RoboIME: On the road to RoboCup 2023

Felipe W. V. da Silva, Antônio S. G. Pereira, Vinicius de F. L. Moraes, Mayara R. Mendonça, Gabriel M. Lima, Henrique Barreto, Iasmin C. Mathias, Lucio E. Horie, Enzo G. Frese, Pedro V. Rocha, Renato da P. Alves, Marcos C. Antunes, Léo V. C. Vasconcelos, Isabel C. de Freitas, Arthur B. S. de Souza, Marcos C. Antunes, Eduardo Baldi, Ebert H. V. Melo, Raquel Belchior F. N., Antônio Plácido C. C. F., João Pedro Gomes C., Alberto F. de Lima. L. S., Franciele Sembay, Lucas G. Corrêa, Gabriel B. da Conceição, Carla S. Cosenza, Lucas B. Germano, Matheus Bozza, Luis D. P. de Farias, João G. O. C. de Melo, Nicolas S. M. M. de Oliveira, Gustavo C. K. Couto, Herbert Azevedo, Luiz R. L. Rodrigues, and Paulo F. F. Rosa

Instituto Militar de Engenharia, Rio de Janeiro, Brasil

rpaulo@ime.eb.br
http://roboime.com.br

Abstract. This paper describes the electronic, mechanical, and software designs developed by the RoboIME Team to join the RoboCup 2023. The overall concepts are in agreement with the rules of Small Size League 2023. This is the tenth time RoboIME participates in the RoboCup tournament.

1 Introduction

RoboIME is a Small Size Soccer League team from IME (Instituto Militar de Engenharia), and this is the 17th time the team participates in competitions. The team has already gotten good results on previous occasions: (i) first place in the Latin American Robotics Competition 2017 (LARC 2017); (ii) second place in seven different competitions, RoboCup Brazil Open 2011, LARC 2012, RoboCup 2018 (division B), LARC 2018, RoboCup 2019 (division B), LARC 2019 and RoboCup 2022 (division B); (iii) third place in LARC 2022.

All students that work on the SSL project are members of the Laboratory of Robotics and Computational Intelligence at IME. The team's previous works were used as reference ([2] [9]), as well as the help from former members of the team as consultants and tutors.

This article describes the team's general information and improvement in the last semester since our TDP for RoboCup 2022 and Poster for RoboCup 2021 has detailed explanations on our previous system. The article is organized as follows: software in section 2, embedded electronics in section 3, and mechanical design in section 4. Conclusions are discussed in section 5.

2 Software Project

This section reports the main improvements and changes since the 2022 LARC project. The main focus of this year was to enhance robot movement, targeting better performance at real-life games, but still using concepts that can work equally well on the virtual robot.

After a long period without in-person competition, combined with a significant discontinuity of members in the software project, right after RoboCup 2022, the focus was to make the robots compete again.

The dribbler, used mainly in virtual competitions, was not so efficient in the real world, consequently, a new approach to the kick was developed, a kick substituting the last kick shown in the last RoboIME TPD[5].

Also, during RoboCup 2022, it was clear that many games can go to penalties, so a simple strategy, not implemented until then, was developed for Latin American Robotics Competition, with minor modifications during the competition. These new features were part of an attempt to make a more precise team score more goals.

Additionally, during both competitions of 2022, there were some trouble in performace due to our path planning algorithm, so we decided to optimize it.

2.1 New RRT algorithm

In 2018, RoboIME made the first attempt of using RRT as the path planning algorithm [3]. It replaced the potential field algorithm we used because of a lot of problems involving trapping the robots in local minima. Then we realized that the simple RRT algorithm was not enough, in terms of performance, to handle the path planning in real time and in a dynamic environment.

Consequently, in the following years [4], we introduced the ERRT optimization [6] along with other improvements. The algorithm was still very expensive, hence we started to alternate between making a line as a path and actually using the RRT.

