First Order Robotics Team Description Paper

Small Size League of RoboCup 2026

Martin Liang, Joel Ng, Louis Yong, Fred Huang, Pi Rong Koh, Samuel Corbett, Han Li, Terry Yu, Lin Jiayu, Phuong Vo, Thomas Aldrian, Ezekiel Lim, Valentin Bruhl, Sarthak Agarwal, Yangping Li, Ningchuan Wang, Sze Yoong Low, Andrew Jiaming Liu, and Sadig Sadikhzada

Imperial College London, Exhibition Rd, South Kensington, London, UK
firstorderrobotics@gmail.com

Abstract. This paper describes the efforts and improvements made by the First Order Robotics Team to compete in the 2026 RoboCup Small Size League Division B. It highlights key hardware upgrades, including a compact PowerCore, optimized dribbler, and enhanced communication and LED systems. On the software side, the team implemented FreeRTOS for task management, ROS 2 for monitoring and deployment, low-latency on-board vision, and a unified Behaviour Tree strategy framework for motion planning and defensive coordination, improving reliability, modularity, and performance across simulation and real-world robot platforms.

1 Introduction

First Order Robotics is a student-run group from Imperial College London, operating under the Imperial College Prototyping and Hacking Society (ProtoSoc). Established in 2024, the team successfully qualified for the RoboCup SSL 2025, though funding constraints prevented participation. This year marks the second year of development, during which the team has made significant improvements to enhance the reliability and capabilities of its robotic systems.

2 Mechanical System

The 2024–2025 mechanical design provides a solid foundation for general robot mobility, and the overall top-down structural stack concept remains valid. However, several issues were identified:

1. **Electronics stack height:** The electronics stack-up adds excessive height and very close to the SSL rule limit.
2. **Internal routing:** Internal cable routing was not sufficiently considered, leading to poor cable management.
3. **Solenoid optimisation:** The solenoid design could be improved by using magnetic simulation to select appropriate coil parameters.
4. **Manufacturing approach:** Many components could be moved from a 2D carbon-fibre & water-jet-cut aluminium stack-up to CNC-milled parts, simplifying the structure and improving dimensional tolerance and accuracy.

2.1 Top-down mechatronic stack optimization

To reduce the overall stack height and simplify internal wiring, we made two key hardware decisions. First, we tightly integrated the capacitor board with the high-voltage (HV) kicker board, which substantially reduced the vertical build-up. Second, to eliminate the messy power-distribution cabling, we consolidated power distribution onto a single board. The detailed implementation is described in the Subsection 3.1.

The resulting design is shown in Fig. 1. As illustrated, the rear electronics stack height is reduced by approximately 15 mm compared with the previous iteration. Overall, this greatly improves the simplicity, accessibility, and serviceability of the top-level electronics.

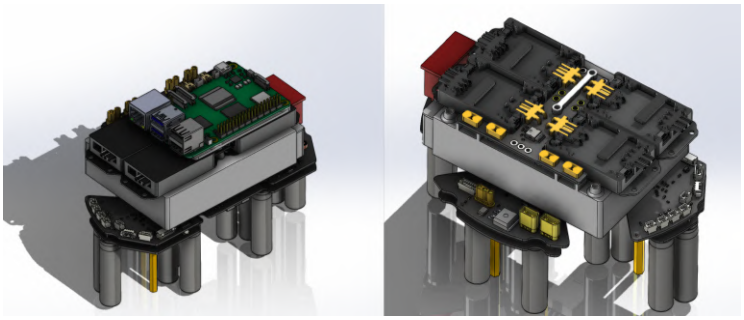


Fig. 1: CAD render of PowerCore electronics stack

2.2 Solenoid optimization with simulation

The solenoid design must provide two key parameters that directly influence both software decisions and HV kicker-board design: (i) the ball exit velocity (for path planning), and (ii) the coil resistance (for selecting appropriate power components in the discharge path). This year, we developed a workflow that leverages MATLAB and JMAG (magnetic simulation) to guide the design quantitatively. The workflow is outlined below:

1. **Define the geometric constraints.** Determine the maximum physical volume available for the coil winding, typically constrained by the wheelbase layout and surrounding components (e.g., motors and plunger length). In our case, the allowable winding region is a cylinder with an inner diameter of 10.3 mm and an outer diameter of 40 mm.
2. **Parameterise the winding and estimate resistance.** Treat the enameled copper wire diameter as a design variable. We then use a MATLAB script to compute the number of turns and total wire length using an orthocyclic winding model (illustrated in Fig. 2) with a packing factor of 90.7% (or lower). From the total wire length and cross-sectional area, the coil resistance is estimated.

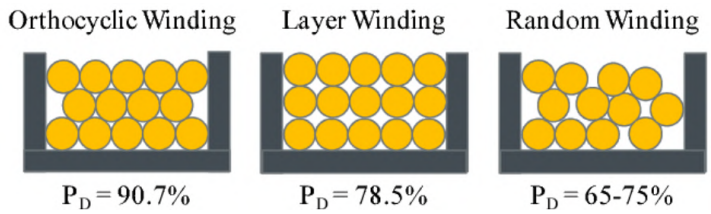


Fig. 2: Various coil packing scheme and corresponding packing factor

3. **Search for a near-optimal design candidate.** The maximum charging voltage is limited by the capacitor rating on the kicker board, while the peak discharge current is limited by the discharge MOSFET. The discharge can be roughly approximated as an RLC transient. However, the full system is mechanically coupled (moving plunger), nonlinear, and time-varying, making direct optimisation difficult. Instead, we use a quasi-steady approximation $I \approx V/R$ together with the turn count N to evaluate the metric K = N I. Since the magnetic force generally increases with both N and I (which are often in tension with each other), K provides a practical proxy for comparing candidate designs. We therefore enumerate commercially available wire diameters in MATLAB and select the combination that maximises K under the geometric and electrical constraints.
4. **Validate with magnetic simulation.** Using the selected turn count and estimated current, we build the corresponding model in JMAG to simulate the transient force profile and estimate the impulse delivered to the ball. An example simulation and result are shown in Fig. 3.

