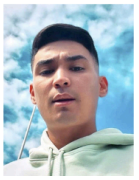
Final Assignment

62410 CDIO-Project

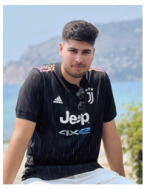

Ahmad
El-Hag
S210030


Muhamed
Sbeihi
S210085


Mohamad
El-Asadi
S224301


Mahdi
Ibrahimi
S210077


Faruk Emir
Degirmenci
S222359


Ahsunalla
Wahidi
S184858

Github Repository:

https://github.com/Veteranlegend/CDIO-EV3-Final.git

Due to many technical issues with committing to github for some team members, we ended up doing pair programming on just one computer for the whole system. The appendix shows which member has worked on each task.

Youtube link to Final Video Status:

https://youtu.be/ZyO5-aNLVIo?si=IKy_D-ewO4l3BHI9

Introduction

Imagine a world where you don’t have to walk around and collect your own golf balls after a long day of playing golf. That is the project that has been assigned to us, a group consisting of 6 students at Denmark's Technical University. Our course named CDIO aims exactly at what was mentioned earlier, a quick, reliable and smart way of collecting golf balls. CDIO which stands for conceive, design, implement and operate is a framework that provides a structured approach. Digging a little deeper, the conceive part is where there is focus on requirements, identifying stakeholders, defining the problem, the short and precise way of saying it would be where there is focus on analyzing the project. The design part would be where we take our requirements and all that we came up with of ideas and turn them into detailed plans for our product. Implementation is the part that focuses on taking what we came up with in our design phase and turning them into actual code, meaning that this is the part where all the coding and testing and developing of the product takes place.

In this report a thorough understanding and overview of our project will be presented, with focus on our design and design choices and implementation of our robot with the ability to collect and release tennis balls. Our coding is all done with the Python programming language and other tools have been used as well, one of those tools being the roboflow website for image recognition.

Status Over Project and Product

Gantt Diagram has been delivered in a pdf file to review.

It is important to highlight the significant changes that occurred in our project during the transition from Delivery 3 to the start of the 3-week period concerning the video status. Initially, our strategy for image recognition relied on using CV2 and color detection. However, we discovered this approach was not so practical. Consequently, we conducted further analysis and testing, which led to the decision to train a machine learning model for image recognition.

To accomplish this, we planned to use Roboflow for annotating and training the model. The team first needed to familiarize themselves with the Roboflow tool. Once proficient, we annotated all course images, marking objects such as the white ball, orange ball, walls, egg, big goal, small goal, and cross.

Following annotation, we trained the image recognition model in Python using the model created and trained in Roboflow.

However, we encountered issues with the sustainability of Roboflow models when using the API. As a result, we switched to training with the YOLOv8 model, which allowed us to download the model – something not possible with the Roboflow-trained model. Despite this, we observed that the model did not accurately detect walls. Increasing the image size or dataset resulted in a slower, unsustainable model. To address this, we utilized CV2 for wall detection and disregarded the wall detections from the model.

These adjustments led to delays, causing us to fall behind the initial project timeline. Consequently, the team had to dedicate additional hours to realign with the original project plan.

Final Status

Overall project health

X	On target
	Risikabel
	Fare!

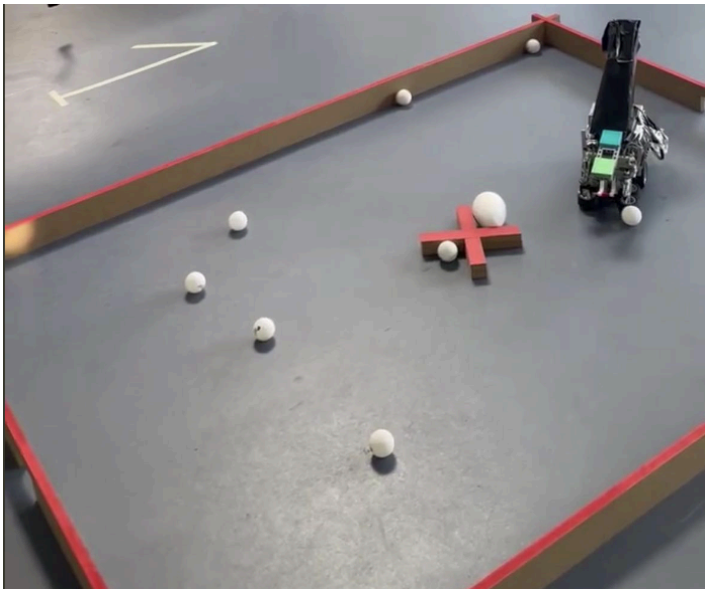
Partial deliveries

Partial deliveries	On target	Risikabelt	Fare	**Notes**
Report	X			Report has been updated and evaluated.
Gantt Diagram	X			Gantt Diagram has been finished and evaluated.
Status Report	X			Status report has been finished and evaluated.
Risks	X			All risks have been revised and evaluated.
Final Video status	x			The final video status has been recorded, edited and evaluated.
ZIP file including all relevant files and documents.	X			All files and documents have been collected together and put into a ZIP file to be delivered.

Competition day: Evaluating system performance

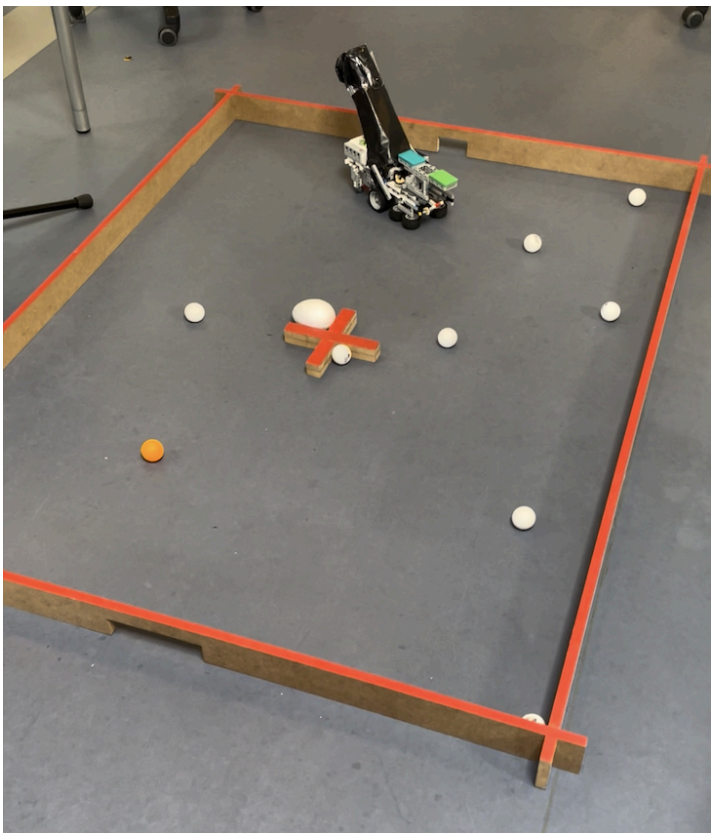
First demonstration:

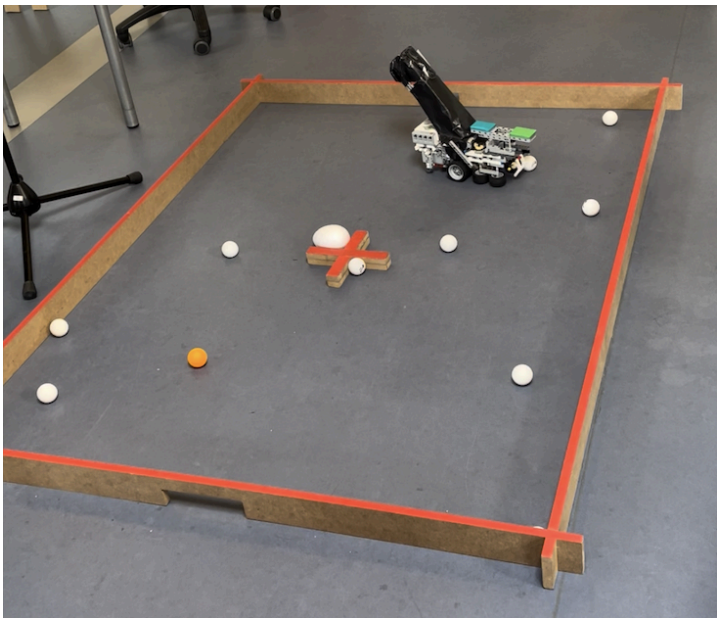
During our first demonstration on the competition day we were met with some obstacles that we did not entirely foreshadow. The obstacle we were met with was the change in the lighting on the course itself, it was completely different compared to what we had previously faced during our test runs. Our robot made some weird decisions during the demonstration because of the lighting, one of them being that it went into the wall a few times and then started going in circles. It was not all bad, we were able to pick up two balls and came close a few other times, but unfortunately we weren't able to place them inside the goal, and it was decided that we landed on the rather low points of -1100.



