GreenTea 2026 Team Description Paper

Naoyuki OKAMOTO[2], Akito ITO[1], Kohei FUJITA[1], Rintaro YURI, Taiyo SATO[2], Yuto HARA, Shun KAYAKI[3], Tomoki NAKAO[2], Haruka YAMAMOTO[1], Shoma NAKAAKI[1], Yuki NISHIMURA[1], and Hirotaka SATO[1]

[1] The Open University of Japan
[2] Graduate School of Informatics, Kyoto University
[3] Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology

contact@greentea-ssl.com
https://greentea-ssl.com/

Abstract. GreenTea was founded in 2019 by students who had participated in RoboCup Junior. To address issues in our robot system identified since 2021, we developed brand-new hardware, electrical circuits, and AI. This TDP evaluates the issues the existing system has faced and outlines the new system. The new system realizes high controllability through rearranged wheels and a newly installed suspension, a compact and power-efficient local vision system, high responsiveness of the kicker and close-range ball position estimation via ball sensors, and an AI system with high maintainability and ease of development.

Keywords: RoboCup, small size league

1 Introduction

GreenTea is an SSL team formed in 2019, primarily by students who participated in RoboCup Junior in Japan. This team brings together individuals from diverse backgrounds and affiliations, consisting of roughly half working professionals and half students. Thanks to the diversity of the members, each member contributes unique knowledge and experiences in various fields such as electrical and electronic engineering, mechanical engineering, and computer science. We have participated in Japan Open from 2022 to 2025. Also, we submitted TDPs for RoboCup 2023 and 2024, although we were unable to participate. Our existing robot system described in TDP2023[5] and 2024[6] was excellent but had many issues to be improved. In this TDP, we summarize the evaluation of the system, and describe the brand-new robot system we developed to overcome the previous issues.

2 Evaluation of our existing robot system

Our existing robot system is based on a robot platform "Sanran" and a game AI "MATCHA" developed in 2021 [5]. We have continuously expanded their functionality to enhance their competitive performance over the past five years. However, their outdated fundamental design has resulted in an extremely complex system architecture. As we undertake a complete overhaul of the robot system, we have evaluated the existing system.

2.1 Mechanical Design

Anisotropy of Omni-wheel Running Resistance The omni-wheel layout of our existing robot adopts an asymmetrical configuration as shown in Fig. 1. Fig. 2 empirically shows that driving performance in the lateral direction is inferior to that in the longitudinal direction with this layout. Such anisotropy in driving performance undermines the advantage of omnidirectional movement inherent to omni-wheels and makes optimal trajectory planning difficult. To quantitatively grasp this anisotropy, we measured the running resistance of the omni-wheels. We controlled the wheel velocities to move the robot at a constant speed and calculated the generated thrust force, i.e., the running resistance, from the motor currents. Assuming the kinematics of the omni-wheels can be expressed by the velocity transformation matrix $\boldsymbol{C}_v$, the thrust transformation matrix $\boldsymbol{C}_f$ and the thrust vector $[f_x\ f_y\ f_\theta]^T$ acting on the robot body can be derived from the principle of virtual work as follows:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \boldsymbol{C}_v \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix},\ \boldsymbol{C}_f = \boldsymbol{C}_v^T,\ \begin{bmatrix} f_x \\ f_y \\ f_\theta \end{bmatrix} = \boldsymbol{C}_f \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{1}$$

where v_i $(i = 1, 2, 3, 4)$ is the ground contact velocity of each wheel, v_x, v_y are the robot's translational velocities, ω is the rotational velocity, f_x, f_y are the translational thrusts, f_θ is the robot's moment, and f_i $(i = 1, 2, 3, 4)$ is the tangential force of each wheel. We drove the robot at a constant speed of 0.5 m/s and calculated the thrust vector $[f_x\ f_y\ f_\theta]^T$ from the steady-state current values and Eq. (1). The translational thrust $\sqrt{f_x^2 + f_y^2}$ was calculated as the estimated value of running resistance. We measured the running resistance over all directions (360°) in 45° increments. The plotted results are shown in Fig. 2. The running resistance in the lateral direction was found to be larger than in the longitudinal direction, confirming that the omni-wheel layout of the existing robot has a directional dependency on running resistance.

