

# RoboFEI 2019 Team Description Paper

Guilherme P. de Oliveira, Vinícius M. Alves, Danilo Pilotto, Wesley de S. Motta, Leonardo da S. Costa, Guilherme Luis Pauli, Marcos A. P. Laureano, Carlos Alberto F. Cadamuro, Reinaldo A. C. Bianchi, and Plínio Thomaz Aquino Junior and Flavio Tonidandel

Robotics and Artificial Intelligence Laboratory  
Centro Universitário da FEI, São Bernardo do Campo, Brazil  
{flaviot, rbianchi, plinio.aquino}@fei.edu.br

**Abstract.** This paper presents the current state of the RoboFEI Small Size League team as it stands for RoboCup International Small Size League competition 2019, in Sydney, Australia, as well as works that are still under development. The paper contains descriptions of the mechanical, electrical and software modules, designed to enable the robots to achieve playing soccer capabilities in the dynamic environment of the Small Size League.

## 1 Introduction

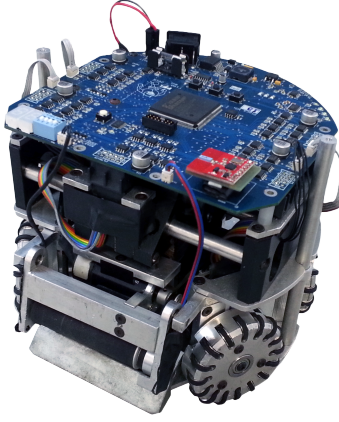
For the RoboCup 2019, the RoboFEI team intends to use the same electronics and mechanical design that have been used over the last years, as shown in Figure 1. Some significant advances were made in our software and firmware, which was verified in the XVII Latin American Robotics Competition (LARC). We have also been able to assemble the first prototype of our new mechanical structure. Our aim now is to improve the recently built software system and the mechanical design. With our researches, we hope to bring innovations and new ideas for the community.

## 2 Electronics

Since RoboCup 2014, RoboFEI's team released the current electronic design as open source to the community. All the schematics, layouts and firmware are available on-line, under a Creative Commons license, in RoboFEI's web-site ([www.fei.edu.br/robofei](http://www.fei.edu.br/robofei)). Currently, the electronic hardware remains the same update of last year, the components description are shown in table 1.

### 2.1 Updated Wheel Control System

We currently use a distributed control system, where, some decisions and controlling actions are made in the high-level AI software, such as determining which velocity the robot should move to reach a desired point in the field, and another



**Fig. 1.** RoboFEI's Robot.

**Table 1.** Current Electronic Description.

Device	Description
Main Board	CPU: Xilinx Spartan 3 FPGA operated with 50 MHz clock
MOSFET Driver	TC4427 working with IRF7389 MOSFET
Current Sensor	ACS712 with A/D converter AD7928
Kick Board	Boost converter topology.
Kick Storage	Two capacitors of 2700 $\mu$ F, in parallel, charging up to 160V in 16s.
Kick MOSFET	IRFSL427, Id 72A DC.
Driving Motor	Maxon,EC-flat-45 50W with hall sensor.
Dribbling Motor	Maxon EC-max-22 25W with hall sensor.
Ball Sensor	TEFT4300 infra-red emission diode and photo diode pair.
Communication	nRF24L01 transceiver, 2 Mbps, 2.4/2.5 GHz.
Power Supply	LiPo, 3-cells (11.1V ) and 2250 mAh capacity.

part of the control is done in the embedded firmware. To improve the performance of the robot it was studied and delimited what each part should do and how.

We fixed that all robots should receive its movement information in its own referential as  $V_x$ ,  $V_y$  and  $\dot{\omega}$ , which are its velocities on  $X$ ,  $Y$  axes and the angular velocity on its own axis respectively. The referential used was that the  $Y$  axis is perpendicular to the front of the robot and the  $X$  axis is to the right of the robot.

In order to do this, all of the kinematics were remodeled and programmed into the firmware, this can be seen in Equation 1.