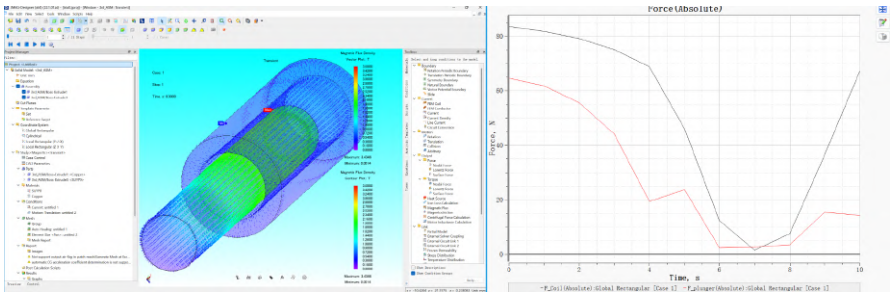


Fig. 3: JMAG solenoid force output simulation

Although this workflow is simplified, it provides a quantitative basis for selecting which enameled wire to purchase and for estimating the expected performance. Due to time constraints, the method has not yet been fully validated experimentally, but it offers a strong starting point for systematic iteration.

2.3 Dribbler improvement

Two major changes were introduced to the dribbler design:

1. **Angle feedback using Hall element sensing.** Our dribbler control scheme requires feedback of the dribbler outer-frame angle to approximate the ball "feed-in" state. Last year, this angle was measured using a parallelogram linkage between a rotary encoder and the dribbler frame. However, this approach was sensitive to assembly tolerances and suffered from backlash in the linkage. In addition, the small rotary encoder provided incremental (rather than absolute) position feedback, which is not ideal for robust control.
   To address these issues, we replaced the mechanism with a Hall-sensor and magnet-based feedback system. As shown in Fig. 4, a radially magnetised N52 magnet is rigidly mounted to the shoulder screw at the centre of rotation. A linear Hall sensor is positioned above the magnet and aligned with the magnetic field direction. The sensor output is amplified using a single-stage, non-inverting op-amp with low temperature drift, and scaled to best utilise the 3.3 V ADC input range.

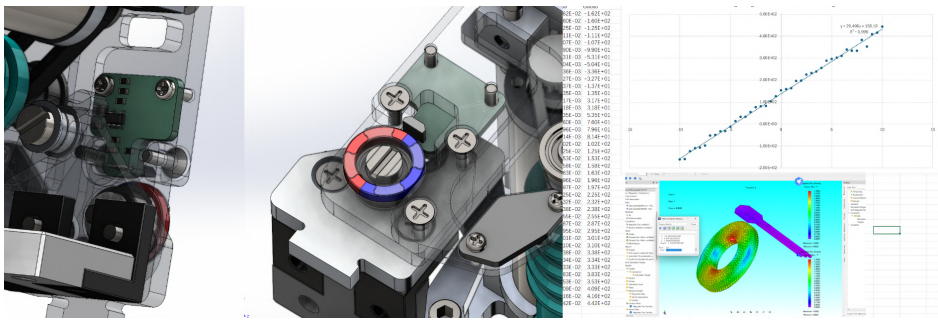


Fig. 4: CAD render of the dribbler hall sensor and the magnetic reading simulation

   We used JMAG to identify the region with the highest magnetic-field linearity (Fig. 4 right) and to estimate the Hall sensor's raw output level, enabling a more accurate selection of the amplifier gain.
2. **Mechanical redesign.** With access to CNC milling this year, we transitioned from last year's carbon-fibre-oriented design to a precision-machined 6061 aluminium-alloy structure, improving dimensional accuracy while also increasing design freedom. In addition, we replaced the gear train with a belt transmission system to avoid the large torque amplification introduced by a high step-down gear ratio.

The motor mounting bracket is designed to slide rearwards, providing straightforward belt tension adjustment. The top damper was also replaced with a single, larger damper featuring a sealed internal oil cavity and M3 mounting points, improving ease of installation and maintenance. The overall assembly is shown in Fig. 5.

Fig. 5: CAD render of the overall dribbler design

3 Hardware

3.1 PowerCore

As discussed in the mechanical section, a major update to the PowerCore is the re-layout and tighter integration of the kicker board with the capacitor array. The primary objective was to reduce the overall stack height by moving tall components to the rear side of the kicker board and adding a cut-out in the capacitor board to accommodate these back-side components. In addition, the new PowerCore integrates whole-robot power monitoring features, including: (1) a VBAT ADC measurement for low-battery warning, and (2) two current-sensing paths for power metering 24 V power track for battery charging and discharging. To further simplify internal routing, we also designed a new power-distribution board featuring horizontally oriented XT30 connectors. This allows the motor ESCs to plug in directly, eliminating the need for bulky and messy wiring harnesses. In addition, a MOSFET power switch is integrated on the distribution board and controlled by a 3.3 V MCU logic signal from the kicker board, enabling more advanced safety features such as an emergency movement shutdown. The updated kicker board PCBA and the distribution board layout are shown in Fig. 6.

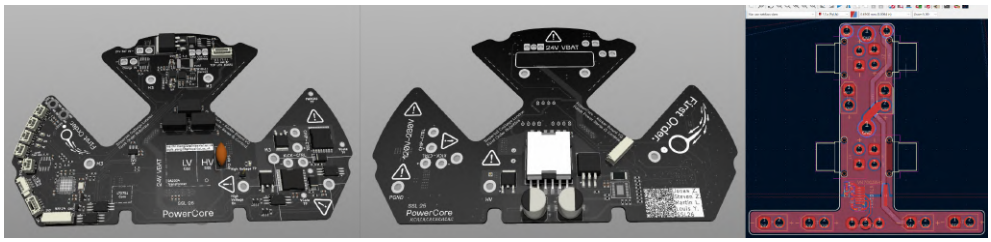


Fig. 6: Digital Render of kicker board PCBA and the distribution board layout

3.2 Capacitor Array

The new capacitor board replaces the previous 12-parallel capacitor array design to introduce variable kick strengths. It also incorporates the safety discharge mechanism, which was relocated from the kicker board, to improve reliability and free up the limited space on the kicker board. The PCBA of the new capacitor array is shown in 7.

