RobôCIn Small Size League
Extended Team Description Paper for RoboCup 2025

Alberto Franca, Beatriz Barros, Davi C. Barbosa, Davi Dubeux, Driele Xavier, Elisson R. S. Araújo, Felipe N. A. Pereira, João G. Melo, José R. Silva, Lucas H. Cavalcanti, Marcela Asfora, Mateus Albuquerque, Matheus Alves, Matheus Paixão, Matheus Vasconcelos, Onias C. B. Silveira, Vinícius Dutra, Victor Araújo, and Edna Barros

Centro de Informática, Universidade Federal de Pernambuco.
Av. Prof. Moraes Rego, 1235 - Cidade Universitária, Recife - Pernambuco, Brazil.
`robocin@cin.ufpe.br`
`https://robocin.com.br/`

Abstract. RobôCIn has participated in RoboCup Small Size League since 2019, won its first world title in 2022 and second in 2023 (Division B), and is currently a five-time Latin-American and Brazilian champion. In 2024, we made our debut in Division A in RoboCup 2024 in Eindhoven, Netherlands, achieving an excellent first campaign by finishing at fourth place and receiving the Most Improved Team Award. This paper presents our improvements to maintain a high level of performance in Division A at RoboCup 2025 in Salvador, Brazil. During 2024, our team has successfully published two papers related to SSL at the 27th RoboCup International Symposium. Over the past year, we have sought to improve some robot control and communication failure points. Furthermore, we will discuss some software experiments with distributed architecture and improvements in ball interception.

Keywords: RobôCIn · RoboCup 2024 · Robotics · Small Size League

1 Hardware

To ensure robustness and consistent performance in competition, our robot underwent a redesign to ensure robustness and consistent performance in competition. The most significant improvements were made to the direct drive system and the electronic boards. In this section, we present the characteristics of the v2024 robot, and in the following ones, we will discuss improvements focused on 2025.

The structure of the v2024 robot consists of several key components, each playing a crucial role in its functionality. The robot modules are shown in Figure 1.

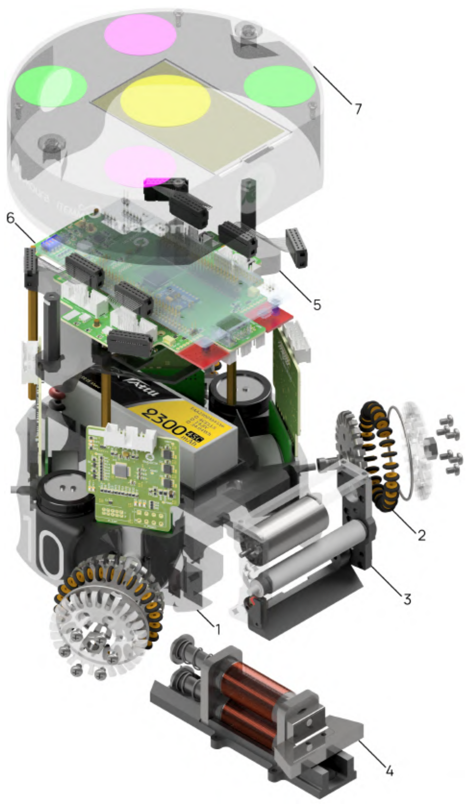


Fig. 1. Robot's exploded view, with all modules.

1 - **Base frame**, which serves as the core structure where all components are mounted.
2 - **Direct drive system**, including a BLDC motor with its control board, adapter shaft, and omniwheel.
3 - **Dribble mechanism**, composed of a dribble bar, BLDC motor, IR set, damping mechanism and chip kick.
4 - **Kicker assembly**, featuring front and chip kicker mechanisms.
5 - **Second floor**, a 3D-printed structure that supports the main board and the kicker board of the robot.
6 - **Main Board**, which includes the voltage regulator and the signal distribution board of the STM32F7 microcontroller.
7 - **3D printed cover**, completing the entire assembly.

The v2024 model retains the direct drive system introduced last year [6], eliminating gear-related issues for smoother, more precise movement. Key upgrades include switching from a fully printed PLA wheel to an aluminum inner wheel component for better precision and durability. The diameter of the wheel increased from 50mm to 60mm and the number of rollers increased from 18 to 26.

A new motor and a new control system improve efficiency by balancing energy use and motor lifespan, while the gearless design improves responsiveness and speed. The electronic boards were redesigned to fix past errors, improve assembly, and integrate a new RCPT connector for easier soldering and component integration. Most component changes were minor, with the focus on parameter adjustments and current limitations. General hardware specifications and comparison with our later v2022 version can be found in Table 1.

