UTBots VSSS 2022 - The development of robots for the Very Small Size Soccer category - Latin American Robot Competition

Eduardo K. P. Veiga[1], Victor H. A. Cardoso[2]
André S. de Oliveira[3], Jean M. Simão[4] and João A. Fabro[5¶]

***Abstract* - This document provides details about the development process of the UTBots VSSS team of soccer playing robots, from Federal University of Technology - Paraná (UTFPR - Universidade Tecnológica Federal do Paraná - Campus Curitiba) for the Very Small Size Soccer (VSSS) category. This paper presents the omni-directionas, four wheels robot design, that is controlled by an Arduino NANO with an wireless module connected for communication between the robots and the AI control systems. This control systems is based on the emergent Notification Oriented Paradigm(NOP), receives its input information directly from a Vision subsystem also developed by the team, and sends the low level movement commands to the robots by a dedicated Communication subsystem. Each subsystem is a specific process, and the communication is done by shared memory, using the Boost library.**

I. INTRODUCTION

The Latin American Robot Competition, or LARC, is an annual competition that has a variety of categories, including the IEEE Very Small Size Soccer (VSSS) in which the robot soccer team UTBots compete since 2017.

Due to the COVID-19 pandemic, the competition was held with a simulator in 2020 and 2021. In that case, there were no related advances in the hardware, and the electronic structure is still the same as the latest face-to-face edition (2019).

During the pandemic years, only simulations were developed, and a new control system algorithm, based on the emergent Notification Oriented Paradigm (NOP), was developed and some improvements in performance were proposed and evaluated.

The remaining of this document gives a brief description of all the developments of the team, including the omni-directional robots, the novel NOP control system and the dedicated vision sub-system. All these systems are integrated for the first time in a complete functional team of real robots.

II. MATERIALS AND METHODS

The VSSS category involves a team of computer-controlled robots competing. However, it should be noted that human interference during the match is prohibited, except when authorized by the referee to correct defects in any robot that makes up the team.

The approach adopted by the team was to use a process that follows the play through images obtained by a camera positioned over the field where the match takes place. Through these images, and colored markers placed on the robots, and the color of the ball, it is possible to estimate the position of each element within the field, that is, the position of all robots of each team as well as the position of the ball.

The positions of the elements, obtained by the computer vision sub-system, are sent to another process, whose objective is to control the game's strategy. In other words, this program has the function of planning and controlling the execution of robot movements during the game. The behaviors that the team must have, such as finding the ball, attempting a goal or frustrating an opponent's goal, are contained in it. Finally, the information generated by the process responsible for the strategy must somehow be sent to robots in the field. For this, another process is used, responsible for receiving these data, adapting and then sending them to the robots through a wireless communication module. The robots also have a corresponding module, which makes them capable of receiving commands that must be executed in the field. These devices have a program capable of capturing the received direction

[*1]E. K. P. Veiga is a Computer Engineering Student at UTFPR-Federal University of Technology - Paraná - Brazil - Campus Curitiba. eveiga@alunos.utfpr.edu.br

[†2]V. H. A. Cardoso is a Information Systems Student at UTFPR. vcardoso@alunos.utfpr.edu.br

[‡3]A. S. de Oliveira is with Academic Department of Electronic (DAELN) da UTFPR, and with CPGEI-Graduate Program in Electrical Engineering and Industrial Informatics, at UTFPR. andreoliveira@utfpr.edu.br

[§4]J. M. Simao is with DAINF - Academic Department of Informatics, and with CPGEI-Graduate Program in Electrical Engineering and Industrial Informatics, at UTFPR. jeansimao@utfpr.edu.br

[¶5]J. A. Fabro is with DAINF, and with PPGCA-Graduate Program in Applied Computing, at UTFPR. He is also the leader of the Laboratory of Embedded Systems and Robotics at UTFPR (LASER-http://laser.dainf.ct.utfpr.edu.br). fabro@utfpr.edu.br

data and transforming them into movement. In the next subsections, the key components of the team are described in more detail.

A. Artificial Intelligence

The strategy to be followed by the execution robot is executed inside a process, and contains since the basic instructions, such as starting tasks for each robot, to more complex ones, such as collision detection.

Basic strategies are developed with simple parameters such as the robot's position and the ball's position. When developing more complex strategies, the position of the ball, the position of each player as well as the position of each opponent, and even the score and the time remaining in the match can be used.