Second demonstration:

For the second demonstration on the competition day we came prepared, or so we thought. We fixed everything wrong with the lighting and everything we thought could go wrong and made tests ensuring us of success. During the beginning of our demonstration our robot made an attempt to pick up a ball but missed unfortunately, and seconds after missing the ball our robot ran out of battery, which we found incredibly weird, since it had been charging for hours before the competition. Unfortunately we could not restart the robot in time, and our time ran out and we landed once again with points that we were extremely unsatisfied with. For the second attempt or the second demonstration we landed on 0 points since we did not hit any walls, cross or the egg during the time our robot was running.





Proces

We have made use of many different tools and a wide range of methods in this project. The reason for the use of these many tools and methods are for a higher chance of success. This section will give an insight on how project management has impacted our project, giving an insight on the methods and tools used, and also give an insight on the changes that have occurred from the beginning of the project till the end of it.

Methods and tools used for project, risk etc.

Project Planning

For our project planning we made use of a Gantt Chart in Instagantt which helped us as a team to create an overall project plan, keeping track of our tasks as well as estimated and actual time on each tasks, also milestone deadlines and assignees. The Gantt Chart was a very essential tool for our project planning to keep a good and overall structure of the project. To help with our time estimation for tasks, we made great use of Planning poker to estimate hours spent. Also with the Gantt Chart, it was a great support to have an overview of each member's responsibility for the given tasks. The Gantt Chart was updated many times during the project and was made sure to be easily understandable for all members to look at.

Risks

Analyzing and identifying risks in our project was also a big part, due to many risks occurring which the team sat down together multiple times to gather different insights and perspectives. During these sessions we used some given templates to help document our risks. The team made sure to document the following for each assignment:

- Overall project health
- Identify risks (Id, Risk, Impact, Probability & Score)
- Acces risks
- Mitigate risks (Risk, Score before, Strategy, Score after, Responsibility & Follow-up date)

These methods helped to analyze, identify and mitigate risks for our project.

Communication tools

For communication we needed various collaborative tools, collaborative tools such as Messenger (Facebook), Discord, Github and Google Docs. We made use of messenger (Facebook) and Discord for day to day communication and various other tasks such as sending files and images and setting up meetings etc. Pair programming was done with the coding on one computer due to commitment issues in GitHub. So throughout the project each member who had made changes to the code had also made sure to leave comments in the coding to update the rest of the team. Google Docs was used for reports and other heavy writing tasks. Google Docs made it so multiple people from the group could work at the same time, and the group could all keep track of each other's progress and work.
With these tools we have managed to work on the project in a way that has given the feeling that the group was always in control, both in teamwork and in the project work.

Project Changes throughout

A few changes were introduced during the project, one of them being the orange ball that now had to be delivered as the first ball. The second thing that was introduced was the egg, a white egg you had to not mistake for a white ball during the course. Another change was the fact that during a meeting with the professor, our robot was not seen upon as steering in the right direction and was criticized for its design decisions and we had to restart the design process and walk in a different direction when it came to the design, not only once, but twice did we start the entire design phase over and rebuild the robot from scratch.

These changes were all bumps along the way, that we as a group managed to overcome with communication and adaptability to changes that we had not seen coming.

Efficiency of project management

In the project we decided to pick out an official project manager for the team due to different reasons. It helps to provide a much more structured approach to the project by outlining the tasks and milestones. It helped the team to break the project into smaller tasks, keeping an overview of the progress and time estimations to make sure the project is on track. It also helped the team to make sure that every member is informed about the status of the project and future approaches. The project management also helped to identify risks throughout the project.

Product

The upcoming parts of the report will give an insight on the development of the product, meaning the development of our ev3 golfbot, the analysis, the design and last but not least the tests. It should be needless to say the development of the product is backed up by the project management framework. The reason for that is that we want to ensure that our product is of utmost quality.

Analysis

We have spent countless hours on the analysis part of the project, the reason for this is that we wanted a clear analysis and vision of the project's purpose. This analysis of ours didn't last a short amount of time as clarified in earlier, it lasted throughout the entirety of the project. This can be seen as we have redesigned our ev3 golfbot several times, and have had to rethink parts of the design of our ev3 golfbot because of issues occurring, which we will specify later on in the report.

We have made use of the FURPS+ analysis method for our analysis, we have done so to define our requirements. We have also made use of the MoSCoW method for prioritizing the requirements. By doing an analysis using these analysis methods, we have hopes of getting a view and perspective on the requirements that would change our design and implementation for the better.

FURPS+ model

Functionality:

The functionality for the robot is as such, that the robot must be able to accomplish the requirements set for it on its own meaning autonomously. The requirements being that the robot must collect 11 tennis balls, and then release every ball into either the small goal or the big goal, with the condition that the orange tennis ball be collected first. The robot must also not mistake the egg placed on the course for a white ball, and the robot must not hit any obstacles set on the course, such as the cross, walls or the egg. All of this means that the robot must be able to locate and identify all the things specified earlier.

Usability:

The usability part focuses on the robot and its autonomous operation. The process of setting it up should both be easy and user friendly. This is achievable with instructions that are clear and precise. On the other hand should the robot not be successful in this process, a way of receiving feedback should be provided, feedback on what went wrong and error handling.

There are ways to achieve this, either with some kind of interface or other method that lets the user get an insight on what exactly went wrong and where.

Reliability:

The robot must be reliable in all aspects, meaning that the robot must be able to go through the course and all its obstacles everytime without any major problems, the robot must not brake or fail its task when in the middle of going through the course. It should also be able to handle any problems occurring during presentation, meaning that if anything does go wrong the robot should be able to restart and start from where it left off.

Performance:

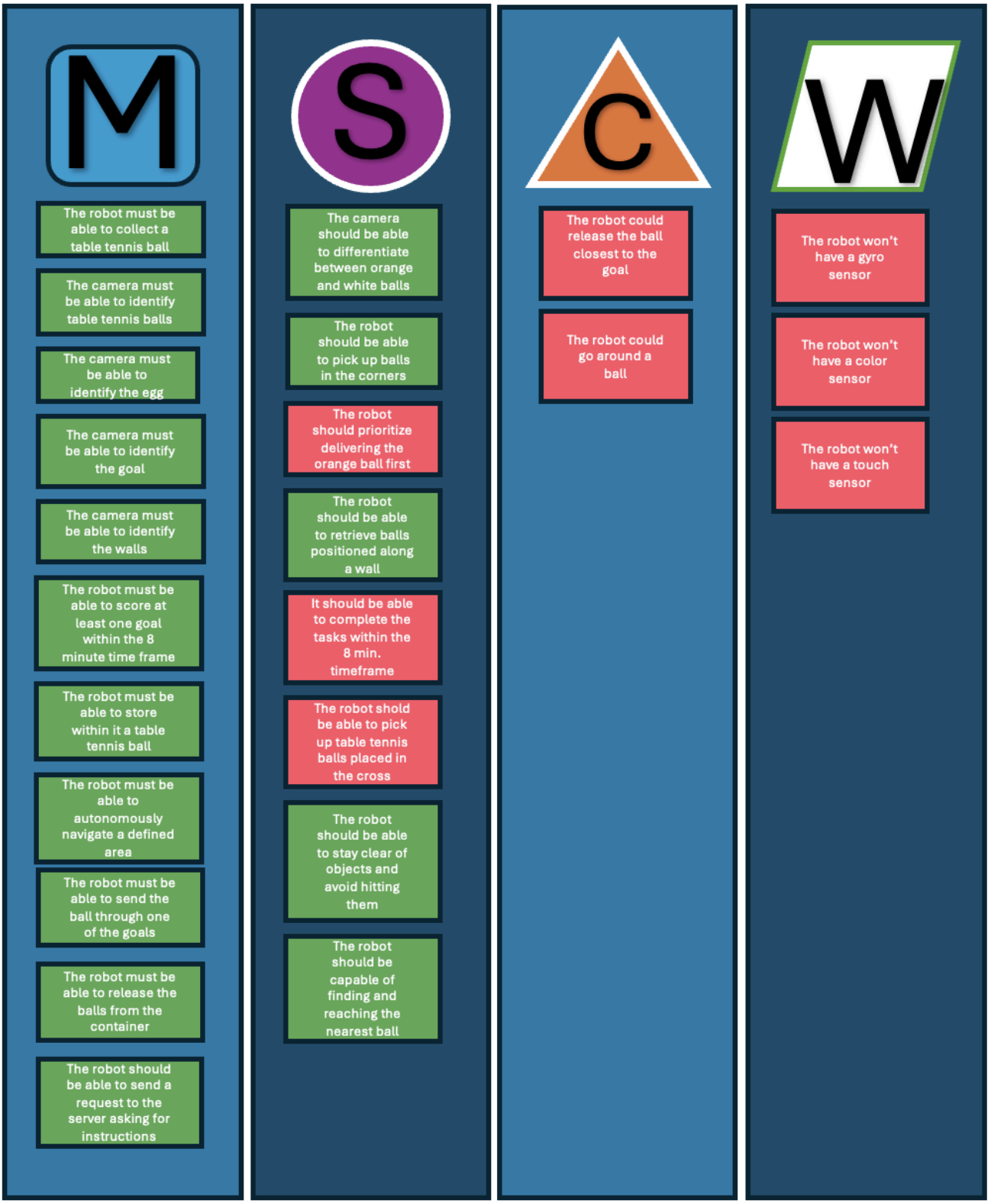
To win the competition the robot must be able to perform and go through the obstacle in a certain time frame. This can partly be done with the robot being able to go through the course as accurately as possible and locate the tennis balls precisely, other ways to achieve this can be with path planning algorithms or other techniques like it.

