ibis RoboCup 2026
Extended Team Description Paper

Hiroyuki Kanai[1], Kotaro Yoshimoto[1],
Taiga Watanabe[1] and Ryoma Okada[1,2]

[1] ibis, Japan
`ibis.ssl.team@gmail.com`
[2] Nara Institute of Science and Technology, Japan
`okada.ryoma.on9@ms.naist.jp`

Abstract. This paper describes the hardware, software, and operational systems of ibis, a Japanese private team participating in RoboCup SSL. Hardware optimization achieved a 39mm center of gravity, boosting acceleration by 28%. The software utilizes a hierarchical Skill/Session architecture with RVO2-based path planning and real-time latency compensation. Furthermore, a debugging "planning factor" and an LLM-based explanation system were introduced. To ensure competition stability, a referee network utilizing L3 switches and multicast routing is also proposed.

1 Introduction

Team ibis is a private team of four working professional team members that has participated in RoboCup since 2024. At the 2024 RoboCup Japan Open, our first appearance, we received the Robotics Society of Japan Award, and at the 2025 Japan Open we placed second among domestic teams. We participate in all domestic scrimmages and publish an unofficial TDPs, making us one of the most active working professional teams in Japan. Our robot achieves a very low center of gravity (CoG) with optimized design, lowering the CoG to 39mm and improving the allowable acceleration by approximately 28%. Regarding the software, we adopted a hierarchical control architecture to enable flexible robot control and coordination among multiple robots. Additionally, by introducing L3 switches into the referee network, enabling multicast routing, and blocking communication between teams, we provided an environment where each team could receive only referee packets. This Extended Team Description Paper (ETDP) describes these contents.

2 Hardware Overview

2.1 Overview

The robot appearance is shown in Fig. 1, and its specifications are listed in the table 1. While following a standard four-wheel omniwheel layout, the platform

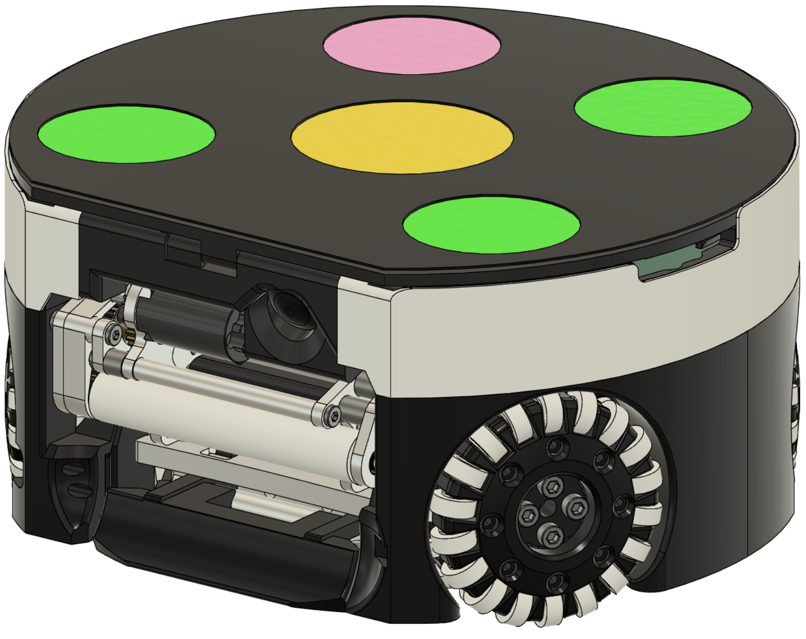

Fig. 1. Overview of the Orion 2025B robot platform.

Table 1. Specifications of the robot

Item	Specification
Robot version	orion (2025 Rev. B)
Dimension	ϕ178 x 90 mm
Weight	1.9 kg (without battery)
Motor	MAD Motor EEE 4006 200 kV
Battery	GAONENG 6cell 22.8 V 1550/1700/1850 mAh
Omni wheel	ϕ56 mm
Omni wheel (Sub wheel)	ϕ10 mm x 18 pcs x 2 layer
Kicker	Straight / Chip
Main computer	Raspberry Pi CM4
Main controller	STM32G474
Sub controller	STM32F303 x4
Camera	60 fps UVC camera FoV 160 deg
Motion sensor	ADNS3080 mouse odom sensor
Ball sensor	IR ball detector