Maintainability of Omni-wheels The omni-wheel of the existing robot consists of 30 sub-wheels per wheel as shown in Fig. 3, and each sub-wheel is constructed from five small parts (tire, shaft, bush, etc.) as shown in Fig. 4 [5]. This

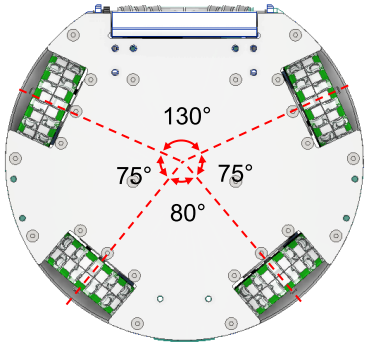


Fig. 1: Omni-wheel layout of the existing robot.

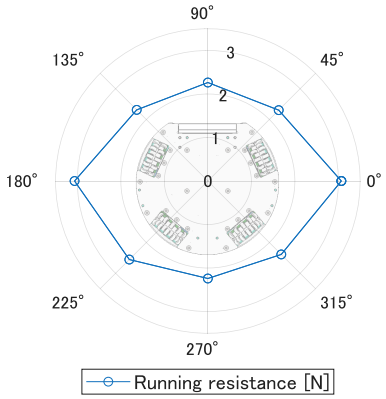


Fig. 2: Directional dependence of running resistance estimated from the motor current. The 0°–180° axis is defined as lateral, and the 90°–270° axis as longitudinal.

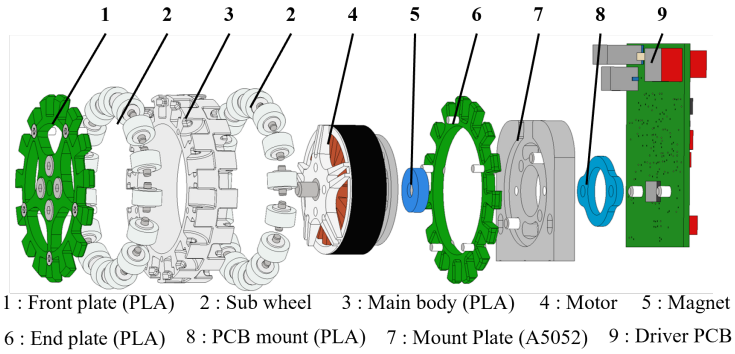


Fig. 3: Components of an existing omni wheel unit[5].

results in an extremely complex structure involving 150 parts per wheel by a simple calculation.

Fibers from the punch carpet laid on the competition field tend to intrude into the gaps of the sub-wheel rotation shafts during driving. Accumulation of these fibers causes increased rotational resistance and slippage. To resolve this, frequent disassembly and cleaning of the sub-wheels are necessary. However, these numerous tiny parts are easily lost during disassembly, and reassembly requires significant man-hours and precise work. This poor maintainability was a serious hindrance to robot maintenance within the limited time during competitions.

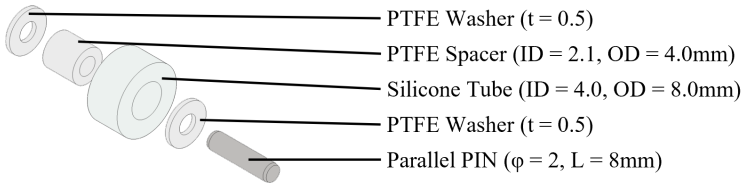


Fig. 4: Sub-wheel structure of an existing omni wheel[5].

2.2 Embedded Systems

Local Vision System Based on the initial design of the 5th generation robot "Sanran", we extended the body height to near the upper limit of regulations to place a wide-angle camera on the upper front of the body. This established a local vision system that captures a high-angle view of the ground in front of the robot, independent of SSL-Vision, implemented using image processing on a Raspberry Pi 4B [6].

However, operating the control system on Linux has revealed the following technical challenges:

- **Delayed Boot Time**: It takes a long time for the system to transition to an operational state after power-on, lacking the responsiveness required when a reboot is needed during a match.
- **Large PCB area requirement**: The increased power consumption of high-speed image processing requires a stable, high-current supply circuit, which results in a high PCB area occupation rate.
- **Data Robustness**: Accidental power loss during a match or shock from collisions poses a risk of data corruption on storage, making file system protection measures indispensable.

