RoboJackets 2023 Team Description Paper

Kevin Fu, Alex Sohrab, Prabhanjan Nayak, Mili Das, Ishan Doma, and Bernardo Perez

Georgia Institute of Technology
https://robojackets.org/

Abstract. This paper describes the improvements implemented by the Georgia Institute of Technology's RoboCup SSL team, the RoboJackets, in preparation to compete in RoboCup 2023 in Bordeaux, France. This year's changes to our mechanical and electrical systems primarily focused on improving stability, consistency, and accessibility. On the software side, we changed our main strategy architecture to a agent-based system instead of a top-down hierarchy, for ease of inter-agent coordination.

1 Mechanical

Last season, we aimed to simplify our hardware by consolidating numerous features of our robot. This season, we were able to enact these changes on a full fleet of six robots, as opposed to the 1-2 test robots from last year. We took inspiration from our electrical subteam and added identification numbers on some of our mechanical parts. This change will improve the process for tracking repairs and will ensure that parts are not lost or misplaced.

With six robots, we also had to focus on ease of repair and consistency across the fleet, and this involved more simplification of our hardware. The wire channels through our midplate were simplified for ease of wiring, the super stand pivot point was also shifted back to improve the dribbler's grip on the ball, and the standoffs connecting the kicker board to the midplate were changed to nylon from ceramic.

Below is a table summarizing this year's changes to the 2022 robot fleet: the upper section describes new changes this year, and the lower section describes changes from last year that were applied to all robots on the fleet.

Table 1. Robot mechanical changes and updates from 2022 to 2023

Part	Changes
Super stands	Chipper boot pivot moved 2.4 mm backwards to increase consistency of dribbler
Midplate	Widened openings for motor wires and engraved ID numbers on them to track repairs
Midplate-PCB standoffs	Changed to nylon from ceramic
Shell	Replaced multiple pieces with a single piece shell, vision patterns now on a separate plate that is thumb-screwed onto the shell
Solenoids	Combined into a single unit mounted to the baseplate
Motor stand	Made independent of midplate; now only attached to baseplate

1.1 Dribbler

Our new dribbler assembly from last year made changes to the super stands to improve manufacturability and ease of assembly. However, once a larger fleet of robots was assembled, the new design showed a decrease in the dribbler's ability to control the ball. Closer inspection revealed that the chipper boot was contacting the ball before the roller was, preventing the roller from controlling the ball. We decided to shift the chipper boot pivot back 2.4mm.

To test, we made a 3D-printed prototype. Our tests showed that the dribbler was able to make contact with the ball without the chipper boot interfering, so we ordered full aluminum parts and installed them on the robots.

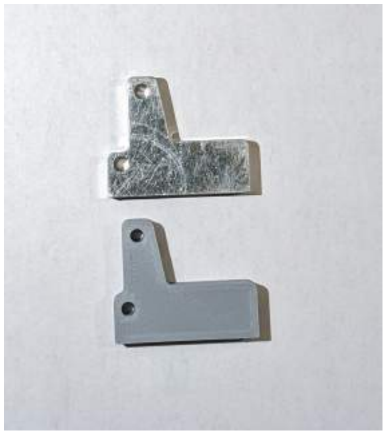

Fig. 1. Comparison of original super stands with 3D printed prototype.

1.2 Chassis

Last year we introduced two new wiring channels to our midplate. Since the design of the robot was being completely changed, the idea of dedicated wire channels for each motor wire was put into practice in the 2022 midplate. However, in practice this change made it harder to assemble robots, as the clips for the motor wires were too wide to be inserted into the channels.

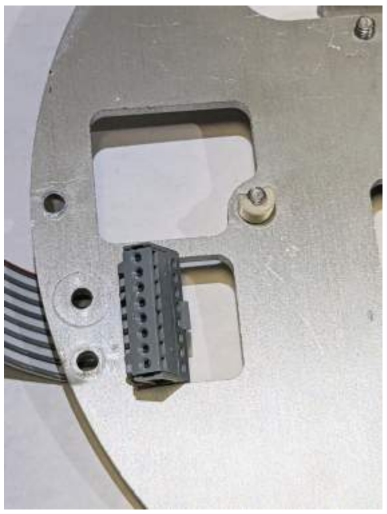

Fig. 2. Comparison of original super stands with 3D printed prototype.