Table 1. Robot specification comparison between v2022 and v2024

Robot version	**v2022**	**v2024**
Driving Motors	Maxon EC-45 flat - 50W	Maxon EC-45 flat - 50W
Max % ball coverage	19.55%	17.91%
Microcontroller	STM32F767ZIT6	STM32F767ZIT6
Gear Transmission	18:60	1:1
Gear Type	External Spur	Not used
Wheel	3D Printed	3D printed and aluminum
Total Weight	2.36kg	2.50kg
Dribbling Motor	Maxon EC-max 22, 25W	Not used
Encoder	MILE 1024 CPT	MILE 2048 CPT
Dribbling Gear	1:1:1	Not used
Dribbling Bar Diameter	13mm	13mm
Max. Kick Speed	6.5m/s	6.5m/s
Communication Link	nRF24L01+	nRF24L01+
Battery	LiPo 2200mAh 4S 35C	LiPo 2300mAh 4S 75C

1.1 Comparison between printed and machined wheels

Last year, we decided to switch to direct drive [6]. As part of this change, we modified the inner part of the wheel material, specifically the component responsible for ensuring contact between the motor shaft and the transmission component. In the v2023 version, we used a fully printed PLA wheel; in the v2024 version, we chose to change part A, as shown in Figure 2, to aluminum to increase the precision and resistance to stress.

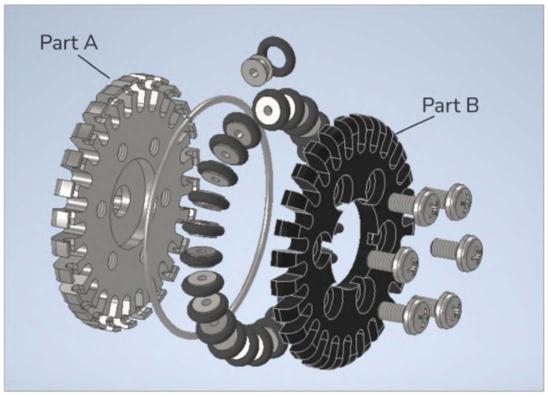


Fig. 2. Wheels exploded view.

During RoboCup 2024, we noticed that our robots exhibited increased resistance to movement. To better understand this behavior, we investigated whether the material change affected the motor response to the control action, potentially introducing mechanical resistance or impacting motion execution efficiency.

The test involved subjecting the same robot to a series of straight-line movements at different angles to determine whether the performance difference between machined and printed wheels remained consistent across various movement orientations. To validate our hypothesis, we collected PWM data sent to each wheel. We evaluated the desired control signal based on the command issued and the actual signal transmitted by the firmware to each motor.

Figure 3 illustrates the behavior of Robots 3 and 5 in the movement test at an angle of 45 °, using both fully 3D printed wheels and modified wheels (with part A machined and part B printed). Upon analyzing the collected data, we observed that for Robot 5 (with machined wheels), the control system did not require additional effort to reach the desired control signal. Therefore, we decided not to continue the investigation, based on the hypothesis that the wheel modification did not influence our perception in the last RoboCup.

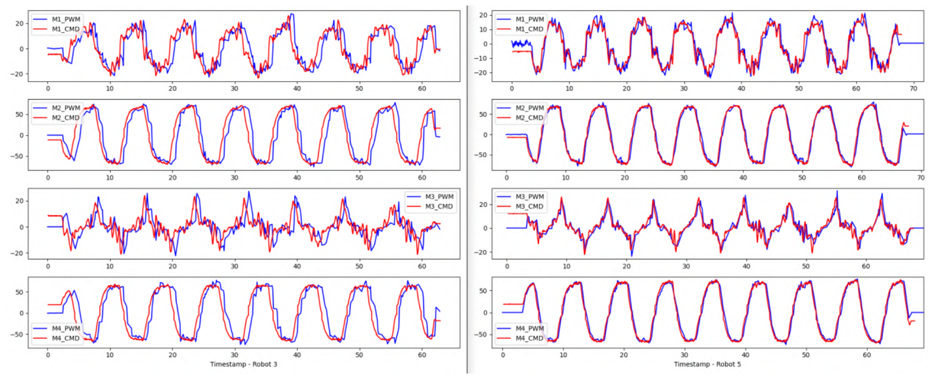


Fig. 3. Behavior of Robots 3 and 5 during movement tests at 45°, on the left and right, respectively.

2 Onboard Control

Up to RoboCup2024, our onboard control loop only had individual PID controllers for each wheel, on which our robot received local velocity commands, i.e., the desired translational and rotational velocities regarding the robot's reference frame. Then, the wheels' desired velocities were estimated using the robot's kinematics matrix and their controllers would make them achieve their desired speeds.

Even though the motors' controllers can work independently from the robot's global velocity at considerably high rates (in our case, 500 Hz), this approach limits the rate at which we can update the robot's desired speed up to the

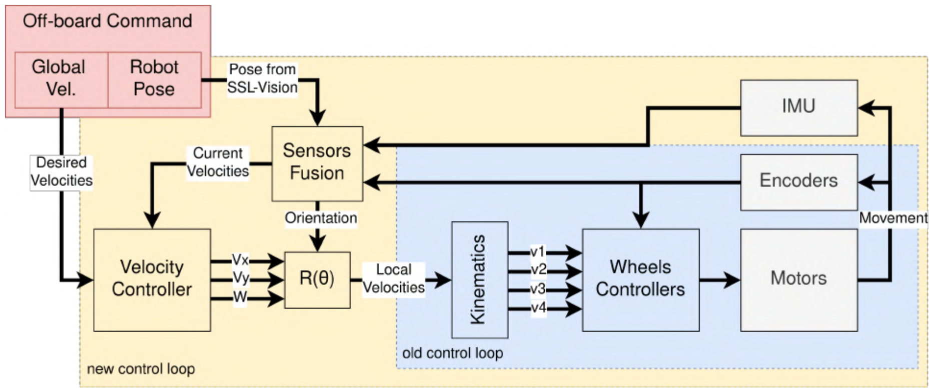


Fig. 4. High level diagram of previous and current onboard control loops.