Issues with Ball Sensors and Position Estimation The existing robot adopted ball detection using only a single point via a photo-interrupter due to physical constraints of the dribble and kick mechanisms. This configuration could only detect the presence or absence of the ball and could not estimate the detailed holding position. We attempted to obtain accurate ball coordinates during dribbling using local vision, but we decided not to pursue this approach further due to degraded detection accuracy at close range and communication latency adversely affecting control. The ball speed after a kick strongly depends on the initial contact position of the ball relative to the kicker plate. However, the existing single-point detection method cannot provide information to correct this initial position deviation. Consequently, this lack of stability in kick speed has been a factor hindering the execution of precise passes and shots.

2.3 AI

Through the development of the software "MATCHA", we constructed a set of mechanisms capable of executing the tasks necessary for the Small Size League (SSL) and conducting a match. This established a basic operating environment for participating in competitions.

However, as development progressed and we gained practical experience, the following issues in implementation and design became apparent.

The first issue is the structural complexity arising from monolithic design. As shown in Fig. 5 (reprinted from [6]), MATCHA was implemented such that each game state (e.g., "Inplay", "Stop") corresponded to an independent monolithic ROS node. While this architecture conceptually adopted STP roles (e.g., Attacker, Defender) as Tactics, the implementation logic was tightly coupled within each state node. This resulted in a "God Object" anti-pattern where the "Inplay Node" became excessively large and opaque, while simultaneously preventing the reuse of basic behaviors (Skills) across different states.

The second issue is the lack of temporal coherence and inter-state continuity. The robot assignment algorithm, which operated based on instantaneous field states, caused frequent switching of controlled robots (chattering) during tasks like pass reception, disrupting action sequences. Additionally, the absence of a mechanism to propagate strategic context from set plays to the "Inplay" state constrained decision-making that requires long-term continuity.

Finally, the limitation of the perception and feedback loop. MATCHA was designed assuming complete top-down control and did not have an uplink from the robot side. Therefore, the AI side could not utilize high-speed information from the local vision system via onboard cameras and relied solely on commands based on SSL-Vision information. Also, although we referenced other teams' implementations for filtering SSL-Vision raw data, the accuracy did not reach the level required for our team's strategic decisions.

Regarding the development infrastructure, issues remained in terms of environment maintenance and future-proofing as follows: We adopted ROS Noetic assuming a native Ubuntu environment for the OS. However, the reliance on manual setup for the environment and external software (such as the GameController) compromised consistency across different machines. Furthermore, the lack of a mechanism to inject edge cases into the simulation hindered the efficiency of testing and debugging. Additionally, since ROS Noetic has reached its EOL (End of Life), migration to the next-generation standard, ROS 2, had become an architectural necessity.

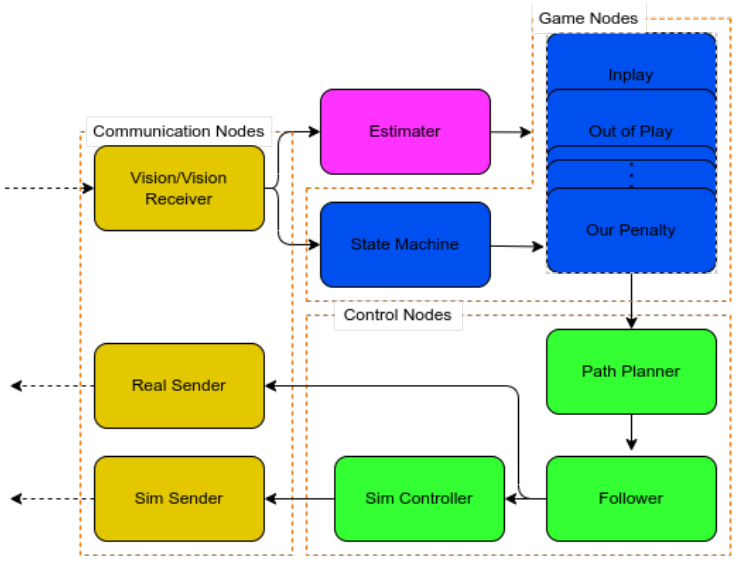


Fig. 5: Overview of the previous AI system "MATCHA". Reprinted from the GreenTea 2024 TDP [6].

