2026 Team Description Paper: UBC Thunderbots

Saurav Banna[b], Ryan Chan[c], Enoch Chiu[c], James Coffin[d], Sung Hyun Eun[a], Thomas Frew[a], William Ha[c], Jay Huang[g], Raiaan Khan[a], Marcel Khayam[c], Clint Lee[c], Lucie Li[e], Sherea Lu[a], Andrew Mao[b], Noah Mould[e], Swarna Raja[c], Leah Song[c], Colin Wang[f], Alanna Yip[e], Paul Zhou[a], Tiger Zhou[e]

Departments of: (a) Mechanical Engineering, (b) Computer Science, (c) Electrical and Computer Engineering, (d) Engineering Physics, (e) Integrated Engineering, (f) Applied Science, (g) Chemical and Biological Engineering
The University of British Columbia
Vancouver, BC, Canada
www.ubcthunderbots.ca
robocup@ece.ubc.ca

Abstract. This TDP details design improvements and developments UBC Thunderbots has made for RoboCup 2026. Building upon the major redesigns initiated in our previous cycle, the team primarily focused on improving the reliability of our new drivetrain and motor driver stacks. Additionally, our team investigated major noise, thermal and layout issues observed in RoboCup 2024 and resolved them in our newest electrical stackup. On the software side, we introduce a simulation-based framework for systematic AI strategy evaluation, featuring standardized gameplay metrics and a flexible runtime manager that enables controlled AI vs. AI testing against multiple versions and external teams' AIs.

Keywords: RoboCup 2026 · Small Size League · Robotic Soccer.

1 Introduction

UBC Thunderbots is an interdisciplinary team of undergraduate students at the University of British Columbia. Established in 2006, it pursued its first competitive initiative within the Small Size League at RoboCup 2009. The team has consecutively competed in RoboCup from 2009 to 2024 and is currently seeking qualification for RoboCup 2026. Motivated by performance and reliability issues observed in RoboCup 2024, the team has tried a variety of new solutions and upgrades in pursuit of the Division B title. This paper details UBC Thunderbots' new developments in mechanical, electrical, and software systems to compete in RoboCup 2026.

2 Mechanical

For 2025-2026, our mechanical team has focused on making improvements to our previous robot design, including major upgrades to the drivetrain motivated by our recent switch to direct drive as well as various testing jigs. We have also been improving our robots' robustness and consistency by optimizing several component designs. Our most recent CAD model is shown below in Figure 1.

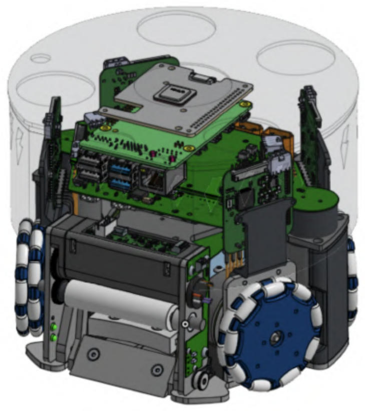


Fig. 1. Mechanical Top Level Assembly

2.1 Drivetrain

Drive Module Many teams use an omniwheel design consisting of a single row of rollers. While this design is compact, the profile is often not perfectly circular due to discrete contact points with the ground, resulting in vibrations and reduced traction during operation. To lessen these undesirable effects, we experimented with dual-row omniwheels (Figure 2b) as an update to the single layer omniwheel we had previously designed for our direct drive[1] as shown in Figure 2a.

Initially, two different types of rollers were considered for the subwheels. X-rings, as in previous years, were used to improve traction by providing two points of contact rather than the single point provided by O-rings. After sucessfully integrating 3D printed rollers for our dribbler last year, we also prototyped 3D-printed rollers for the wheels with the same TPU 60A filament[1]. While the higher number of X-rings in the previous design (Figure 2a) reduces variations in wheel height, the TPU rollers provide a continuously round profile for the

