NAMeC - SSL - Team Description Paper Small Size League RoboCup 2024 Application of Qualification in Division B

M. Bernet, E. Bouvier, A. Calugi, B.Chew, P. Félix, J. Gautier, C. Godinat, C. Gubanski, C. Labbé, J. Lindois, T.W. Meunier, E. Miqueu, N. Morera, E. Schmitz, C.A. Vlamynck

IUT - Université de Bordeaux, Gradignan, France
contact@etienne-schmitz.com (corresponding author)

Abstract. This paper outlines the latest developments and enhancements undertaken by NAMeC, the RoboCup Small Size League team located in the University of Bordeaux, France.

1 Introduction

Since our debut in the Robocup arena in 2018, NAMeC has participated in three competitions : Montreal, Sydney and Bordeaux. As we set our sights on the forthcoming 2024 RoboCup event in Eindhoven, this year's Team Description Paper (TDP) underscores our commitment to excellence and continuous improvement. We are making an incremental upgrade to our hardware and electronics system before a major overhaul scheduled for next year. Simultaneously, we are are rewriting our embedded system entirely in Rust to enhance performance, reliability and safety. In software development, we are focusing on the primary challenge faced by our team : debugging and visualization within our CRAbE framework.

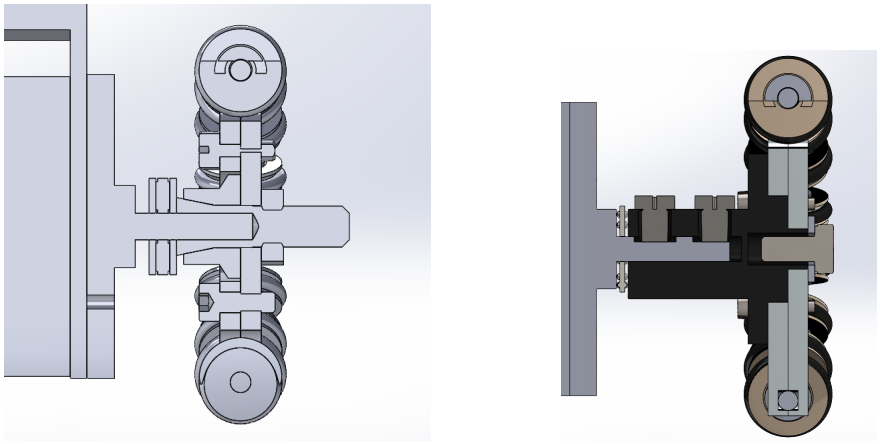
2 Mechanics

During our participation last year we identified a few vulnerabilities, particularly concerning the attachment of the wheels and the shell. These areas have become our primary focus for improvement this year, alongside structural modifications designed to facilitate the integration of new electronic PCBs scheduled for 2025.

2.1 Wheel attachment system

The previous wheel mounting mechanism, which utilized a pressure screw mounted to the engine axle, encountered multiple issues. The process of attaching the wheels was not only technically complex but also challenging to tighten properly and nearly impossible to align with precision.

Consequently, this method was replaced by a hinged system that significantly improved torque and force transmission. Additionally, this new system offers the benefit of being considerably easier to install.
The distinction between the two systems is illustrated in Figure 1.



(a) Old mounting system (b) New mounting system

Fig. 1: Comparaison of old and new mounting systems

2.2 Shell

The shell previously utilized by the team was crafted from a sheet of plastic (PEHD 300) which underwent laser cutting before being folded, affixed to a wooden base, then shaped into a circular form in compliance with regulations. However, this design had its drawbacks; it tended to shift during matches and was complex to assemble.

In response, we decided to upgrade to a new shell design, fabricated from a single piece of 3D-printed PP GF30 (Polypropylene Glass Fiber 30). This material not only offers enhanced durability but also ensures a more stable and rigid structure compared to the former version of the shell.
Similar approaches are employed by other teams, including Tigers Mannheim [7] and RoboJackets [5], in their robot designs.

3 Electronics

Our objective for this year is to internalize Printed Circuit Board (PCB) manufacturing capabilities within our team. Small improvements has been made to prepare 2025 and the next major update of our electronics.

3.1 Modular Architecture

During the previous year, all of our PCBs were produced by CATIE[1]. After thorough deliberation, we have decided to introduce a new electronic architecture for our robot, as illustrated in Figure 2.
This endeavor extends beyond mere improvement of the robot's electronics; it encompasses also the integration of a modular design paradigm. Such an approach will simplify the process for new team members to modify and repair modules, drawing inspiration from RoboFEI[3] or ER-Force[2].