The changes performed better than before but it still had some problems. The ERRT stored waypoint cache from the previous iterations in order to find the path to the goal faster but it still had to rebuild the tree in every iteration. This way, even if the RRT was called less than necessary because of the line path, it was still hard to make an efficient path at execution time.

Because of that, we started to implement a new way of using the RRT, based on RRT* and Informed RRT* [8] with the main purpose of retaining the whole tree between the iterations rather than only a waypoint cache. This way, the algorithm focuses on adapting the existing tree to the movement of the agent and the obstacles by performing two ways of rewiring. Additionally, it is a multi query algorithm that can quickly find and refine the path to any point in the environment, making it a real-time execution algorithm.

The new method consists of building a tree and growing it until some user-defined density. This tree is going to be retained between the iterations and will only be expanded (if it still can grow) and rewired.

The expansion starts by sampling a random point in one of 3 different ways: uniformly in the environment, in the line connecting the goal and its closest node in the tree, and inside an ellipse similar to informed RRT*[7] The rewiring is executed around the root, so it can change while the agent moves, and also at random locations to accelerate the path refining.

The path itself is planned by searching a k-step path that has the minimum total distance and leads closer to the goal when the path isn't already found. When it is found it only updates the path by backtracking from the goal. If an obstacle blocks a node, its distance to root (that is called cost to reach) becomes infinity, and then rewiring will find a new clear path away from the obstacle.

Additionally, the cost to reach is calculated by adding the distances from node to parent, until it reaches the root. But this calculation would be expensive if not optimized. So, we implemented something similar to a segment tree with lazy propagation, so that the query is $O(\log n)$ time complexity and the updates are $O(1)$

With respect to the practical improvements from the old RRT, we cannot assure yet because its implementation is still in the testing phase. But the picture below shows how the algorithm responds to a moving obstacle when the target and the agent are fixed.

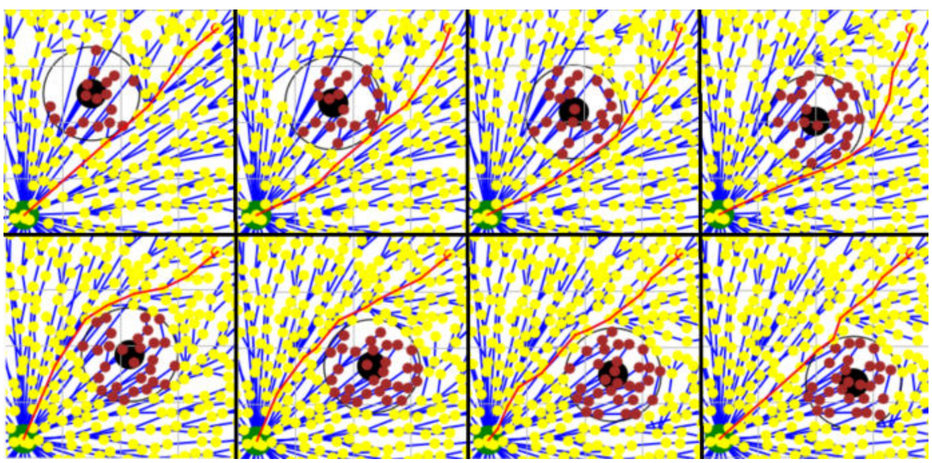

Fig. 1: The tree nodes are in yellow, the edges are in blue, the obstacle is in black and the brown nodes are intersected by the obstacle. When a node is intersected, its child nodes are rewired, so a new path is found.

In terms of optimizations for the algorithm, there are some improvements it could have. The complexity of the path planning is $O(\log n)$, the expansion is $O(1)$ and the rewire iterations are fixed and ordered in a queue so it is $O(1)$ too. But inside the expand and rewire calls and in the environment update, it needs to calculated the cost to reach and to find the nearest nodes. In ERRT we used the K-NN search method to perform the search in $O(\log n)$, but for simplicity we implement a square grid search that has an $O(\sqrt{n})$ complexity.