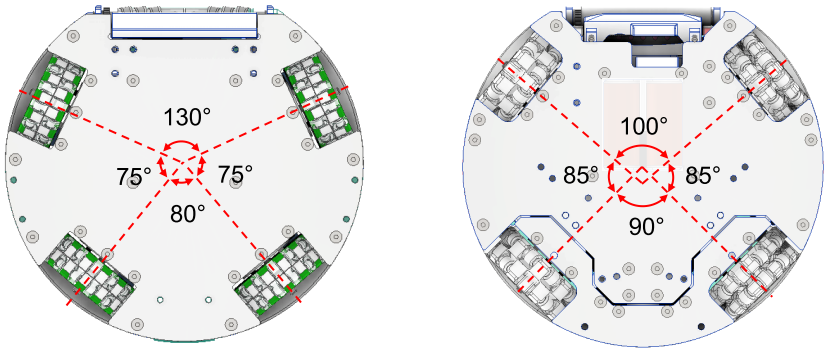
3 Design of the New Robot System

3.1 Mechanical Design

Optimization of Omni-wheel Layout To address the limitations of the existing robot, we adopted a near-symmetrical wheel arrangement approximating 90-degree intervals (Fig. 6b). This new configuration is expected to reduce kinematic anisotropy and improve lateral movement performance compared to the existing one.

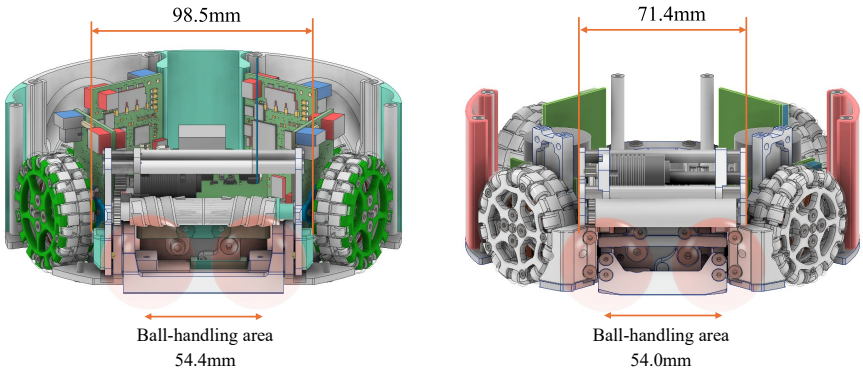
Miniaturization of Kicker and Dribbler With the change in omni-wheel layout, the area available for the dribbler and kicker at the front of the robot decreased as shown in Fig. 7. In the new robot, we redesigned the dribbler and kicker structures. By adopting the structure shown in Fig. 7b, we succeeded in maintaining a ball-handling area comparable to that of the existing robot, despite the reduced overall width of the mechanism.

Implementation of a Simplified Suspension Mechanism We introduced a slip suppression control framework for the omni-wheel drive mechanism to improve acceleration performance. This method assigns the maximum non-slipping thrust to each wheel based on the wheel's ground load calculated from the robot's center of gravity and acceleration. Since wheels with larger ground loads can be assigned larger thrusts, overall acceleration performance is expected to improve. However, standard four-wheel omni-wheel mechanisms have four points of contact, characterized by statically indeterminate ground loads. Although it can be solved by assuming contact stiffness, slight unevenness of the floor in a real environment can cause one wheel to float, creating a significant discrepancy between the calculated and actual ground loads, which was a challenge. In the newly developed robot, we installed a mechanism (simplified suspension mechanism)



(a) Existing robot. (b) New robot

Fig. 6: Comparison of Omni-wheel layouts between the existing and new robot.



(a) Existing robot. (b) New robot.

