ITAndroids Small Size League Team Description Paper for RoboCup 2024

André Gonçalves, Andrei Freiberger, Arnon Sousa, Arthur Souza, Artur Silva, Bruno Yamamoto, Emanoel Morais, Felipe Filho, Gustavo da Hora, João Barbosa, João Santos, Lucas Neves, Lucas Barros, Luiz Fusinato, Luiz Alves, Luís Mendonça, Marcos R. O. A. Máximo, Mariana Castro, Mateus Viana, Murilo Pedroso, Nando Farias, Paulo Façanha, Valerio Barros, Yves Sousa

Autonomous Computational Systems Lab (LAB-SCA)
Aeronautics Institute of Technology (ITA)
São José dos Campos, São Paulo, Brazil
Contact: itandroids-small@googlegroups.com
Website: http://www.itandroids.com.br

Abstract. ITAndroids is a robotics competition group associated with the Autonomous Computational Systems Lab (LAB-SCA) at the Aeronautics Institute of Technology (ITA). Our focus over the last year has primarily been on enhancing our software capabilities, with notable advancements in testing, logging, path planning, and in our GUI. Additionally, we have made substantial changes in electronics, particularly in the redesign of communication and movement boards. While mechanics saw more punctual improvements, notable progress has been made in the reconstruction of the kick mechanism and cover. Our efforts are geared towards establishing ourselves as a prominent Small Size League (SSL) team in both the Brazilian and global robotics scenarios. This paper outlines our recent developments, ongoing projects, and future aspirations.

1 Introduction

ITAndroids is a robotics research group associated to the Autonomous Computational Systems Laboratory (LAB-SCA) at Aeronautics Institute of Technology. As required by a complete endeavor in robotics, the group is multidisciplinary and contains about 70 students from different undergraduate and graduate engineering courses, and about 20 of these are in currently in the SSL team.

This paper presents our efforts in developing a Small Size team to compete in RoboCup 2024 with focus to works since our Team Description Paper for RoboCup 2023 [8]. The rest of the paper is organized as follows. Section 2 explains our electronics projects and the most recent developments. Section 3 presents our efforts regarding mechanics. Section 4 explains our artificial intelligence main work and most recent refinement. Section 5 shows the development of our software in general. Section 6 comments about our recent focus on low-level controller. Section 7 concludes the paper and shares ideas for future work.

2 Electronics

Currently, ITAndroids SSL has been working with the 2nd generation of robots, where the main focus is to improve the overall operation of the main board, both in terms of the application of the electronics that have already been designed, and in terms of connections and integration with the robot's mechanical design.

2.1 Next steps for the second generation

The plan was to correct any errors in the 2nd generation prototypes and at the same time maintain the advancement of new ideas, as will be presented in the following topics.

Mainboard v2 In the second half of 2019, we started designing a new mainboard. Key changes included the replacement of the FPGA with a STM32H753BI [17] microcontroller, boasting 480 MHz and 2 MB of Flash memory. This shift was motivated by the team's greater familiarity with STM32 microcontrollers in the humanoid league and the high performance offered by the selected model. Additionally, the motors were upgraded to Maxon EC-45 50 W [10], requiring a redesign of the power supply system and the adoption of the A3930 MOSFET Driver [11] for motor control.

The adoption of a STMicroelectronics microcontroller facilitated advancements in embedded software. The firmware was adapted to utilize Hardware Abstraction Layer (HAL) drivers, and a significant upgrade was the integration of the open-source Real-Time Operating System (RTOS) FreeRTOS kernel [1]. FreeRTOS is able to ensure a deterministic and stable response from the controller, enhancing overall system integrity during matches, but care is needed when configuring tasks and priorities. The team are currently experiencing communication issues that may be related to the way the communication task was configured. Furthermore, it was noticed that real-time embedded software is more sensitive to maintenance-driven small code changes, which could cause firmware malfunction or even complete failure. Code maintenance then became more complex and less accessible for most team members. Despite this, we have good expectations for the time determinism that the use of FreeRTOS can provide to the team once we finish reviewing and fixing the code already developed.