Therefore the total complexity of the algorithm is $O(\sqrt{n} \cdot \log n + K \cdot \sqrt{n} \cdot \log n + \log n) = O(\sqrt{n} \log n)$. It could be improved to run at $O((\log n)^2)$ but it is still better than rebuilding the tree in the old RRT with $O(n \cdot \log n)$ complexity. Another issue is that when the tree is still growing, the uniform samples create larger than common edges that are not recognized by the moving obstacles. As the tree grows it becomes less common but there is a small chance of planning a path that intersects an obstacle.

2.2 Improving the "new kick to" tactic

The new way to shoot the ball allows the robot to keep his front end positioned towards the ball, regardless of the dribbler.

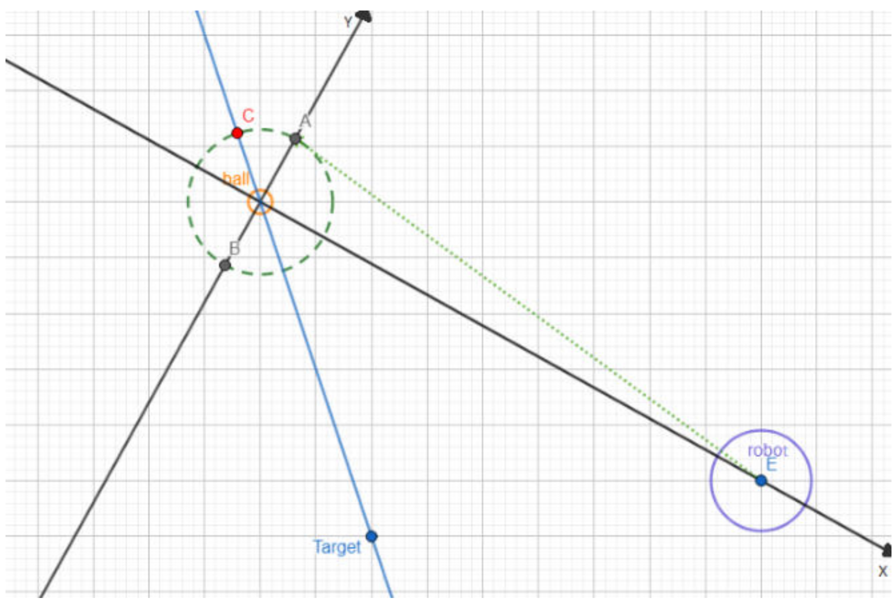

Fig. 2: Illustrative image for "the new kick to"

A new tactic to kick was made with six skills, the finite state machine states, based on the model skill, tactic, and play (STP), which will be explained later.

First, it was decided to use the ball as reference, with the origin on it and the rotated X-axis passing through the robot, as shown in the image. The finite state machine checks one by one the conditions to enter each skill, and the first one to satisfy the conditions changes the state or keeps it.

So we have points A (X_a, Y_a), $B(X_b, Y_b)$, $C(X_c, Y_c)$, and the coordinates of the robot (X_r, Y_r) and the target (X_t, Y_t), calculated by the AI, considering the new axis. C is the point on the line ball to target and behind a distance R from the ball, where R is a radius parameter. A and B are on the line perpendicular to the one that passes through the ball and the robot and also intersects the circumference with parameter R centered on the ball.

The tactic runs as a finite state machine (FSM) with the following states, that is the skills, called in order:

- (Default) go straight to the ball: While the sign of X_r is equal to the sign of X_c and $|X_r - X_C| > R$, the robot goes straight to the ball;
- Take distance: While the sign of X_r is equal to the sign of X_c and$|X_r - X_C| < R$, the robot moves away from the ball;
- Go to point A or B: While the sign of X_r is not equal to the sign of X_c and $|X_r - X_C| > R$, the robot goes to point A if $Y_C > 0$, else goes B;
- Circulate clockwise or anti-clockwise: While the module of the distance between the robot and the ball is approximately R, the robot turns around looking towards the ball with rotation orientation based on the smaller angle to travel;
- Lining up: While the robot is close to C and the orientation is not satisfactory, it lines up to the position for the best kick;
- Shoot: Final state, while the robot is $|X_r - X_C| < R$, the sign of X_r is equal to the sign of X_c and the orientation is correct i.e. towards the ball, the robot goes for the kick.

