RoboCupJunior Soccer 2024

Design Document

LNX Robots

1 Abstract

We are a team of students from Gymnázium Grosslingová 18 high school, Bratislava, Slovakia. This design document highlights the biggest improvements to our robots since last year and should be used as a general guide which led to our design choices and the way we went around implementing solutions. We mainly focused on our hardware stepping up a notch so it can keep up with the best robots worldwide and making faster and smarter software design, which is able to play on the field without getting stuck in awkward situations which resort in lack of progress.

Since we are the first generation of this team, we did not have any materials to use from previous competitors, so we had to buy everything new with the money our sponsors gave us. We average around 7000 € on both of our robots combined and production cost with all the failures. We stay in touch via Discord, so we know who is working on what at the moment and we also use git to work on software.

2 Hardware

2.1 Overview

Last year we faced a lot of hardship in a world championship and got bullied by most of the working robots. Since we lacked a good drive system and camera view of the whole field, we were slower at localizing the ball and even when we localized it first, we still were second at the ball due to our slow and imprecise drive system. This year we decided to combat these issues and improve our drive system and employ a multidirectional camera while also keeping the front camera for lots of its benefits. Last year we lacked a dribbler and kicker – we had to also add those.

2.2 Specifics

2.2.1 Motor control

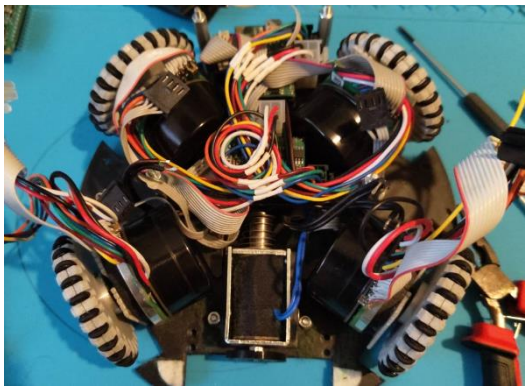


Figure 1: Undercarriage

We used to use Pololu 3203 brushed motors; however, these proved to be insufficient, so we looked for other teams help to pick our motors. For this years version of our robots, we found out most teams use motors which got handed down to them and have little to no knowledge of them; or are Joinmax motors which are no longer purchasable, at least from Europe to our concern. We decided to read through multiple TDPs from the major SSL league, which has the same diameter limit as our league, and saw that most of them were using brushless Maxon 45 Flat 70W motors direct drive (also team from Brisbane college used them in soccer open and they recommended them to us). This years design uses 4 Maxon 45 Flat 70W motors all in direct drive configuration and are controlled by Escon 24/2 motor drivers which ensure precise speeds of our motors.

2.2.2 Dribbler

Last year we were unpleasantly surprised by the number of teams with a working dribbler in Bordeaux. We could only find 2 teams with properly working dribblers (from which one had to reapply double sided tape every match). We saw that all of the dribblers in soccer open were top mounted (meaning having a rotating axis above the ball), however, when we looked at the SSL robots, we realized that every team has a bottom mounted dribbler (meaning having a rotating axis below the ball, most of the time as low as possible)[1]. We started wondering why this could be and after drawing out what happens with the ball as is entering both types of dribblers.

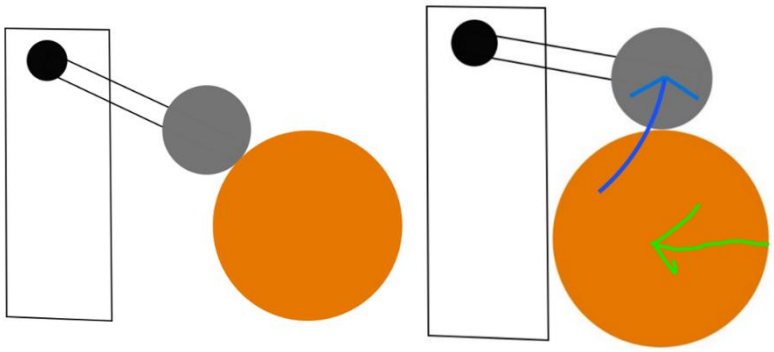


Figure 2: Top mounted dribbler (rough sketch)

[1] There is also the simplest version with stationary dribbler. However, this one seems to be barely better than no dribbler, since the ball needs to be in one exact spot, so the robot can actually apply a backspin on the ball

The problem with top mounted dribblers is that for them to get to “captured” state they firstly need to push the ball away from the robot and only after that only capture it (by this they are risking getting stuck in a loop of always pushing the ball away and never really capturing it).