Supportability:

There are things that we as a group must be prepared for and things that we must be equipped with in order to adhere to the supportability part of the FURPS+ model, meaning that in case somethings happens to go wrong or an issue may arise right before the beginning of the competition, we must have with us the solution or answer to what went wrong so that we can find a solution and fix it right away. Not only is this true when it comes to the software, but also true when it comes to the hardware, meaning that if something were to happen to any part of the robot, we must be able to replace that part quickly and efficiently. Therefore both the software and the hardware must be designed or built in such a way that we can easily fix any issue that may arise.

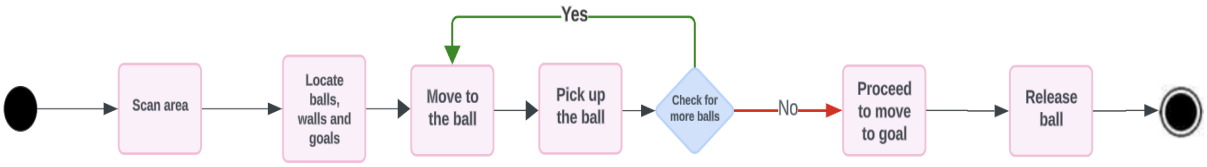
MoSCoW-Model

We made use of the MoSCoW model for the prioritization of our requirements. The way the model works is that the must have requirements are the ones that are essential and most important and therefore have to be implemented for the robot to function well. The should have requirements can be implemented after the must have requirements since they aren’t as high up in the prioritization list along with the could have requirements. Last but not least we wouldn't have requirements, and to put in a simple way, those are the requirements that won’t make an appearance in the project, meaning they won’t be implemented. The requirements that we have successfully implemented are shown with a green color in the MoSCoW model below, and the requirements that we haven’t implemented are shown with a red color.


M
The robot must be able to collect a table tennis ball
The camera must be able to identify table tennis balls
The camera must be able to identify the egg
The camera must be able to identify the goal
The camera must be able to identify the walls
The robot must be able to score at least one goal within the 8 minute time frame
The robot must be able to store within it a table tennis ball
The robot must be able to autonomously navigate a defined area
The robot must be able to send the ball through one of the goals
The robot must be able to release the balls from the container
The robot should be able to send a request to the server asking for instructions
S
The camera should be able to differentiate between orange and white balls
The robot should be able to pick up balls in the corners
The robot should prioritize delivering the orange ball first
The robot should be able to retrieve balls positioned along a wall
It should be able to complete the tasks within the 8 min. timeframe
The robot shold be able to pick up table tennis balls placed in the cross
The robot should be able to stay clear of objects and avoid hitting them
The robot should be capable of finding and reaching the nearest ball
C
The robot could release the ball closest to the goal
The robot could go around a ball
W
The robot won't have a gyro sensor
The robot won't have a color sensor
The robot won't have a touch sensor

Activity Diagram

The activity diagram shows the process of how the robot collects the balls and takes them to the goal. It begins by scanning the area with image recognition, then locates balls, walls, and goals. Next, it moves to a ball, picks it up, and checks for more balls. If more balls are found, it repeats the process and if not, it proceeds to move towards the goal and releases the ball. This cycle continues until all tasks are completed.



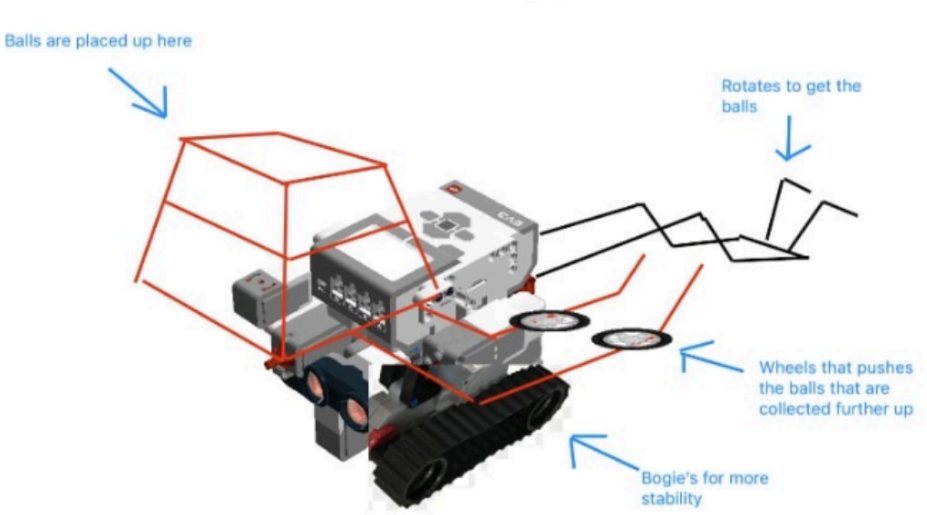
Design

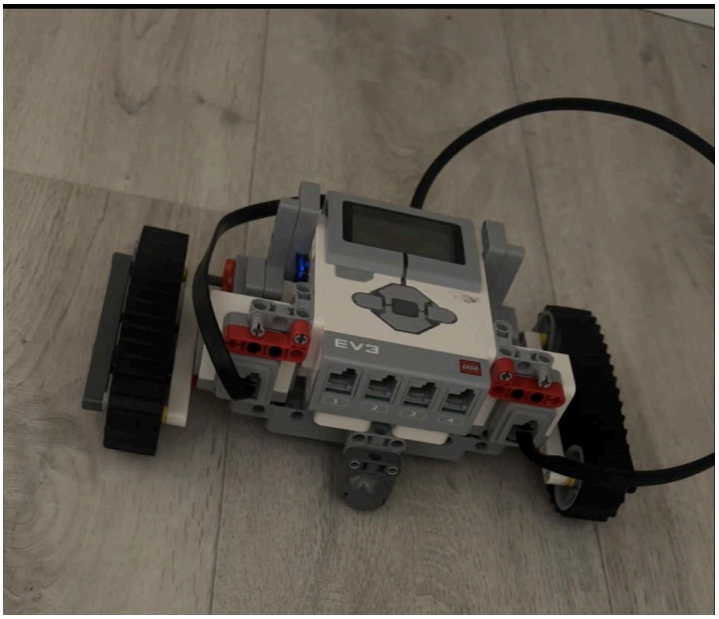
For the design part, we focused on making sure that all of the requirements are changed into much more detailed design, which will stand out as a more structured overview for the development process of the hardware & software.

Hardware

1. Prototype

Not far into building our first prototype of our EV3 golfbot we were criticized for our design decision by our professor. Our professor was not happy with our decision on using boogie's (tank wheels), our thoughts differed when it came to the reason for using the boogie's(tank wheels), we backed our decision of using the boogie's with the idea of a robot that would be incredibly stable, and the way to that would be with the boogie's. Our professor did not see that as a good idea or decision, the reason for that was because the floor we would present our robot on would be normal flat ground, and boogie's are usually used for uneven ground, and that we would sacrifice some speed for stability, stability that we could find in other ways anyways, but told us that we would have the last say in the matter whether we would stay with the prototype or decide to rethink our design and start over. In the end we decided that we would start over with the design phase and rebuild our robot in a fashion that would suit it for an indoors test ground for the final day presentation.





