Technical Binder

Part I:

Season process

Process of the 2025 season

Steps:

1. Game analysis.
2. Defining requirements and rating requirements [SRR].
3. Thinking about systems and rating team systems [Prototype Concept].
4. Selecting alternatives [SDR].
5. First-stage prototyping [Initial Prototype].
6. Choosing final systems [PDR].
7. Advanced planning [Second prototypes + Robot prototype + Modeling].
8. Design review - final approval [CDR].
9. Manufacturing and assembly.
10. Problem solving and improvements.

Additional Points:

- Alumni roles.
- Building the field.
- Daily schedule.
- Status updates.
- Workshop opening time.

- Competition teams.
- Role of the head of teams (team leaders).

Step I - Game Analysis

Date: KICKOFF, 01.04.
Definition: The first part of the process is game analysis – trying to deeply understand the game and its rules.

In the first step – the challenge is presented: a joint screening of season videos is held.
After watching, there is a short food break to allow the team to release their excitement and give the captains time to organize the thinking groups.
Then, the group splits into thinking teams [which were prearranged with team leaders]. In the groups, several tasks are performed:

- Re-watching the season videos.
- Completing the document – Challenge Analysis (appears in the appendices) and answering the questions in it:
    - Learning from Previous Seasons: Trying to think about previous seasons with similar challenges.
    - Understanding the Game: Based on the manual, understanding what the robots need to do in each phase of the game (Autonomous, TELEOP, and END-GAME). Usually, this information can be found in the manual under the "MATCH PLAY" category.
    - Cycles: What is the robot's path in the game? Where does it pick up field pieces from? Where does it need to bring them? What is the scoring method – is it PICK & PLACE or cycles?
    - Scoring Table: Filling out the "Task Scoring" document with all scoring methods for the challenge and how many points each scoring method yields. Additionally, writing all the ways to earn RP (Ranking Points).

Reading the Manual: In the thinking group, the manual is reviewed, and important parts are written down in the Challenge Analysis document. Since the manual is long and most of it doesn't change, teams are allowed to focus on four main sections:

- ARENA: How is the arena structured? Where do you collect field pieces? Where do you place them? Where is the human player located?

- MATCH PLAY: How does the game work? What happens during the game? How many field pieces are there? (and what types?) Do the rules change during the game? What is the human player's role?

- RULE VIOLATIONS: What earns a yellow card? What earns a red card? What results in fouls? What are the penalties for fouls? What is not allowed in the game?

- GAME RULES: The game rules. It is advisable to go over the blue rules (those that change each year) and put less emphasis on the green rules.

Additional Notes: Relevant points that the team notices regarding the game/manual that should be highlighted.
Afterwards, everyone sleeps on the challenge [after an explanation about the next day], and the following day returns for thinking about the requirements and rating them.

Step II - Defining Requirements

Date: 01.05

Definition: Before we can move on to thinking about robots and systems, we must first define what we want the robot and its systems to do. Therefore, we need to define requirements for the robot. A requirement: what the robot will do (not how!).

We will begin with a quick group discussion at the start of the workday. During this discussion, we will explain mainly what we expect from the teams in terms of requirements ranking and the schedule for the day.

After the group discussion, we will continue working in thinking groups to define the requirements until the lunch break.

Each group will present and write down in the document – the Requirements Document (see appendices) the requirements they raise. That is, anything the robot can do or a general description of the robot [e.g., preferred weight and size relative to the field tasks]. Attention must also be given to control and software requirements!

Examples of requirements:

- Place a coral anywhere in the reef
- Collect from the floor
- Collect from the feeder

Not requirements:

- Collecting balls with a 4BAR
- Two-stage elevator

Regarding requirements that involve unspecified numbers [e.g., “place 3 corals in 7 seconds”], initially, they should be presented with unknowns [“shoot X corals in Y seconds”], and later, in the ranking stage, arbitrary data will be determined. The options should be limited.

The definition of requirements is completed by the lunch break. During the break, the strategy team will go over the requirements from all the teams and consolidate them into one document, filtering out irrelevant requirements.

After the lunch break (and other activities), the groups will reconvene to perform the requirements ranking stage (using the document that the strategy team consolidated).

Step III - Ranking Requirements - SRR

Date: 01.05

Definition: How much is a requirement worth? We may have many requirements, so we need to know what to focus on. Therefore, we will rank each requirement from 1 to 10 and then classify them into the categories of Must, Nice, and No.

The groups will rank the requirements in the Requirements Ranking document (see appendices), with the requirements already filled in by the strategy team after consolidating them during the lunch break.

Phase performed in groups:

Part One - Ranking 1-10:

After defining the requirements and the strategy team consolidating them, each group will begin ranking the requirements through discussion and using the conclusions from the challenge analysis done the night before.

You can also try thinking about ranking the requirements in the following way - cycles:

Cycles involve a round of robot actions during a match (e.g., go collect algae, collect, go shoot, and shoot). The cycles will be performed with a mock field (we will mark locations in the room), and the robot will be represented by someone on a wheeled chair.

The robot being simulated will be a different combination of requirements that were raised. After each attempt, we will record how much time it took, how many points were scored in the match. If a new requirement comes up during the cycles, it should be recorded and added.

After performing the cycles, each group will discuss each requirement. This discussion is very important, as it allows for accurate and realistic results.

Each group will rank its requirements on a scale of 1-10 based on how many points the requirement is expected to score and the level of difficulty (do not rank 10 for "shooting from anywhere on the field"). It is important to note that the ranking should be relative and not rank every requirement as 10, as certain requirements may affect others.

Each ranking of a requirement must be accompanied by a justification explaining the given ranking.

All of this should be summarized in an organized table with columns: Requirement - Score - Justification.

A group that does not justify its rankings will have its rankings disregarded in the strategy team meeting.

Part Two - Ranking Must, Nice, No:

Afterwards, each group will categorize their requirements into the MNn scale:

- MUST: Requirements that the robot must perform. Exactly what the robot will do.
- Nice: Requirements that would be nice for the robot to perform, but are not essential.
- No: Requirements the robot will not perform, and there is no need to focus on them.

At this stage, each group will define the robot's requirements according to its opinion and decide which requirements are essential for the system's performance.

Once the groups finish ranking the requirements, the strategy team [under the supervision of at least one of the captains and in the presence of the lead team] will meet to review the rankings from the groups. Based on the teams' rankings, they will set a strategy for the robot and adjust the requirement rankings accordingly. By the end of the evening, there will be a final Requirements Ranking document that has been written and updated by the strategy team.

Step IV - Thinking About Robot Profiles

Dates: 01.06-01.07

Definition: Now that we have finished defining what the robot will do, it's time to define how. In this stage, we will design different "profiles" for robots. A robot profile is an idea for a robot composed of a combination of systems.

Each thinking group will need to think of at least three robot profiles, from which two profiles will be selected to move on to the SDR meeting.

We will begin with a closing SRR meeting in a group setting, where the strategy team leader and captains will present the results of the requirements ranking (score and MNn) from the previous day to the entire group. This will "lock" the team's strategy into systems.

Afterward, we will continue in the thinking groups. Based on the ranked requirements and the established strategy, the members of the thinking groups will come up with ideas for "robot profiles" – that is, ideas for robots made up of combinations of systems. Each robot profile must meet all the MUST requirements (it may also meet NICE requirements, but this is not mandatory).

Each thinking group is required to come up with at least three different robot profile ideas, with at least two different systems for each robot element (e.g., in the 2023 season, one profile could include a robot with a telescopic claw and two-axis collection, and another profile could include a robot with an elevator and roller gripper). Of course, there are many ideas that combine different systems, e.g., a robot with a roller gripper and telescopic arm.

To propose a robot profile, you must fully complete the Robot Profile Proposal document (see appendices) with all the details of the robot profile. Each system in the profile should be detailed and explained well, and all criteria in the document should be addressed so we can ensure the proposal is complete and that all aspects have been considered.

Criteria [Summary of Proposal Structure]:

- Detailed explanation of the system: General and detailed description of the system.

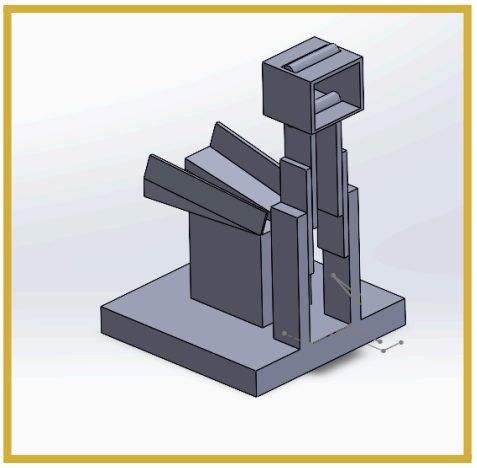
- Description of the process the system performs: From electronics to full movement (e.g., motor - winch - cable - end of the arm).
- Control processes with possible sensors: How will the software know what is happening in the system?
- System capabilities and goals: What the system is supposed to do, its unique features.
- Clear drawing of the system: A hand-drawn sketch of the system to help with understanding.
- Possible issues with the system: Trying to identify potential problems in advance.
- Specifications for integration between systems: How the different systems integrate into the robot.
- Requirements the robot meets: Explanation of how each system meets the requirements defined in the requirements ranking.
- Control methods and control requirements: How will the systems be controlled?
- Number of motors: How many motors are needed and how they will be distributed.
- Automation ideas: What automation can be implemented for each system or system combination?
- Links to similar robots: Searching for similar robots from past seasons.
- Material requirements: What materials are needed to build the robot?
- Team requirements related to the system: What will the teams need to do related to the system?
- Clear drawing of the robot: A general sketch of the robot.

Each robot profile will need to fill out all of these criteria in the Robot Profile Proposal document.

For each robot profile, a **blockbot** will also be required – a very basic 3D model, mainly based on geometric shapes without going into details. The blockbot will give us a visual representation of the robot profile and will serve as an aid in understanding the profile proposal.

This provides a full translation of your original text! Let me know if you need any further clarification or adjustments.

Example of a Blockbot



Step V - Ranking Profiles and Selecting Alternatives - SDR

Step V - Ranking Profiles and Selecting Alternatives - SDR
Dates: 08.01
Definition: In this stage, we need to evaluate the different profiles we proposed and determine which ones best meet the requirements. After we perform the ranking, we will choose the leading profiles to move forward into the prototyping stage.

Stage One: Profile Ranking (In Groups)
After we have developed the profile proposals, each group will rank each of the profiles it proposed based on how well each profile meets the defined requirements (such as the MUST requirements).

This stage includes the following:

- Requirement: How does each profile meet it?
- Each group will assess every profile and score it based on each requirement, including justifications and comments explaining why that score was given.

Scoring:
After the ranking, the groups will determine which proposal is better and which profile is more suitable to proceed to the next stage.

Group Discussion:
After the ranking, each group will discuss the data from the rankings and decide which two profiles will move on to the next stage (based on the goals of the requirements and strategy).
During the ranking stage, the leadership team will go between the groups and ensure that the ranking is done accurately and seriously. Graduates will also be assigned to assist groups as needed.

Stage Two: SDR Meeting (Final Filtering)
After the initial ranking, we move on to the next filtering stage in the SDR meeting. This meeting will include captains, mentors, the leadership team, heads of thinking groups, Rosh Team members, and the Strategy Rosh Team. The meeting will be broadcast live via

Zoom for anyone interested in watching, and team members are welcome to stay in the workshop and assist if needed.

In this meeting, each group will present the two profiles they selected, with details of each profile:

- Explanation of how each profile meets the requirements
- Explanation of how the robots work within the system
- Presentation of the Blockbots for each profile

After the profile and Blockbot presentations, overlapping profiles will be filtered out, and if any such profiles exist, the proposal will be reconsidered to only proceed with those that do not overlap. A focused discussion will be conducted, assuming the goal is to select two profiles that will progress to the next stage – Prototyping.

SDR Discussion Topics:

- Feasibility of the idea: How realistic is the idea to implement?
- Simplicity of the idea: Avoid overloading the teams with overly complicated ideas.
- Fit with the team's strategy: How well do the robots in the profiles match the requirements ranking defined earlier.

After the filtering, the two leading profiles will be selected to move on to the prototyping stage.

After the Selection – Preparing for the Next Stage:
After we have selected the two profiles that will move forward into the prototyping stage, each Rosh Team (Mechanical Rosh and Electronics Rosh) will sit with the system leads for each of the profiles and work with them to define objectives and prepare work plans for the prototypes.

Preparation for the prototyping stage includes:

- Defining objectives: Each profile can move to the prototyping stage only after clear objectives have been defined.
- Work plan: System leads will prepare a detailed work plan, including a schedule and deadlines.

- Approvals and monitoring: Each system lead will need to go through an approval process with the leadership team, including monitoring points and progress tracking for each system.

If approvals are received from all area managers, all system teams will begin working on the prototypes based on the information obtained during the SDR meetings and the leading developments.

Completion of this Stage:
After the SDR stage, the teams will be able to proceed to the prototyping stage, where physical models and components selected for the final profiles will be built.

This completes the translation! Let me know if you need any additional clarifications or help with anything else.

Stage VI - Initial Prototypes

Dates: 09.01-14.01

Definition:

This stage focuses on the physical construction of prototypes, where we begin testing and evaluating the systems in the chosen profiles. The goal is to learn and understand how the systems function, evaluate performance, and experiment with different versions until we reach the best option to advance to the PDR meeting.

Steps of the Initial Prototyping Phase:

- Work Planning and Goals:

  Before starting the model construction, each system leader will review the prototype phase work plan with their team, which includes approval from the team leader. The entire team will review the defined goals and examine a structured and calculated approach to the system.

- Prototype Execution:

  During this phase, each team will start building the system prototype while ensuring it meets the defined goals. This includes performance evaluation, version changes, and creating different variations if necessary. For example, if dealing with a roller gripper, different options for the number or types of wheels can be tested to determine which performs best.

- Process Documentation:

  Throughout the work, each team will document the entire process systematically. There are two main types of documentation:

  - Verbal Documentation: Each system leader will write a daily summary of the progress made, issues encountered, and tasks for the next day. The goal is for the team manager or anyone stepping in for an absent member to

seamlessly continue the work.

- Test Tables Documentation: During prototype development, numerous system performance tests will be conducted. All these tests must be documented in tables for future reference.

- Modeling Planning:
  Some modelers will begin working on the modeling plan for the various systems, including creating “instructions for the modeler” on how to model the system. This planning will streamline the modeling stage and optimize robot construction in the future.

- Process Control:
  The team leader is responsible for overseeing and managing the system teams under their supervision. There are stages in the process where system teams must receive team leader approval to proceed. The team leader reviews the prototype and provides feedback on issues that need to be resolved.

- End of the Prototyping Phase:
  At the end of this phase, we will gather all the data obtained from the prototypes to move on to the next stage, where the final systems for the season will be selected (PDR meeting).

Stage VII - Final Robot Profile Selection - PDR

Date: 14.01

Definition:
In the SDR meeting, we selected two profiles for the robots, each including several systems. At this stage, we will review the prototype phase results and choose a final profile, so we have a single profile that includes the robot's final systems. The meeting will follow the season leaders format and will be broadcasted to the group on Zoom.

- Preparation for the Meeting:
  Each system team must prepare a detailed explanation in advance about the system's test results (final results in documentation tables), along with a summary of the process they went through, including:
    - Level of difficulty
    - Problems encountered
    - Things that worked excellently
- Data Presentation:

  In the group meeting, each system leader and team leader will present their data and the final variation of their system. After the data presentation, we will hold a discussion, followed by a vote on the final systems from each profile. The main criteria for selection are:
    - System performance
    - Difficulty level in construction
    - Integration with other systems in the profile
    - Additional considerations raised during the discussion

- Distributing Systems Among Teams:
  After selecting the final systems, we will assign the systems to the system leaders. If necessary, we will reduce the number of teams or reallocate modelers between teams to fit the modeling to each system.