The present algorithm implemented by the team is using an emerging programming paradigm, which is the Notification Oriented Paradigm (NOP), described in detail in the next subsection. This new paradigm allows the development of more optimized and less coupled code, thus facilitating adaptation to future changes.

In order to assess the robots' field performance level, several "attack versus defense" simulations were carried out, where it is possible to verify the strengths and weaknesses of both attack and defense. These simulations can be entire matches or specific plays to see how robots behave. In fact, many games are against the team's old algorithms, so that it's possible to evaluate performance growth in a practical way.

B. NOP and decision-making

NOP was proposed as a new paradigm in order to provide a new logical model that intends to contribute to higher productivity and quality in the development and execution of automated processes. This paradigm aims to promote a faster system execution performance and greater construction facility of complex systems, especially parallel and distributed systems.

Although NOP inherits some IP (Imperative) and DP (Declarative) paradigms' concepts, it also has a new way of structuring and executing the logic of computational programs, which would justify its classification of a paradigm [1]. Unlike the behavior of IP programs, where the logic is totally dependent on the execution sequence, or Rule Based Systems, in which the execution sequence is abstracted and dependent on the monolithic inference mechanism, NOP allows the expression of software dynamics and its cause-and-effect logic by means of notifications, which may be executed in parallel.

The process of decision-making with NOP is rather different from common paradigms. To briefly explain it, in spite of what happens in IP, including object-oriented programming, in which the loop of iteration is explicitly created through commands such as while and for, all decisions are performed by rules, which are only notified when an observed variable has its value changed. The following pseudo-code shows the basic structure of NOP.

```
fbe
 attributes
   ...
 end_attributes
 methods
   ...
 end_methods
end_fbe

inst
  ...
end_inst

strategy
  ...
end_strategy

rule
 condition
   ...
 end_condition
 action
   ...
 end_action
end_rule
```

Figure 1: Generic Example Code of NOPL language

C. Robot Structure

The robot structure is responsible for supporting the various components of the robot, for protecting the integrity of the electronic components in possible collisions during the match and for driving the ball.

During the matches in 2019, it was noticed that wearing a uniform to protect only the electronic modules, leaving the wheels exposed, left the robot with a very high center of mass in the Z axis, leaving it unbalanced. Adding to the fact that the wheel built by the team were very adherent, when a robot arrived at one of the sides of the field, or an opposing robot collided, it began to "climb", falling. That's why a uniform that goes from the top to the bottom of the robot was implemented. With this, the robot will no longer try to climb the wall, the ball will be driven by the uniform instead of the wheel, and the center of mass is lowered, thus providing greater control in relation to driving the ball and stability to the robot.

SolidWorks 3D modeling software was used, making it easier to model complex materials, such as screws. In Figure 2 the current modeling of the robot can be seen. The LASER has a 3D printer, where the modeled parts were printed, making it possible to carry out tests and corrections. In Figure 3 is presented the corresponding result of the printed and assembled robot.

Due to the fact that the robot uses 4 wheels, there is no space to place H-bridges on the structure. That's why the motors used in the robot are of the servo type, modified to allow continuous rotation. Unfortunately, the accuracy of the motor is lost in the modification process, making the control system open-loop. Tests were carried out with factory-modified engines, but the accuracy gain only increased by about 10%. However, it is interesting to use these motors, due to the time gain by not having to modify them.