camera's frame rate, which is usually 70 Hz during RoboCup SSL matches. Also, the information acquired by the SSL-Vision's camera has some delay until it reaches the robots' and wireless communication packets can be lost due to various reasons, which also affects the time required to update the robot's commands. Therefore, more layers of our control loop are necessary to introduce more layers of our control loop to improve the navigation of our robots, enabling them to have faster updates and, consequently, more precise movements [1].

We improved our architecture by changing our wireless communication commands now to send velocities regarding the field's reference frame. In addition, the robot now has a new control loop that allows it to maintain the global velocity commands desired by the IA off-board, as illustrated in Figure 4. For this, the packet sent to the robot was updated according to Table 2. The new packet includes a flag to explicit identify the frame of reference of the velocities in the packet (either global or local to the robot). In this way, the same packet can be used to test current and previous implementations interchangeably, simplifying testing.

Table 2. Comparison between previous and current communication packets.

Packet	Bytes [0:x]	Bits [101:118]	Bits [119:138]	Bits [139:158]	Bit [159]
Previous	(...)	X	X	X	X
New	(...)	angle	pos_x	pos_y	is_global_velocity

The implemented architecture is shown in Figure 4, where different stages of the implementation are shown. The inner block (in blue) highlights our past approach, which received local velocity commands and used them as references for controlling the wheels' speeds. The outer block contains the newly added features, where $R(\theta)$ is a rotation matrix that converts global velocities to the robot's coordinate frame, using the robot's current orientation θ. This orientation is updated by information from SSL-Vision and sent over the radio. Although, new packets do not arrive, the angle is estimated using gyroscope feedback using a simple integrator ($\hat{\theta}_{k+1} = \hat{\theta}_k + \omega_k \cdot T$). This way, the local velocity keeps being updated even though the global desired is not, due to the angle in the rotation matrix being continuously updated.

3 Communication System

For the v2025 robot, we implemented a significant upgrade to the communication system, replacing the nRF24 radio modules with Wi-Fi operating in the 5GHz band. This transition was driven by challenges faced in previous competitions and the technical advantages offered by the new approach, which ensures greater reliability and efficiency in the communication system.

3.1 Communication System up to v2024

In the communication system used until v2024, we used the Nordic nRF24L01+, a transceiver chip operating in the ISM 2.4GHz band. The communication subsystem introduced in the robot v2019 consistently demonstrated stability and served as a reliable foundation for our optimized communication protocol [3] and our old communication system. However, as competitive scenarios became more demanding, certain limitations of the system became increasingly evident.

– **Communication Rate:** The communication rate of the v2024 system was measured at 12 ms per packet in experimental environments, as shown in Figure 5. Although this rate was sufficient for most scenarios, it proved inadequate in situations requiring higher system efficiency, such as in the larger field used in Division A. This issue was also addressed by Kordas et al. [11], who demonstrated that increasing the number of bytes per packet or increasing the distance between transceivers leads to a higher packet loss rate.
– **Packet Size Limitation:** One of the main limitations of the nRF24L01+ system was the restriction on packet size, capped at 32 bytes per transmission. Our team used 13 bytes per robot, as illustrated in Table 3, which means that each packet could contain information for only one robot at a time. This limitation required a high degree of data compression, reducing the richness and granularity of the transmitted information.
– **Interference and Packet Loss:** Interference caused by other devices operating at the same frequency is a common scenario in competition venues. This interference resulted in high packet loss rates, directly affecting the performance and reliability of the communication system. Fig. 5 also highlights this issue, showing that some packets took more than 100 ms to reach our robot.

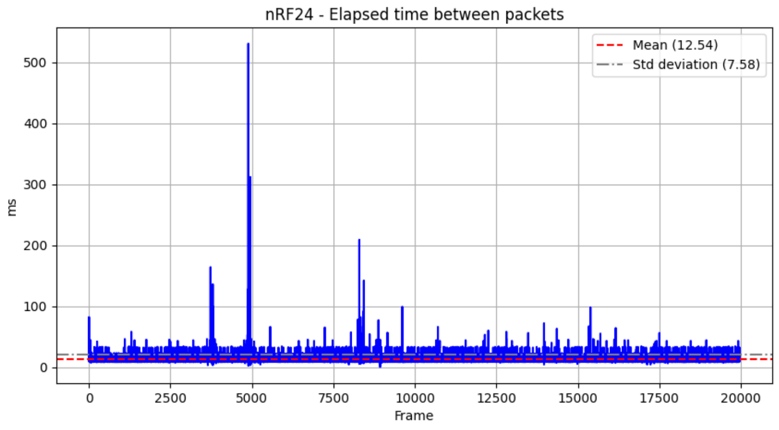


Fig. 5. Elapsed time between packets using nRF24L01+.

Table 3. nRF24L01+ packet structure and its fields.