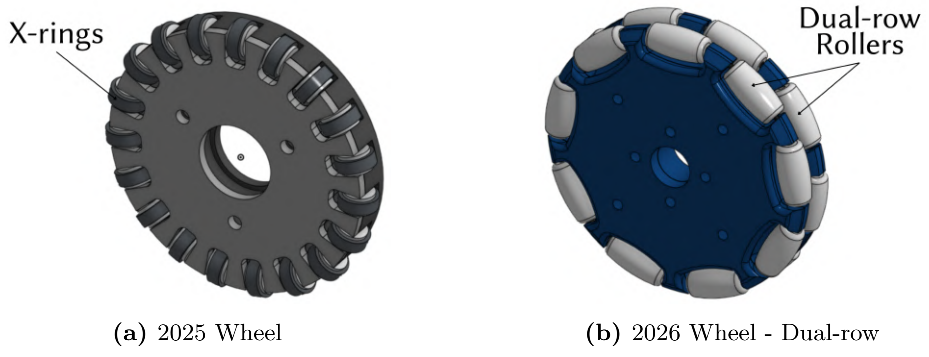


Fig. 2. Wheel design comparison

wheel. Observing the stationary friction forces using an inclined plane similar to the test conducted in Ri-one's 2023 TDP [2], the TPU 60A rollers had a greater gripping force than the rubber X-rings. The 3D-printed rollers enable quick future iterations, customization, and manufacturability. However, life cycle testing to asses the durability of the rollers are needed before full implementation of this design. Similar to 2025[1], the 2026 version continues to feature individual dowel pins for each roller, facilitating easier removal and maintenance.

Shaft Coupling Generally, drivetrain motors applicable to SSL are sold with either a circular or D-shaft, the former being less expensive and more accessible to teams. However, circular shafts are significantly more difficult to couple to the wheel, creating a greater margin of error for the wheels to slip or shift. Previous teams have often relied on permanently attaching the wheel and shaft using methods such as Loctite, which is not always ideal for repairability and modularity[3].

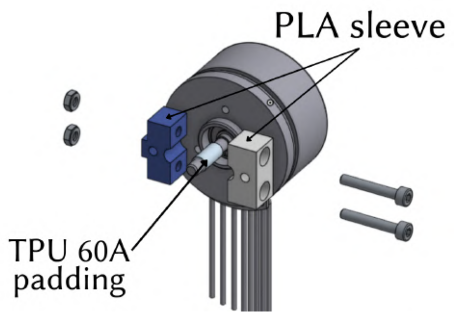


Fig. 3. Shaft Coupling

Wanting a less permanent coupling option, we found that commercially available options, such as shaft collars, were either incompatible with the wheels, damaged the motor shaft, or could not clamp onto the shaft without significant slipping. We also investigated taper lock bushings using 3D printed variations with PLA and TPU. However, without a key slot, these connections mainly prevent axial slippage and still allow the wheel to slip radially when sufficient force is applied.

Our current shaft adapter design (Figure 3) consists of a PLA sleeve similar to a clamp shaft collar that surrounds a TPU 60A padding for grip. Both the sleeve and the padding can be 3D printed and adapted for different motor shafts and wheel sizes. Compared to the shaft collars and taper lock bushings, this method has shown much reduced radial slippage. Further testing, particularly involving the full robot weight, is still needed to confirm its effectiveness under expected driving scenarios.

2.2 Dribbler

Dribbler Roller Testing Jig Switching to 3D-printed rollers[1] enabled us to quickly manufacture and test various roller shapes. This year, we have improved our printer and slicer settings to successfully print rollers with threads used for centering. As we continue to refine our manufacturing processes, we hope to be able to experiment with more complex designs (e.g. thinner/larger threads, rollers that slant inwards in the middle) and ensure their replicability across the fleet of robots.

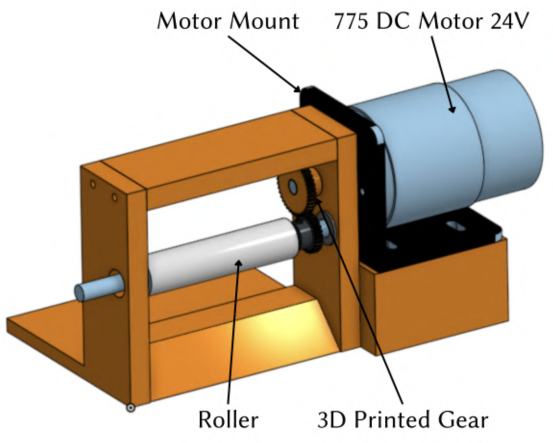


Fig. 4. Dribbler Jig

To evaluate these different roller designs, we designed a simple testing jig that enables quick and repeatable testing. The jig is meant to imitate the dribbler on the robot but it is separate from the chassis and able to move around. Because the actual dribbler motors were unavailable, we used a standard DC motor to drive the roller. For simplicity, the motor is mounted off to the side of the jig, as it is larger than the standard dribbler motor and cannot be accommodated within the crossbar. With this jig we hope to be able to conduct more tests with various roller types, gathering both qualitative and quantitative data to determine the ideal design for our dribbler.