Bottom mounted dribblers however don't have this disadvantage since as the ball moves inside the robot to get into the captured position (position where the ball is maximum 15mm inside the robot), the dribbler moves with it while maintaining contact with the ball.

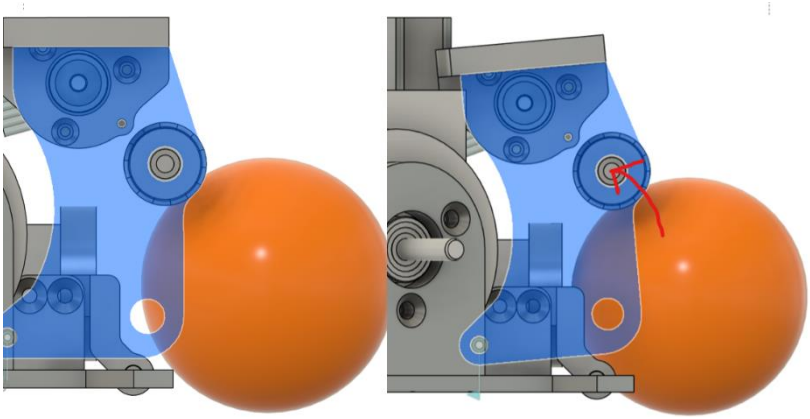


Figure 3: Bottom mounted dribbler

The difference was obvious, and we realized top mounted dribblers are normalized in soccer open just because everybody was doing them this way without thinking about why it should be made that way.

We use a spiral silicone bar for a surprisingly good ball centering at high enough dribbling speeds and helps with rotating and lateral movements with the ball. We 3D printed a mold and then poured silicone inside it to get silicone molded into the shape we want. We use A60 shore of the silicone for increased robustness. We yet need to compare the A40 silicone to A60 to get a final result of which one is better.

2.2.2.1 Dribbler dumping

After making 5 versions of dribblers (mainly just trying to make it as compact as possible) we ended with one powered by brushless Maxon EC-max 22. We then tried experimenting with a counter force on the dribbler, since we thought if there could be a force pushing the dribbler back down, we should have even better control over the ball in situations when the ball starts to move away from the dribbler. We tried multiple variations of these dumpers but all were proven to be counter productive on 8 out of 9 types of golf balls with the one we however got incredible ball control, which sparks hopes for more testing in the future.

