Engineering Notebook

الاتحاد السعودي للروبوت والرياضات اللاسلكية
The Saudi Federation for Robotics & RC Sports

KRT1

Team Number

KRT

Team Name

King Fahd University of Petroleum and Minerals (KFUPM)

School

V4.0 Date 3.27.25

VEX ROBOTICS

KFUPM
جامعة الملك فهد للبترول والمعادن
King Fahd University of Petroleum & Minerals

REC FOUNDATION

Table of Contents

KFUPM Robotics Team
Presents:
Engineering Notebook

Team Layout

The team's layout was decided from beginning of the season and it is as follows:

- Dr. Hassan Al-Mubarak Assistant Professor at the Control and Instrumentation Engineering department acts as the general supervisor of the team and directs the team to the major goals that we need to accomplish.
- Mohsen Al-Zamzam is a senior student majoring in Control and Instrumentation Engineering. He acts as the team's coach and helps guide the team in day-to-day building and troubleshooting of the robots.
- The team was split into two groups both tasked to one of the two robots with one group working on Theeb and the other working on Sleekh.
- This was made so to increase the coordination and focus of the team by making sure the two can work simultaneously and having clearer goals when it comes to deciding on the building and strategy of the robots.
- Work on the robots was assigned to the team members with regard to their various skills whether that be hardware or software related. With being assigned leadership positions due to some having previous experience in VEX competitions. The assignment of team members is as follows in Page 5.

Team Layout

Team Sleekh

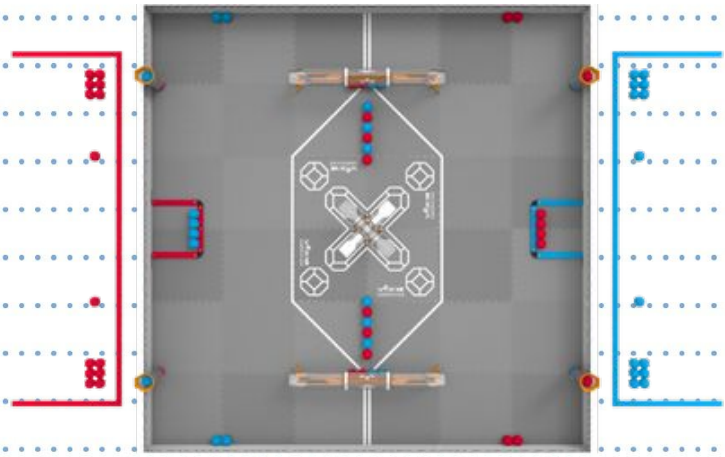
NAME	ROLE
Montather Alkhabaz	Leader
Yousef Qashqri	Hardware & Documentation
Ahmed Nasser	Software Developer & Documentation
Hashim Al-Sadah	Hardware & 3D
Amin Sraj	Software
Ali Al-Shaikhi	Hardware
Hashim Khalifah	Hardware

Team Theeb

NAME	ROLE
Mohammed AlKadhim	Leader
Hashem Altujar	Hardware & Notebook
Ahmed Hanafy	Software Developer & Documentation
Abdulmajeed Alsulimani	Hardware
Abdulmajeed Alfaifi	Hardware & 3D
Majed Alhammadi	Hardware & CAD Documentation
Ahmed Alfaraj	Hardware & Documentation
Asad Al-Dubaisi	Software Team Lead

KRT Match Strategy Plan

Autonomous Period Strategy



The optimal steps to win the autonomous round is the following:

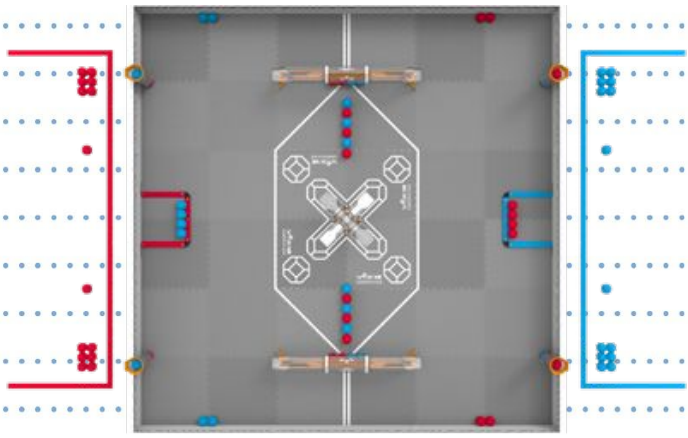
1. Fight for the vertical balls in the middle for set amount of time
2. Fill the middle goal 3 balls in upper & lower
3. Go to loader
4. Load from the loader
5. Fill the long goal with 6 balls
6. Descore
7. Return to parking
8. Best case total 88 Points + 10 Autonomous Points

The key in the plan is to adapt to the enemy's behavior.

Based on our number of balls collected from the middle. Which means if we have 5 balls from the middle this displays that our opponent is not that competitive in autonomous because he wasn't able to take our balls, therefore we play the best we could.

In worst case scenario we will command the robots to change their behavior and sacrifice some steps to ensure a good number of points in autonomous.

Autonomous Period Strategy



Backup Plan:

1. Robot 1 fill lower goal with 2 balls
2. Robot 2 fills lower goal with 1 ball
3. Robot 1 & 2 goes to loader
4. Fill long goal with 1 or 2 balls and descore
5. Descore until there is small time
6. Fill all balls in long and descore
7. (Optional) Park

This makes a total of X points in best case with a priority to save the goals.

The question is how are we going to evaluate which steps to sacrifice and which to keep and what should we do if the opponent is not very competitive but also not an easy one ?.

We decided to go with the plan of **evaluation function** which should be set by programmers when testing robots to set timeouts in seconds and the probable points (outcome or reward) return in certain steps. Which makes our robot adapt to enemy's behavior and decide best course of action autonomously.

Autonomous Period Strategy

Plan Weakness & Mitigation

Problem: The plan requires both robots to go in the middle and fight for the balls which takes time and might not get a good result.

Mitigation: We should make a totally different plan for the selected opponents and let this be only for unknown and new opponents because it has the best chance.

Problem: What if one robot collected 5 and the other 3 then there is a conflict in the overall behavior of the plan.

Mitigation: We should explore about robots communication if possible we could adjust that, otherwise this is a real issue for the total plan.

Problem: Sometime the enemy is tough and we might know previously that fighting for the middle is hard and might not have a lot of reward if we kept the evaluation function fixed. Then let's say ten seconds is the time for the middle fight and if we only get two balls, this will result in a time loss instead we might be able to take two balls in just the first five seconds of the game.

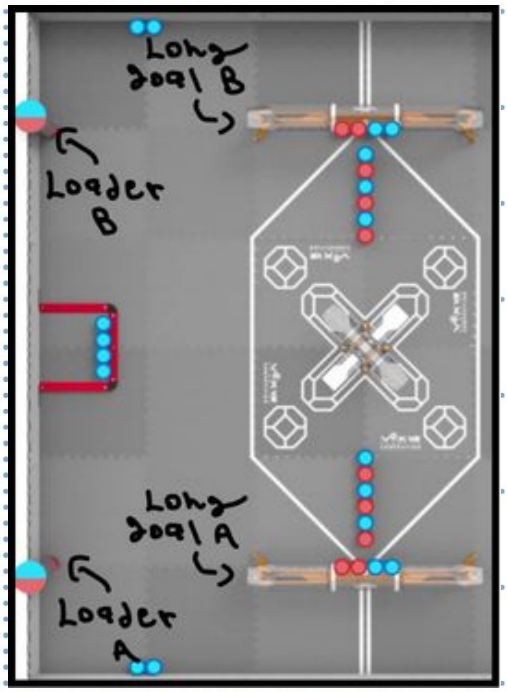
Mitigation: If we were able to make the robot collect data in the game and adjust to it that would be great. The reality for now that we can't in short time so we should have multiple programs with different evaluation values to make robot prioritize certain behavior when needed.

Driver Controlled Period

In the Driver Control period, our strategy focuses on these key points:

- Compete for every available block on the field.
- Score efficiently across all four goals without the two robots interfering with each other.
- Secure all control zones by either descoring opponent blocks or filling all goals with our blocks.
- Follow a plan that remains achievable within the 90 seconds' period.
- Keep the workflow easy to drive and execute with the controller.
- Confirm that robots designs are mechanically capable of performing the documented plan.

Firstly, to indicate which are the loaders (A&B) and the long goals (A&B):



Driver Controlled Period

This driver control period strategy is used when the autonomous plan works perfectly, so we can continue with the same successful approach under driver control.

Steps:

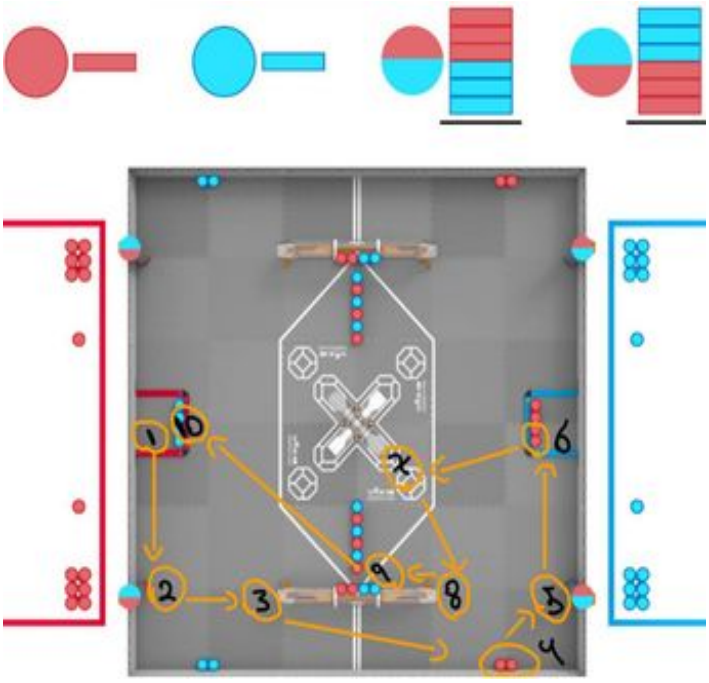
Step Number	Robot A Actions	Robot B Actions	Notes
1	Exists from the parking area smoothly	Go quickly to the loader	The initial position of both robots (A&B). A: Start inside the parking area B: Start near Long Goal A initial position = final position of the autonomous period blocks in storage: Robot A: 0 Robot B: 0
2	Load 6 blocks from the loader A	Load 6 blocks from the loader B	The loaders should be unloaded during autonomous. Each loader should be loaded by 6 blocks during control period. blocks in storage: Robot A: 6 Robot B: 6
3	Score 6 blocks in the long goal A	Score 6 blocks in the long goal B	We should fill both long goals to maximize our points. blocks in storage: Robot A: 0 Robot B: 0

4	Go to the 2 blocks that are next to the field wall	Go to the 2 blocks that are next to the field wall	We should exploit every block in the field to maximize our points. blocks in storage: Robot A: 2 Robot B: 2
5	Collect the 3 blocks from the opponent's loader A.	Collect the 3 blocks from the opponent's loader B.	This step does not mean that it is necessary that our blocks in the opponent's loaders, it can be around the loaders. blocks in storage: Robot A: 5 Robot B: 5
6	Go to the opponent's parking area and collect 2 blocks only.	Score 4 of 5 blocks in the upper goal	The paths of both robots must be different, as it is extremely difficult for two robots to collect all 4 blocks located within a confined area without causing interference or time loss. blocks in storage: Robot A: 7 Robot B: 1
7	Score 4 of 7 blocks in the lower goal	Go to the opponent's parking area and collect the remaining 2 blocks	Switch the roles, to grantee that interference doesn't happen. blocks in storage: Robot A: 3 Robot B: 3

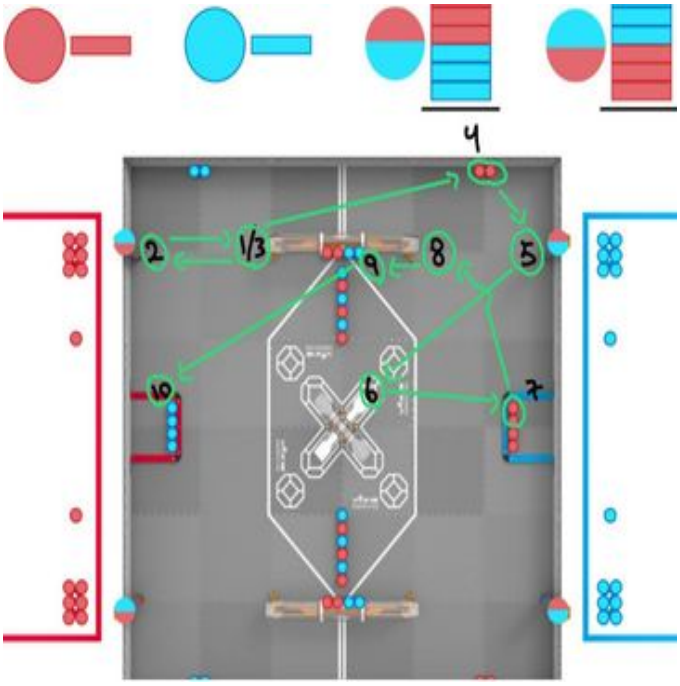
8	Score the remaining 3 blocks in the long goal A	Score 3 blocks in the long goal B	We should fill the long goal as much as possible to ensure we have the control zone and to maximize our points. blocks in storage: Robot A: 0 Robot B: 0
9	Decore	Decore	This step isn't always necessary; it is necessary in these cases: Case 1: The control zone in any goal for the opponent team. Case 2: The most blocks in the goals for the opponent team. blocks in storage: Robot A: 0 Robot B: 0
10	Go to the parking area	Go to the parking area	We will maximize our points If both robots go the parking (30 pints). blocks in storage: Robot A: 0 Robot B: 0

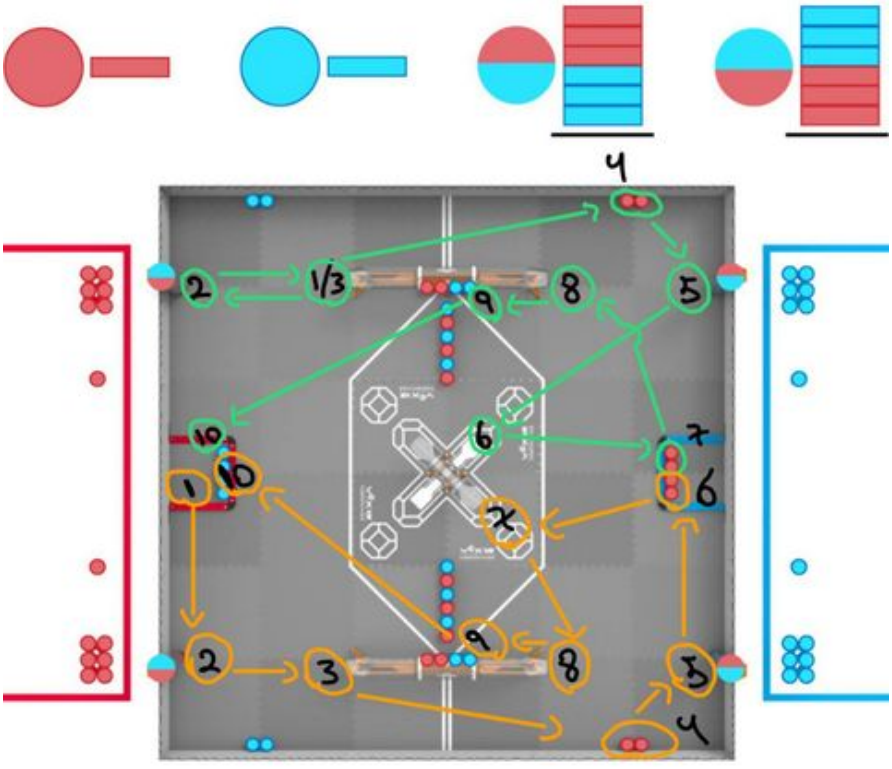
These figures are for clarifying the steps of each robot, not for showing actual robot motion.

Robot A Movement:



Robot B Movement:





Scoring in control period			
Goal	Scored Blocks	Control Zone	Scored Points
Long Goal A	9	yes	27
Long Goal B	9	yes	27
Upper Goal	4	yes	12
Lower Goal	4	yes	12
Parking	Both Robots, 30 Points		
Total points = 142			

Scoring in control period + autonomous period			
Goal	Scored Blocks	Control Zone	Scored Points
Long Goal A	15	yes	55
Long Goal B	15	yes	55
Upper Goal	7	yes	29
Lower Goal	7	yes	27
Parking	Both Robots, 30 Points		
Total points = 196 + 10 (Autonomous point)			

Q1: What happens if the Autonomous phase does not perform ideally, or if opponent robots attempt to disrupt our strategy?

A1: We will allocate additional driver practice time to prepare for unexpected disruptions and develop flexible counter strategies for different scenarios, ensuring adaptability during the match.

The strategy for both competition phases (Autonomous and Driver Control) requires that Robots A and B meet the following design and capability requirements:

- Both robots must fit within the parking zone at the same time, so length and width are critical.
- Both robots must be able to pass under the long goals to collect blocks that are under them, requiring careful height design.
- Robots must operate at high speed while maintaining stability and avoiding tipping or oscillation.
- Robots must be capable of scoring at all goal heights.
- Robots must efficiently collect blocks from both loaders and the ground.
- Robots must support effective descoring when interacting with opponent goals.
- Robot A must have a minimum storage capacity of 7 blocks, and Robot B a minimum of 6 blocks, to ensure uninterrupted execution of the strategy.

Q2: What is another benefit of designing the robots to pass under the long goal?

A2: Passing under the long goal improves robot mobility and flexibility on the field and reduces the likelihood of being trapped or disrupted by opponents, allowing for better repositioning and continued scoring.

Q3: Which robot is Slee5 and which is Theeb?

A3: Based on the required storage capacities, Theeb is assigned as Robot A with at least a 7-block capacity, while Slee5 is assigned as Robot B with at least a 6-block capacity.

Q4: Is the color sensor beneficial? and should both robots include it?

A4: Yes, a color sensor can be very useful, in fact game changing. As observed in the field layout, the middle 12 balls are mixed up together in alternating color. Having a color sensor that immediately sorts through the balls at the start of the autonomous round could lead us to having a 6 ball advantage ahead of the enemy.

KRT Software Design

Software Design Philosophy

The software for this VEX U project was designed with reuse, scalability, and future AI integration in mind. Instead of creating a competition-specific codebase, the system was structured so it can be preserved and extended for VEX AI development. This allows work done during VEX U to directly contribute to more advanced, agentic control systems without requiring a full redesign.

The core design principle is modularity. Sensor drivers were separated into independent components responsible for data acquisition and processing, allowing sensors to be added or modified without affecting higher-level logic. Control systems such as PID controllers were implemented as reusable modules, enabling consistent and tunable control across multiple subsystems.

Filtering layers were added between sensors and controllers to reduce noise and improve stability. By standardizing sensor outputs before they are used by control logic, the system becomes more reliable and easier to integrate with higher-level decision-making algorithms.

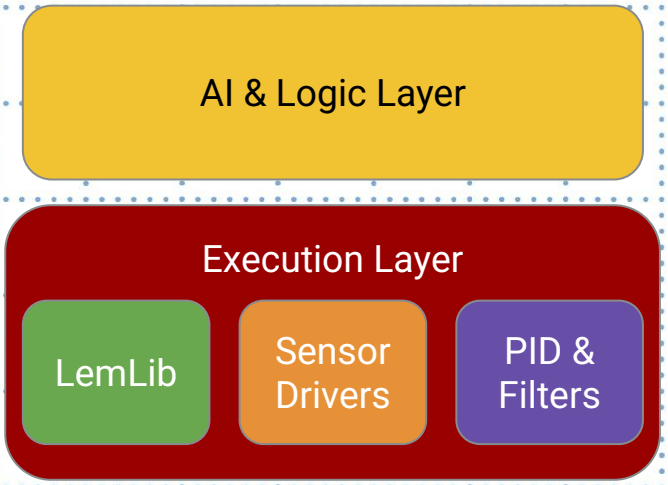
The architecture also supports pushback compatibility between VEX U and VEX AI. Low-level motor control is decoupled from high-level behavior, making it easier to transition from scripted logic to agent-based or AI-driven control in the future.

Overall, this software design emphasizes clarity, reuse, and long-term value. By focusing on modular components, clean abstractions, and forward compatibility, the codebase provides a strong foundation for both competitive performance and future AI integration.

Software Design Architecture

Our main idea for this year to design our robot's software is to separate the software into two-layers, execution and logic. The goal of this approach the following :

- Reduced time for developing new techniques in the field since all control functions of the robot to move and collect and make action is already there
- Ease the process of developing and integrating AI to our robot
- Make the robot more robust in autonomous to handle problems and dynamically act accordingly
- Developing the strategy algorithm independent of the execution layer which make it more time-effective during the development



One important note of this design is that the execution layer should be able to give signal to the logic layer if any command fails so that we could make our robot make decision correctly.

Robot’s Sensors:

Programming robot in autonomous period would be tedious if we do not have the right combination of sensors so after thinking about the problem we have decided the following:

Position Tracking:

The robot have to determine where it’s position in the field to make the proper actions correctly and this fusion of sensors should be robust where if any one of them failed the others should take care of the problem.

- Vex GPS Sensor
- Vex IMU Sensor
- PMW3091 Optical Flow Sensor

The reason behind this combination of sensors is that Vex GPS provides accurate readings but slow which make another problem.

What if we were in time competition for neutral zone in the autonomous period how do we adjust and fix robot’s behavior to adapt fast ?. Here where PMW3901 optical flow sensor. This sensor provide less accurate measurements for position but can report position offset very quickly and could be reliable in short distances, combining those two sensors where GPS correct the position periodically to and Optical Flow reports position between those reading intervals would make the position more accurate and could adapt in fast-reaction situations.

Finally, the IMU Sensor is a sensor would help the robot to make accurate turns and additionally it could be combined to measure the sanity of GPS and Optical Flow reading’s with respect to acceleration and direction.

Action-Specific Sensors:

Those sensors are needed in action to perform certain task like taking a ball and filtering the team balls from enemy's one. For those two tasks we are using :

- Vex AI Vision Sensor
- Vex Optical Sensor

The AI Vision would help for navigation to get balls when it reaches certain locations, and the color sensor would filter the opposite team balls if they got inside the robot.

Realtime Sensor Opreations

A major software challenge in this project was enabling multiple advanced sensors to operate simultaneously in a reliable and synchronized manner. The AI Vision sensor, GPS sensor, and Optical sensor each provide critical information for autonomous decision-making, but they differ significantly in data rates, processing demands, and communication behavior. Running these sensors at the same time introduced issues related to timing conflicts, blocking execution, and inconsistent data availability.

Early approaches revealed that treating sensor updates sequentially or within a single execution loop limited performance and reduced system responsiveness. Delays from one sensor could negatively impact others, which is unacceptable for autonomous control where real-time perception and feedback are required. This problem became more pronounced as the system scaled toward more complex, AI-driven behaviors.

Through experimentation and evaluation of different software approaches, it became clear that a solution was required that could reliably handle concurrent sensor operation while maintaining clean integration with the rest of the control system. The resolution to this challenge was the adoption of PROS, which provided the necessary capabilities to support simultaneous sensor processing and stable system behavior.

This decision allowed the AI Vision sensor, GPS sensor, and Optical sensor to function together as intended, forming a consistent and dependable perception layer for the robot.

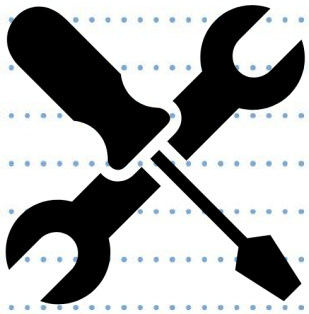
PROS Operating System

PROS is a free and open-source operating system that allows programmers using the **VEX V5 Brain** to access advanced functionalities of the brain. One of the main features of PROS that surpasses writing C++ with **VEXcode** is that it fully supports **object-oriented programming** and allows the programmer to **fragment the code into multiple files**, making large projects easier to organize and maintain.

However, that is not all. Another major feature provided by PROS is **Tasks**, which allow the VEX V5 Brain to perform multiple jobs at the same time. This makes it possible for certain code to run in the background while other parts of the program continue executing. These tasks are managed and scheduled automatically by the PROS operating system.

The key utility of **Tasks** is the ability of polling from the needed sensors installed on robot while also controlling and moving the robot and this fit our need in term of position tracking and objects finding with robot's sensors.