- Work Plan:

  After the systems are assigned, the leading team will sit with the system leaders and plan a work plan until the CDR meeting. The plan will include:
    - Defining goals for each team for the advanced planning phase

    - Defining tasks for each team with deadlines and control points (including modeling considerations)

- 2D Sketch:
  Immediately after the meeting, the modeling team leader will prepare a 2D sketch of the robot in SolidWorks. The sketch will show the integration between the systems and provide basic integration for the robot. It will also help find fixed dimensions that cannot be determined in the prototype phase (such as arm extension). This stage will be completed the same evening so the sketch is available to the team the next day at the start of work.

- End of Phase:

  The meeting concludes once we have the robot's final systems, system distribution among leaders, a work plan for each team, and a 2D sketch of the robot.

Stage VIII - Advanced Design

Dates: 15.01-24.01

Definition:

The systems for the season have been defined, and now we move on to the complete and detailed design of the robot. During this stage, the modeling team works almost continuously on the robot's design, while the other teams focus on integrating all systems together on the robot, answering questions from the modeling team (such as determining specific dimensions), working on the electronics system, and continuing efforts to improve and optimize the robot's final systems.

Objective:

The goal of this stage is to finalize everything that hasn't been determined yet—final system dimensions, integration, and electronics. This will be achieved through several aspects:

- Advanced System Prototype (Quick and Final Model):

  The prototype built in the previous phase was mainly for technical testing, with less emphasis on precision and without control systems. At this stage, a complete system prototype will be built to function like the final system, including electronics and control. The objective is to build with enough precision to match dimensions for the final build.

  - Prototype Robot (Optional - If Time Permits):

    All advanced prototypes should eventually combine into a fully functional robot. For example: Orbit 2016 prototype robot. This allows for testing integration, code implementation, and control.

- Full Robot Modeling:

  The best way to test integration on precise robot dimensions is through complete modeling. During the advanced planning phase, the modeling team works together to complete a full assembly of the robot, including all systems, so it can be handed

off to the manufacturing team.

- System Leaders – System Team Leaders (STLs):

  It is the responsibility of the system leaders, along with the STLs, to ensure the preparation of equipment lists needed for the system that is not available in the workshop:

  - International Orders: Must be prepared within two days after the PDR meeting to ensure the orders arrive in time for assembly.
  - Local Orders: Must be prepared by the CDR meeting.

- Manufacturing List:

  To ensure organized manufacturing later on, a list of all parts that need to be manufactured, sorted by system, must be prepared in advance.

- Build Instructions (Work Plan for Post-CDR Stages):

  Since we plan the robot first and then manufacture and build it, we can plan all the work in advance, even down to the level of screws. Each system leader, together with the STL, will prepare a detailed work plan that will be integrated into Monday (project management platform).

Stage IX - Design Review and Final Approval - CDR

Date: 25.01

Definition:

This stage marks the conclusion of the robot design process. Up until now, we have worked on planning how the robot will look and function. After this meeting, we aim to begin executing the “build instructions” and proceed with constructing the complete robot.

CDR Meeting:

After the robot’s final design is completed, a group meeting [CDR] is held. In this meeting, each system leader presents the full robot model, highlights the work done, discusses issues encountered and how they were resolved, and talks about the performance of the robot’s final systems.

Additionally, since this meeting provides a complete view of the robot, the team will hold a group vote to select the robot’s name.

After the presentation of the model and the discussions, all remaining questions will be addressed, and answers will be provided for every raised issue. Then, we will review the “build instructions,” and work can officially begin!

Stage X - Manufacturing, Assembly, and Wiring

Dates:

- Manufacturing: 26.01-01.02
- Assembly: 30.01-08.02

Definition:

At this stage, we begin working according to the pre-prepared work plan and follow it through to completion. Manufacturing is carried out in the order listed in the manufacturing list and is documented in the production tracking table (see appendices).

Manufacturing Process:

Once the manufacturing of a system is complete, the mechanics team starts assembling the system and continues the process until the entire robot is fully built. After assembly, the electronics team receives the robot for wiring, which takes up to 3 days. Once wiring is complete, the robot is transferred to the software team.

No Resting!

Throughout this entire process, the mechanics and electronics teams do not stop working. When waiting for parts or during manufacturing, they continue working on improving and optimizing the robot's systems, ensuring no delays in the process.

If a prototype robot is available, the software team works on it until the final robot is fully assembled and ready.

Stage 11 - Problem Solving and Improvements

Dates: 09.02-23.02 [One Day Before the First Competition]

Definition:

At this stage, the goal is to identify and fix all issues found in the robot, as well as make improvements if necessary. As always, new problems are likely to be discovered during use.

Problem Detection and Correction Stage:

The objective of this stage is to achieve a highly reliable robot that performs optimally in competitions.

- Creating a Checklist:
  We will create a list of things to examine on the robot, including extreme scenarios that may occur.

- Scanning the Robot:
  We will go through the checklist step by step to identify any issues.

- Problem Resolution:
  Once issues are identified, we will create work plans to address each problem systematically.

- Repetition:
  We will repeat the detection and correction process continuously until the competition, constantly improving the robot.

Driver Training:

During this stage, alongside robot repairs, we will conduct driver training to ensure the drivers are well-practiced with the system and know how to operate the robot optimally during the competition.

You can see the different dates in an organised way:

	יום	ראשון	שני	שלישי	רביעי	חמישי	שישי	שבת
0	תאריך	29.12	30.12	31.12	1.1	2.1	3.1	4.1
	משימה							KICKOFF! ניתוח המשחק
	אירוע נוסף							בניית מגרש
1	יום	ראשון	שני	שלישי	רביעי	חמישי	שישי	שבת
	תאריך	5.1	6.1	7.1	8.1	9.1	10.1	11.1
	משימה	**הגדרת דרישות ודירוג דרישות**	**SRR פגישת** חשיבה על פרופילים	סיום חשיבה על פרופילים	דירוג פרופילי רובוט צוותי **בחירת - SDR פגישת חלופות**	בניית פרוטוטייפים בסיסיים	בניית פרוטוטייפים בסיסיים	בניית פרוטוטייפים בסיסיים
	אירוע נוסף	בניית מגרש	בניית מגרש	בניית מגרש	בניית מגרש			
2	יום	ראשון	שני	שלישי	רביעי	חמישי	שישי	שבת
	תאריך	12.1	13.1	14.1	15.1	16.1	17.1	18.1
	משימה	בניית פרוטוטייפים בסיסיים	בניית פרוטוטייפים בסיסיים	סיום פרוטוטיפים בסיסיים **בחירת - PDR פגישת מערכות סופיות**	תכנון רובוט מתקדם	תכנון רובוט מתקדם	תכנון רובוט מתקדם	תכנון רובוט מתקדם
	אירוע נוסף							
3	יום	ראשון	שני	שלישי	רביעי	חמישי	שישי	שבת
	תאריך	19.1	20.2	21.1	22.1	23.1	24.1	25.1
	משימה	תכנון רובוט מתקדם	תכנון רובוט מתקדם	תכנון רובוט מתקדם	תכנון רובוט מתקדם	תכנון רובוט מתקדם	תכנון רובוט מתקדם	**קביעת - CDR פגישת עיצוב רובוט סופי**
	אירוע נוסף							
4	יום	ראשון	שני	שלישי	רביעי	חמישי	שישי	שבת
	תאריך	26.1	27.1	28.1	29.1	30.1	31.1	1.2
	משימה	ייצור חלקים	ייצור חלקים	ייצור חלקים	ייצור חלקים	ייצור חלקים והרכבת הרובוט	ייצור חלקים והרכבת הרובוט	ייצור חלקים והרכבת הרובוט
	אירוע נוסף							
5	יום	ראשון	שני	שלישי	רביעי	חמישי	שישי	שבת
	תאריך	2.2	3.2	4.2	5.2	6.2	7.2	8.2
	משימה	חיווט הרובוט	חיווט הרובוט	חיווט הרובוט	חיווט הרובוט	חיווט הרובוט	חיווט הרובוט	**סיום הרכבת הרובוט!**
	אירוע נוסף							
6	יום	ראשון	שני	שלישי	רביעי	חמישי	שישי	שבת
	תאריך	9.2	10.2	11.2	12.2	13.2	14.2	15.2
	משימה	תוכנה פתירת בעיות ושיפורים	תוכנה פתירת בעיות ושיפורים	תוכנה פתירת בעיות ושיפורים	תוכנה פתירת בעיות ושיפורים	תוכנה פתירת בעיות ושיפורים	תוכנה פתירת בעיות ושיפורים	תוכנה פתירת בעיות ושיפורים
	אירוע נוסף							
7	יום	ראשון	שני	שלישי	רביעי	חמישי	שישי	שבת
	תאריך	16.2	17.2	18.2	19.2	20.2	21.2	22.2
	משימה	תוכנה פתירת בעיות ושיפורים	תוכנה פתירת בעיות ושיפורים	תוכנה פתירת בעיות ושיפורים	תוכנה פתירת בעיות ושיפורים	אימוני נהגים	אימוני נהגים	אימוני נהגים
	אירוע נוסף							
8	יום	ראשון	שני	שלישי	רביעי	חמישי	שישי	שבת
	תאריך	23.2	24.2	25.2	26.2	27.2	28.2	1.3
	משימה	אימוני נהגים	**ISR District #1**			**ISR District #2**		
	אירוע נוסף							
	יום	ראשון	שני	שלישי	רביעי	חמישי	שישי	שבת
	תאריך	2.3	3.3	4.3	5.3	6.3	7.3	8.3

Additional Points in the Season Process:

Roles and Responsibilities Overview

- Alumni Role:
  When an alumnus plans to visit the workshop, it is recommended they notify the captains so they can be assigned a relevant task and location to help. However, surprise visits are also welcome, and the person in charge of the work will determine where the alumnus can assist or let them decide.

- Mentorship Program:
  To optimally integrate new members, each experienced team member (11th-12th graders) will mentor one or two younger members. The mentor ensures their mentees attend, engage in activities, and checks in on their progress. This approach fosters team cohesion, effective supervision, and increases motivation to participate.

- Field Construction:
  Two team members will be exempt from the planning stages and will be responsible for building the field (with support from available team members), ideally by the SDR meeting and no later than the PDR meeting. Alumni are encouraged to help during the kickoff evening and throughout the season.

- Daily Workshop Organization:
  The workshop will be tidied up twice each workday.

    - Before Dinner: A quick organization to ensure all tools are returned to their place.
    - End of the Workday: A thorough clean-up before locking up the workshop to ensure it's ready for the next day.

- Status Meetings:
  The leadership team holds a weekly status meeting every Saturday night to review the week's achievements and anticipate potential challenges for the upcoming week, along with solutions.

Additionally, at the start or end of each phase, a seasonal leadership meeting is held to discuss improvements for the upcoming stage (also included in SDR and PDR meetings).

- Workshop Opening Time:
  In the 2022 season, complaints were raised regarding the inconsistent opening time, giving the impression that the workshop was not a consistent place to be. On the other hand, an STL must be present to open the workshop.

  Therefore, the approach is adjusted:

  - A fixed “workshop opening time” will be determined and may vary as needed.
  - The keyholder group will coordinate who can open the workshop, and the opening time will be announced at the end of the previous workday.
  - The workshop is only open to those who have finished classes. Students with ongoing classes may only arrive afterward, except in exceptional cases approved by the captains.

- Competition Teams:

  The drivers, coach, and STLs for strategy, pit, photography, and public relations will be selected before the season begins. Other roles will be chosen during the season based on their contributions (e.g., mechanics team members who worked extensively on the robot are likely to be on the pit crew).

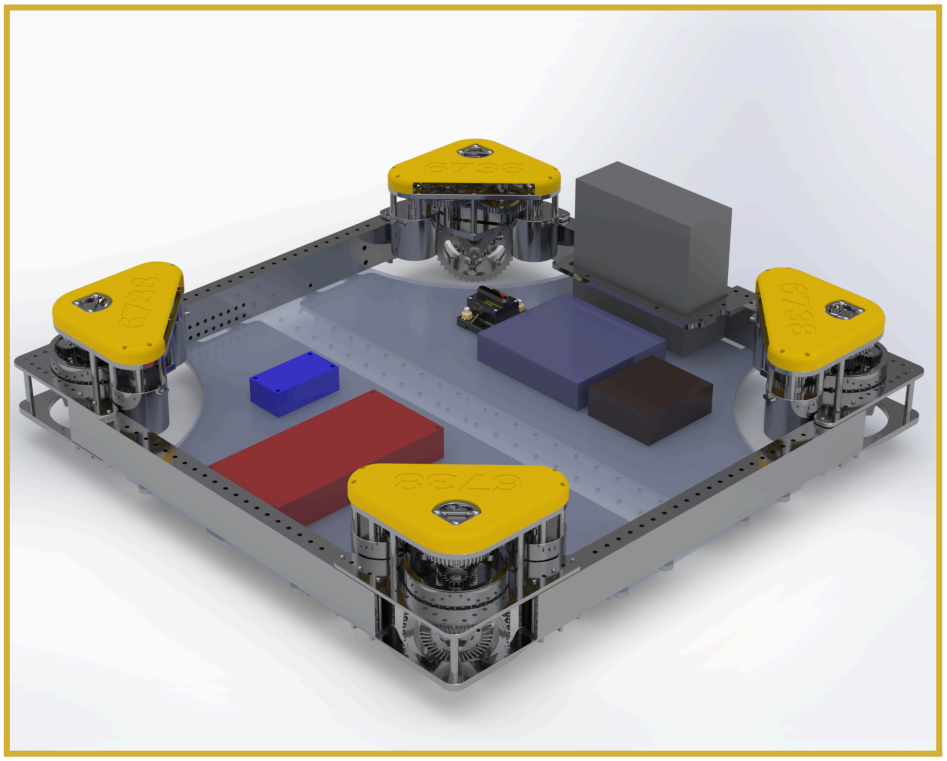
  The selection process for competition teams begins after the PDR meeting, starting with strategy team members, STL for scouting, and community speakers. Other roles will be assigned around two to three weeks before the first competition.

- STL Role:
  Since STLs are not system leaders, their primary responsibility is oversight. They ensure workflows run smoothly, confirm integration, and assist where needed.
  Additionally, between the SDR and PDR meetings, STLs are responsible for their robot profile.