2.3 Kicker

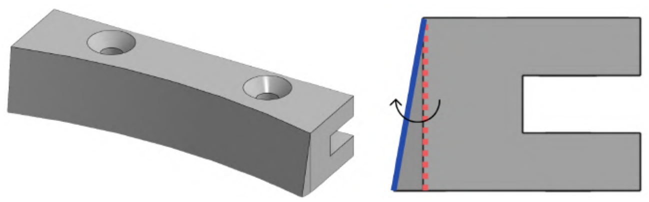


Fig. 5. Kicker Head and Side Profile (Old = - - - -, New = ——)

When designing the kicker, maximum kick speed is a primary performance metric. However, we observed that initial ball velocity alone does not fully characterize kick effectiveness. In practice, the ball experiences significant deceleration due to ground friction, making average velocity over the travel distance a more relevant metric for gameplay performance. As a result, we are investigating methods to reduce post-impact energy losses.

One approach is to vary the kicker face angle (Figure 5) to influence the ball's launch angle and spin characteristics, with the goal of reducing rolling friction and maintaining higher average velocity. An inclined kicker face applies a tangential force component in addition to the normal force, theoretically inducing more backspin. Backspin can generate an upward aerodynamic lift force through the Magnus effect, causing the ball to remain airborne slightly longer and effectively "float," reducing ground contact time and associated frictional losses. Additionally, a small upward launch angle may temporarily reduce ground contact, further decreasing frictional losses.

Kick Speed Tester There are limited solutions that allow teams to quickly and quantitatively evaluate the effectiveness of changes made to the kicker system. In the past, we measured ball velocity using software-based methods, which required camera setup with a high enough FPS to observe small increments, calibration, and the involvement of multiple team members. There was no method for an individual to quickly gather data. To address this gap, we developed a kick

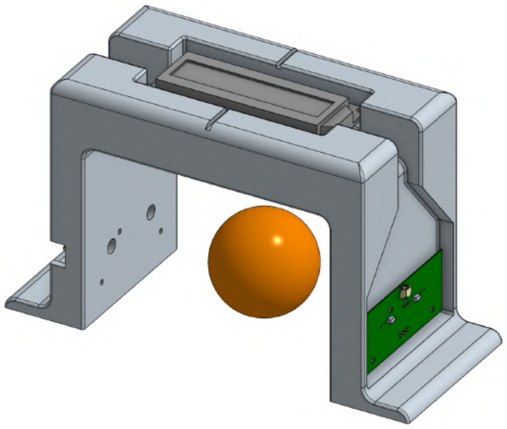


Fig. 6. CAD for Kick Speed Tester

speed tester (Figure 6) that measures the velocity of the ball at a certain point, providing objective numerical data that can be used as feedback when iterating on the solenoid, chipper, or kicker design. This is a development from our previous kick speed tester, which focused more on preliminary PCB design[1]. This new speed tester reduces data collection from 60 minutes to less than a minute as no manual processing is required to obtain the speed. The system uses a double break-beam sensor setup to measure the time of flight over a known distance, with the calculated ball speed displayed on an LCD in real time. This enables rapid, repeatable testing of different hardware configurations by a single member.

3 Electrical

3.1 Electrical System Overview

This year, our electrical design projects focused on maximizing efficiency in both firmware and physical layout, in response to many cases of suspect cable management and prolonged testing of software algorithms due to missed diagnostics. Our changes included supplying more information using a HV feedback circuit setup to accurately determine when our kicker/chipper solenoid actuation capacitors were fully charged, and reducing redundancy in the USB/UART connection between the Power Board and the UI Board through direct header pins. As well, our unique current sensing circuit delivered real-time current readings of our motors to our software team to increase debugging accuracy. Our projects make us confident that our electrical troubleshooting capabilities will be exponentially more successful during gameplay and in strategy development.