The ”new kick to” tactic was mainly used in the last games and it was working, nevertheless not as well as expected, because there was a problem when another robot was competing the ball, preventing the ally robot from approaching the destination, keeping the ball stopped many times. Another problem is that the robot, for long distances, when it gets high velocity, can not spin while turning around the ball so quickly, since it can fall over in the field, provisionally, thus a velocity limiter was implemented.

In conclusion, looking only at the objective to make more precise kicks regardless of the dribble, the tactic does the work, and even so it can be optimized given the problems listed.

2.3 New Penalty Tactic

A new way to perform penalty attacks, focusing on maximizing the chances of conversion has been developed. Usually, our robot was commanded to kick from the starting point, but this was inefficient in terms of utilization of the full time of the penalty play, in addition to long shots being a lot easier for the goalkeeper to defend.

With this in mind, we made a simple tactic that solved both problems and improved the goal conversion significantly. This tactic is an FSM based in three states:

- Go to ball: approximate to ball until is close enough to kick.

- Little kick: low power kick to goal.
- Shoot to goal: full power kick to goal.

Thereby, the state machine was implemented following the diagram below.

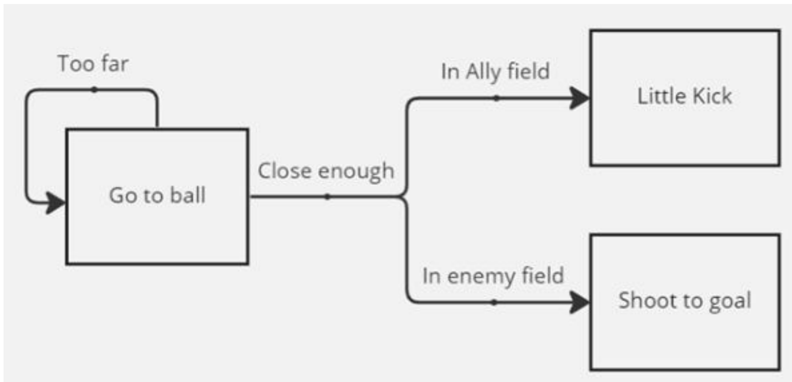

Fig. 3: Fluxogram detailing the new penalty tactic

3 Electronics Project

For RoboCup 2023, our team comes with some minor changes in hardware and firmware. The main focus of our team's work this year is on the reliability. In 2022 we had multiple problems with some electronic components failing and firmware bugs arising during games. Besides that, we had many communication drop problems that we intend to address this year. Despite the range of necessary modifications, new additions and improvements made in the project, the modularization was preserved, carrying the previous model technologies in the 2023 model.

3.1 Firmware

Our robot's firmware uses an object-oriented programming architecture, written in the C++ language and the STM32CubeIDE development environment. This year's changes include fixes to some bugs that arose during last year's games, like one that made the kick capacitor not charge after a chip kick, a new SX1280 library coded by our team and the necessary modifications to interface with the motherboard and the new peripherals, like the SX1280 module and the INA169 current sensor. This aims to solve our 2022 communication drop problems and to avoid H bridge MOSFETs and motor failures. Also, the code is written using better practices and standardizations than the previous one, making it easier to debug and add new features in the future.

3.1.1 Robot Communication In 2022, we had communication drops on the competition day that prevented us from using the feedback system, as packet drops made the delay too big. This year we changed the nRF24L01 radio that we were using to the SX1280 module[1], that has greater transmit power and reception sensibility. It also can operate on a wider frequency range, allowing us to operate on a frequency that is less crowded by neighbors.

