NAMeC - Team Description Paper Small Size League RoboCup 2023 Application of Qualification in Division B

P. Félix[1], O.Ly[1], G. Passault[1], E. Schmitz[2], S. Loty[3], C.Laigle[3], A.Chauvel[3], T W.Menier[1], V. Chaud[1], L. Paillé[1], E.Miqueu[1], B.Chew[1]

[1] IUT - Université de Bordeaux, Bordeaux, France
[2] ENSEIRB-MATMECA, Bordeaux-INP, France
[3] CATIE, France
`contact@etienne-schmitz.com` (corresponding author)

Abstract. This paper presents a short overview of the current state and goals of the SSL RoboCup NAMeC team.
The team has been competing since 2018 with a big break made since the COVID-19 pandemic and wants to relaunch the team in the 2023 competition scheduled on our city.

1 Introduction

The NAMeC team faced challenges competing in the last three years, mainly due to the impact of COVID on training as a result of successive lockdowns, the departure of team members who had graduated.
At the same time, a second team called NAELIC created by a large number of students has failed at the first participation in Robocup 2021.
During this year and the big turn-over we have faced, new students and some former members in support wants to participate on the team and relaunched the team for the following years.
Starting this year, we revised our approach for the competition to avoid mistakes that we made in previous years. We recognized that the coding and the debugging process had become too complex for students and detracted from the goal that we set when we launched the team[3].
To address this issue, we have simplified our process and reduced our ambitions to give priority to a robust robot that makes things simple before moving towards some of the more complex fields of robotics.
We made a process to document all of our decisions to avoid a situation like the transition that we went through in the last years, in case of high turnover in the following years.
Additionally, we are working to lower the barrier of entry to the SSL, making it more accessible to students and fostering the next generation of competitors without impact to scientific rigor and fairness in competitive matches.

2 Mechanics, Electronics and Firmware

During this third year, we made some slight changes in the mechanics of the robots. At the same time, we recreated a new version of electronics and we rewrote the firmware of our robots entirely.

2.1 Rework of some mechanical part

Since our last participation we took the same robot that we have at Sydney, inspired by some division A's team and improved the fragile parts of our robots. We changed the wheel fixation, the magnetic part of the kicker and the reloading part of the kicker.

The old wheel system used the strength of a straight collet but had many disadvantages. To tighten the collet we need to strongly tighten the nut which holds the wheel. But for it to be tight enough to have a good fixation, we needed to block the rotation of the wheel and the motor. We decided to change this fixation to a new piece and a new motor axis. The axis has a flat part and the new piece uses a pressure screw to secure the wheel.

The kicker axis had two parts, the rear half being magnetic and the front half not, but the junction lacked strength and broke regularly.
We chose to change it to one piece dug onto the rear half of the part in order to place a magnet.

In our last iteration of our robots the fixation for the reloading part of our kicker was made in 3D printing and in other side serves to support the dribbler card.
Because of the disappearing of the dribbler card and the fragility of this part, we take the idea of the fixation part[4] made by some members of the NAELIC team.
In the future, we will try to remove elastics and use springs instead, but at this time, we have not tested this part.

2.2 Redesign of the electronics parts

For the past years, we have been working on a new version of our electronic card that handles some issues found during our last participation on Sydney and the last TDP[2].
To make some space in our robot, we put the micro controller of our dribbler inside the new mainboard.
The last iteration of our kicker card emits some electromagnetic noise with high power kicks and resets the motor cards.
We need further investigate how to handle the issue but we have made some improvement in this area.

Motor cards At the moment, the cards controlling the motors have a micro controller STM32F1, we will replace it with a STM32L4 to work with floating point numbers for our upcoming work.
During the Sydney, an ancient member of our teams worked on field-oriented control[8] to work at low-speed. It was not deployed at Sydney because there were still some issues.
The person that had done the job left the team and at the same time, we began to work on a new card that uses the Trinamic components(TMC)[13] to handle the issue and doesn't use our own motor card.
During our tests, we can use all of the motors when he was outside of the robots. But when we mounted with the kicker part, we have some trouble and the card was made in an unknown states.
We need to continue to investigate more and for the moment, we use a PID control. We put a CRC check-up to measure a number of packets corrupted by SPI transmission. We see that 1% of packets are corrupted during a long-session of usage of robots.

2.3 Reworking of the firmware

Because of the change of the robot's micro controllers in all cards, we rewrote our firmware entirely
We use the STM32CUBEMX for motors and dribblers card and MBED for the micro controllers that handle all of the mainboard.
With this rewriting time, we made the debugging process more user-friendly and well-documented.