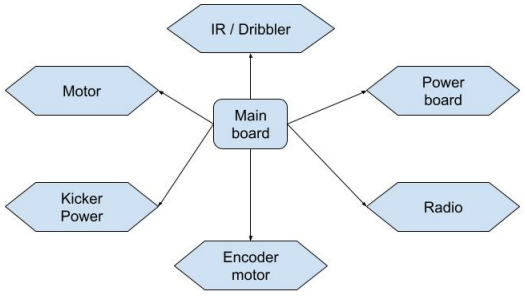


Fig. 2: New modular architecture

We also encountered issues with the integration of the robot :

- The radio connector poses a challenge as only two of the three ports can be connected due to the introduction of the new kicker card last year. Additionally, when we attempt to attach the NRF24L01+ connector, we face the inability to simultaneously flash our microcontroller via JTAG, as the port is obstructed by the radio module's PCB.
- The battery tester presents a risk of short circuit on the main board if improperly connected to the battery, as it may come into contact with the GND of the radio module.
- The direct connection of the battery to the motor power port exposes a deficiency in voltage regulation, potentially leading to uncontrolled power supply fluctuations.

[1] https://www.catie.fr/

– Some Jtag connectors in the motor card at the bottom of our robot short circuit when they touch the structure.

3.2 Battery tester

During the current year, improvements were made to the battery tester with the objective of mitigating the risk of short circuits on the mainboard. The new battery tester circuit is depicted on the figure 3.

On this circuit we extract the voltage of the cell individually, that cell voltage is also reduced by using the AOP, to reduce the utilization of the ADC pin on the MCU that we use to test the battery.
Our other goal for the competition is to improve the external encoder on the motor to remove the card from the bottom of the structure; that change may reduce the risk of interfering with the communication between the mainboard and the motor card.

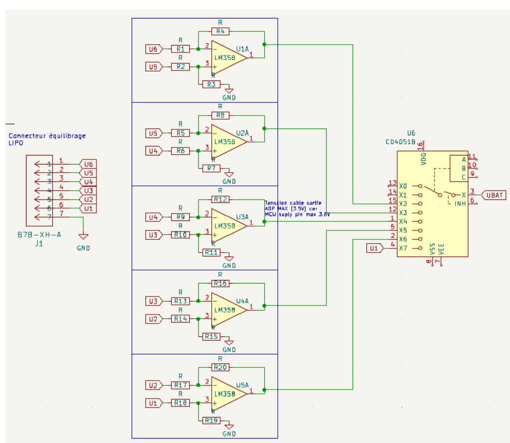


Fig. 3: New modular architecture

4 Embedded system

This year, we made significant changes in our embedded systems development, particularly by rewriting our firmware from C++ to Rust.. Our focus has been on analyzing the benefits of this change for our systems.
This section outlines the advancements made, provides a comparative analysis of Rust versus C++, and highlights specific features that have been improved or introduced during this transition.

4.1 Firmware Rewriting in Rust

Our firmware has been completely rewritten in Rust, a decision motivated by Rust's guarantees of memory safety, concurrency support, and overall performance.
This rewrite aims to leverage Rust's modern language features to create more reliable and efficient embedded systems for our robotic platform.
Similar initiatives have been undertaken by other teams, notably Luhbots Soccer Team[8] and A-team[1], as documented in their respective 2023 technical development papers (TDP).

Advantages of Rust over C++ The transition to Rust for our embedded systems development has introduced several key advantages, notably:

- **Memory Safety:** Rust's ownership model eliminates common bugs found in C++ due to mishandling of pointers and memory.
- **Concurrency:** Rust's adoption of asynchronous programming models facilitates safer and more efficient concurrent operations, significantly diminishing the risk of data races.
- **Direct Memory Access (DMA) Enhancements:** DMA allows the MCU to wait until a transfer is complete instead of handling each byte as it arrives. Due to its complexity, DMA sees less widespread adoption in embedded systems despite its performance advantages. Rust's safe abstractions, including its async features, make working with DMA more accessible and less error-prone compared to C++, enhancing our system's real-time data processing capabilities.
- **Pin Verification:** Pin and peripheral configuration is enforced at compile time which significantly reduces the risk of runtime errors resulting from hardware misuse.
- **Cargo Integration:** Utilizing Cargo to manage dependencies, automate builds, and incorporate community-developed crates into our firmware, streamlines development and enhances feature integration. Additionally, the seamless integration with probe-rs enables firmware to be uploaded and executed using the same commands as a native project, further simplyfing the development process.

These enhancements have collectively bolstered our embedded systems, making them more reliable, efficient, and easier to maintain.

However, it's important to note that Rust's embedded ecosystem is still relatively young. This means that support for certain hardware may be missing and some functionalities may need to be implemented from scratch.