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} -\text{sen}(\phi) & -\text{cos}(\phi) & 1 \\ \text{sen}(\phi) & -\text{cos}(\phi) & 1 \\ \text{sen}(\phi) & \text{cos}(\phi) & 1 \\ -\text{sen}(\phi) & \text{cos}(\phi) & 1 \end{bmatrix} * \begin{bmatrix} V_x \\ V_y \\ \dot{\omega} \end{bmatrix} \quad (1)$$

Where  $v_i$  is the velocity for each of the four wheels and  $\phi$  is the angle of each wheel, which in our case is  $33^\circ$  for the old structure and  $32^\circ$  for the new one.

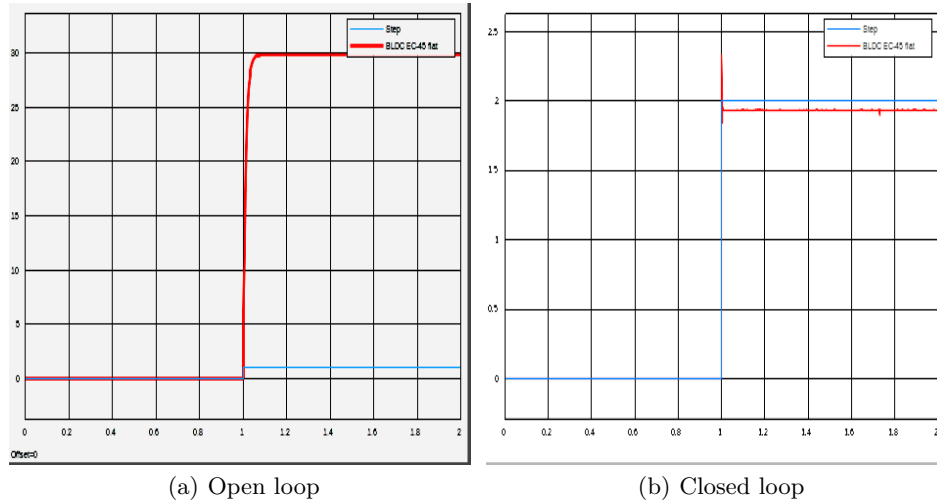
We also parameterized all velocity values on firmware and high-level software to work on the range from  $-100$  to  $+100$ , where each 10 units represents approximately  $0.2m/s$  for the  $X$  and  $Y$  axis.

To obtain the transfer function of the motor we've used its data sheet. The transfer function used by the team is shown in Equation 2 and was implemented using MATLAB and Simulink.

$$\frac{\dot{\theta}(s)}{V(s)} = \frac{33.5 * e + 3}{77.355 * e - 4 * s^2 + 13.203 * s + 1122.25} \quad (2)$$

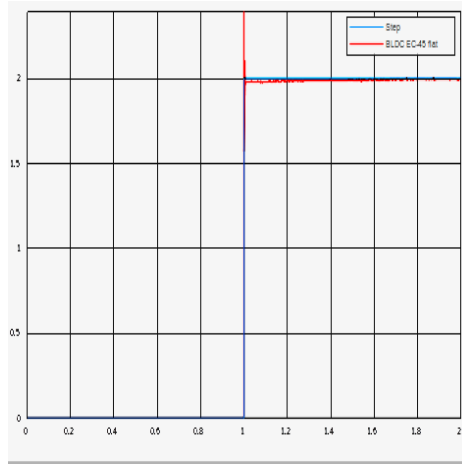
Where  $\dot{\theta}(s)$  is the angular velocity of the motor,  $V(s)$  is the voltage applied,  $e$  is the Euler constant and  $s$  is the Laplace complex variable.

The step response in open loop of the system is shown in Figure 2(a) and in closed loop in Figure 2(b). Where the red curve is the system output and the blue one is the step input.



**Fig. 2.** Step response.

Utilizing the recommendations for sintonizing controllers it is possible to have zero steady-state error and decrease the overshoot in closed loop [2]. The system output in closed loop with a PI controller can be seen in Figure 3. Where the red curve is the system output and the blue one is the step input.