Pros:

- The ball container is placed on top of the robot, which does not interfere with the robot's sensors.
- The robot is more stable when the ball container is located on one side, and arms for catching balls are on the opposite side of the container.
- The wheels that convey the ball to the container are located close to the arm that catches the ball, thus minimizing the time needed to catch and send the ball to the container.
- Bogies were used to enhance the robot's stability, as they have more contact with the floor compared to using four wheels.

- Thanks to the length of the arm wheels, it is possible to hold more balls than the container can officially accommodate.
- The arm wheels also assist in emptying the container without any balls getting stuck.

Cons:

- Bogie's struggles with turning.
- There might be instances where balls fall if the container and the wheel arm lack space, potentially causing multiple balls to flatten at once.
- It could be challenging for the wheels to pass the ball on to the container as the surface of the balls might be slippery.
- The bogie's design slows down the robot's speed because it has more contact with the ground than the 4-wheel version of the robot, which means we spend more time for the robot to move towards the balls on the field.
- Due to the long arms for grabbing balls and the wheel arm that sends the balls to the container, the robot's weight will be higher, which makes the robot slower.

2. Prototype

Upon building our EV3 robot, we have implemented the iterative design process. This approach was chosen because it allows us to systematically refine and improve our robot's design through a series of iterative cycles. By breaking down the design process into smaller, manageable steps, we can address potential challenges and distractions in a controlled manner, rather than being overwhelmed by them all at once. Iteration enables us to test, analyze, and refine each component of the robot, ensuring that it meets the desired objectives effectively.

The iterative design process is a good strategy for several reasons. First, it allows us to take a step-by-step approach, focusing on one aspect of the design at a time. This helps us to maintain clarity and focus, reducing the risk of overlooking critical details or making rushed decisions. Additionally, by continuously testing and refining our design, we can identify and address any issues or weaknesses early on, before they become more significant problems later in the development process. Add to that, iteration encourages collaboration and feedback among team members. As we iterate through different versions of the robot, we can gather

input from various stakeholders, including team members, mentors, and potential users. This feedback loop enables us to incorporate diverse perspectives and insights into the design process, resulting in a more robust and user-friendly final product.

The iterative design process played a crucial role in the development of our robot's gathering mechanism. From the outset, our primary objective was to create an efficient and reliable method for collecting the 11 balls required for the task. However, as we began prototyping and testing different designs, we encountered various challenges that required continuous refinement and adjustment.

One of the initial challenges we faced was ensuring that the tube designed to contain the 11 balls functioned smoothly without any obstructions. Despite our best efforts, we found that some balls were falling in between the LEGO parts, disrupting the collection process. To address this issue, we iteratively modified the design by connecting additional LEGO parts underneath the tube to prevent the balls from falling into the gaps, ensuring smooth and consistent ball collection.

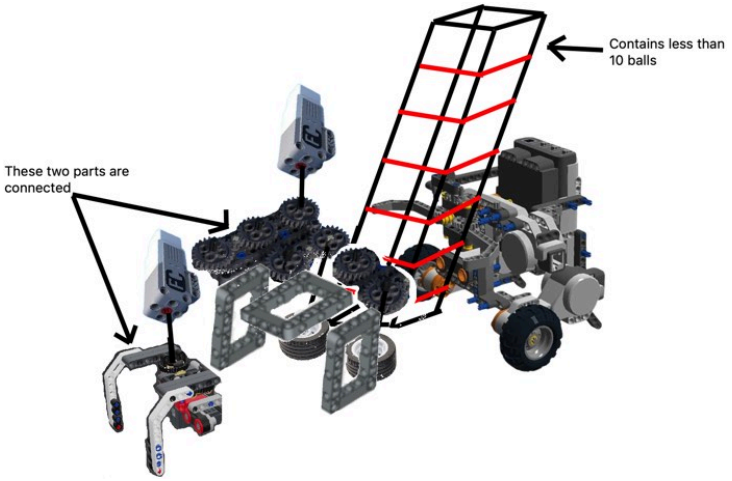
Additionally, we implemented a mechanism for grabbing and releasing the balls, allowing the robot to both collect and empty them as needed. This mechanism involved the use of two medium tires mounted on an axle, designed to grip the balls securely during collection and release them smoothly during disposal. Through iterative adjustments and testing, we optimized the design to ensure reliable operation and efficient ball handling throughout the task. Balance proved to be a persistent issue throughout the iterative process. As we experimented with different configurations and mechanisms, we encountered stability problems that threatened the functionality of the robot. To overcome this challenge, we employed creative solutions, such as adjusting the robot's dimensions and incorporating various counterbalancing techniques. By continually refining the design and testing for stability, we were able to achieve the necessary balance while maintaining the integrity of the robot's structure. Another significant aspect of the gathering mechanism was the development of the hand mechanism responsible for pushing the collected balls into the tube. This component required careful consideration and experimentation to ensure efficient ball transfer without compromising the robot's stability. Through iterative testing and refinement, we devised a mechanism using different sizes of gears to achieve the desired pushing action while maintaining balance and structural integrity.

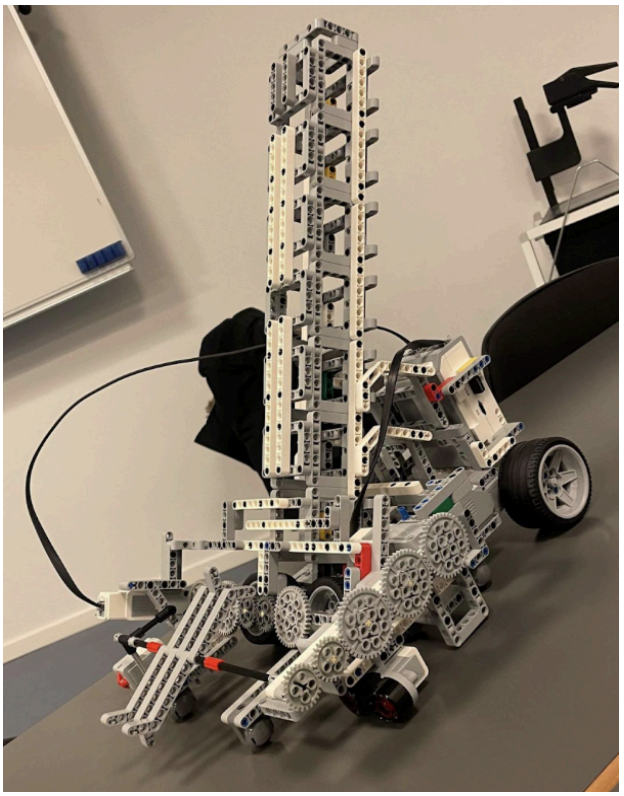
During the iterative development process of our EV3 robot, we encountered challenges with the alignment of the medium tire mechanism and the overall stability of the robot. Despite our efforts to ensure precise alignment, we discovered that the tires were not perfectly aligned vertically. This inconsistency posed a significant problem, as it affected the reliability and effectiveness of the mechanism. To address this issue, we sought innovative solutions to realign the tires, ensuring smooth operation and reliable ball handling.
In addition to the challenges with the medium tire mechanism and the overall stability of the robot, we also encountered an issue with the EV3 brick itself. The brick appeared to be loose and exhibited significant movement up and down, impacting the stability and reliability of the entire robot system. Recognizing the importance of a secure and stable foundation for the EV3 brick, we took proactive measures to address this issue.

To mitigate the problem of the loose EV3 brick, we designed and implemented a new fundament specifically tailored to provide a secure and robust mounting platform. This new fundament was engineered to ensure that the EV3 brick could sit tightly and securely without excessive movement. By reinforcing the connection between the EV3 brick and the robot's structure, we were able to minimize the undesirable movement and instability, thereby enhancing the overall performance and reliability of the robot during operation.

In addition to mechanical challenges, we also encountered difficulties in connecting the EV3 robot to the computer for testing purposes throughout the project. Despite various attempts to establish a connection, we faced obstacles preventing us from accessing and testing the robot's functionality effectively. After extensive troubleshooting, we identified that the SD card inside the EV3 was causing the connection problem and hindering the robot's functionality. Upon removing the SD card and rebooting the EV3, we successfully restored normal operation and resolved the connection issue.

Once the connection problem was resolved, we utilized the EV3 Classroom app to test the functionality of the robot and evaluate its performance in simulated scenarios. This allowed us to verify that all mechanisms and functionalities were working as intended, enabling us to identify any remaining issues or areas for improvement. By leveraging the EV3 Classroom app, we were able to conduct comprehensive testing and ensure that the robot met all requirements and objectives effectively.