Part II:

Mechanics

The Chassis

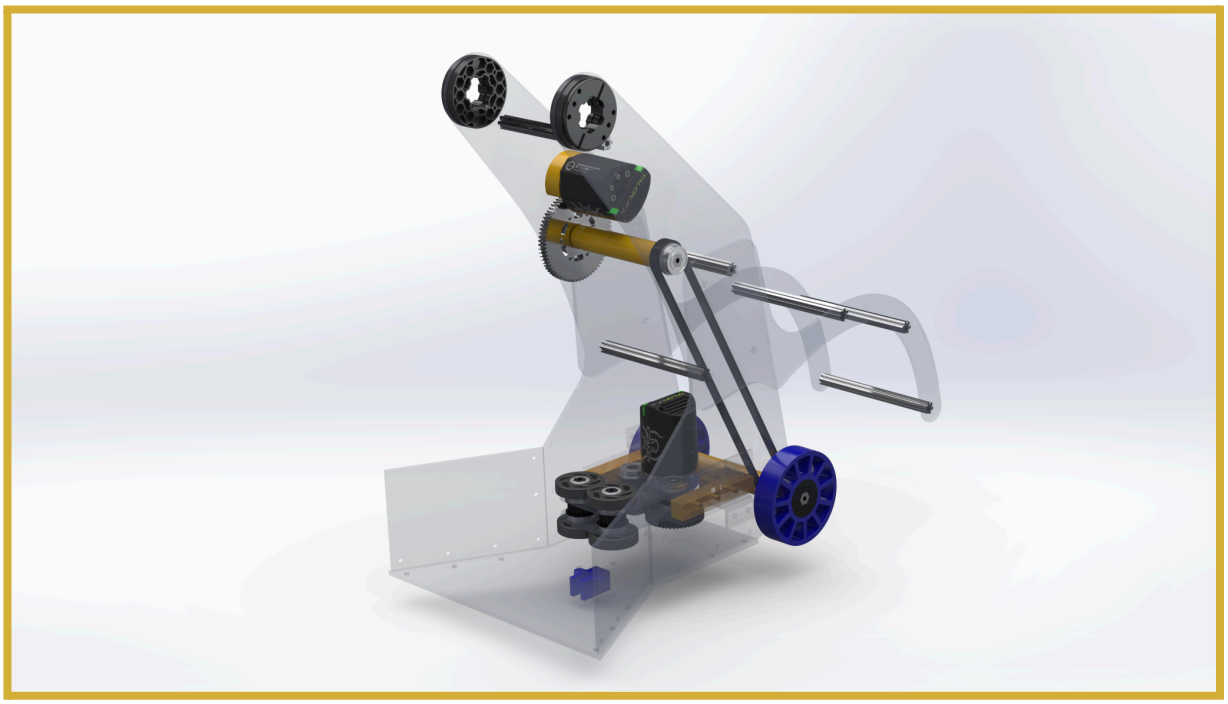


The robot was meticulously designed with a focus on enhancing its capabilities across a range of areas, with the aim of ensuring optimal performance in varying and challenging conditions. Its cubic structure, measuring 70x70 cm, was carefully chosen to ensure maximum stability and balance, essential attributes for performing complex and precise tasks. The robot's frame, constructed from strong aluminum profiles (3X20X50 mm), provides high resistance to mechanical loads, ensuring the robot's long-term survivability, even in harsh and intensive conditions.

The electronics board, which serves as the robot's control center, is made of lightweight and durable polycarbonate, a material that protects the sensitive electronic components from potential damage and ensures their proper functioning over time. The robot is equipped with MK4I drive modules, known for their high performance and adaptability to changing terrain conditions, while improving maneuverability and navigation. The drive system includes an L3 reduction unit with a gear ratio of 6.12:1, enabling high responsiveness and improved performance, and bringing the robot's maximum speed to 5 meters per second, while maintaining stability and accuracy, even in dynamic and changing conditions.

NEO motors were selected for the rotation system, which excel in high power and precise control, allowing the robot to perform rapid and accurate rotational movements, while improving responsiveness and maneuverability, and providing a quick response to environmental changes. For the robot's general propulsion, a KRAKEN motor is integrated, ensuring fast and efficient movement while maintaining stability in varying conditions, and enabling the robot to handle a wide range of tasks, including those requiring power and speed. The integration of these advanced components ensures exceptional performance and maximum stability in a variety of tasks, and allows the robot to successfully cope with complex challenges, while improving its capabilities in various fields, and providing a precise and efficient response to changing needs.

Griper



The gripper's rotation axis, the MAX Spline Shaft, is used to drive the gripper in a precise rotational motion. The shaft is connected to two KRAKEN motors with a gear ratio of 47.6:1, enabling high performance and maximum control over the system's movement. The motors are connected to two CUSTOM MADE gearboxes, which provide the necessary power to drive the shaft smoothly and stably.

In addition, the system includes chains connecting the gearboxes to the MAX Spline Shaft, with the elevator supporting the system and enabling rotational operation. The MAX Spline Shaft is characterized by exceptional structural strength and high precision, while maintaining stability throughout the process. It is firmly connected to the elevator stages, allowing it to move freely while maintaining maximum rotational accuracy. In fact, this shaft forms the backbone of the rotation system, with the combination of motors and gearboxes enabling balanced and precise driving, ensuring optimal functioning of the entire system.

The MAX Spline Shaft is connected to two polycarbonate plates, precisely positioned and attached to the gripper. These plates allow the gripper to perform very precise rotation, thereby enabling it to perform the required actions smoothly and stably. A HEX axis is also attached to the polycarbonate plates, on which 3-inch wheels are mounted. These wheels play an important role in the ALGAE collection system and work in cooperation with

additional polycarbonate plates that make up the entire ALGAE system and hold the ALGAE firmly.

The combination of the HEX axis with the wheels and the polycarbonate plates allows the robot to rotate the gripper precisely and stably, while maintaining optimal power and grip on the ALGAE. Every component in the system, from the polycarbonate plates to the wheels themselves, has been carefully designed to ensure smooth operation while maintaining a high level of accuracy. This design ensures that the process is reliable and efficient, so that the robot can collect and transfer the ALGAE in the best possible way throughout the competition.

The system has been designed precisely and professionally so that it can perform all the required tasks optimally. First, it is designed to lower the ALGAE from the REEF in the most efficient way, while maintaining stability and high performance throughout the process. After lowering the ALGAE from the REEF, the system enables smooth and rapid transfer of the ALGAE to the PROCESSOR. In addition, the system is capable of shooting the ALGAE into the NET accurately and efficiently, in the fastest way.

Furthermore, the system has been designed to be fully coordinated with the CORAL system, so that it integrates perfectly as part of the overall upgrade of the gripper system. This compatibility is particularly important in order to create an integrated system in which each component cooperates optimally with the other parts. The ALGAE system is, in fact, a kind of complement to the CORALS system, contributing to a more unified, controlled, and efficient operation. This design makes it possible to fully utilize the system's capabilities and improves the robot's performance throughout the process, so that the final result is both accurate and reliable.

Thanks to the full compatibility and precise operation, every component in the system, from collection to the final stages of shooting the ALGAE, is performed smoothly and balanced, which ensures a high level of cooperative work with the other parts of the system and with the CORALS system.

The connection of the gearbox to the subsystem responsible for algae is made using an aluminum axis connected to it with cellular polyurethane pulleys. The gearbox connected to the ALGAE subsystem includes a 9/1 reduction created by a KRAKEN-X60 motor that rotates an 18T gear which rotates a 72T gear which rotates an axis with a pulley that ultimately rotates the complementary wheels that create efficient and reliable algae

collection and release. On the other side of the 3-inch system with a hardness rating of 50A, they are responsible for collecting and releasing the algae.

the printed parts to the polycarbonate plates. These brackets provide additional support to the structure and allow for greater flexibility in the mobility of the various components.

The wheels, an essential component in the gripper's function, have been specifically designed for optimal coral collection and release. 2-inch diameter wheels with a hardness rating of 30A ensure good grip and smooth operation, enabling the system to collect and release coral efficiently and quickly.

The CORAL system, which constitutes the collection and release mechanism, includes a custom-made gearbox with a gear ratio of 1:3.3. This gearbox, based on a KRAKEN-X60 motor and precise gears, creates a reverse motion that is transmitted to the complementary wheels via pulleys. This combination ensures reliable and rapid collection and release of coral.

The choice of KRAKEN-X60 motors was made with the understanding that this year's tasks, collecting and placing coral on the reef, require high power and speed. These motors provide the necessary performance and ensure that the gripper performs its tasks efficiently and quickly.

All components of the gripper have been carefully selected, with an emphasis on strong connections, durable axes, and gears that are compatible with the powerful motors. This combination creates a high-quality and stable gripper system that provides the accuracy and performance required for success in the task.

The gripper, which forms the very core of the coral collection system, has been meticulously designed to perform complex tasks on the playing field. The system is composed of several components that operate in harmony, each making a crucial contribution to the overall function.

At the base of the gripper, two 5 mm thick polycarbonate plates serve as a central platform. These plates, connected to the wheels, ensure smooth and precise movement across the surface, allowing the system to move efficiently and accurately.

To ensure maximum stability, the gripper is equipped with L-brackets that connect the printed parts to the polycarbonate plates. These brackets provide additional support to the structure and allow for greater flexibility in the mobility of the various components.

The wheels, an essential component in the gripper's function, have been specifically designed for optimal coral collection and release. 2-inch diameter wheels with a hardness rating of 30A ensure good grip and smooth operation, enabling the system to collect and release coral efficiently and quickly.

The CORAL system, which constitutes the collection and release mechanism, includes a custom-made gearbox with a gear ratio of 1:3.3. This gearbox, based on a KRAKEN-X60 motor and precise gears, creates a reverse motion that is transmitted to the complementary wheels via pulleys. This combination ensures reliable and rapid collection and release of coral.

The choice of KRAKEN-X60 motors was made with the understanding that this year's tasks, collecting and placing coral on the reef, require high power and speed. These motors provide the necessary performance and ensure that the gripper performs its tasks efficiently and quickly.

All components of the gripper have been carefully selected, with an emphasis on strong connections, durable axes, and gears that are compatible with the powerful motors. This combination creates a high-quality and stable gripper system that provides the accuracy and performance required for success in the task.

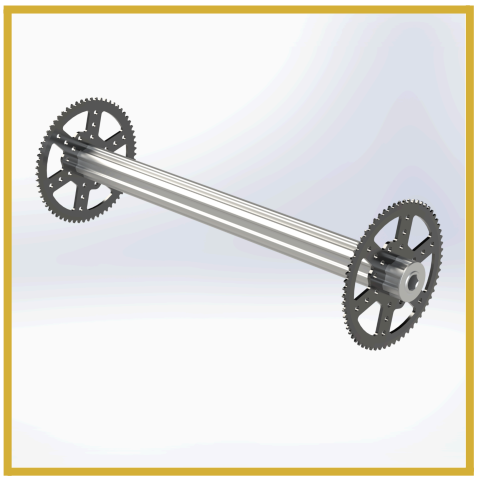
Using the MAX Spline Shaft

As part of the planning and development of this year's robotic gripper, we made a strategic decision to integrate the MAX Spline Shaft as the central rotation axis. This choice stemmed from our deep understanding of the shaft's unique advantages, particularly regarding stability and durability.

The MAX Spline Shaft, known for its meticulous engineering structure, provides a robust platform for the gripper's rotation. Its mechanical properties ensure that the shaft remains stable even under varying loads, preventing distortions and bending that could impair performance. This advantage gives the robotic gripper high movement accuracy and long-term reliability.

Another significant advantage of the shaft is its ability to serve as an integral platform for assembling the gripper's components. All parts of the gripper connect directly to the shaft, which simplifies the assembly process, improves structural stability, and reduces the risk of malfunctions. An additional advantage of the MAX Spline Shaft lies in its unique shape, which allows each mounted part to benefit from a greater number of gripping points. This feature contributes to a higher structural stability of the gripper.

To ensure a strong and efficient connection, the MAX Spline Shaft is designed to allow the integration of HEX shafts on both its sides. HEX shafts, known for their strength and reliability, provide a stable and robust connection, enabling the transfer of high torque without fear of slippage or wear.



The MAX Spline Shaft is characterized by multiple gripping points between the clamp and the shaft, which ensures stability and durability. These gripping points create a strong connection, reduce displacement, and allow for uniform load distribution, thereby preventing damage and contributing to reliable performance..

Drawing Conclusions Between Competitions - Gripper Replacement

After the ISR District 1 competition, we identified a significant failure in our gripper system: the inability to complete the "Reach L4" task, which was defined as a critical "Must" requirement. Since placing the CORAL in L4 was one of the basic requirements of the system, we decided to make a fundamental change to the gripper for the upcoming tasks, aiming to optimize its performance.

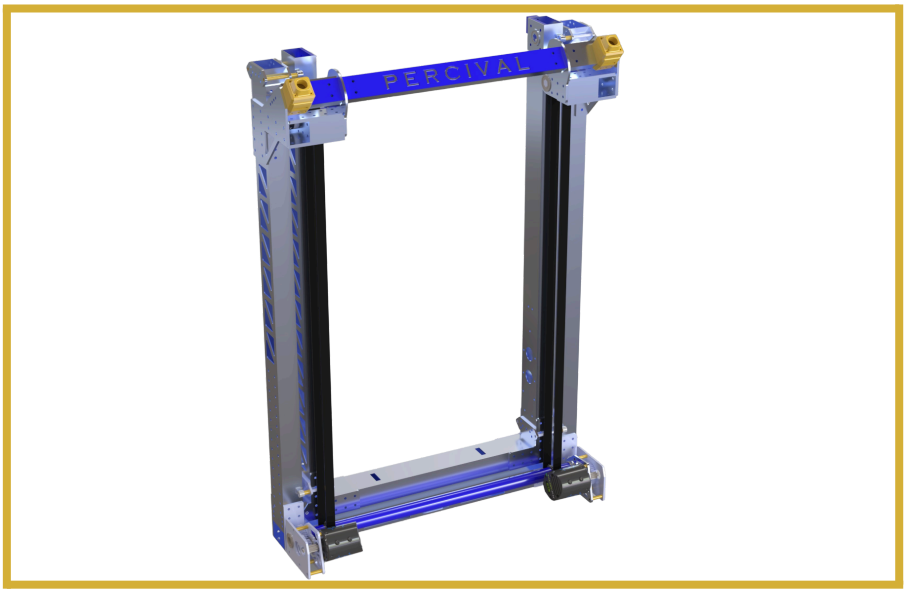
The previous gripper was based on a structure of two parallel polycarbonate plates, between which 3-inch wheels were installed. These wheels were intended to collect and transfer the CORAL smoothly but did not provide the required ability to reach L4. In addition, 4-inch wheels were incorporated into the gripper, the purpose of which was to center the CORAL and insert it straight and stably into the system. While these wheels helped to some extent in the process, they did not ensure sufficient accuracy in reaching L4, which impaired the system's efficiency in tasks requiring precise CORAL placement.

We understood that achieving the required accuracy and performing the tasks properly necessitates a significant improvement in the gripper system. We aim to upgrade the system so that it not only performs the existing tasks more efficiently but also meets the future requirements of competitions, while maintaining stable and high performance.

Picture of Old Griper

Significant improvement in the gripper system. We aim to upgrade the system so that it not only performs the existing tasks more efficiently but also meets the future requirements of competitions, while maintaining stable and high performance.

The Elevator



At the heart of every successful lifting system stands a strong and durable steel axis. In our case, this axis serves as a central component in the elevator's lifting mechanism, forming the backbone of the entire system. This axis is connected to two KRAKEN motors with a gear ratio of 1:8.5, enabling high performance and precise control over the axis's movement.

The motors are connected to two custom-designed and manufactured gearboxes, created using the team's machine. These gearboxes provide the necessary power to lift the elevator smoothly and stably. Additionally, the system includes chains connecting the gearboxes to the second stage, while the first stage provides the required support and enables the lifting operation.

The steel axis is characterized by high rotational capacity and exceptional strength, while maintaining stability throughout the process. It is located at the bottom of the elevator and allows for free movement of the system, while maintaining maximum precision in the drive. In fact, this axis serves as the spine of the elevator system, and the motors' ability to drive it accurately and balanced ensures optimal operation of the entire system.

The first stage of the elevator system, made of 60x40x2 mm metal profiles, is connected to the chassis profiles using plates and provides stability to the elevator stages. The stage is also connected to 6 mm thick plates, which are attached to the chassis profiles. The second stage, also made of 60x40x2 metal profiles, is connected to the steel axis via a chain. The axis, together with the motors and gearboxes, provides power and precise

movement, ensuring stable and reliable operation of the system, while maintaining complete synchronization between all its components.

All these features create a highly reliable system that allows for smooth and precise control of the elevator's height, while maintaining stability, durability, and optimal performance throughout the competition.

Integration in the Robot

During the work on the robot, we emphasized maintaining perfect integration between the various systems, while preventing any collision or interference between all the robot's components. Through in-depth and creative thinking invested right from the beginning of the season, we successfully designed a system where every part of the robot's structure operates optimally without hindering the operation of other parts. The goal was to ensure that we could build an efficient, strong, and stable robot, while maintaining a high level of integration between all the parts.