Robot's Services



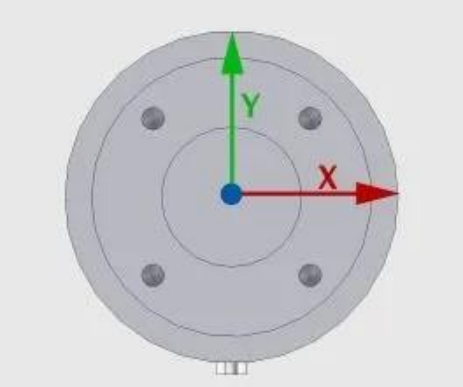
The key utility of Tasks is the ability to continuously poll sensors installed on the robot while still allowing the robot to be controlled and moved normally. This fits our needs for real-time position tracking and object detection using the robot's sensors without blocking other parts of the program.

Using PROS Tasks, an **AI Vision Sensor service** was implemented to constantly detect balls on the field and determine which one is closest to the robot. This service runs in the background and provides updated target data that can be used by other parts of the code at any time.

A **position tracking service** was also created to continuously update the robot's location and orientation. By running as a separate task, the robot always has access to live position data, which is essential for accurate navigation and autonomous behavior.

Finally, an **optical sensor service** was implemented to filter balls entering the intake system. This task continuously checks the color of detected balls and allows the robot to reject enemy balls while accepting friendly ones, all without interrupting driver control or autonomous logic.

Motion Control



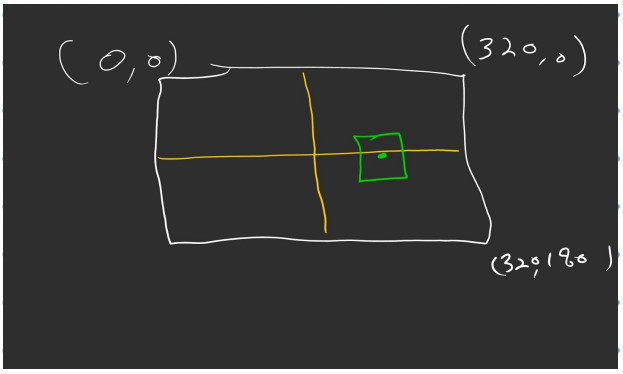
Another significant challenge in this project was achieving accurate, repeatable robot motion and drivetrain control. Autonomous tasks require the robot to move precise distances, maintain headings, and follow paths consistently despite variations in battery voltage, friction, and mechanical tolerances. Simple time-based or open-loop motor commands proved unreliable and led to drift, overshoot, and inconsistent positioning.

As autonomous routines became more complex, these inaccuracies compounded. Small errors in heading or position early in a routine resulted in large deviations later, making reliable navigation difficult. Additionally, integrating drivetrain control with higher-level logic highlighted the need for predictable motion behavior that could be reused across multiple autonomous actions

.

Testing showed that a more structured approach to drivetrain control was necessary—one that could combine sensor feedback, motion constraints, and control algorithms into a unified system. Addressing this problem was essential not only for competitive autonomous performance but also for future compatibility with AI-driven navigation and decision-making systems.

AI Vision Driver



An AI Vision Driver was implemented to detect game objects and estimate their real-world distance using the AI Vision Sensor. This allows autonomous and AI-based logic to make decisions based on vision data rather than fixed positions.

The AI Vision Sensor reports the detected object's width in pixels within a 320-pixel-wide image. With a known horizontal field of view of 61 degrees, this pixel width can be converted into an angular width. Since the real diameter of the game object is known (3.25 inches), the distance from the camera to the object can be estimated using geometric relationships between angular size and physical size.

As an object moves closer to the robot, its detected pixel width increases; as it moves farther away, the width decreases. This relationship enables distance estimation without the need for additional range sensors. The AI Vision Driver encapsulates detection, measurement, and calculation into a single reusable component, providing higher-level systems with clean distance estimates instead of raw image data.
Although vision-based distance estimation is affected by noise and lighting conditions, it provides sufficiently consistent results for autonomous targeting and decision-making when combined with filtering. Overall, this driver forms an important perception layer that supports intelligent motion control and future agentic AI behaviors

LemLib as Motion Controller & Tracker

To solve the drivetrain accuracy and navigation consistency problems, we adopted LemLib as the robot's motion controller and tracking system. The main objective was to create a reliable motion layer that could accurately estimate robot position while also executing movement commands precisely and repeatably.

As a **tracker**, LemLib provides a structured way to combine drivetrain sensors (such as encoders and inertial measurements) into a consistent estimate of the robot's pose (position and heading). This matters because autonomous performance depends on knowing where the robot actually is, not just what it was told to do. Without accurate tracking, small drift in heading or distance quickly compounds into large positioning errors.
As a motion controller, LemLib enables closed-loop drivetrain movement and path following. Instead of controlling the drivetrain using raw motor power or timed movements, actions can be expressed as targets such as driving to a point, turning to a heading, or following a path. LemLib continuously corrects error using feedback, improving repeatability across battery levels, field conditions, and mechanical variation.

This approach also improves software organization. **High-level autonomous logic** can focus on what the robot should do, while LemLib handles how the drivetrain achieves it. That separation makes routines easier to tune, easier to reuse, and more compatible with future AI-driven navigation since higher-level systems can depend on a stable, predictable motion layer.
Overall, using LemLib as both a tracker and motion controller strengthened autonomous reliability, reduced compounding error, and provided a scalable foundation for more advanced decision-making and agentic control.

LemLib Lateral and Angular Motion

LemLib provides a structured and efficient way to control robot movement by separating motion into **lateral (linear)** and **angular (rotational)** components. This separation allows for precise navigation and smoother control, especially in autonomous robot behavior.

Lateral Motion

Lateral motion refers to the robot's movement forward and backward along a straight line. In LemLib, this type of motion is responsible for controlling how far the robot travels and how accurately it reaches a target distance.

To calculate lateral movement, LemLib relies on sensors such as:

- Tracking wheels

These sensors provide real-time feedback about the robot's position, allowing the system to continuously adjust motor output. Lateral motion control focuses on:

- Accurate distance tracking
- Smooth acceleration and deceleration
- Reducing wheel slip and overshoot

This is especially important in autonomous routines where the robot must move between predefined points reliably and consistently.

Angular Motion

Angular motion controls the robot's rotation around its center, allowing it to change direction or maintain a specific heading. LemLib typically uses sensors such as:

- IMU (Inertial Measurement Unit)

to measure the robot's orientation.

Angular motion is essential for:

- Turning to precise angles
- Correcting heading drift during movement
- Maintaining alignment with field elements

By continuously monitoring the robot's angle, LemLib ensures that the robot rotates smoothly and stops at the intended orientation without unnecessary oscillation.

Combining Lateral and Angular Motion

One of LemLib's key strengths is its ability to combine lateral and angular motion. This allows the robot to move toward a target position while simultaneously correcting its heading.

This integration results in:

- Smoother paths
- Higher positional accuracy
- Better performance in complex autonomous routes

By managing both distance and direction together, LemLib enables advanced navigation behaviors that are difficult to achieve with basic motion control systems.

LemLib Odometry:

We use LemLib's pose tracking primitives (lateral + angular motion fusion) for robust odometry. Odometry combines high frequency relative motion (tracking wheels / optical flow) with absolute or slow correcting sensors (IMU, GPS) so the robot keeps an accurate pose estimate during autonomous routines.

Why this design ?
Tracking wheels / optical flow provide fast local displacements (good for short bursts and quick reactions).
IMU gives reliable heading for angular corrections.

Implementation summary:
Hardware: 2 tracking wheels (left, right & forward, backward) + 1 IMU. with GPS, run it in the background and apply corrections periodically.
On match start, set an initial pose with chassis.setPose(START_X, START_Y, START_HEADING).

Tuning & calibration checklist:
Calibrate tracking wheel distances and counts-per-inch.
Calibrate IMU heading offsets.

To wrap up, We implement odometry using LemLib's lateral + angular motion model. A background tracking task fuses high rate relative measurements (tracking wheels or optical flow) with heading from the IMU and periodic GPS corrections to produce a live Pose(x,y,heading). The pose is published to LemLib's chassis so autonomous motions, operate on a continuously updated estimate. We maintain pose sanity checks, perform periodic pose resets at known landmarks, and tune encoder/IMU scale factors.

LemLib PID Control

PID (Proportional–Integral–Derivative) control is a fundamental control algorithm used in LemLib to regulate both lateral and angular motion. PID controllers allow the robot to correct errors between its current state and its desired target.

Proportional (P)

The proportional term produces an output that is directly related to the current error. The larger the error, the stronger the correction applied.

- Improves response speed
- Helps the robot react quickly to large errors
- Excessive P values may cause oscillation

Integral (I)

The integral term accumulates error over time. It is mainly used to eliminate steady-state error, which occurs when the robot stops slightly short of the target.

- Corrects long-term inaccuracies
- Useful when friction or load prevents full movement
- High I values may reduce stability

Derivative (D)

The derivative term predicts future error based on the rate of change of the current error. It helps dampen motion and reduce overshoot.

- Improves smoothness
- Reduces oscillations
- Enhances stopping accuracy

PID Usage in LemLib

In LemLib, PID controllers are applied to both lateral and angular motion. The PID constants are tuned based on factors such as:

- Robot weight
- Wheel type and size
- Gear ratios

Proper PID tuning allows the robot to move efficiently, stop accurately, and maintain stability during motion. This makes PID control a critical component of LemLib's motion system.

Motion Chaining:

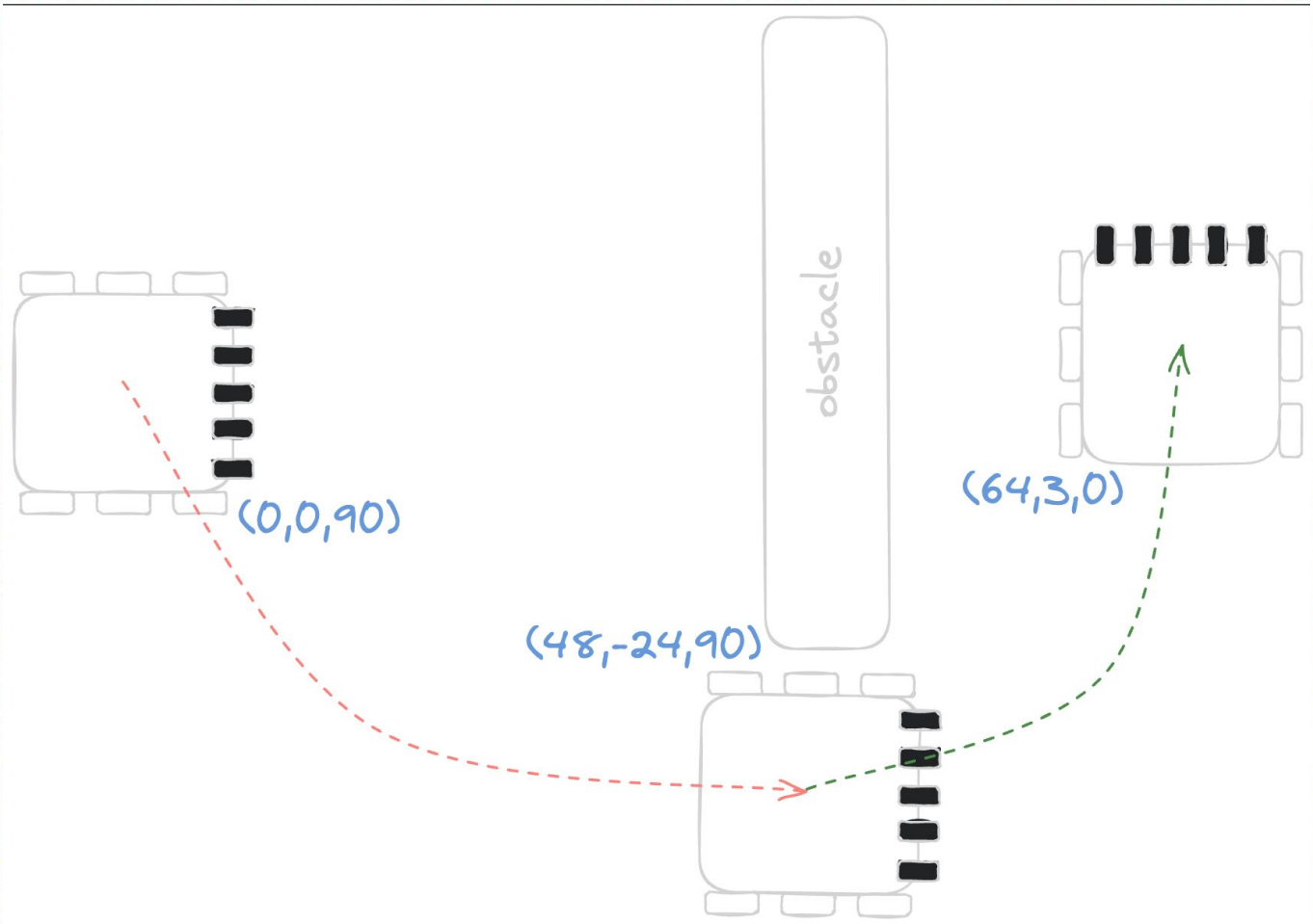
Motion chaining is the process of linking multiple LemLib motions (moveToPoint / moveToPose / swingToHeading) so the robot does not come to a full stop between segments. This trades a small loss in terminal accuracy for significantly faster traversal ideal for match time path tweaks and high speed game routines.

Integration notes with our two-layer architecture:

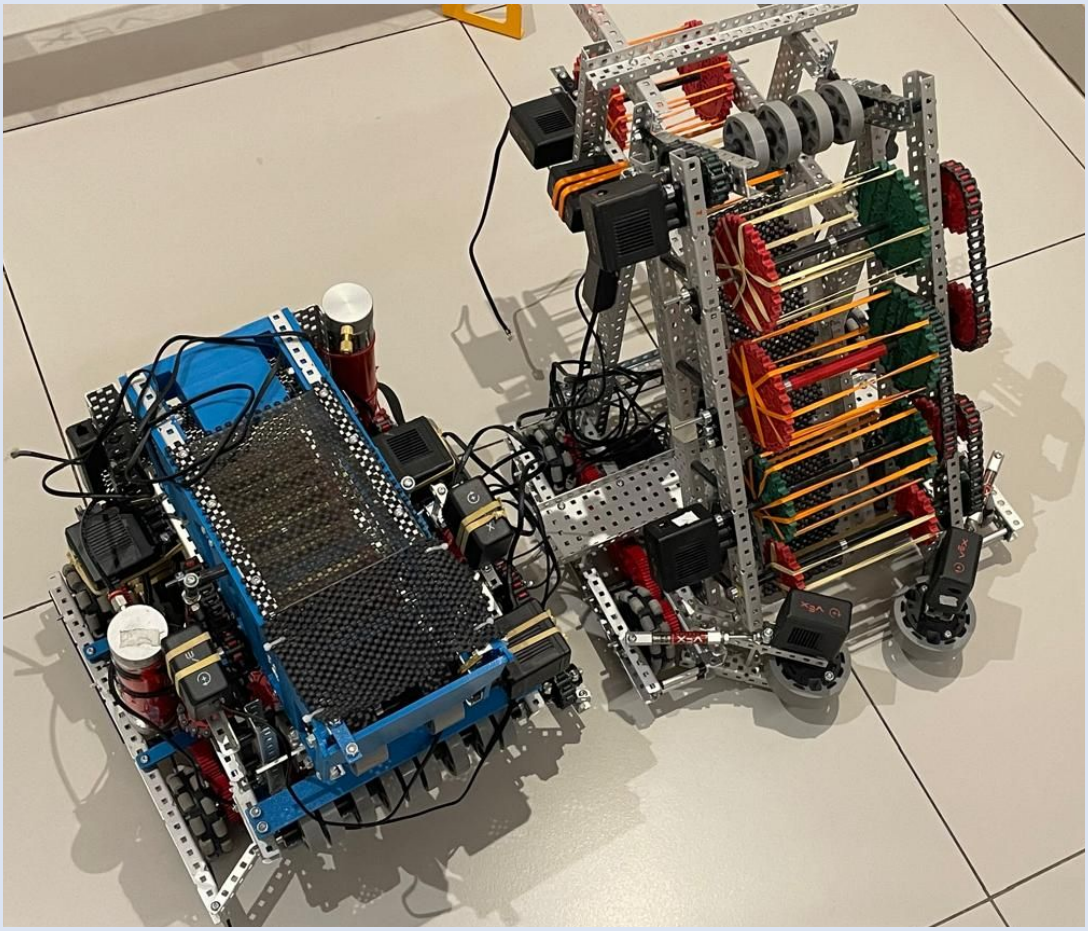
Execution layer handles the chained motions and parameters. Logic layer (strategy) decides which chained segments to schedule and when to cancel.

Safety & testing checklist:

- Test each chain in isolation on field mockup.
- Measure scoring accuracy vs speed for multiple minSpeed values.
- Ensure chassis is in motion returns reliable status before relying on it.
- Provide a safe cancel that gracefully stops motion controllers.

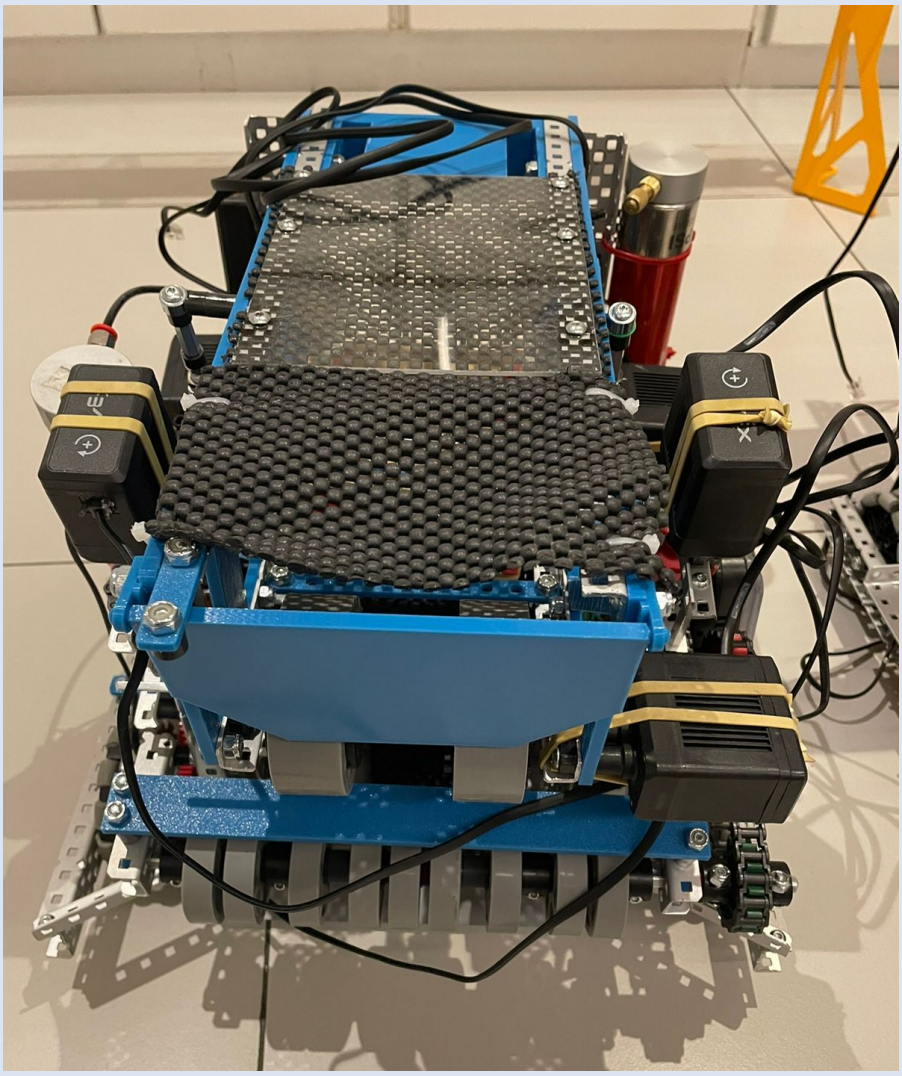


The Robots: Sleekh and Theeb



Sleekh (left) and Theeb (Right) Comparison

Sleekh



Sleekh (front view)

Sleekh Design

The original Sleekh plan was as follows:

1- The desired robot (Slee5) dimensions are 14'x 14.5' x 12.5'. We are aiming to reduce the height of the robot as much as possible while having the maximum possible balls capacity, so the robot can pass under the long goal without any hassle.

2- Slee5 will be driven by 4 x 3.25' omni-wheels using 4 x 11W motors with blue cartridges and 1:1 36t gears (600 rpm), and 2 x 2' omni-wheels are used for tracking purposes. We could add 2 x 3.25' traction-wheels for more stability, but we don't have them currently.

3- The base dimensions are(later). At the front of the base there are 2 acute angles which helps in the intake so that the balls doesn't get scattered when the robot is taking them, also it helps near the arena walls.

4- The intake is made of 9 vertical 2' flex wheels with a little movement in the upper direction. The path of the balls is shaped like C or U sideways, the upper part of the path height is adjustable using 2 x 50mm stroke pneumatic cylinder

Sleekh Design

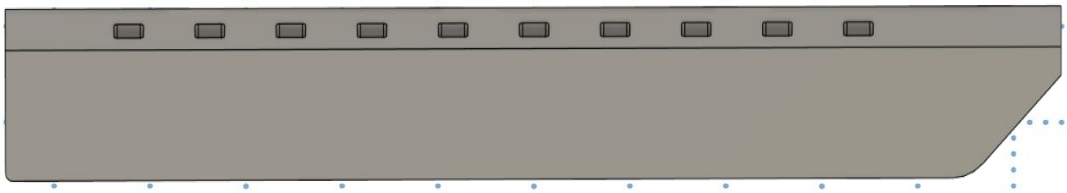
Our robot **Sleekh,** has been initially designed with the following features:

Base: H-Drive. We used an H-drive for Sleekh's base and it included a 1:1 gear ratio with 36T and 600RPM consisting of four motors and four omnidirectional wheels.

Four motors were used since we tried six motors and it made no substantial difference in the acceleration of the robot.

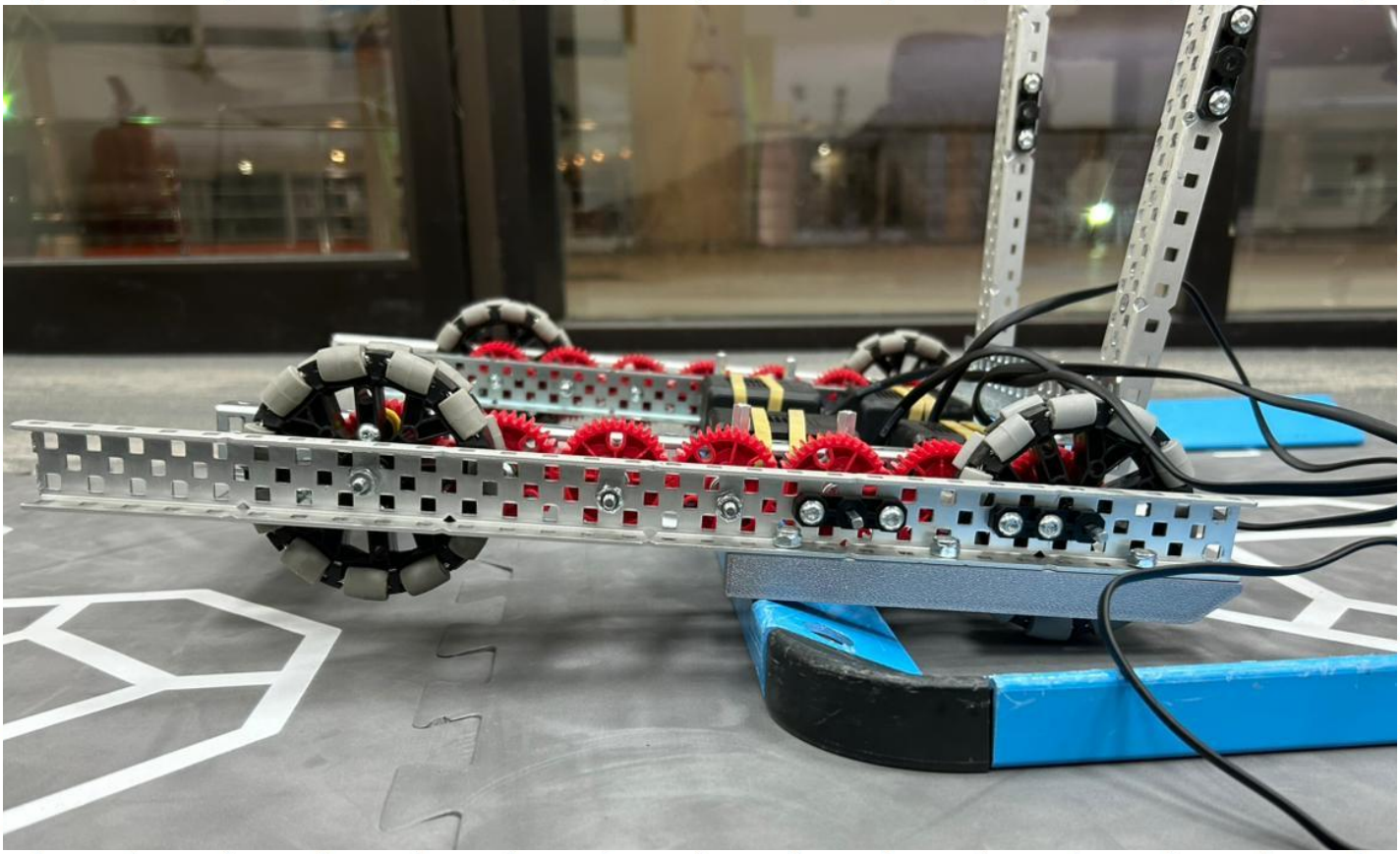
Omnidirectional wheels were used since we wanted the robot to glide seamlessly and easily on the ground. We found omnidirectional wheels gave the best traction and gliding ability without sacrificing much in terms of stability. To add onto that point we tested out six wheels with four being omnidirectional and two being traction wheels. We found out that this variation of the base did not glide as easily as we wanted and instead it made the robot more rigid on the ground.