**Fig. 3.** System output after sintonizing the controllers.

The overshoot right after the step occurs given that there is no load on the motor and is not relevant to the performance of the robot. The equation of the controller can be seen in Equation 3.

$$u(s) = 3 + \frac{5}{s} \quad (3)$$

And the gains are:  $P = 3$  and  $I = 5$ .

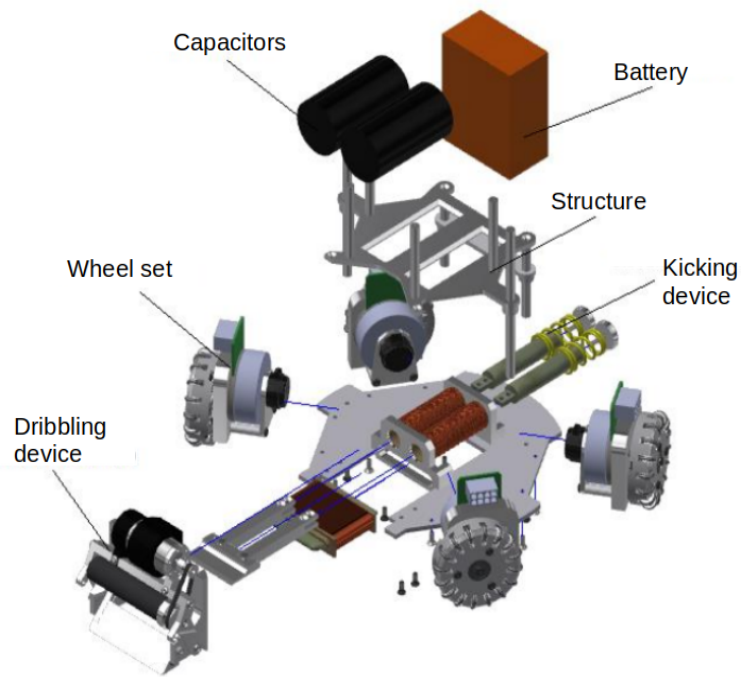
We've noticed that after the new controller we are able to reach higher velocities than before while maintaining control of the robot.

### 3 Mechanics

We've developed a new mechanical structure last year, however it still needs some adjustments before testing in an actual game. The new structure is in most part made of 3D printing, which makes our robots lighter, faster and cheaper than the old ones. The new structure can be seen in Figure 4. Currently we aim to participate in division A and be a competitive team in it, to achieve this most of our effort are in reducing the cost to produce more robots whilst maintaining its quality, in order to reach eleven robots.

Our new structure is based on a methodology called Reliability-Centered Maintenance (RCM). It is defined in [3] that RCM is a process used to determine what must be done to ensure that any structure continues to do what the users want throughout its whole life cycle.

The main results expected when using RCM are: improvement in the quality of the product, greater efficiency in maintenance and longer life cycle for expensive actives [3]. All of these aspects have great importance in our case where



**Fig. 4.** Exploded view of the new structure.

maintenance time is crucial between games and there are a lot of expensive parts in our robot, for example, the motors and some machined parts.

Using this methodology the new structure has reduced the maintenance time in 14% and now includes the directional kick. Since we plan on having eleven robots in the future, it's important that the maintenance of each doesn't take much time.



**Fig. 5.** Final geometry of the encoder support in blue.

Changes were made on the encoder support in order to avoid their misalignment, the new support can be seen in Figure 5, they are now fixed by two rods in the lower part of the motor support, this way we avoid shocks caused by other robots to interfere with the encoders. The support must be made of aluminum since the encoder plate holes comes from the manufacturer with 2.5mm metric threads, since after a few tests with 3D printing we found that trying to print threads smaller than 3mm would not give us a good result no matter how we did the configuration of the printer.

## 4 Software

As said in our latest TDP [4], we were developing a whole new software for the team. Now we are almost done implementing all the basic functions needed, a few of the new features will be shown here.