This year's robot chassis was designed with dimensions of 70*70 cm, which provides us with the perfect balance between a wide and organized workspace and high maneuverability on the field. In this system, the gripper – which is closed at the beginning of each match – opens immediately at the start of the match to prevent any possibility of collision with the elevator during the critical moments of initial operation. This immediate opening allows us to maintain smooth and stable movement of all systems without encountering obstacles.

During the match, when the gripper moves on its axis, complete control over its movement is maintained, and the system includes advanced software and mechanical controls that prevent any chance of collision with the elevator. The team's programming and engineering departments invested heavily in the design and development of algorithms and mechanical solutions so that every movement is performed in a synchronized and precise manner, thereby successfully ensuring that the system operates as smoothly as possible.

The great success of this system also stems from the elevator's central location within the robot. The decision to place the elevator in the center of the robot arises from strategic thinking about the robot's center of mass. This placement allows for perfect balance of the robot and helps in achieving maximum stability when moving across the field. In doing so, the robot maintains high maneuverability and speed without risking tipping over or colliding with other robots during operation.

Manufacturing with a CNC Machine and 3D Printer

The investment in an advanced CNC machine and a state-of-the-art 3D printer has truly revolutionized our team's manufacturing processes. We are now able to produce more complex and precise components than ever before, while significantly shortening production times and saving costs.

CNC Machine:

The CNC machine allows us to process a wide range of materials, including aluminum, plastic, and other metals, with high accuracy and speed. We use it to manufacture structural components for the robot, such as frames, brackets, and connecting parts. The manufacturing process begins with a computer-aided design of the component using advanced CAD software. The design is then transferred to the CNC machine, which performs the machining automatically, according to the design instructions.

One of the most prominent advantages of the CNC machine is its ability to produce components with complex geometries that could not be manufactured using traditional methods. In addition, the machine allows us to make design changes easily and quickly and to produce prototypes and different variations of the components.

3D Printer:

The 3D printer enables us to produce customized components from polymeric materials, such as PLA, ABS, and PETG. We use it to manufacture functional components for the robot, such as gears, sensor mounts, and complex connecting parts. The manufacturing process begins with a computer-aided design of the component using advanced CAD software. The design is then transferred to the 3D printer, which prints the component layer by layer, according to the design instructions.

The 3D printer allows us to produce components with complex geometries and great design freedom. In addition, it allows us to manufacture components customized for specific needs and to optimize weight and performance.

Integration and Advantages:

The combination of the CNC machine and the 3D printer allows us to produce hybrid components that combine the advantages of both technologies. For example, we can

manufacture a structural component from aluminum using the CNC machine and add 3D-printed functional components to it.

The use of these advanced technologies enables us to:

- **Improve robot performance:** By manufacturing more precise, lighter, and stronger components.
- **Shorten production times:** By automating manufacturing processes.
- **Save costs:** By manufacturing components in our internal workshop.
- **Increase design flexibility:** By having the ability to manufacture customized components.
- **Improve system integration:** By manufacturing precise components that fit the robot perfectly.

Ultimately, the investment in these technologies allows us to build a more advanced and innovative robot capable of meeting the most complex challenges.

Part III:

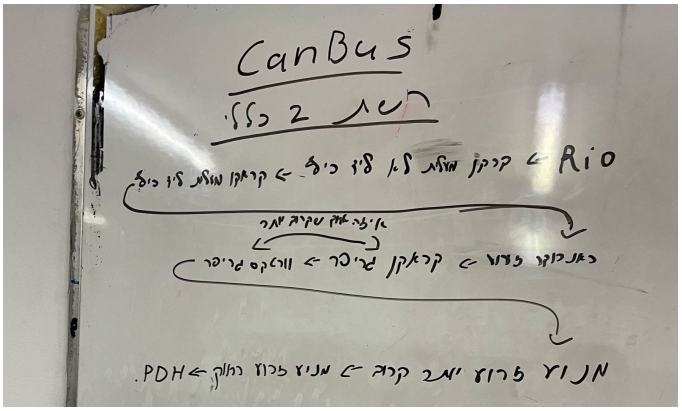
Electronics

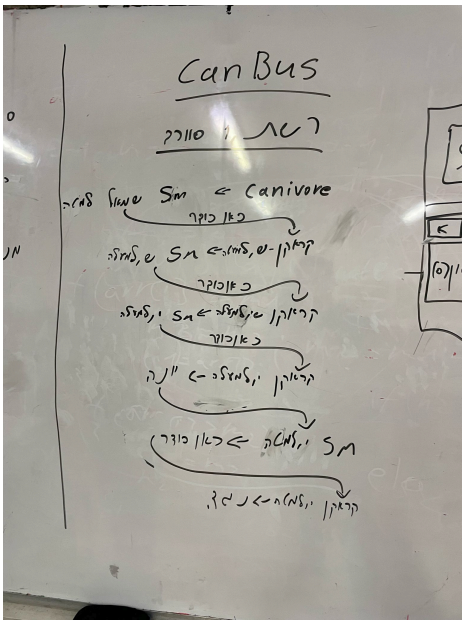
Working Process

The work process of the Electronics team within the group is divided into a series of central meetings: an integration meeting and a control meeting. Afterward, the team moves on to the wiring phase, where the plans from the previous meetings are implemented.

Itegration meeting

The meeting is intended to form a general overview of the electronic aspects of the various systems in the robot, in preparation for the next stages of design. During the meeting, we consider, among other things, the ideal placement of sensors, cables, and control components (such as motor controllers, gyro sensors, etc.).
 This is a team meeting attended by electronics team members, where each participant suggests an idea, which is critically discussed in order to identify potential flaws. Additionally, we consider the best way to route our canbus, as shown in the following diagrams.





Control meeting

This is a general and joint meeting of the electronics and software team members, aimed at fully planning the integration of control components into the robot's systems. The meeting is based on the insights and conclusions from the integration meeting, which

help us decide on the placement, connection, and types of sensors to be integrated into the robot's systems.

Choosing motors

Swerve Drive Motor: We chose the Kraken because it is a more efficient motor than the Vortex, as based on our experience from previous years, the Vortex does not perform well as the drive motor for the swerve.

Swerve Rotation Motor: For the rotation motor, we chose to use the NEO motor because it is reliable, and we felt that using a Kraken would be wasteful since it performs just as well as the NEO. We preferred to use our Krakens in other places.

Gearboxes: We decided to use the Kraken because it is the most powerful motor, and the Vortex does not handle heavy loads well (for the same reason, we also used these motors in the gripper).

Electronics the robot:

General wiring

During the wiring process, we found creative ways to route the cables to the appropriate components in a manner that would keep everything organized and easy to maintain. Each cable that enters the ROBORIO has a label indicating which encoder the cable is connected to. All the cables were routed through profiles or cable organizers to protect them from cuts and scratches. We made sure that all cables entering the components were organized, accessible, and easy to reach. Additionally, we decided to place all the electronic components in visible and accessible locations so that we could monitor their status, and we ensured that no cable was left loose that could disconnect.

Protecting the electrical components:

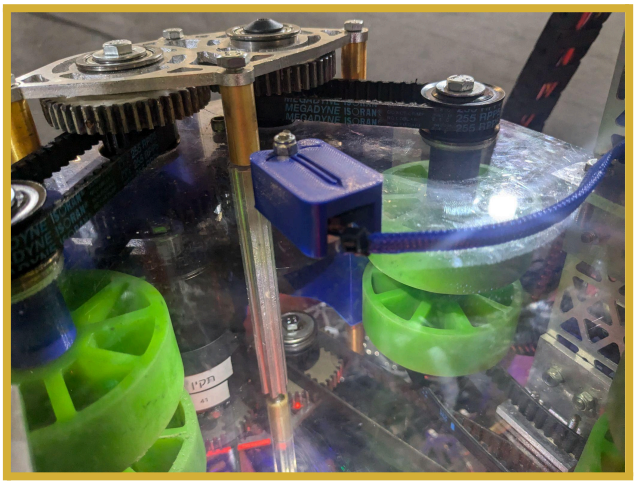
This year, we placed a strong emphasis on protecting the components in the robot from dust, drilling, and other mechanical issues. To protect the components, we put shields on all the sensitive ports in the robot and created 3D prints for the sensitive components (such as the orange pi).

Swerve Sensors

We connected an absolute encoder model CANCoder to the swerve modules so we could set and reset the direction of each swerve module. After testing the results against the price, we decided to place the Kraken motor on the drive because it is the fastest and most reliable, and the NEO motor at the angle since we have a lot of experience with it and it is known as a reliable motor.

Gripper Sensors

In the collection system, we placed a limit switch at the bottom of the gripper, whose role is to detect when the CORAL is inside the gripper and activate the software accordingly. We chose the limit switch because it is convenient for wiring and reliable, especially after dealing with many wiring issues last season when using the Beam Break sensor, which was difficult for the electronics team. Additionally, the limit switch is a sensor that is connected on one side, which makes it easier for us to perform aesthetic wiring. The sensor is activated when the coral enters the system and presses the lever, allowing the robot to detect whether it has a coral in the gripper.

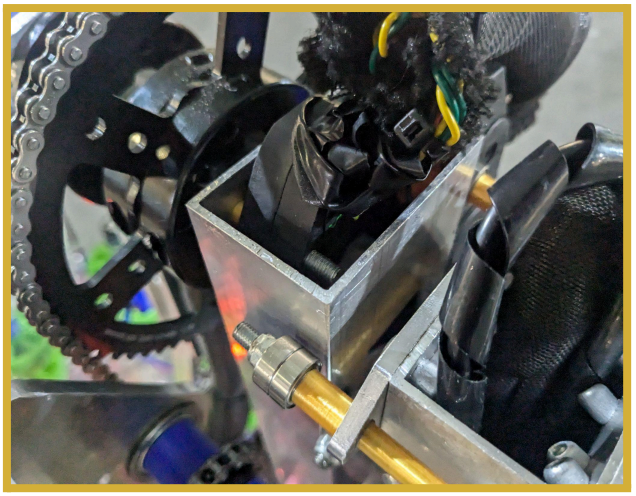


Elevator Sensors

At the top of the second stage of the elevator, we placed an angle sensor (encoder) whose role is to detect and measure the precise angle at which the gripper is positioned at any given moment. This sensor is a critical component of the system, as the angular position of the gripper directly impacts its ability to perform gripping and releasing actions accurately and efficiently.

The use of the angle sensor allows us to receive continuous real-time data on the gripper's position, which enables better control and significant improvement in movement precision. This data helps the robot's kinematic system adjust the gripper's angle in accordance with the changing angles of the REEF branches.

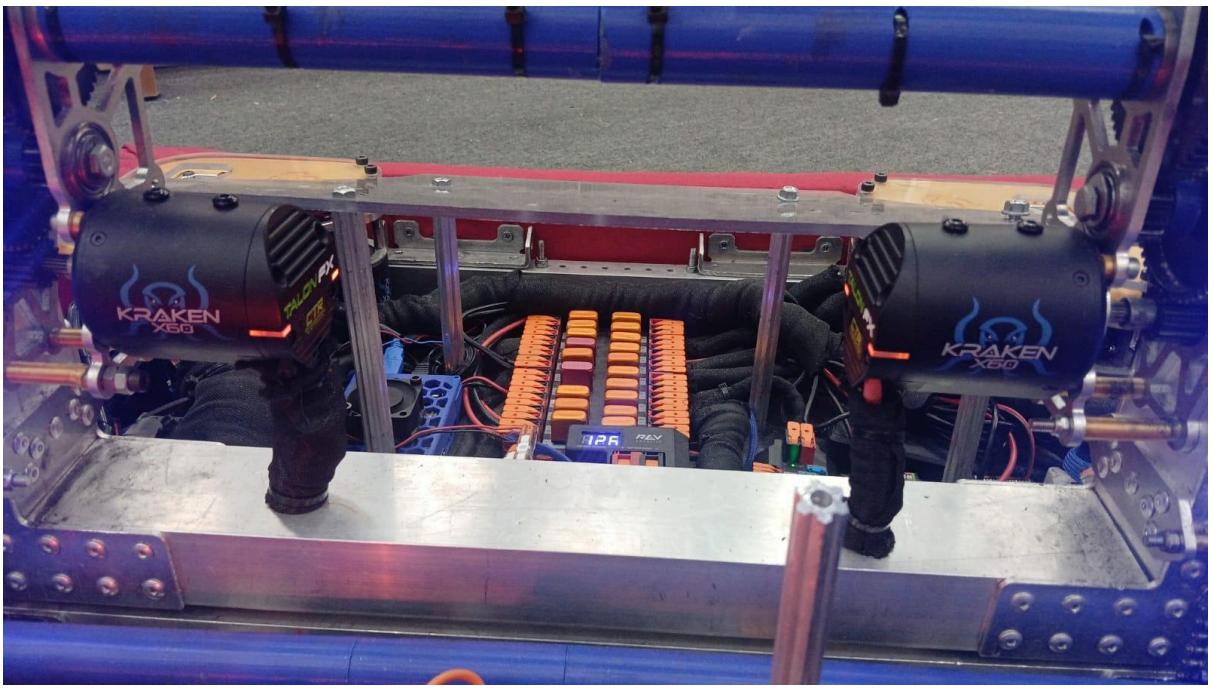
Another advantage of the angle sensor is its ability to make automatic adjustments and prevent errors caused by slipping or mechanical inaccuracies, which contributes to the overall improvement of the robot's performance. In combination with other sensors and motion control systems, we can ensure the gripper is optimally positioned, improving the efficiency of collecting and releasing corals during the game.By integrating all these sensors, we created full and seamless collaboration with the robot's software, allowing for the most efficient use of every moment of the match.



Wiring cables through profiles

To maintain maximum aesthetics in the robot’s wiring, we decided to route all the elevator cables through the profiles, so they are attached to the motor and 'disappear and reappear' at the PDH, as shown in the picture.

We routed the cables using several methods. Initially, we tried simply pushing the cables until they fit, but we quickly realized this wasn’t working. So, we took a metal wire and routed it through the profiles until it reached the top of the elevator, where the motors are. We attached the cables to the wire and then pulled it, and the cables moved up. We gathered all the cables at the top and then, with a hook, we routed them from the top down to the PDH.



Component list

- RoboRIO - Connects the computer and software to the robot and sends commands via the CAN bus.
- PDH (Power Distribution Hub) - Receives power from the battery and supplies it to all components on the robot.
- Orange Pi - Responsible for image processing and camera management.
- Battery - Supplies power to the PDH and, consequently, to the entire robot.
- VRM (Voltage Regulator Module) - Supplies power to small components.
- Spark Max - Provides power and transmits data to the Neo motor via the CAN bus.
- MPM (Mini Power Module) - Supplies power to small components.
- CANivore - Gives the option to add another CAN bus network, reducing the load on the main CAN network.
- CANCoder - An absolute encoder that provides the angle of the desired component and retains its data even when the robot is powered off and rotated.