Fig. 7: Digital Render of Capacitor Array

3.2.1 Array Architecture The kicker now employs three MOSFET switched capacitor banks, each consisting of four parallel 265µF/330V electrolytic capacitors. This configuration allows for three kick strengths through selective bank discharge: WEAK (discharge capacitor bank 3 only), MEDIUM (discharge capacitor bank 3 and 2) and STRONG (discharge all capacitor banks). Overall capacitor discharge is controlled by the kicker board MOSFET, regardless of individual bank-switch states, shown in Figures 8.

The discharge of all 3 banks provides a theoretical maximum energy of 173J. However, the capacitors are only charged up to 120V, as this voltage is sufficient to propel the ball at a maximum speed of 6.5m/s.

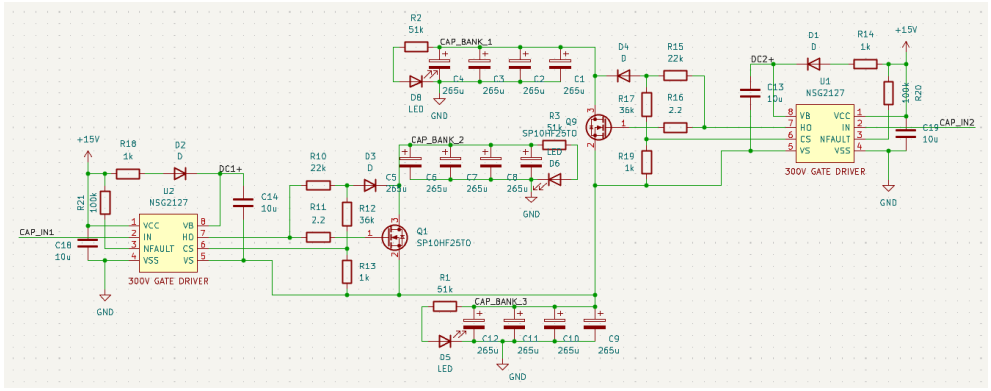
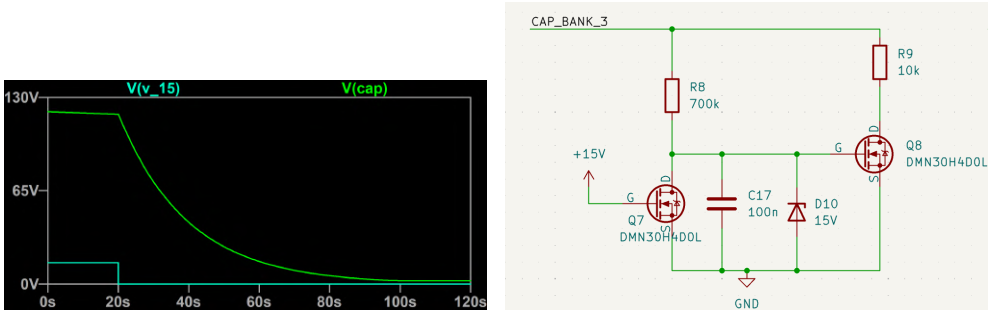


Fig. 8: Digital Render of Capacitor Array

3.2.2 Safety Discharge When the robot is powered off, loss of the 15 V supply independently triggers identical safety discharge circuits on each capacitor bank. The banks discharge from 120 V to $\approx 3\,V$ (limited by the voltage threshold of the DMN30H4DOL MOSFET) in 75s.



(a) LTspice simulation safety discharge (b) Safety discharge architecture

Fig. 9: Safety discharge simulation and architecture.

This safety discharge mechanism introduces minor leakage that requires the charger to replenish continuously. Currently, it is charged automatically every 30s if no action has been taken to ensure that the capacitor banks are always fully charged to kick the ball.

3.3 Transmission Board

3.3.1 Transmission Board This year, we decided to design and manufacture a custom transmission board for communicating with the robots. This board is connected to the host PC via USB, with one NRF24 module per robot. This allows us to minimise delays due to transmit commands and switching channels, as commands can now be sent to each robot in parallel.

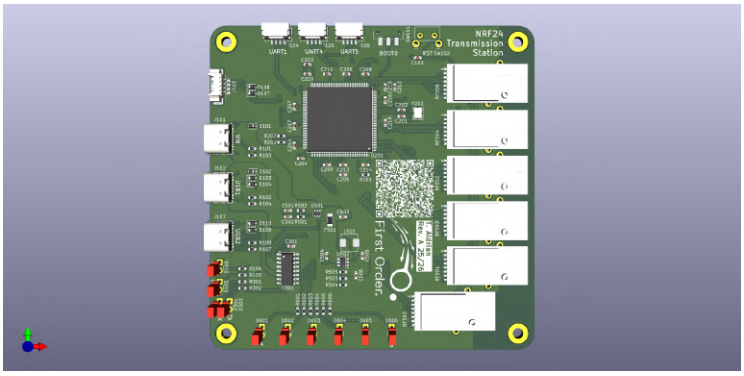


Fig. 10: Digital Render of Transmission Board

Due to the potential power usage of all NRFs transmitting simultaneously being higher than 500mA (the maximum for USB2.0), we have a secondary USB-C connector for power input. These ports are multiplexed by a TPS2116 chip, which also provides reverse current protection. There is also extra protection with an efuse, and TVS diodes for transient voltage protection. We have also broken out multiple UART ports from the MCU, including one with USB-TTL adapter for redundancy. The decision was made to have no buttons or physical controls, as it was decided this would be easier to implement on the host PC, and we wanted to reduce possible failure points.

3.3.2 Transmission and Receiver on Robot Last year, we selected the NRF24L01 as the RF solution for communication between the controller PC and each bot. A PCB was designed with this chip and its peripherals, with signals output through an on-board PCB antenna. This setup has proven effective, and we continue to use this model this year. Two improvements have been proposed:

– To enhance transmission stability and ensure a secure connection, each NRF24 module is now decoupled from the main control PCB and installed in a dedicated area, making it easier to replace if damaged. Additionally, each module is equipped with a power amplifier, low-noise amplifier, and an SMA antenna connected via an IPEX coaxial cable.
– Each robot is now equipped with two NRF24 modules, with one dedicated solely to receiving (Rx) and the other to transmitting (Tx). This allows for greater communication bandwidth. Although considerable redundancy is reserved for current software development, this setup ensures sufficient data transfer bandwidth in the future.