By the end of 2022, the mainboard PCB design was completed, as depicted in Figure 1. In early 2023, the team made tests with firmware implementation, with a fully functional mainboard with concluded firmware by the middle of the first half of the year. Several small issues have been reported, but none that would prevent the first iteration from being used in competition, such as at LARC 2023. Many of the issues have been fixed on the board itself, and specific improvements are being fixed in the second iteration of the board.

Although repeated bench tests and field tests were successful, the participation of ITAndroids' SSL in LARC was quite limited due to severe communication problems between the station and robots, caused by packet losses. Some

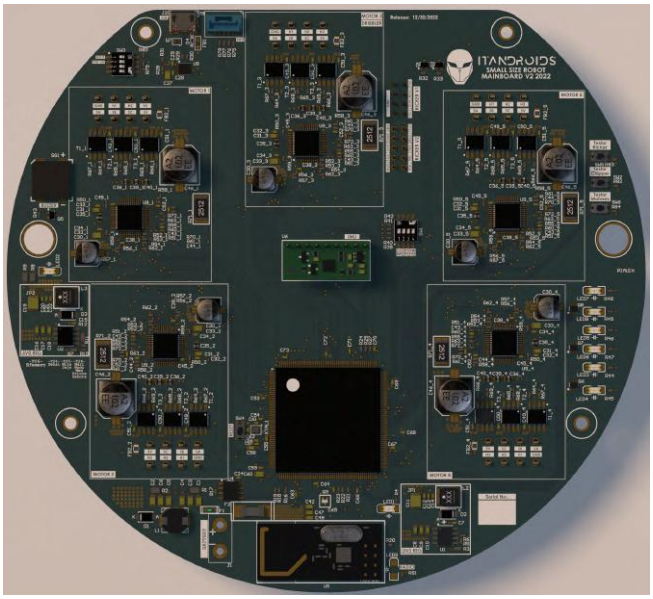


Fig. 1: A rendered image of Mainboard v2 from Altium Design.

hypotheses were raised for the problem, including: interference caused by an overly crowded spectrum, conflict and poor adaptation of the firmware interface of the NRF24L01 module with the real-time system, RF antenna with low sensitivity and low transmission power, radio station poor performance, or problems related to USB communication between the central computer and the station. All of these hypotheses are under study by the team and have already triggered solutions that are under development and testing, as is the case with the revision of the firmware and real-time system and the design of the new radio station.

Modular Structure With the aim of better handling each electronic functionality of the main board, we began the modularization process, which proved to be essential for the project's evolution. The main motivation for this was the ease of maintenance in modules, which, in case of failures, did not render the main board completely unusable. The Fig. 2 is a diagram that shows the configuration envisaged for modularization, that indicates a simplified diagram of the way we want to divide the function of each board that makes up the modules. Therefore, there will be a main module that will carry the STM32 and the other modules via pinhead connection; These other modules will be the radio, the kickerboard, the brushless motor module and the voltage regulator module.

We intend to have a physical model ready by the end of 2024 and will apply it to the robot in 2025.

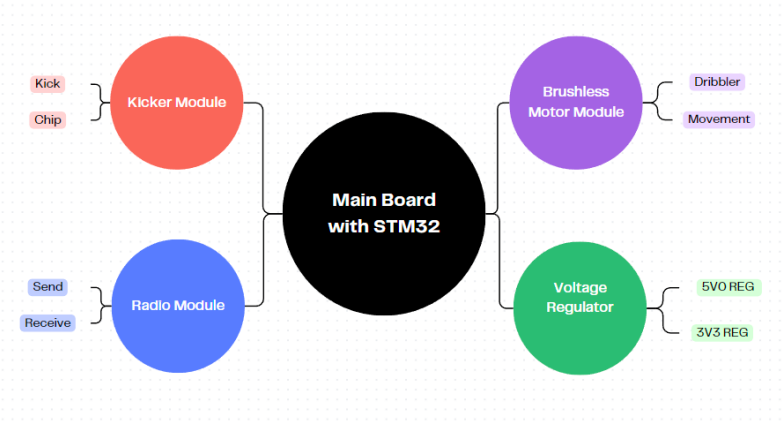


Fig. 2: New electronic board diagram.

