, False)
    # Line follow backwards while lowering the arm
        light_drive_back(200, 2750)
    # turn to get back into position
        turn_angle(100)
    # drive back a tad
        drive_main(150, 0)
        wait(1000)
```