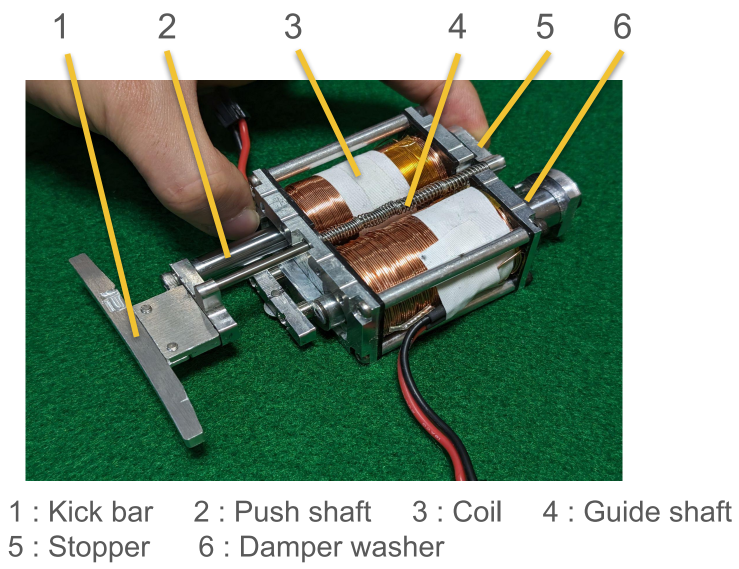

Fig. 2. Parallel-coil kicker layout.

adopts a direct-drive double-layer omniwheel, a parallel-coil kicking device, and a low CoG design. In addition, it includes an onboard camera and a mouse sensor, providing a configuration that considers future extensibility.

2.2 Low Center-of-Gravity, Low-Profile Chassis Design

In RoboCup SSL, the maximum robot envelope is constrained by the rules, and the maximum acceleration is therefore limited by wheel friction and the CoG. To improve locomotion performance, we optimized the overall hardware design and lowered the CoG as much as possible.

1. The kicking device was designed with reference to the 2023 GreenTea design [1]; as shown in Fig. 2, coils are arranged in parallel to achieve a low CoG and low profile.
2. The heavy battery significantly affects mounting orientation, therefore it was mounted horizontally to lower the CoG as much as possible.
3. The power board is integrated with the battery box directly above the kicking device. By minimizing structures in the thickness direction, the mounting heights of the power board and battery are kept as low as possible.
4. The motor driver board, main board, and sub board are split and placed on the left and right of the battery, resulting in a structure where only the top plate exists above the battery.

With these low-CoG and low-profile design choices, the overall height of the robot including the top plate is 90 mm, and the CoG height in CAD is 39 mm

Table 2. Comparison of CoG heights.

Robot (state)	CoG height
Orion 2025B (with battery)	39.0 mm
Orion 2025B (without battery)	36.0 mm
GreenTea 2024 [2]	47.0 mm
CompilationError 2024 [3]	51.0 mm
INPUT v2019 (drive unit only) [4]	49.2 mm
INPUT v2022 (drive unit only) [4]	39.7 mm

with battery and 36 mm without the battery. As shown in Table 2, while the SSL robot has a CoG height of approximately 50 mm, our CoG is more than 20% lower. In a 2D model, if the front and rear wheel contact points are located at ±50 mm from the vertical line through the CoG, the tipping-limit acceleration is given by

$$a_{\max} = g \cdot (50\ \mathrm{mm}/h)\,. \tag{1}$$

For $h = 50$ mm, $a_{\max}$ is 9.81 m/s^2 (1.00 g), and for $h = 39$ mm, $a_{\max}$ is 12.58 m/s^2 (1.28 g). Thus, lowering the CoG to 39 mm increases the allowable acceleration by about 28%.

2.3 Omniwheel

We use an omniwheel with 18 side rollers arranged in two layers. For the side-roller grip, a high-friction silicone tube is used, and PTFE spacers serve as bearings to balance low cost and low friction. The main wheel is 3D-printed, and its shape prevents external impacts from directly affecting the side rollers, improving durability.

2.4 Motor

For omniwheel drive, we adopted the MAD Motor 4006 200 kV drone motor. The lightweight, high-output motor enables direct drive together with the two-layer omniwheel within a limited chassis space where thickness tends to increase. A magnet for a magnetic encoder is bonded to the motor shaft end, and an AS5047 magnetic encoder and a thermistor are mounted on the rear side to measure rotor angle and temperature.