4.2 Radio

Our new firmware will allow us to have bidirectional communication in order to both send commands and receive feedback from the robots. As we only have

one channel for receiving data from the 6 robots, we plan on implementing Time Division Multiple Access (TDMA) where each robot has a timeslot for sending data.
We will periodically send out a 'beacon' packet containing a timestamp, after which the robots will synchronise their clocks and await their respective timeslots for data transmission. For a detailed exploration of similar strategies, refer to the Tigers Mannheim work in their Extended Team Description Paper [6].

5 Software

An introspection into our previous competition in Bordeaux revealed a significant bottleneck in our team's performance : the absence of real-time feedback from our robots and a comprehensive system for their visualization.
This constraint not only obscured our comprehension of the robots' field behavior but also impeded our prompt issue identification. Recognizing the imperative necessity to rectify these challenges, our team initiated a meticulous overhaul of our framework, prioritizing the refinement of our visualizer. This endeavor aimed to augment real-time feedback and comprehensive system visualization.

5.1 Rewriting on our pipeline

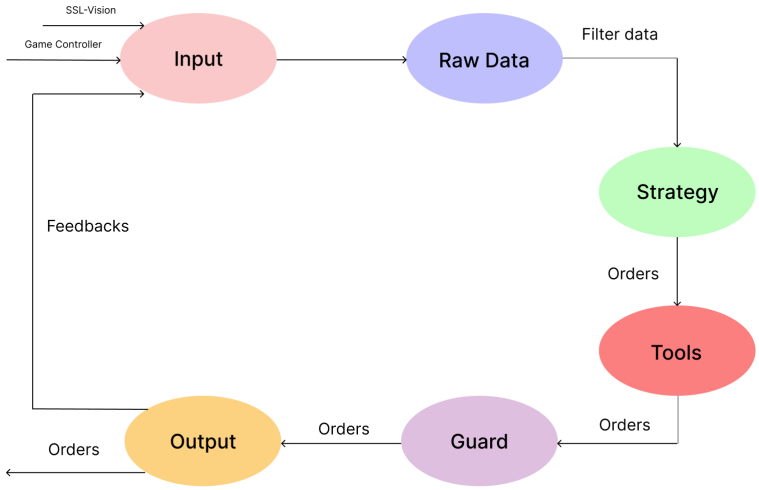
Since our migration to Rust, as discussed in our last Team Description Paper[4], the exigency for swift modifications within our codebase resulted in a state of disorganization and increased complexity. Recognizing the untenability of this situation, we opted to streamline and restructure certain aspects of our framework.
This decision was driven by our commitment to maintaining a clean, efficient, and manageable codebase, even in the face of pressing deadlines and the dynamic challenges of the competition.

With this recent overhaul, we have streamlined the architecture of our system into four primary components:

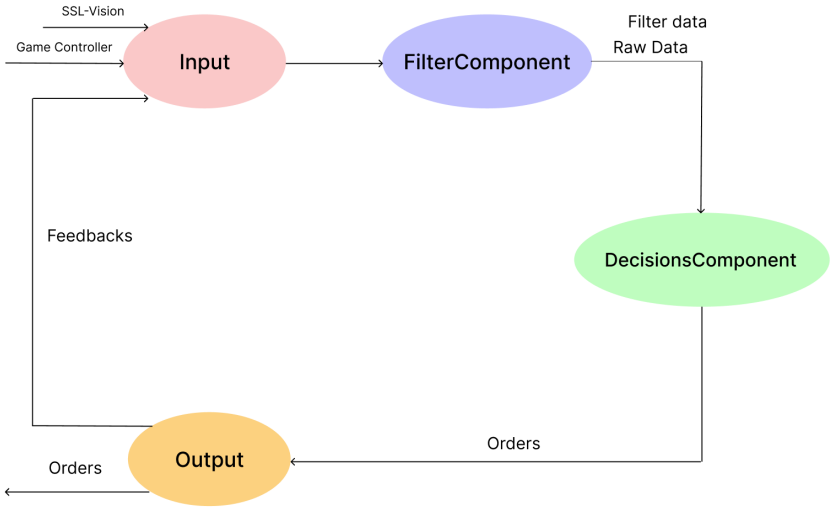
- **InputComponent** - Responsible for gathering and preprocessing input data (vision, gamecontroller, etc).
- **FilterComponent** - Filters and refines the data for more accurate processing.
- **DecisionComponent** - Analyzes the filtered data to make strategic decisions.
- **OutputComponent** - Executes the decisions, interfacing directly with the robots.

Notably, the previously separate *ToolComponent* and *GuardComponent* have been consolidated into the **OutputComponent**, enhancing efficiency and reducing complexity. This modification is illustrated in Figure 4.

For the FilterComponent, substantial enhancements have been made to optimize its functionality. Previously, the process was divided into three distinct stages:



(a) Old pipeline



(b) New pipeline

Fig. 4: Comparaison of old and new pipeline systems

– **Pre-filter**: This initial stage involved converting protobuf data into our custom types. To streamline operations, this functionality has now been integrated into the **InputComponent**.
– **Filter**: The core filtering stage remains within the FilterComponent, focusing on refining and processing the data more efficiently.