3. Prototype

Due to our motivation and eagerness to win the competition, and based on testing the capabilities of our prototype 2 in relation to its goals, we decided to change course by redesigning the robot entirely. We encountered several practical issues with the robot's performance in prototype 2, such as its size, which posed risks because of the possibility of

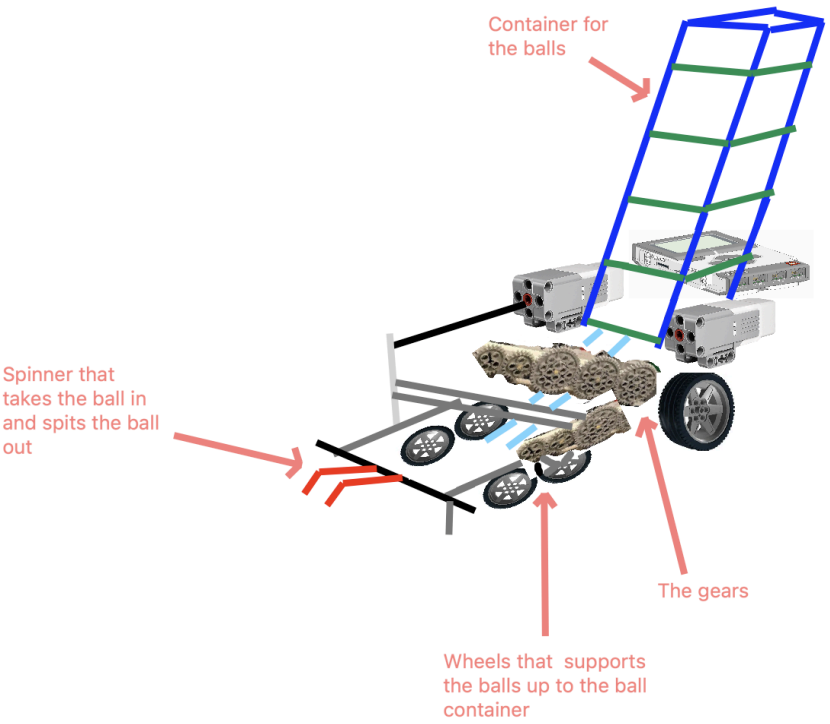
the balls being located anywhere within the designated area and especially close to the walls. So flexibility is crucial and the robot must perform its tasks reliably without risking failure.

One challenge we faced was ensuring the robot could smoothly collect balls whether it was stationary or moving. The original design didn't consistently achieve this, prompting us to consider a redesign to enhance security and efficiency.

We decided to build a smaller, more flexible, and reliable robot capable of performing the same tasks with better performance. Our new prototype 3 is much smaller, with a shorter container for the balls. We completely revamped the mechanism responsible for moving the balls up and down, making it more practical and efficient.

During the redesign process, we encountered minor issues, such as building a mechanism to push balls inward in case they were close to a wall or the ball is just out of the robot's range so we built a hand mechanism for that. Under the process we still had to focus on implementing that mechanism without risking falling into the same problem as of prototype 2, so we had to do it in a way where it must be achieved without exceeding the robot's intended width. Additionally, we encountered a problem that is related to the balls getting stuck between the mechanism and the container, for that we built a staircase like feature to ensure smooth movement. Moreover, We struggled with setting up the gears to maintain a smooth flow under pressure and ensuring the mechanism worked without getting off track.

Our new design was inspired by continuous feedback from TAs and observations of other groups' robots. We also got some inspiration and ideas from the internet whenever we got stuck so that we creatively could overcome all the obstacles we faced. We strived to make the robot as fast as possible without compromising quality. Working as a team to accomplish this mission was not easy, but the satisfaction of overcoming these obstacles made it all worthwhile.

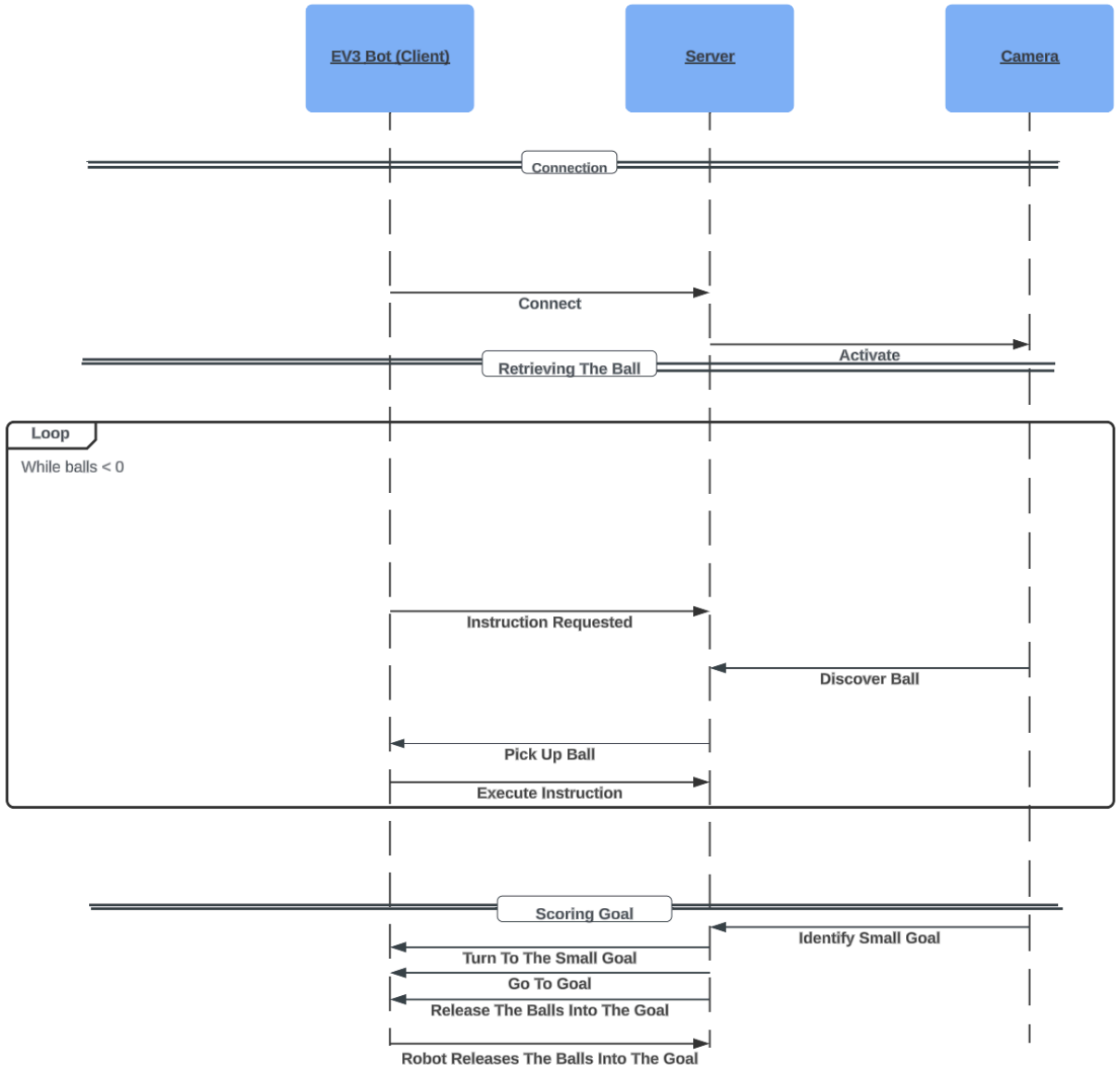


Software

To implement our software, it was important to also design the software which would help us to keep a structured system for our software. Here we made use of different diagrams to get a better understanding and overview on the system of the project.