Brushless Motor Module The motor module we designed has as its main functionality the activation of the robot's brushless motors, of which there are four used in movement and one used in the dribbler's rotating cylinder. The integrated circuit used was the A3930, which is well established for use in brushless motors. The main improvement applied in this module was the reading of current from each phase of the brushless motor; in this case, the motor used has three phases. For this, we used the AD8216 integrated circuit, which works together with a shunt resistor to monitor the current. In this way, we intend to apply control through this current loop in the future. The module will be connected via pinhead to the main board, which allows for both secure connection and handling of this board on a protoboard, facilitating testing, maintenance, and enhancements of this board.

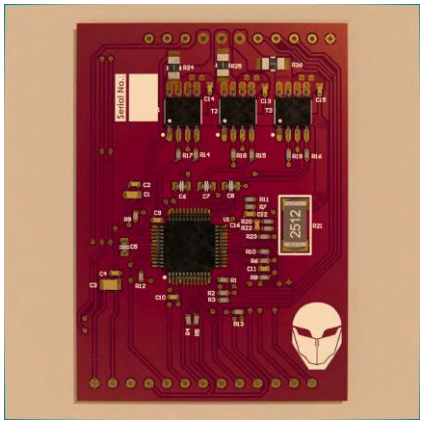


Fig. 3: A rendered image of Brushless Motor Module.

By the end of 2023, we completed the design of the Fig. 3. We hope to obtain the physical model as soon as possible and start testing for real application on the robot.

2.2 Transceiver Station

To follow the development of the second generation of robots, a new radio station is being developed for broadcast communication with the robots. The first generation of the station was based on the 2.4 GHz RF module NRF24L01 [12] and Arduino Uno, assembled only with jumpers, which caused frequent problems such as poor electrical contact.

The second generation, as shown in Fig. 4, is a shield designed by ITAndroids for the STMicroeletronics NUCLEO-F446RE [18] development board. This shield is a transceiver station, optimized for sending and receiving messages. It now contains two NRF24L01 modules with antennas, which allow greater transmission power and greater reception sensitivity. It aims to solve communication problems identified in previous competitions. Each module is configured as RX or TX using a different channel, facilitating send and feedback interactions in less complex ways. The choice to design a shield instead of a board is to simplify the project, using internal modules on the development board. The choice of the development board and microcontroller aims to follow the team's standard of developing with STM32 Microcontrollers, given the team's greater familiarity with this brand.

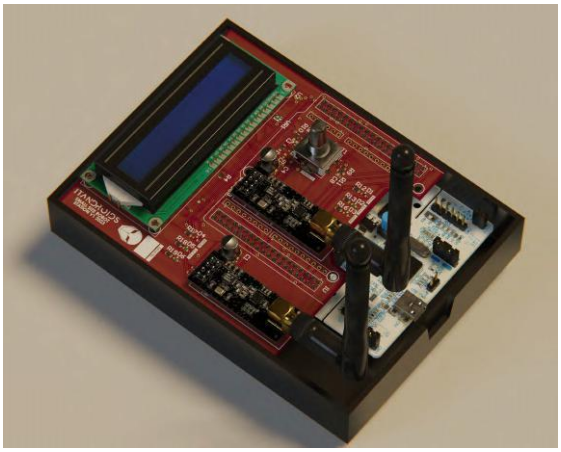


Fig. 4: A rendered image of the Transceiver Station v2 designed in Altium Designer.

Although for now the firmware is quite simplified to perform basic operations of sending and receiving messages, it is expected that the new station will be

an independent platform for testing communication and scanning excessively busy channels in the 2.4 GHz spectrum, which could be avoided during matches, ensuring greater reliability in communication.

3 Mechanics

The past year has seen modest progress in the mechanical development of the ITAndroids team's robot. Notable advancements include significant improvements to the robot's cover and the exploration of volumetry to integrate a new kicker/chipper mechanism.

