Team Description Paper:

Robot rescue for the identification of victims in a Dangerous Environment for CBR - Rescue Maze - Midas Touch^-1

Lucas Nogueira Roberto, Davi Pereira Alves Colares, Marcus Vinícius; LARC 2019

***Abstract*— This report has the purpose of specifying the strategies used by the Midas Touch ^ -1 team and details about the robot construction, as well as programming details. The robot was designed with OnShape and SolidWorks software, outlined on MDF boards, designed to be replaced by a posteriori acrylic structures, spaced apart by metal hexagonal spacers. The boards were cut from the drawings made on Onshape, on these are present the Arduino Nano, Arduino MEGA and Raspberry Pi B + which were programmed respectively in C ++ in Arduino Software (IDE) and in Python. The Arduino MEGA is connected to motors, infrared temperature, ultrasonic and color sensors, and to the motors. Arduino Nano is connected to servomotors. Raspberry Pi is connected to the Arduinos via serial communication, to an LCD and to a Logitech WebCam. To map the maze we used Treumax and to solve it we used an algorithm similar to the search in depth in conjunction with a modification of the algorithm D * called D * Lite.**

I. INTRODUCTION

In recent years, the field of robotics has been an essential piece on the task of developing safer and more efficient ways of acting on environments where the lives of people are at risk. In general, accidents generate lands that are vulnerable, making it difficult to rescue the victims of a disaster. For example, regions with high indices of radiation where humans couldn't approach, or fires, where the heat and smoke stop the firemen of rescue people. The purpose of this paper is to show and explain the creation of the Midas Touch ^ -1 team robot to solve the challenge proposed by the Latin American Competition of Robotics.

II. MAIN GOAL

The main goal of our team in respect of Rescue Maze was to create a robot capable of run through the maze, leave the rescue kits to the maximum number of victims possible in a way to minimize the time used. To identify the characteristics of his surroundings our robot utilizes not only fixed sensors to improve our robot performance but a camera using image processing to identify letters through the course of the challenge and optimized algorithms to solve the maze.

III. TEAM

The team was divided into functions, to divide the work and organize the project development, the functions are:

1. Lucas Nogueira: Responsible for the implementation of the algorithms to solve the labyrinth, robot movements, and webcam configuration in order to identify the visual victims. (*contact: lucasnogueira064@gmail.com*)

2. Davi Colares: Responsible for the construction, for the mechanical parts, for the design, and for modeling the CAD of our Robot. (*contact:davicolarespersonal@gmail.com*)

3. Marcus Vinícius: Responsible for the electronic system and for the logic involved in programming the sensors (*contact: marcus.vmsc@gmail.com*)

IV. CHALLENGE

The maze is limited by walls, which can be linear (Part of the arena) or floating (In the middle of the arena). The arena represents a hostile environment, which is separated into two pavements, one larger than the other. They are located at different heights and are connected by a ramp. The floor can be covered by white, silver, and black tiles, symbolizing, respectively, neutral areas, safe points, and danger zones, the robot needs to pass through the maze, leave the rescue kits to all victims and go back to the start line the fastest possible.

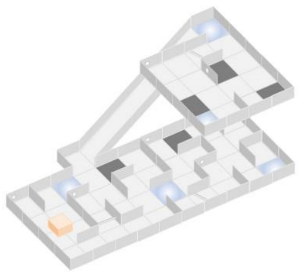


Image 1: Arena

A. Victims

The victims can be categorized into two groups:

1. *Heated victims:*
They are represented by heat fonts, located at approximately 7 cm from the ground. Each shall receive one rescue kit.

2. *Visual Victims:*
This type is represented by letters printed on black. The letter itself 4cm high and its center is located at approximately 7cm from the ground. There are three variations of visual victims (harmed, stable, and unharmed) and each shall receive a specific number of rescue kits: i. Harmed (letter "H") should receive 2 rescue kits; ii. Stable (letter "S") should receive only 1 rescue kit; iii. Unharmed (letter "U") receives no kit

Image 2: Letters

B. Arena

In order to be able to train our robot and have the best possible preparation to the competition our school provided a special arena to our and future teams of robotics

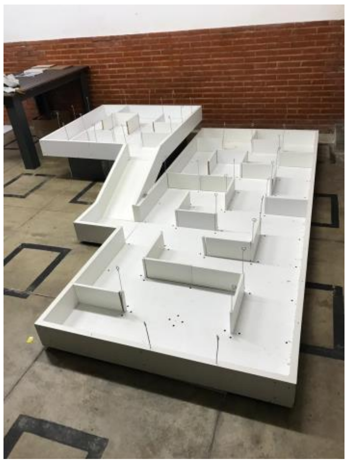


Image 3: Arena in real life

V. ROBOT

Here follows the essential topics that compose our robot, especially designed to solve the Rescue Maze task.