A 3D-printed piece was modeled to make parking the robot in the parking space much more easier. Without this piece the parking walls would catch on the gears of the base. This piece allowed the parking walls to not touch the gears and make parking Sleekh much more easier.



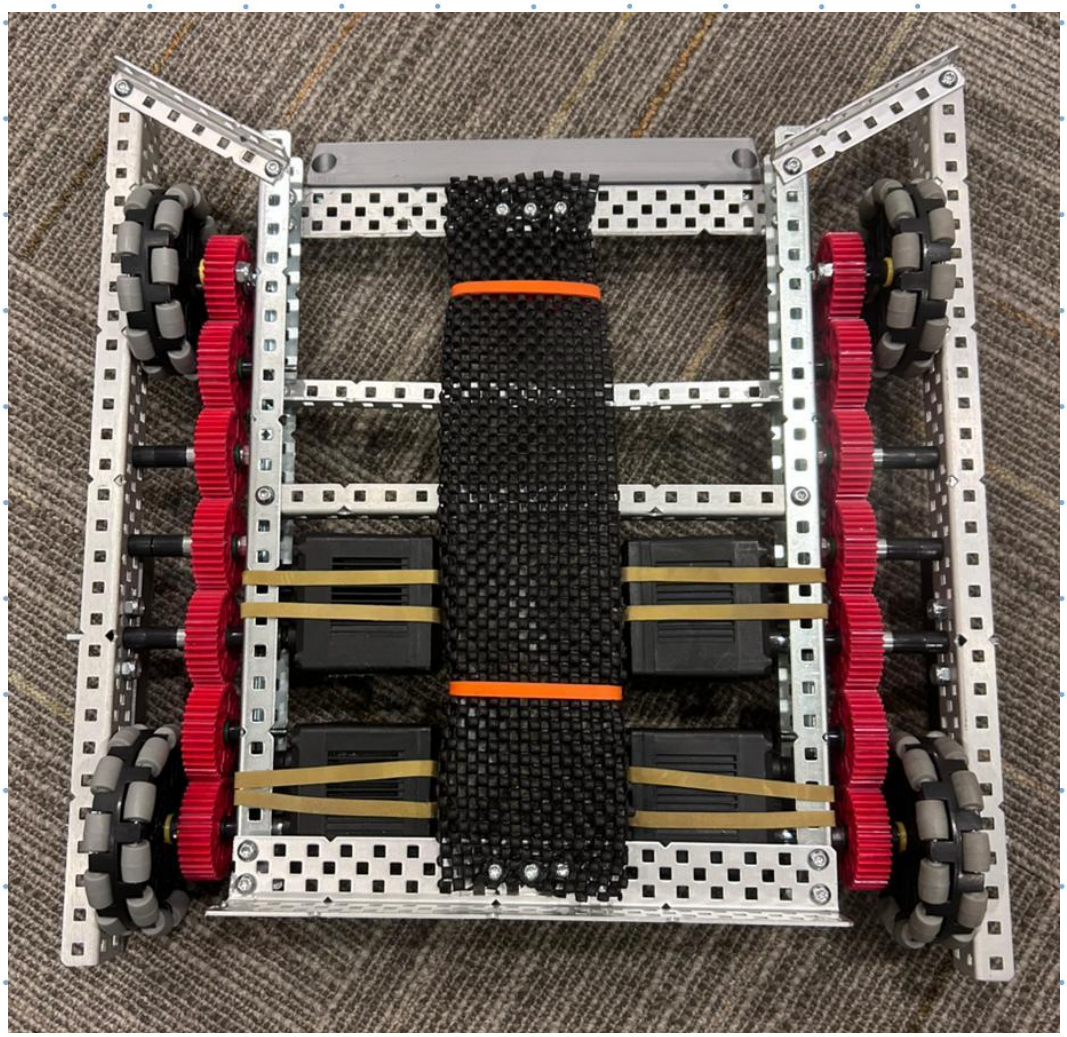
3D-Modeled piece

Sleekh Design



This photo displays how the 3D printed piece can ease the robot's entry into the parking area

Sleekh Design



Completed base with acrylic piece and 3D printed ramp

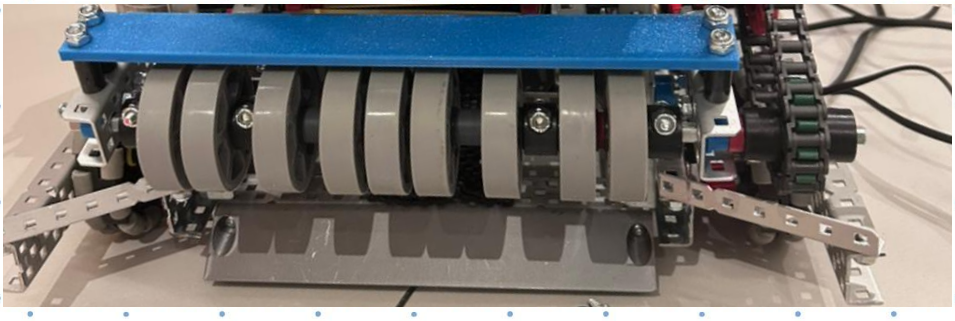
To guide the blocks into the base we 3D modeled and printed a ramp. In the center of the base we molded an sloped acrylic base and placed a rubber mat, secured with rubber bands, on top of it. Its purpose is to guide the blocks up along their way in the C-Channel.

Sleekh Design

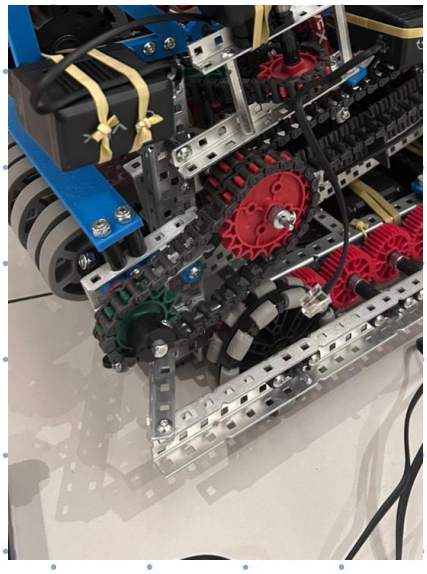
For the intake we used a shaft with nine vertical flex wheels that were positioned on a hinge that would give the shaft the ability to so move up an inch to allow for flexibility in case of an block being stuck at the intake. This solution gives our robot a bit of margin of error.

Originally we had chosen only five flex wheels as the intake method. This proved to be inefficient since blocks would get stuck in the large spaces between the the flex wheels. It was found that increasing the number of flex wheels to a total of 9 made this issue occur much less.

The shaft is connected to a gear which is connected to a chain that links multiple gears to one motor. This was done to reduce the amount of motors in our robot.



Picture of the shaft with the flex wheels



Picture of the hinge and the gears and chain

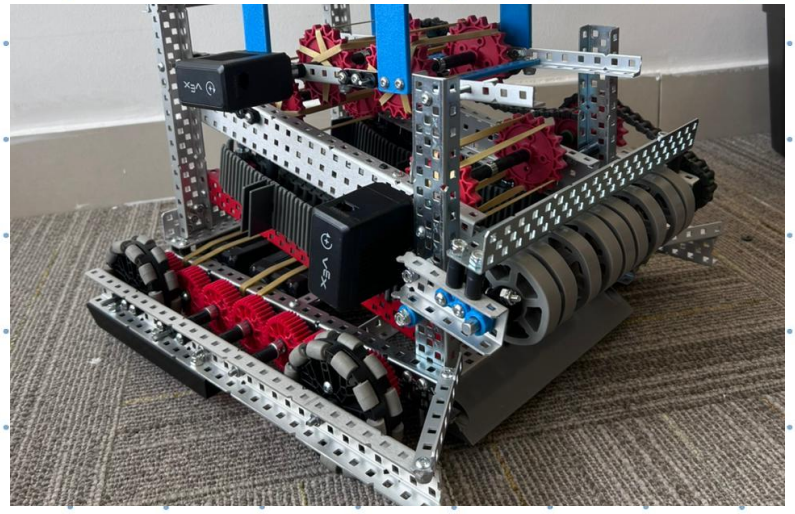
Sleekh Design

Sleekh was designed to incorporate a C-Channel storage and scoring system. It was decided that this system was the most optimal choice since we wanted Sleekh to be short enough to pass under the long goal for strategic purposes that allowed our robot to be more agile and faster to respond to challenges.

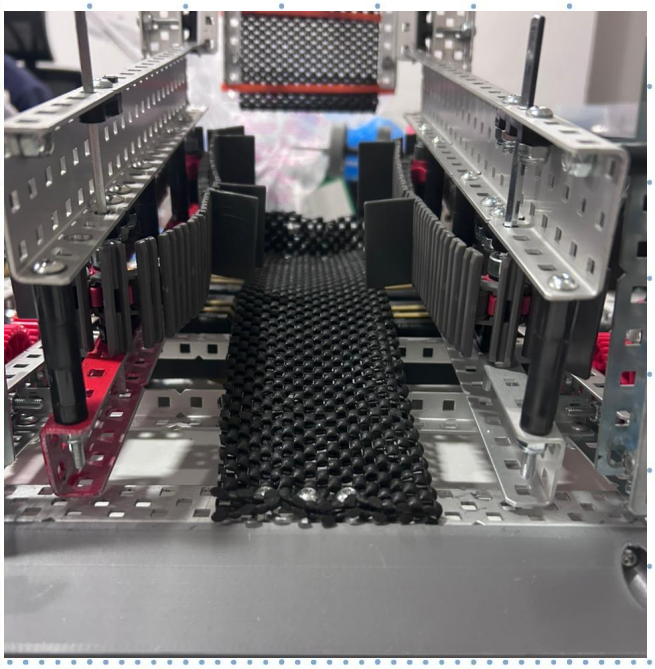
The C-Channel system was the shortest out of the other options we thought of and those were the S-Channel and Z-Channel designs.

Our C-Channel design is consisted of two layers. One of these layers would fluctuate from being lifted up or down by a pneumatic system. This has the purpose of being able to score in both the long goal and middle goal without causing the robot to be too tall.

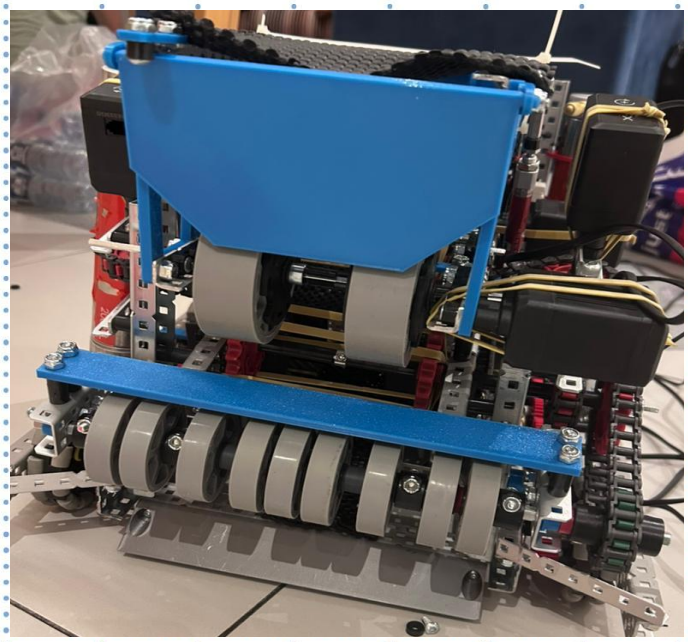
The first layer has two conveyor belts on both the left and right side. After the speed the block gains from the intake it is directed two gears connected by a shaft that has rubber bands placed on the gears teeth (gear system) that pushes the blocks to the conveyor belts which then direct them to another gear system which pushes the blocks upward onto the second layer.



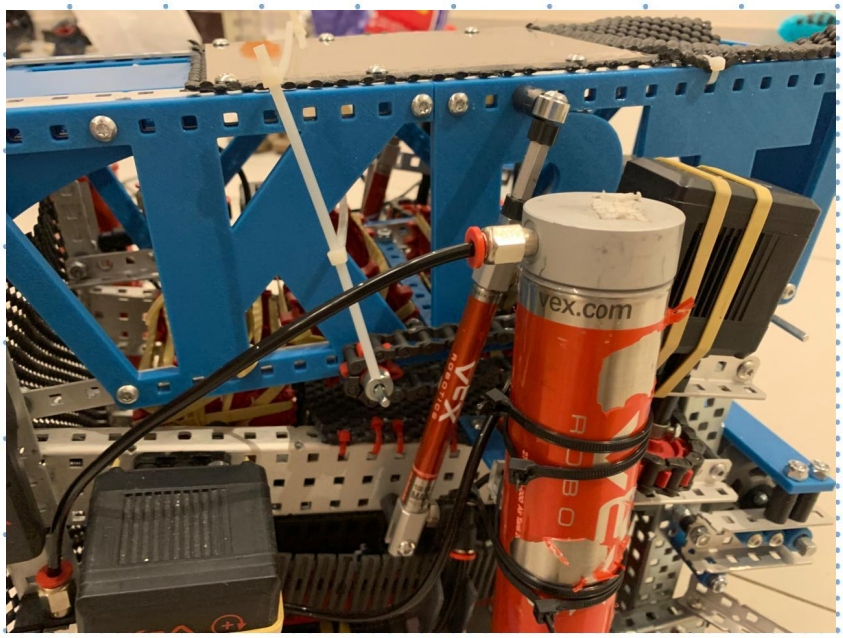
Picture of conveyor belt and two gear systems on the bottom



Sleekh Design



Flex wheels and scoring gate



Second layer showing gear systems and 3D printed casing

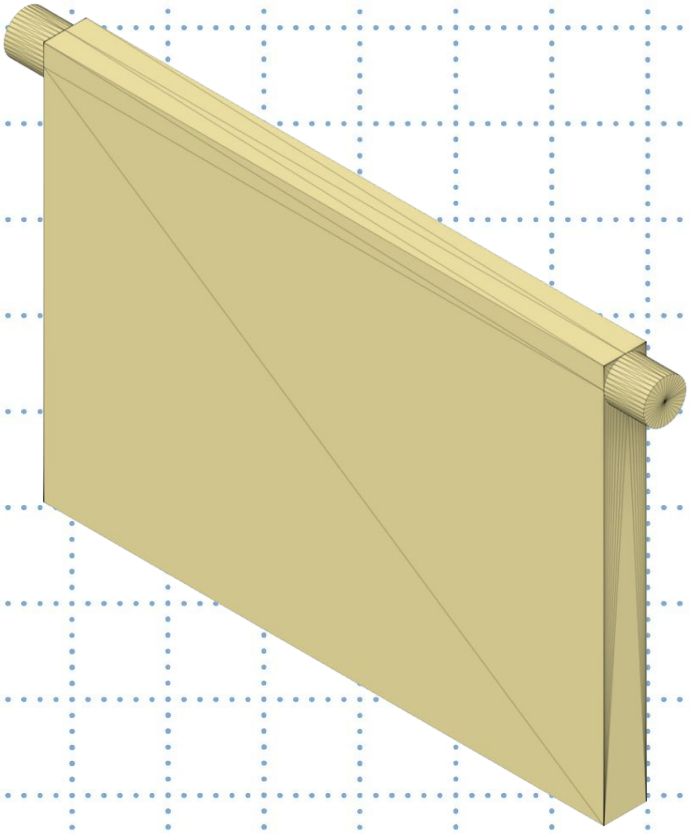
The back of the robot consists of a mesh rubber net which guides the blocks from the first layer to the second layer with a 3D printed ‘elbow’ that is a curved piece the directs the blocks smoothly towards the gear systems.

The second layer of the robot consists of an acrylic roof and two gear systems as defined before and two large flex wheels at the end of the chamber to guide the blocks when scoring.

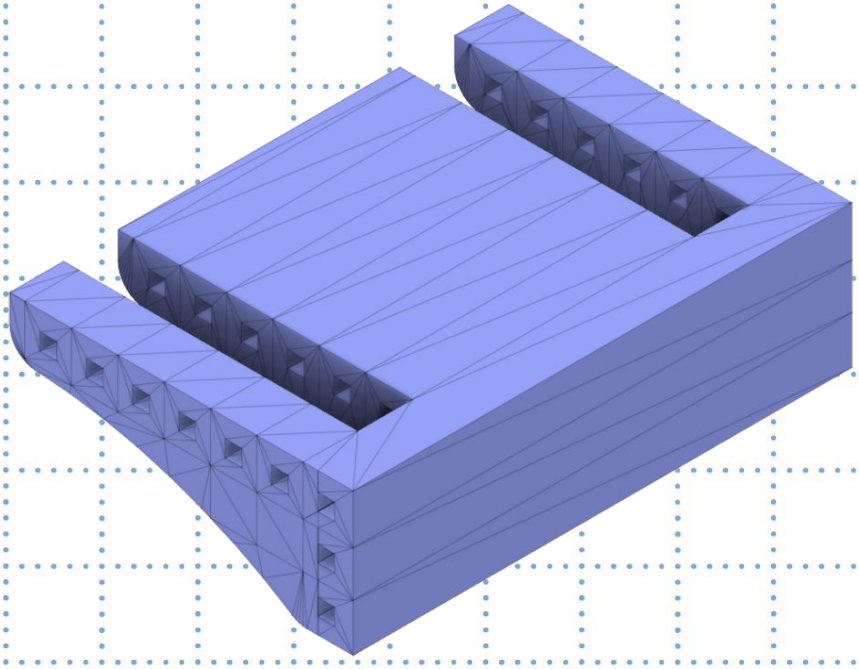
The housing of the chamber is made out of 3D printed modeled pieces that are supported by aluminum vex c-channels that give extra support.

The end of the chamber is closed off with a 3D printed gate that allows the robot’s driver to control when blocks are released. This was designed to make sure no blocks are released with direct action and this reduced the chaos that can occur during games.

Sleekh Design



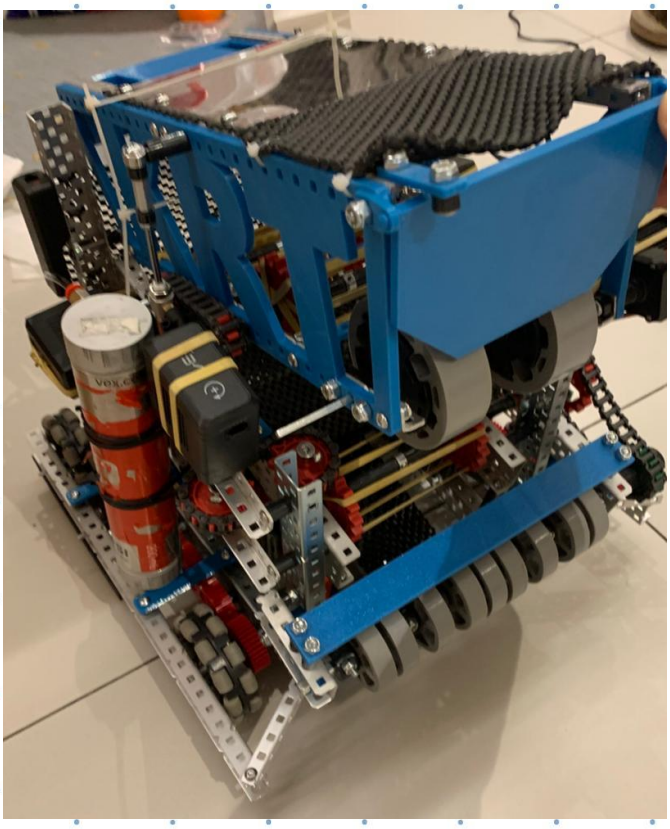
3D printed scoring gate

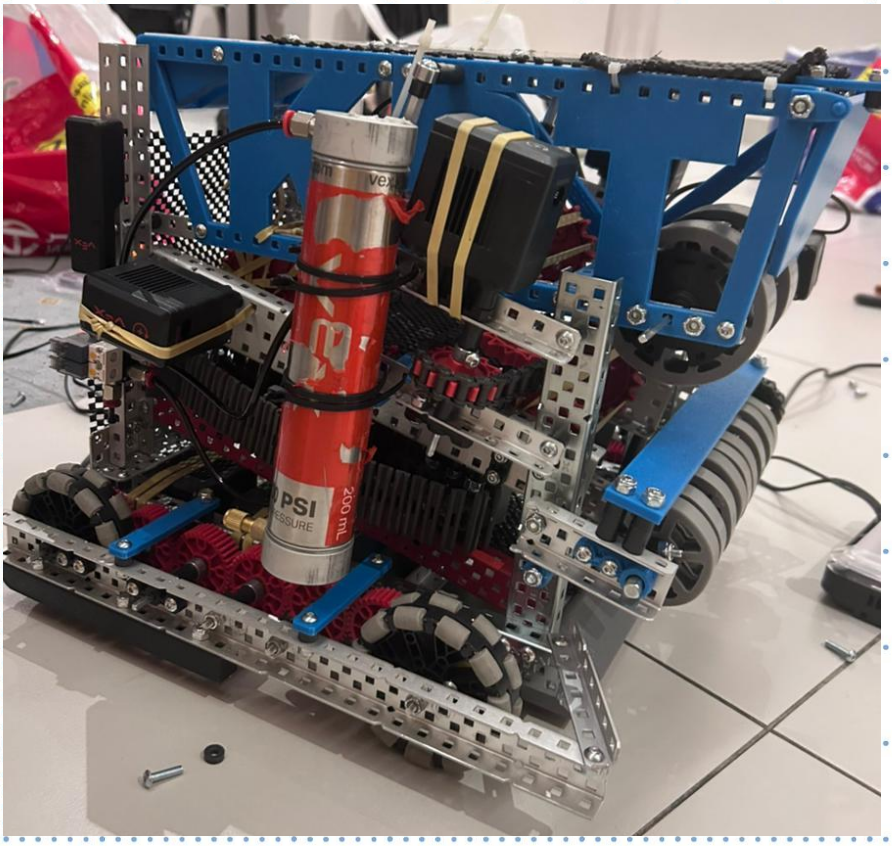


3D printed curved ‘elbow’

Sleekh Design

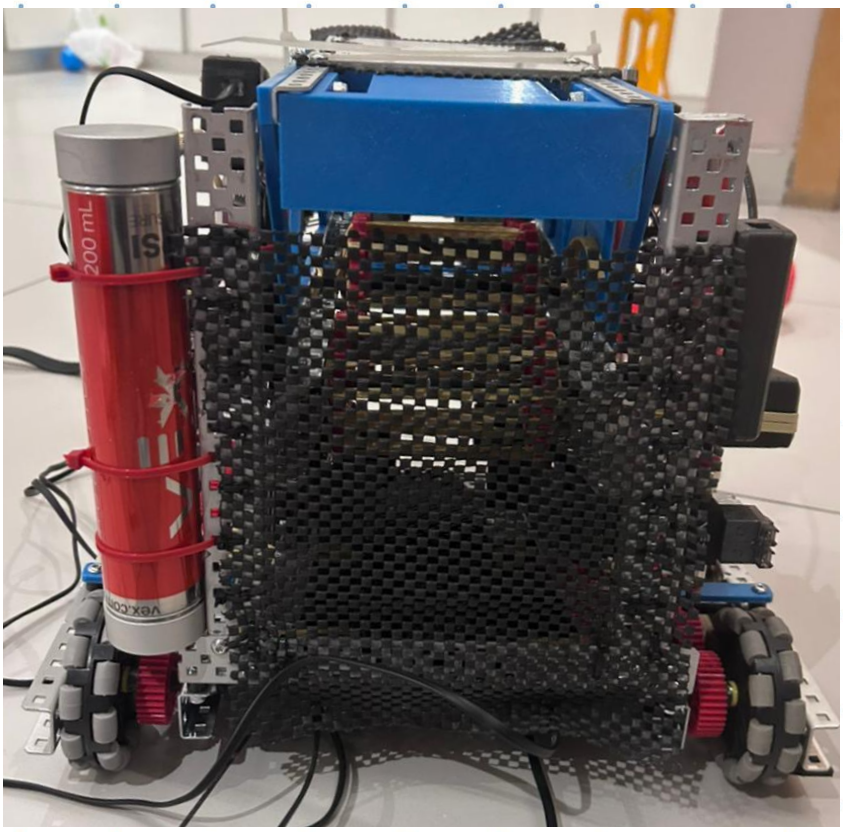
Side view of Sleekh with extended pneumatic system