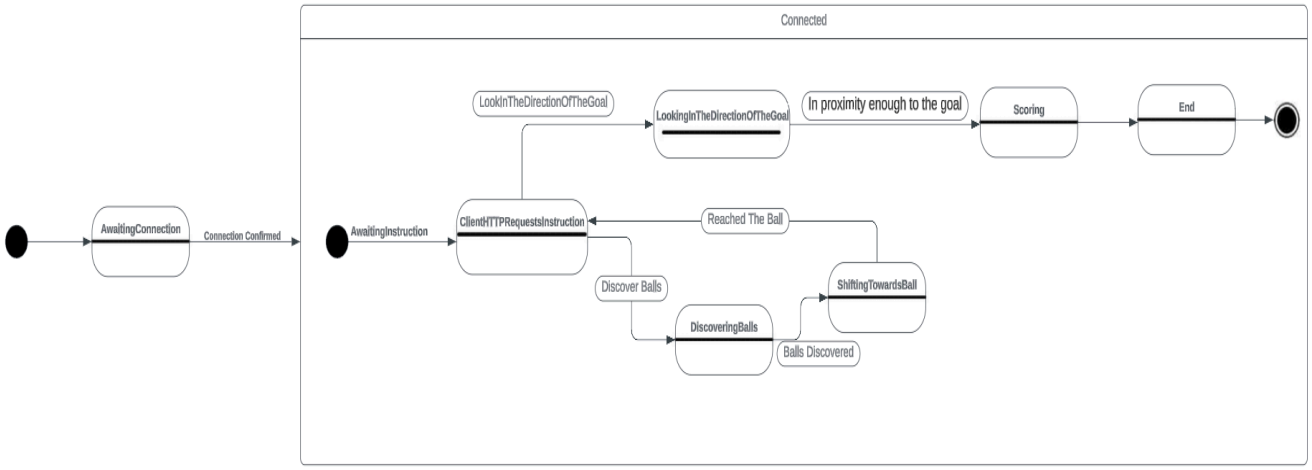
Sequence Diagram

We made use of the sequence diagram in order to get an overview of our system and how it behaves in a series of activities.



State Diagram

The following diagram named the state diagram plays an important role when it comes to the understanding of the logic of the events and what happens in the system. The state diagram shows some of the interactions that happen between the so called client, server and the EV3 robot, and the states.



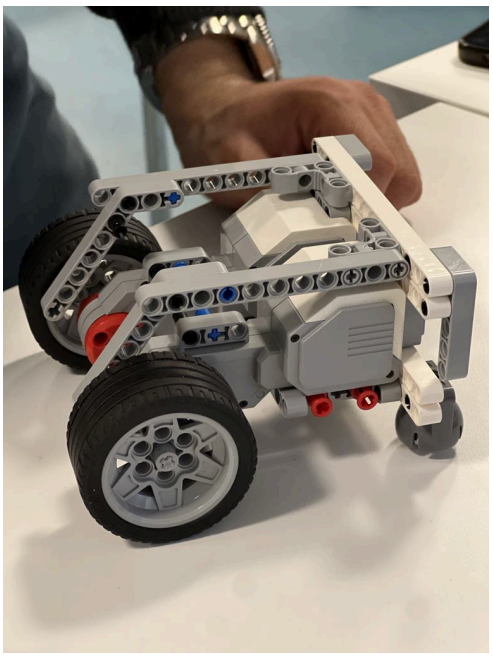
Implementation

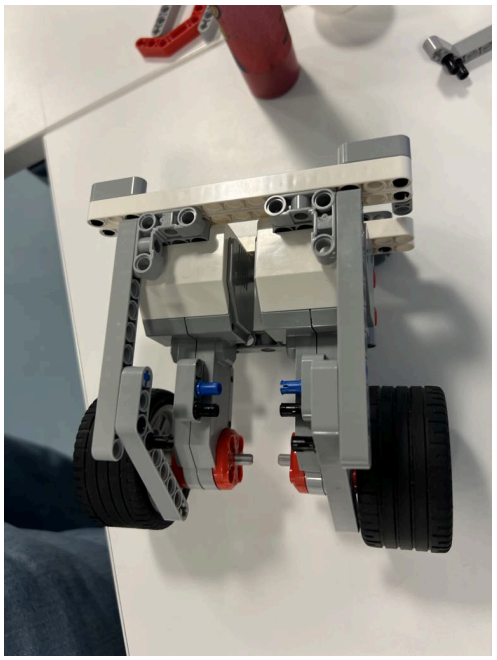
To build a structured and logical system, we rely on our designs and visualizations for both software and hardware. These visuals act as blueprints, guiding us through the development process. This planning ensures everything integrates smoothly, reducing errors and making the development phase more efficient. Our detailed designs are essential for creating reliable and scalable systems.

Hardware

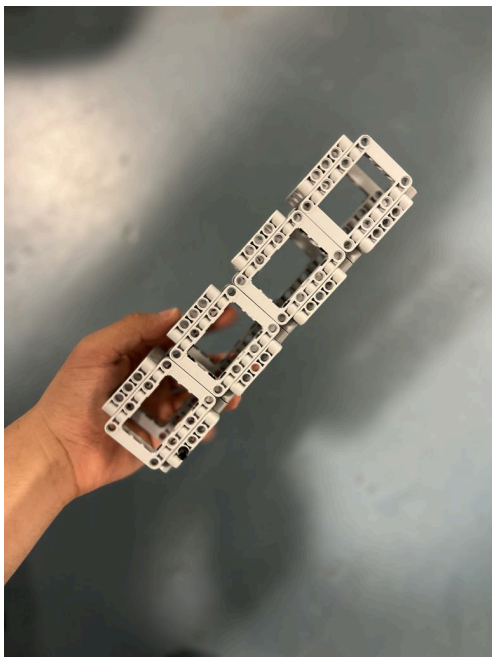
The way we started building our EV3 golfbot was by building the most important parts separately, such as the container, the base of the robot like the wheels and skeletal structure etc. When all our parts came to look like our design we began to put the parts together.

We made the base of the robot in a fashion where each of the two motors that were handed to us would support a wheel on each side of the robot. A metal wheel ball would help in front of each of the wheels as support for the front.

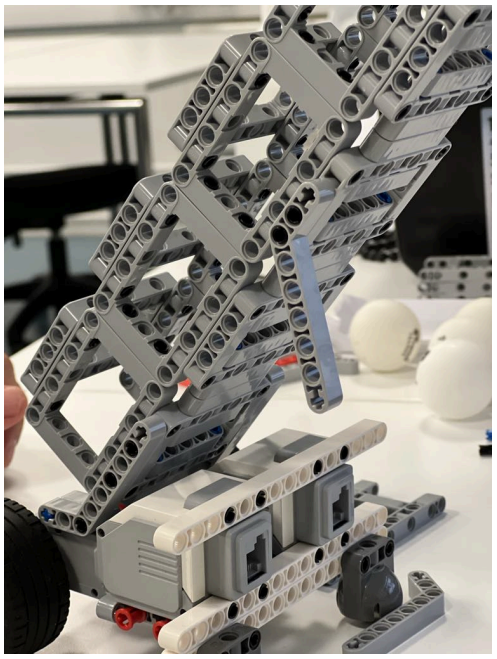




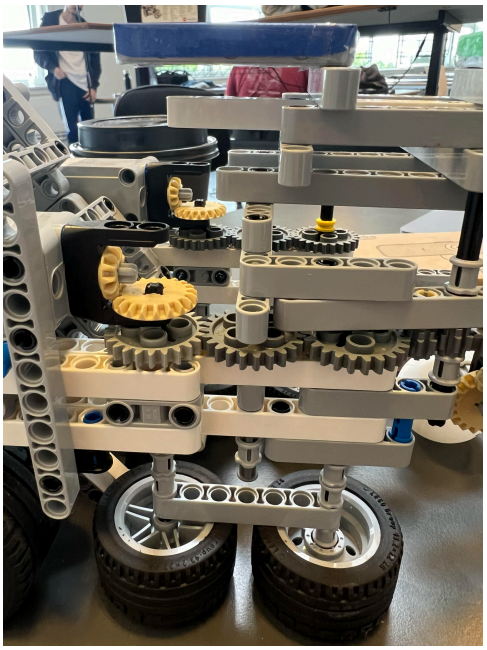
We designed the container so that all 11 balls were able to fit in one go. We decided that for efficiency reasons for the competition day, so that our robot wouldn't have to make several trips to the goal and back.

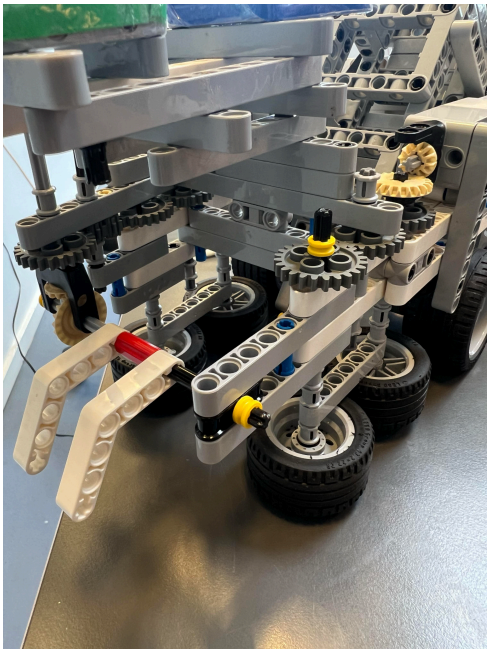


The container was placed in an angle that would not only make it easy for the pickup of the balls but also the release of the balls. It would be quite difficult for the balls to reach the containers if we had left it as is in the picture, so what we did was using small pieces at the entrance of the container that would work as a ramp, and as a result the balls had no trouble climbing the container and leaving the container.