Fig. 7: Dimensions and ball-handling area of the kicker and dribbler.

that allows the rear wheel section of the omni-wheel drive to rotate independently as shown in Fig. 8, ensuring a structure where four points are always in contact with the ground. This allows the ground load of each omni-wheel to be uniquely determined. Specifically, it can be calculated by solving the equations Eqs. (2)–(5) related to ground load, which is less susceptible to the influence of ground unevenness and contact conditions.

$$f_{v1}p_{1x} + f_{v2}p_{2x} + f_{v3}p_{3x} + f_{v4}p_{4x} + M_y = 0 \tag{2}$$
$$f_{v1}p_{1y} + f_{v2}p_{2y} + f_{v3}p_{3y} + f_{v4}p_{4y} + M_x = 0 \tag{3}$$
$$f_{v1} + f_{v2} + f_{v3} + f_{v4} = mg \tag{4}$$
$$f_{v2} - f_{v3} = 0 \tag{5}$$

where $f_{v1}, \cdots, f_{v4}$ are the ground loads of each wheel, $p_{1x}, \cdots, p_{4x}$ are the X-coordinates of each wheel position, $p_{1y}, \cdots, p_{4y}$ are the Y-coordinates of each wheel position, M_y, M_x are the moments around the y-axis and x-axis, and mg is the gravity acting on the robot mass. In a standard four-wheel omni-wheel robot, only Eqs. (2)–(4) hold true, making the solution indeterminate. However, the simplified suspension mechanism equalizes the ground loads of the rear wheels (f_{v2}, f_{v3}), satisfying Eq. (5) and allowing the solution to be uniquely determined.

Improvement of Omni-wheel Productivity and Maintainability Fig. 9 shows a comparison of the structure of the newly developed sub-wheel to overcome the challenges of the existing type. The newly developed sub-wheel integrates the bearing part and the tire part through two-stage injection molding, reducing the number of parts and improving assemblability. Also, whereas the outer periphery of the existing sub-wheel used cut silicone tubing, resulting in a rectangular ground contact shape, the new sub-wheel achieves an arc-shaped ground contact profile by changing the manufacturing method to injection molding. This reduced vibration during driving and realized smooth travel.

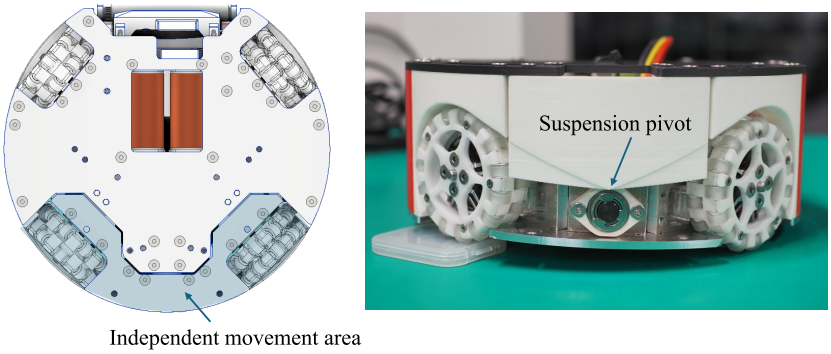
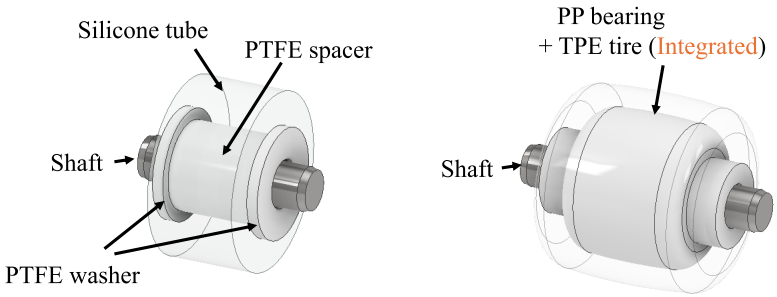


Fig. 8: Structure of the newly developed suspension mechanism.



(a) Existing sub-wheel structure. (b) New sub-wheel structure.