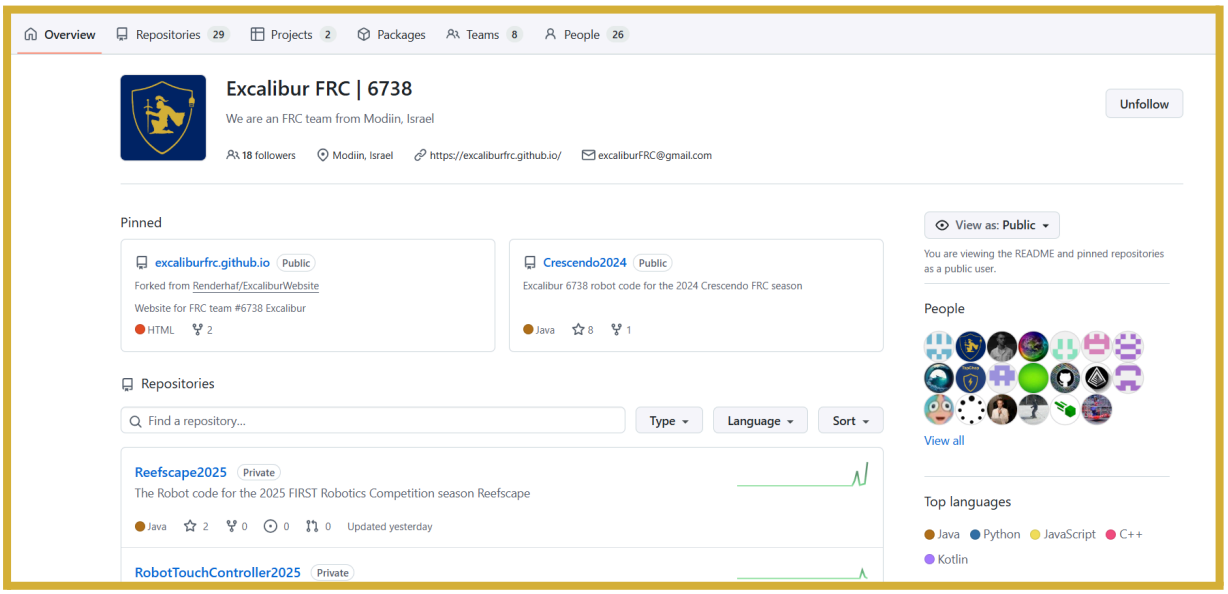
Part IV

Software

Programs and Tools

We develop our robot using the Java programming language and the WPIlib library. Additionally, we integrate various third-party libraries created by the community and leading companies, such as REVLib, PathPlanner, and Phoenix 6, to maximize the efficiency and performance of our code. This season, for the first time, we are also utilizing *ExcaLIB*, a custom library we developed, which will be discussed in greater detail later.

Furthermore, we primarily work with JetBrains' IntelliJ development environment and use Git and GitHub for version control and collaboration.



Team Operations

Our software team consists of approximately 12 members and is divided into two sub-teams: Control and Automation.

Each sub-team is responsible for managing the robot at a different level of abstraction. The Control team programs the individual systems and develops a physical model to represent them, while the Automation team utilizes these systems to enhance the robot's speed, precision, and autonomy as much as possible.

ExcaLIB

ExcaLIB is a library we developed this season to streamline programming and eliminate redundancy in our robot's code. It was created with the understanding that most of the robot's systems can be abstracted into four common mechanisms frequently used in FRC. Each of these mechanisms supports both manual control and advanced control strategies, such as PID and Motion Profiling.

One of the key advantages of working with mechanisms is encapsulation—they inherently manage all physical models, conversions, and calculations. In our library, each mechanism is represented by a dedicated class; for example, an arm mechanism is implemented as *arm.java*. What makes these classes innovative is that they are entirely abstract, allowing for any combination of mechanisms while ensuring unified physical calculations. For instance, three arm mechanisms can be combined to form a three-jointed robotic arm.

All mechanism classes inherit from the base class *Mechanism.java*, which provides fundamental system control by sending the desired output to the system's motors

Base Class - ***Mechanism.java***
The *Mechanism.java* class contains fundamental methods, such as this method for setting the output of the motors and a method for configuring all motors in the mechanism to Coast mode.

```java
public Command manualCommand(DoubleSupplier outputSupplier,
SubsystemBase... requirements) {
        return Commands.runEnd(
                () -> setOutput(outputSupplier.getAsDouble()),
                () -> setOutput(0),
                requirements
        );
}
```

All Mechanism classes inherit (extends) from the Base class - ***Mechanism***

Mechanism Representing an Angular Arm (*Arm.java*) - This mechanism represents an angular arm, enabling precise angle control using a **PID controller** and an advanced, customized **feedforward** control system. This year, it is used to control the robot's arm, ensuring it reaches the desired angle as quickly and accurately as possible.

This method allows precise control of the mechanism's angle -

```java
public Command anglePositionControlCommand(
            DoubleSupplier setPointSupplier,
            Consumer<Boolean> toleranceConsumer,
            double maxOffSet,
            SubsystemBase... requirements) {
      final double dutyCycle = 0.02;
      return new RunCommand(() -> {
            double error = setPointSupplier.getAsDouble() -
                        ANGLE_SUPPLIER.getAsDouble();
            double velocitySetpoint = error / dutyCycle;
            velocitySetpoint =
                        m_VELOCITY_LIMIT.limit(velocitySetpoint);
            double phyOutput =
                     m_ks * Math.signum(velocitySetpoint) +
                      m_kg * m_mass.getCenterOfMass().getX();
            double pid = m_PIDController.calculate(
                  ANGLE_SUPPLIER.getAsDouble(),
                  setPointSupplier.getAsDouble()
            );
            double output = phyOutput + pid;
            super.setVoltage(m_VELOCITY_LIMIT.limit(output));
            toleranceConsumer.accept(
                  Math.abs(error) < maxOffSet
            );
      }, requirements);
}
```

This method integrates advanced control techniques such as **Soft Limits** and a **physical model**.

Mechanism Representing a Rotating Wheel - (*FlyWheel.java*) - This mechanism controls the speed of a rotating wheel. It is used for the swerve drive wheels and can also represent wheels in a shooting system.
This method allows for controlled regulation of the mechanism's speed, ensuring precise and efficient operation -

```java
public Command smartVelocityCommand(DoubleSupplier
velocitySupplier, SubsystemBase... requirements) {
      return new RunCommand(() -> {
            TrapezoidProfile profile = new TrapezoidProfile(
                  new Constraints(maxAcceleration, maxJerk)
            );
            TrapezoidProfile.State state =
                  profile.calculate(
                        0.02,
                        new TrapezoidProfile.State(
                              super.m_motor.getMotorVelocity(),
                              getAcceleration()
                        ),
                        new TrapezoidProfile.State(
                              velocitySupplier.getAsDouble(),
                              0
                        )
                  );
            double ff =
                  m_gains.ks * Math.signum(state.position) +
                  m_gains.kv * state.position +
                  m_gains.ka * state.velocity;
            double pid = m_pidController.calculate(
                  super.m_motor.getMotorVelocity(),
                  state.position
            );
            setVoltage(pid + ff);
      }, this);
}
```

Mechanism for Linear Motion Control - (*LinearExtention.java*) - This mechanism controls linear motion in systems such as lifts, telescopic arms, or linear actuators, using a Trapezoid Profile combined with advanced PID and Feedforward control. This year, it is used to control the elevator subsystem in our robot.

This method enables precise control over the mechanism's length, optimizing movement using the trapezoidal profile along with advanced control techniques -

```java
public Command extendCommand(DoubleSupplier lengthSetPoint,
SubsystemBase... requirements) {
      return new RunCommand(() -> {
            TrapezoidProfile profile =
                  new TrapezoidProfile(m_constraints);
            TrapezoidProfile.State state =
                  profile.calculate(
                  0.02,
                  new TrapezoidProfile.State(
                        m_positionSupplier.getAsDouble(),
                        super.m_motor.getMotorVelocity()
                  ),
                  new TrapezoidProfile.State(
                        lengthSetPoint.getAsDouble(),
                        0
                  ));
            double pidValue = m_PIDController.calculate(
                        m_positionSupplier.getAsDouble(),
                        state.position);
            double ff =
                  (Math.abs(m_positionSupplier.getAsDouble() -
                  lengthSetPoint.getAsDouble()) > m_tolerance) ?
                  m_gains.ks * Math.signum(state.velocity) +
                  m_gains.kv * state.velocity +
            m_gains.kg *
                        Math.sin(m_angleSupplier.getAsDouble()) :
                  m_gains.kg *
                        Math.sin(m_angleSupplier.getAsDouble());
            double output = ff + pidValue;
            setVoltage(output);
      }, requirements);
}
```

Mechanism Representing a Turret System - (Turret.java) - This mechanism controls the angle of a turret, allowing guided angle adjustments. It is used for controlling the angle of the swerve drive wheels and can also represent a turret system.

```java
public void setPosition(Rotation2d wantedPosition) {
      double smartSetPoint =
            m_rotationLimit.getSetPoint(
                  getPosition().getRadians(),
                  wantedPosition.getRadians()
            );
      double pid =
            m_anglePIDcontroller.calculate(
                  m_POSITION_SUPPLIER.getAsDouble(),
                  smartSetPoint
            );
      double ff =
            m_angleFFcontroller.getKs() * Math.signum(pid);
      super.setVoltage(pid + ff);
}
```

In addition to the mechanisms in our library, there are several other useful tools and utilities in the ExcaLIB library:

Dynamic Soft Limits

Dynamic Soft Limits are generic software limits that can restrict the speed or position of a specific mechanism (or any element in the code). Our soft limits are dynamic, meaning the restrictions on the systems can change based on factors like position, speed, or the system's physical limitations.

For example, in this year's robot, we have an arm on an elevator, and above the arm, there's an aluminum profile. The soft limit restricts the arm's angle and the elevator's position based on each other's location. If the elevator is extended, the arm cannot go beyond the profile, as it would cause the elevator to close and the arm to collide with the profile.

The soft limits are divided into two different classes:

SoftLimit.java

A base class for dynamic soft limits.

This method is used to apply a limit to a specific value, ensuring it remains within the defined boundaries set by the dynamic soft limits -

```java
public double limit(double val) {
      if (within(val)) {
            return val;
      }
      if (val > m_maxLimit.getAsDouble()) {
            return m_maxLimit.getAsDouble();
      }
      return m_minLimit.getAsDouble();
}
```

ContinuousSoftLimit.java

A class that enables limits for systems where the restriction is continuous (e.g., limiting an angle or angular velocity).

```java
public double getSetPoint(double measurement,
      double wantedSetPoint) {
      double upperSetPoint, lowerSetPoint;
      if (wantedSetPoint > measurement) {
            upperSetPoint = wantedSetPoint;
            while((upperSetPoint - 2 * Math.PI) > measurement) {
                  upperSetPoint -= 2 * Math.PI;
            }
            lowerSetPoint = upperSetPoint - 2 * Math.PI;
      } else if (wantedSetPoint < measurement) {
            lowerSetPoint = wantedSetPoint;
            while((lowerSetPoint + 2 * Math.PI) < measurement) {
                  lowerSetPoint += 2 * Math.PI;
            }
            upperSetPoint = lowerSetPoint + 2 * Math.PI;
      } else {
            return wantedSetPoint;
      }
      if (upperSetPoint > super.getMaxLimit()) {
            return lowerSetPoint;
      } else if (lowerSetPoint < super.getMinLimit()) {
            return upperSetPoint;
      }
      return Math.abs(measurement - upperSetPoint) <
            Math.abs(measurement - lowerSetPoint) ?
            upperSetPoint : lowerSetPoint;
}
```

Vector2D.java

A class representing a mathematical vector in space. We use this class to represent linear motion/velocity along an axis, which allows us to perform various physical calculations.

Mass.java

Handles external forces and the distribution of mass between mechanisms. For example, in a two-jointed arm, the position and mass of the second segment directly affect the forces acting on the first segment.

Motor.java

To enable the integration of masses, vectors, and mechanisms, we created a wrapper class for the motor types we use. The motors we instantiate as objects in the code are accessed through the class representing the motor (for each motor type we use). All the classes that represent different motor types implement our Motor.java interface, ensuring consistency and generality in our code.

When we create mechanism objects, we don't simply create motors for activation. Instead, we pass the motors as parameters to the mechanism, so the mechanism can control the motors according to the specific physical models tailored to it.

Logs

Sometimes things go wrong, and to identify what went wrong and fix it later, we use a tool called **Monologue** for logging. The logs are saved on a USB drive that is directly connected to the RoboRio.

Through these logs, the robot records everything that happens during the workshop and each match, from the data received from all sensors to the robot's position on the field or the battery voltage.

To read the logs, we use software developed by the **Mechanical Advantage #6328** team, which allows us to read and analyze the logs in a very convenient, intuitive, and efficient way. It also enables displaying all the information on graphs and in a 3D view.

This ensures that any problem that occurs on the robot will not happen again.

Drive System - Swerve

This year, we decided to develop code for the swerve system ourselves, without relying on external libraries. The code consists of a class representing a single swerve module (SwerveModule.java). This class includes various methods that enable intelligent control of the module.

For example, this method allows setting the speed and direction of the module intelligently, based on different operations from the mechanisms in EXCALib (the FlyWheel class represents the rotation of the swerve wheel, and the Turret class represents the rotation of the module) and using different classes such as Vector2D.

Java

```java
public Command setVelocityCommand(
Supplier<Vector2D> moduleVelocity) {
      return new ParallelCommandGroup(
            m_driveWheel.setDynamicVelocityCommand(() -> {
                  Vector2D velocity = moduleVelocity.get();
                  double speed = velocity.getDistance();
                  if (speed < 0.1) {speed = 0;}
                  boolean optimize = isOptimizable(velocity);
                  return optimize ? -speed : speed;
            }),
            m_turret.setPositionCommand(() -> {
                  Vector2D velocity = moduleVelocity.get();
                  double speed = velocity.getDistance();
                  if (speed < 0.1)
                        return m_turret.getPosition();
                  boolean optimize = isOptimizable(velocity);
                  Rotation2d direction =velocity.getDirection();
                  return optimize?
                        direction.rotateBy(
                              Rotation2d.fromRadians(Math.PI)
                        ) : direction;
            }),
```

```
            new RunCommand(() -> {
                  m_setPoint.setY(moduleVelocity.get().getY());
                  m_setPoint.setX(moduleVelocity.get().getX());
            })
      );
}
```

For integrating the four modules, we used an additional class called **ModulesHolder**, whose purpose is to perform simple operations on all four modules.

The class representing the entire system is **Swerve.java,** which contains various methods for controlling the system, such as:

A method for driving the swerve system based on values passed to it:

- A **Vector2D** representing the desired linear velocity
- A **double** value representing the desired angular velocity
- A **boolean** value that allows switching between robot-oriented driving (based on the robot's coordinate system) and field-oriented driving (based on the field's coordinate system).

Java

```
public Command driveCommand(
      Supplier<Vector2D> velocityMPS,
      DoubleSupplier omegaRadPerSec,
      BooleanSupplier fieldOriented) {
      Supplier<Vector2D> adjustedVelocitySupplier = () -> {
            Vector2D velocity = velocityMPS.get();
            if (fieldOriented.getAsBoolean()) {
                  Rotation2d yaw = getRotation2D().unaryMinus();
                  if (!AllianceUtils.isBlueAlliance()) {
                        yaw = yaw.plus(pi);
```

```
                    }
                    return velocity.rotate(yaw);
               }
               return velocity;
          };
          Command driveCommand = m_MODULES.setVelocitiesCommand(
                    adjustedVelocitySupplier,
                    omegaRadPerSec
          );
          driveCommand.setName("Drive Command");
          driveCommand.addRequirements(this);
          return driveCommand;
     }
```

1. A method for the robot's autonomous navigation to a specific point on the field using **Profiled PID**.

```java
public Command pidToPose(Supplier<Pose2d> poseSetpoint) {
     return new SequentialCommandGroup(
          new InstantCommand(() -> {
               m_xController.calculate(getPose2D().getX(),
                    poseSetpoint.get().getX());
               m_yController.calculate(getPose2D().getY(),
                    poseSetpoint.get().getY());
               m_angleController.calculate(
                    getRotation2D().getRadians(),
               poseSetpoint.get().getRotation().getRadians()
               );
          }),
          driveCommand(() -> {
               Vector2D vel = new Vector2D(
               m_xController.calculate(getPose2D().getX(),
                    poseSetpoint.get().getX()),
               m_yController.calculate(getPose2D().getY(),
                    poseSetpoint.get().getY())
```

```
                );
                double distance =
                      getPose2D().getTranslation().getDistance)
                        poseSetpoint.get().getTranslation());
                vel.setMagnitude(Math.min(vel.getDistance(),
                           velocityLimit.get(distance)));
                if (!AllianceUtils.isBlueAlliance())
                      return vel.rotate(pi);
                return vel;
          },
          () -> m_angleController.calculate(
                      getRotation2D().getRadians(),
                poseSetpoint.get().getRotation().getRadians()
          ),
          () -> true
    )).until(finishTrigger).withName("PID To Pose");
}
```

2. Method to turn the robot to a certain angle.

```
Java
public Command turnToAngleCommand(
Supplier<Vector2D> velocityMPS,
Supplier<Rotation2d> angleSetpoint) {
      return new SequentialCommandGroup(
            new InstantCommand(
                  () -> m_angleSetpoint = angleSetpoint
            ),
            driveCommand(
                  velocityMPS,
                  () -> m_angleController.calculate(
                              getRotation2D().getRadians(),
                              angleSetpoint.get().getRadians()),
                  () -> true
            )
      ).withName("Turn To Angle");
}
```

Autonomous and Automations

This year, we aimed for a very high level of automation – as long as no issues occur, the driver should touch the controller as little as possible, and the robot should be nearly fully autonomous.
We achieved this by integrating the following tools:

Odometry

We use an algorithm from the WPIlib library to perform basic tracking of the robot's position on the field using various sensors in the swerve system, such as the built-in encoders in the motors and the gyro sensor. This algorithm is called by the RoboRIO at a higher frequency compared to other parts of the code to ensure positioning accuracy on the field.