– **Post-filter**: The final stage, which involved organizing the filtered data into our *World* struct, has been re-evaluated for improved data integration and system coherence.

We have now implemented a dedicated pipeline for each object, which systematically organizes and assigns all relevant data directly to the object's corresponding 'world' field by reference.

5.2 Visualizer

In our previous Team Description Paper[4], we introduced our user interface project, Aquarium, which at the time was still under development. This year, we have made significant strides in its completion, employing the React framework to enhance its functionality. The primary improvements to Aquarium are focused on three key areas:

– **Log-replay**: A crucial feature that allows users to replay match logs for analysis and review. This functionality has been significantly improved by adopting JSON as the format for log storage. The use of JSON not only facilitates seamless integration and manipulation of log data but also facilitates the platform's compatibility with various data sources and tools, making it an invaluable tool for debugging and strategy refinement.
– **Controller Integration**: We have introduced support for controllers to send commands through our software
– **Field Visualization**: The field visualization has been significantly improved, now featuring more detailed and accurate representations of the game environment, with added functionality to zoom and rotate the field for an improved viewing experience.

6 Conclusion

In this Team Description Paper, we highlighted the significant advancements made in preparation for the 2024 RoboCup event in Eindhoven. Our focus spans across multiple fronts, including mechanics, electronics, firmware, and software.

Mechanically, we've addressed vulnerabilities related to our wheel attachment and shell designs, resulting in increased stability and durability. Our electronic upgrades, particularly the adoption of a modular architecture, have strenghthened reliability by effectively mitigating risks associated with battery testing and motor card interference.

With the transition of our firmware development from C++ to Rust, we expect to see improvements in terms of reliability and performance.

Furthermore, our software enhancements have delivered tangible benefits, notably in real-time feedback and system visualization. By integrating features such as log replay and enhanced field visualization into our Aquarium visualizer, we've significantly augmented our ability to analyze on the field.

In conclusion, through our concerted efforts in mechanics, electronics, firmware, and software, we aim to make a meaningful contribution to the Small Size League.

7 Acknowledgment

We would like to thank the Nouvelle-Aquitaine Region, the University of Bordeaux, the Bordeaux Institute of Technology[2], and especially the Department of Computer Science for their support for the NAMeC project[3].

A big thank you to the former members of NAMeC and our sponsors for their support.

[2] IUT: https://www.u-bordeaux.fr/en/about-us/organisation-operations/training-components/institute-of-technology

[3] NAMeC: Nouvelle Aquitaine Mécatronic Club

References

1. C. Avidano, S. Barnette, M. Barulic, L. Medrano, J. Neiger, R. Osawa, E. Peterson, J. Spall IV, J. Spalten, W. Stuckey, and M. Woodward. *The A-Team Technical Description Paper 2023*, 2023.
2. P. Bergmann, T.Engelhardt, T. Heineken, V. Hopf, M. Schmid, M.Schmidt, F. Schofer, K. Schuh, and M.Stadler. *ER-Force 2022 Extended Team Description Paper*, 2022.
3. L. da S. Costa, W. de S. Motta, D. Pilotto, G. L. Pauli, J. V. L. Aguiar, G. M. Schiavetto, B. Bollos Correa, M. A. P. Laureano M. Mariano Lucatelli, R. A. C. Bianchi, P. Thomaz Aquino Junior, , and Flavio Tonidandel. *RoboFEI 2020 Team Description Paper*, 2020.
4. P. Felix, O.Ly G. Passault, E. Schmitz, S. Loty, C.Laigle, A.Chauvel, T W.Menier, V. Chaud, L. Paille, E.Miqueu, B.Chew, J.Gautier, T.Decabrat, R.Denieport, and P.M.Ancele. *NAMeC - Team Description Paper Small Size League RoboCup 2023 Application of Qualification in Division B*, 2023.
5. B. Perez, D. Lam, S. Challa, K. Fu, H. Gynai, A. Sohrab, C. Clark, A. Srinivasan, and A. Gordon. *RoboJackets 2022 Team Description Paper*, 2022.
6. A. Ryll, M. Geiger, N. Ommer, A. Sachtler, and L. Magel. *TIGERs Mannheim Extended Team Description for RoboCup 2016*, 2022.
7. Andre Ryll and Sabolc Jut. *TIGERs Mannheim - Extended Team Description for RoboCup 2020*, 2020.
8. L. Seegemann, F. Zeug, P. Ebbighausen, L. Waldhoff, M. Westermann, S. Knackstedt, M. Känner, T. Füchsel, R. Hart, and T. Pahl. *Luhbots Soccer - Team Description for RoboCup 2023*, 2023.