3.2 Power Board

We introduced complete isolation of the high-voltage systems last year on the power board with a new flyback circuit, isolated gate drivers and a 24V-15V DC-DC converter. This year, our main goal was to integrate features to complement the new isolated design.

HV Feedback Circuit Actuation of our kicker and chipper solenoids is achieved by discharging a pair of 240 V capacitors, which are charged using an LT3750 flyback converter. This charges the capacitors to a programmed voltage and signals completion before entering an idle state. Previously, the voltage of the capacitors was monitored post-charge via the analog pin of an ESP32 using a voltage divider and unity buffer.

In recent years, however, the power board was redesigned with galvanic isolation to separate the high-voltage and low-voltage circuitry. As a result, the system lost the ability to directly and continuously monitor capacitor voltage from the low-voltage domain through our previous model.

To restore HV voltage monitoring under these adjusted conditions, a new feedback path was implemented. The design fed the 240 V rail through a modest voltage divider before entering an ISO224 reinforced isolated amplifier. As seen in Figure 10, the ISO224 amplifier fed into the ADS7945 IC, which was chosen due to its reasonable input voltage and ability to digitize and feed the amplifier's differential output directly into the ESP32 through SPI. This telemetry enables accurate, isolated measurement of the capacitor voltage, allowing the system to detect and compensate for voltage decay, trigger recharge events as needed, and ensure controlled discharge for safety.

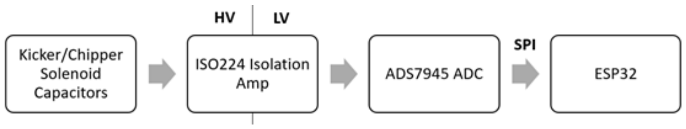


Fig. 7. Block diagram of HV feedback circuit layout

UART Communications Previously, a USB connection via an FTDI board (to convert the USB into a UART connection) bridged the Raspberry Pi and the Power Board in order to flash firmware to the ESP32.

The idea for this year's improvement was to switch to direct UART communication between the Raspberry Pi and the Power Board, via the UI board. Noting that signal integrity had been impacted in previous implementations by large amounts of noise, we verified communications by stress-testing through sending

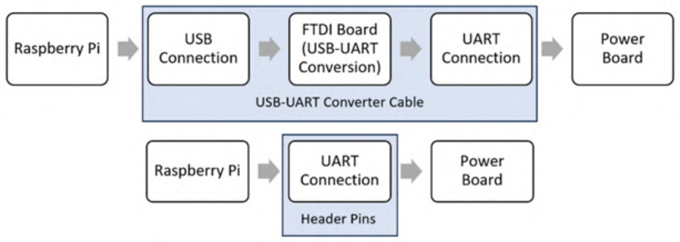


Fig. 8. Block diagram for previous and current UART communication setup

messages with checksums. As a result, we saw no noise issues impacting the quality of our data through this method.

Currently, the boot and reset buttons on the powerboard are required to be pressed in order to flash, which became an accessibility flaw. In response, we set up onboard flashing through micro-USB, with additional capability to turn the ESP32 automatically into bootloader mode, using the logic circuit on the ESP32 Dev-C boards. Another key issue was avoiding excess bus traffic via the shared pins, which was resolved through a switch to connect/disconnect the two separate flashing systems of the USB or the Raspberry Pi.

3.3 Motor Driver Board

The motor-driver board is a custom three-phase inverter designed to drive and control the brush-less DC (BLDC) motors of our robots. It is currently undergoing its two-year redesign cycle with the aim to modularize the hardware into five separate PCBAs to simplify system-level debugging and assembly. The previous iteration of the motor-driver board (MD v5) presented issues with cost, thermal management, current sensing, and overall debuggability. With the latest version (MD V6), each individual board will consist of its own power stage (inverter circuit), MCU, and current sensing circuits.

Power Stage & Hot Loops The inverter's power stage is an integral to BLDC motor control as it converts high DC voltage into three 120 degree phase-shifted currents and voltages via varying PWM signals that control three sets of MOSFET half bridges. A key design consideration when working with the power stage is the hot loop, which carries the highest magnitude current in the circuit.