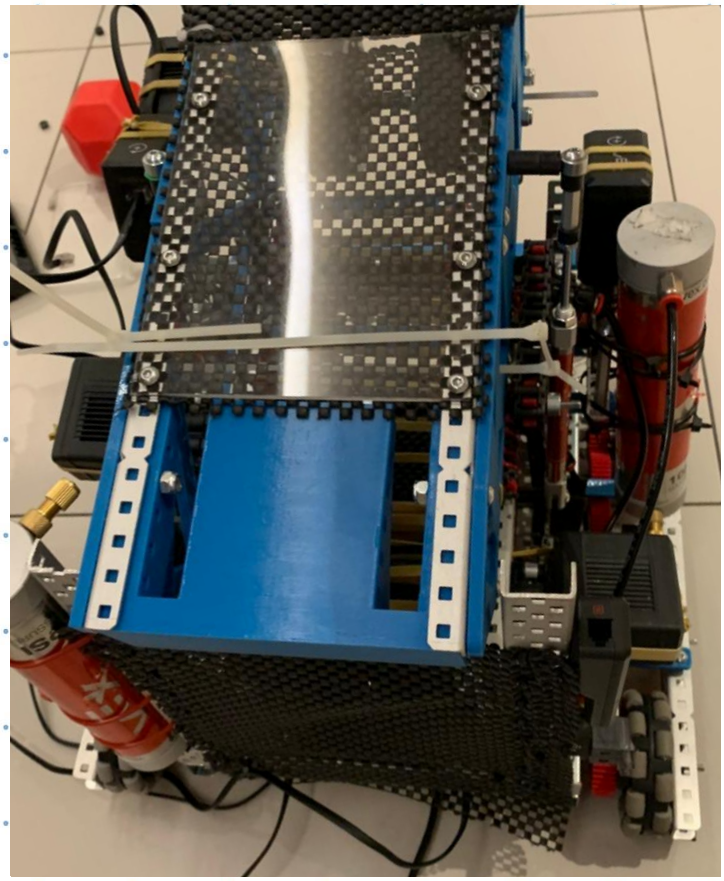
Side view of Sleekh at relaxed position

Sleekh Design

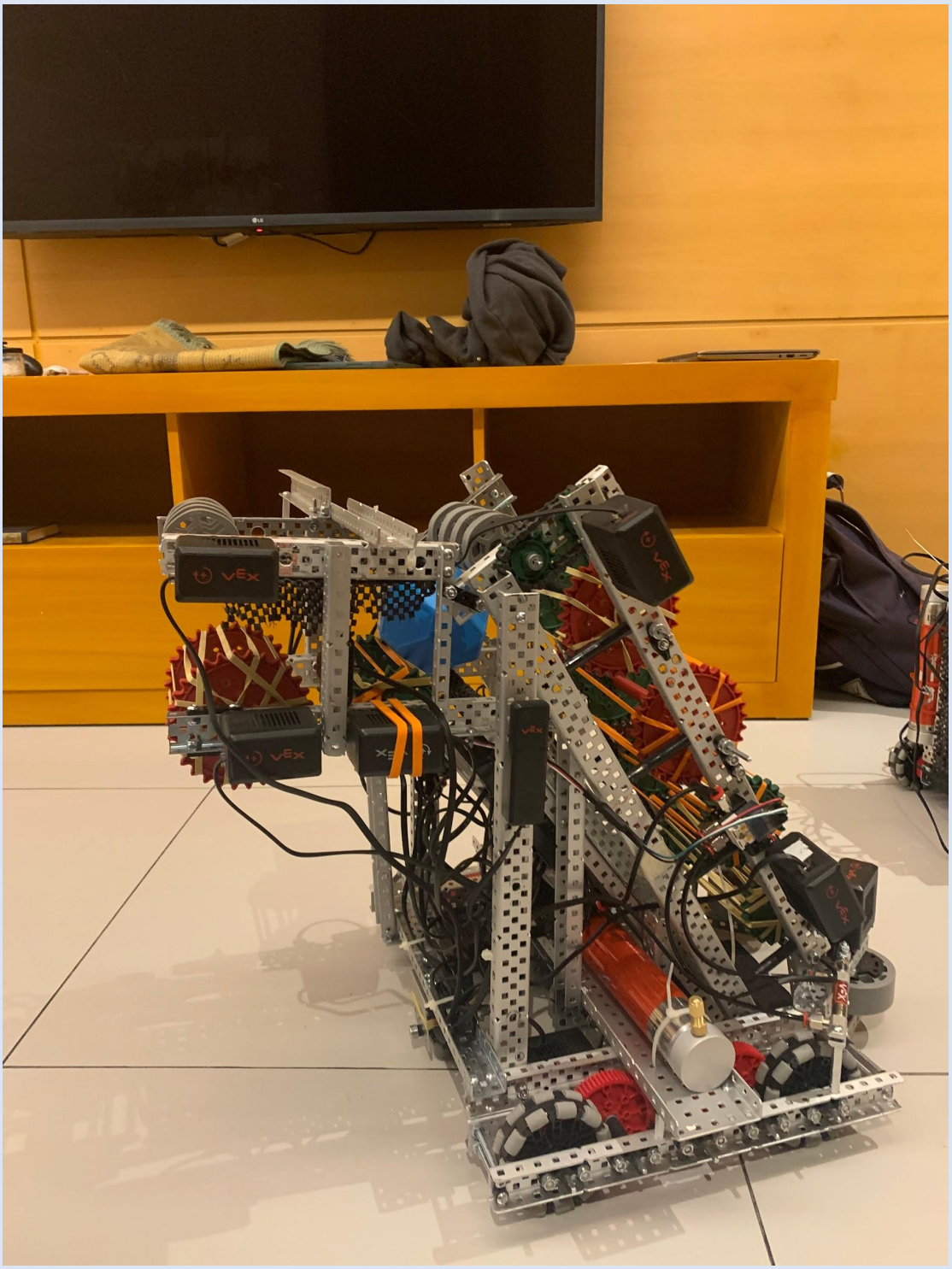


Back view of Sleekh

Top view of Sleekh



Theeb



Theeb V1: Initial Design

Our robot **Theeb,** has been initially designed with the following features:

- **Base:** X-drive
  - *Reasoning*: Omnidirectional movement allows for easier control and, therefore, higher precision when intaking and scoring points compared ot the H-drive base.
- **Intake:** Two extended metal arms with four spinning flex wheels and two pneumatic pistons that (de)compress.
  - *Reasoning*: The design allowed for the fluid intake of balls from the loaders when the arms were closed AND fluid intake of balls from the ground when the arms were open.
- **Scoring Mechanism:** S-Shape
  - *Reasoning:* Allowed for high storage of balls; allowed for control of scoring in the middle or long goal through a pneumatic piston.

(Further reasoning is explained in the conception)

Theeb V1: Conception

During the second meeting held on september 20th the team discussed what will theeb's new design will be taking into consideration the new regulation and theme.

The shape of the base and goal scoring mechanism was the major taking point of the meeting.

We came to the conclusion that due to the push back theme favoring offense and fast movement that a **X-drive base** would be optimal to meet our goals.

With the X-drive's 45 degree motors it would allow seamless movement and turning.

This ,however, would be a difficult endeavour as experienced members did not experiment with this design before. This would mean that experienced members and new members would have to learn together to craft the base.

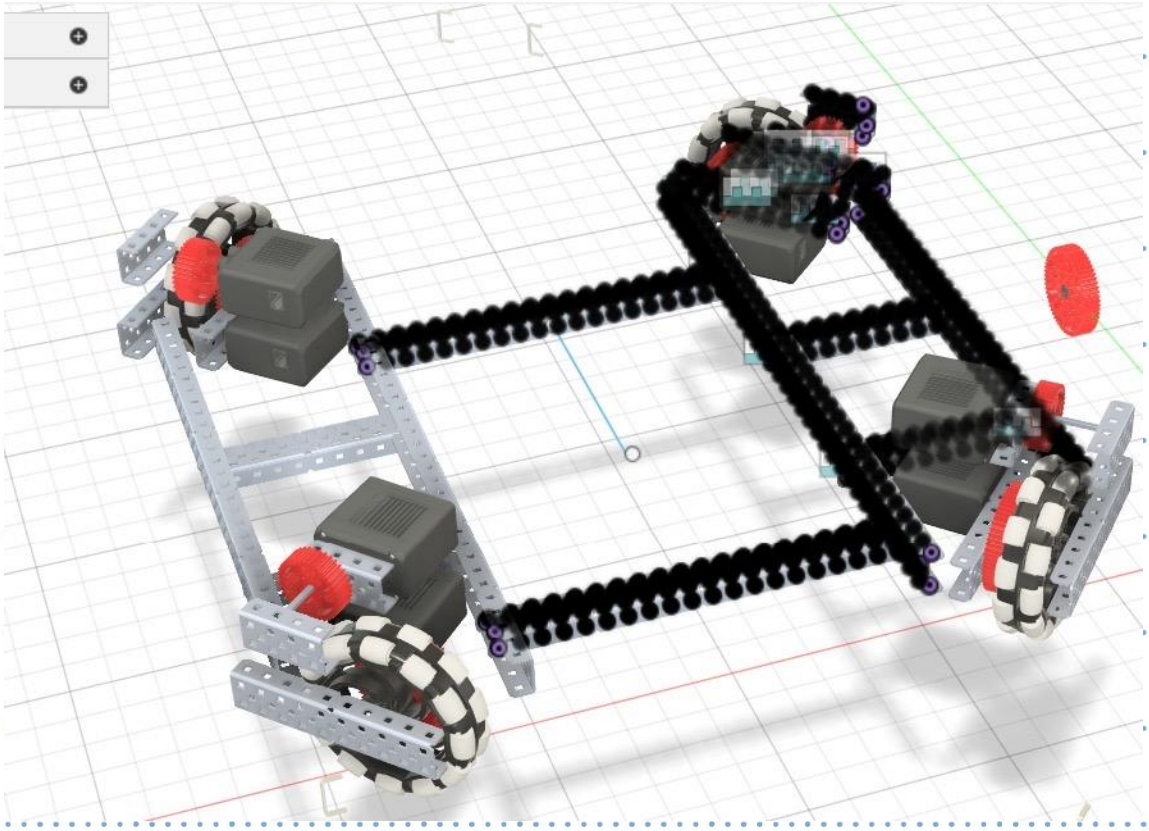
Lastly, the team used an H-drive the prior season and had difficulties with programming the auto for both Theeb and Sleekh especially with how to deal with turning and going to a specific region of the field. An X-drive design would allow for smooth and precise turning as a consequence of the 45 degree motors for the driver of the Theeb and Sleekh and for the auto programming.

Theeb V1: Design Inspired By

Vex V5 Design Tutorial by Creator Academy Australia
https://www.youtube.com/watch?v=NbYCmbmTWXc

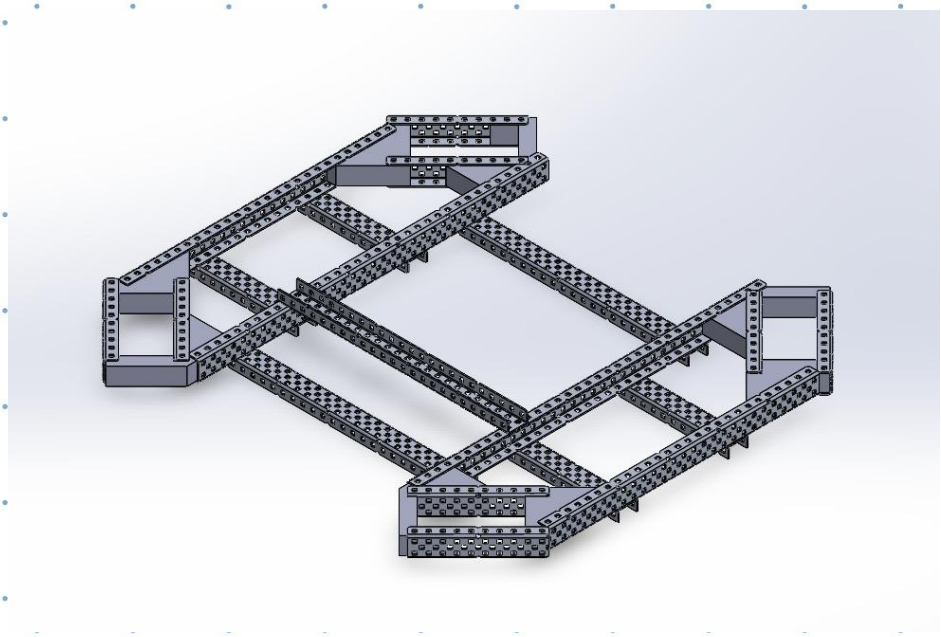
Theeb V1: Conception

The S-Shape was picked for the scoring mechanism because it balances between a high ball capacity and simplicity to build.