3.1 Kicker/chipper system

A new mechanical system is being developed for the kicker and chipper mechanism in order to solve a lack of space problem that was occurring with the new generation robot. We started to work with it recently, so unfortunately we do not have results to show yet.

3.2 Cover

The steel cover used in the previous generation did not suit the needs of the new generation robots, so a new cover made using 3D printing was developed. The primary reason for changing the cover was its material conductivity, which caused issues in the robot's electrical components, in addition to its difficult manufacturing process and less precise results. The cover model is shown in Fig. 5. It is 130.5 mm high and 89 mm in radius. The walls are 2 mm thick, printed in PLA with a layer height of 0.15 mm.

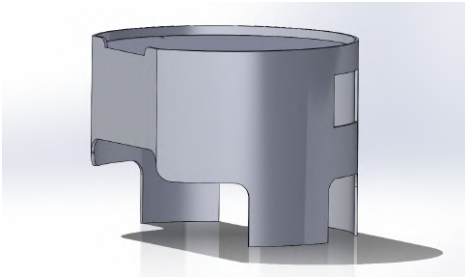


Fig. 5: New cover.

In the near future, some changes will be made to the printing process in order to improve the cover's structural quality.

4 Artificial Intelligence

4.1 Behavior tree

Our decision making process is based on a Behavior Tree implementation, as explained in [9], as our team has much experience with this technique [15]. The advances we had in the last year were actually related to good coding practices and exploring the modularity of the Behavior Tree. In this regard, we determined the most basic behaviors we needed – such as going to a certain position, aiming to a certain direction, kicking the ball, and chipping the ball –, trying to promote code reuse. Then we redesigned our existing strategy, but with stronger code standards.

4.2 Opponent Modeling Strategy

Right now, our team is developing a system to predict the trajectory of robots on the opposing team. This system will aid in formulating more complex strategies for decision-making. We are employing deep neural networks to forecast robot trajectories within our code. These models have been trained using data from previous RoboCup matches (position and orientation in each timestep). Currently, we are in the testing phase, experimenting with various network architectures (Multi Layerered Perceptrons, LSTMs, CNNs) and fine-tuning hyperparameters. Similar research has been carried on by former ITAndroids members [6, 16]. However, it has not been possible yet to implement such models in real competitions. Our goal is to learn from these previous works and develop an opponent modeling tool that can be used properly in the RoboCup context.

Among the tools used in this study, there are:

- Keras API for neural network implementation [5].
- SSL logtools for parsing log files and use them as training and test data (league software).
- Optuna, a python framework for hyperparameter optimization [3].

Data-driven approaches such as those mentioned offer considerable advantages, as they allow robotic systems to learn from real matches data. These strategies can improve our robots' capability of making quick and precise decisions in the complex and dynamic environment which they are immersed in.

5 Software

In this section we dive into our efforts related to our software in a general manner.

5.1 C++ Dependencies

It is a known issue for *C++* developers the lack of an official dependency or library manager. Some projects choose to compile all dependencies manually,

others use platform-specific managers, like the package managers in GNU/Linux, and even including in the project the source code of the version you need is a common option. Our team has suffered over the years with this, and we have decided to start migrating to *VCPKG* [2].

VCPKG, developed by Microsoft, is recognized as a promising solution to the issue of dependency management in *C++* development. Its centralized repository of precompiled libraries and tools offers a streamlined approach to managing dependencies, addressing a well-known problem in software development.

One potential obstacle we anticipate in migrating to *VCPKG* is the impact on disk usage, particularly concerning the downloading and storage of source code for compilation. *VCPKG* operates by fetching source code from its centralized repository and compiling it locally on the user's machine. As a result, the process may require significant disk space, especially for projects with extensive dependencies. Each addition or update of a dependency through *VCPKG* entails downloading the corresponding source code, contributing to an accumulation of data over time.