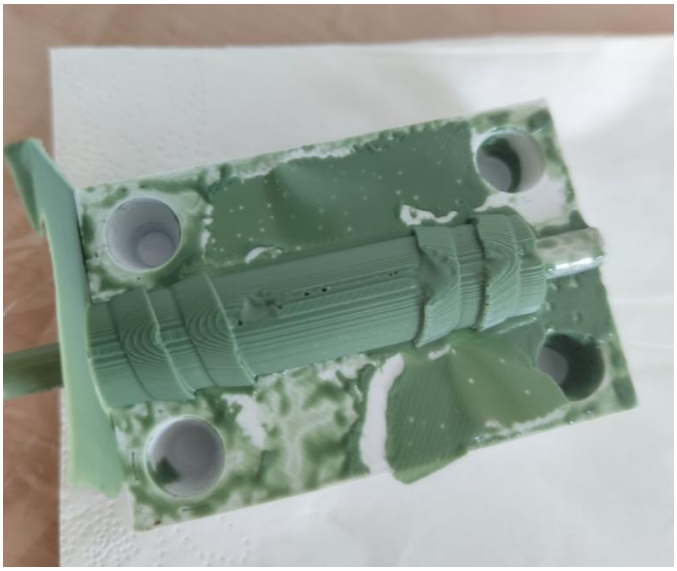


Figure 4: Dribbler silicone bar inside the mold

Figure 5: Dribbler dumping

2.2.3 Kicker

For a kicker, we used Tremba HMF-2620 solenoid. 48 V from onboard step-up convertor was provided, instead of the nominal 12 V, for increased strength. This solenoid is suboptimal and we plan to wind our own in future.

We found out that with every volt we fed to the solenoid we were able to exponentially increase the length which the ball traveled. What also made a huge difference was the distance between the ball and the end of our kicking surface. We noted that at 1mm distance the ball was able to travel around 240cm on a carpet while with 3mm distance the ball was able to travel only around 180cm.

Figure 6: Kicker testing

We got even better results when the ball was directly touching the kicking surface; however, we were unable to successfully keep the ball with our dribbler in direct contact with kicking surface, possibly because of high friction and thus had to resort to a small gap between our kicking surface and the ball.

2.2.4 Omnidirectional Mirror

From the beginning, we wanted to make the mirror as perfect as possible. Based on simple geometry, we developed an ordinary differential equation that was numerically computed so that all lengths in the virtual image are directly proportional with the actual lengths on the floor level.

This method was tested in Blender, which we used as a simulation environment for the cameras to ensure that they capture just the right view.

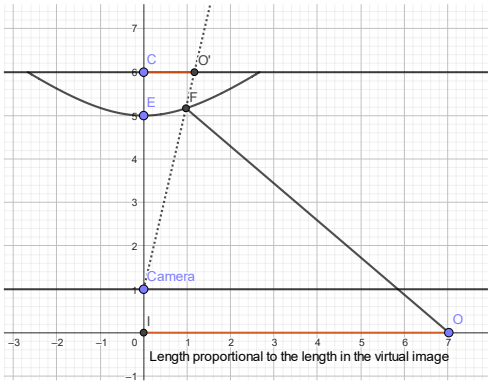


Figure 7 : Mirror diagram

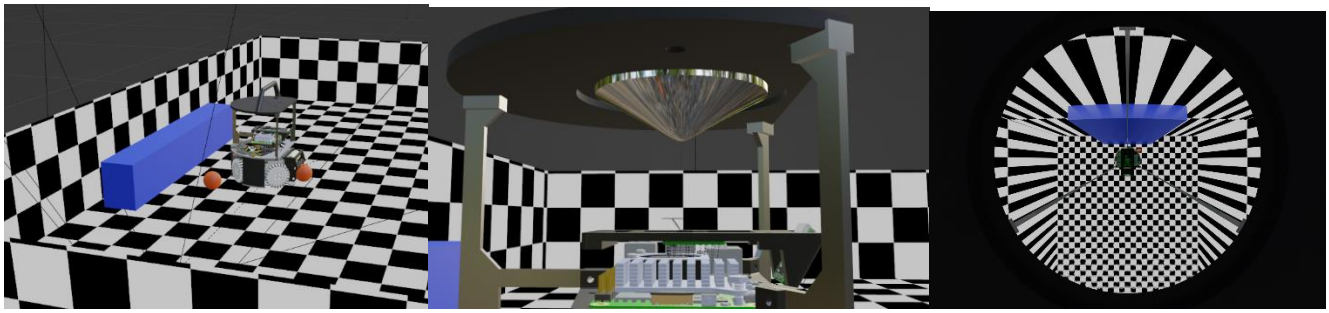


Figure 8: Blender simulation

As for the manufacturing of the mirror, we tried various methods. The first was thermoforming laminated polystyrene using vacuum[2].

This method showed potential, but was insufficient due to the laminate not fully fit the mold because of the low quality self-made tools. Our next step was to outsource the manufacturing process using the machine in a local workshop. This approach seemed better, but the image was still quite deformed.