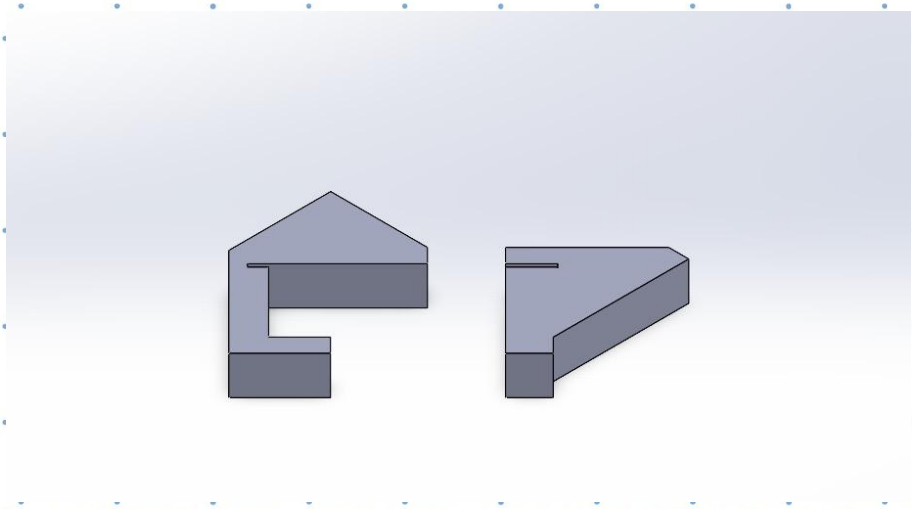


Theeb's base Initial CAD design

Theeb V1: Conception



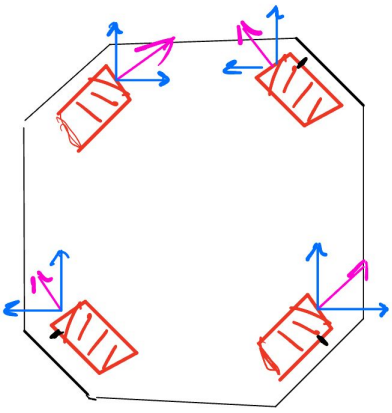
Theeb's Final CAD Design



Theeb's X-Base Corners CAD Design

Theeb V1: Vector Analysis

Moving Forward

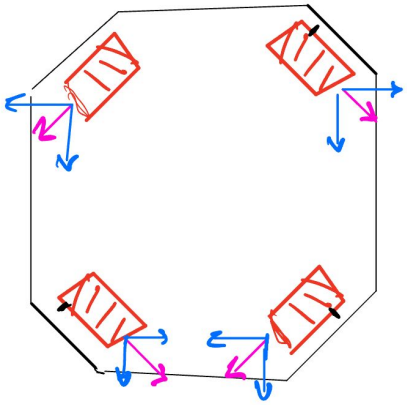


Motor

Velocity vector

velocity vector component

Moving Backwards

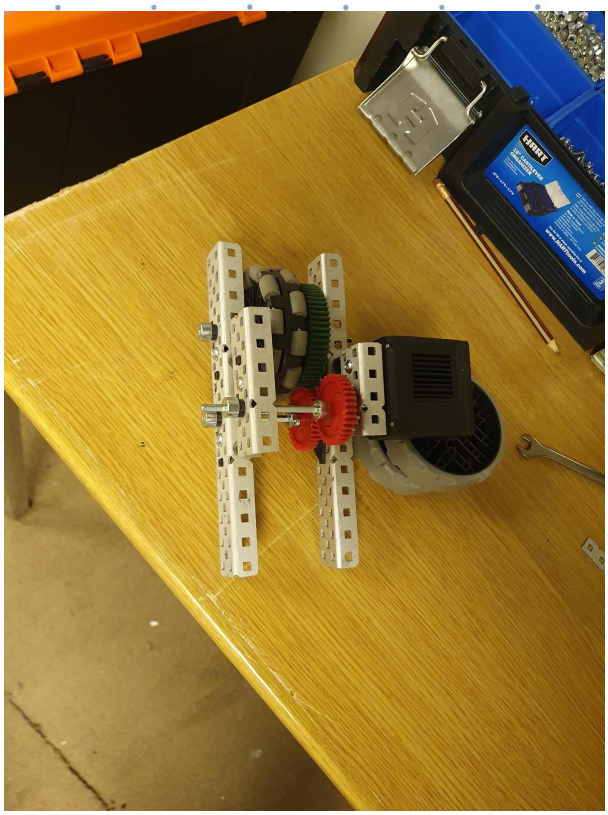


Motor

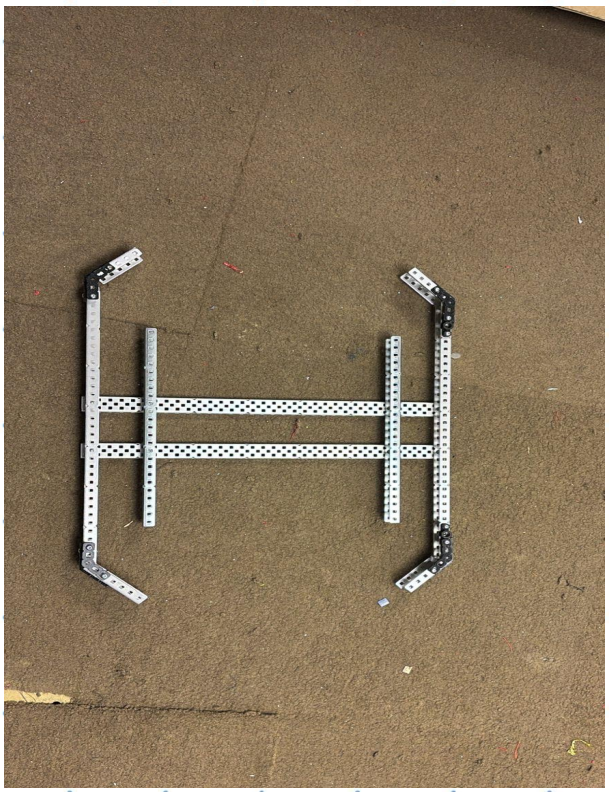
Velocity vector

velocity vector component

Theeb V1: Progress

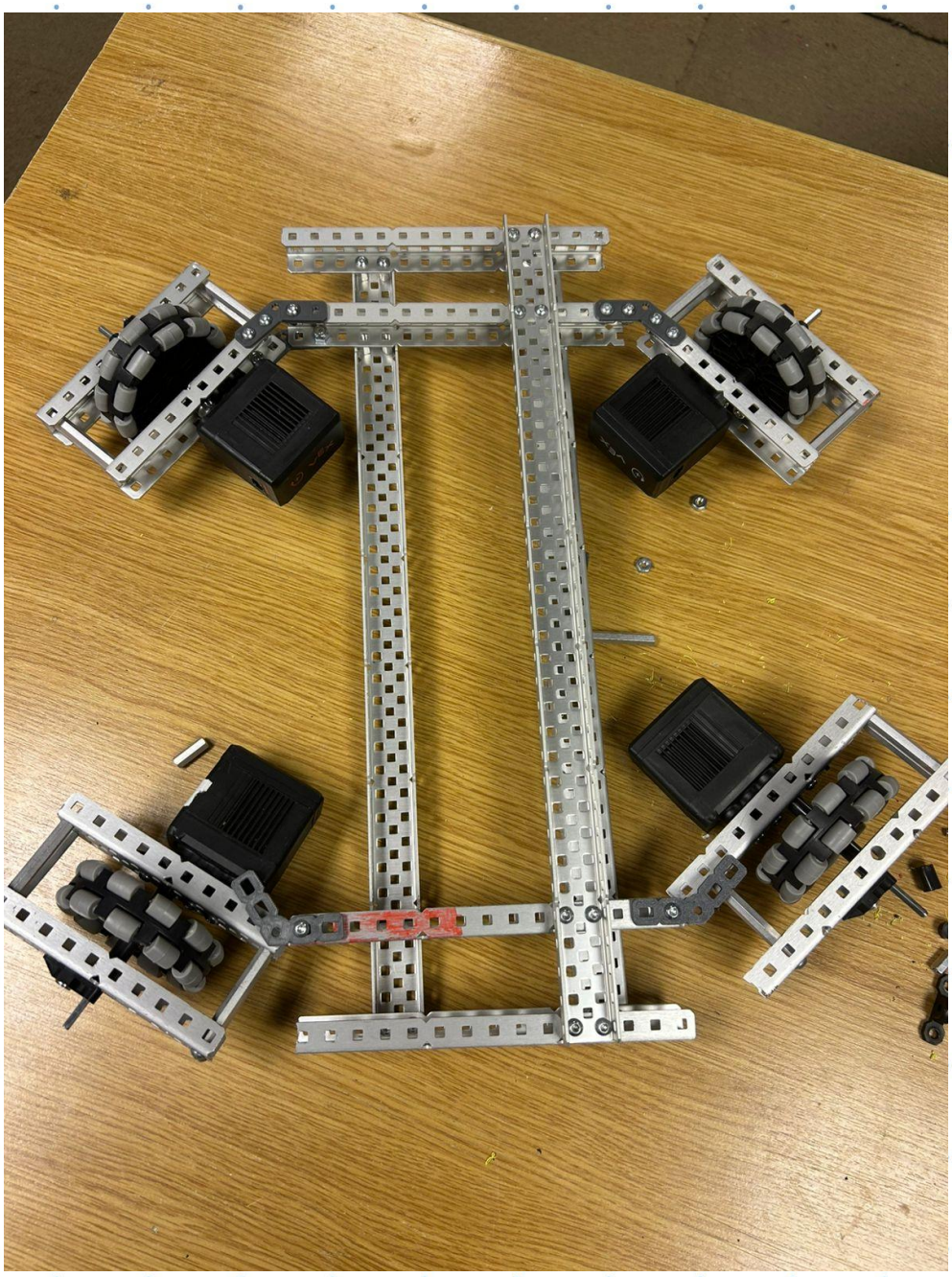


First motor Built

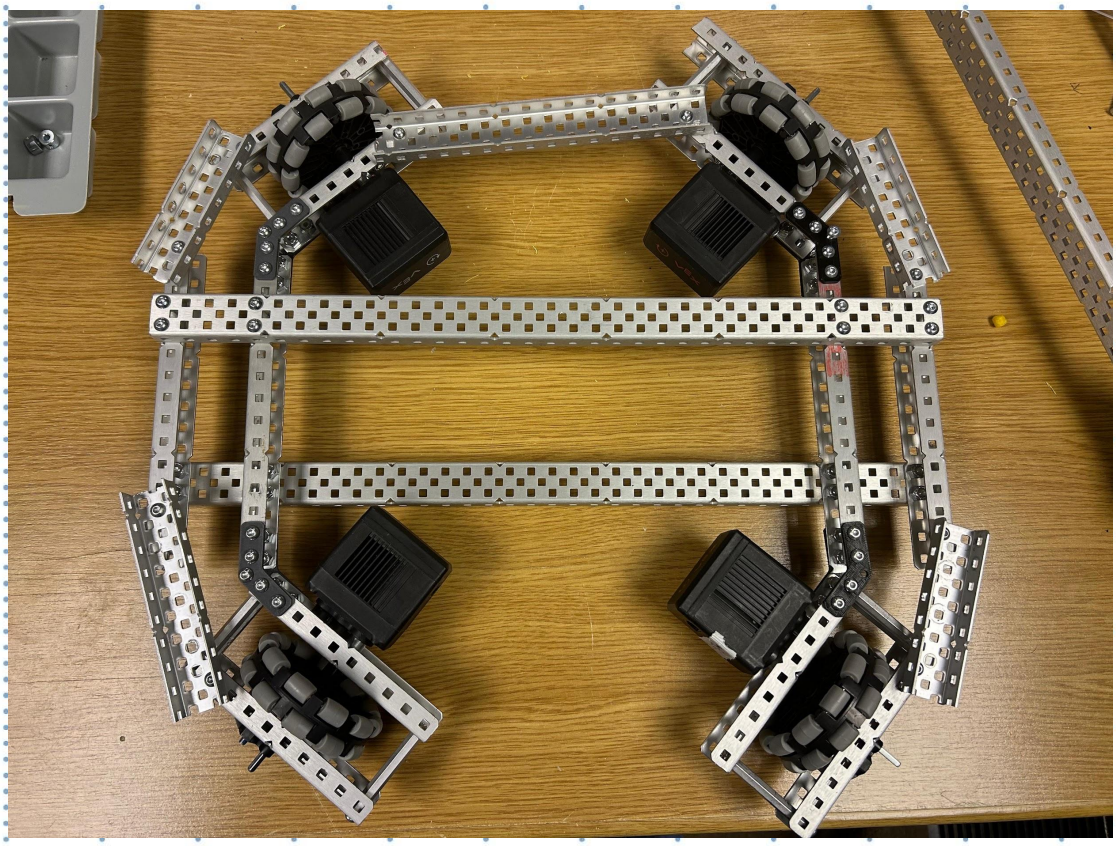


Frame of base

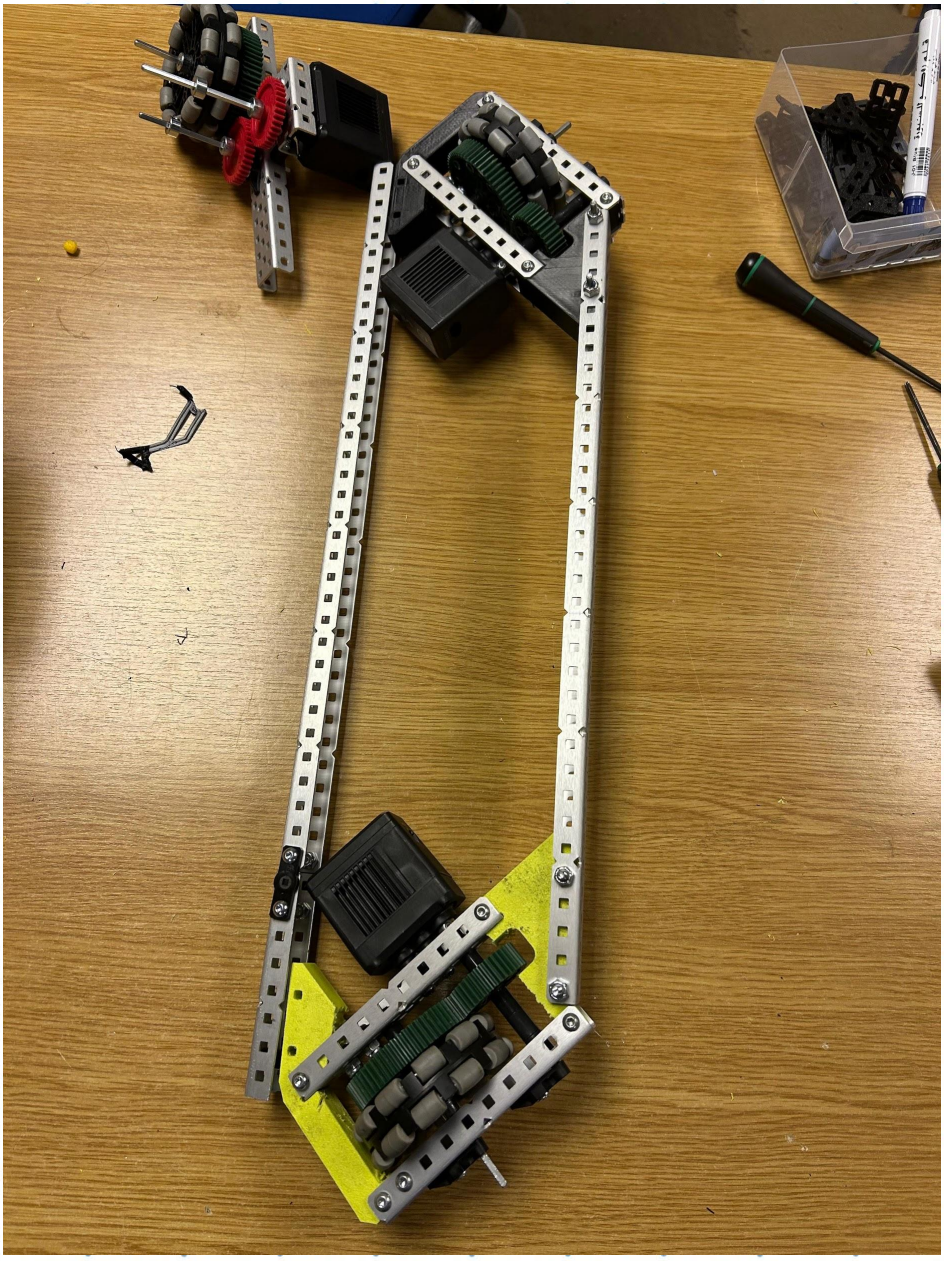
Theeb V1: Progress



Theeb V1: Progress

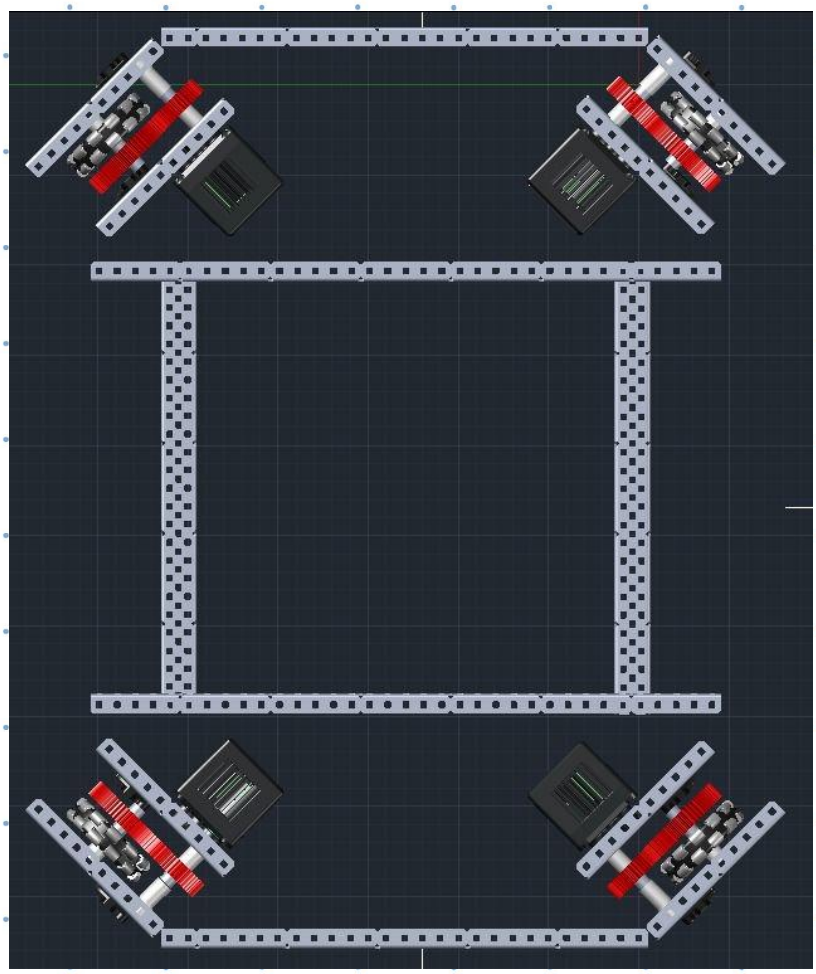


Theeb V1: Progress



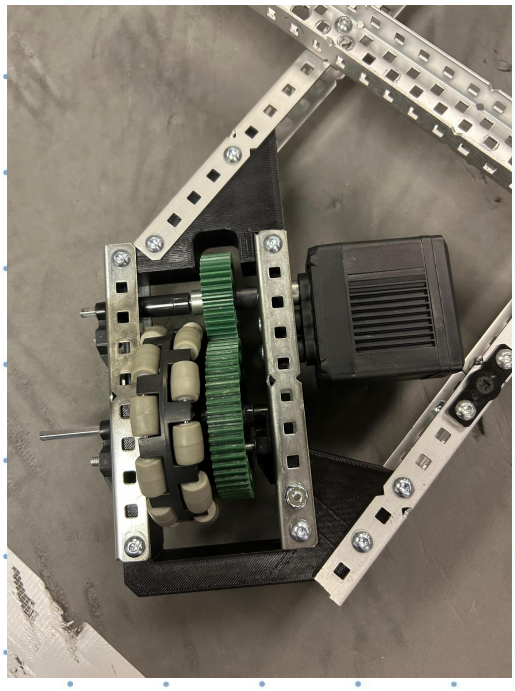
Breakthrough on how to connect the motors using the 3D printed corners

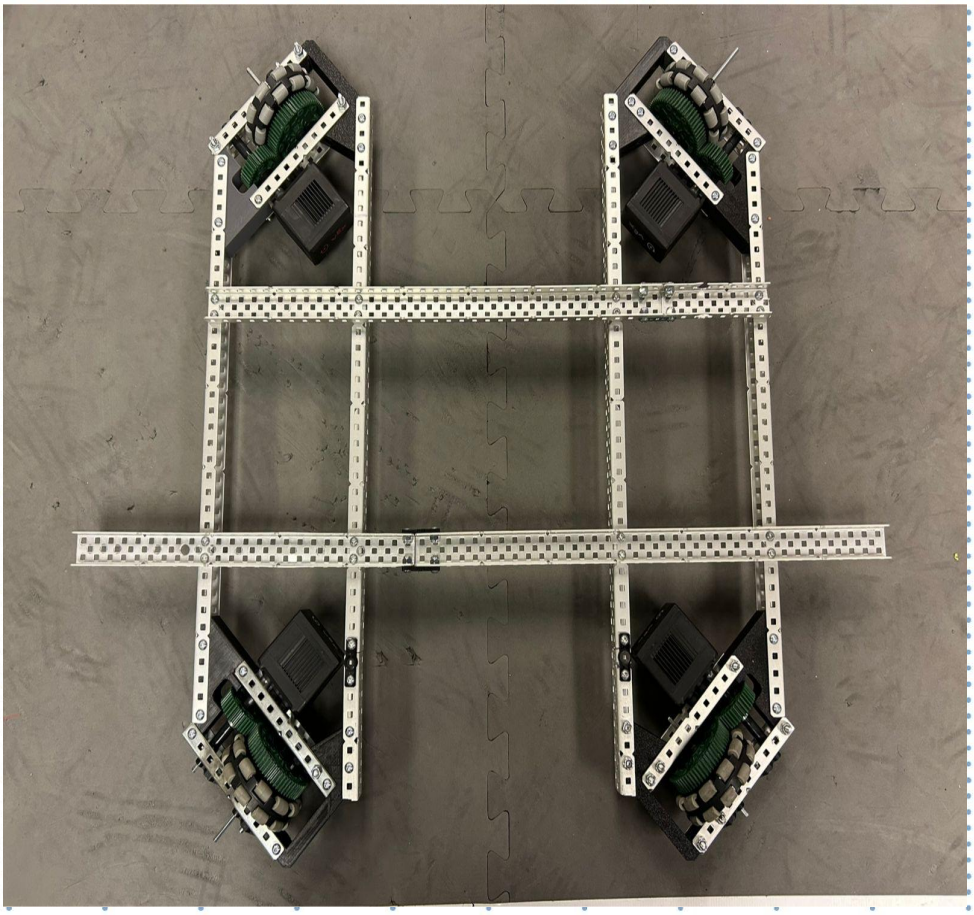
Theeb V1: Progress



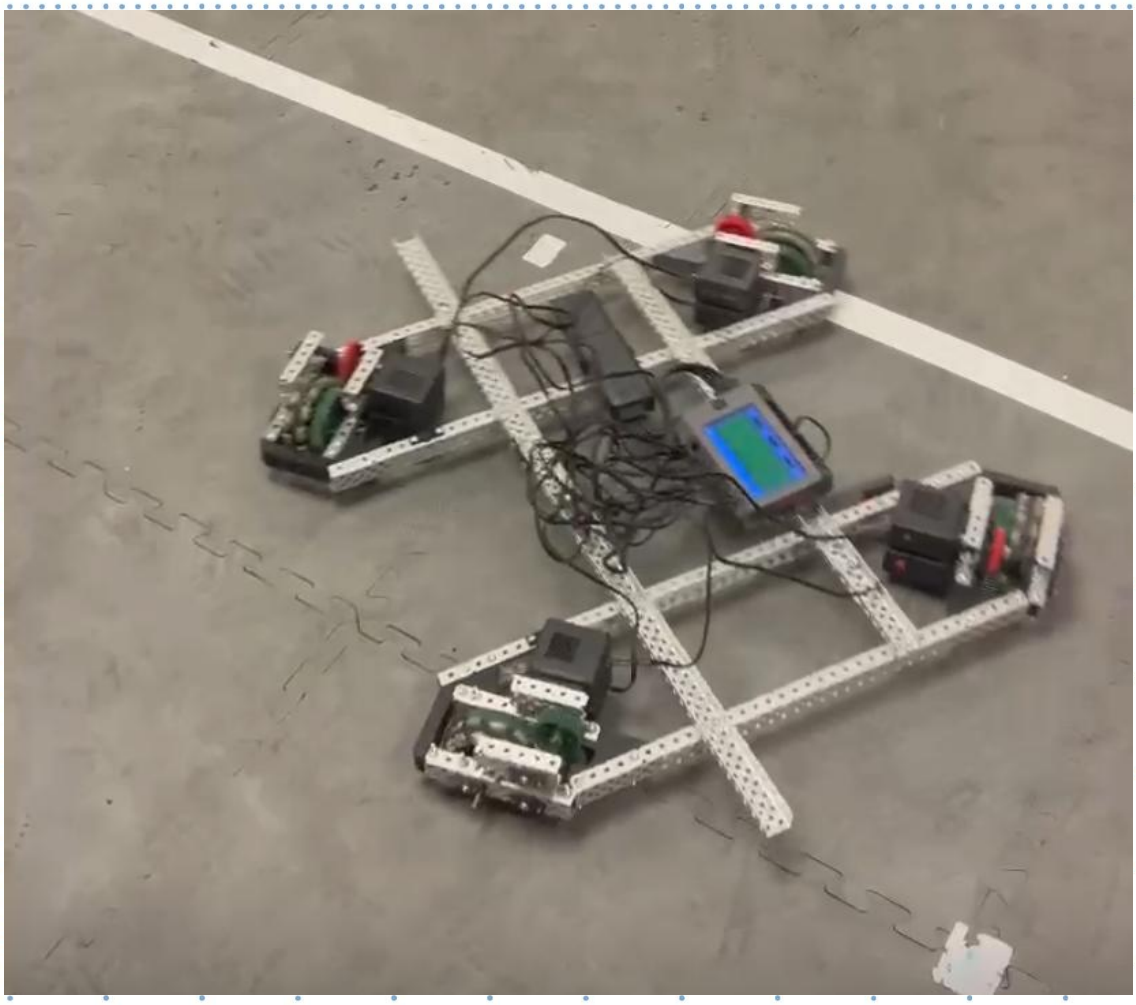
CAD design after discovering the breakthrough

Theeb V1: Progress





Theeb V1: Progress

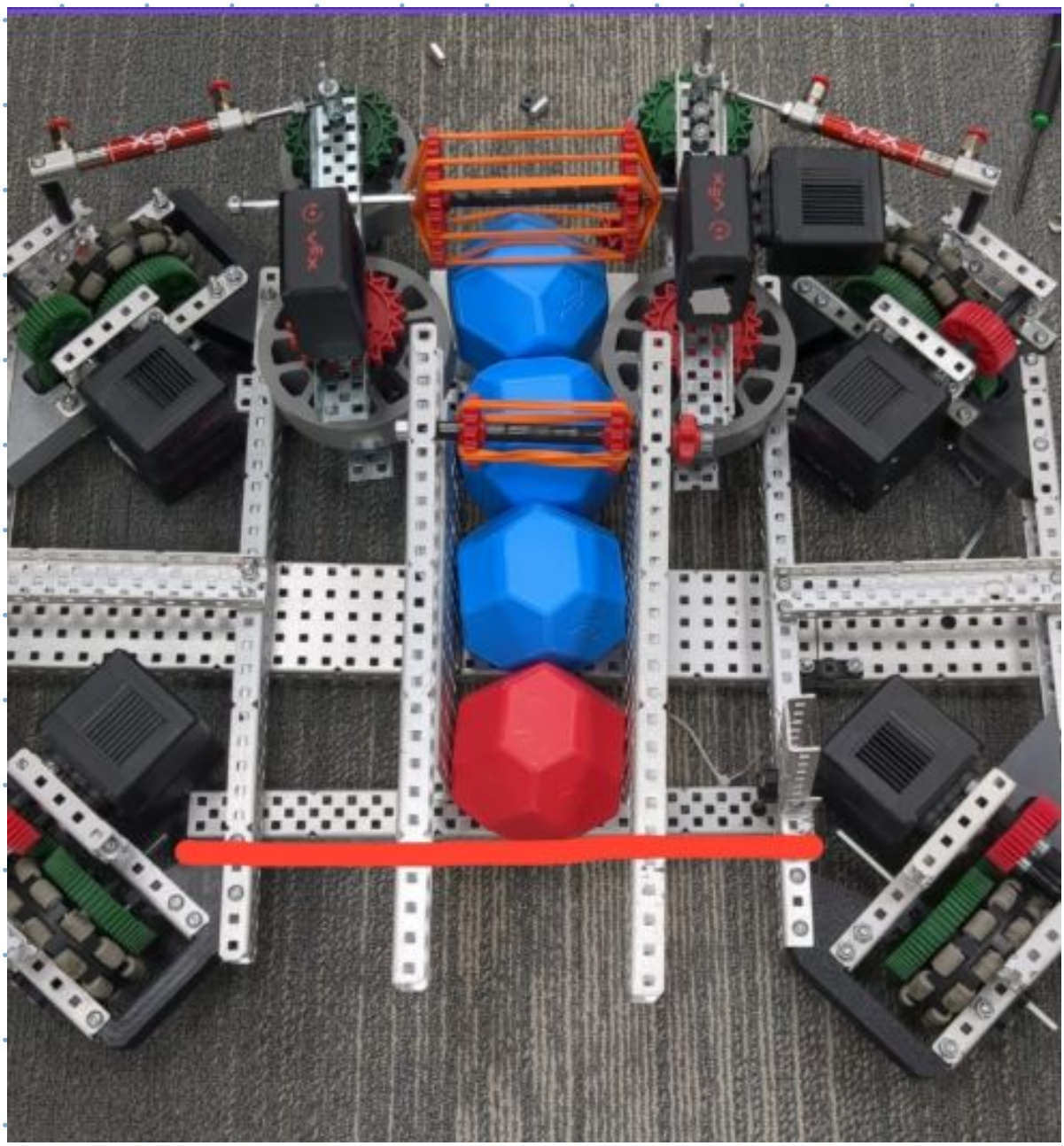


X-drive prototype

Theeb V1: Progress

Intake

Theeb V1: Progress



X-Bridge Drive Train

Theeb V1: Progress

Theeb V1 (In progress)

Theeb V1: Problems Faced

- The X-Drive was a challenge to design, build and test
- The X-Drive took a long time to fully finish and required 3D parts to build
- The robot was heavy and slow.
- The base large it was 19in x 16in and prevented us from double parking
- Slow intake and scoring mechanism
- Ball capacity was not optimized as much as possible, which is why it was lower than expected.
- Color sorting mechanism was not implemented.
- Theeb did not hold up during practice matches

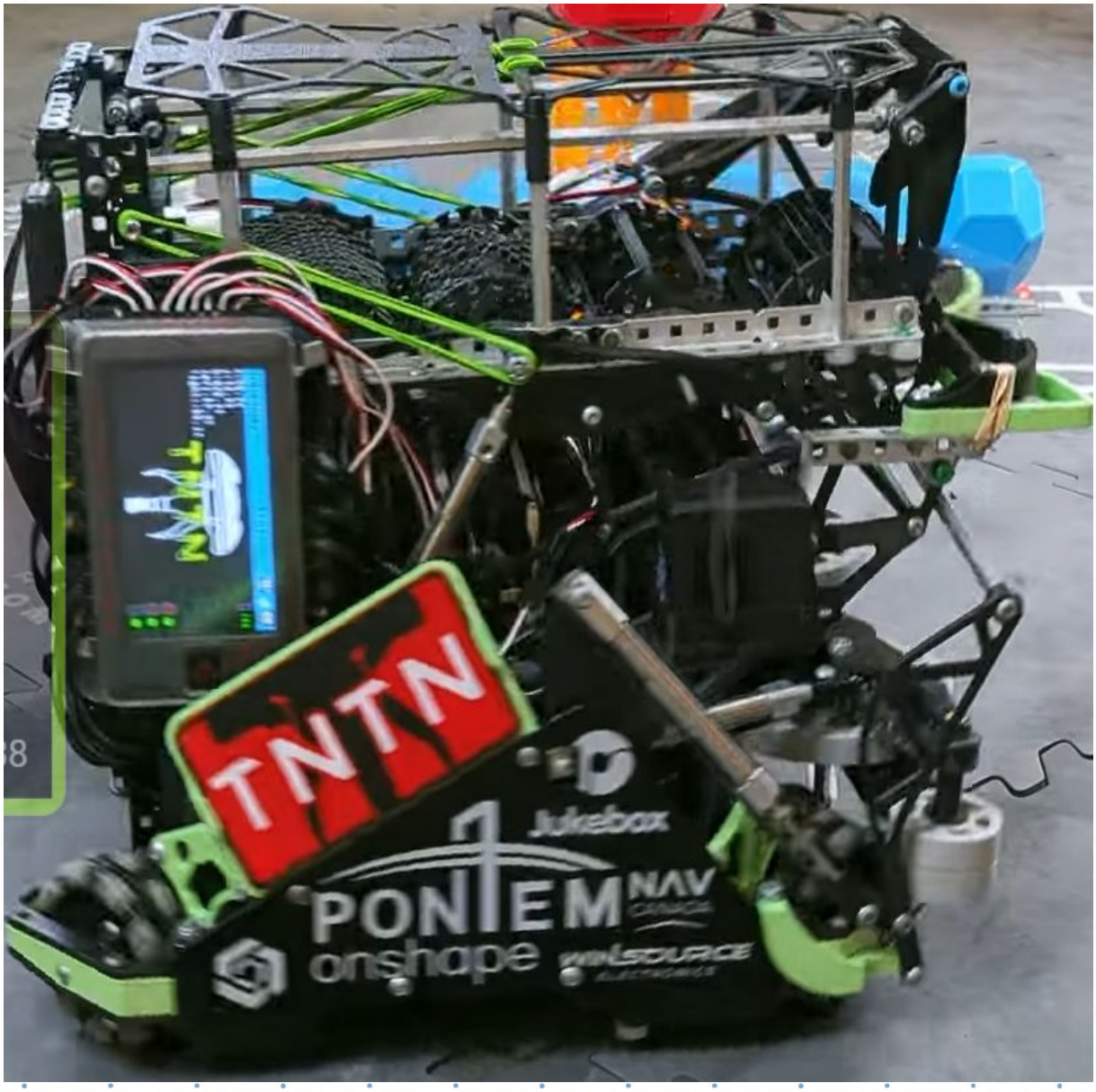
Theeb V1: Problems Faced (*with testing*)

Feature	Spec (Slow Pushback Build)	Notes
Ball capacity	6 balls	Small capacity
Intake speed (slow)	~60–100 RPM	Slow intake
Weight (heavy)	8 kg	Heavy + low center of gravity
Speed of moving	Slow: ~0.8–1.2 m/s	Low speed
Speed of rotating	Slow: ~120–180° /sec	Slow turning
Outtake power	Medium–High: 70–100% motor power	Not strong enough → Needed more friction and narrower outtake.

Theeb V1: Solutions Brainstormed

- Our team decided to switch to the H-drive, due to its faster speed, simplicity, and smaller size.
- The base dimensions should be 14 inches maximum for both length and width.
- To construct a smaller size, our team decided to build a C-pathway for the scoring mechanism to save space.
- Add a color sensor.