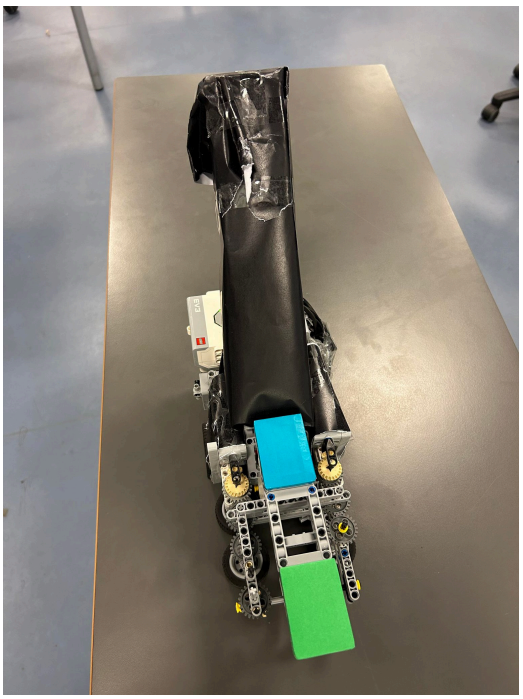


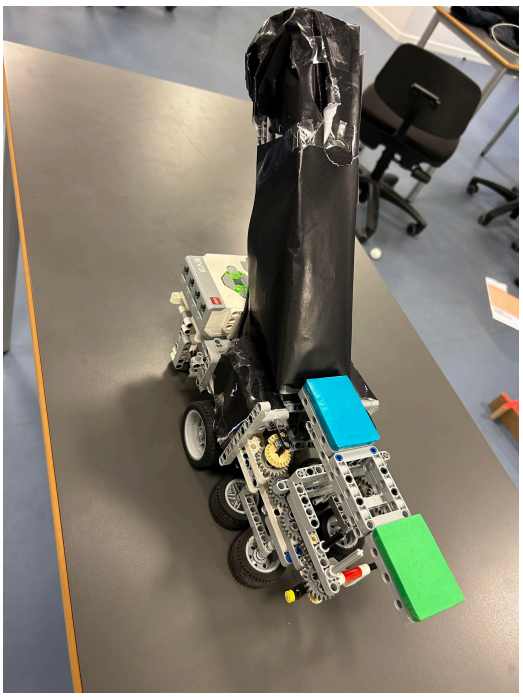
For the gears we decided to split it into two, one side would be focusing on the two wheels on the inside that would assist the balls on their way, this side uses a five gear system, and is powered by a small motor. The other side would also be powered by a small motor but use a ten gear system, the reason for this is that this side would also be responsible for two things, one being the spinner at the beginning of the robot that spins and takes the ball in and the other being the two wheels on the inside responsible for the assisting of the balls.

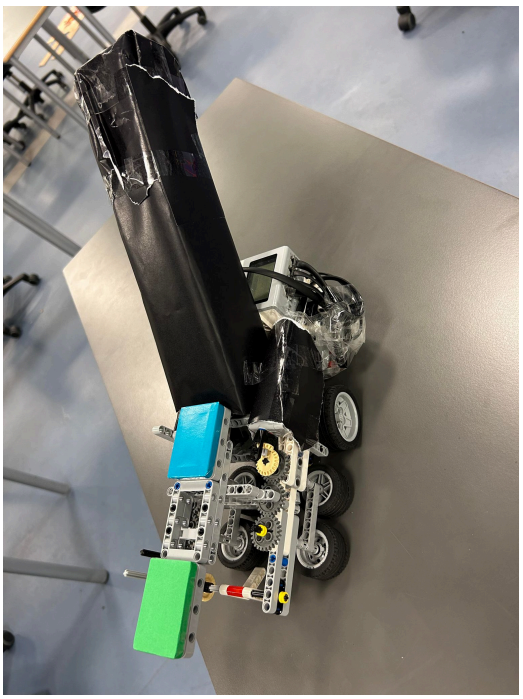




Lastly we added some black mat color paper on to the storage and sides of the robot to make sure that it will not be mistaken by any other objects on the image recognition.

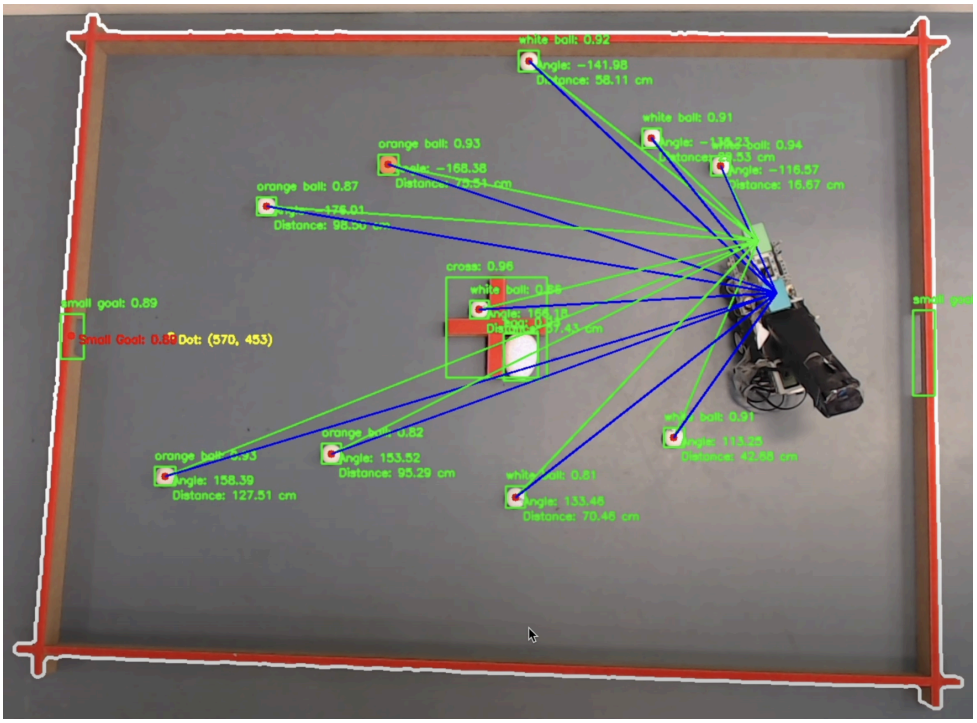






Software

In this part we will talk about the software we have used. We will look at some of the code that we have written and get an insight on why we have made the decisions we have in the code.



In the above picture, our camera is shown detecting all the objects on the course. We used OpenCV (CV2) for detecting green, blue, and wall elements. For the rest, we utilized the YOLO model. The steps we took involved annotating photos of the course, the objects, and various scenarios in which the ball and other objects could be situated. These scenarios enable the machine learning model to recognize and understand what is happening on the course, such as identifying which ball is orange, which is white, or what the egg is on the course.

```
30  def preprocess_frame(frame):
31      hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
32      h, s, v = cv2.split(hsv)
33      v = cv2.equalizeHist(v)
34      hsv = cv2.merge([h, s, v])
35      return hsv
```

The “preprocess_frame” function takes an image and enhances it. It first converts the image from BGR to HSV color space. Then it equalizes the histogram of the Value “(V)” channel to improve contrast. Finally it merges the channels back together and returns the processed HSV image.

So essentially we use this code to clean up any noise that might be present.

```
37  def detect_green(frame):
38      hsv = preprocess_frame(frame)
39      lower_green = np.array([20, 50, 100])  # Adjusted lower bound
40      upper_green = np.array([90, 255, 255])  # Adjusted upper bound
41      mask = cv2.inRange(hsv, lower_green, upper_green)
42      contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
43      if contours:
44          largest_contour = max(contours, key=cv2.contourArea)
45          x, y, w, h = cv2.boundingRect(largest_contour)
46          return (x + w // 2, y + h // 2)
47      return None
```

The function first processes the input frame by converting it to HSV color space using the “preprocess_frame” function. It then defines the HSV range for the color green with lower and upper bounds. Using these bounds what it does is, it creates a mask that isolates the green areas in the image. It finds contours within this mask and checks if any are found. If contours are present it identifies the largest one and calculates its bounding rectangle, returning the center coordinates of this rectangle. If no contours are found then the function returns as seen in the code “None”.