Figure 9: Thermoforming at home

We then tried to order CNC-machined mirror on pcbway.com, but the image was still deformed. Our current hypothesis is that the calculation and simulation did not include the camera lens distortion. The fix for this deformation was later implemented in software.

[2] https://volzo.de/posts/rapid-mirrors/

2.2.5 Front Camera

First tested camera – Raspberry Pi Camera Module 2 with IMX219 drop-in camera sensor

- Unusual purple haze was visible all around the edges of the image
- Customized the IMX219 tuning file
    - Removed values from *Automatic lens shading correction* parameter to revert it to the default state.
- Result was a green haze in the center of the picture
    - Better – it does not interfere very much with colors intended to detect

Second tested camera – Raspberry Pi Camera Module 3 Wide

- Newer, supposed to be better in everything
- Autofocus
    - Turned out to be a problem
    - Vibration caused by motion shook the lens
    - Worse than the first one

Third tested camera – Arducam B0310

- Same sensor as the second camera
- Manual focus
- No problems

2.2.6 Mirror Camera

First tested camera – Arducam B0310

- Good experience
- 16:9 – too much unused space

Second tested camera – Arducam B0262 with Arducam 90 degree wide angle lens

- Same sensor size
- 4:3
- A little too small depth of field – not focused when ball is farther

2.2.7 Chassis

We designed the key parts of our chassis so that they could be easily 3D printed and eventually replaced by aluminum parts. In the first version, the all the grey parts of the robot (Figure 10) were made out of aluminum, except for the undercarriage, which was 3D printed. This undercarriage turned out to be a major weakness – it repeatedly broke during our matches on RoboCup Junior Europe 2024. The second version has fully aluminum undercarriage.

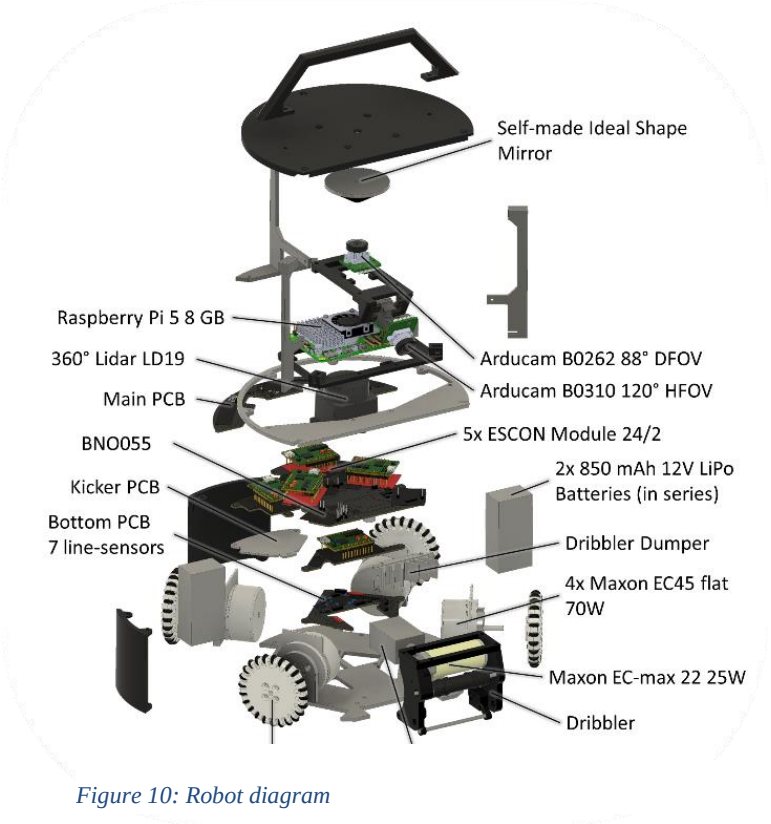


Figure 10: Robot diagram

Figure 11: Aluminum undercarriage

All aluminum parts were laser cut and ordered online. Some of them were later glued together, in some we made holes and threads for screws. We also added PTFE circles to protect our motors from axial impacts.