2.5 Dribbler

The dribbler is shown in Fig. 3. A compact design is achieved by using a small, high-output brushless motor for Mini-Z RC cars. The roller diameter is 12 mm, and the maximum roller speed is 15000 rpm at a 24 V supply. The damper mechanism has a simple rotating structure. Since the impact from the front is applied directly to the rotating shaft, the ball bearings would be damaged.

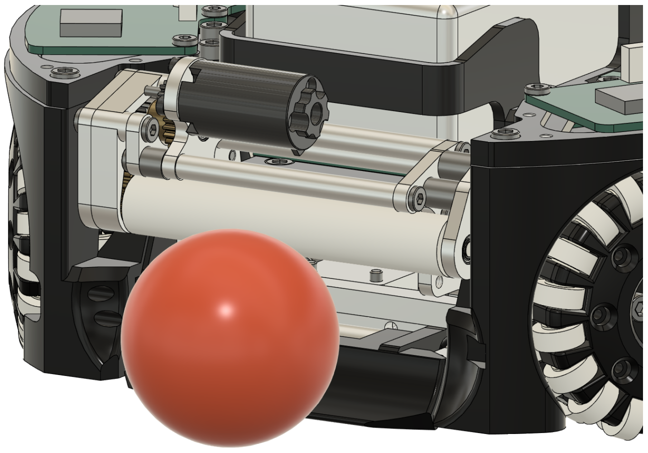

Fig. 3. Dribbler mechanism.

Therefore, we adopted plain bearings to provide sufficient durability. Because there was no space to add a return spring to the damper, a sponge is used to return the dribbler to its nominal position.

2.6 Mouse Sensor

Wheel odometry is used to estimate the robot's self-position. However, it has too much slippage to be used for control, and errors are particularly noticeable during acceleration and deceleration. Adding a dedicated measurement wheel solely for odometry is difficult due to chassis space constraints. As an alternative, we mounted an optical floor-tracking sensor based on a mouse sensor. At present, it is used only for logging to evaluate tracking stability, and we plan to integrate it into robot control in the future.

2.7 Front Camera

Near field edges, the SSL-Vision camera view can be occluded by the robot, causing the ball to be hidden. In addition, the SSL-Vision system has a relatively large latency of about 100 ms. To detect the ball with low latency and without visibility issues, a camera is mounted on the robot. The onboard camera is connected via USB to the Raspberry Pi CM4 on the main board, and it computes the ball position and size (distance) and sends them to the host computer and the robot control microcontroller.

2.8 Circuit Specifications and Design Philosophy

As shown in Fig. 4, the robot electronics are divided into five boards of four types. Each board has its own microcontroller, and inter-board data is exchanged