MsgType	***RobotID***	***Vx***	***Vy***	***W***	***FrontKick***	***ChipKick***	***ChargeKick***	***KickStrength***	***Dribbler***	***Speed***	***Command***	***isLocked***	***isTurbo***	***Free***	***TOTAL***
4 b	4 b	20 b	20 b	20 b	1 b	1 b	1 b	8 b	1 b	11 b	8 b	1 b	1 b	3 b	13 Bytes

3.2 Migration to Wi-Fi

For the **v2025** version of the robot, we have replaced the communication through `nRF24L01+` with Wi-Fi in the 5GHz band. This decision was driven by several factors, including the obsolescence of the current module, interference issues, packet size limitations, and the need for a more robust and scalable communication system.

Although more modern Nordic modules, such as the `nRF51` and `nRF52` [12] series, were viable alternatives, they still operate in the 2.4GHz range, which experiences high congestion in competition environments due to extensive use by other teams and leagues, Wi-Fi networks, Bluetooth devices, and industrial equipment. The migration to Wi-Fi 5GHz helps to mitigate this issue and brings several improvements, such as:

Reduced Interference and Increased Stability: Wi-Fi in the 5GHz band has more channels available than our previous `nRF24L01+`, significantly reducing congestion and packet collisions and ensuring a more stable and reliable communication system [4].

Higher Transmission Rate and Low Latency: By transitioning to Wi-Fi, we achieved a significant increase in data rates, allowing faster updates of commands sent to robots.

More Efficient Bidirectional Communication: With Wi-Fi, we implemented a high capacity bidirectional communication channel, enabling high frequency telemetry reception and improved debugging and monitoring during matches.

Adoption of a Standard Used by Other Teams: The migration to Wi-Fi also follows a trend observed in several league teams, such as **UBC Thunderbots** [5], **GreenTea** [21], **ZJUNlict** [23], **RoboJackets** [7], and **Immortals** [20]. All of these teams already use this technology with good performance and reliable results. Adopting a well-established standard facilitates future knowledge exchange and technical comparisons with other teams.

Better Scalability and Compatibility with Future Technologies: Using Wi-Fi opens up new possibilities for integration with other technologies, such as:

- **Use of more sophisticated protocols**, such as `UDP multicast`, can be used to optimize packet transmission to multiple robots.
- **Integration with distributed systems**, enabling remote testing and advanced data analysis.
- **Ease of implementing redundancy systems**, ensuring greater reliability of communication.

Our approach was defined after conducting a test with a second team during the Brazilian Robotics Competition of 2024 [2], which provided valuable insights into the advantages and challenges of adopting Wi-Fi for communication. Based on the results, we refined our system to ensure greater efficiency, reliability, and scalability.

We used a Raspberry Pi 5 [18] as an intermediary between the central software and the robot microcontroller for communication implementation. Although in the CBR 2024, it was only used to relay packets, this choice was made with a future perspective where the Raspberry will also play an active role in embedded processing. In this way, we can decentralize the calculations, reducing the dependency on the central software and making the robots more autonomous.

Currently, communication is based on UDP multicast, where the Raspberry receives the packets sent by the central software and relays them to the microcontroller via Ethernet. Each robot, once connected, establishes a `UDP unicast` channel with the Raspberry, allowing it to filter the received data and send only the relevant information to each unit.

In the following, we detail the architecture of our communication, addressing its topology, packet structure, protocol, and connection management.

– **Network Topology:** We adopted a star topology for communication between the robots and the central, where each robot has bidirectional communication with the central. Figure 6 illustrates this configuration.

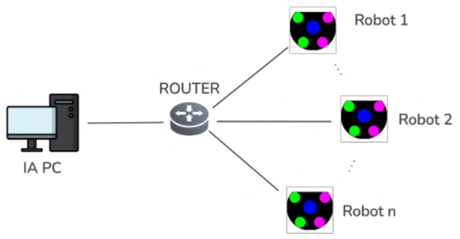
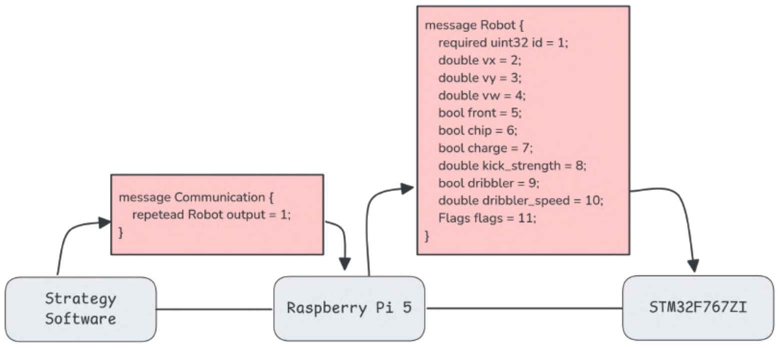


Fig. 6. Network overview.