5.2 Logger

In this subsection, we outline our implementation of a logger using the *spdlog* library [7], tailored to meet our basic logging needs with minimal complexity.

spdlog is a versatile *C++* logging library that offers simplicity and ease of use, making it an ideal choice for our logging requirements. With its intuitive methods and lightweight design, *spdlog* enabled us to quickly integrate logging capabilities into our software without unnecessary overhead.

While our logger configuration remains basic, *spdlog* offers extensive customization options, allowing us to tailor logging behavior to our specific requirements as our needs evolve. It supports asynchronous mode, custom formatting, both multi-threaded and single-threaded loggers, and Qt integration. These options provide flexibility in optimizing logging performance and output according to the demands of our application.

5.3 Path Planner

Our path planning algorithm is based on visibility graphs, as written in [9]. Over the last year, we have made some adjustments to it, due to practical reasons.

Our algorithm considers the robot as a point and obstacles as circles, initially with the radius of twice the radius of a robot. With the real world imperfections and the dynamic nature of the league, it is required to increase this obstacle's radius. Therefore, we create a safe zone for the robots to determine paths using our algorithm. Unfortunately, again due to the dynamic nature of the league, it is possible for the robot to enter into this safe zone, thus originating a problem (since for the algorithm the robot is inside an obstacle). Unluckily, if you give your robots a bigger safe zone, this situation may occur more than you expect.

More on this problem, what you want to do if your algorithm does not find a path is not so easy to determine. You might want the robot to just go straight

ahead to his destination, since in a situation where you have two robots fighting for the ball this may be desired – otherwise you will lose each fight –. Looking from another perspective, you might want to simply skip this iteration. Despite the risk, we have chosen to do the first option, since in the game (in our perspective) it is more important to fight for the ball.

Another problem we faced was the choice of what to do with small segments of paths. Sometimes the algorithm seems to find a path with a segment – circular or straight – so small that it actually does not matter, and to try to follow this segment actually decreases the performance of following the path. The difficult part of this problem is how to find a good threshold (both for angles and for straight lines), as there is no clear way to measure the average performance.

5.4 Testing

In this subsection, we discuss our endeavors over the past year to enhance the testability of our codebase and to promote a culture of test-driven development (TDD) within our robotics team. Utilizing the *GoogleTest* framework, we started to integrate unit testing seamlessly into our development process.

Enhancing Testability Recognizing the importance of testability in software development, we have made concerted efforts to refactor and redesign our codebase to improve its testability. This includes breaking down complex functionalities into smaller, more modular components and adhering to best practices in software design, such as separation of concerns and dependency injection. By designing our code with testability in mind, we aim to facilitate the creation and maintenance of comprehensive unit tests.

Development of Unit Tests In line with our commitment to test-driven development, we have prioritized the development of unit tests for critical components of our codebase. Leveraging the *GoogleTest* framework, we have created a suite of unit tests to validate the behavior and functionality of individual units of code. These tests cover a wide range of scenarios, including edge cases, boundary conditions, and typical usage scenarios, ensuring robustness and reliability in our software.

Creating a Culture of TDD In addition to developing unit tests, we have focused on fostering a culture of test-driven development within our team. This involves encouraging developers to write tests before implementing new functionality, emphasizing the importance of test coverage and ensuring that tests are integrated into the development workflow.

5.5 Graphical User Interface

Last year, our team recognized that the GUI is a powerful tool that was underutilized by us, despite its capability to offer not only visual feedback on the robot's

Fig. 6: Current Graphical User Interface.

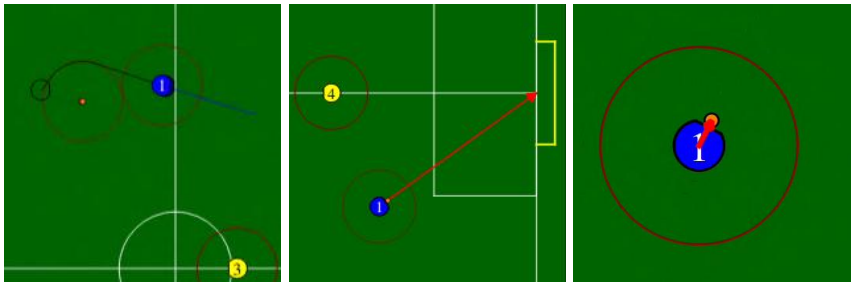
actions but also an easier way of interacting with the software. Keeping this in mind, we have focused on developing key features such as displaying valuable information provided by the game controller and implementing new command widgets. The current state of our GUI may be seen in Fig. 6.