The hot loop leads to various problems, such as switch-node ringing, MOSFET damage from excessive heat generation, and a decline in signal integrity. Thus, it is imperative that we mitigate the effects of the hot loop. The two primary factors that influence the consequences of hot loops are loop inductance and the change in current over time ($\frac{di}{dt}$). Keeping this in consideration, we aimed to minimize the loop area.

For our inverter circuit, the key elements of our hot loop and signal return path consists of the DC link capacitor bank (X-CAP), MOSFET half-bridges, high voltage and ground, and the shunt resistor.

To minimize our hot loop, we have intentionally put our components as close together as possible on the board. The shorter the traces, the shorter the return path, and thus the smaller the loop inductance. Additionally, we have avoided routing through multiple planes, which ensures that our hot loop stays on the top signal layer. In this way, we will minimize the electromagnetic interference (EMI) emitted from the power stage to neighboring sensitive circuits, thereby boosting the reliability of our robot's performance.

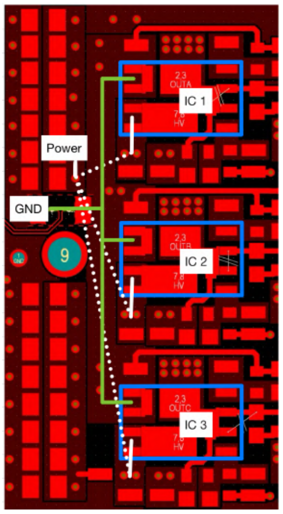
To avoid hot loops that traverse to multiple planes like our previous design, shown in 9a, we changed the placement of our half-bridge ICs to enable high voltage and ground to be connected on the same plane. This can be seen in 9b, where the white traces are the high voltage trace, blue is the half-bridge IC, and green is the ground trace. We found out that this is the only configuration which enables us to maintain the hot loop on the same plane and the current sense resistor differential pair output to reach the current sensing circuitry. The output phases, gate-drive, bootstrap, and snubber circuits are placed on the bottom signal plane.

Current Sensing Circuit As referenced in the Hot Loops section, the previous motor-driver design prioritized minimizing the area of the high-current hot loop in order to reduce EMI. However, this layout choice resulted in a long current sense trace to the current sense op-amp. While the current sense is a differential pair, it was observed to still be susceptible to crosstalk and switching noise from the power stage. In the revised design, we aim to address this trade-off by modifying the half-bridge MOSFET IC placement so the differential current sense lines can be routed much closer to the shunt resistor whilst preserving a compact power loop in the power stage. This change improves the signal integrity for the current measurement without compromising the noise performance of the power circuitry.

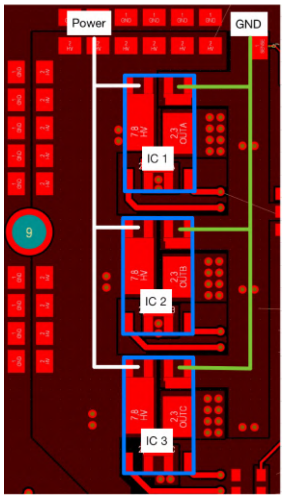
Bootstrap The challenge of high-side switching is the high-side switch's requirement for a voltage higher than its source to turn on. The bootstrap circuit is an essential component that solves this issue by providing a simple implementation of a floating gate driver without the need for a separate voltage supply.

With the previous design, the half-bridge circuits included individual discrete components, which were included on the motor driver boards. These components include a capacitor (C47), a resistor (R26), and schottky diode (D9) as shown in Figure 10.

However, we later discovered an existing diode integrated inside the MCU at its BOOT pins (3 total, 1 for each phase), where it connects in series with V_{CC}.



(a) Old half-bridge IC PCB placement (dotted lines are routed from power plane)



(b) New half-bridge IC PCB placement

Fig. 9. Comparison of Power Stage layouts

This allows us to remove both the Schottky diode and resistor from the motor driver PCBAs, saving board space and cost.