Those changes make it possible to receive feedback from sensors, improving feedback interpretation by the AI, looking to fix package loss issues and making debugging easier. The feedback packages include battery charge, wheel speeds, ball possession, kicker charge and current measurements from critical parts of the circuit.

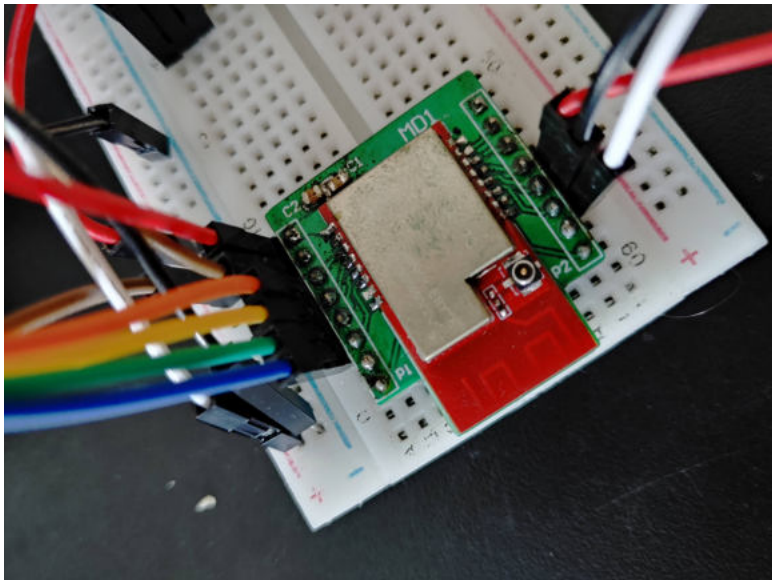

Fig. 4: First prototype of our 2023 radio module

3.2 Control

For this year, we had some changes in the robot mechanics so we will use the same Ziegler-Nichols method from last year again to tune the PID controller. Furthermore, last year when we used the 3-cell battery the H bridges were burning, so we ended up using 2-cell batteries and tuning the PID controller accordingly. We will be using the 3-cell batteries again so it will be necessary re-tune our PID controller.

3.2.1 Ziegler–Nichols's method The Ziegler-Nichols rule is a heuristic PID tuning rule that attempts to produce values for the three PID gain parameters (Kp, Ti, and Td), given two measured feedback loop parameters derived from measurements (Tu - the period of the oscillation frequency at the stability limit - and Ku - the gain margin for loop stability). The method's goal is to achieve good regulation (disturbance rejection) from the generated parameter values.

3.3 Board Designs

The board design created for the 2022 version demanded adjustments after a few issues the team faced during RoboCup 2022, LARC 2022, and in the tests that followed both competitions.

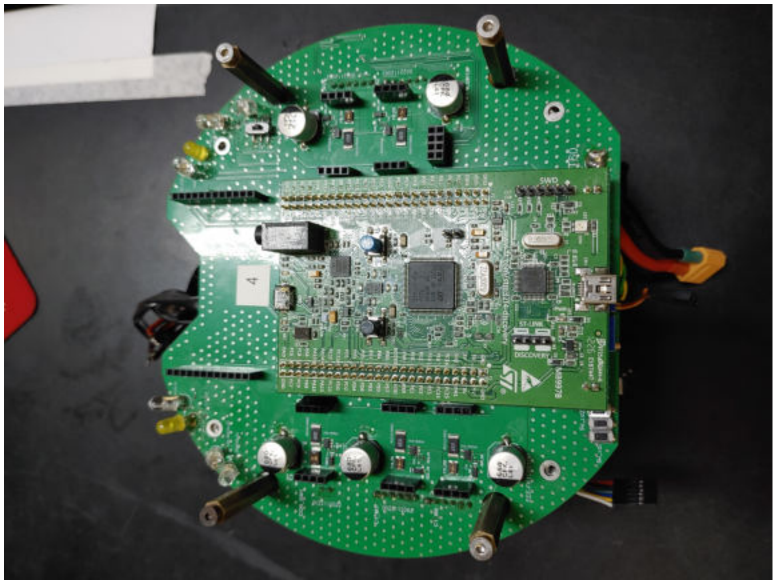

Fig. 5: Motherboard and microcontroller module that we will take for this year

3.3.1 Main Board For the 2023 version, we will maintain most of the 2021 motherboard project only making the necessary changes to accommodate the SX1280 module, which requires more GPIO pins than the nRF24L01 module that we were using. Also, we will make some checks on the INA169 circuit before implementing its reading.