### 4.1 Software Structure

Given that handling all the interface events can use quite a lot of processing in some situations, therefore we've decided to have multiple threads handling certain parts of our software. In the current moment we have three threads in total, in which, the first is used to handle everything related do the interface, receiving data from the SSL Vision and SSL Refbox, sending and receiving data to the robots. The second thread is used to process the states of the game, Referee commands, calculate the positioning of the robots, decide which robot should be a defender or attacker, calculate the position of the goalie and trajectory of each robot using the path-planner. And the third thread is used only to maintain a real-time loop for the motion control of the robots.

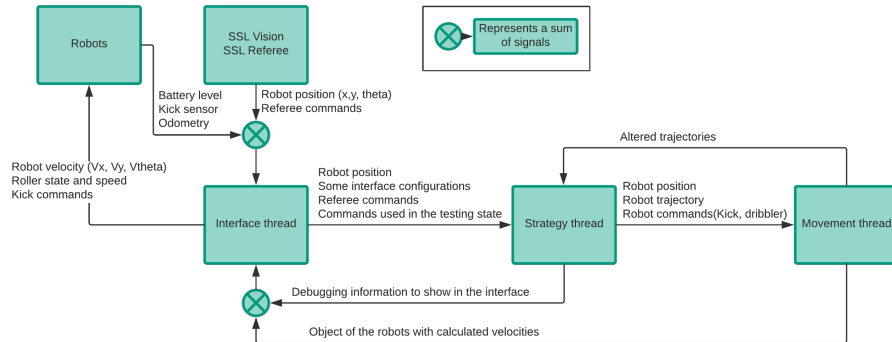


Fig. 6. Current software structure.

We still aren't using a sophisticated controlling technique to control the position of our robots, but in a near future when it is implemented there will already

be a functional environment to run the controller, currently this real-time loop runs in an interval of less than  $15ms$ , and it is fixed that it will only return an information in the end of this period, this guarantees for us that approximately every  $15ms$  our robots are receiving information correcting their velocities. In the Figure 6 is it illustrated how this structure works.

## 4.2 User Interface

Some problems we've had with our past software versions were the lack of organization on the interface and absence of configurable characteristics of the robots and skills.

The new software has only three tabs. The first is the tab used in game game, it contains the image of the field, some information's about the robots, like battery level, kick sensor and current skill, this tab can be seen in the Figure 7. The second tab is used for configuration of all the parameters available, a few of them are, serial port of the radio, SSL Vision and SSL Referee connection parameters, some constants used in skills and many others, this tab can be seen in Figure 8. The third tab is completely dedicated to do various tests with our robots, this facilitates discovering bugs in the software and mechanical/electronic problems with the robots such as bad contact in some connections of the board, this tab can be seen in Figure 9.

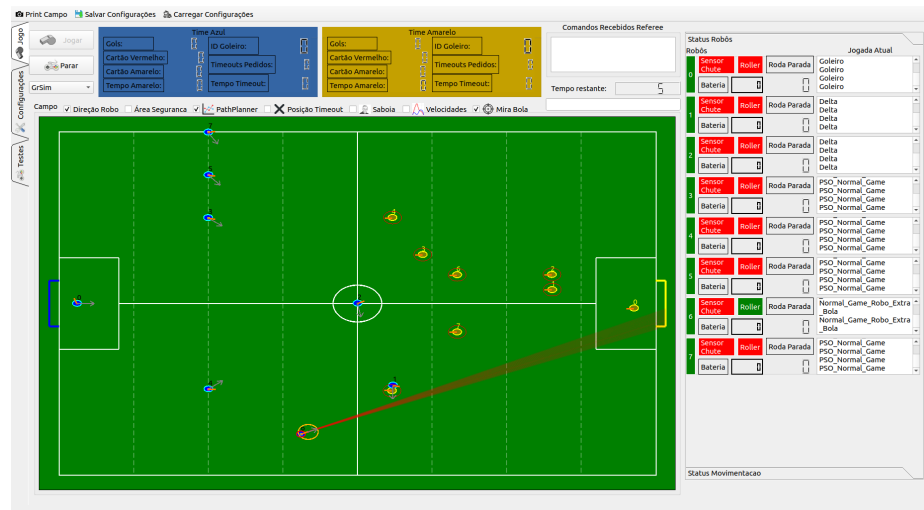


Fig. 7. In game tab.