Theeb V1: Solutions Inspired By



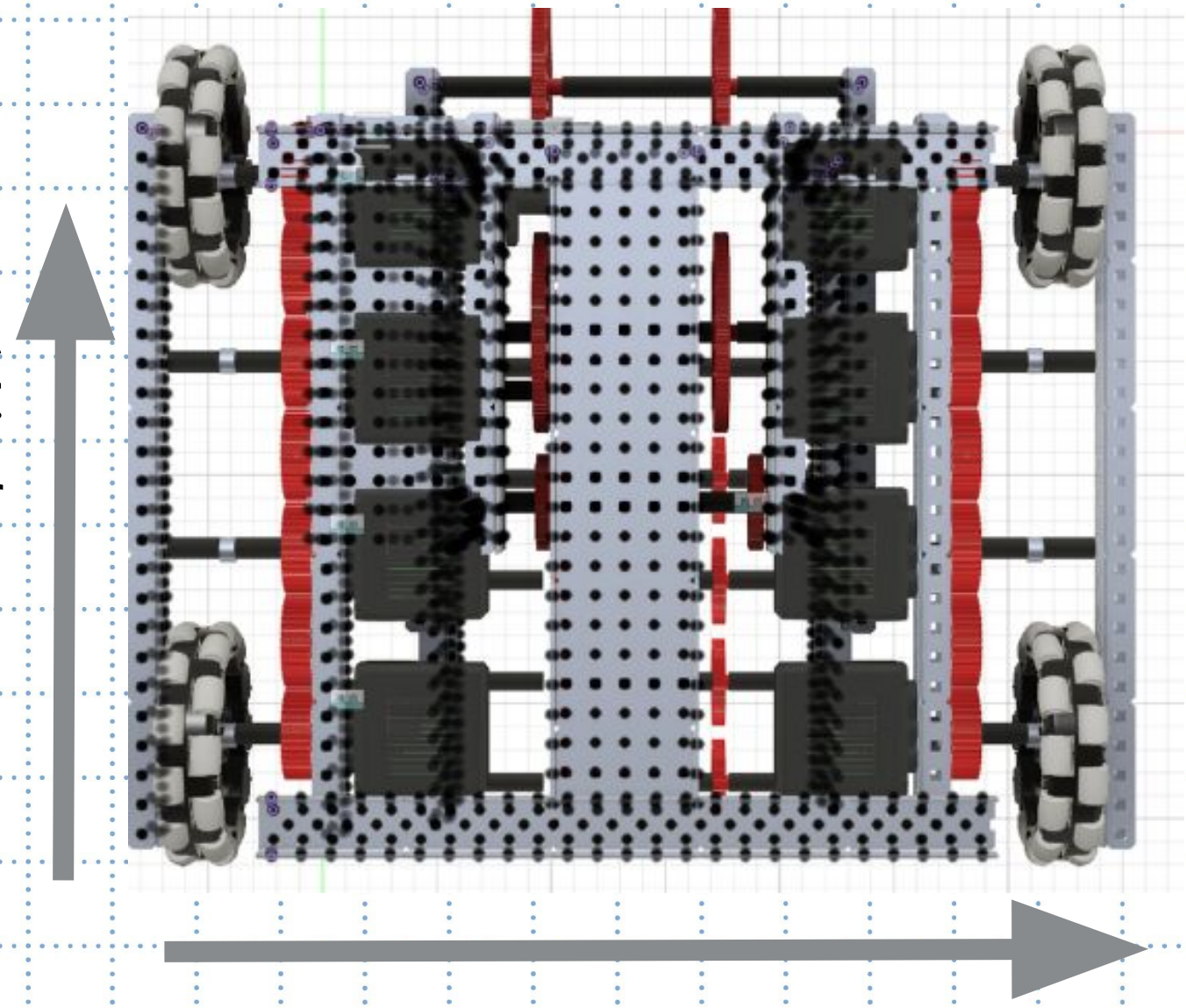
TNTN C-pathway robot:
https://youtu.be/-ed_7oiixMo?si=r59BNnAjQTddtvTz

Theeb V2: Initial Design

The second version of **Theeb** was made to tackle issues encountered after playing against **Umm Al-Qura University**

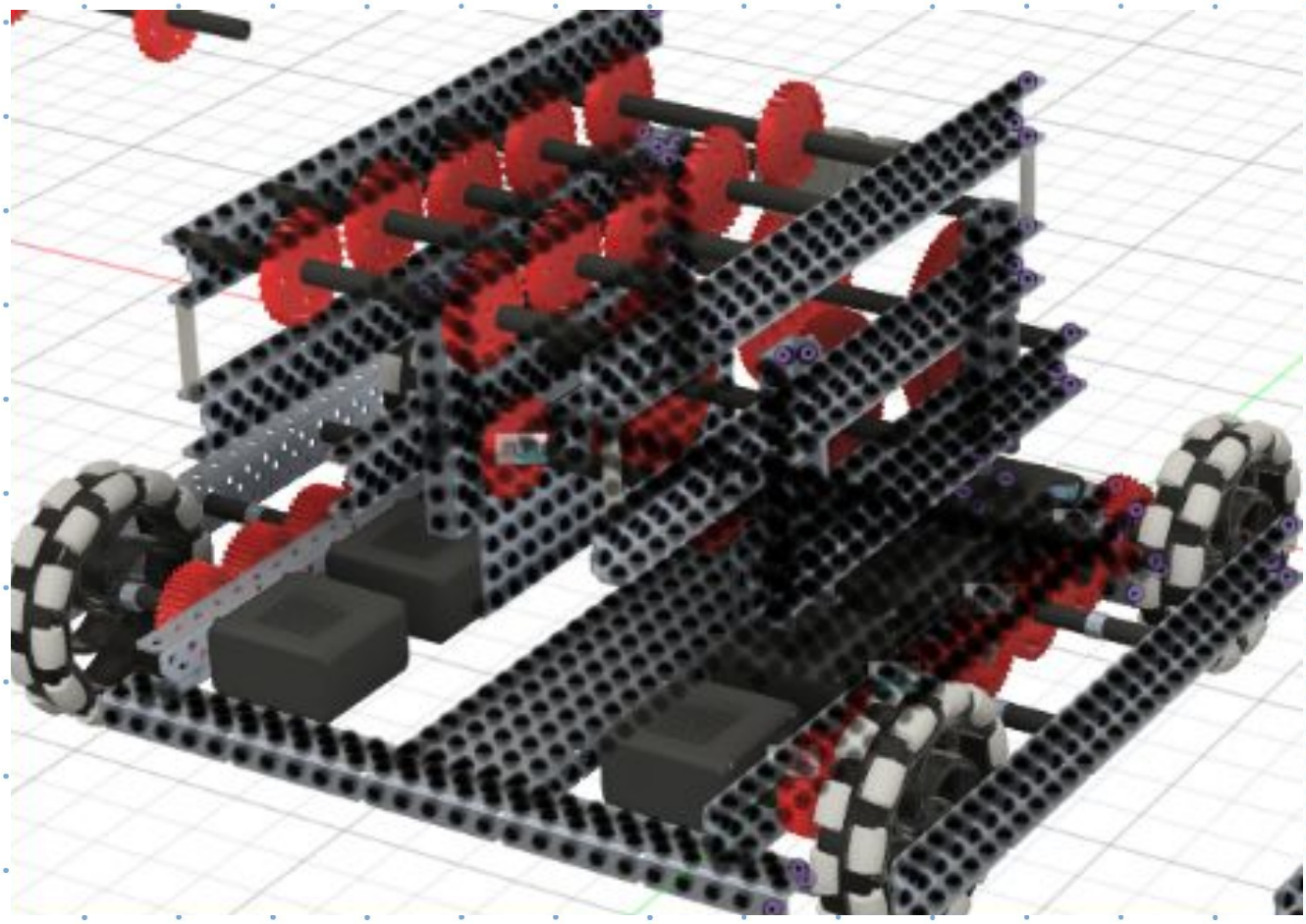
- **Base:** H-drive with 4 wheels and 6 motors.
  a. *Reasoning***:** the X-drive base required a large frame to house the wheels at a 45° angle which prohibited us from double parking. Furthermore, the base felt unresponsive and slow due to its heaviness when playing
- **Intake:** Match loader and a shaft equipped with flex wheels inside **Theeb**
  - *Reasoning*: Allowed us to intake 4 balls simultaneously and the match loader could be used to push balls from the parking zone to allow for parking
- **Scoring Mechanism:** C-Shape
  - *Reasoning:* While playing we discovered that many of the balls would go below the long goal, so the C-Channel design would allow us to make **Theeb** short enough to go under the long goal.

Theeb V2: Progress



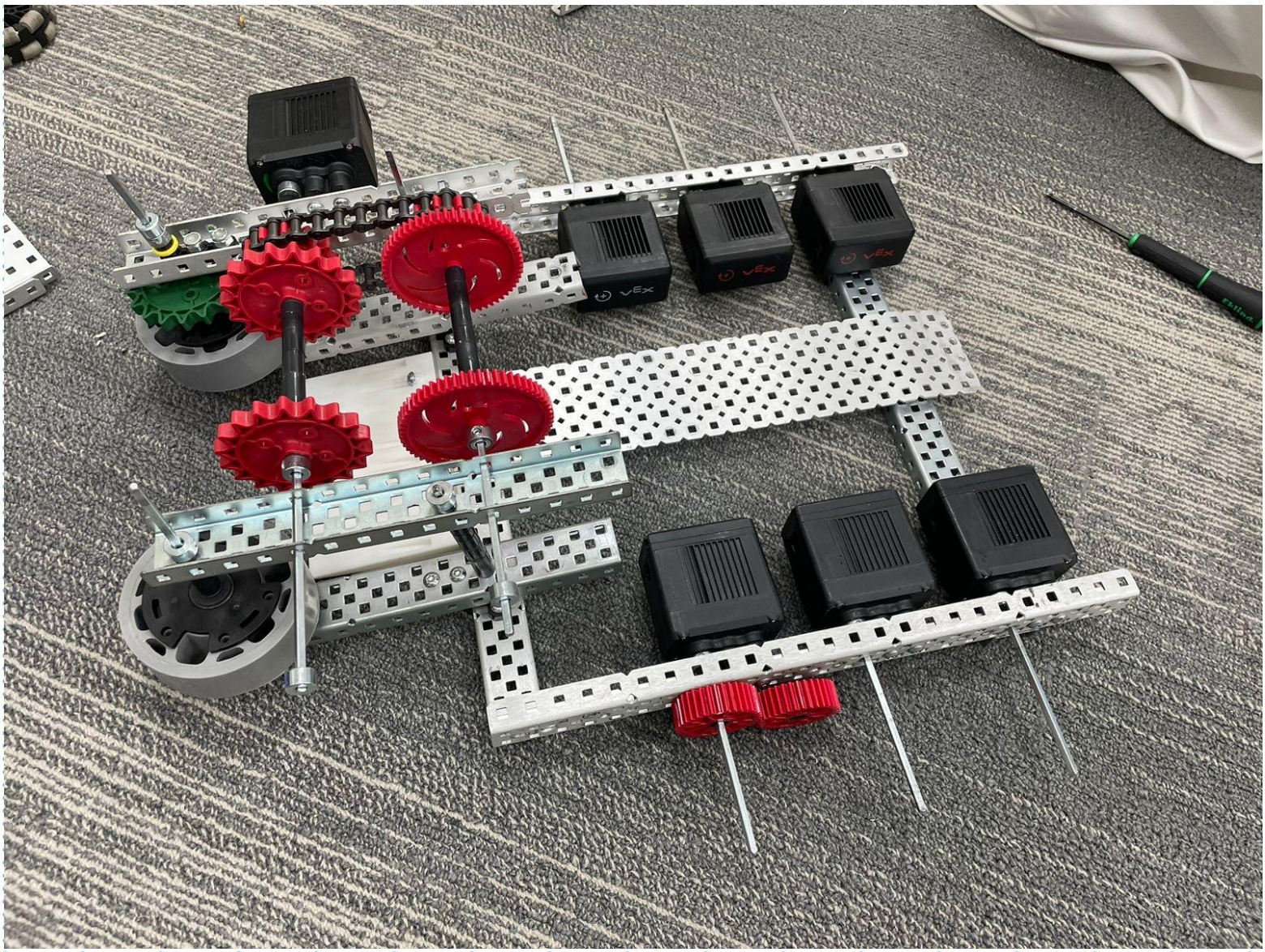
Theeb V2 (CAD top view)

Theeb V2: Progress



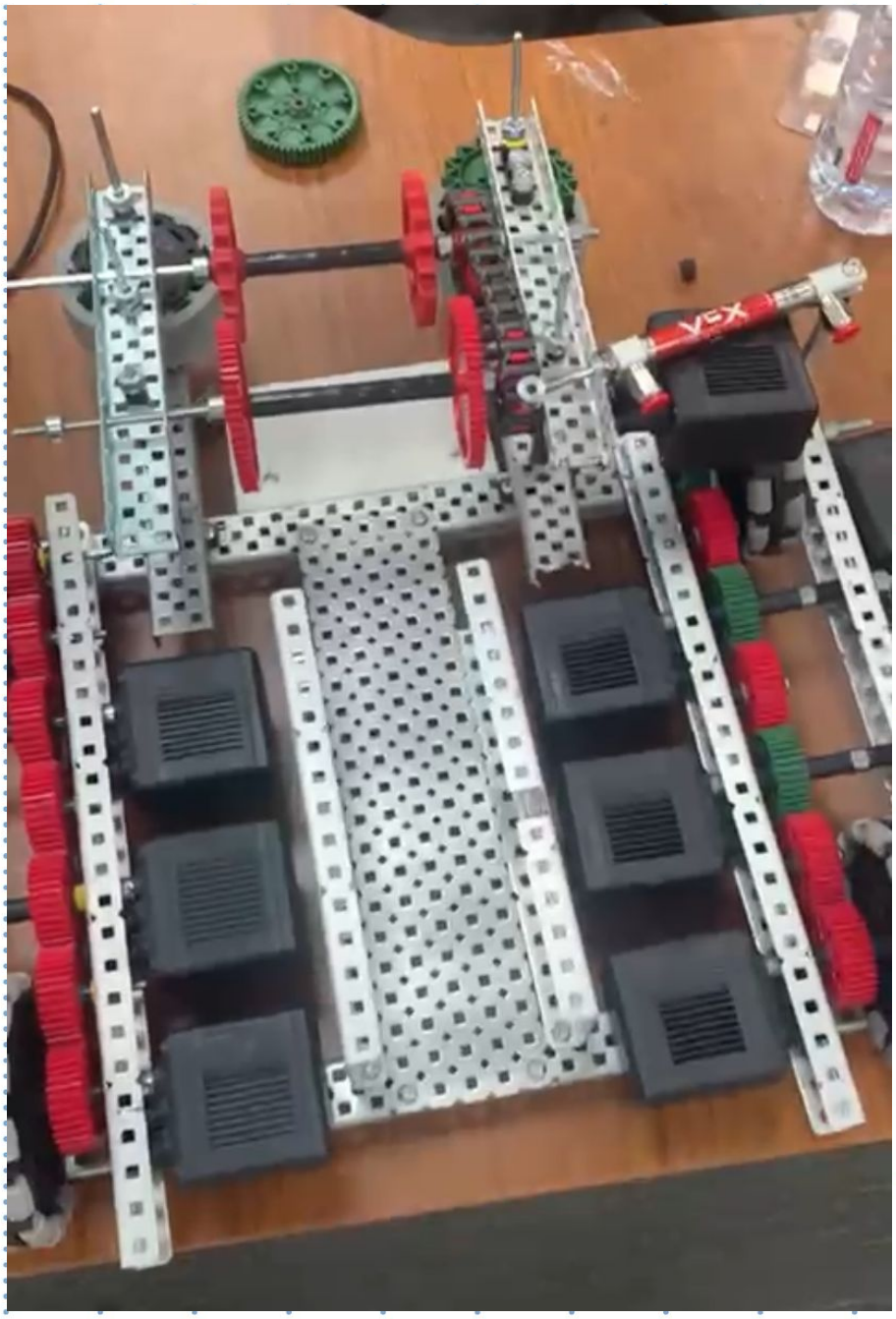
Theeb V2 (Full CAD model)

Theeb V2: Progress

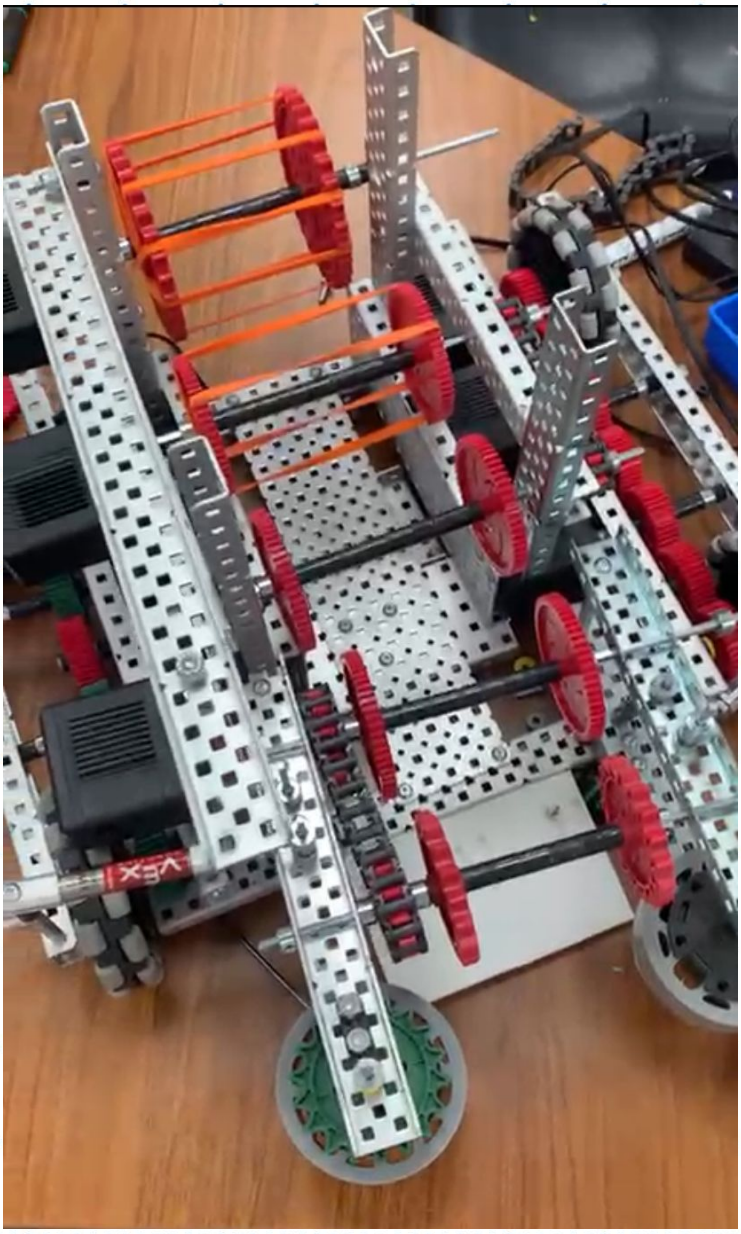


Theeb V2 (In Progress)

Theeb V2: Progress

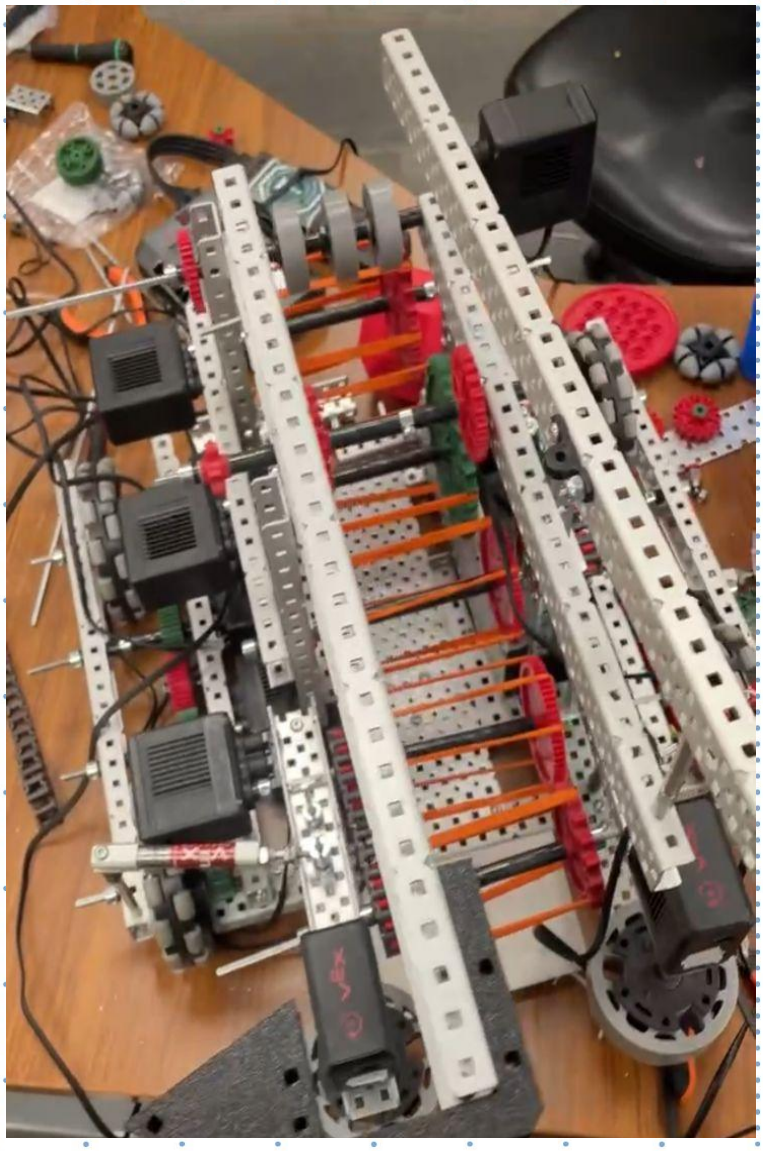


Theeb V2: Progress



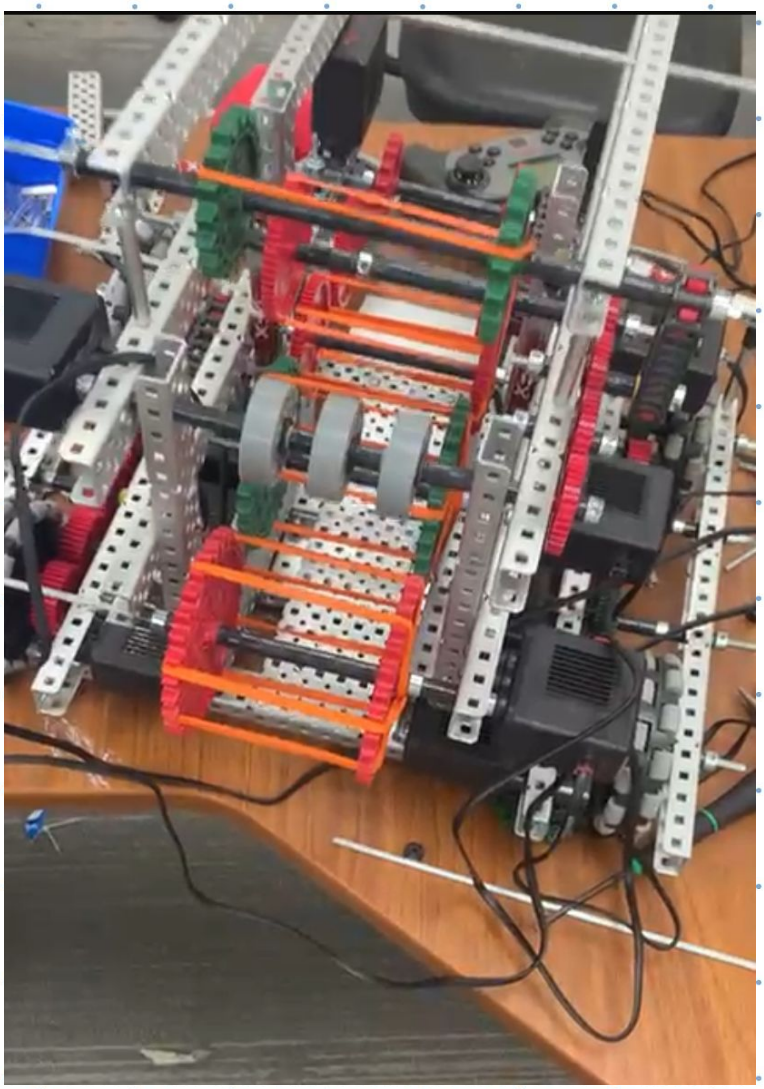
Theeb V2 (In Progress)

Theeb V2: Progress



Theeb V2 (In progress)

Theeb V2: Progress



Theeb V2 (In progress)

Theeb V2: Problems Faced

- Due to leadership decisions, Theeb V2 did not match the strategy we intended, as Theeb was decided to play a more defensive playstyle.
- Therefore, the base gear ratio and channel pathway had to be completely redesigned.
- There were minor issues/mistakes with regards to the construction itself.

Theeb V2: Solutions Brainstormed

- Change Gear Ratio to provide higher torque.
- Change to a S-channel pathway that matches the defensive playstyle.
- Optimize drivetrain to maximize space.