3.4 LED Robot ID Display

To streamline testing, a custom overhead LED display was developed to facilitate rapid robot ID reconfiguration. The system utilizes SK2812 LED back SMD modules paired with a diffusion layer to provide the uniform color consistency required for reliable capture by the vision camera. While the hardware employs a cascaded data architecture (see Fig. 11) and parallel power distribution to simplify control, the high current draw, around 3A, imposes a practical limit on the total number of LEDs.

The top LED display utilizes a dedicated MCU board to maintain a modular and self contained control architecture (see Fig. 12). Because robot IDs are assigned statically before a competition begins, the LED system functions autonomously, eliminating the need for real time communication with the main control system. This innovation is mainly used to speed up internal testing as rules state that only paper is allowed for robot ID during competition.

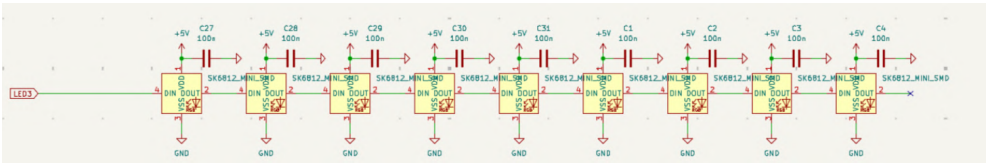


Fig. 11: Cascading LED structure

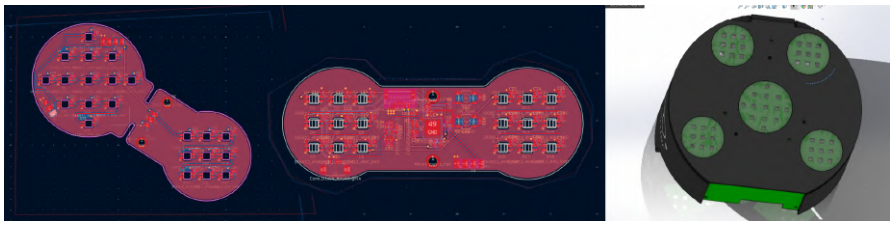


Fig. 12: LED board PCB layout

4 Embedded Systems

4.1 Code Rebase with FreeRTOS

Compared to our previous TDP we have made significant improvements to our embedded codebase by employing FreeRTOS in place of a monolithic polling control loop. We split this loop into a few distinct tasks:

– **Movement** which implements a PID controller, and sends corrected current values to each motor

- **Communication:** receives and decodes data from the NRF24 module.
- **Kicker:** communicates to the kicker board over UART.
- **Dribbler:** controls the speed of the dribbler.

Compared to the previous polling loop architecture, this FreeRTOS approach allows tasks to block whilst waiting on I/O or events. This frees up CPU time for other work, enabling higher frequency update loops for time critical tasks (e.g. motor velocity updates). This lowers overall system latency, improving responsiveness to the commands issued. FreeRTOS also promotes modularity, where tasks can easily be enabled or disabled. Tasks can also be shared as modules between both the robots and the transmission board.

FreeRTOS is particularly effective at improving latency on the transmission station. Parallel utilisation of the NRF modules significantly reduces CPU time spent servicing each NRF, allowing packets to be routed with minimal delay.

4.2 Low-Level Computer Vision

As the overhead camera system is susceptible to inaccuracies and vision dead zones, on-board computer vision is used to enhance ball detection and tracking. A Raspberry Pi 4 paired with an IMX219 Raspberry Pi Camera Module v2 is used to detect the ball position in front of the robot. This on-board vision system enables low-latency visual tracking while reducing communication network load.

Ball identification is achieved through applying a color mask in HSV color space, which undergoes denoising by identifying the largest connected sets of pixels within the mask. Ball tracking is achieved through a hybrid set of algorithms that identifies a circular contour around the ball. For the majority of the camera frame, the contour found was the minimum enclosing circle (MEC) that encloses all the masked pixels. This method, however, fails around edges, where the center of the circular contour falls outside the camera frame. The second strategy of ball tracking relies on identifying edge pixels, then using a Random Sample Consensus (RANSAC) algorithm to fit a circle based on these edge points (seen in Figure 13) [2].

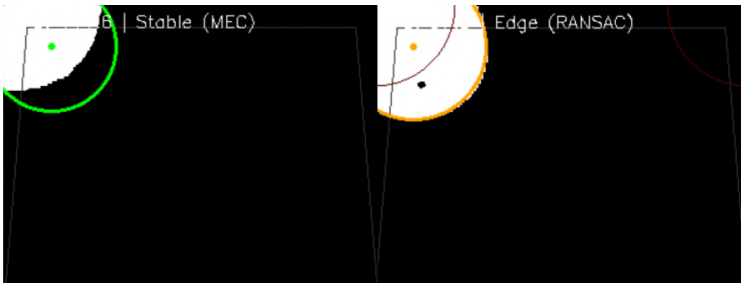


Fig. 13: Failure of MEC tracking near the top corner, and correction after switching to RANSAC sooner inside new circular boundary

As RANSAC is inherently more unstable than MEC, additional robustness constraints were incorporated, such as restricting the maximum predicted contour displacement across frames. This system will be fully integrated into an autonomous ball-chasing algorithm in future work.

4.3 ROS 2 Infrastructure for Monitoring and Deployment

To improve observability across the robot stack, we developed a unified ROS 2-based integration framework spanning the host workstation (Ubuntu 22.04) and the onboard Raspberry Pi companion computer. The framework enables fine-grained monitoring of subsystem state, enforces consistent launch configurations across robots, and supports scalable firmware flashing and software deployment in multi-robot setups.

Leveraging ROS 2's distributed communication model, each robot initialises a deterministic runtime configuration on power-up by reading a hardware identifier (ID) from the embedded electronics. This ID is used to assign namespaces, parameters, and topic/service names before launching a communication node that bridges robot telemetry and control interfaces to ROS 2 messages. The node publishes a standardised set of initialisation and status data, enabling automatic robot discovery and registration on the host. ID-based namespacing prevents topic collisions and ensures that monitoring, logging, and control pipelines scale cleanly with robot count.