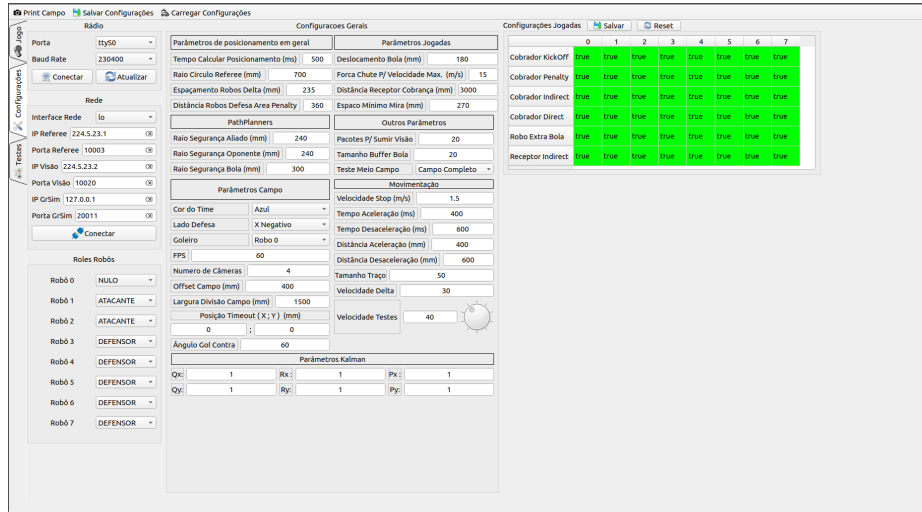


Fig. 8. Configuration tab.

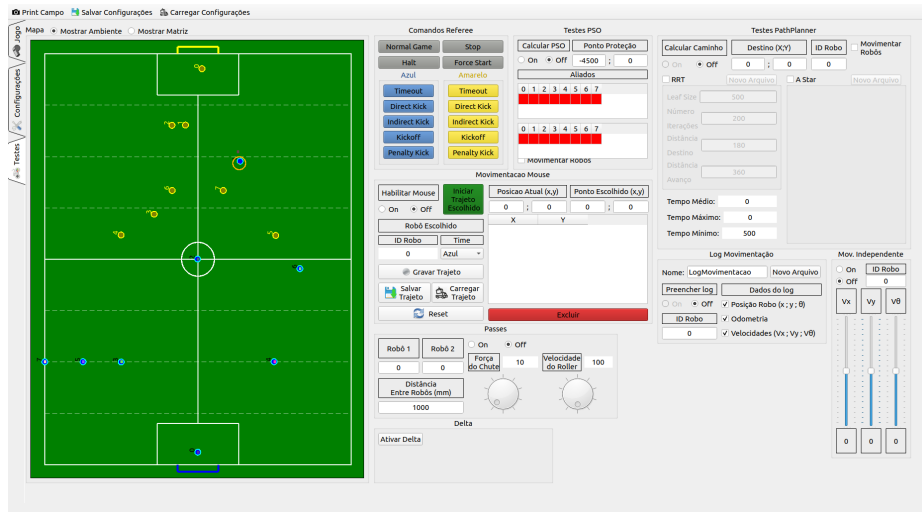


Fig. 9. Testing tab.

### 4.3 New Positioning System

The PSO proposed by [5] is an optimization algorithm based on a population of particles that have obtained recognition in the solution of several problems, with simplicity and using few computational resources. The PSO algorithm objective finds an optimal solution in a search space, through the exchange of information



between individuals of a population determining which trajectory each individual should take in the search space.