Gear Ratio Used

Theeb V2: Solutions (*With testing*)

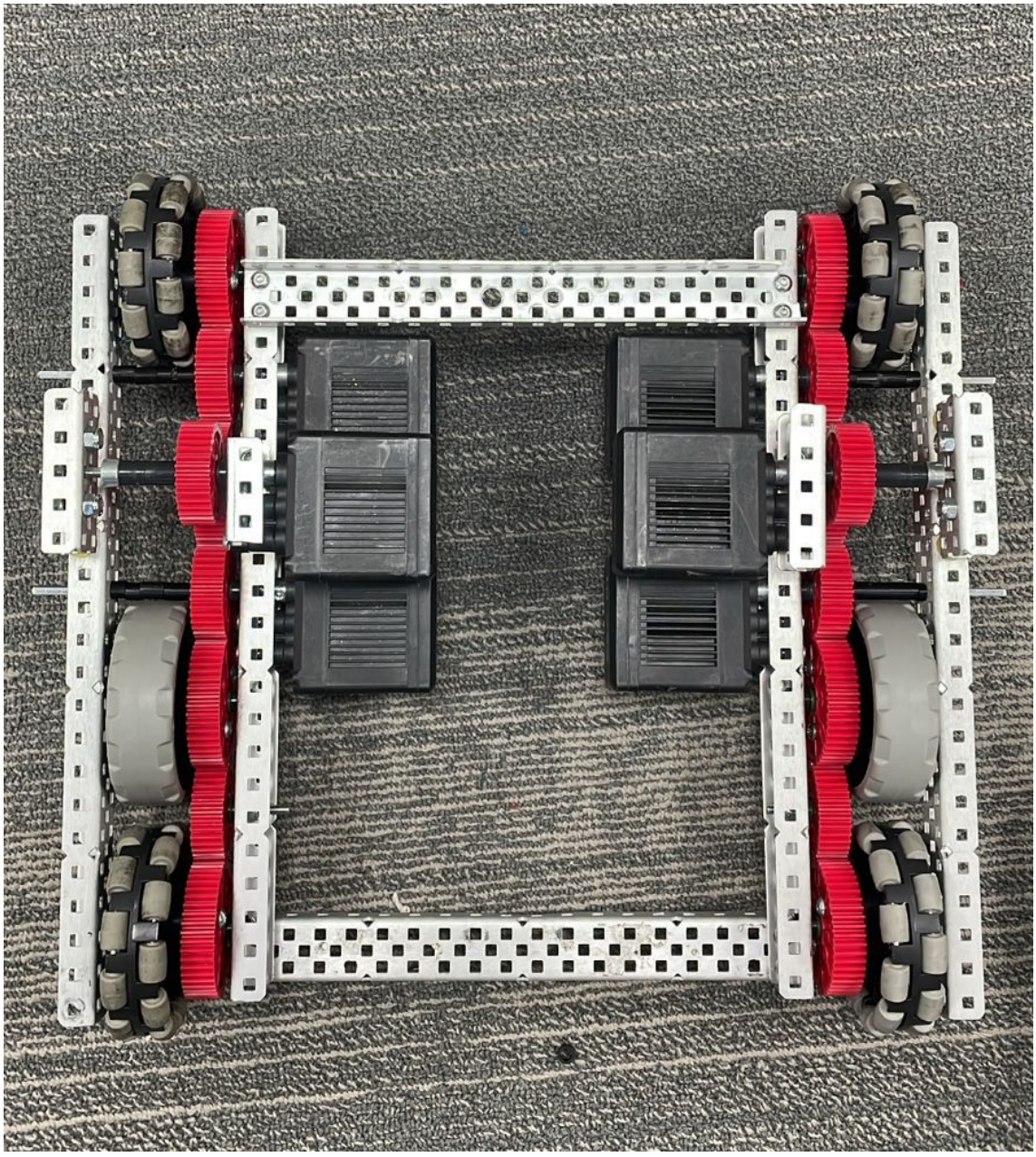
Feature	Spec (Moderate Pushback Build)	Notes
Ball capacity	8 balls	Slightly increased capacity
Intake speed (medium)	~100–140 RPM	Faster intake, still controlled
Weight (medium-heavy)	~7 kg	Lighter, but stable center of gravity
Speed of moving	Moderate: ~1.2–1.6 m/s	Noticeably quicker movement
Speed of rotating	Moderate: ~180–240°/sec	Faster turning, better
Outtake power	High: 85–100% motor power	Improved output, still needs grip

Theeb V3 Initial Design

Theeb's third design was made due to problems faced while building the base and new ideas the team found in Theeb V2.

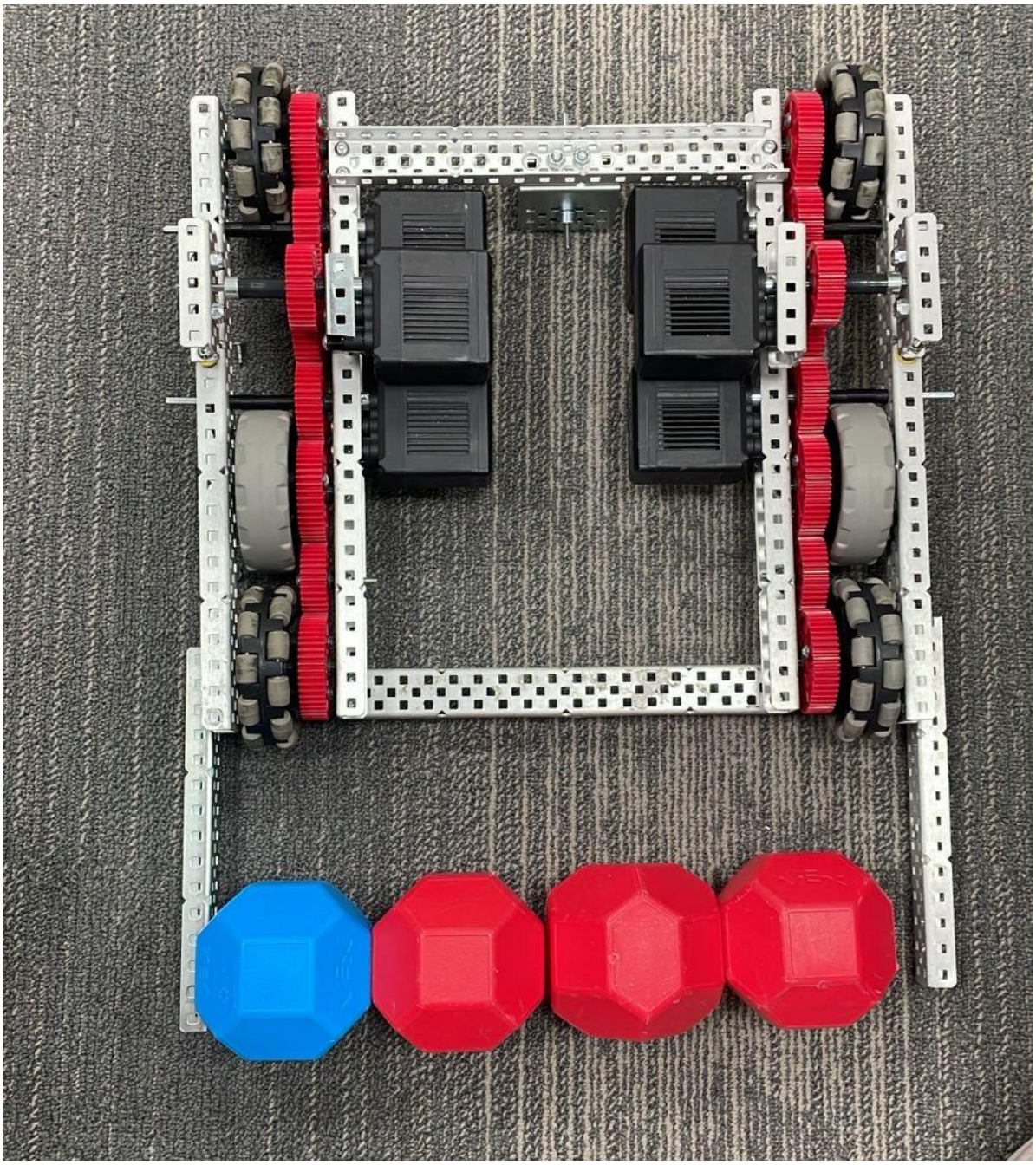
- **Base:** H-drive with 6 wheels and 6 motors
  - *Reasoning*: The base remained an H-drive, however, the gear ratio used initially used in the second design caused the wheels to be unresponsive and heavy to turn. As a result only the gear ratio and the placement of the motors was changed
- **Intake:** The intake remained the same with minor adjustments to the placement of the C-Channels to fit the new scoring mechanism
  - *Reasoning*: The intake was performing well under tests and no major revisions were required
- **Scoring Mechanism:** S-Shape
  - *Reasoning:* The old C design lacked enough space to hold an adequate amount of balls. The S-Shape will us to hold a bigger amount.

Theeb V3 Progress



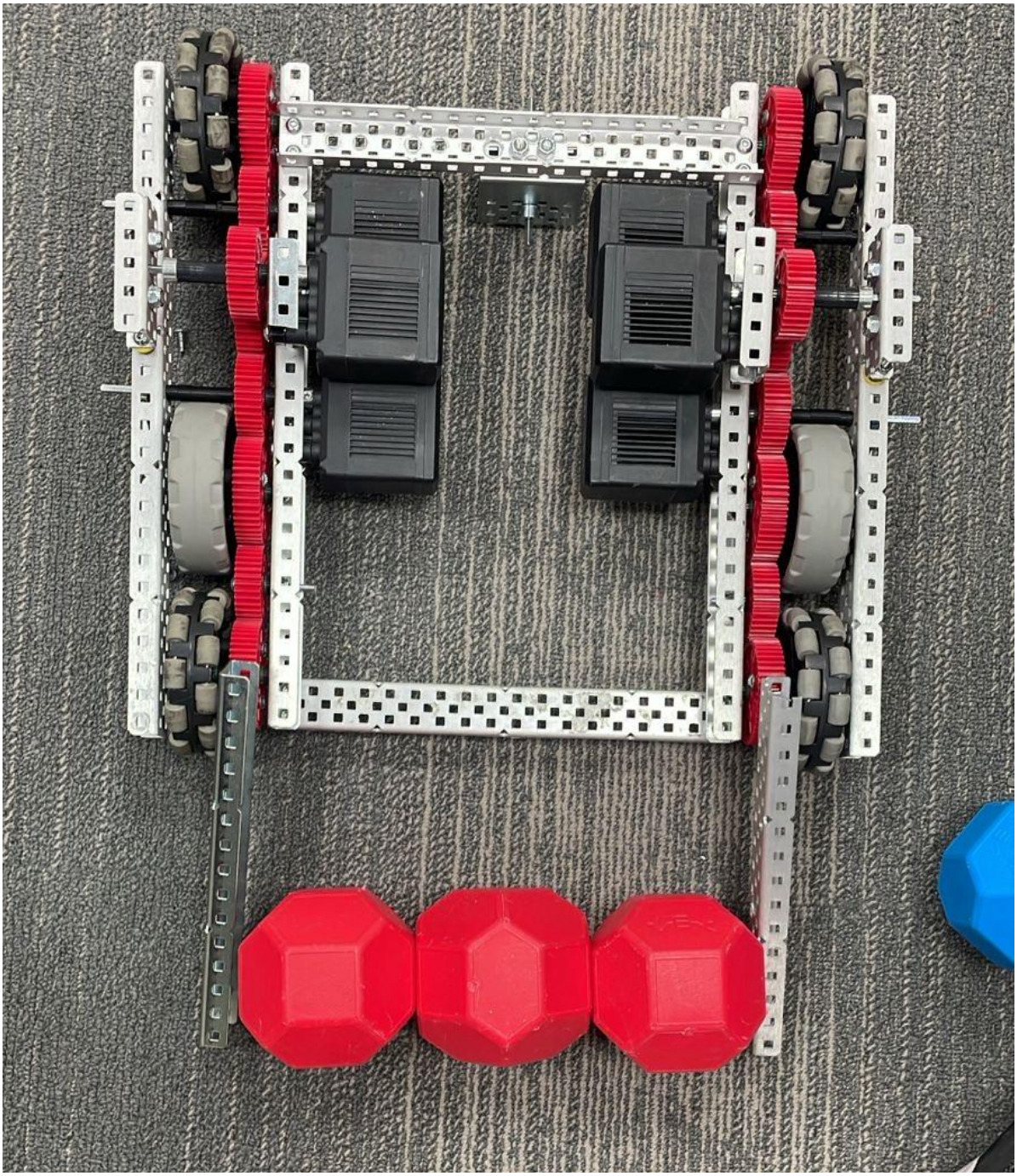
Theeb V3 (Base)

Theeb V3 Progress



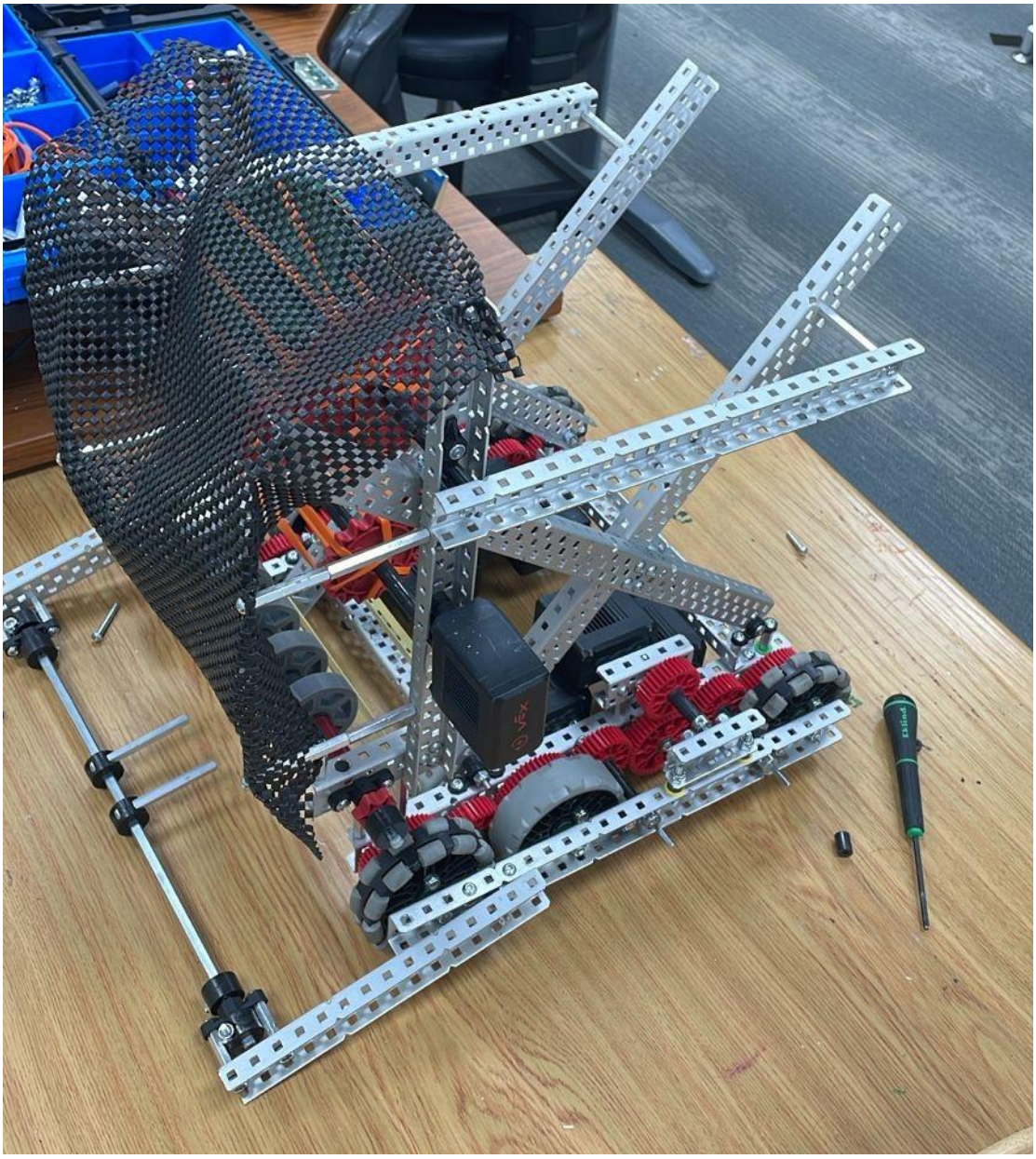
Testing length for match loader

Theeb V3 Progress



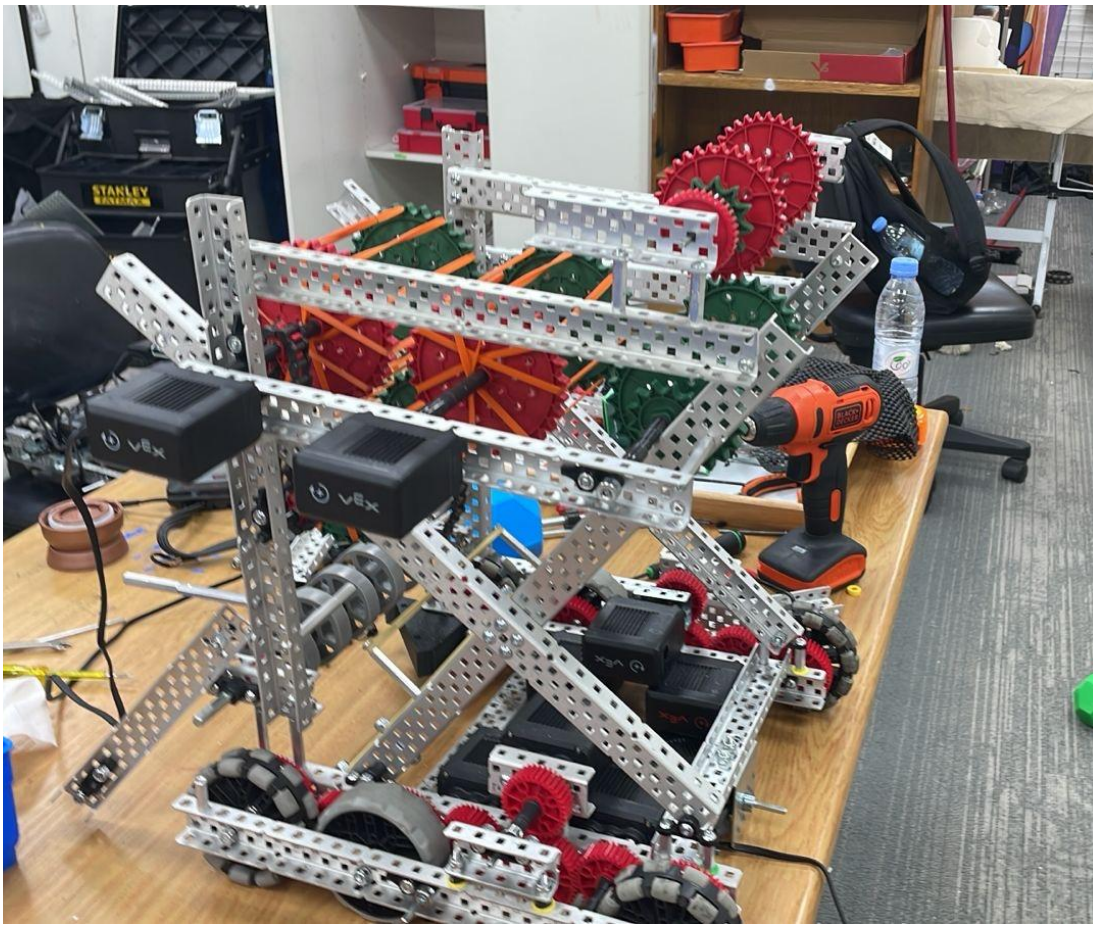
Testing length for match loader

Theeb V3 Progress



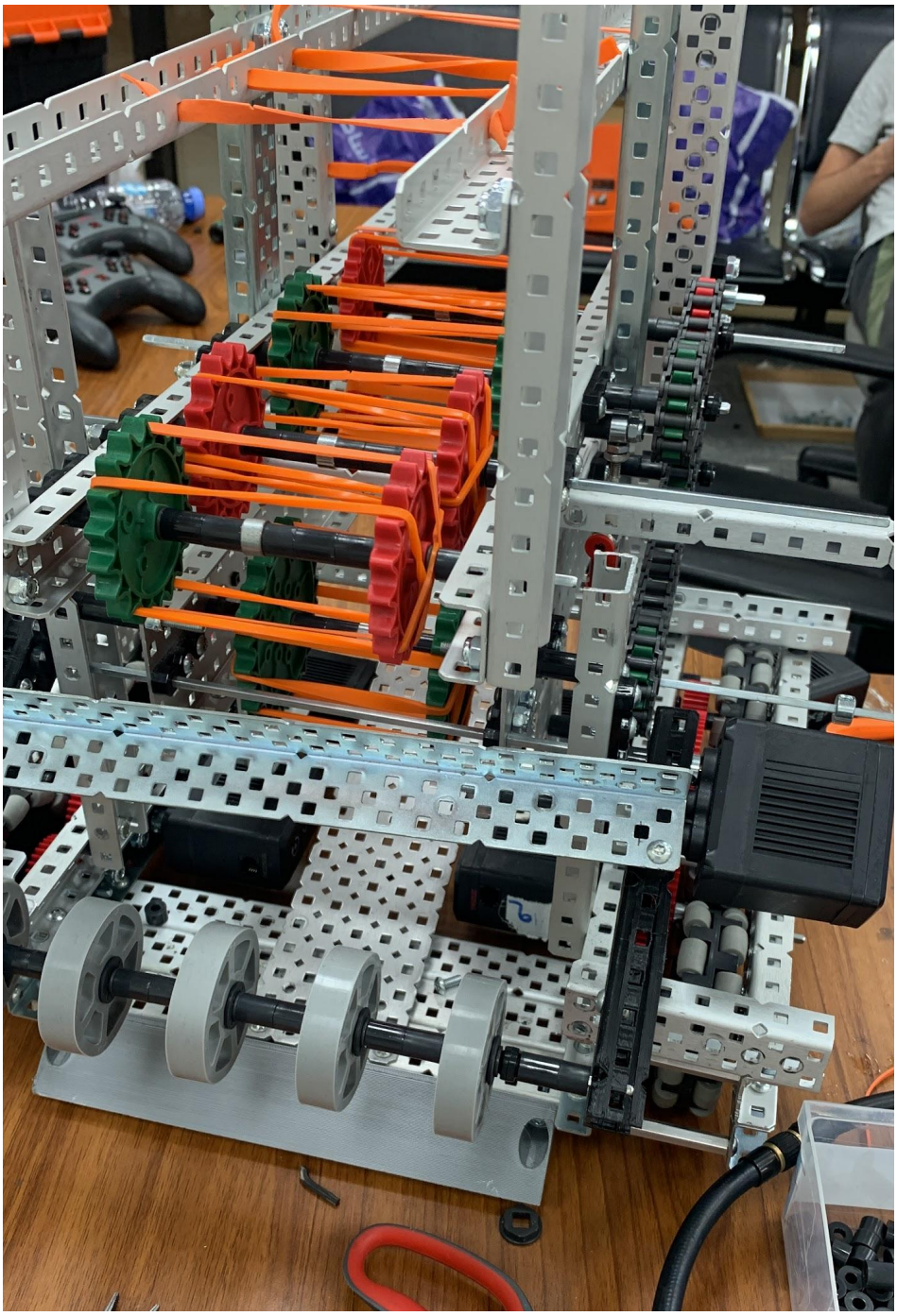
Theeb V3 (In progress)