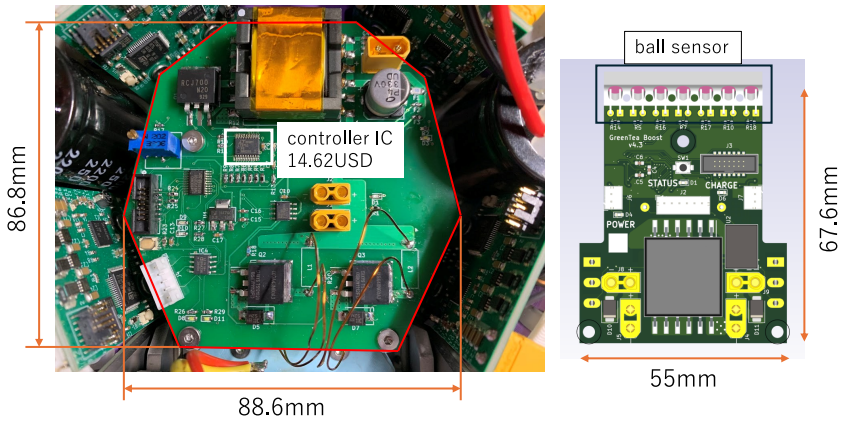
Fig. 9: Comparison of existing and new sub-wheels.

3.2 Embedded Systems

Local Vision By porting the ball detection method from the previous system to the M5Timer Camera F (ESP32-based), we achieved miniaturization, improved power efficiency, and faster boot times. The module size was reduced from 85 mm × 56 mm to 48 mm × 24 mm, and the power required for operation was reduced from 5 V/3 A to 5 V/1 A. Additionally, By adopting an OS-less configuration, the boot time was reduced from approximately one minute to less than one second.

Ball Sensor and Kicker Board In the existing kicker board, the control of the flyback converter to charge the solenoid drive capacitor was performed by a dedicated controller IC. In the new board, this control is implemented directly on the microcontroller,eliminating the need for a dedicated IC and resulting in a more compact and cost-effective design.

In addition, we mounted reflective ball sensors [4][2] on the same board and implemented a close-range ball position estimation. A comparison between the



(a) Previous kicker board. (b) New kicker board.

Fig. 10: Comparison between the existing and new kicker boards.

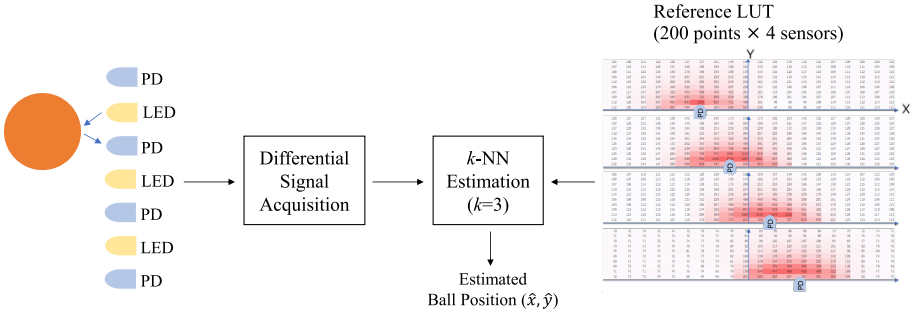


Fig. 11: Ball detection system.

existing board and the new board is shown in Fig. 10. The MOSFET that drives the primary side of the flyback converter transformer is driven by the PWM peripheral circuit of the STM32G431KBU3. The pulse width is calculated in real-time from the electrical parameters of the transformer and rectifier diode, the power supply voltage, and the output capacitor voltage detected based on the flyback converter circuit theory [7].

Next, the ball position estimation logic using the newly developed reflective ball sensor is shown in Fig. 11. The ball sensor consists of 3 infrared LEDs and 4 infrared phototransistors. The LEDs are driven synchronously, and the detection values of the phototransistors are taken as the difference between when the LEDs are ON and OFF to remove the influence of DC ambient light. The disturbance-removed detection values are compared with a pre-created lookup table, and the estimated ball position is obtained by the k-nearest neighbor method (k=3). The lookup table was created by manually placing a ball at 200 points set in a grid pattern in an area of 50 mm forward and 100 mm left and right relative to

the dribble position in front of the robot, and recording the detection values at that time.

Power Board We made the power board independent and equipped it with an ESP32 to allow independent control of the robot's logic power supply and power circuit supply. By implementing battery cell voltage detection and self-power-off, we realized over-discharge protection for the robot as a standalone unit. We also implemented a function to remotely stop the power supply in case of loss of control via a wireless control system independent of the robot's main control.