The main feature we have been working on the GUI are the draw requests, which helps us visualize the robots' behavior and decision making. Our idea was to make a draw requester, which the entire code can use to request a drawing on the screen. When some part of the code calls a method requesting a drawing, the requester implementation handles the creation of a draw request (which encapsulates a Qt implementation of the drawing) and sends it to a buffer. As the updating of the GUI occurs in another thread, we used a double buffering approach: to be precise, we used two buffers for each thread that can request a draw. This approach with draw requests was pretty much inspired by *rviz* and *Roboviz* [13, 19].

By providing a visual representation of the robots' behavior, the draw requests not only aid in debugging and troubleshooting but also serve as a valuable tool for refining our algorithms and optimizing performance on the field. Some examples of them are shown in Figs. 7a, 7b, and 7c.

Expanding on this feature, we have implemented separate visualizations for detecting and estimating the positions of the ball and robots. These visualizations, with the estimation overlaying the detection, provide us with a comprehensive understanding of our filtering algorithms' efficacy during gameplay. This enables us to obtain valuable insights for further enhancing our software's performance.

Furthermore, this approach provides an insight into how to store a replay with the behavior of our entire code. Hence, storing the requested drawings with



(a) Trajectory (blue) and planned path (black).

(b) Aiming.

(c) Kicking.

Fig. 7: Examples of draw requests applications.

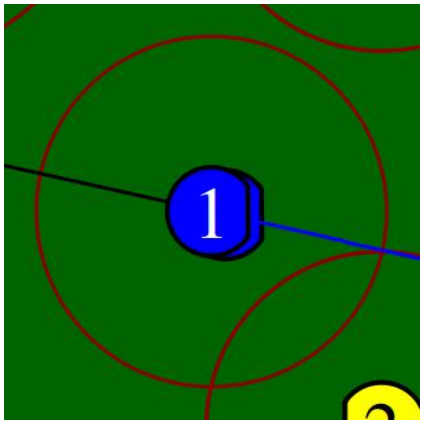


Fig. 8: Estimation and detection overlap.

its timestamps, we can debug what happened in the past (e.g., during a previous match).

Another useful feature we have developed was a widget to adjust all the parameters in real-time on the GUI, as shown in Fig. 9, whose backend was inspired by Rhoban [14]. With almost the same code implementation, it was also possible to create a similar widget for selecting which parts of the code its draw requests were going to be shown in the screen, as shown in Fig. 10. The equivalent widget to select log-related options is yet to be developed.

6 Control

Last year, our control team had focused more on upgrading low-level controllers, by experimenting new control topologies for motor control and new ways to obtain its gains, such as considering the whole robot's dynamic [4]. As we are

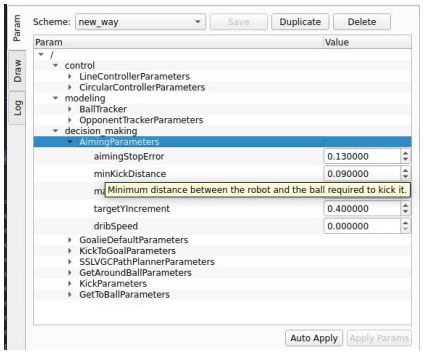


Fig. 9: Tree including all parameters.

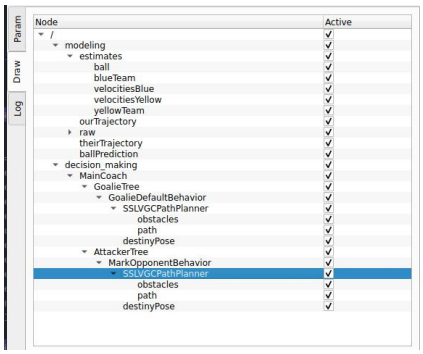


Fig. 10: Tree to select which requests to show on the GUI.

still on early stages of this work, unfortunately we do not have any major result about it to show yet.