Theeb V3 Progress



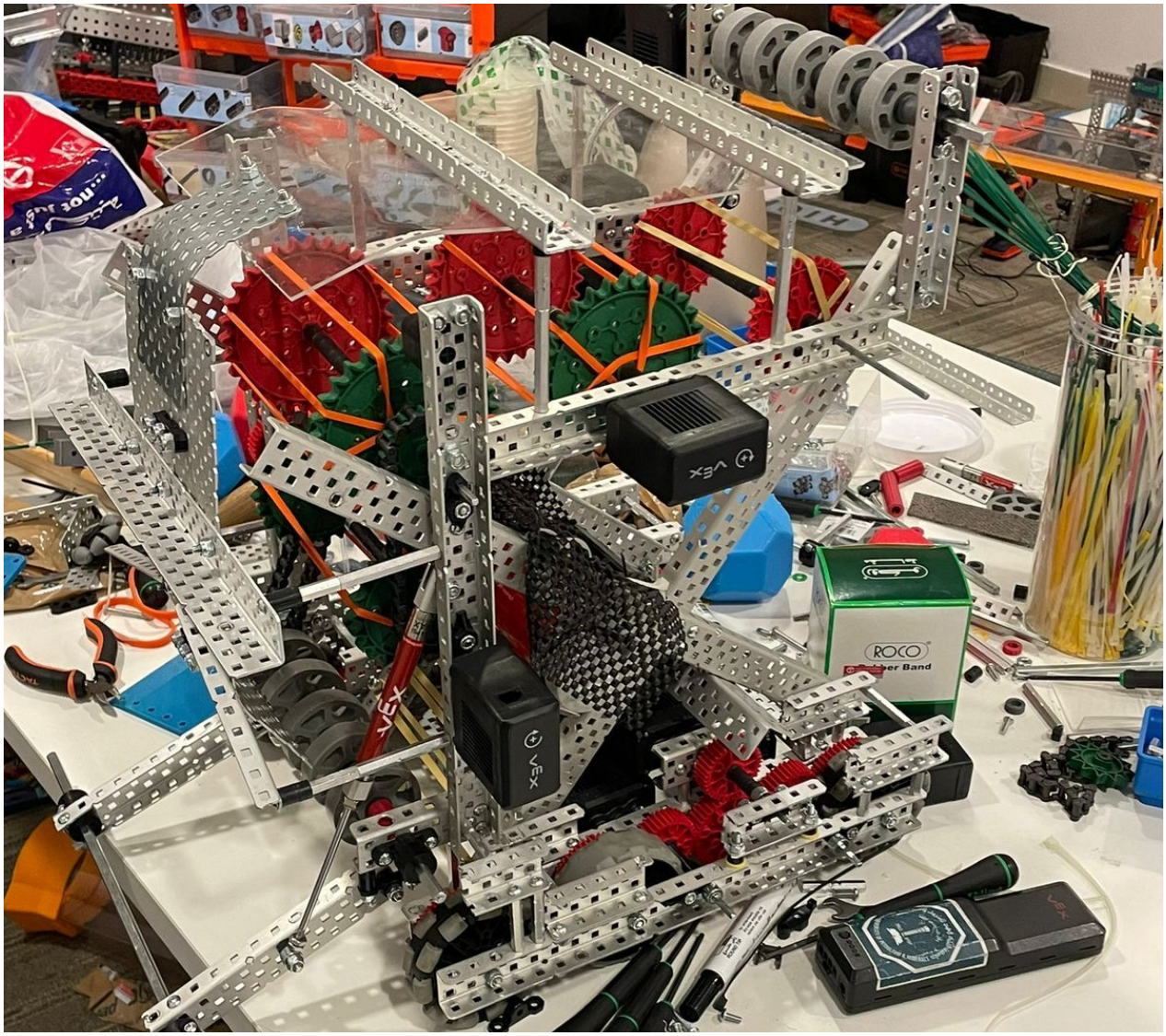
Theeb V3 (In progress)

Theeb V3 Progress



Theeb V3 (In progress)

Theeb V3 Progress



Theeb V3 (Fully built)

Theeb V3: Problems Faced

- A critical issue in the base was found were the wheels were heavy to turn and only one side of Theeb's base was able to move.
- The base was unstable and resulted in rumbling and unpredictable motion
- Using 6 wheels added a lot of stress in the motors and caused them to fidget in place
- The gear ratio used was not adequate to meet the speed we wanted
- The match loader did not properly drop the balls consistently in the loaders
- The design of V3 Theeb had a lot of empty space that resulted in an increased amount of dead zones in comparison with other versions

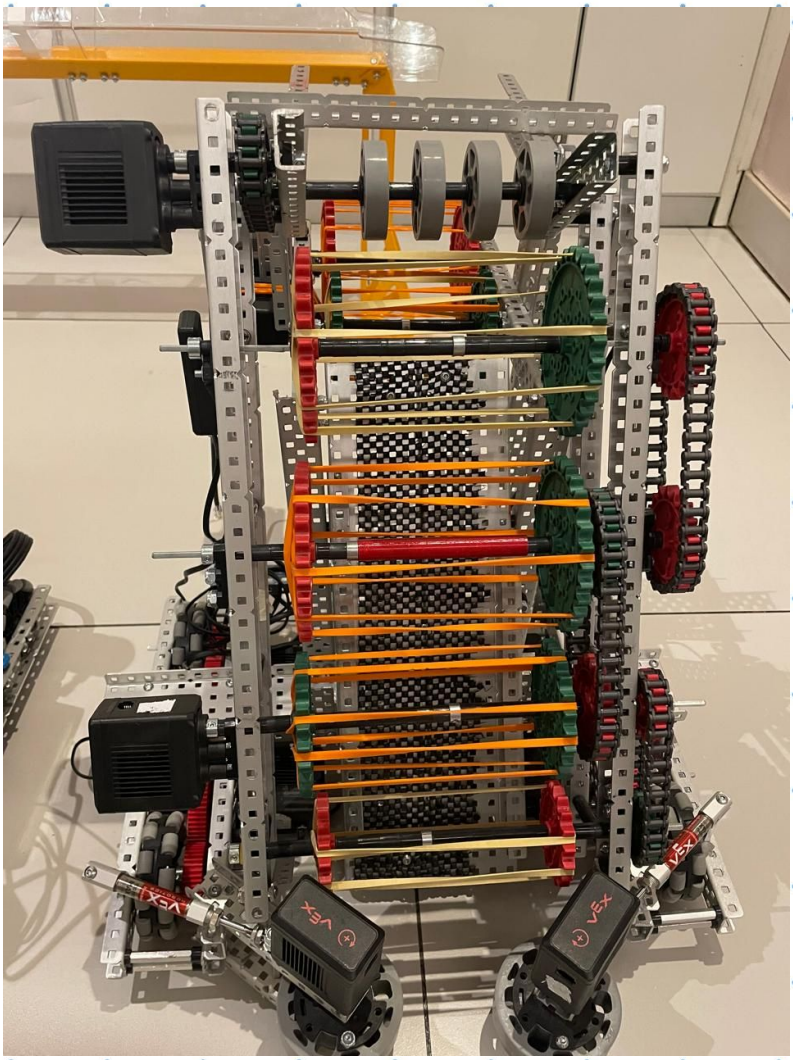
Theeb V3: Solutions Brainstormed

- Returning to a simpler design for the base would allow us to build, test and troubleshoot faster.
- Using our Initial design for the intake and removing the match loader would give us ample time to train the drivers due to the reduced testing time.
- Using a different shape that supplied us with a lot of power to push the balls powerfully to the long goal to dominate the control zone. While reducing the amount of dead zones is the most optimal solution
-

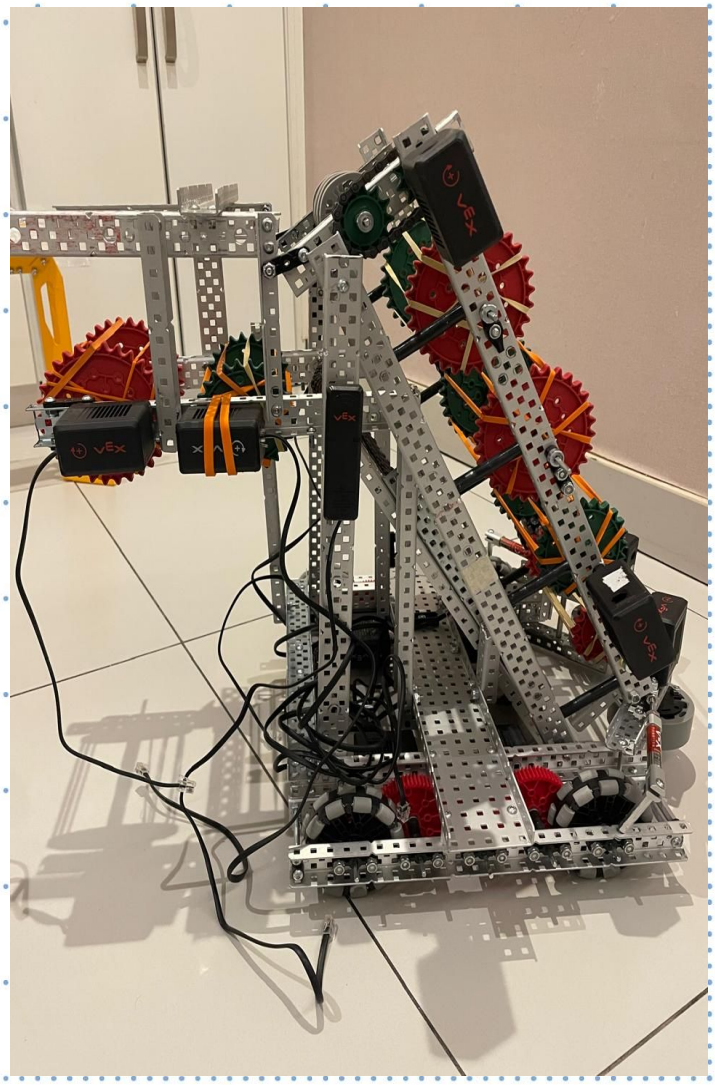
Theeb V4

A crucial error was found in **Theeb's** third base where the motors were unable to turn the wheels. Attempts to fix the base were unsuccessful which forced us to make a new design for **Theeb** to accommodate time constraints

- **Base:** H-drive with 4 wheels
  - *Reasoning*: With testing we found out that the 6 motors design was overkill and resulted in draining the battery quickly with minimal differences in performance
- **Intake:** Two extended metal arms with four spinning flex wheels and two pneumatic pistons that (de)compress
  - *Reasoning*: We returned to **Theeb's** initial intake as it was easy to implement and reliable to use in a match
- **Scoring Mechanism:** Linear
  - *Reasoning:* The linear shape allowed for fast scoring of the balls and minimize the potential for balls to get stuck in dead zones.
- The robot implements a **color sorting** mechanism to save ball capacity.

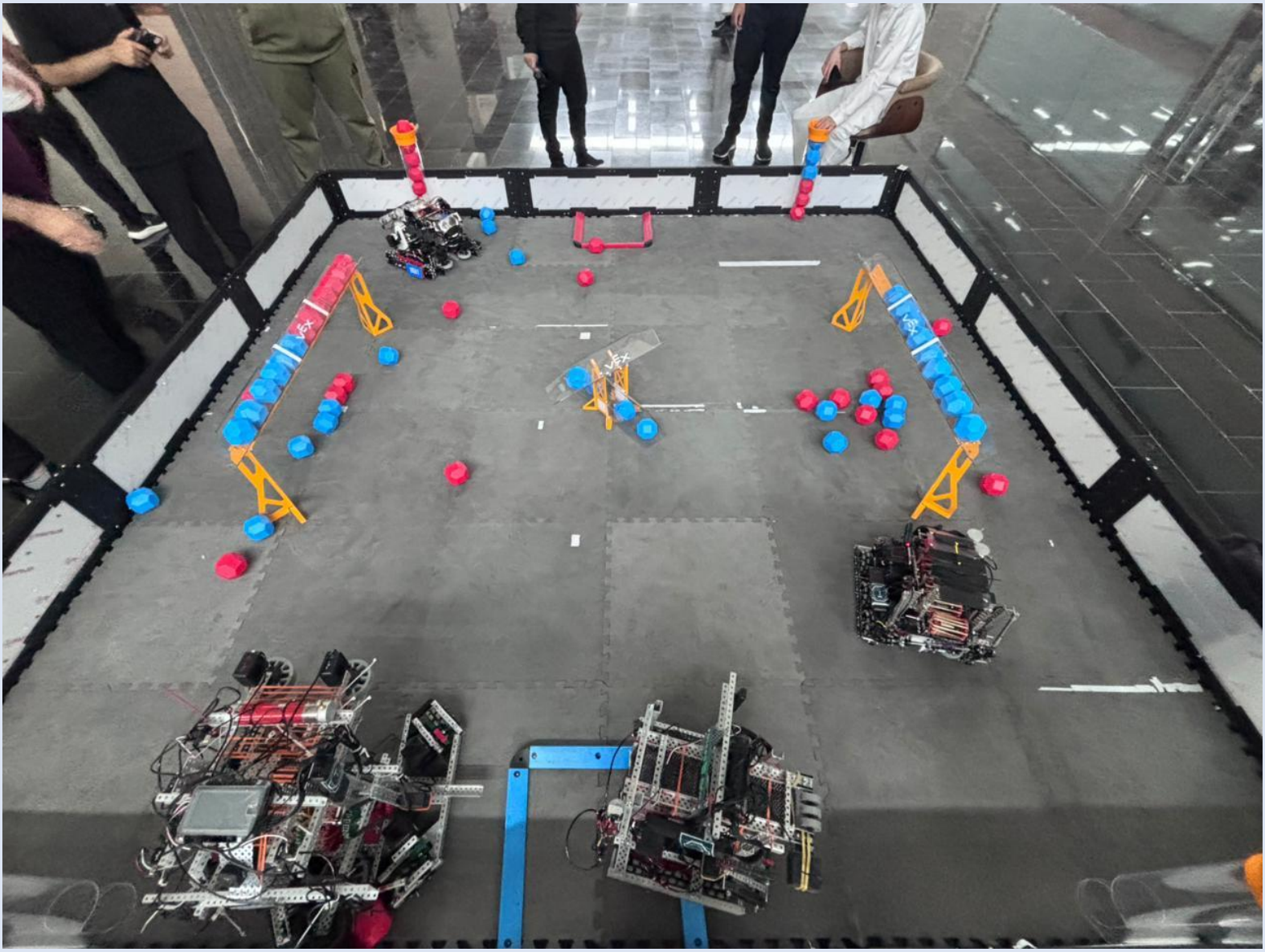


Theeb V4 (Front View)



Theeb V4 (Side View)

Friendly Matches: KFUPM Vs Umm Al-Qura University



KFUPM Vs Umm Al-Qura

On November 13th, KFUPM faced Umm Al-Qura University in multiple rounds of friendly matches.

The outcomes helped KRT understand the shortcomings and drawbacks of many design aspects of both Theeb and Sleekh. Below is a table of the match outcomes:

Team Name	Match Outcome	Result
KFUPM	32–40	Loss
Umm Al Qura University	40–32	Win
KFUPM	35–35	Draw
Umm Al Qura University	35–35	Draw
KFUPM	40–32	Win
Umm Al Qura University	32–40	Loss

KFUPM Vs Umm Al-Qura

What we learned:

- Our strategy was incomplete; our drivers did not coordinate well.
- More driving practice was needed.

Theeb Notes:

- Intake worked successfully for loaders, but struggled with ground intake.
- Theeb's size was **large, preventing it from going under the long goals and also preventing parking.**
- X-base maneuverability was effective but at the cost of reduced torque and speed.
- Descoring was essential to implement
- S-Channel was slow and prioritized storage capacity over speed.

KRT Team Meetings:

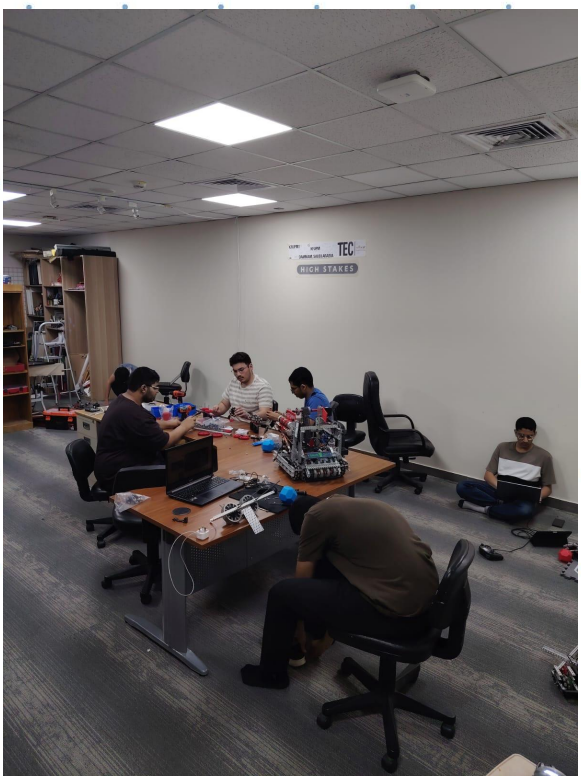
KRT Team Meeting 10/9/2025

- Introductory meeting to VEX pushback competition.
- A tutorial on all the VEX pushback parts, rules, and general game structure was introduced to the new members.
- The new members were tasked with building the introductory robot DEX found on the official VEX website: https://content.vexrobotics.com/docs/25-26/v5rc-push-back/docs/dex_build_instructions.pdf?v=2
- New members were trained and instructed carefully about how each VEX part was categorized and what they were used for.
- Task and role distribution was initiated for the new members.
- The task of the first week would be to
  a. Read the manual
  b. Watch content revolving around Vex U
  c. Build a tutorial robot

KRT Team Meeting 20/9/2025

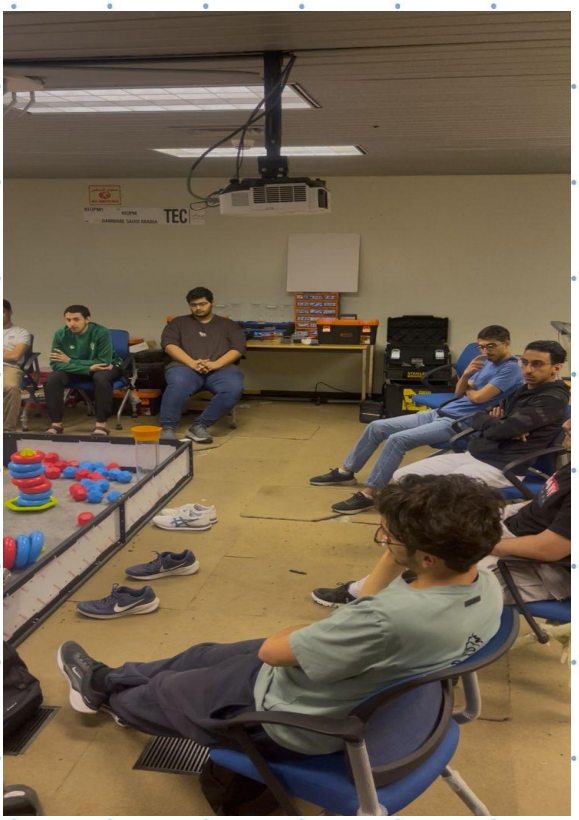
The team held a meeting to discuss the designs of **Sleekh** and **Theeb** the outcomes of the meeting was as follows:

- **Sleekh** and **Theeb** would be identical.
- Both would have an X-base to increase the directions of motions and make the Auto path easier.
- The C-Shape was chosen as a scoring mechanism due to its simplicity and high ball capacity
- The gear ratio for the base was yet to be chosen due to our unfamiliarity for the subject
- We would start working on the base from the next week



KRT Team Meeting 14/10/2025

- Urged members to continue building the robots before the initially planned Eastern province qualifiers match on December 28 by November 15.
- Urged the notebook documentaries to consistently keep track of all the work being made, especially the 3D design team.
- Discussed the relocation of the robotics lab to a new location at the KFUPM student mall.



KRT Team Meeting 16/11/2025

- Advised team members to remain during the midterm break to continue work on the new robots (Theeb V2 and Sleekh V1).
- Urged members to finish the robot builds before November 15 to face off against the female's team and a local high school robotics team.
- The initial strategy of the team and the new robot mechanisms were discussed.

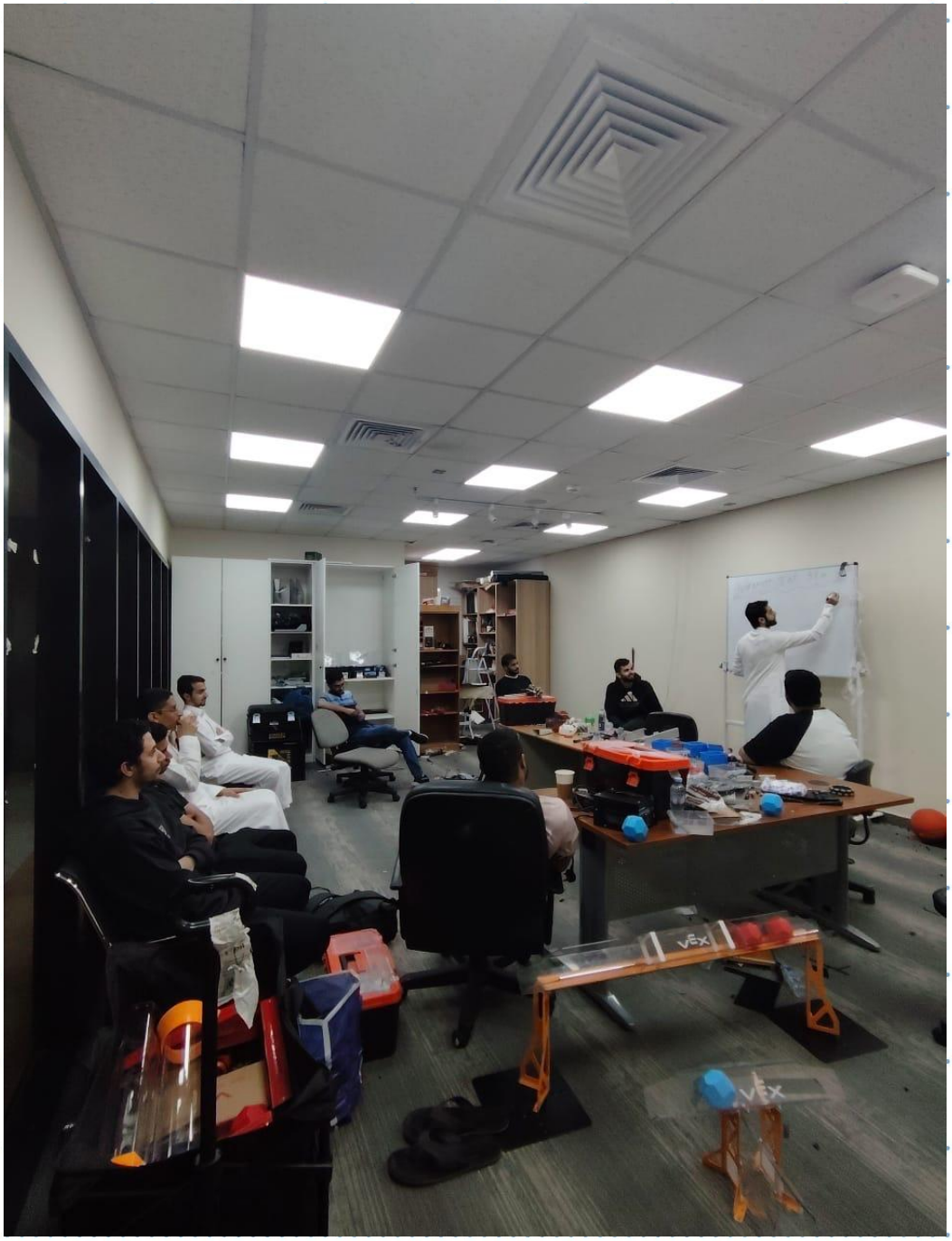
KRT Team Meeting 18/11/2025

- The KRT game strategy was finalized, and the team documented the important points in the notebook.

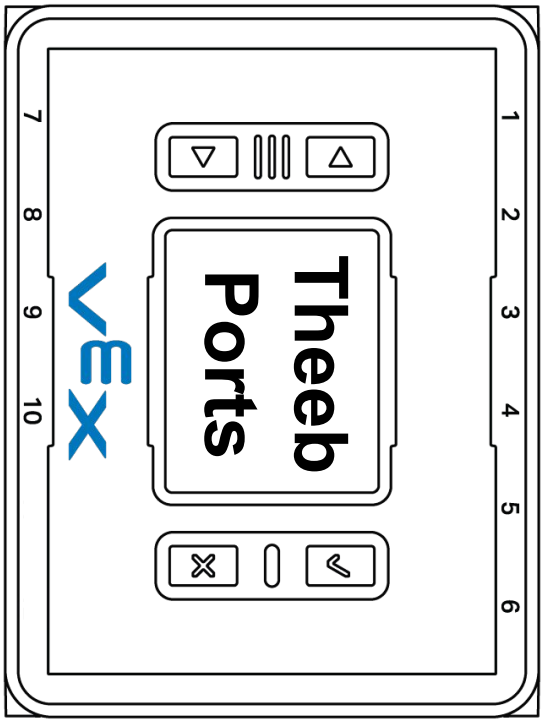
KRT Team Meeting 8/12/2025

- Member lab attendance was discussed, and members were encouraged to come more frequently there was a suggestion that a minimum of 8 hours a week to be mandatory.
- Role allocation was discussed. Dr. Hassan mentioned that from now on Sleekh, Theeb, and Software leaders will have to give specific tasks to the various members that are part of their teams.
- Strategies were discussed pertaining to the hardware of the robots, and the software team was directed to continue their work regarding testing out the sensors and not to wait on the hardware to be finalized.
- Finalized game strategies for both robots were asked for by Dr. Hassan.
- Bootcamp scheduled for the first 5 days of the vacation or the last 5 days of it and confirmation will be released later.
- It was stated that the Eastern Region competition will be around January 18-19.

KRT Team Meeting 8/12/2025



Motor 7 Right	7
Unused	8
Motor 1 Right	9
Motor 2 Right	10
Motor Left 1	19
Motor Left 2	20



1	Bluetooth
2	Unused
3	Motor 3 Right
4	Motor 4 Right
5	Motor 5 Right
6	Motor 6 Right