On the host, robot registration triggers automated launch routines for visualisation and inspection. Foxglove is integrated as a unified dashboard for real-time monitoring of vision streams and per-robot telemetry, including state estimates and subsystem diagnostics [3]. This operator-centric interface enables early detection of anomalous behaviour (e.g. communication degradation or sensor dropouts) and accelerates testing and iteration. The ROS 2 launch system further provides a single, reproducible entry point for complete experimental configurations, with optional data recording for post-run analysis.

5 Software

5.1 Unified Strategy Runner

As the team executes its strategies both in simulation (grSim and rSim) and on real hardware, cross-platform compatibility is a critical consideration [7, 6]. Due to small but distinctive differences in platform interfaces, the team initially maintained diverging codebases for testing on each platform. This resulted in code inconsistencies, version control challenges, and increased software bloat.

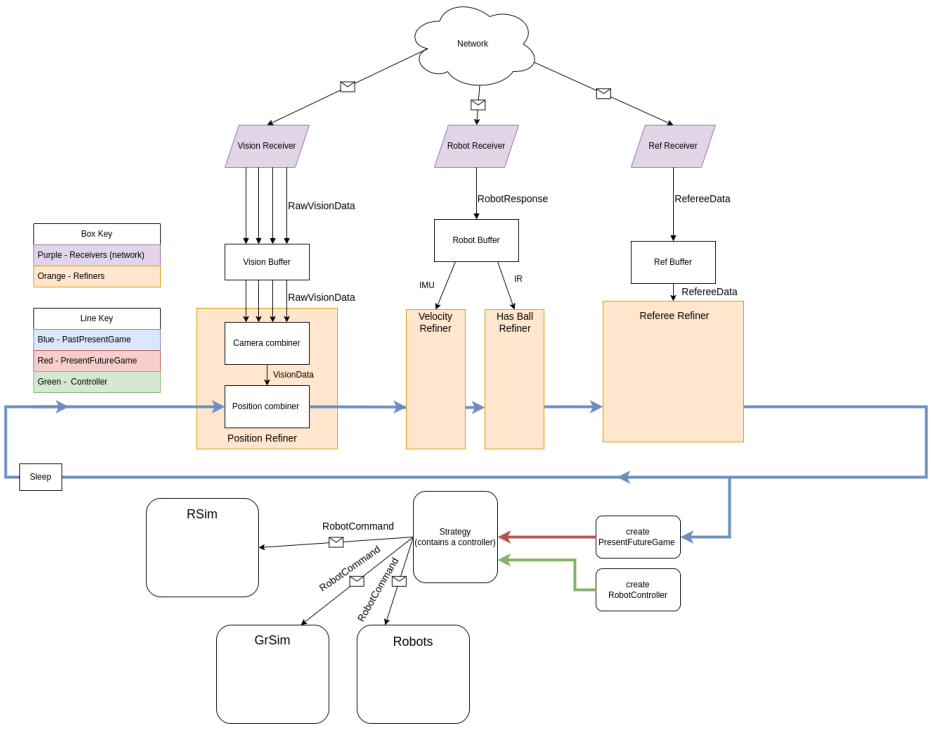


Fig. 14: System design drawing of Unified Strategy Runner (by Hamish Starling and Luke Moran in 2025)

To resolve this, we established a unified system to enforce data standardisation across platforms (seen in Fig. 14). We store all our information in a data frame, where it is populated and updated in our `Refiners` on each timestep before being used for strategies. Crucially, the refinement step is performed in the following order:

1. Position data from the vision source (SSL-Vision or the rSim environment).
2. Robot information, including IMU and IR sensor data, used to update velocity estimates and ball possession status. For rSim, this data is obtained from the simulation environment.
3. Referee information received from AutoReferee, which is used to override the strategy logic. The rSim environment does not provide referee information.

This allowed the team to work on a unified Strategy framework (discussed in Subsection 5.4) that could easily be tested across all three platforms.

5.1.1 Replay System One of the new features incorporated into the Unified Strategy Runner was a replay system compatible with both simulation and real-world execution. During runtime, the Strategy Runner streamed and recorded reduced game frames, which could later be replayed using an rSim renderer. This system also supported step-through playback, enabling frame-by-frame inspection of robot behaviour.

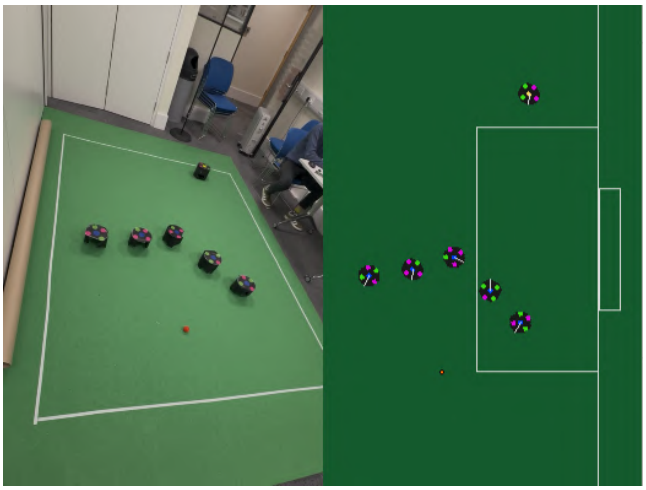


Fig. 15: Side-by-side comparison of real-game footage and replay rendering

5.2 Motion Planning

5.2.1 Fast Path Planning For motion planning, we implement a global planner based on the Fast Path Planning (FPP) approach proposed by Rodriguez et al. [8]. The planner constructs a piecewise-linear trajectory between the start and goal positions by recursively subdividing straight-line segments that intersect obstacles. An initial straight-line segment is first checked for collisions; if collision-free, it is returned directly. Otherwise, a subgoal is generated, splitting the segment into two sub-segments that are evaluated recursively until a collision-free path is obtained (Fig. 16).