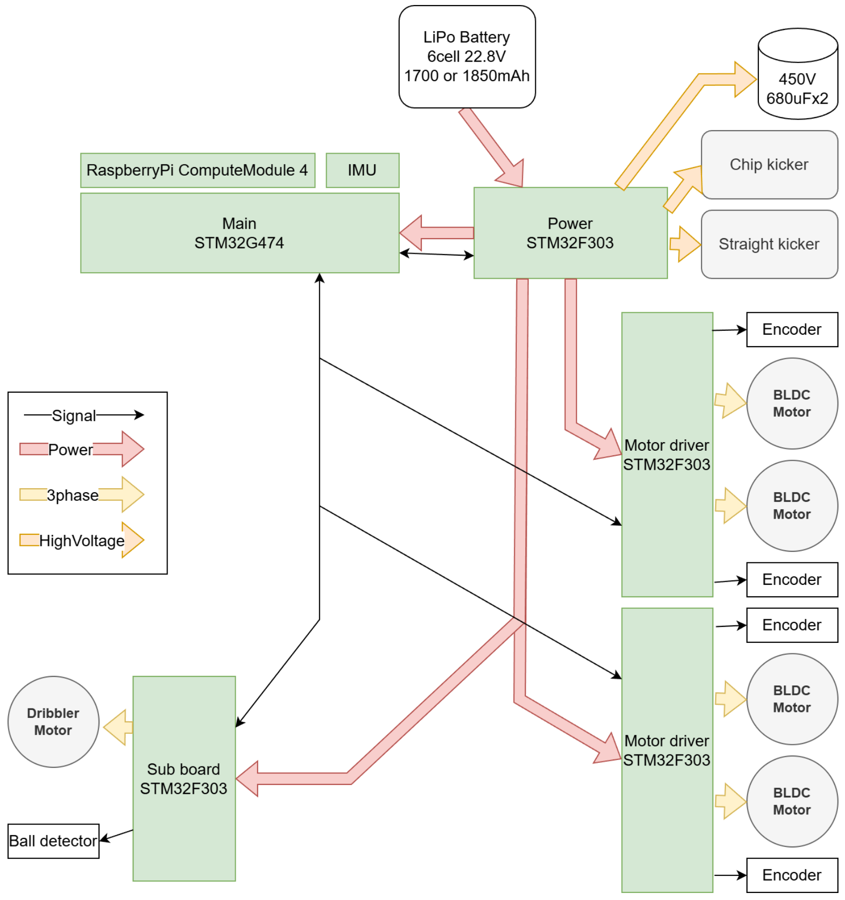

Fig. 4. Circuit block diagram of the robot electronics.

over CAN. Power connectors are unified to XT30, and CAN communication uses DF13 connectors. The wiring harness is standardized, and extra expansion connectors for daisy-chain connections are provided to allow future expansion. Because boards are separated by function (robot control, dribbler control, power control, and motor drive), a single board can be replaced when faults occur or when extending functionality, which improves maintainability and development speed.

2.8.1 Main Board

The main board integrates the STM32G474 main microcontroller for overall robot control, the Raspberry Pi CM4 for communication with the host computer and camera processing, and an IMU for yaw angle control. It also includes switches for operation checks and debugging, as well as a push-button switch, enabling standalone verification of the motor, kick device, and dribbler without external equipment.

2.8.2 Power Board

This board supplies power to the entire robot and includes charging and discharging circuits for the kick device, along with protection functions. It uses an STM32F303 microcontroller to monitor temperature, current, and voltage, shutting down power in abnormal conditions and controlling charge and discharge for the boost circuit. The boost circuit is a simple chopper configuration; by using high-performance SiC MOSFETs, it achieves more than 100 W output and a charging time within 1 s. The discharge circuit uses three parallel Si FETs to achieve both high voltage and current capability, allowing hard switching during discharge. Two 450 V 680 uF electrolytic capacitors are installed. Because higher-voltage capacitors tend to provide higher energy density, the design uses a relatively high charging voltage. The maximum charging voltage is designed as 450 V; however, failures occurred when charging to the maximum, and the required output decreased after improvements to the kick device, so the current charging voltage is 350 V.

2.8.3 Sub Board

The sub board carries the ball detection sensors and the dribbler motor ESC. It uses an STM32F303 microcontroller to modulate the IR LEDs for the ball detection sensors and to output PWM signals to the dribbler ESC. Some robots also include an OLED display to show ball detection sensor values and battery voltage. The ball detection sensors are redundant to improve fault tolerance, and their detection positions are intentionally offset, enabling two-stage ball position detection.

2.8.4 Motor Driver Board

This board drives two omniwheel motors per board, and two boards are mounted on the left and right sides of the robot. Motor angle is measured by a magnetic encoder mounted on the motor, and vector control is performed only in the voltage domain. By limiting control to the voltage domain, current sensors can be omitted, and two motors can be driven within the limited computational budget of a low-cost microcontroller.

3 Software Architecture

Our software system "**crane**" utilizes a hierarchical control architecture to achieve flexible robot control and coordination among multiple robots.

Fig. 5 shows the overall system architecture. The system receives and processes information from external systems. SSL-Vision's raw data (per-camera positions with noise) is filtered and tracked by an external SSL-Vision Tracker, which provides tracked data including velocity information. The World Model