– **Packet Structure and Data Encoding:** The packets have a similar structure to what we used previously, as illustrated in Table 3, but now we employ `Protocol Buffers` [8] for serialization. This choice allows for more efficient compression and facilitates interoperability of the system.
Since each packet contains the ID as a unique identifier, it enables the differentiation of data intended for each robot. We use a broadcast approach, where we encapsulate the packets of all robots into a single package to send, as shown in Figure 7, eliminating the need to send multiple packets. In this way, each robot receives only one package and accesses the data addressed to it using its ID.



```
message Communication {
   repetead Robot output = 1;
}
```

```
message Robot {
   required uint32 id = 1;
   double vx = 2;
   double vy = 3;
   double vw = 4;
   bool front = 5;
   bool chip = 6;
   bool charge = 7;
   double kick_strength = 8;
   bool dribbler = 9;
   double dribbler_speed = 10;
   Flags flags = 11;
}
```

Fig. 7. Diagram of packet transmission.

- **Communication Protocol:** The Raspberry Pi maintains a `UDP unicast` connection with the robot's microcontroller, as seen in Figure 8, established right at the start of operation. During this process, the microcontroller sends the robot's ID, which the Raspberry uses to extract only the data corresponding to it from the received multicast packets.

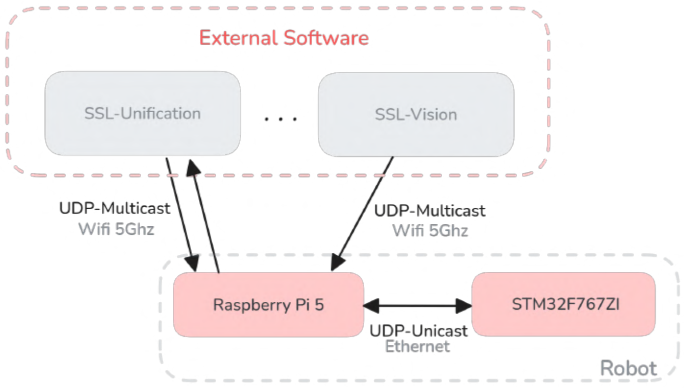


Fig. 8. Communication system diagram.

The transition from `nRF24` to **Wi-Fi 5GHz** represents a significant improvement for our communication system, providing greater stability, lower latency, richer data packets and better scalability. In addition to mitigating the issues faced in previous competition editions, this change prepares our team for future advancements, ensuring a more modern and efficient communication system. Figure 9 shows a comparison of the time interval for receiving packets between our old and new communication systems.

4 Containerized Microservices Architecture

Last year, we began experimenting with a distributed architecture for the software. RobôCIn is a robotics team that competes in five different categories: Small Size League (SSL) [16], Very Small Size Soccer (VSSS) [10], Simulation 2D [19], Drones [14], and more recently @Work [17]. As a result, the team maintains multiple distinct codebases and architectures, each specifically designed to meet the technical requirements and constraints of its respective domain. While this specialization has many advantages, such as easier development, reliable performance and simplified testing and debugging, which led to spaghetti code, increased coupling, and, as we developed without care for modularity and re-usability, it also led to software maintenance, collaboration, and scalability issues.

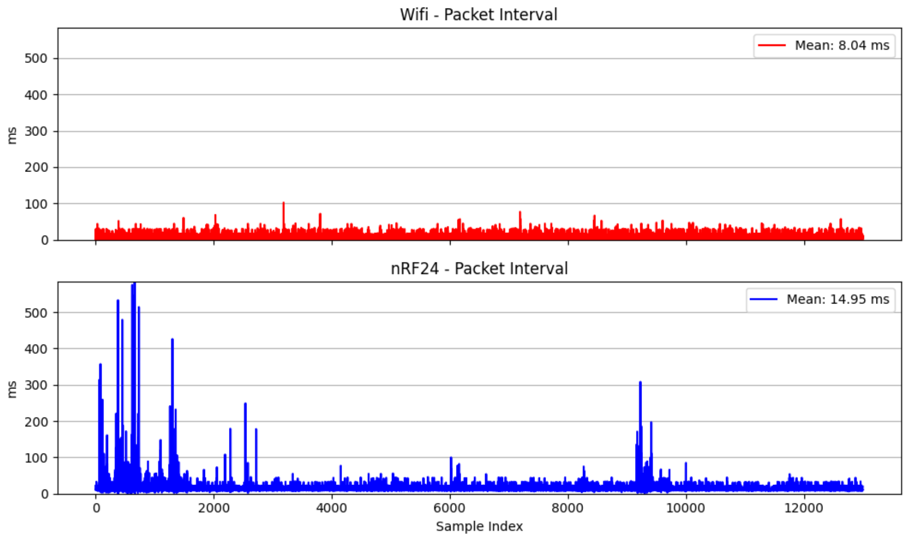


Fig. 9. Comparison of packet interval using nRF24 and Wifi (5Ghz)

To address these challenges, the team is currently experimenting with solutions aimed towards distributed and containerized computing, aiming to facilitate the maintenance of crucial components of the software and reduce the cognitive load onto developers from our current solution, while preserving the flexibility needed for category-specific adaptations and leveraging self-managed systems properties [9].