However, this algorithm is not perfect and accumulates drift over time because it relies on the movement of the swerve wheels. Sometimes, the robot collides with other robots or field elements, or the wheels slip, which increases the drift.

Photon Vision

Due to the accumulated drift of the odometry, we use advanced software based on April Tag recognition across the field. This software is used to correct the drift accumulated by odometry over time.
We use Photon Vision to accurately and quickly determine our position on the field. We run Photon Vision on an Orange Pi processor, and with the Orange Pi, we operate two cameras simultaneously, which allows us to achieve a wide field of view and more rapid and accurate localization.

Part V:

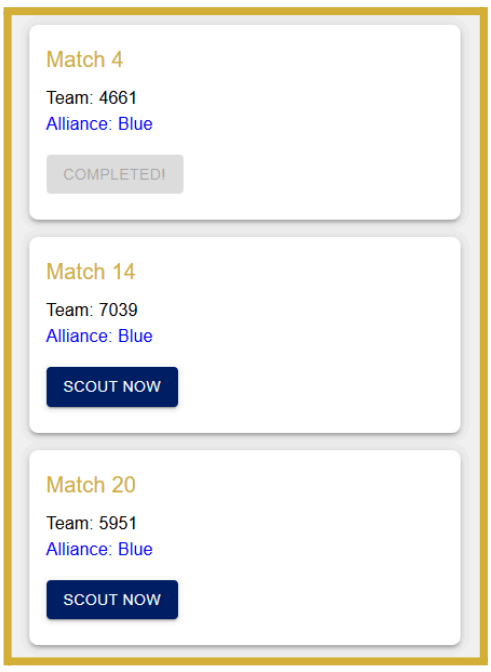
Strategy & Scouting

Our team has been active for the 9th year, and one of the things we've learned over the years is that a good strategy can change the competition. After all, we cannot control the list of matches, nor can we change the robot during the competition. The only thing we can control during the matches is the strategy for the match, and later the strategy for choosing alliances. To that end, we have built an advanced strategy system built from a scouting system and a data processing and analysis system.

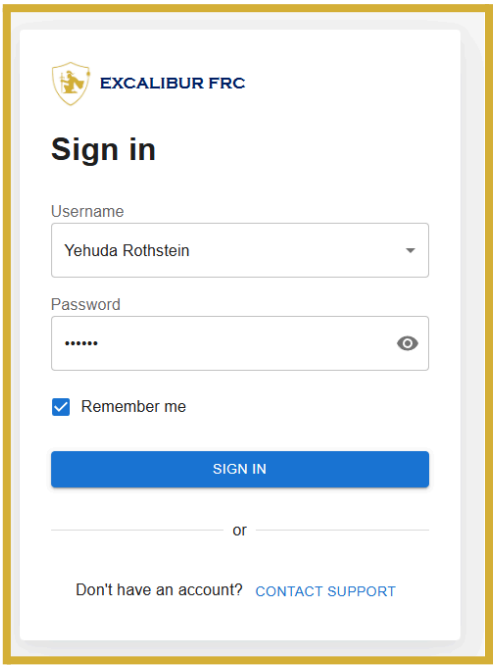
Scouting system

The first stage is scouting data collection. For this we have a large and professional scouting team that undergoes training before the competition the scouts fill out a form (from a scouting website we built) about each group each match the form contains the raw data that we found useful for strategy planning. This year we decided to build and renew our scouting system from scratch. The scouting website we built is built using the React interface and the Firebase database for information storage. We chose the React platform because of the wide internet support for the language and the Fire-base database because it interfaces conveniently with Google sheets where we built the strategy system.

Each of the scouts has a user account with which they can easily access the matches assigned to them.

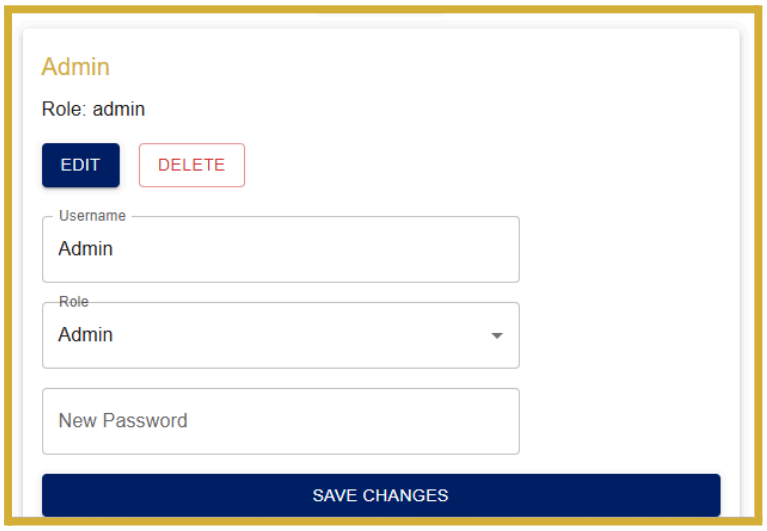


match selection:



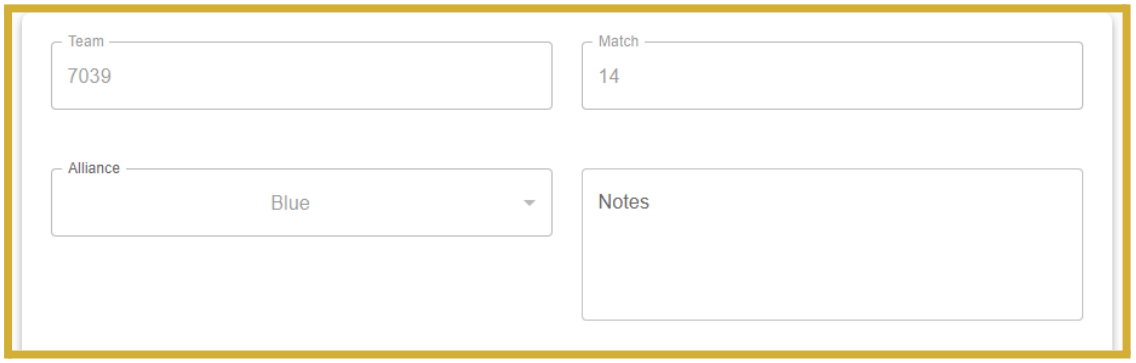
system enter:

To prevent malfunctions that may occur throughout the competition day changing and correcting the scouts' details is accessible to the scouting manager:



example:

Upon entering a match independent details are automatically entered in order to prevent small errors.



match details.

The allocation of matches to scouts is done automatically and is accessible for changes using the scouting manager's account.

match chart.

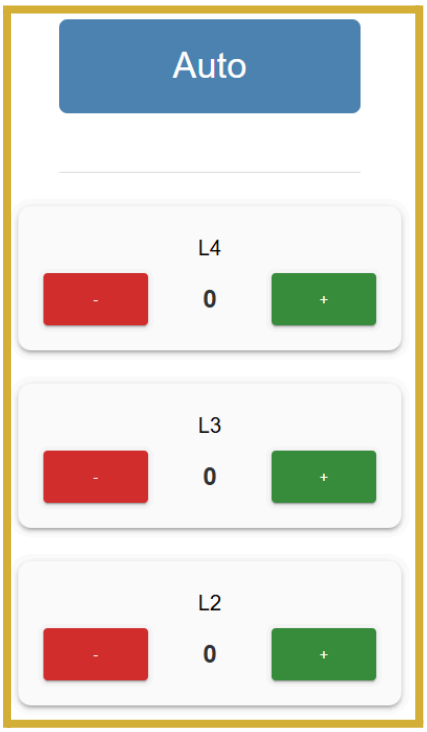
We divide scouting into 3 categories corresponding to the 3 stages of the game:

1. Autonomous
2. Teleop
3. EndGame

This is so that when we approach planning the game strategy we will work in an orderly and systematic manner that relates to each stage of the game and to each robot in the alliance.

Autonomous stage

In the autonomous phase the scout is presented with five counters one for each level on the Reef and another counter for launching algae balls.



Autonomous scouting:

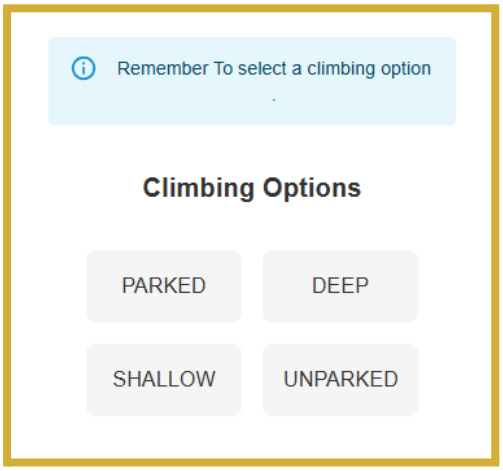
Teleop stage

In the Teleop phase similar to the Auto phase several counters are displayed:

- A counter for each level on the Reef
- A counter for shooting Algae into the Processor
- A counter for defensive Pins
- A counter for shooting Algae into the Net
- A counter for removing Algae

Endgamestage

בIn the Endgame phase the scout is presented with four buttons from which they choose one of the climbing options.



Scouting climbing.

Offline Work

Since there is not always internet at competitions upon completion of the form a barcode is displayed to the scout another scout who has an internet connection can scan the barcode and upon scanning the form will be sent to the database.

Barcode example.

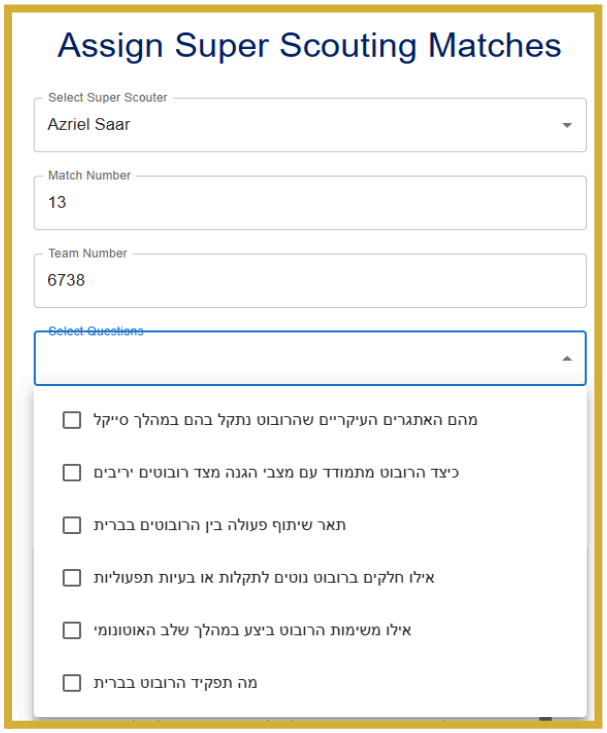
The Super Scouting

In order to build a suitable and winning strategy there is sometimes a need for more advanced data to collect this data specific people must be assigned who will deal only with the advanced data and will place less emphasis on raw data therefore in parallel to the scouting team a superscouting team works a team that is trained to observe a group during a match and try to better understand the relative advantages of each robot as well as the relative disadvantages of each robot.

Advanced Data:
As mentioned there is the raw data - how much a robot scores whether a robot climbs whether it passes under chains etc. but superscouting focuses on more advanced data data that is sometimes difficult to impossible to quantify with a number this is why the superscouts undergo specific training and advanced simulations.
Examples of advanced data:

- What are the main challenges the robot encounters during a cycle?
- How did the robot cope with defense (if any)?
- Describe the cooperation between the robots in the alliance.
- Which parts of the robot are prone to malfunctions or operational problems?
- What is the robot's role in the alliance?
- What tasks did the robot perform during the autonomous phase?

The head of superscouting who is a member of the strategy team chooses which questions to assign to each superscout before each match the questions are automatically assigned to the superscout.



Data Analysis System

This year we decided to refine our data analysis system after examining several options. We decided to continue building advanced systems using Google sheets. We integrated the use of the Google appScript platform to connect the system to API systems of databases such as TBA and Statbotics. In this section we will present the main functionality of the system.

Basic team analysis

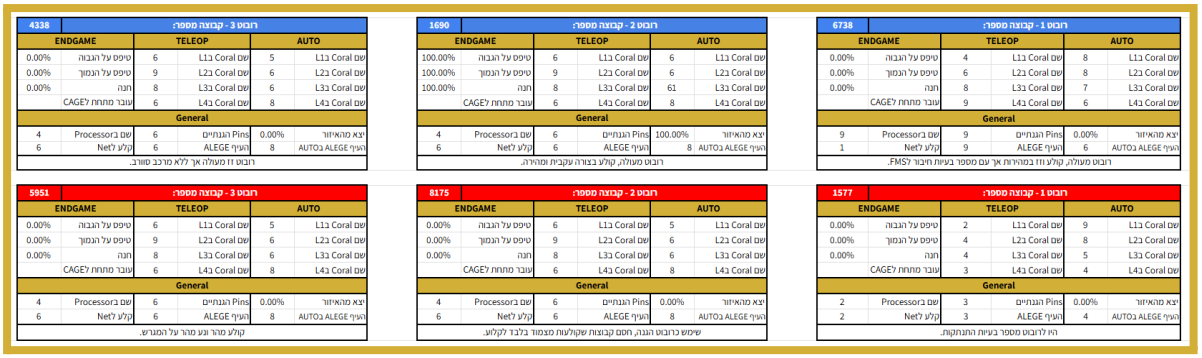
The group analysis consists of several slots where one of the slots is editable and in it the number of the group for which you want to receive the data is entered the data will change automatically when a group number is entered. In addition we connected a language model (ChatGPT 3.5) to the system which takes the superscouting data and summarizes it in one concise sentence

6738	רובוט 1 - קבוצה מספר:				
ENDGAME		TELEOP		AUTO	
0.00%	טיפס על הגבוה	4	שם Coral בL1	8	שם Coral בL1
0.00%	טיפס על הנמוך	6	שם Coral בL2	8	שם Coral בL2
0.00%	חנה	8	שם Coral בL3	7	שם Coral בL3
	עובר מתחת לCAGE	9	שם Coral בL4	6	שם Coral בL4
General					
9	שם בProcessor	9	Pins הגנתיים	0.00%	יצא מהאיזור
1	קלע לNet	9	העיף ALEGE	6	העיף ALEGE בAUTO
רובוט מעולה, קולע וזז במהירות אך עם מספר בעיות חיבור לFMS.					