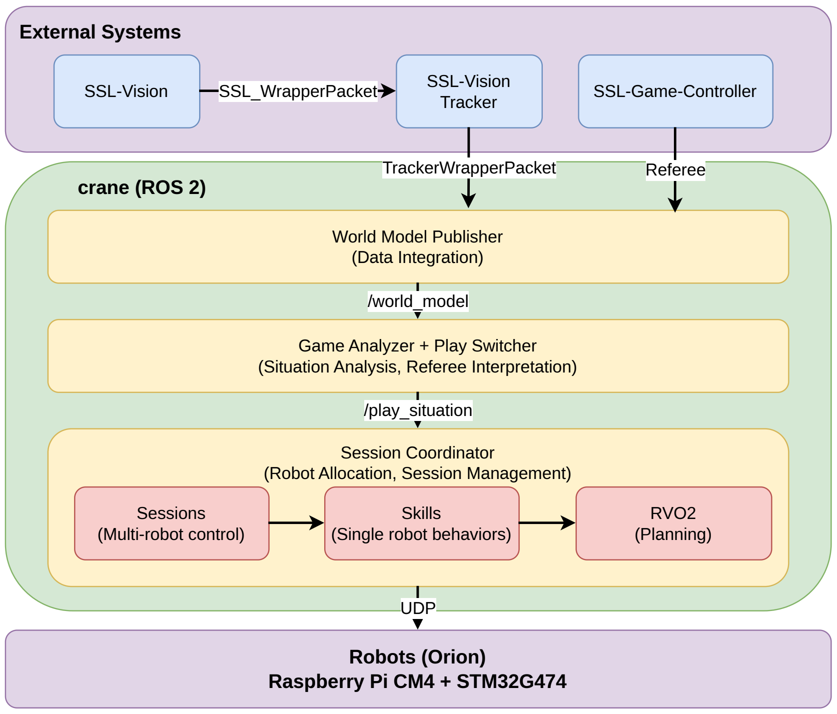
External Systems
SSL-Vision
SSL_WrapperPacket
SSL-Vision Tracker
SSL-Game-Controller
TrackerWrapperPacket
Referee
crane (ROS 2)
World Model Publisher (Data Integration)
/world_model
Game Analyzer + Play Switcher (Situation Analysis, Referee Interpretation)
/play_situation
Session Coordinator (Robot Allocation, Session Management)
Sessions (Multi-robot control)
Skills (Single robot behaviors)
RVO2 (Planning)
UDP
Robots (Orion)
Raspberry Pi CM4 + STM32G474

Fig. 5. Overview of the crane software architecture.

Publisher integrates this tracked data with referee information from SSL-Game-Controller to build the world model. The Game Analyzer analyzes the game situation, the Session Coordinator manages robot allocation, and control commands are generated through the Skills layer with path planning and sent to robots.

The control architecture is organized into two key layers: *Skills* and *Sessions*.

3.1 Skill and Session System

Multi-robot systems require addressing two distinct concerns: individual robot behavior control and multi-robot coordination. To separate these concerns, crane adopts a two-layer architecture consisting of Skills and Sessions, as shown in the lower portion of Fig. 5.

A *Skill* is the basic unit of control for a single robot. Each Skill embeds a finite state machine (FSM) to manage internal states and implements behavior as a state transition system. For example, an Attacker skill may have states such as `APPROACH`, `RECEIVE`, and `SHOOT`, transitioning between these states based on ball position, distance to the ball, and other conditions. Skills can incorporate other Skills as sub-components; for instance, the Attacker skill is composed of multiple sub-skills including Kick, GoalKick, and Receive.

A *Session* is a management unit for coordinating multiple robots. Sessions receive the current world model (positions of robots and the ball, game state, etc.) as input, and determine which Skill should be assigned to each robot and which robot should execute which role.

This two-layer structure separates individual robot behavior logic from multi-robot coordination logic, enabling independent development and testing of each layer.

3.2 Obstacle Avoidance with RVO2

Previously, we used the A* algorithm for path planning. However, A* requires field discretization, resulting in non-smooth paths, and struggles to handle dynamic obstacles within the control cycle. To achieve smooth obstacle avoidance with better dynamic response, we transitioned to the Reciprocal Velocity Obstacle (RVO) approach [5], shown as the RVO2 component in Fig. 5.

The RVO-based approach enables robots to avoid dynamic obstacles (particularly other robots) while maintaining smooth, oscillation-free trajectories. Each robot independently computes collision-free velocities while implicitly accounting for the reactive behavior of other agents. By operating in velocity space rather than on discretized grids, RVO provides smooth, natural robot motion at the 60 Hz control frequency.