Debug process We want to know when we start the robots, if all the parts were passed well and they have no errors.
We use our buzzer and different frequencies in the mainboard to handle these methods.

At the same time, we made extensive use of the JLINK Suite Software for our debugging toolkit, especially RTT Viewer and Ozone.

During development mode, we want to debug easily the systems by using the functionality of virtual terminals provided by RTT Viewer[11].
The idea was to see the current state of the mainboard card on the PC side, which robots were connected, and some useful information printed in virtual terminal and errors occurred.

3 Software

This year, our efforts are concentrated to lower the entry barrier, improve the control of our robots and visualize the current state of our software.
We want to make sure our robots doesn't make a lot of mistake during the playing time. We implemented safeguards and visualizing process to handle the issue listed before.
We work essentially on implementing correctly our Kalman filters and and obstacle avoidance program.

3.1 Rewriting in RUST

Before this year, all of our software was written in C++ and the rewriting of our software during the past year made that we lost a lot of functionality since Sydney.
With the idea to lower the barrier to entry, to have a robust language to handle error-handling before compiling time and to keep the performance of a low-level language, we made the choice to rewrite our AI framework entirely a final time in Rust due to its numerous advantages
Rust's ownership system allows memory safety to be handled at compile time. Additionally, Rust has numerous tools such as Cargo and rustup that simplify our process like dependency management and our cross-compilation. In figure 1, we see our architecture of our software.

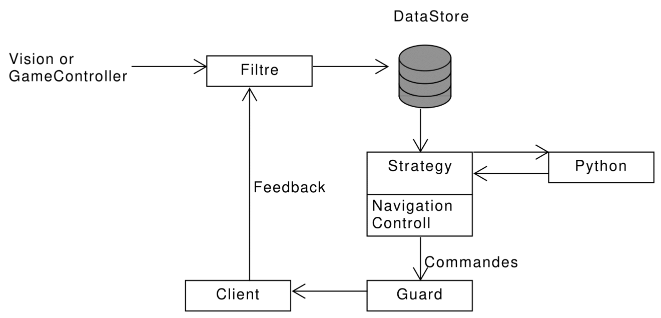

Fig. 1. Architectural diagram

3.2 Visualization process

During our debugging process in previous year, we lack a visualization process. For a team, it's important to have the possibility to replay previous match and see a lots of data to understand what it is happens during this match.

We created a software that we call aquarium, our UI that controls and view the state of our games.
The functionality of this software has

- Visualize data graphics to our state of AI and strategy times
- Visualize our field and annotate with some useful information.
- Control our robots, teleport our robots during simulation time
- Replay previous match with log files
- Can be used to visualize JTAG output directly

3.3 Control loop and Obstacle avoidance

During the following years, we see the importance to have well-filtered data for our strategy and our control loop.
We want to make an effort to implement a Kalman filter[9] this year and at the same time to have a robust obstacle avoidance that permits to made a simple actions such as kick, dribbler and go to at a position.
For obstacle avoidance, we have the objective to implement this paper[1]. If we have a robust control pipeline, we can in following years work on improving the capabilities of our robots.

3.4 Reinforcement Learning

On experimental field, some of our members work on reinforcement learning. The idea is to make a Robocup SSL environment, which will be a multi agent environment, train a model to take the position of all robots and return actions as a function of time.
The main problem is what we would use to get realistic physics. Available simulators aren't fit for such a task and we will need to build a custom environment. We started exploring two different paths, a 2D physics based simulator, using the Box2D physics engine and one which would use Mujoco[12] as the physics engine.
For the reinforcement learning environment API, we looked at gym, which is a popular reinforcement learning environment python library developed by OpenAI. Gym is widely supported by reinforcement learning libraries such as Stable Baselines 3 and Tianshou and would be easy to integrate.
OpenAI stopped maintaining gym and the Farama foundation[5] now develops it as gymnasium. The members of Farama helped with questions we had about the framework and the other tools they are developing. We plan on using Tianshou for the reinforcement learning algorithms with either PettingZoo [6] (multiagent gymnasium) or dm_control [14] for the env base. We can get the robots models in mujoco using onshape-to-robot[7] , as mujoco accepts urdf. Mujoco can then be used to generate a mjcf xml, which we can then split into components and use with dm_control's PyMjcf to make a Robot model in python. We can then make a dm_control env using dm_control.locomotion.soccer envs as reference. Once the env is working, we can benchmark multiple RL algs on it such as DQN and PPO, which are the two we want to try out for now.

4 Education

Educational robotics is now approachable in most schools. It allows theoretical problems to be illustrated in practical and concrete applications, and above all makes it easier to access higher level problems such as navigation.