To assess the feasibility of this new architecture, the team conducted a practical validation by implementing a prototype leveraging an external open-source containerized and distributed solution targeted towards SSL named SSL Core [22] and tested it during RoboCup Brasil 2024 [15] competition. Using this prototype, we evaluate developer quality of life, new modules schemas, system resilience, and reliability in a real competition environment. Through this testing, the team identified several issues that need to be addressed, as well as positive aspects that demonstrated the potential benefits of the approach. The insights gained from this evaluation provide a foundation for refining the containerized service-oriented approach, guiding future improvements in software design and overall system performance for our team.

4.1 Containerized distributed services

The goal of developing a foundational infrastructure that meets the needs of RobôCIn's categories makes flexibility essential. In this context, the microservices-based architecture of SSL Core [22] stands out as a key factor in its selection. Its structure enables component decoupling, simplifying development, maintenance, and expansion. The microservices communicate with each other through UDP packets, and using Protocol Buffers [8] to create the messages used in this communication ensures interoperability between systems and opens possibilities for using different technologies. For instance, it is possible to have a computer vision service in Python that communicates with a decision-making service in C++. Additionally, SSL Core is compatible with multiple platforms without operating system version restrictions. For these reasons, SSL Core [22] was chosen as the foundational infrastructure for the prototype development.

4.2 Leveraging the existing infrastructure

To conduct the validation, the team leveraged existing SSL Core [22] capabilities, built on top of that a prototype of a strategy pipeline, and tested it in a real competition environment. The existing infrastructure was already capable of receiving and processing data from the league software through its Gateway, Vision, and Referee services. Also, it had an interface to render the field using data sent from the Playback service. These already existing services are the green hexagons in Figure 10. The yellow rectangles represent the services we developed on this basis, namely:

- **Decision:** responsible for strategizing the team's plays. It is different in the figure because its development was not completed and integrated;
- **Behavior:** focuses on defining the individual actions of the robots based on the strategy;
- **Navigation:** transforms each action into its corresponding movement;
- **Communication:** sends and receives data to and from the robots;

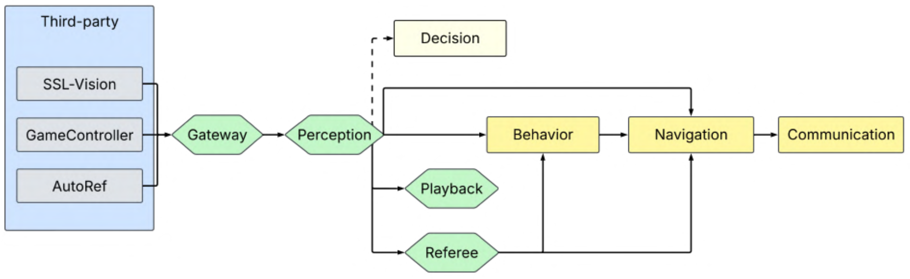


Fig. 10. Services and flow of information.

This structure is very similar to the one of our currently employed SSL-Unification software [13], allowing for a meaningful comparison between the monolithic and services-oriented implementations.

We noted that development was greatly simplified due to a lesser coupling and higher cohesion inherent in service-oriented approaches, improving developer quality of life and reducing strategy development time. Also, it was a lot easier to integrate new features and services, something that would be difficult in a monolithic architecture. At runtime, we did not find any noticeable performance issues due to containerization nor high latency on the network and robots could be controlled in the field without noticeable delays, although we had network instability when communicating with the SSL simulator. The main compromises were noticed on wrong and inexistent business logics, most noticeably in Command messages which were incorrectly parsed from SSL Core; and also on the inexistence of a control dashboard to alter parameters, which is something our currently monolithic solution highly depends on. Additionally we lacked observability of the containers, hindering our ability to detect and debug crashes in the services when they occurred.

Migrating to a distributed solution has known problems such as higher coordination complexity, data synchronization and more points of failure that together contribute to a higher maintenance cost and complexity, but from our experience at RoboCup Brasil 2024 the benefits outweigh the detected costs and we expect to keep working on this solution to harness the positive sides into further competitions and experiments.

5 Ball Interception

We developed an algorithm to estimate the best point where the robots can intercept the ball's movement. In SSL, the games are very dynamic, with both the ball and robots moving quickly across the field. Therefore, our game strategy is based on advancing the ball through passes toward the opponent's goal, making quick and precise receptions crucial elements for the success of the plays.

Opposing teams constantly attempt to intercept passes, making it crucial to find passing positions that minimize this risk and improve receptions. Our approach follows two key principles: first, the robot must reach the estimated point before the ball; second, it should seek to intercept the ball as quickly as possible, considering the actions the robot must perform when intercepting the ball, such as redirecting it to another location. This ensures efficient ball reception, maintaining possession and play continuity.

5.1 Limitations of the previous heuristic

The previous heuristic was based solely on Torricelli's equation to determine the point in the ball's trajectory where it would reach a specific, fixed velocity. Thus, a fixed velocity was set for the robot to receive the ball, and Torricelli's equation calculated the necessary displacement until the ball reached the defined speed. The position where the ball would reach this velocity was then calculated based on the displacement and the ball's initial position, and this position was passed to the teammate responsible for receiving the ball. However, this approach had several issues due to its rigidity and inability to adapt to different scenarios:

- **Lack of synchronization between ball and robot:** Since the destination point was calculated without considering the robot's arrival time, the ball often reached the point before the robot, leading to a loss of possession.
- **Interception at distant points:** Often, the robot could intercept the ball earlier, but the fixed velocity pushed the interception point too far, allowing opponents to intervene.
- **Oscillations in the destination point:** Vision imprecision caused speed fluctuations, making the fixed-velocity interception point unstable and disrupting ball reception.