Static constraints are handled separately before RVO computation. When a robot's intended target position would violate game rules—such as crossing the field boundary or entering the opponent's penalty area—the system computes the nearest rule-compliant position along the path from the current position to the original target. This adjusted position is then used to determine the preferred velocity for RVO. Game-specific constraints, such as maintaining minimum distances from the ball during certain referee commands, are also enforced through this preprocessing step. This separation of concerns allows RVO to focus on dynamic collision avoidance while ensuring that all static and rule-based constraints are always satisfied.

3.3 Latency Compensation

SSL-Vision typically has a latency of about 100 ms, which corresponds to a 40 cm position offset for a robot moving at a maximum speed of 4 m/s. Controlling based on this outdated position degrades obstacle avoidance and ball handling accuracy in the dynamic SSL environment.

To compensate for this latency, we transmit both the SSL-Vision-based robot position and the measured SSL-Vision latency to each robot. Each robot accumulates odometry data (from wheel encoders and IMU) in a ring buffer at a 2 ms cycle (500 Hz).

Upon receiving the SSL-Vision position, the robot integrates the accumulated odometry over the latency period to estimate its current position. The corrected position is calculated as follows:

$$\mathbf{p}_{\text{corrected}} = \mathbf{p}_{\text{vision}} + \sum_{i=0}^{N} \mathbf{v}_{\text{odom}}[i] \cdot \delta t \tag{2}$$

where N is the number of latency cycles, computed from the sum of host server latency, elapsed time since the last SSL-Vision update, and elapsed time since the last command reception; $\mathbf{v}_{\text{odom}}[i]$ is the odometry velocity increment in the global frame; and $\delta t = 2$ ms is the sampling period.

This corrected position is then sent back to crane on the host server, which uses it for subsequent control commands. This method enables the control loop to operate with near real-time state estimation, significantly reducing the effects of SSL-Vision latency.

This approach provides an additional benefit. Because our robots are designed with an extremely low profile, the markers can be occluded from the SSL-Vision camera when approaching taller opponent robots, causing temporary loss of detection. This issue occurred at Japan Open 2025 and caused at least one goal to be conceded. The position feedback from the robot side enabled continued control even during such SSL-Vision occlusion situations.

4 Interpretability and Commentary

4.1 Overview

In the development and operation of complex multi-robot systems, making the system's decision-making processes understandable is crucial. We approach interpretability from two perspectives: (1) developer-oriented debugging support, and (2) spectator-oriented match commentary. The former enhances development efficiency and supports root-cause analysis, while the latter increases the educational and entertainment value of RoboCup SSL. This section describes both approaches.

4.2 Interpretability for Developers: Decision Provenance Recording

To debug efficiently using logs, we must record not only commands but also why each command was selected.

Inspired by Autoware's Planning Factor [6], crane logs the decisions made across the processing pipeline from higher layers (Sessions) to lower layers (Skills) for each robot.

4.2.1 Self-Documenting Code with Planning Factors

State transitions and decision rationales in Skills are recorded using the `addPlanningFactor` function. This function serves a dual purpose as both source code comment and runtime log: it helps human readers understand runtime decisions when reading the code, while simultaneously being logged for later analysis.

The following example illustrates this approach:

```
Point Receive::calculateTargetPosition() {
  Point closest = /* calculate intercept point */;

  // Decision: Is robot close enough to target?
  if (robot()->getDistance(closest) < 0.1) {
    command->addPlanningFactor("Receive",
      "Robot close to target, forcing closest policy");
    return closest; // Early return
  }

  // ... continue with normal policy ...
}
```

Listing 1. Example of Planning Factor usage in the Receive skill.

In this example, when the robot is close enough to the target (within 0.1m), the early return policy is selected and this decision is immediately recorded via `addPlanningFactor`.

4.2.2 Debugging Utilization

Planning Factors are included in each robot's command message and can be monitored during execution. Critically, control commands and Planning Factors are bundled in the same message. Each message contains: (1) control commands (target velocity, kick power, dribble speed, etc.), (2) Planning Factors (Skill names, states, decision rationales, etc.), and (3) robot position and velocity.

This design reveals why a particular command was generated. Commands alone show only "what" the robot is doing, but Planning Factors reveal the "why": which state of the Attacker skill was active, which policy was selected, and which conditional branches were taken.

When problematic behavior occurs, we can trace which decision led to the unexpected command by inspecting the Planning Factors in the robot's command message. This enables both human debugging and allows LLM-based coding assistants to investigate root causes in complex systems.