Stack-Up Changes PCB stackup is one of the most important aspects to consider at the start of any PCB project. This year, we changed our 4 layer PCB layout from **signal-ground-power-signal** to **signal-ground-ground-signal**. This improves signal integrity and EMI performance by providing all signal layers with a continuous, low-impedance ground reference. This reduces crosstalk, lowers EMI, and provides a cleaner ground reference for gate-drivers and current sense circuits. Another advantage is that it provides more copper area for heat dissipation. However, this requires us to route the high voltages on the signal plane with very limited space. Another downside includes the limited availability for routing on ground planes, as it induces EMI on the ground plane. We cannot route signal traces on the power plane from before because there is no power plane anymore. This makes routing signals more difficult and requires a larger area to route signals. We chose this stack-up because we believe the improved signal integrity and EMI performance provided by this stack-up outweigh the increased routing complexity and layout constraints it introduces. We are able to fit the power stage into the two layers of the signal plane by removing unnecessary components, and move the bootstrap circuit from the top signal layer to the bottom signal layer while maintaining signal integrity. Overall,

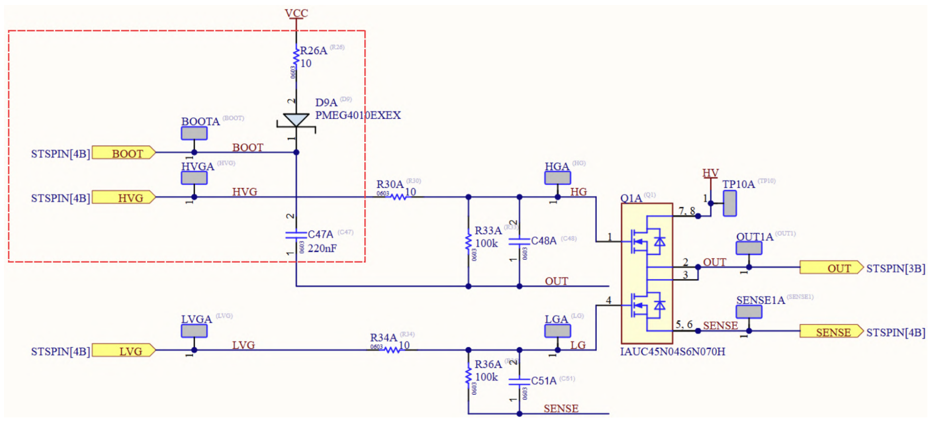


Fig. 10. Half Bridge with Bootstrap Circuit

this stackup prioritizes noise control, signal robustness and heat dissipation but increases power-routing complexity.

4 Software

All our software and firmware is open source and is available from our Git repository: `https://github.com/UBC-Thunderbots/Software`.

4.1 Simulator-Based AI vs. AI Strategy Evaluation

Thunderbots' software has reached a degree of maturity where meaningful research into higher-level team strategy is feasible. However, as is common across the SSL, constraints related to robot manufacturing, maintenance, and fleet availability significantly limit the ability to conduct gameplay testing on real hardware. As a result, our simulation framework has become the primary environment for rapid iteration and evaluation, enabled in large part by the league-wide adoption of the SSL Simulation Protocol. Following the 2021 virtual competition, many teams' AI systems support this protocol, allowing their software to interface with simulators such as grSim [4] and the ER-Force simulator [5] and facilitating cross-team virtual scrimmages.

In the current Thunderbots development environment, simulated matches pit two identical versions of our AI. As described in our 2020 and 2022 TDPs [6,7], we use scenario-based simulated tests to automatically verify the correctness of gameplay behaviours. While these simulated tests are effective for debugging and verification of gameplay logic, they do not provide actionable insight into whether a given strategy change measurably improves overall match outcomes. When both teams behave identically, changes to play selection, tactic assignment, or coordination logic cannot be meaningfully compared against a baseline.

Hence, improvements to higher-level behavior are often evaluated qualitatively, either through visual inspection of simulation runs or during real-world scrimmages. These approaches are time-consuming to set up, difficult to reproduce, and poorly suited for systematic experimentation.