This problem was solved by reverting to the old design, which was one channel for both motor wires on either side. The channels were redrawn to be symmetric across the midline for aesthetics.

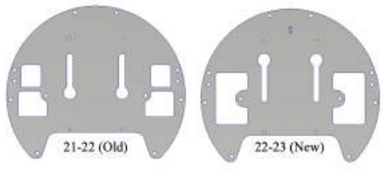

Fig. 3. Comparison of 21-22 midplate with 22-23 midplate.

In addition, another change from 2021-2022 was the usage of ceramic standoffs to connect the midplate and the electrical boards. When maintaining robots with these standoffs, the high friction between the ceramic threads and the metal screws (as well as the standoffs' slippery exterior) made it difficult to remove them, and many had to be snapped off because of this problem. (A snapped standoff is visible in Figure 5.) We considered two possible solutions. One, plastic spacers could be placed on both ends of the standoffs to make them easier to remove. Alternately, another solution could be to replace the standoff material. We decided to go with the latter option. Ceramic is more resistant to deformation under heat and moisture than nylon is, so we may switch to the other option if the nylon standoffs prove unreliable.

1.3 ID System

On a more clerical note, our electrical subteam uses a spreadsheet to keep track of electrical board status, since boards (especially the kicker board) tend to malfunction easily. The spreadsheet keeps track of which boards are working using number labels on each board.

Since we were transitioning to a new design, we took the opportunity to establish a similar system for our robots. This started with etching identification numbers on our midplates. This will likely be most useful with more significant parts/subassemblies such as the solenoids, the dribbler assembly, and the drivetrain. More stable parts such as the shell will likely not need identification numbers.

2 Electrical

This year's modifications to the electrical system primarily focused on rectifying integration issues with our software stack. This includes revisions to the Robot Shell ID Board and the Radio Board. The table below summarizes this year's changes to the electrical system:

Table 2. Robot fleet electrical changes from 2022 to 2023

Part	Changes
Robot Shell ID Board	Replaced Ambient Light Sensor with RGB Sensor
Radio Board	Radio Module Replaced

2.1 Robot Shell ID

The original idea for this board came from TIGERs Mannheim [1]. Previously, our Robot Shell ID board automated the process of robot identification using TSL2572 light-to-digital converters. However, these sensors proved difficult when it came to implementation because it was hard to translate light intensity differences into color differences, especially given varying lighting conditions. To revise this, we changed the board so that we would have VEML6040 color sensors, which sense red, green, blue, and white light [2].

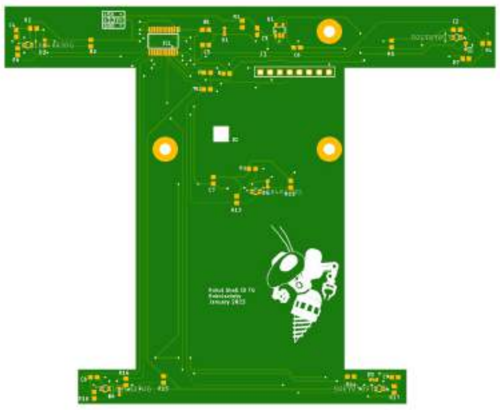

Fig. 4. The new board used for automatic ID assignment. The four VEML6040 sensors are located on the top and bottom corners of the board, below the colored papers used for robot identification.

2.2 Radio

This year's modifications to the Radio Board included replacing the old WiFi module ISM43340 with the Realtek RTL8720DN. The previous module is obsolete (the manufacturer no longer supports it), and the new module was selected because it meets our requirements with WiFi 5GHz support [3]. We are still in the process of testing and implementing these changes on our fleet, but the new board design is below.

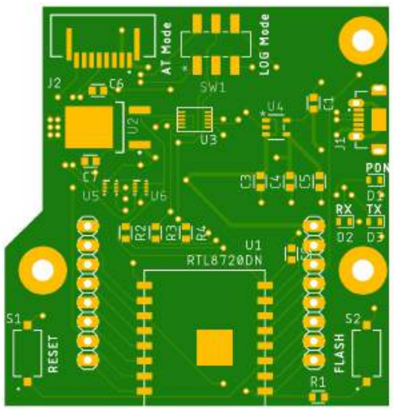

Fig. 5. The new board including the RTL8720DN module.