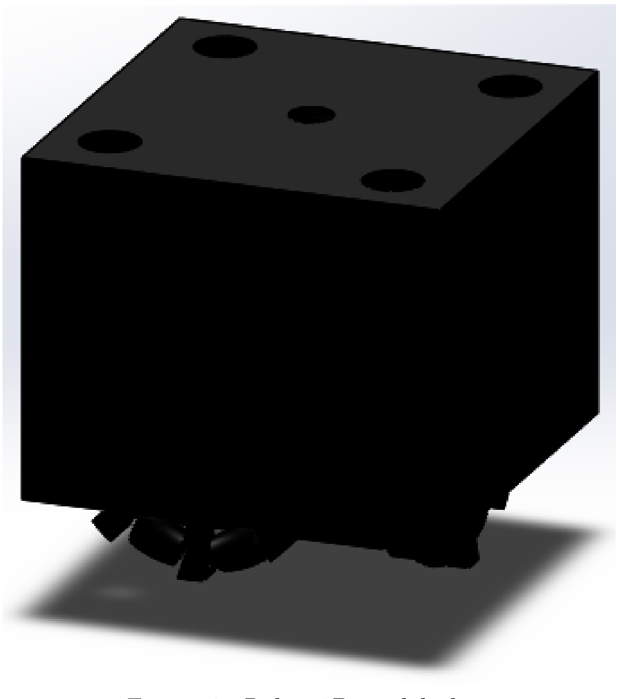
Figure 2: Robot 3D model photo

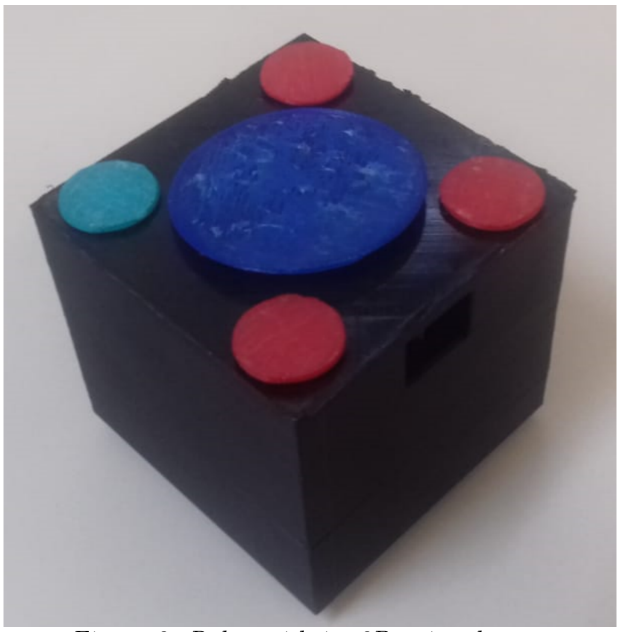
Figure 3: Robot with its 3D printed cover

D. Motherboard and Electric Circuit

An Arduino NANO is being used for capturing, interpreting data and controlling the motors, and communication is done through the radio module NRF24L01. Servomotors have an internal control system, so just send a PWM (Pulse Width Modulation) signal to them. The electrical circuit then serves only for powering the motors and Arduino, and also for transmitting the PWM signal to the motors. Because this circuit has a high complexity in terms of wires, a printed circuit is in use for power and communication between the Arduino, the radio and the motors.

E. Omnidirectional Wheels

Most common robots with two wheels can move in a certain direction only if the robot rotates so that it acquires the proper angular inclination to align itself linearly, that is, the robot moves only in directions parallel to its wheels. On the other hand, omnidirectional robots are robots capable of moving without the mandatory rotation, on their own axis, of direction, thus being able to move in any direction in the horizontal plane with agility because it combines linear and angular movement, thus reducing the number of maneuvers necessary to reach a certain point of interest. Only the UTBots team used this type of movement in LARC competitions in 2017, 2018 and 2019.

For this omnidirectional movement, special wheels are needed. Such a device is called "Omnidirectional Wheel", which is composed of a main wheel that is responsible for providing torque for movement in the normal plane to the motor axis. There are also smaller "casters", located on the periphery of the main wheel, and these secondary wheels have free rotation and an axis of rotation orthogonal to the axis of rotation of the main wheel 4.

Like the other structural parts of the robot, the wheels were modeled by the team and printed. However, the way in which the inner axle of the smaller wheels was chosen proved to be inefficient during the competition, causing the robot to release these wheels during the games. As a result, this year modifications were made to increase its strength and integrity. Another question investigated was the friction between the smaller wheels and the main one, making it difficult to move. Tests are being carried out to find out the best shape and coating for the smaller wheels, in order to have the best grip on the ground and little friction with the main wheel. Figure 4 presents the wheel used by the team.