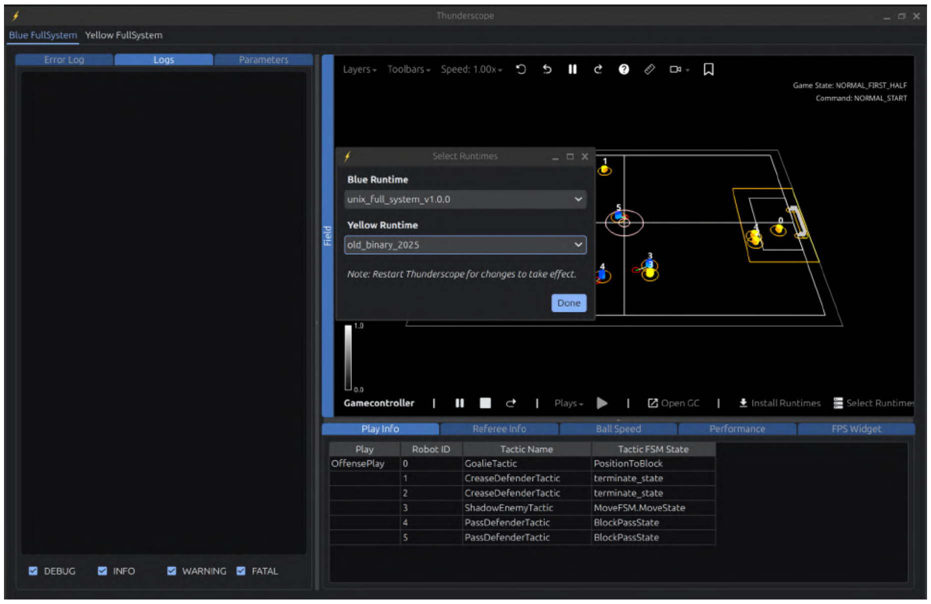
To address these limitations, our team has experimented with two complementary goals over the past year. The first is the establishment of a standardized set of gameplay metrics that allow strategy efficacy to be evaluated quantitatively within simulation. The second is the development of a runtime manager through which different AI implementations can be run interchangeably within the same simulation environment, enabling controlled AI-versus-AI evaluation. Crucially, this includes the ability to run previous releases of our own AI as well as AIs developed by other teams. Together, these components support reproducible A/B testing of strategic changes and provide a foundation for automated evaluation workflows.

Gameplay Evaluation Metrics For gameplay evaluation, we focus on metrics that directly reflect match outcomes and decision quality at the team level. We track friendly goals scored, friendly shots on target, total friendly shot attempts, enemy goals scored, enemy shots on target, and friendly fouls. From these values, we compute interpretable confidence measures such as friendly shot conversion rate, enemy shot conversion rate, and effective defensive blocking rate. In addition, shooting frequency is tracked to contextualize conversion metrics, distinguishing between strategies that score efficiently from few opportunities and those that generate a high volume of attempts. These statistics are intentionally simple and derived directly from vision data and match events reported by SSL Gamecontroller, ensuring consistency across simulators and teams.

All metrics are collected automatically during simulation runs and presented in the simulator. In practice, we have found that these values provide a stable baseline for detecting performance regressions or improvements when comparing two AI variants over repeated simulated matches.

AI Backend Manager To support rapid experimentation, we introduce an external AI backend selection mechanism integrated into Thunderscope, our main GUI application for controlling our software. Rather than requiring all launched AIs to be built and executed from the same source tree, users select AI backends through the Thunderscope UI, each of which defines how a particular AI process is launched and managed. This functionality is implemented in Python via an abstract `RuntimeManager` interface, which encapsulates the logic required to start, stop, and monitor an AI process that communicates using the SSL Simulation Protocol. This design decouples AI execution from the developer's local build environment and removes the need for ad hoc process management when running AI-versus-AI simulations.

Thunderbots AI binaries are compiled via automated build pipelines and published as versioned releases on GitHub, and developers can install these binaries



Game State: NORMAL_FIRST_HALF
Command: NORMAL_START

Blue Runtime: unix_full_system_v1.0.0

Yellow Runtime: old_binary_2025

Note: Restart Thunderscope for changes to take effect.

Play	Robot ID	Tactic Name	Tactic FSM State
OffensePlay	0	GoalieTactic	PositionToBlock
	1	CreaseDefenderTactic	terminate_state
	2	CreaseDefenderTactic	terminate_state
	3	ShadowEnemyTactic	MoveFSM.MoveState
	4	PassDefenderTactic	BlockPassState
	5	PassDefenderTactic	BlockPassState

Fig. 11. UI for selecting AI backends through the Thunderscope interface

directly through the Thunderscope interface. These binaries are stored under a dedicated directory which is scanned at runtime to discover available options and present them to the user for selection. This allows developers to quickly compare local changes against known reference versions without rebuilding historical commits or maintaining multiple checkouts.