3 Software

RoboJackets has been a member of the RoboCup Small Size League for nearly 15 years. As a result, the team code base has a large amount of legacy code, with some areas being used frequently (e.g. motion control) and others no longer in use.

The focus for software this year was to create a minimum viable product that allowed for communication and coordinated strategies between robots. This task was approached from the standpoint of improving the integration between the C++ planners and our gameplay stack by switching from STP to an agent-based model. This change will also unify our codebase to be entirely written in C++, no longer using Python for gameplay.

3.1 Motivation

Last year, the gameplay system implemented to refactor STP included an additional Role layer between Skills and Tactics. This was an improvement from our previous gameplay system; however, the improvements were only a bandaid.

The gameplay rewrite started in 2020 was never designed to handle passing well, which resulted in temporary fixes to larger, fundamental design issues up until the 2022 refactor.

To have effective passing, our robots need to know three things:

1. **Which robots** are executing the pass
2. **What target point** the receiver should capture the ball at
3. **What time** time the receiver should expect the ball to be at the target point

The Role system allowed us to handle (1) more effectively. (2) and (3) require an intimate connection with our motion planning algorithms than our current system can provide. The root issue is that our non-gameplay code is designed in a distributed manner with ROS nodes, while our gameplay system was a tree where everything started and ended with Gameplay Node.

We were running into issues where our gameplay system and our planning system have separate conceptions of when a robot is done with a task. This leads to unexpected and hard-to-debug behavior. For example, when trying to pass effectively, our Receive (gameplay) skill is prematurely switched to a different one because gameplay believes it is done, even though the planner could work for a few more milliseconds to get a better grasp of the ball. It would be more intuitive if the planner provided an estimated time-to-completion and a signal for when the goal is complete. This fits the ROS Action model, which defines a client-server model where communication has 3 parts: a goal request, feedback topic, and result service.

Thus, while our agent-centric approach is inspired by ER-Force's software architecture (see Figure 6 below), we will utilize ROS for agent communication and role scheduling. We intend to use ROS's unique features such as the Action model and first-class support for messages between concurrent nodes to implement an agent-based model. Going forward, gameplay will be referred to as strategy, since we are not using plays.

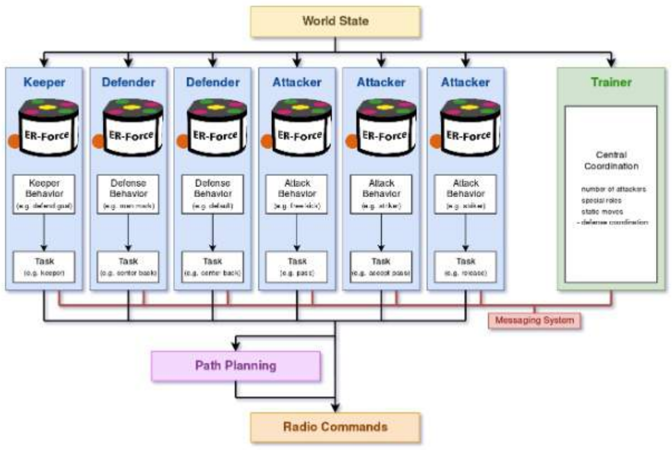

Fig. 6. ER-Force Strategy Diagram [4]

3.2 Overview

While ER-Force implements a custom communication layer, our new design relies on ROS Action Clients/Servers to implement a multi-agent system. Each individual robot has its own Action Client, and the Planner Node is refactored into an Action Server. This allows each robot to have full access to the scope of what our planning stack offers. A Coach ROS node will assign each of these robots a Position, either a Goalie, Offense, or Defense. Each position will have Roles that are chosen based on intercommunication between the robots to execute various multi-agent sequences.