Team analysis page

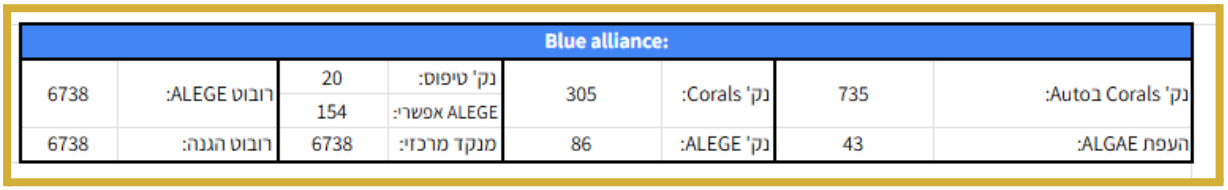
Strategy control panel

Before a match in one of the tabs in the system there are six slots where a basic analysis of a group is done and you can enter the numbers of all the groups competing with us in the match in order to get a reliable overview before the match.



strategy control panel.

In order to have additional data analysis beyond the existing data in the system there is an advanced data analysis system located within the dashboard. The system analyzes the data into several categories we have chosen.



Blue alliance:						
נק' Corals באוטו:	735	נק' Corals:	305	נק' טיפוס:	20	רובוט ALEGE:	6738
				ALEGE אפשרי:	154		
העפת ALGAE:	43	נק' ALEGE:	86	מנקד מרכזי:	6738	רובוט הגנה:	6738

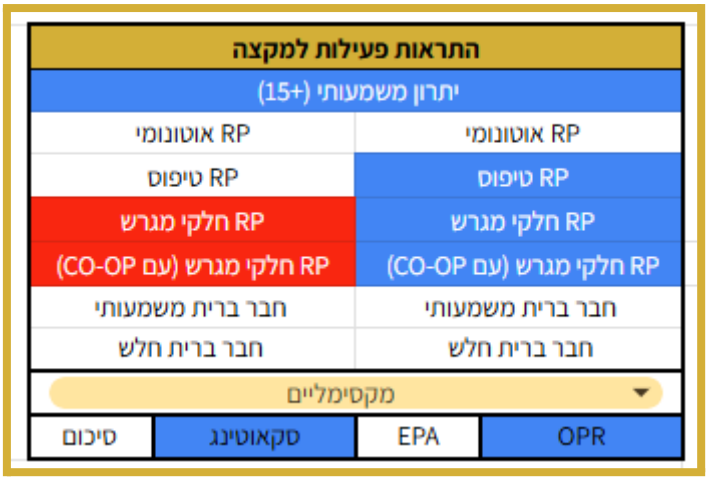
advanced data analysis in the system.

בנוסף, כחלק מזה המערכת מוציאה דו"ח ניקוד משוער בהתבסס על הטפסי סקאוטינג:

נקודות לרובוט:		
כחול 1:	6738	261
כחול 2:	1690	585
כחול 3:	4338	240
ברית:	סה"כ:	1086

Advanced data analysis, predicting qual results.

In addition as part of this the system generates an estimated scoring report based on the scouting forms:



estimated scoring.

Selecting the desired type of data

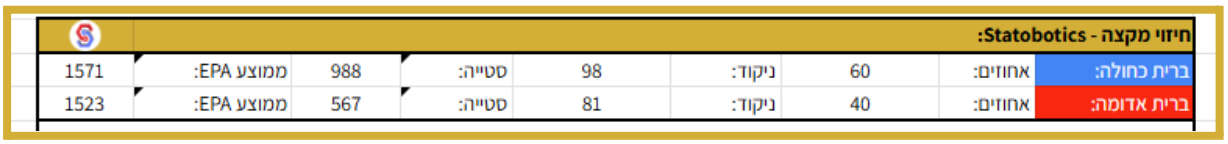
Sometimes, we want to see the maximum data for a group, averages, the average of the last three rounds, or data from the most recent round. To provide maximum control and customization over the data we want, the system includes a dropdown list from which the desired data type can be selected."

Menu for selecting the desired data type.

Integration and interfacing with external APIs

For our convenience, we have connected the system to Statbotics and TBA. We will use them throughout the competition and have integrated them into the system to predict matches.

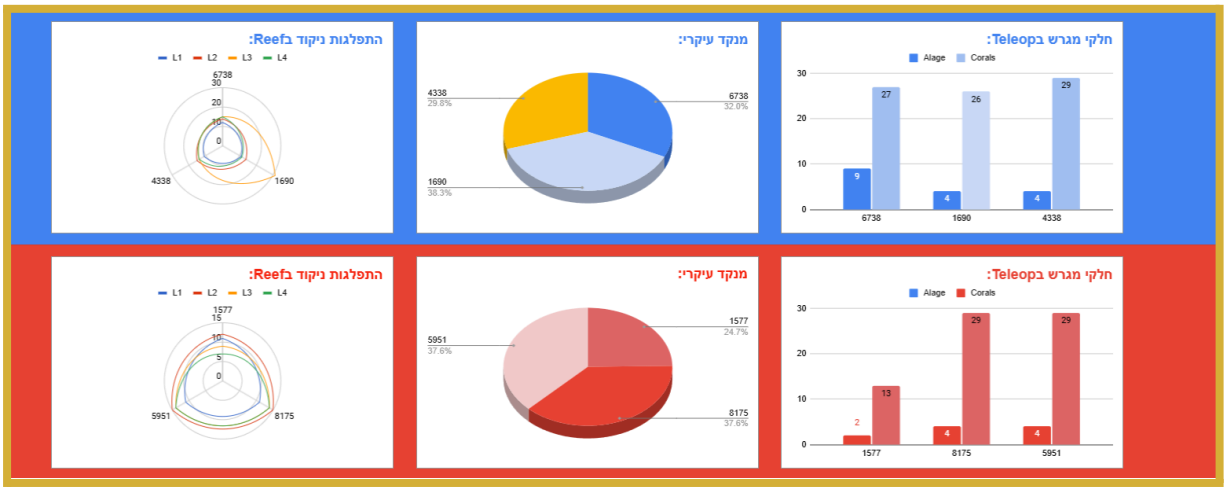
Estimated score displayed according to the OP model.



Estimated score according to the Statbotics model.

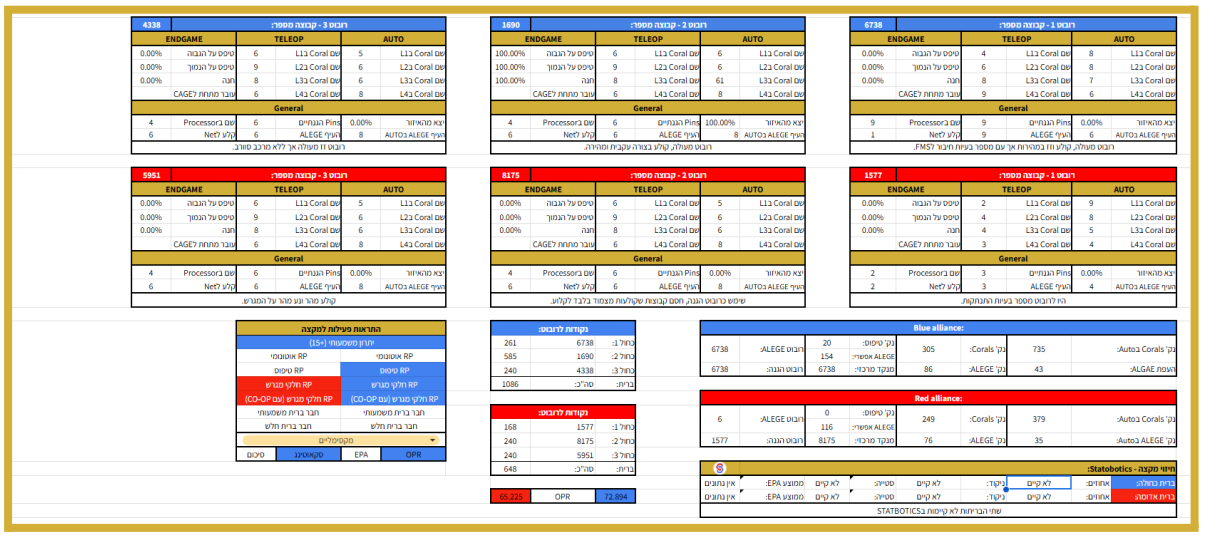
Use of graphs

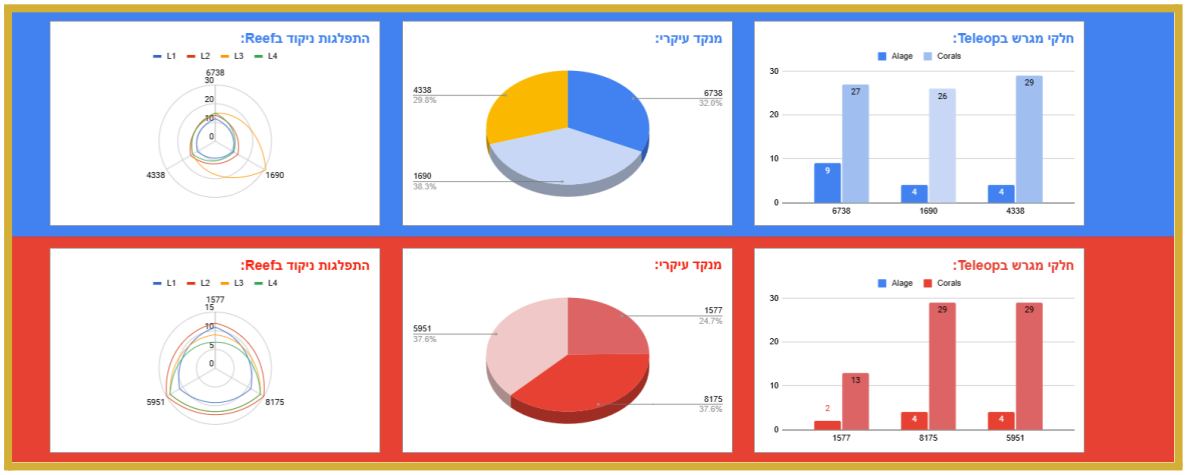
To make it easier to read information and to allow for an intuitive understanding, we have added graphs to the system that help analyze the data.



Use of graphs

After adding all of these, the strategic dashboard looks like this:





Part VI:

Appendices

Example of a Challenge Analysis Document

איך עובדים הסייקלים - שאלות מנחות כמו:

- האם יש דרך להעביר חלק מגרש ממקום למקום בלי לנסוע (2024)
  - חוקית אפשר אבל לא יעיל
- האם יש מקומות במגרש צרים / שמסוכן לעבור בהם מבחינת פאולים שאני יכול להשתמש לטובתי? או נגד היריב שלי
  - קרוב לריף ובזמן הטיפוס באזור הקייג'.
- איך הברית המנצחת באיינשטיין תחלק את הסייקלים שלה במשחק
  - מוריד כדורים מהריף שעובר כשהוא מסיים להגנה, אחד שנקד קורלים, ואחד שמנקד אלג'ים לרמה שהקבוצה מחליטה, ואז עוזר לעשות קורלים.
- האם שווה 3 מנקדים או גם הגנה
  - רובוט הגנה יכול לעזור ברוב מהמקרים, חלק מזה בגלל שאין אזור מוגן מסביב לקורל פידר, ולכן אפשר לעשות שם הגנה יעילה, בנוסף אין מקום ל4 רובוטים במסלול של הסייקלים וכן רובוט הגנה יפריע מאוד לקבוצה היריבה. (אסטרטגיה אפשרית - לחסום אחד מהקורל פידרים ואז יהיה ממש צפוף לקבוצה היריבה). אבל, ישנם 2 קורל פידרים לכל קבוצה ורק רובוט אחד יכול לעשות הגנה בכל רגע, ולכן אפשרי שההגנה לא תהיה יעילה.
- איך כדאי לחלק איזה רובוט בברית מנקד איזה חלק מגרש
  - רובוט אחד מוציא כדורים מהריף ואחד מנקד קורלים(אם הולכים על שני מנקדים והגנה).

איך עובדים הסייקלים - שאלות מנחות כמו:

- האם יש דרך להעביר חלק מגרש ממקום למקום בלי לנסוע (2024)
  - חוקית אפשר אבל לא יעיל
- האם יש מקומות במגרש צרים / שמסוכן לעבור בהם מבחינת פאולים שאני יכול להשתמש לטובתי? או נגד היריב שלי
  - קרוב לריף ובזמן הטיפוס באזור הקייג'.
- איך הברית המנצחת באיינשטיין תחלק את הסייקלים שלה במשחק
  - מוריד כדורים מהריף שעובר כשהוא מסיים להגנה, אחד שנקד קורלים, ואחד שמנקד אלג'ים לרמה שהקבוצה מחליטה, ואז עוזר לעשות קורלים.
- האם שווה 3 מנקדים או גם הגנה
  - רובוט הגנה יכול לעזור ברוב מהמקרים, חלק מזה בגלל שאין אזור מוגן מסביב לקורל פידר, ולכן אפשר לעשות שם הגנה יעילה, בנוסף אין מקום ל4 רובוטים במסלול של הסייקלים וכן רובוט הגנה יפריע מאוד לקבוצה היריבה. (אסטרטגיה אפשרית - לחסום אחד מהקורל פידרים ואז יהיה ממש צפוף לקבוצה היריבה). אבל, ישנם 2 קורל פידרים לכל קבוצה ורק רובוט אחד יכול לעשות הגנה בכל רגע, ולכן אפשרי שההגנה לא תהיה יעילה.
- איך כדאי לחלק איזה רובוט בברית מנקד איזה חלק מגרש
  - רובוט אחד מוציא כדורים מהריף ואחד מנקד קורלים(אם הולכים על שני מנקדים והגנה).

מהירות מערכת איסוף וירי היא יותר פקטור כי הסייקלים קצרים, כמעט וחייב לעשות קו-אופ.

ניתוח אתגר 2025:

כל הדרכים לאסוף את החלק - שאלות מנחות כמו:

- מאיפה יותר כדאי וחשוב שיהיה לנו אופציה לאסוף, ומאיפה פחות ולמה
- האם צריך לאסוף את החלק בוואריאציה מסויימת (2023) או החלק מגרש זהה ככה שלא משנה מאיפה תבוא לאסוף אותו)

תשובה: קורל - אפשר מהפידר בשני אוריינטציות (עומד בזווית, שוכב בזווית), מהרצפה (שוכב, שלושה עומדים)

כל הדרכים לנקד - שאלות מנחות כמו:

- האם אפשר להעביר את החלק מצד אחד של המגרש לצד האחר בלי לנסוע
- האם יש הבדל בניקוד של חלק אחד לחלק השני אם כן, איך אנחנו יכולים לפעול בגלל זה. ובנוסף, תעלו יתרונות וחסורונות לכל דבר
- האם יש מקומות שונים לעשות נקודות? אם כן, איך אפשר לנקד בכל אחד והאם הניקוד שונה? תעלו יתרונות וחסורונות לכל דבר

תשובה: להניח גלילים על הסגול ב4 רמות שונות לשים כדור בתוך הרשת לשים כדור בתוך הזה של ההיומן פלייר. וטיפוס גבוהה ונמוך.