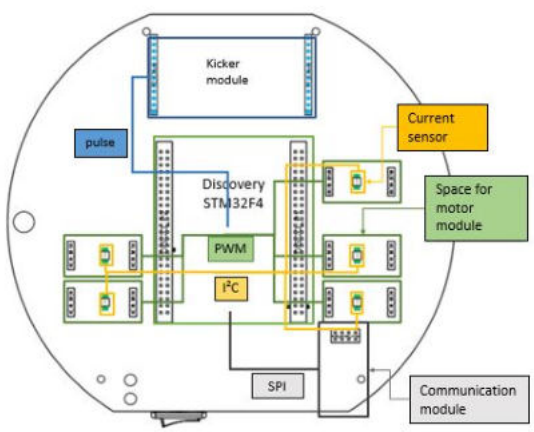

Fig. 6: Main board's block diagram

3.3.2 Motor module For last year's project, we developed a motor module using SI7655 and SIS812 MOSFETs to support the bigger current demanded by the larger motors. In the end, this proved to be a flawed design, and most of our H bridges were burned in the first games. For this reason, we will change the custom H bridge design using SI7655 and SIS812 MOSFETs to a BTS7960B motor drive integrated circuit. This IC was successfully used on our team's robots that compete in other categories. Besides, it proved to be a reliable solution even for bigger motors.

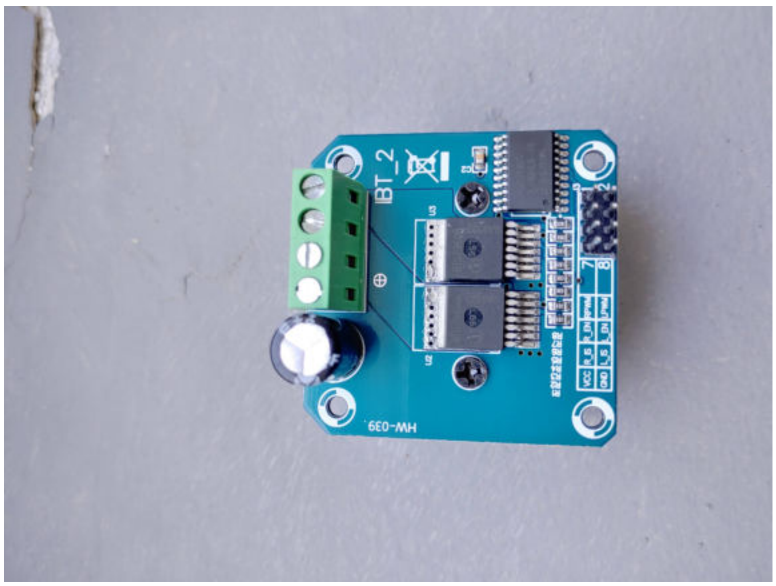

Fig. 7: Module containing the BTS7960B motor drivers that we plan to use this year

3.3.3 Kicker module The kicker module circuit was maintained in its older version except for some minor component replacements. The discharge IGBT was changed to the IGW75N65H5XKSA1-ND and the charge MOSFET is also going to be replaced due to its discontinuity by the manufacturer. They were both changed to more robust ones to improve reliability and avoid failures.