The velocity  $V$  and the position  $P$  of the particles is updated from the equations 4 e 5 respectively:

$$V_{ir}(t+1) = \omega V_{ir}(t) + c_1 r_1 (pbest(ir, t) - P_{ir}(t)) + c_2 r_2 (gbest(t) - P_{ir}(t)) \quad (4)$$

$$P_{ir}(t+1) = P_{ir}(t) + V_{ir}(t+1) \quad (5)$$

Where  $V$  denotes velocity,  $ir$  is the robot  $r$  from particle  $i$ ,  $\omega$  is the inertia factor used to balance global and local exploration,  $r_1$  (cognitive part)  $r_2$  (social part) are randomly distributed in the range  $[0, 1]$ , and  $c_1$  (confidence parameter for the cognitive part) and  $c_2$  (confidence parameter for the social part) are constant parameters called acceleration coefficients.

According to [6], a soccer match is divided into two attitudinal concepts: defense and attack. Being that defense of a team must a) cancel the finishing situations, b) recover the ball, c) prevent the opponent from progressing, d) protect the goal and e) reduce the opponent's playing space; and in the attack: a) maintain possession of the ball, b) build offensive actions, c) progress through the opponent playing field, d) create situations of finalization and e) finalize in the opponent's goal. These attitudinal concepts and other tactical principles will be prioritized in the development of objective functions applied to the defense positioning.

To test the possibility of applying the PSO algorithm, in LARC 2018 some mathematical functions were implemented to search for defense positions according to the following rules:

- The minimum distance of the robots in relation to the opponent to make it difficult to move and the possibility of receiving or making passes or kick to goal;
- If the view of the robots in relation to a certain point (in this case, the goal) is blocked;
- If all opposing robots have at least one allied robot blocking the view of the goal, prioritizing the opposing robot with possession of the ball.

Initially, the grSim simulator [7] was used to validate the proposal in a real game situation as seen on Figure 10.

The algorithm runs every 200ms and the previous solution is always passed to the new run. The algorithm returns the positions of the robots and the strategy system decides which robot moves to each position (based on the shortest distance).



Fig. 10. grSim.

Tests conducted during LARC 2018 have shown that this approach can be used to find field placements during an SSL soccer game, especially in non-stop play. However, with the active game it is necessary to predict the possible movements of the opposing robots to calculate the future positioning of the robots of the team, being this the main challenge for the next competition. Other objective functions are being developed and tested to improve defense positioning (without possession of ball) and also in attack (with possession of ball).

## Acknowledgements

We would like to thank, in advance, the Small Size League Committee, for the consideration of our material. We would like also to immensely thank the staff of Centro Universitário FEI, for all the help we always received from them.

## References

1. Mathew MacDougall, Gareth Ellis, Amy Hashemi, Emma Jackson, Oon Chu Lip, Nicolas Ivanov, James Petrie, Khashina Tonks-Turcotte, Chantal Sousa, Kenji Lai, Bowen Xu, Qiqi Li, Eric Goto, Dana Deutsch, Anthony Buonassisi, and Marcus Lee. 2018 team description paper: Ubc thunderbots. ., 2018.
2. Karl Johan Åström and Tore Hägglund. *PID controllers: theory, design, and tuning*, volume 2. Instrument society of America Research Triangle Park, NC, 1995.
3. John Moubray. *Reliability-centred Maintenance (RCM)*. Lutterworth, Inglaterra Industrial, 2000.

4. Guilherme P. de Oliveira, Vinicius M Alves, Danilo Pilotto, Lucas A. da Silva, Julia B. Drapella, Danilo R. Vassoler, Igor do N. Alves, Raphael B. Bezerra, Fernando T. H. Darsono, et al. Robofei 2018 team description paper. ., 2018.
5. J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, Nov 1995.
6. Israel Teoldo da Costa, Julio Manuel Garganta da Silva, Pablo Juan Greco, and Isabel Mesquita. Principios taticos do jogo de futebol: conceitos e aplicacao. *Motriz. Revista de Educaçã~o F~sica*, 15(3):657–668, julho 2009.
7. Valiallah Monajjemi, Ali Koochakzadeh, and Saeed Shiry Ghidary. *grSim – RoboCup Small Size Robot Soccer Simulator*, pages 450–460. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.