3.3 AI

Core Architecture Regarding the perception system, which receives information from SSL-Vision, AutoRef, and the Game Controller, top priority was given to achieving early system operation. Consequently, we tentatively adopted the filtering logic from Tigers AutoRef[8] for processing SSL-Vision data to acquire stable field information.

We designed a new AI system named "RIKYU" based on the STP architecture,which improves the independence of robot actions and strategic implementations.[1]. While the STP architecture provides high clarity, ambiguity remained regarding the separation of the group-control and individual-control layers during coordinated tasks. To address this, we proposed the introduction of a Multi-Tactics layer, which manages the tactics of a robot group (e.g., Line Defense) and evaluates their success, and a Solo-Tactics layer, which handles the tactics and success criteria for individual robots. This novel hierarchical separation enables complex coordinated control while maintaining high modularity—a feature difficult to achieve with traditional monolithic designs.

By encapsulating Solo-Tactics and lower-level Skills within each robot, the proposed architecture naturally allows these components to be migrated to the robot side in the future. This design choice is expected to facilitate the integration of distributed decision-making and onboard perception, such as local vision systems, while preserving the clarity of high-level strategy and coordination handled by the central Play and Multi-Tactics layers.

Furthermore, since the Skill layer clearly defines the interface between the AI and robot hardware, the quality of the hardware can be explicitly measured by the variety of available Skills and the precision of their execution. Consequently, the primary objective of hardware improvement is clearly defined: to expand the coverage and enhance the accuracy of these Skills.

To maintain clarity and maintainability, we decoupled the logic (state machines) from the interface (hardware abstraction). Fig. 12 shows an example behavior of our STP system during a kickoff. The Play Manager evaluates the field situation and determines the optimal strategy, assigning a Multi-tactic to each group of robots. Importantly, while the Play handles this high-level decision-making and allocation, it delegates specific execution logic to the lower layers. Each Multi-tactic assigns a Solo-tactic to each robot, and Solo-tactics enqueue atomic Skills. Moreover, to address the discontinuity of strategic context, we implemented a mechanism to trigger signals during state transitions. This enables timing synchronization or go/no-go branching based on success/failure across states.

Development Ecosystem To address the legacy issues of environment reproducibility and obsolescence, we modernized the technical stack by migrating to ROS2 Jazzy. We standardized C++ as the exclusive development language, ensuring system consistency and long-term maintainability. To prevent the software from becoming siloed or dependent on specific individuals, we established a