3 Software

3.1 Raspberry Pi

Our main program runs on Raspberry Pi 5. It's architecture is based on multiprocessing. Program is divided into multiple processes which can utilize all 4 cores of Raspberry Pi and achieve significant performance boost in comparison with single core application.

Each IO component of the robot (for example: camera, compass, ...), which is later referred to as **interface**, runs in the separate process. This process is launched through wrapper which is called **module**. Module is the class where you can define function which will be executed on the separate process and shared variables.

Communication between modules is done through shared memory.

The biggest advantage of this architecture is that the end user (someone who would program strategy) does not have to deal with anything connected with multiprocessing. If any data is needed, getter on the module can be called to obtain them.

Camera module, Lidar module, Bluetooth module, Logger module, UI module, Undercarriage module, visualizer module

3.1.1 Camera module

Camera Module takes care of processing of frames from camera. OpenCV is used to detect specific colors in the image. There are multiple steps in process of color detection.

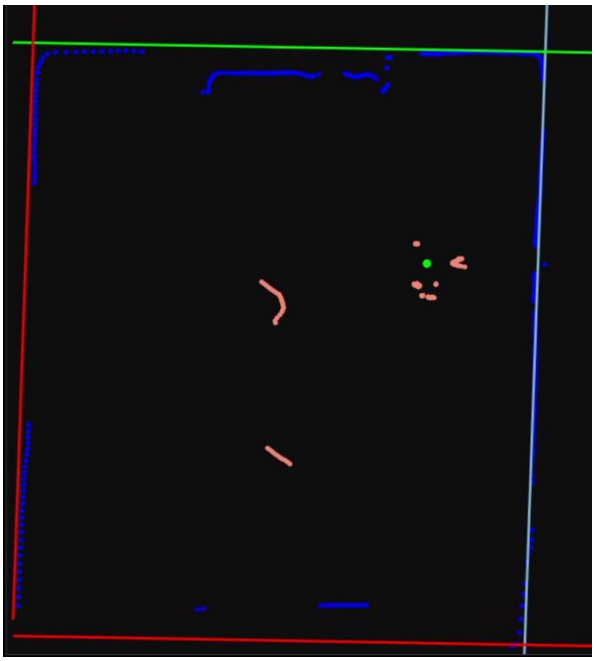
1. Raw frame is read from Camera
2. Raw frame is cropped so only part of the image which contains the playfield is kept
3. Cropped frame is converted from RGB to HSV
4. OpenCV is used to find contours in the HSV frame
5. Bounding box is created around the longest contour

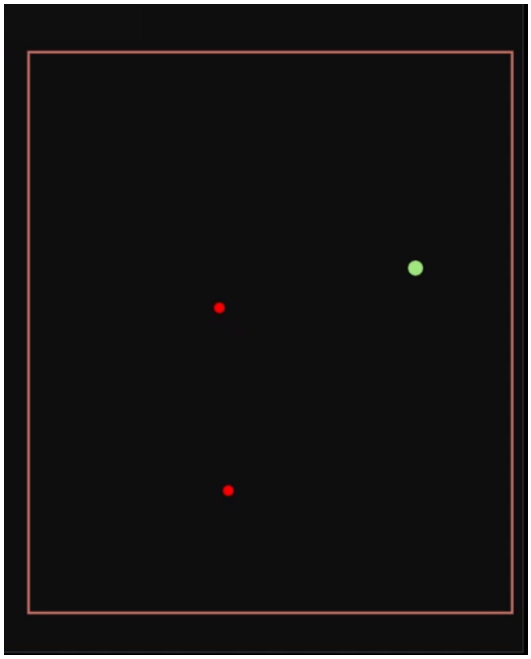
Mirror Camera Module proceeds the same way, except it detects a rotated bounding box of the ball and outputs the coordinates of the nearest edge (bottom of the ball).