4 Mechanical Project

The mechanical project went through some changes when compared to the previous year's version. Currently, the team is focusing on improving not only the robot's efficiency and robustness, but also the ease of maintenance. Besides that, RoboIME is constantly seeking new solutions for the project and through much research and information exchange with other teams some changes were made in the mechanical project. Below, are described the developments and the planning for RoboCup 2023.

4.1 New Solenoid Supports

Last year we faced some problems with the kick system due to the breaking of the 3D printed solenoid supports in both RoboCup 2022 and LARC 2022. To address this issue, for 2023 we made some changes to the design and manufacturing of the supports. We still use 3D printing technology to create them, but we have made them thicker this year. This increase will enhance the durability and stability of the solenoids and thus increase the longevity of the supports.

In addition to the changes in the thickness of the supports, we have also made adjustments to the dimensions and shape of the solenoids themselves. These modifications will result in an increase in the force generated by the solenoids, which will in turn lead to stronger and more accurate kicks.

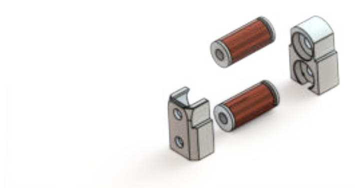

Fig. 8: New Solenoid Supports

4.2 Omni Wheels

The robot's omni wheels have been completely modified. The main objectives of this change were to ease the robot's maintance and reduce costs. The omni now features a single layer of 15 small wheels each (see figure 9) and its diameter was modified from 35 mm to 45 mm. The use of a bigger wheel combined with the new motors will provide more speed to the robot. The body of the omni wheels will be printed, wich will reduce our costs and the weight of the robot.

4.3 Planning for RoboCup 2024

For Robocup 2024, new robots are going to be assembled without reusing parts from the robots that are already assembled. For the upcoming RoboCup, some robot parts that were made from metal will be made of 3d printed plastic, in order to be easier to manufacture and test new possibilities.

Fig. 9: Omni Wheel

5 Conclusions

For this competition, the aim is to consolidate the progress made in the previous years, experimenting with changes in the software project allied to the new electrical and mechanical projects. Our software was improved by developing a new navigation algorithm, a new kick logic, and a new attack strategy. In electronics, besides the firmware improvement, we had also to update the module of communication from nRF24L01 radio to SX1280 module in order to achieve a better reliability on communication. Finally, the new mechanical project (also designed to adapt for the new motors) includes the new omni wheels, the improvement of the Kick System, and better positioning for battery and motors. By applying these features, we should be able to perform better plays and get great results at RoboCup 2023 tournament.

5.1 Acknowledgement

This research was partially supported by the Army's Department of Science and Technology (DCT), Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro - FAPERJ(grant E-03/2022 Apoio a Equipes Discentes em Projetos de Base Tecnologica para competições - 2022); Fábrica de Material de Comunicação e Eletrônica (FMCE/IMBEL) and the IME alumni association. Special thanks to all former members of RoboIME. Without their support, this team would not be here.

References

1. Chris Carstensen Nicolai Ommer Andre Ryll, Mark Geiger. Tigers mannheim (team interacting and game evolving robots) extended team description for robocup 2018. pages 3–5. Available at `https://ssl.robocup.org/wp-content/uploads/2019/01/2018_ETDP_TIGERs_Mannheim.pdf`.
2. Gabriel F. Santos Felipe W. V. da Silva Antônio S. G. Pereira Erick T. da Silva. Leonardo G. Gonçalves Lucas B. Germano Lucas G. Corrêa Henrique Barreto Mayara R. Mendonça Pedro Henrique Marques Isabel C. de Freitas Davi H. M. Pontes

Luciano de S. Barreira Luis D. P. de Farias João G. O. C. de Melo Matheus Bozza Matheus P. de Souza Nicolas S. M. M. de Oliveira Onias C. B. Silveira Rebeca P. dos Reis-Yugo Nihari Gabriel H. M. Silva Guilherme M. Gonçalves Gustavo C. K. Couto Vinicius de F. L. Moraes Luis R. L. Rodrigues Carla S. Cosenza, Gabriel B. da Conceição and Paulo F. F. Rosa. Roboime larc 2020: A virtual challenge. pages 1–6. Available at `https://github.com/roboime/roboime-tdp`.