מה קורה באנדגיים - שאלות מנחות כמו:

- האם משימ/ות האנדגיים מביאות יותר ניקוד ממשימות הטלאופ
  - כמשימות מביאות כמות מכובדת של נקודות

- האם שווה לעשות את משימת האנדגיים או פשוט להמשיך לעשות את משימת הטלאופ
  - שווה לעשות את משימות האנדגיים בגלל האר פי שאתה יכול להשיג, שיחסית בר השגה ולא בלתי אפשרי, וגם בגלל שאפילו אם אתב מטפס לגבוה, אתה מקבל 6 נקודות, שכנראה יותר נקודות ממה שהיית מקבל בסייקל באותו זמן
- מה קבוצה גרועה תעשה באנדגיים
  - סייקלים ובסוף תחנה
- מה קבוצה בינונית תעשה באנדגיים
  - תטפס לגבוה
- מה קבוצת טופ תעשה באנדגיים
  - תטפס לנמוך

Ranking of Requirements

בשלב הזה תצטרכו לדרג כל דרישה בין MUST NICE NO, ובין 1 - 10. 1-3 זה לא. 4 - 7 זה נייס. 8 - 10 זה חובה.

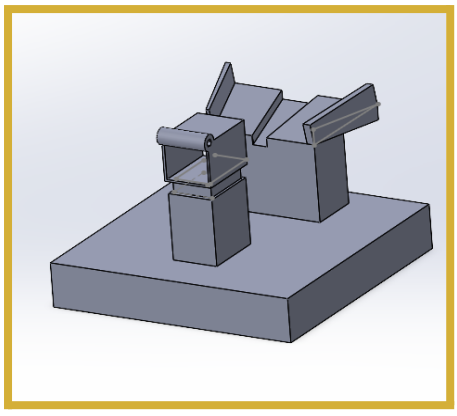
דרישה	MNn	משקל	נימוק	הערות	הערות
מרכז מסה נמוך	Must	9			
רובוט מהיר	Must	9			
רובוט KISS	Must	8			
רובוט שיכול להוציא אלג'י מהריף	Must	10			
רובוט שיכול לאסוף מהמגלשה של ההיומן פלייר	Must	10			
רובוט שיכול לשים קורל בשלב 1	Must	9	עזר לRP, קל לעשות		
רובוט שיכול לשים קורל בשלב 2	Must	10	אחלה ניקוד, עזר לRP, וזה לא שונה מ3		
רובוט שיכול לשים קורל בשלב 3	Must	10	אחלה ניקוד, עזר לRP, וזה לא שונה מ2		
רובוט שיכול לשים קורל בשלב 4	Must	9			
רובוט שמחזיק את הגיים פיס חזק	Must	8			
איסוף נגעת נסעת	Must	9			
רובוט נמוך	Nice	5	ההגבלת גובה השנה היא 106 סנטימטר		
רובוט עם מערכות נגישות לתיקון	Nice	7			

פרופיל רובוט 1-

מערכות ברובוט (כולל הנעה):

1. סוורב
   - 1.1 (הסבר על המערכת) - מערכת הנעה עם dof 3
   - 1.2 (תיאור תהליך הפעולה (מה קורה מרגע הפעלת המנוע עד לתזוזה הסופית)) - הרובוט זז
   - 1.3 (בקרה) - הרבה אינקודרים ואודומטרי וגיירו
   - 1.4 (יכולות ומטרות המערכת)
   - 1.5 (שרטוט)
   - 1.6 (נקודות כשל אפשריות)

2. מערכת איסוף
   - 2.1 המערכת מורכבת שתי פלטות מקבילות המסתובבות סביב ציר, בין הפלטות עוברים 4 צירים מלבד ציר הסיבוב, כך שיש 2 צירים גבוהים יותר ו2 צירים נמוכים יותר, והישר העובר דרך ה2 הגבוהים מקביל לישר העובר דרך ה2 הנמוכים. על שתי זוגות הצירים יש רצועות טיימינג העוברות בינהם, כך שיש שתי שורות של רצועות טימינג המקבילות זו לזו. בין רצועות הטיימינג משתי הצדדים יש פלטות החוסמות את החישוק אך הם בזווית כך שהוא יתמרכז.
   - 2.2 כשמערכת האיסוף פתוחה והרצועות מסתובבות נאסף חישוק, וכשהיא סגורה היא נעמדת כלפי מעל ונשארת ב פריים פרימטר. שולטים בפתיחה/סגירה שלה עם ציר הסיבוב
   - 2.3 בקרת P למהירות איסוף ובקרת PDF לזווית של האיסוף.
   - 2.4 להכניס חישוקים מהרצפה לרובוט
   - 2.5 (שרטוט)
   - 2.6 backlash, חטיפת מכות - זה מחוץ לfp



כאן תצטרכו למלא כמה (1-10) הפרופיל שלכם עונה על כל אחד מהדרישות שהוגדרו בדירוג דרישות. יש לערוך רק משבצות כחולות.

שם הפרופיל:	פרופיל רובוט 1		דירוג פרופיל:	124.2
ראש קבוצת החשיבה:	אורי קורנגוט			
דרישה	זמן	כמה הפרופיל עונה על הדרישה? (1-10)	נימוק	הערות
מרכז מסה נמוך	General	6	כשהוא סגור המרכז מסה נמוך וכשהוא פתוח לא	
רובוט מהיר	General	10	יש סוורב עם המרה גבוהה	
רובוט KISS	General	8	מערכות שכבר התנסנו בהן ולא כלכך מסובכות	
רובוט שיכול להוציא אלג'י מהריף	General	8	לא ברור אם הגלגל יצליח להגיע עד לאורך המתאיימי	
רובוט שיכול לאסוף מהמגלשה של היומן פלייר	General	10	יש מערכת שמתאימה לזה	
רובוט שיכול לשים קורל בשלב 1	General	10	יש מערכת שמתאימה לזה	
רובוט שיכול לשים קורל בשלב 2	General	10	יש מערכת שמתאימה לזה	
רובוט שיכול לשים קורל בשלב 3	General	10	יש מערכת שמתאימה לזה	
רובוט שיכול לשים קורל בשלב 4	General	10	יש מערכת שמתאימה לזה	
רובוט שמחזיק את הגיים פיס חזק	General	8	מחזיק יחסית חזק	
איסוף נגעת נסעת	General	1	אין איסוף מהרצרפה	

Robot Profile + Profile Rating

Points for the Prototyping Process

This document discusses the failure points of the 2023 season and ways to address .(them—methods for improving the prototyping process (shown in the appendix

General Points about the Process:

The key to the success of this process is the goals we define upfront, which determine exactly how the prototype will function and essentially create our work plan. Therefore, emphasis must be placed on this, and because of the importance of the goals, we cannot proceed without them. Before starting work, the goals must be approved.

Control. Without control, everything will fall apart. At least during this stage, there should be daily or every-other-day status updates (we're talking about a week's process) with the system leaders to ensure everything is progressing well. If it's not moving forward, we will address it in greater depth and tackle the root of the problem.

Deadlines. We often end up drifting with deadlines, and this can no longer be allowed. If a system doesn't meet deadlines, no matter how good it is, we simply cannot allow exceptions at this stage.

The process we had last season was good, but within our team, it didn't work due to the reasons listed here.

Every system interacting with a field object goes through a proof-of-concept prototype, even if the team has built it for several years in a row.

During the prototyping week, once we finish with the defined goals, we can move forward with additional goals.

In the initial prototype, our primary goal is to select the final systems for the robot. Therefore, for me, the most basic goal is proof of concept, and even dimensions. Once these are completed (and, of course, approved), we can start working on more advanced goals. Ultimately, by the end of the process, when choosing systems, we want as much data as possible on the system.

There's no such thing as "theoretically" or "it will work." A system that reaches the selection stage without sufficient data is not moving forward. Even if theoretically, the system will work fine.

Daily updates to the team. Every day, the system leader will send the progress of that day and goals for tomorrow to their team (and possibly the entire team). This way, feedback can be received from mentors, and everyone stays updated, making people more motivated to come and invest.

How the process works in practice:

We started with concepts, and now we are preparing to work on the prototype. At this stage, to get a better picture and start in the right direction, we'll define goals. Once the goals are defined, we will pass them on for the necessary approvals. No approval? We don't move forward!

Once we get approval, we start working on the prototype...
Once the prototype is finished, we'll open up the goals again to ensure we have all the data we need. If we don't have it? We'll sit down and get the missing data. If we have everything? Great! We'll pass it for approval, and if there is any, we'll improve the initial prototype and collect more data for a better understanding of the system.

SDR (System Design Review): We've defined the systems for the robot, now we think more deeply about the rest of the data. The goal at this stage is that the prototype should gather all the final data, and the system should be the same in the prototype and in the modeling.

The points mentioned earlier are constantly integrated into this process. Status updates and monitoring are essential for control. We stick to deadlines and work in an organized manner when selecting systems.

More Detail:

There are several types of goals—proof of concept, finding dimensions, integration, etc. For each system you build, you define goals (e.g., what distance the wheels of a roller gripper should be from each other or whether the structure, when placed on pistons, will hold the cone [proof of concept]). For each system, you think about what you want to test, what factors affect your choice, and what will help you in the future? Clearly, the most basic thing you need to do to move a system forward is proof of concept.

Approval is only granted for goals that are defined in detail (details refer to exactly what you're testing and how). If something hasn't been defined—let's say one of the goals is to find the wheel distance, but something in the work process is a bit messy or unorganized (this is just an example, but you get the point)—then we need to think logically. When testing an elevator, for instance, proof of concept isn't the only goal.

In the end, the primary goal of the initial prototypes is to finalize the systems for the robot. Therefore, during the meeting where systems are selected (SDR?), we need as much data as possible for each system for several reasons:

We have more knowledge about the system and understand its potential failure points.

We have a more accurate comparison of systems. When we have a lot of data about one system compared to another, it's easier to think and understand which system is better based on various considerations (difficulty, risks, materials, size, integration, chains/belts/gears, etc.). With all this data, it's much easier to choose the right system.

When improving prototypes, we aim to refine data like dimensions, even build a new prototype to understand a particular structure, a proto-bot, new goals, etc. Ultimately, all this data is fed into the SDR.

When analyzing data deeply, we try to identify the root causes of delays or failures in the prototype. Is the problem with the system itself? Is the team's management ineffective? Is the work too slow?

Ways to Improve the Prototyping Process

Mechanics - Prototyping Presentation: A presentation for mechanical training detailing the ideal prototyping process. This presentation must be shared with the entire team before the season begins.

Ideal Goals for Prototyping Stages:

Basic Prototype:
Main Goal: The goal of the basic prototype is ultimately to arrive at the final systems for the robot that will move on to the advanced prototype stage.

Build a system using a simple yet flexible approach to test a few things and see if they work, or perhaps just to prove it works. Some systems may be things we are unsure will work at all.

Determine which materials are needed for the continued development of the system (for the advanced prototype, so that once we pass the basic stage, we already have all the necessary materials ready in the workshop, like gears, sensors, etc.).

The basic prototype stage should be short (a few days) and focused.

Advanced Prototype:

Main Goal: To finish the design process. The goal is for the final advanced prototype to be a one-to-one replica of what will go on the robot. Of course, there will be additional side elements for integration, and these too should be tested by connecting the two advanced prototypes/proto-bot into a single structure.

Plan in depth, including the smallest details of the system.
Test how the system works in real-world conditions and within its planned structure.
Ensure that control systems are functioning (sensors/software control).

Production Line Tracking Table:

שם חלק	מק"ט	סוג ייצור	חומר	עובי /צבע	כמות	איפה מיוצר	סטטוס	תאריך	עבר אישור רשצ	קובץ STEP
פלטות חיבור לזרוע	2025-GRP-021-PLATE4-CNC	CNC	אלומיניום	6mm	4	בסדנא	ייצור גמור	25.1.25	p	%D7%A7%D7%99%D7%9D%20%
פלטות גיבוקס זרוע	2025-GRP-011-PLATE4-CNC	CNC	אלומיניום	4mm	2	בסדנא	ייצור גמור	28.1.25	p	%D7%A8%20%D7%9E%D7%99%
ספיסרים	2025-GRP-051-SPACER-3D	3D	pla	זהב	8	בסדנא	ייצור גמור	28.1.25	p	%D7%A8%20%D7%9E%D7%99%
ספיסרים	2025-GRP-031-SPACER-3D	3D	pla	זהב	4	בסדנא	ייצור גמור	28.1.25	p	%D7%A8%20%D7%9E%D7%99%
ספיסרים	2025-GRP-032-SPACER-3D	3D	pla	זהב	4	בסדנא	ייצור גמור	28.1.25	p	%D7%A8%20%D7%9E%D7%99%
ספייסר	2025-GRP-011-SPACER-3D	3D	pla	כחול	1	בסדנא	ייצור גמור	28.1.25	p	%D7%A8%20%D7%9E%D7%99%
קיים לחיישן צבע	2025-GRP-013-PLA-3D	3D	pla	כחול	1	בסדנא	ייצור גמור	28.1.25	p	4%D7%A8%20%D7%9E%D7%99
קיים לחיישן צבע	2025-GRP-012-PLA-3D	3D	pla	כחול	1	בסדנא	ייצור גמור	28.1.25	p	4%D7%A8%20%D7%9E%D7%99
פלטת רולר גריפר	2025-PROTOTYPE-012-PLATE4-CNC	CNC	פוליי	5mm	1	בסדנא	ייצור גמור	28.1.25	p	%A8%20%D7%9E%D7%99%D7%
פלטת רולר גריפר	2025-PROTOTYPE-011-PLATE4-CNC	CNC	פוליי	5mm	1	בסדנא	ייצור גמור	28.1.25	p	%A8%20%D7%9E%D7%99%D7%
פרופיל מחזיק	2025-GRP-011-PROFILE-MANUAL	CNC	אלומיניום	40*20*2	2	בסדנא	ייצור גמור	30/1	p	%A8%20%D7%9E%D7%99%D7
פרופיל מחזיק	2025-GRP-012-PROFILE-MANUAL	CNC	אלומיניום	40*20*2	2	בסדנא	ייצור גמור	28.1.25	p	%A8%20%D7%9E%D7%99%D7
פלטה גיבוקס ציר זרוע עם מנוע	2025-ELEV-081-PLATE6-CNC	CNC	אלומנים	4mm	2	בסדנא	ייצור גמור		p	
פלטה גיבוקס ציר זרוע בלימנוע	2025-ELEV-081-PLATE6-CNC	CNC	אלומנים	4 mm	2	בסדנא	ייצור גמור		p	
ספייסר לגלגלים מקדימה	2025-GRP-021-SPACER-3D	3D	pla	כחול	2	בסדנא	ייצור גמור	29.1.25	p	l/1un4rhiwK-dqfl_8AQDepzZxJWR
ספייסר לגלגלים מקדימה	2025-GRP-021-SPACER-3D	3D	pla	כחול	4	בסדנא	ייצור גמור	29.1.25	p	/1cnRUuXaiqcsKKXWEMSjPBzuE
גריפר האחרוריים הבי למעלה	2025-GRP-021-SPACER-3D.STEP	3D	pla	כחול	4	בסדנא	ייצור גמור	29.1.25	p	/14_-Ip5kG6JnXZMpw3hZAKqxD9
ספיסר בתוך הגיבוקס למעלה לציר	2025-GRP-021-SPACER-3D.STEP	3D	pla	כחול	2	בסדנא	ייצור גמור	29.1.25	p	l/1eMcrD77hohA0wzT3skAdYQJT
ספיסר בתוך הגיבוקס למחה לציר	2025-GRP-021-SPACER-3D.STEP	3D	pla	כחול	2	בסדנא	ייצור גמור	29.1.25	p	l/1S7Y3NSvJyCn_8MyP5nOWtpX9
ספיסר בתוך הגיבוקס למעלת החיצו	2025-GRP-021-SPACER-3D.STEP	3D	pla	כחול	2	בסדנא	ייצור גמור	29.1.25	p	d/1i5BRe8BlA4BfYL27sV7TakD7Z
ספיסר בתוך הגיבוקס למעלה למעלית האמצ	2025-GRP-021-SPACER-3D.STEP	3D	pla	כחול	2	בסדנא	ייצור גמור	29.1.25	p	d/1rBY1CuFIR3Ej5RyapcU3Pk-IR