Due to these issues, it was extremely difficult to find a single desired velocity value that was good enough for all the cases where the ball position estimation was used. Furthermore, the previous heuristic did not allow for the inclusion of additional conditions that could improve the destination point estimate, as it was solely reliant on the ball's speed.

5.2 Ball Interception Estimation with the Binary Search Algorithm

In its traditional use, the binary search algorithm finds an item in an ordered list by repeatedly dividing the list in half until the possible locations are narrowed down to a single point. For the interception context, the algorithm was adapted to search for the best point on the ball's trajectory. The "best point" is the location where the robot can intercept the ball as early as possible while adhering to any existing constraints. Figure 11 illustrates this process.

In Figure 11, the algorithm treats the trajectory of the ball's movement as a sequence of time-ordered points. It performs a binary search, adjusting the interval based on whether the robot or the ball arrives first. If the robot can reach the point before the ball, as demonstrated in the first iteration, the search moves to an interval closer to the ball; otherwise, it continues in the farther interval. Similarly, if the ball reaches the point before the robot, as in iterations 2 and 4, the search moves to an interval further away from the ball. Through this iterative process, the algorithm can identify the point where the robot arrives at the ball's movement trajectory as early as possible.

However, if the algorithm proceeds only in this manner, the robot will reach the point simultaneously with the ball, colliding with it instead of intercepting or redirecting it. To prevent this, the algorithm was adjusted to find the first point where the robot reaches the ball's trajectory slightly earlier. This ensures the robot has sufficient time to intercept the ball's movement properly. Moreover, the algorithm avoids disruptions from ball velocity oscillations caused by vision imprecision by not relying on a fixed velocity to determine the best intercept point. As a result, the destination point remains stable even if the ball reaches it at varying speeds across different frames.

Another significant improvement that enhances stability in the estimation is the algorithm's consideration of the ally robot's position when calculating the interception point along the ball's trajectory. As the ally moves closer to the trajectory with each new frame, the time required for the ally to reach the

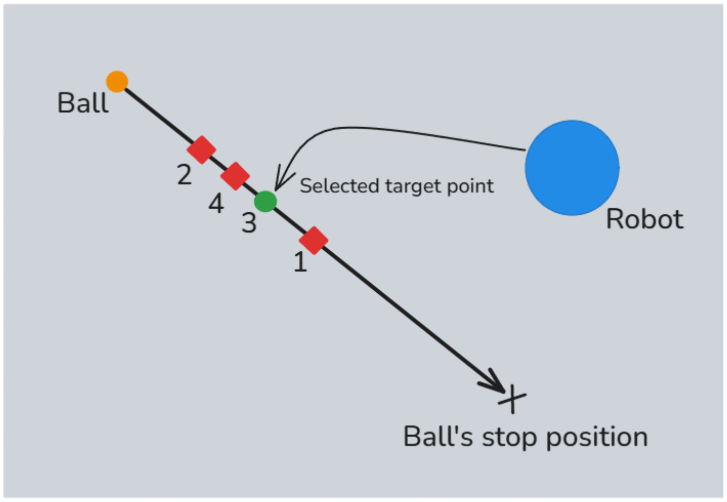


Fig. 11. Representation of the binary search algorithm adaptation to find the best point on the ball's trajectory. The number below each dot on the trajectory represents the corresponding iteration of the algorithm. The red diamonds are points that don't satisfy the stop condition for the algorithm and the green dot is the best point.

optimal point decreases. Consequently, this time remains shorter than the ball's arrival time, ensuring that the estimated interception position stays consistent.

5.3 Constraints and improvements

Some constraints had to be applied to ensure the algorithm operated correctly and avoided failures. The first was limiting the search to points up to where the ball reaches zero velocity, preventing errors from searching for points outside the valid range.

Another challenge arises when the robot is near or already at the destination point, and the ball is approaching quickly. In these cases, the remaining time for the ball to reach the point may be less than the predefined time margin. As a result, the algorithm shifts the intercept point too far, causing the robot to waste time in unnecessary movement and miss the opportunity to intercept the ball earlier.

To address this situation, our approach ensures that the robot continues toward the destination point when it is sufficiently close to it, even if the remaining time for the ball to reach that point becomes shorter than the defined margin. This avoids unnecessary repositioning and ensures a quicker interception.

6 Acknowledgement

First, we would like to thank our advisors and the Centro de Informática (CIn)-UFPE for all the support and knowledge during these years of project and development. We also would like to thank all our sponsors: Moura Batteries, ITEMM Institute, Microsoft, CESAR, Neurotech, Incognia, HSBS, Embraer, Instituto Nacional de Engenharia de Software (INES), Maxon Motors, Mathworks, Tractian and STMicroelectronics.