F. Changes in Internal Control

Currently, robots operate in an open-loop control system, that is, there is no comparison of the real position of the robot with the desired theoretical position, since there is no sensor that gives feedback to the system, since as they are used 4 motors and 4 wheels for each robot the team lacks available space in the mechanical structure. To change this, the implementation of an accelerometer and a gyroscope in each robot is being studied, with the aim of providing feedback to the control and, consequently, reducing

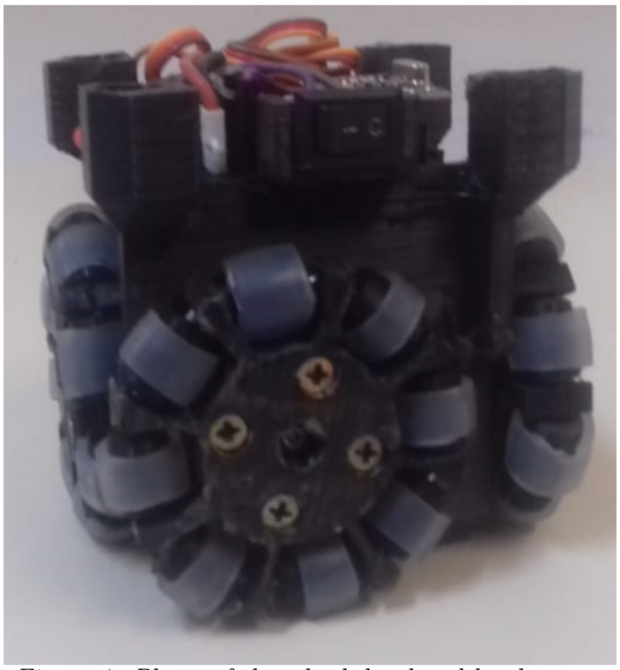
Figure 4: Photo of the wheel developed by the team

the error in the movements of the robots.

One of the problems that has already been noticed is that the robot “shakes” a lot during movement (a lot of change in the Z axis), causing variation in the sensor reading. A suitable filter is being developed so that the sensor can be read in order to improve the control accuracy.

G. Batteries

Two Lithium-Polymer (LiPo) batteries with a capacity of 850 mAh with an electrical voltage of 3.7v are used in series. The main choice criterion for these was the volume, which should be the smallest possible, and within this, the one with the greatest amount of charge was chosen. These batteries have a size of 3.5 x 5 centimeters, a depth of less than 1 centimeter each, and have an autonomy of about 1 hour with the robot moving at maximum speed.

H. Control System

Control system is the generic name for all the processing done in the hardware auxiliary to the robot team up to the point of data transmission to the embedded system inside each robot by wireless communication. This process is done in 3 modules: Vision, Intelligence and Communication System.

Due to the fact that in previous competitions there was no intuitive notion of what was really happening during the game, the team developed a macro system in C++, with functions for managing shared memory using the *Boost inter process communication library* [9], defining the shared memory usage protocol for each module and with a graphical interface to visualize the behavior of the modules during the game, thus allowing an intuitive interaction with the modules themselves. With this it is possible to know when the robots are not doing what was asked, or if the intelligence is making erroneous interpretations about the commands it should interpret. This graphical interface was made using the Qt[10] library.

A manual control was also developed in the main interface. This function is useful for Debug, or for quick tests, as it can be used both with and without image recognition, assuming different behavior in each case. A library was developed to facilitate the integration between the modules, using the PkgConfig [11] software for its configuration.

I. Vision System

The vision module involves all the processing from capturing the camera image to detecting the robot and ball positions on the field. The Logitech C920 [1] camera is being used to capture the image, which has FULL HD resolution and is capable of providing 30 frames per second.

Each robot must have an identifier, which is an area on its surface with a certain color, with a color assigned to the team, so you can identify your team and the opposing team by that color. In addition, there are other identifiers to be able to determine the angle that the robot is positioned in relation to the field. As it is not possible to know in advance which pattern is used by the opposing team, these other identifiers are only useful on friendly robots. To identify this robot angle, the geometric arrangement of the identifiers and their colors must be such that there is a reference to calculate it and it is not ambiguous.

The vision system developed uses the OpenCV [12] computer vision library. To identify the objects in the field, the color tracking method was used, which consists of using filters to choose which *pixels* of the image are in the region of a certain color, and then using an OpenCV algorithm that finds contours in a binary image, so each contour represents an object with the color filtered. After knowing the location of all colored objects in the image, an algorithm is used that decides what each entity in the field is, that is, the ball, opposing and allied robots, and even for allied robots it is calculated the angle as described above. Thus, this information is written in shared memory along with the sample time so that the intelligence can make decisions about the movement of allied robots. This system also contains a graphic interface developed with QT to calibrate the colors used during the game.