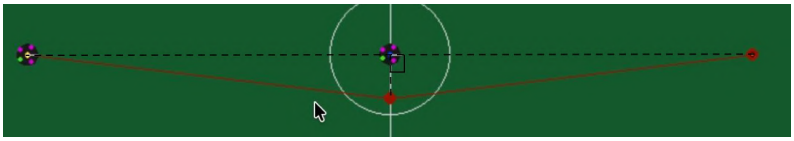


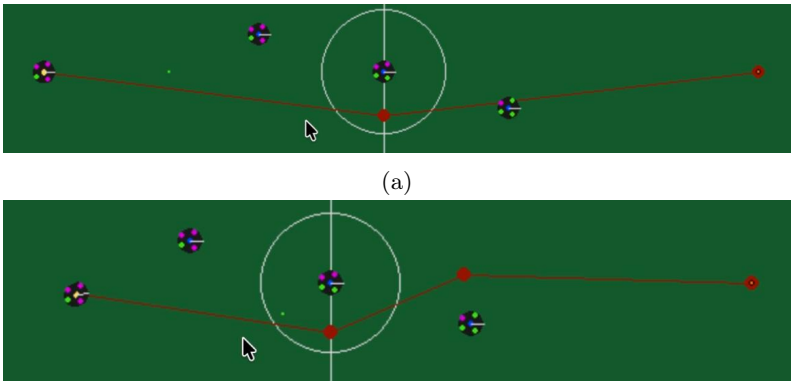
Fig. 16: Creation of a new path using a subgoal

In our implementation, subgoals are generated at a fixed offset of three robot diameters from the closest obstacle along the colliding segment, at angular offsets of $\pm 90^\circ$ relative to the segment direction. If the generated subgoal is in collision, it is

iteratively displaced further away from the obstacle. Subgoals are generated for both angular directions, and the resulting candidate paths are evaluated, with the shortest collision-free path selected. This differs from the original FPP formulation, which does not prescribe a fixed-distance or bidirectional subgoal generation strategy.

Unlike the original FPP algorithm, which considers all obstacles globally, our planner introduces a proximity constraint to account for dynamic obstacles such as friendly and enemy robots. Only obstacles within a 3 m radius of the robot are considered during planning, reducing unnecessary replanning caused by distant robots whose future positions are uncertain (Fig. 17). As the robot moves, obstacles enter and exit this proximity region, triggering local trajectory updates.

To further improve robustness in dynamic environments, obstacle motion is explicitly modelled by incorporating obstacle velocity. Each obstacle is represented as a line segment spanning its current position and a predicted future position over a short horizon (Fig. 18). Collision checks are performed against these line-segment obstacles, enabling limited predictive avoidance. This velocity-aware obstacle representation is not included in the original FPP formulation.



(a)

(b)

Fig. 17: In (a), the trajectory generated by the FPP algorithm ignores the presence of the rightmost enemy robot, as it is out of range. When the robot moves closer in (b), the enemy robot enters the proximity range and a new trajectory is generated to include it as an obstacle.

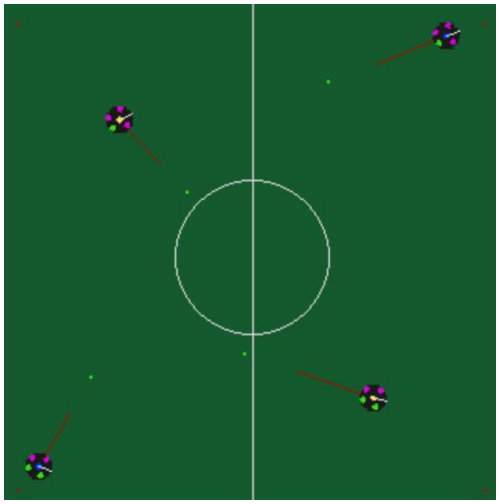


Fig. 18: Representation of Obstacles as a line segment

5.3 Data Processing

5.3.1 Kalman Filter Random noise from the overhead camera can interfere with our motion planning algorithms and cause our robots to deviate from their path, move erratically, and fail to reach their targets.

We addressed this by applying a Kalman filter to the data before it is passed to the motion planners [1]. It first uses the robot's last known velocity and kinematics formulae to estimate its current position. Then, it updates its estimate when it receives new vision data, weighing the information based on the system's noise level.

We validated the filter's effectiveness with an rSim simulation of a robot cycling among 10 targets with Gaussian noise (standard deviation of 10 cm) added to clean vision data. The filter achieved a 35% improvement in accuracy, measured by the decrease in the residual vector.

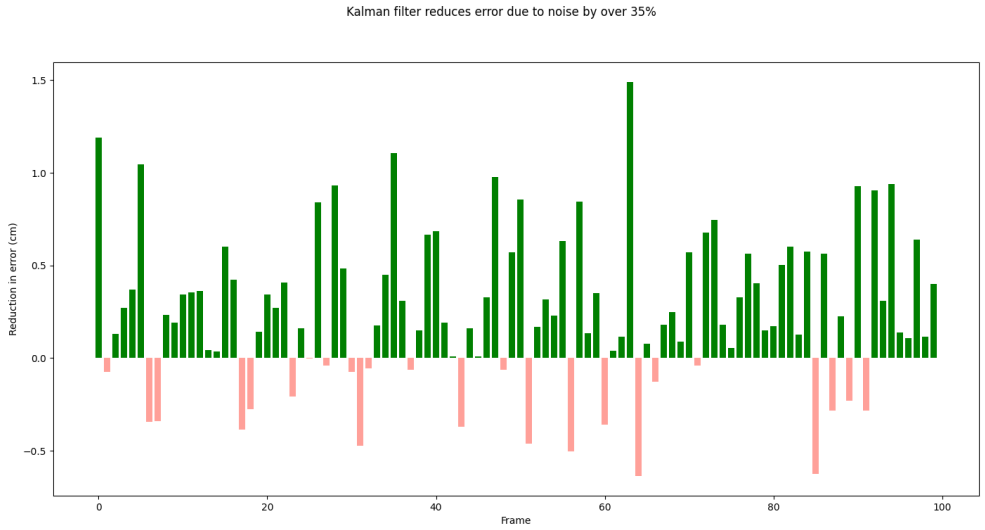


Fig. 19: Kalman filtering reduces the error in position data due to random noise. The above is a random subset of the aforementioned simulation data.