One might think, what would happen to the colors if the lighting changes during the day? For that exact reason we created another file named “findcolor.py” where we use it to find the color when we click on an element on the screen.

```
122    # Function to find the closest ball to the green object
123    def find_closest_ball(green_center, balls_centers, conversion_factor):
124        if not green_center or not balls_centers:
125            return None, float('inf')
126        closest_ball = None
127        min_distance = float('inf')
128        for ball_center in balls_centers:
129            observed_distance_px = np.sqrt((green_center[0] - ball_center[0]) ** 2 + (green_center[1] - ball_center[1]) ** 2)
130            real_distance = calculate_real_distance(observed_distance_px, conversion_factor)
131            if real_distance < min_distance:
132                min_distance = real_distance
133                closest_ball = ball_center
134        return closest_ball, min_distance
```

So here the function checks if the green center or balls' centers are empty and returns “None” and infinity if it is as so. What happens is it initializes variables for the closest ball and the minimum distance as infinity. For each ball center it calculates the observed distance in pixels between the green center and the ball center. Then what happens is it converts this distance to the real distance using a conversion factor. If the real distance is smaller than the current minimum distance, then it updates the closest ball and the minimum distance. In the end it returns the closest ball's center and the minimum distance.

```
167    # Function to generate a dot for the small goal
168    def generate_small_goal_dot(small_goal_center, conversion_factor):
169        if small_goal_center and conversion_factor:
170            angle = 0  # Assuming the dot is directly in front of the goal, you may need to adjust this angle as per your setup
171            distance_px = int(20 / conversion_factor)
172            small_goal_dot = (
173                small_goal_center[0] + distance_px,
174                small_goal_center[1]
175            )
176            return small_goal_dot
177        return None
```

What happens in the code is that the function first checks if the “small_goal_center” and “conversion_factor” are provided and if they are provided it assumes an angle of 0 degrees which indicates that the dot is directly in front of the goal. It calculates the distance in pixels and it does that by dividing 20 units by the conversion factor. Then it computes the coordinates of the small goal dot by adding this distance to the x-coordinate of the small goal center while keeping the y-coordinate the same. It

returns the coordinates of this small goal dot and If the required parameters are not provided it returns “None”.

This was basically our idea on how to score the goal as effectively as possible. It worked for us, there are other ways to do this, but we went with the one we thought was best for us. Another thing that would be noteworthy to mention is, that we used the same kind of idea for our balls close up on walls, a dot is placed on the wall right behind the ball and using the dot the robot is able to pick up the ball.

Testing

To make sure that requirements are met and to validate the performance of our system, we completed different types of testing for the system which can be seen below:

	Test Cases for Image Recognition	Expected Results	Results	Fail/Pass
2	Colors (Front & Back)	The camera must recognize the two colors on the robot.	The colors are recognized.	Pass
3	White balls	The camera must recognize the white balls.	The white balls are recognized.	Pass
4	Orange ball	The camera must recognize the orange ball.	The orange ball is recognized.	Pass
5	Different light settings	The camera should recognize all objects in different light settings.	The camera had trouble detecting all objects in different light settings.	Fail
6	White Egg	The camera must recognize the White Egg.	The White egg is recognized.	Pass
7	Walls	The camera must recognize the walls.	The walls are recognized.	Pass
8	Cross	The camera must recognize the cross.	The cross is recognized.	Pass
9	Different ball placements	The camera must recognize balls in different placements.	The balls are recognized in different placements.	Pass
10	Goals (Big & Small)	The camera must recognize the Big and small goal.	The Big and small goals are recognized.	Pass
11	Moving Ball	The camera must recognize and update robot if a ball moves location.	Balls which move location are recognized and updated to the robot.	Pass
12	**Test Cases for Demonstration**			
13	8-Minute testing, battery test	The EV3 Brick battery must hold out for atleast 8 minutes.	The EV3 Brick battery lasts minimum of 8 minutes.	Pass
14	Pickup Orange Ball First	The Robot should pickup the Orange ball first.	The Robot does not pick up the orange ball first.	Fail
15	Pickup White balls	The Robot must pickup atleast some white balls.	The Robot picks up some white balls.	Pass
16	Avoid Cross obstacle	The Robot should avoid the cross obstacle.	The Robot avoids the cross obstacle.	Pass
17	Avoid Egg obstacle	The Robot should avoid the Egg obstacle.	The Robot avoids the Egg obstacle.	Pass
18	Avoid Walls hit	The Robot should avoid hitting the walls.	The Robot avoids hitting the walls.	Pass
19	Score balls through one of the goals	The Robot must score balls through either small or big goal.	The Robot scores balls through either small or big goal.	Pass
20	Store balls	The Robot must store balls when collected.	The Robot stores balls when they are collected.	Pass
21	Navigate a defined area	The Robot should navigate around the defined course.	The Robot navigates around the defined course.	Pass
22	Release balls from storage	The Robot must release balls from storage.	The Robot releases balls from storage.	Pass
23	Request to server for instructions	The Robot must send requests to the server for instructions.	The Robot sends requests to the server for instructions.	Pass

These test cases have been done multiple times to ensure reliability, functionality and performance of the system. It’s important to notice that even if the test has passed, bugs and issues can still occur but are more likely not going to happen, after testing multiple times.

Conclusion

Reflecting on our work, though challenging, we managed to successfully complete our project. In the end the must have requirements of the project were met, which in itself speaks of accomplishment. The so called accomplishment wouldn't have been possible without our well thought out design and implementation of the ev3 golfbot, which was mainly the main focus of the project. Despite the hard work put into the project, the competition day did not go as planned. While we did manage to pick up a few balls, we were not satisfied with how it all went down, and felt we could do a whole lot better.

We managed to get our most important requirements implemented, and as indicated in our MoSCoW model, our must have requirements are fulfilled. We also did get most of our should have requirements as well, and we are quite satisfied with those accomplishments as well.

Our project was not made in a traditional way, using some of the other famous approaches on a project of this magnitude. By using the iterations, we managed to not only get a lot of feedback on our project along the way, but also embrace an agile way of thinking, always building and improving on our approaches. It is also for this reason that we have as many different prototypes as we do.

We are not doing justice by the course by only naming a few aspects, but given space constraints, we highlight the following. We have gained a tremendous amount of experience in this course. Working with the CDIO framework, and coding with the python language that we were not as familiar with, and using libraries such as OpenCV for our image recognition, we also made use of roboflow website for the machine learning, which gave us the model for our image recognition. It is without a doubt that the experience on project planning, team management and risk management all will gain us tremendously in our future careers as software developers.

Appendix

Appendix A: Work Distribution

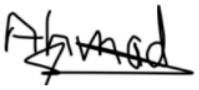
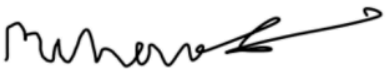
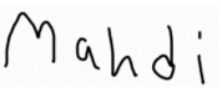
The table below documents how much each group member has contributed to each subtask in the project work and report writing.

Subtasks	Faruk Emir Degirmenci	Mohamad El-Asadi	Ahsunalla Wahidi	Muhamed Sbeihi	Mahdi Ibrahimi	Ahmad El-Hag
Server & Client connection	40%			30 %		30%
Function: Collect balls and store		35%	30%		35%	
Function: Release balls in goal	35%	35%			30%	
Function: Collect Nearest ball			35%	35%		30%
Function: Avoid Cross & Walls		30%	30%			40%
Function: Avoid white Egg	33,33%			33,33%	33,33%	
Course navigation / Distance & Angles	15%	15%	25%	15%	15%	15%
Image Recognition	16,66%	16,66%	16,66%	16,66%	16,66%	16,66%
Hardware						
Gear mechanism		30%			35%	35%
Storage / Suction	30%		35%	35%		
Spinner mechanism			30%		35%	35%
Tests	30%	35%		35%		
Status Reports		35%	30%		35%	
Status Videos	35%			35%		30%

Report Writing	Faruk Emir Degirmenci	Mohamad El-Asadi	Ahsunalla Wahidi	Muhamed Sbeihi	Mahdi Ibrahimi	Ahmad El-Hag
Introductio n			50%	25%		25%
Status Over Project And Product		33,33%	33,33%	33,33%		
Overall project health	15%				70%	15%
Proces	50%			42%	8%	
Product	5%			40%		55%
Design		62%	38%			
Implementa tion	25%		25%		25%	25%
Testing	75%	25%				
Conclusion			25%	25%	50%	
Appendix		50%				50%

Appendix B: Team Signatures

APPROVALS (signatures) :

Person	**Signature**
Ahmad Mosbah El-Hag Ali	
Muhamed Sbeihi	
Ahsunalla Wahidi	
Mahdi Ibrahimi	
Mohamad El-Asadi	
Faruk Emir Degirmenci