3.3 Design Architecture

Action Client/Server | As described in the overview, each robot has been transitioned into an agent by relying on ROS Action Clients/Servers. It's one of the operating system's unique tools that communicate through an Action message. Each action message requires a goal, result, and feedback parameter that both the Action Client and Server utilize for completing tasks. This specifically solves the issue of miscommunication between the gameplay and planner nodes in the past. The Action Client will then determine which Role to enact based on its Position and send a goal request for that specific planner from the Planner Action Server Node. Once the goal has been accepted by the action server and received by the action client, feedback will be continuously sent between the two bodies until the goal is finished. When that happens, the process restarts between the client and the server. Note that this communication occurs for all clients with the server as they attempt to accomplish their Roles.

As a result of this system, robots can now directly subscribe to the world and match state, rather than being dependent on a top-level node to pass this information. They can directly publish their intent to the planner node, rather than passing through a hierarchy of indirection. Robots also have full access to ROS parameters, instead of strategy needing its own parameter server. This allows for quicker debugging and instantaneous changes to robot behavior.

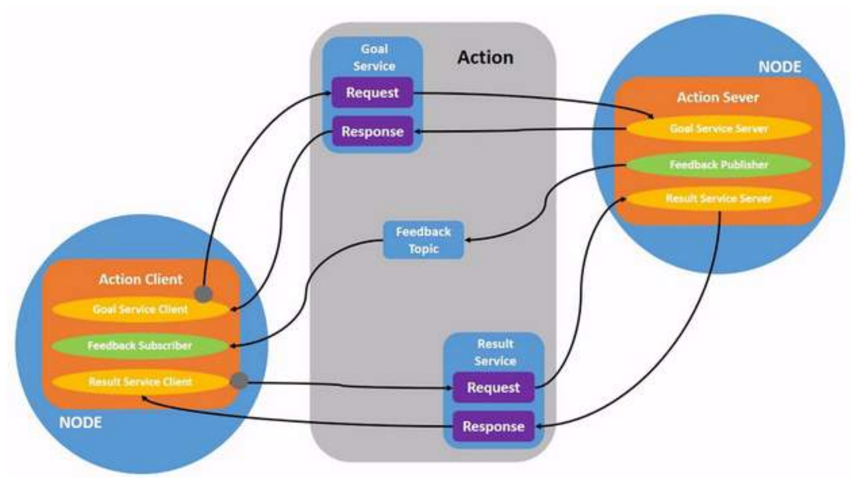

Fig. 7. ROS Actions [5]

Coach | The new design has a single ROS node that tells each Action Client what it should be doing, in a broad sense. For a RoboCup SSL match, the Coach simply has to communicate to the robots when the referee has issued a change that affects their high-level behavior. The three main positions are Goalie, Offense, and Defense. These positions are assigned based on the positional and hardware requirements for the Position. For example, a robot without a functional kicker will not be assigned the Offense Position.

The Coach should process and publish the match situations and global overrides in a format that each robot can process, so each robot can simply say "Coach is saying it's a Ball Placement for the opponent, I should back away from the ball, and also do so slowly because we are in STOP" instead of each robot figuring that out for themselves. The key point is that regardless of what extra information the Coach publishes, now we don't have to create a new class for every combination of match situations possible, which will make compliance with the rulebook much easier.

4 Open Source

RoboJackets continues to open source all aspects of development. Links to software, electrical, and mechanical materials can be found on the RoboCup SSL website [6].

References

[1] A. Ryll and S. Jut. *TIGERs Mannheim - Extended Team Description for RoboCup 2020*. 2020.

[2] Reinhard Schaar. *Designing the VEML6040 RGBW Color Sensor Into Applications*. URL: https://www.vishay.com/docs/84331/designingveml6040.pdf.

[3] Realtek Semiconductor Corp. *UM0401 RTL872xD Datasheet*. URL: https://files.seeedstudio.com/products/102110419/Basic%20documents/UM0401_RTL872xD_Datasheet_v3.4_watermark.pdf.

[4] A. Wendler C. Lobmeier D. Burk and B. Eskofier. *ER-Force - Extended Team Description for RoboCup 2018*. 2018.

[5] J. Perron G. Biggs and S. Loretz. *Actions*. URL: https://design.ros2.org/articles/actions.html.

[6] *Open Source Contributions*. URL: https://ssl.robocup.org/open-source-contributions/.