The `RuntimeManager` abstraction is intentionally open and does not assume a fixed language, build system, or packaging format. AI runtimes are defined solely by their launch configuration and their adherence to the SSL Simulation Protocol, enabling integration of heterogeneous systems such as TIGERs Mannheim's Sumatra AI [8], which is implemented in Java and launched via a Gradle-based execution workflow. This flexibility allows us to easily evaluate strategy changes against AIs developed by other teams.

Applications for RL This evaluation framework directly supports reinforcement learning workflows. In our 2025 TDP [1], we presented a reinforcement learning framework for training an attacker agent using simulation-based rollouts. By standardizing AI-versus-AI simulation and metric collection, we can craft reward signals based on match outcomes such as goals scored, shots generated, and goals allowed. The ability to interchange AI backends further allows training and evaluation against a diverse set of opponents, including different versions of our own AI and AIs developed by other teams, exposing learned policies to a wider range of strategic behaviours and difficulty levels during training.

The ability to perform controlled, repeatable AI-versus-AI experiments in simulation significantly improves confidence in strategy-level changes prior to deployment on physical robots. Our framework addresses a common pain point across SSL teams and encourages more systematic and reproducible strategy evaluation. We anticipate that this functionality can be extracted into a standalone tool which we will make open source to further support cross-team experimentation within the league.

5 Conclusion

We believe that the design changes detailed above will lead to significant improvements in both performance and maintainability of SSL robots. We look forward to implementing these designs at RoboCup 2026.

6 Acknowledgements

We would like to thank our sponsors as well as the University of British Columbia, namely the Faculties of Applied Science and Science and the departments represented within our team, including Mechanical Engineering, Computer Science, Electrical and Computer Engineering, Engineering Physics, Integrated Engineering, and Chemical and Biological Engineering. Without their support, developing our robots and competing at RoboCup would not be possible.

References

1. A. Bahrami, A. Abousaleh, A. Yip, A. Shen, A. Sidhu, A. Balamurali, A. Guha, C. Lee, D. Sametoglu, E. Chiu, G. Foo, J. Coffin, J. Jung, L. Chee, L. Song, L. Li, M. Rillera, P. Zhou, R. Khan, R. Nedjabat, S. Srivastava, S. Raja, T. Kong, T. Frew, T. Yang, V. Ko, W. Grellier, W. Ha, Y. Zhou, and Y. Elhagrasy, "2025 Team Description Paper: UBC Thunderbots," 2025.
2. T. Hirohashi, R. Hotta, T. Tasaka, Y. Matsumoto, K. Wakabayashi, D. Tomioka, Y. Naito, K. Mitsuishi, T. Obara, A. Mitsushima, R. Murai, S. Okunishi, K. Ochiishi, and S. Otake, "RoboCup 2023 Team Description Paper Ri-one," 2023.
3. A. Ryll and S. Jut, "Extended Team Description for RoboCup 2020," 2020.
4. V. Monajjemi, A. Koochakzadeh, and S. S. Ghidary, "grsim - robocup small size robot soccer simulator," in *RoboCup*, 2011.
5. P. Bergmann, T. Engelhardt, T. Heineken, V. Hopf, M. Schmid, M. Schmidt, F. Schofer, K. Schuh, and M. Stadler, "Er-force 2022 extended team description paper," 2022.
6. P. Dumitru, G. Ellis, J. Fink, B. Hers, J. Lew, M. MacDougall, E. Morcom, S. H., C. Sousa, W. Van Dam, G. Whyte, L. Zhang, S. Zheng, and Y. Zhou, "2020 Team Description Paper: UBC Thunderbots," 2020.

7. A. Almoallim, C. Sousa, D. Sturn, D. Antoniuk, F. Crema, H. Bryant, J. Lew, J. Liu, K. Wakaba, L. Bontkes, and Y. Zhou, "2022 Team Description Paper: UBC Thunderbots," 2022.
8. "Sumatra - central ai of tigers mannheim." `https://github.com/TIGERs-Mannheim/Sumatra/`, 2026. GitHub repository.