RSK[4] is an educational omniwheel robot kit[10] proposed by Grégoire Passault from the Rhoban team[5], equipped with an external tracking system. This allows users to discover some higher level aspects of the SSL league. To fulfill our goal of bringing in new competitors, we associated a movement with the RSK. We created a process that allows the code produced in this league to be ran with our robots and the ssl-vision pipeline.

4.1 RSK Project

The RSK – Robot Soccer Kit – is a simplified soccer competition, derived from the SSL, which allows the youngest programmers to participate in a robotic soccer competition.

- 4 omniwheel robots (omnidirectional): two per team, equipped with a kicker allowing them to shoot an 8g foam golf ball
- A printed carpet and a stem equipped with a vision system (1 camera)
- A localization system is provided, as in SSL: markers on the interchangeable cover are spotted by the camera.
- Using the game controller software, the robots and the ball are spotted on the field. Robot control is done through a Python API for navigation
- Communication via Bluetooth

Such an environment erases some of the difficulties of SSL by managing localization and providing turnkey bots. This favours the higher level aspects of soccer by providing a Python API for navigation.

The RSK project is entirely open-source: software, electronics and mechanical design. Everything is available on GitHub[6].

This new league does not officially exist but is a proposal in the process of being validated by the various committees of Robocup Junior.

4.2 Welcome young Robocuper

We have created a bridge to SSL using our SSL infrastructure:

- SSL-vision
- The covers of RSK robots are replaced by SSL covers
- For communication, information sent via radio waves is sent via Bluetooth

[4] Robot Soccer Kit: see https://robot-soccer-kit.github.io/ for all material and videos.
[5] Rhoban: a four champion team in the kid-size soccer humanoid league. See https://www.rhoban-project.fr/
[6] https://github.com/rhoban/junior-ssl

– The Python API is implemented in our framework rest
– A suitable ground of 6 m by 4 m

Additionally, we created a Python layer to work on the strategies to make a transition between the usage of the Python API in SSL Junior and our Rust base code during the first months for a new member. All new members of the current team have participated in RSK competitions: this year we will continue to train young people for the junior Robocup.

5 Conclusion

We want to thank the CATIE, and especially the Rhoban team for helping us during this change of the team.

We would like to thank the Nouvelle Aquitaine Region, the University of Bordeaux, the Institute of Technology of Bordeaux[7], and especially the Department of Computer Science for their support for the project NAMC[8].

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

[7] IUT: https://www.u-bordeaux.com/Education/Colleges-Institutes/Institute-of-Technology
[8] NAMC: Nouvelle Aquitaine Mécatronic Club

References

1. Lounis Adouane. Toward Fully Autonomous Vehicle Navigation: From Behavioral to Hybrid Multi-Controller Architectures. . *11th International Workshop on Robot Motion and Control*, 2017.
2. J. Allali, J. Bezamat, P. Félix, S. Loty, X. Muller V. Mignot, R. Pigret-Cadou, T. Saliba, and E. Schmitz. *NAMeC - Team Description Paper Small Size League RoboCup 2020 Application of Qualification in Division B* , 2020.
3. A. Boussicault, L.Hofer P.Félix, O.Ly, S N'Guyen, G.Passault, and A.Pironne. *AMC, Team Description Paper Small Size League RoboCup 2018 Application of Qualification in Division B* , 2018.
4. G. Passault et al. E.Schmitz, P. Félix. *NAELIC - Team Description Paper Small Size League RoboCup 2021 Application of Qualification in Division B*, 2021.
5. Farama foundation. `https://farama.org/`, Last accessed on 2023-01-28.
6. Farama foundation. `https://pettingzoo.farama.org/`, Last accessed on 2023-01-28.
7. Rhoban Github. `https://github.com/Rhoban/onshape-to-robot`, Last accessed on 2023-01-28.
8. J.Allali, J. Bezamat, A. Boussicault, P. Félix R. Denieport, O. Ly, S. N'Guyen S. Loty, V. Mignot, G. Passault, and T. Saliba. *NAMeC - Team Description Paper Small Size League RoboCup 2019 Application of Qualification in Division B* , 2019.
9. R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.
10. Grégoire Passault, Clément Gaspard, and Olivier Ly. Robot soccer kit: Omniwheel tracked soccer robots for education. In *2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 34–39, 2022.
11. SEGGER. `https://www.segger.com/products/debug-probes/j-link/tools/rtt-viewer/`, Last accessed on 2023-01-28.
12. Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.
13. Trinamic. `https://www.trinamic.com/`, Last accessed on 2023-01-28.
14. Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.