5.4 Strategy Design

Previously, our strategy layer relied on finite state machines using the STP framework [4, 5]. However, it became increasingly difficult to maintain as complexity grew. To facilitate modularity and reusability, we have migrated to a Behaviour Tree (BT) architecture shown by Fig. 20, using the `py_trees` library [9].

We have structured our BT framework into three core hierarchical components to ensure scalability and reusability:

- **Strategies:** These represent the top-level execution units (the Object that is used during a match / runnable modules for sim testing). A strategy acts as the root of the tree, orchestrating the overall behaviour for a specific set of robots for the given game situation.
- **Plays:** These are modular, self-contained subtrees designed to be used in conjunction with other trees. A single Strategy often comprises multiple Plays, allowing us to compose complex behaviours from pre-tested logic blocks without redundancy.
- **Nodes:** These are the atomic building blocks of our trees, strictly separated into two main pillars that execute per tick:
  - **Action nodes:** Leaf nodes responsible for executing commands (e.g., `MoveStep`, `KickStep`).
  - **Condition nodes:** Leaf nodes responsible for querying the state of the world (e.g., `HasBall`, `GoalScored`) to guide control flow.

For shared variables across the tree, we utilised a shared `Blackboard` to manage data flow between these components without tight coupling. For example, a `SetBlackboardVariable` node writes target coordinates or robot IDs, which are subsequently read by action nodes such as `MoveStep` or `GoToBallStep`.

To handle the unpredictable nature of real-world matches, we employ Sequence, Selector, and Parallel composites to define logic flow [9]. Unlike finite state machines, which require more explicit transitions for every possible interruption, these composites allow us to construct priority-based hierarchies. Selectors enable immediate fallback routines if a preferred action fails, while Sequences enforce strict preconditions. For instance, by ensuring specific parameters are set before a skill like `dribble_subtree` is executed, Fig. 20. This structure allows the system to gracefully recover from failures, such as losing the ball mid-dribble, by automatically switching behaviours without complex error-handling code.

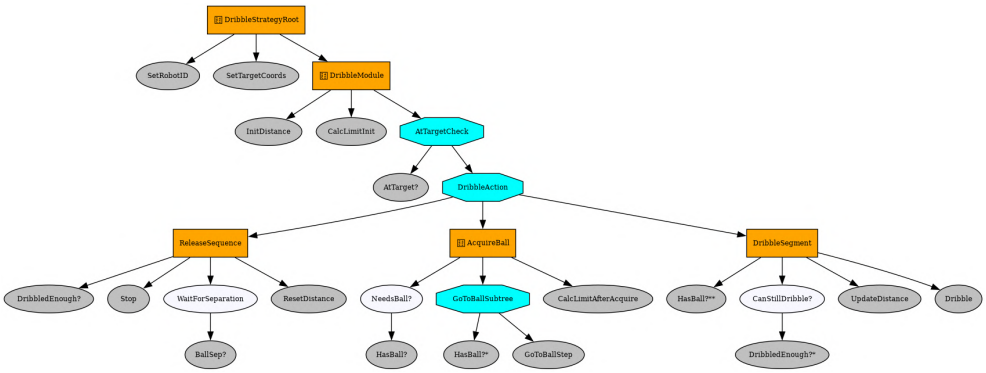


Fig. 20: Dribbler Strategy

5.5 Simulator (rSim) Adaptations for Dribbling

To support the development of active ball handling, we extended rSim [6] to incorporate dribbler mechanics. A key limitation of the baseline physics engine was its inability to naturally model the conservation of momentum upon ball release.

To address this, we implemented a momentum approximation in which the environment computes a `release_kick` vector. This vector applies a forward impulse to the ball via a momentary kick command proportional to the robot's forward velocity at the moment of dribbler disengagement. This mechanism enables us to validate our DribbleStrategy in a risk-free Software-in-the-Loop (SIL) environment, serving as an essential verification step before deployment to more advanced simulators or physical hardware.

5.6 Core Subtrees

Building on last year's strategy design [4], we migrated our system to a Behaviour Tree (BT) architecture to improve modularity, reusability, and long-term maintainability. This refactoring allowed the strategy to be decomposed into a set of core subtrees that capture the fundamental skills required for gameplay: robot movement (including controlled dribbling) and ball interaction (passing and shooting).

5.6.1 Dribble The dribble behaviour is designed to comply with competition rules that limit continuous ball possession to one metre. When the target location exceeds this distance, the motion is segmented into multiple dribble cycles. Each cycle consists of acquiring the ball, advancing while tracking travelled distance, releasing the ball upon reaching the limit, and reacquiring it after sufficient separation. This alternating strategy enables reliable long-distance ball transport while remaining rule-compliant.

5.6.2 Passing Passing was redesigned from a monolithic if-else structure into a modular BT composed of reusable sub-behaviours. This improves clarity, robustness, and extensibility. Parallel nodes allow coordinated actions such as simultaneous alignment of the passer and receiver, while fallback nodes provide graceful recovery from partial failures (e.g., loss of possession or interception). The resulting structure ensures prerequisite conditions are enforced while maintaining responsiveness under uncertainty. Iterative refinement led to the inclusion of dynamic interception and recovery behaviours for the receiver, significantly improving reliability in non-ideal conditions.

5.6.3 Score Goal The scoring behaviour was migrated from a finite state machine to a BT, enabling improved robustness through the use of fallback conditions. A ray-casting-based heuristic is used to estimate scoring likelihood by identifying blocked and open regions of the goal based on opponent positions, assuming straight-line ball trajectories and circular robot geometry. This estimate is combined with additional metrics such as distance to goal to determine whether to attempt a shot or defer to alternative behaviours such as passing or dribbling.

5.7 Basic Defence Strategy

The basic defence strategy is designed to prevent opponents from scoring while enabling smooth transitions to offence. Three robots cooperate in this strategy: two defenders and one goalkeeper.

Defenders are responsible for intercepting threats outside the penalty area to reduce pressure on the goalkeeper, while also attempting to regain possession when possible. Entering the penalty area is strictly prohibited for defenders. The goalkeeper operates primarily within the penalty area and cooperates with the defenders to minimise the probability of conceding a goal.