Another work we had, a more practical one, as we migrated to a new robots' generation, was to implement a new motor model on our simulations for the obtainment of motor control's gains.

7 Conclusion and Future Work

This paper describes some of our efforts during the year of 2023 and for the next months until RoboCup 2024. As we are only starting to have our spotlight on SSL scenario, we are also looking forward as we hope to be open source soon. For RoboCup 2024, we hope to have an improved version of the ideas that are in progress, in addition to applying some of the prototypes that we have been working on since 2023.

Acknowledgment

We would like to acknowledge the RoboCup community for sharing their developments and ideas. We especially acknowledge RoboFEI, Skuba and TIGERs for open sourcing their electronic and mechanic designs. Moreover, we would also like to thank members of CMDragons, RoboFEI and RoboIME for helping in various contexts. Finally, we thank our sponsors Altium, CENIC, Field Pro, Intel, ITAEx, MathWorks, Micropress, Neofield, Polimold, Rapid, Siatt, Solidworks, ST Microelectronics, Virtual Pyxis and Visiona.

References

1. Freertos™. https://www.freertos.org/index.html (2023)
2. vcpkg: A c++ package manager for windows, linux, and macos. https://github.com/microsoft/vcpkg (2023)
3. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019)
4. Celso, F., Maximo, M., Yoneyama, T.: Model predictive control for omnidirectional small size robot with motor and non-slipping constraints. (2023)
5. Chollet, F., et al.: Keras. https://keras.io (2015)
6. Coimbra, F.: Opponent modeling by imitation in robot soccer (2021)
7. Gabriel Simões: spdlog c++ logging library. https://github.com/gabime/spdlog (2023)
8. Gonçalves, A., Freiberger, A., Martins, A., Penna, A., Moreira, B., Yamamoto, B., Simplício, E., Morais, E., Pironi, G., da Hora, G., Mendes, L., Neves, L., Zoppi, L., Maximo, M., Martins, M., Egger, M., Pedroso, M., Farias, N., Barros, V., Wakugawa, V., Queiroz, Y.: ITAndroids Small Size Soccer Team Description Paper for RoboCup 2023
9. Maranhão, A., Schuch, A., Azevedo, A., Rodrigues, A., Lima, E., ao Sarmento, J., Santos, M., Maximo, M., Sales, M., Dias, M., Lima, R.: ITAndroids Small Size Soccer Team Description Paper for RoboCup 2020
10. Maxon: Maxon EC-45 Flat Brushless Motor 50W Datasheet. https://www.maxongroup.com/medias/sys_master/root/8833813119006/19-EN-263.pdf
11. Microsystems, A.: Datasheet A3930. https://www.allegromicro.com/~/media/Files/Datasheets/A3930-1-Datasheet.ashx?la=en
12. NordicSemiconductor: nrf24l01+. http://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf
13. Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng", A.: "ros: an open-source robot operating system". In: "Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics". Kobe, Japan (May 2009)
14. Rouxel, Q., Passault, G., Hofer, L., N'Guyen, S., Ly, O.: Rhoban hardware and software open source contributions for robocup humanoids. In: Proceedings of 10th Workshop on Humanoid Soccer Robots, IEEE-RAS Int. Conference on Humanoid Robots, Seoul, Korea (2015)

15. Silva, G.L., Maximo, M.R., Pereira, L.A.: A minimalist open source behavior tree framework in c++. In: 2021 Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE). pp. 306–311. IEEE (2021)
16. Steuernagel, L.: Opponent modeling for robocup small size league robots (2021)
17. STMicroelectronics: STM32H753xI (April 2019), rev 7
18. STMicroelectronics: STM32 Nucleo-64 Boards User Manual (August 2020), rev 14
19. Stoecker, J., Visser, U.: Roboviz: Programmable visualization for simulated soccer. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011: Robot Soccer World Cup XV. pp. 282–293. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)