References

1. Abbenseth, J., Ommer, N.: Position control of an omnidirectional mobile robot (2015)
2. Araujo, E., Paixao, M., Andrade, M., Xavier, D., Barros, E.: Robocin - team b small size league team description paper for robocup brazil 2024 (2024), https://drive.google.com/file/d/1OX0kFTnVls2UYJj0omZMfCTG1nCGSHve/view?usp=drive_link, roboCup Small Size League, Goiania, Brazil, 2024
3. Cavalcanti, L., Joaquim, R., Barros, E.: Optimized wireless control and telemetry network for mobile soccer robots. In: Robot World Cup. Springer (2021)
4. Cisco Systems, I.: 802.11ac: The fifth generation of wi-fi. https://davidhoglund.typepad.com/files/white_paper_c11-713103.pdf (2013), white Paper
5. Dumitro, P., Ellis, G., Fink, J., Hers, B., Lew, J., MacDougall, M., Morcom, E., Sawiuk, H., Sousa, C., Van Dam, W., Whyte, G., Zhang, L., Zheng, S., Zhou, Y.: 2020 team description paper: Ubc thunderbots (2020), robocup Small Size League, 2020
6. Franca, A., Barros, B., Gomes, C., Silva, C., Alves, C., Barbosa, D.C., Xavier, D., Ara´ujo, E., Pereira, F., Batista, H.A., Cavalcanti, L.H., Asfora, M., Alves, M., Paix˜ao, M., Vasconcelos, M., Vin´icius, M., Melo, J.G., Silva, J.R., Cruz, J.V., Leite, J., Santana, P.H., Oliveira, P.P., Rodrigues, R., Morais, R., Teobaldo, T., Dutra, V., Ara´ujo, V., Barros, E.: Robocin small size league extended team description paper for robocup 2024 (2024), https://ssl.robocup.org/wp-content/uploads/2024/04/2024_ETDP_RoboCIn.pdf, roboCup Small Size League, Eindhoven, Nederland, 2024
7. Fu, K., Sohrab, A., Nayak, P., Das, M., Doma, I., Perez, B.: Robojackets 2023 team description paper (2023), roboCup Small Size League, Bordeaux, France, 2023
8. Google Inc.: Protocol buffers (2025), https://developers.google.com/protocol-buffers, acessado em: 03 fevereiro 2025
9. Huebscher, M.C., McCann, J.A.: A survey of autonomic computing—degrees, models, and applications. ACM Comput. Surv. **40**(3) (Aug 2008). https://doi.org/10.1145/1380584.1380585, https://doi.org/10.1145/1380584.1380585
10. IEEE: Very small size soccer, https://ieeevss.github.io/vss/index.html
11. Jan Kordas, Petr Wagner, J.K.: Wireless transceiver for control of mobile embedded devices. In: International Multiconference on Computer Science and Information Technology (2010)
12. Nordic: nrf52 series socs, https://www.nordicsemi.com/Products/nRF24-series#:~:text=support-,nRF52%20Series%20SoCs,-Recommended%20for%20new

13. Oliveira, A., Gomes, C., Silva, C., Alves, C., Souza, D., Xavier, D., Silva, E., Martins, F., Cavalcanti, L., Maciel, L., Paixão, M., Vasconcelos, M., Vinícius, M., Melo, J.G., Moura, J.P., Silva, J.R., Cruz, J.V., Santana, P.H., Oliveira, P.P., Rodrigues, R., Fernandes, R., Morais, R., Teobaldo, T., Silva, W., Barros, E.: Robôcin extended team description paper for robocup 2023 (2023), robocup Small Size League, Recife, Brazil, 2023
14. Organization, F.R.L.: Flying robots league wiki, `https://www.robocup.org.br/wiki/doku.php?id=flying`
15. Organization, R.B.: Robocup brasil 2024 website, `https://robotica.robocup.org.br/`
16. Organization, R.S.S.L.: Robocup small size league organization website, `https://ssl.robocup.org/`
17. Organization, R.W.: Robocup @work organization website, `https://atwork.robocup.org/`
18. Pi, R.: Raspberry pi 5: The everything computer. optimised., `https://www.raspberrypi.com/products/raspberry-pi-5/`
19. RoboCup: Simulation 2d, `https://ssim.robocup.org/`
20. Salehi, A., Shirazi, M.M., Tabasi, M., Najafi, O., Nobaveh, A., Fazeli, M., Ghasemieh, M.A., Niknezhad, M.R., Talaeezadeh, M.: Immortals 2024 extended team descriptionpaper (2024), roboCup Small Size League, Eindhoven, Nederland, 2024
21. SATO, H., OKAMOTO, N., ITO3, A., KAYAKI4, S., NaoyaSUGISHITA5, NAKAAKI6, S., NAKAO7, T., NISHIMURA1, Y., HARA8, Y., FUJITA9, K., Yuri, R.: Greentea 2023 team description paper (2023), roboCup Small Size League, Bordeaux, France, 2023
22. SSL-core: Ssl-core, `https://github.com/ssl-core/ssl-core`
23. Zhao, A., Yu, P., Huang, Z., Shen, N., Yang, J., Yu, J., Chen, Z., Wang, L., Xiong, R.: Zjunlict extended team description paper (2024), roboCup Small Size League, Eindhoven, Nederland, 2024