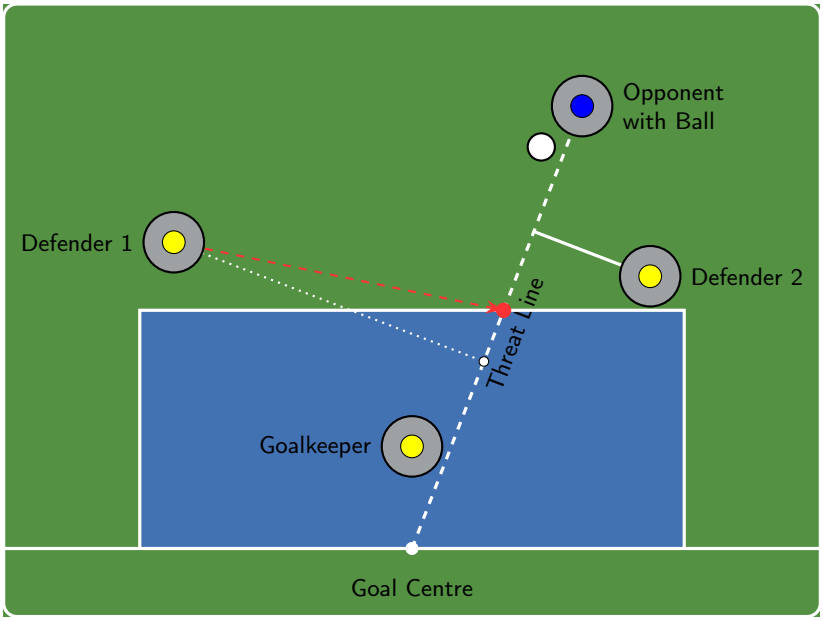


Fig. 21: Geometric construction with penalty area constraint clamping.

As illustrated in Fig. 21, defensive positioning is computed using a geometric projection onto the threat line defined between the ball-carrying opponent and a reference point (typically the goal centre). The defender targets the foot of the perpendicular projection onto this line, allowing effective interception independent of the opponent's orientation.

To ensure rule compliance, defenders are prevented from entering the penalty area through geometric constraints. Illustrated by "Defender 1" in Fig. 21, if the computed

interception point lies within the penalty area, it is projected onto the nearest boundary point along the same threat line (red dot), yielding a safe interception target on the penalty box edge. This ensures that defenders remain legally positioned while still blocking direct shooting or passing lanes.

5.8 Goalkeeper Strategy

The goalkeeper is constrained to a defensive line aligned with the goal line. Its position is determined by predicting the intersection of the ball's estimated trajectory with this line.

If the predicted intersection lies within the vertical bounds of the goal mouth, the goalkeeper moves to that point to block or collect the ball (Fig. 22). When the prediction is invalid or outside the goal area, the goalkeeper falls back to a conservative position near the centre of the goal. This fallback position is adaptively adjusted based on the projected positions of nearby teammates to maintain robust coverage under uncertainty.

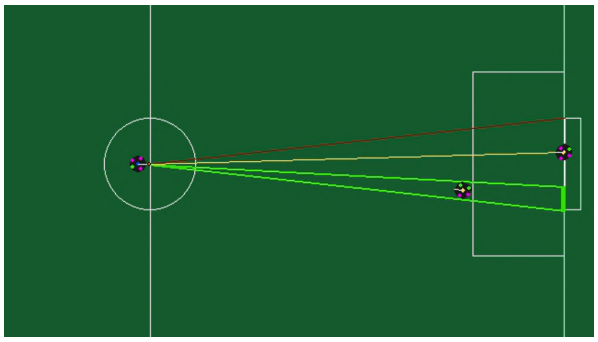


Fig. 22: Goalkeeping strategy illustration.

Motion is executed using a point-to-point controller with dribbling enabled, allowing the goalkeeper to effectively block or secure the ball while maintaining alignment with the goal.

5.8.1 Zonal Division To improve defensive coverage, the two defenders are assigned complementary zones corresponding to the left and right halves of the field. This zonal division ensures that threats originating from either side are addressed promptly, enabling coordinated interception while avoiding redundant coverage.

6 Conclusion

This paper outlined how First Order Robotics has significantly advanced its hardware and software capabilities for RoboCup 2026. Key improvements include mechanical optimisations, a redesigned PowerCore and dribbler system, and enhanced communication and LED displays. On the software side, the team implemented FreeRTOS for efficient task management, ROS 2 for monitoring and deployment, low-latency onboard vision, and a unified Behaviour Tree strategy framework with motion planning and defensive coordination. These developments collectively improve reliability, modularity, and performance, providing a robust platform for autonomous multi-robot gameplay.

References

1. Becker, A.: Kalman Filter from the Ground Up (Third Edition). Becker, Alex (2024)
2. Fischler, M.A., Bolles, R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)
3. Foxglove Technologies: Foxglove studio. https://foxglove.dev (2024)
4. Liang, M., Ng, J., Huang, F., Yong, L., Koh, P.R., Starling, H., Moran, L.: First order robotics team description paper: Small size league of robocup 2025. In: RoboCup 2025 Small Size League Team Description Papers (2025)
5. Lobmeier, C., Burk, D., Wendler, A., Eskofier, B.M.: Er-force 2018 extended team description paper. RoboCup Soccer Small Size League (2018)
6. Martins, F.B., Machado, M.G., Bassani, H.F., Braga, P.H.M., Barros, E.S.: rsoccer: A framework for studying reinforcement learning in small and very small size robot soccer. In: Alami, R., Biswas, J., Cakmak, M., Obst, O. (eds.) RoboCup 2021: Robot World Cup XXIV. pp. 165–176. Springer International Publishing, Cham (2022)
7. Monajjemi, V., Koochakzadeh, A., Ghidary, S.: grsim – robocup small size robot soccer simulator. vol. 7416, pp. 450–460 (06 2012). https://doi.org/10.1007/978-3-642-32060-6_38
8. Rodríguez, S., Rojas, E., Pérez, K., López, J., Quintero, C., Calderón, J.: Fast path planning algorithm for the robocup small size league. In: RoboCup Small Size League Conference / Proceedings (2025), https://sinfoniateam.github.io/sinfonia/papers/paper2.pdf
9. Stonier, D.: pytrees. https://github.com/splintered-reality/py_trees (2025)