3.1.2 Lidar module

On of our biggest improvements this year was addition of the lidar. Data from the lidar requires a lot of processing to be able to extract playfield position from it.

Our algorithm is based on finding lines, which contain the highest number of points, in lidar point cloud. For this purpose, we use Hough transform. The line with the highest number of points we call **primary line (**blue line in the image**)**. After finding the primary line we find a line perpendicular to this line. This line is called **orthogonal line** (green line in the image). Since we had some problems with our algorithm detecting goal as a wall, we shift both lines back to match the wall correctly. After this, parallel lines to primary and orthogonal lines are found in the correct distance from them, according to the playfield dimensions (red lines in the image). These 4 lines form perimeter of the playfield.





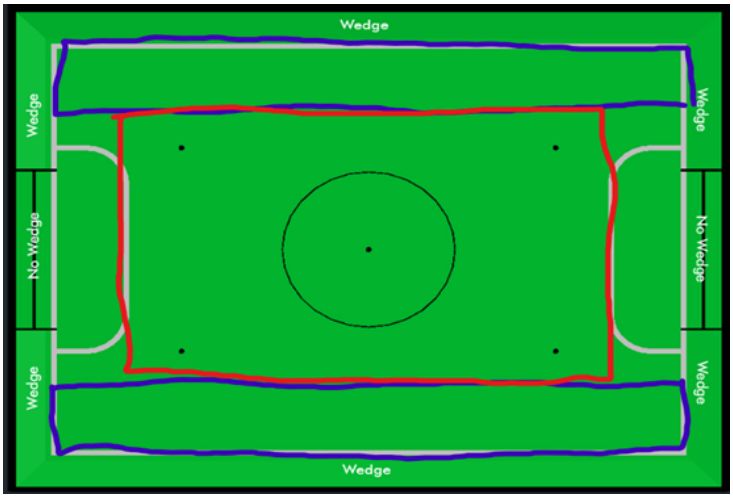
Then we can identify points, which belong to the playfield walls and other points are then sorted to groups and evaluated as other robots on the field.

3.2 Strategy

3.2.1 Attacker

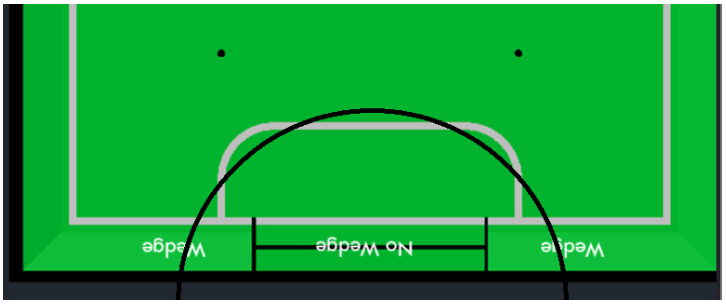
Attacker's behavior is divided into two main parts (zones).

In the red zone the robot tries to get around the ball and get it from the north. This option is the safest because an enemy does not have any chance to push the ball behind us. In the blue zones however, the ball cannot be gathered from the north because the is not enough space and is therefore gathered directly and the robot performs a spin with it afterwards.



After successfully collecting the ball, the robot rotates to the goal and shoots it. If the goal cannot be seen with the camera the robot uses the lidar and shoots to the position where the goal should be located.

3.2.2 Defender



The defender keeps his position on an imaginary circle formed near our own goal. More specifically the robot goes to the intersection of the circle with a line connecting the ball and the middle of the goal. This solution has an advantage that the robot also blocks the goal when the ball is in the corner. If the robot does not see the ball, enemy position is used instead of the ball position.

The robot has also the ability to shoot the ball to the opposite side of the playfield to reduce the risk of getting a goal.

3.3 Visualizer

To evaluate performance, we made a visualizer in C++, which is communicating with our program through websockets. Data from the robot are displayed there. There is also an option to calibrate camera detection, line detection and motor speeds. Having this type of debugging tool helped us tweaking our robots to make them play just right.