Due to CPU matrix processing limitations, we had a delay of about 1 second between acquiring the image and sending the information to the intelligence, due to the fact that we were only at 10 frames per second. That’s why we decided to do some GPU tests, using

[1]C920 HD PRO WEBCAM - https://www.logitech.com/pt-br/product/hd-pro-webcam-c920

the CUDA [13] library and the functions for GPUs available in OpenCV. However, there are no ready-made localization functions that are fast like the algorithm to find contours, in addition to not having some other functions that do a lot of filtering in the image, being necessary to use many more operations to achieve the same result. With that, even with the use of the GPU, the result was practically the same. A greater gain can be obtained if the team manages to develop their own localization algorithm that is as fast as the function currently used (tests performed showed a 30% gain in terms of CPU usage, but still far below what we expected). As a result, it was decided to reduce the resolution from FULL HD to HD, obtaining with the CPU algorithms the 30 frames that the camera used can provide.

III. CONCLUSIONS

The algorithm for managing robots and functions proved to be functional and agile, as well as the applied strategy for coordination between attacker and defender. Nevertheless, it is still necessary to make improvements in the area of collision detection, which in the new paradigm still presented some divergences of functioning, and navigation of robots.

Therefore, a next objective is to integrate the system based on the notification-oriented paradigm, which has been developed in the last years, with the robot structure, described above, used in the presential editions of LARC. It is also necessary to correct errors and make improvements in the AI strategy algorithms.

ACKNOWLEDGMENT

The authors Eduardo K. P. Veiga and Victor H. A. Cardoso would like to thank the Federal University of Technology - Paraná for the support.

References

[1] Silveira, Cristiano Bertulucci. “Servo Motor: Veja como Funciona e Quais os Tipos”, Available at: https://www.citisystems.com.br/servo-motor.

[2] PK2Lab, “Alteração de Servomotor”, Available at: http://pk2lab.blogspot.com.br/2011/12/alteracao-de-servomotor.html.

[3] Arduino, “Arduino NANO”, Available at: https://store.arduino.cc/usa/arduino-nano.

[4] Nordic Semiconductor, “nRF24L01 Single Chip 2.4GHz Transceiver Product Specification”, Available at: https://www.sparkfun.com/datasheets/Components/nRF24L01_prelim_prod_spec_1_2.pdf.

[5] Arduino, “Arduino FIO”, Available at: https://store.arduino.cc/usa/arduino-fio.

[6] Digi, “Xbee/Xbee – PRO S1 802.15.4 (Legacy) RF Modules”, Available at: https://www.digi.com/resources/ documentation/digidocs/PDFs/ 90000982.pdf.

[7] Pereira do Nascimento, T. “Controle de Trajetória de Robôs Móveis Omni-direcionais: uma abordagem multivariável” Available at: http://www.ppgee.eng.ufba.br/teses/ ac64916 ca6ef3bdd2f466bffdd30af7e.pdf.

[8] John S. Palmisano, “OMNI-WHEEL ROBOT – FUZZY”, Available at: https://www.societyofrobots.com /robot_omni_wheel.shtml.

[9] Boost C++ Libraries, “Boost”, Available at: https://www.boost.org/.

[10] Qt Libraries, “Qt”, Available at: https://www.qt.io/.

[11] Nicholson, Dan. ”Guide to pkg-config”, Available at: https://people.freedesktop.org/ dbn/ pkg-config-guide.html.

[12] OpenCV Library, “OpenCV”, Available at: https://opencv.org/.

[13] NVIDIA. ”CUDA Zone”, Available at: https://developer.nvidia.com/cuda-zone.

[14] Simão, Jean M. “Paradigma Orientado a Notificações (PON)”, Available at: http://www.dainf.ct.utfpr.edu.br/ jeansimao/PON/PON.htm.

[15] ROS. ”ROS- Robotic Operating System”, Available at: https://www.ros.org/.

[16] Linhares, R.R., Simão, J.M., Stadzisz, P.C.. ”Noca - a Notification-oriented Computer Architecture”. IEEE Latin America Transactions 13(5) (2015) 1593-1604

[17] A. F. Ronszcka and C. A. Ferreira and P. C. Stadzisz and J. A. Fabro and J. M. Simão. ”Notification-Oriented Programming Language and Compiler”. In: 2017 VII Brazilian Symposium on Computing Systems Engineering (SBESC), Curitiba, Nov /2017, pp. 125-131.