4.3 Commentary for Spectators: LLM-based Commentary System

RoboCup SSL matches are fast-paced and complex, making it difficult for spectators to understand tactical decisions and robot coordination. Prior work on automated commentary for spectators includes TIGERs Mannheim's AudioRef [7], which uses template-based speech synthesis to announce match events. We aim to generate more entertaining and natural commentary by leveraging LLM real-time APIs.

As part of crane, we developed an LLM-based commentary system called `crane_commentary` using the Gemini Multimodal Live API. Gemini's native audio output capability enables low-latency voice commentary generation.

In *reflex commentary mode*, the system generates enthusiastic commentary when critical events (goals, shots, saves, possession changes) occur, receiving event triggers from crane's event detection system via ROS2.

Analyst commentary mode activates during quiet periods (threshold: 5 seconds). The system incorporates SSL rule knowledge and team profiles, and through Function Calling, it actively retrieves match state, robot status, formation analysis, and recent highlights to provide context-aware tactical analysis.

5 Stable Operation of the Referee Network

The previous RoboCup SSL Referee Network, as shown in Fig. 6, was connected on a single Layer 2 (L2) network. In this case, as shown in Fig. 6(a), there is a risk that Referee software running on the local network could unintentionally flow into the Referee Network. Furthermore, as shown in Fig. 6(b), packets intended for robots could also end up connected on the same L2. Consequently, as seen in Fig. 6(c), the team would receive not only packets from the Referee PC but also packets from Team A and Team B. This not only congests the Referee Network's bandwidth but also prevents the normal reception of packets essential for the competition, significantly impacting the match.

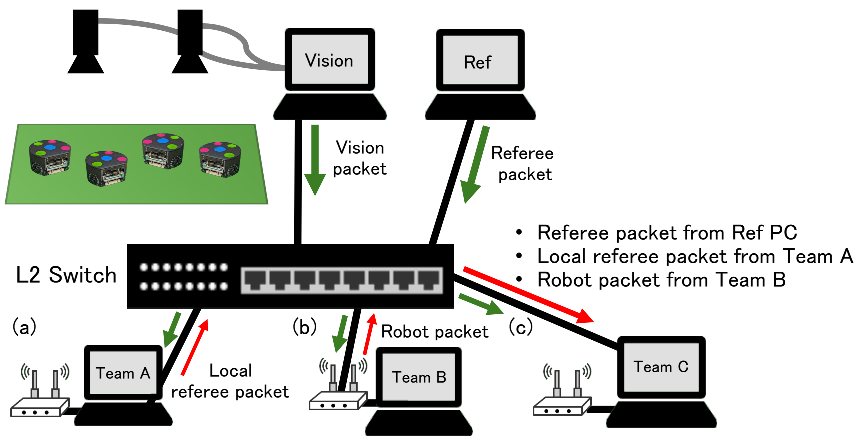

Fig. 6. RoboCup SSL previous Referee Network.(a) When a local referee packet is accidentally sent to the referee network (b) When the router connecting to the robot is directly connected to the referee network (c) When receiving packets from another team