3. Carla S. Cosenza Gustavo C. K. Couto Lucas Germano Lucas G. Corrêa Luciano de S. Barreira Luis D. P. de Farias Luis R. L. Rodrigues João G. O. C. de Melo Matheus Bozza Matheus P. de Souza Nicolas S. M. M. de Oliveira Onias C. B. Silveira Rebeca P. dos Reis Renan P. de Souza Sergio G. S. Dias Yugo Nihari Paulo F. F. Rosa. Roboime: From the top of latin america to robocup 2018. pages 1–12. Available at `https://github.com/roboime/roboime-tdp`.
4. Carla S. Cosenza Gustavo C. K. Couto Luciano de S. Barreira Luis D. P. Farias Luis R. L. Rodrigues Jan L. L. Segre Matheus C. Castro Nicolas S. M. M. de Oliveira Onias C. B. Silveira Renan P. de Souza Victor Bramigk Yugo Nihari Paulo F. F. Rosa. Roboime: on the road to robocup 2019. pages 1–12. Available at `https://github.com/roboime/roboime-tdp`.
5. Vinicius de F. L. Moraes Mayara R. Mendonça Gabriel M. Lima Henrique Barreto Iasmin C. Mathias Lucio E. Horie Enzo G. Frese Pedro V. Rocha Renato da P. Alves Marcos C. Antunes Léo V. C. Vasconcelos Isabel C. de Freitas Arthur B. S. de Souza Marcos C. Antunes Eduardo Baldi Ebert H. V. Melo Raquel Belchior F. N. Antônio Plácido C. C. F. João Pedro Gomes C. Alberto F. de Lima. L. S. Franciele Sembay da Conceição Carla S. Cosenza Lucas B. Germano Matheus Bozza Luis D. P. de Farias João G. O. C. de Melo Nicolas S. M. M. de Oliveira Gustavo C. K. Couto Herbert Azevedo Luiz R. L. Rodrigues Felipe W. V. da Silva, Antônio S. G. Pereira and Paulo F. F. Rosa. Roboime: From the top of latin america to robocup 2018. pages 1–16. Available at `https://github.com/roboime/roboime-tdp`.
6. Manuela Veloso James Bruce. Real-time randomized path planning for robot navigation. pages 1–6. Available at `http://www.cs.cmu.edu/~robosoccer/cmrobobits/papers/02iros-rrt.pdf`.
7. Timothy D. Barfoot Jonathan D. Gammell, Siddhartha S. Srinivasa. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 2997–3004, 2014. Available at `https://arxiv.org/pdf/1404.2334.pdf`.
8. Perttu Hamäläinen Kourosh Naderi, Joose Rajamäki. Rt-rrt*: A real-time path planning algorithm based on rrt*. pages 113–118, 2015. Available at `https://users.aalto.fi/~hamalap5/FutureGameAnimation/p113-naderi.pdf`.
9. Gabriel B. da Conceição Felipe W. V. da Silva Antônio S. G. Pereira Gabriel T. Pinheiro Gustavo A. Testoni Davi H. M. Pontes Daniel S. C. Bello Ana C. A. Monteiro José L. de O. Schramm Gustavo C. K. Couto Leonardo G. Gonçalves Lucas B. Germano Luiz R. L. Rodrigues Guilherme M. Gonçalves Vinicius de F. L. Moraes Gabriel H. M. Silva Mayara R. Mendonça Ana L. B. da Silva Matheus Bozza Luis D. P. de Farias-João G. O. C. de Melo Nicolas S. M. M. de Oliveira Lucas G. Corrêa, Carla S. Cosenza and Paulo F. F. Rosa. Roboime: From the top of latin america to robocup 2020. pages 1–21. Available at `https://github.com/roboime/roboime-tdp`.