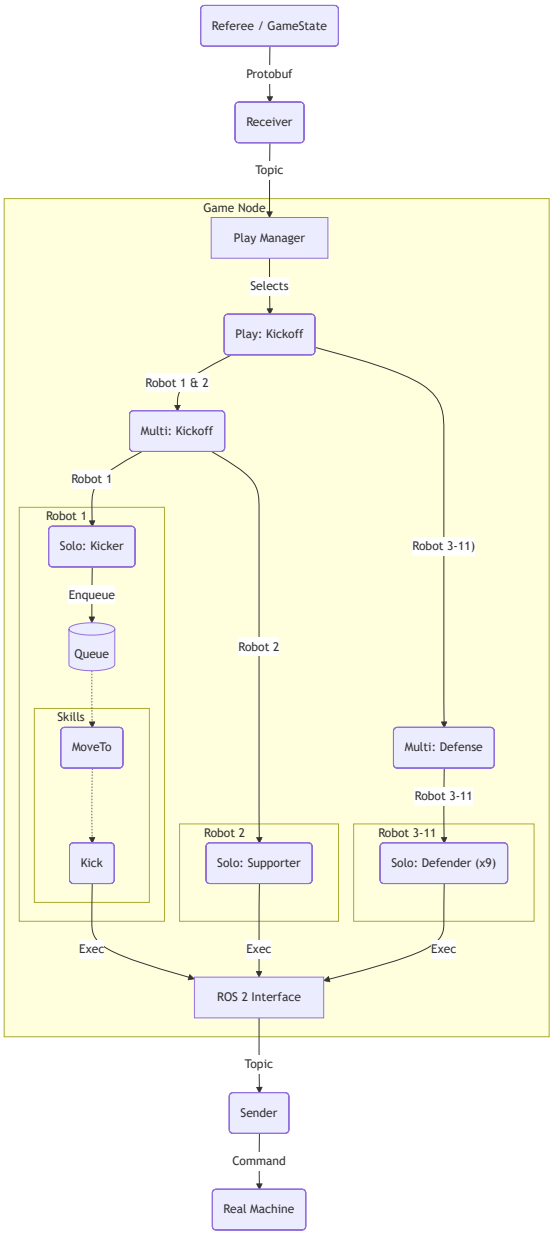


Fig. 12: Schematic overview of the new AI system "RIKYU". This diagram illustrates the flow during a kickoff situation as an example. The Play Manager receives information from external sources and sets the state of Play. The Play then groups robots by responsibility and assigns a Multi-tactic to each group (e.g., "Kickoff" or "Defense"). Each Multi-tactic then assigns a Solo-tactic (e.g., "Kicker" or "Defender") to each robot. Solo-tactics add Skills to a queue. Robots execute the actions corresponding to the queued Skills via the interface.

development process that mandates code reviews, ensuring that multiple members are constantly aware of the code structure. Furthermore, we implemented coding standards and utilized formatters, linters, and Continuous Integration (CI) to maintain code quality mechanically.

To ensure flexibility and reproducibility of the development environment, we constructed a container-based development infrastructure using Docker. This eliminates dependency on the host OS (native Ubuntu environment) and manual setup procedures, allowing for the immediate deployment of an identical execution environment across all developers' machines. This approach significantly reduces the onboarding cost.

For the purpose of validating strategies and edge cases, we implemented a simulation manipulation interface that allows programmatic control of the entire field state. This function enables the direct injection of arbitrary robot and ball states, allowing for the deterministic reproduction of complex situations (edge cases) that were previously difficult to recreate. This lays the foundation for automated scenario-based testing, accelerating the development cycle through software-defined quality control.

4 Conclusion and Future Work

Through the complete overhaul of both hardware and AI described in this work, we have constructed a robust and highly controllable robot development platform.

The RoboCup SSL is unique as the only league that frees teams from the constraints of physical body control (e.g., bipedal locomotion), allowing for the verification of high-speed multi-agent coordination involving 11 robots in a real-world, noisy environment rather than simulation. However, it has been pointed out that the league tends to overfit to the "Global Vision" environment (overhead cameras), a challenge recognized by the league itself since 2007 [3].

Building upon our newly developed system, we will fully integrate onboard Local Vision as the next phase of development. Specifically, while leveraging the SSL's advantage of inexpensive, high-speed wheeled platforms, we plan to implement object recognition on edge devices with limited computational resources, using this data for self-localization and coordinated actions. This approach represents a modern solution to a long-standing challenge. It demonstrates that the advanced integration of autonomous navigation and perception—scalable to future humanoid applications—can be achieved even by teams without access to expensive humanoid hardware.

References

1. Browning, B., Bruce, J., Bowling, M., and Veloso, M. Stp: Skills, tactics, and plays for multi-robot control in adversarial environments. *Proceedings of The Institution of Mechanical Engineers Part I-journal of Systems and Control Engineering 219* (12 2004).
2. Eischer, M., Blank, P., Danzer, A., Hauck, A., Hoffmann, M., Reck, B., and Eskofier, B. M. Er-force extended team description paper RoboCup 2015.
3. RoboCup Small Size League Committee. Small size league roadmap. `https://ssl.robocup.org/wp-content/uploads/2024/08/2007_ssl-roadmap.pdf`, 2007. Accessed on 2026-01-18.
4. Ryll, A., and Jut, S. Extended team description for robocup 2020.
5. Sato, H., Okamoto, N., and et.al. Greentea 2023 team description paper.
6. Sato, H., Okamoto, N., Ito, A., Kayaki, S., Nakaaki, S., Nakao, T., Nishimura, Y., Hara, Y., Fujita, K., and Yuri, R. GreenTea 2024 team description paper.
7. Texas Instruments Inc. Designing a DCM flyback converter. `https://www.ti.com/lit/ta/ssztcw6/ssztcw6.pdf`.
8. TIGERs Mannheim. AutoReferee: An Automated Referee for the RoboCup Small Size League, 2025. Accessed: 2026-01-19.