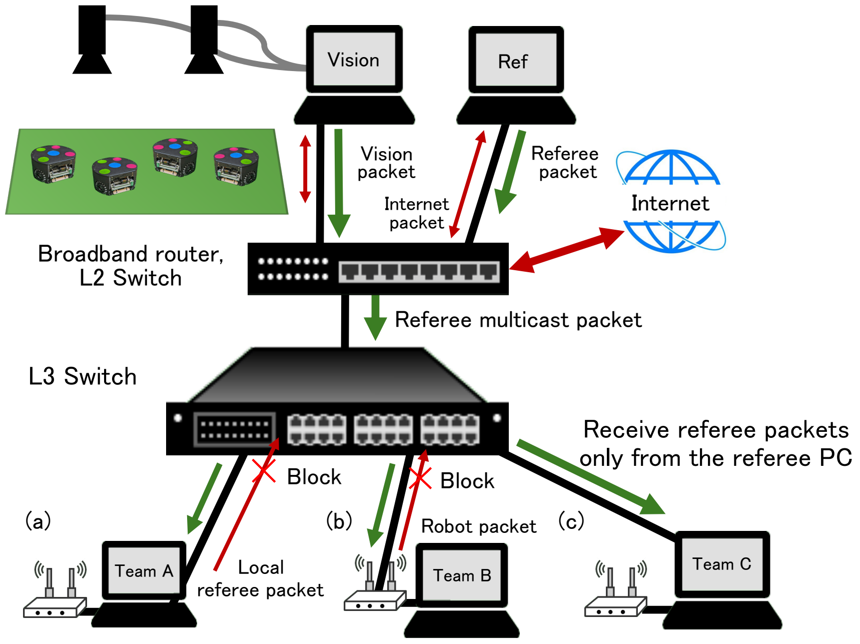

Fig. 7. Proposed RoboCup SSL Referee Network

To resolve this issue, an L3 switch was added as shown in the network diagram in Fig. 7. On the L3 switch, we configured multicast routing rules using Protocol-Independent Multicast Sparse Mode (PIM-SM). The rendezvous point (RP) in PIM-SM was set to the IP address of the broadband router prepared for the referee. Additionally, we set up a port-based Virtual Local Area Network (VLAN) for each team. As a result, each team sends Join messages to the RP, enabling multicast packets to be distributed from the RP. Furthermore, the network between teams is configured to be isolated, allowing only connections to the referee network. Therefore, even if a local referee packet is accidentally transmitted (as in (a)), or a robot packet is transmitted onto the referee network (as in (b)), the packet is discarded at the L3 switch and cannot be received by other teams (as in (c)). As a result, each team receives only the packets necessary for the competition, while also reducing bandwidth pressure. Furthermore, the broadband router prepared for the referee allows Network Address Translation (NAT) only for the referee PC, ensuring only the referee PC can connect to the Internet.

By adding an L3 switch, network latency due to packet distribution is expected. However, this delay is only a few milliseconds, significantly smaller than SSL-Vision's delay, and its impact is considered negligible. More importantly, only packets essential for the competition are distributed to each team, enabling a stable network environment. During SSL practice sessions in Japan, we tested the network shown in Fig. 7 and confirmed it functions without issues. On the other hand, each team must take care not to transmit local referee packets or robot control packets onto the referee network. A simple solution can be achieved by separating the network card (NIC) that receives referee packets, and the NIC that sends and receives robot control packets. Each team is required to communicate using this method.

6 Conclusion

In this paper, we introduced our robot, software, and operation tools, and summarized our hardware optimization focusing on lowering the center of gravity, improving debugging efficiency by collecting logs and recording decisions during operation, and the design and verification of a tournament operation network for RoboCup 2026 Japan Open. In the future, we will continue to improve control using onboard sensors and expand the operation tools to further improve stability and reproducibility in a competitive environment.

References

1. Hirotaka Sato, Naoyuki Okamoto, Akito Ito, Shun Kayaki, Naoya Sugishita, Shoma Nakaaki, Tomoki Nakao, Yuki Nishimura, Yuto Hara, Kohei Fujita, and Rintaro Yuri. Greentea 2023 team description paper, 2023.
2. Greentea blog 240525: ロボット重心測定. Accessed: 2026-01-24. URL: https://greentea-ssl.com/2025/06/21/2025-06-21-MeasuringCenterOfGravity/.
3. Yuxuan Wu, Kaiyang Liu, Jiaming Fan, Hao Zhang, Jiangyong Li, Minghe Wang, Qinfeng Li, and Jun Li. Compilation error team description paper for small size league of robocup 2024, 2024.
4. Masaki Yasuhara, Tomoya Takahashi, Hiroki Maruta, Hiroyuki Saito, Shota Higuchi, Takaaki Nara, Keitaro Takeuchi, Yota Sakai, and Kazuki Ishibashi. Input 2022 team description paper, 2022.
5. Jur van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. RVO2 Library: Reciprocal Collision Avoidance for Real-Time Multi-Agent Simulation, 2016. RVO2 Project; Accessed: 2026-01-25. URL: https://gamma.cs.unc.edu/RVO2/.
6. Autoware Foundation. Planning factors: Autoware documentation. Accessed: 2026-01-26. URL: https://autowarefoundation.github.io/autoware-documentation/main/design/autoware-architecture-v1/interfaces/ad-api/features/planning-factors/.
7. TIGERs Mannheim. Audioref: A voice for the game controller!, 2024. Template-based speech synthesis for match event announcement; Accessed: 2026-01-26. URL: https://github.com/TIGERs-Mannheim/AudioRef.