A. Materials

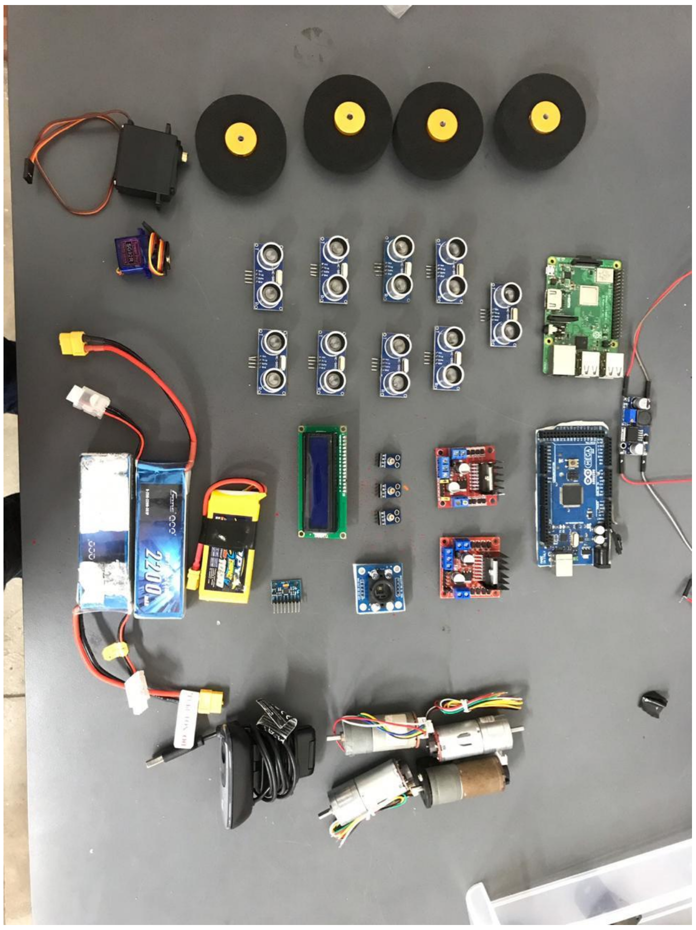
Boards:

Raspberry Pi 3 B (Main):
LCD Display
Webcam

Arduino Mega (Secondary):
1x Color Sensor
1x Gyroscope
3x Temperature Sensors
9x Ultrasonic

Arduino Nano (Tertiary):
4x Motors
1x Mini Servo (Blue)
1x Medium Servo (Black)

Image 4: Material



B. Mechanics

Initially, the CAD was developed to help in design a robot that had the ability to move in any direction and develop the same functions on either side. Our robot is created so that it can explore the maze using 9 ultrasonic sensors, 1 of which is underneath and 8 overhead, giving a complete view of the surroundings of the robot.

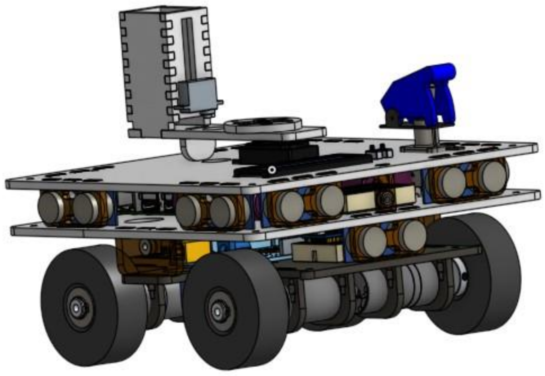


Image 5: Robot

In addition, to detect heat victims the robot will use 3 temperature sensors and to detect visual victims it will use a webcam and a servo to move the camera 180 degrees.

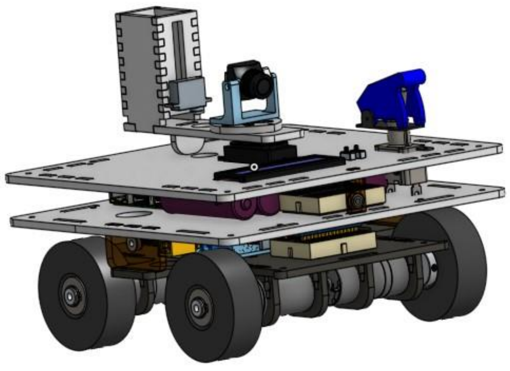
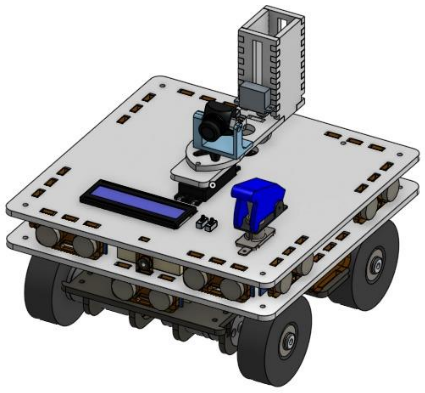


Image 6: Robot

For the movement were used expanded neoprene wheels, together with 210 RPM encoder motors, in this way the robot can orient itself in the maze and balance the speed and torque required. In the first floor of the robot, there will be 4 motors responsible for the movement, also having encoders that facilitate in the control of the robot through the maze, in addition, one of the battery will be between them and just above the Arduino Mega. In this floor, we will also have an H-Bridge to control the motors that will be connected in pairs to these. In this area will also be two of the three temperature sensors and one of the 9 ultrasonic. Just up on the second floor will be the Raspberry which will control the camera, the servos, and the LCD at the top, in addition to processing the program responsible for solving the maze. The last layer will be occupied by the Webcam and the Rescue Kits dispenser, both on a servo that will allow greater freedom of direction for dispensing kits and for recognizing visual victims.



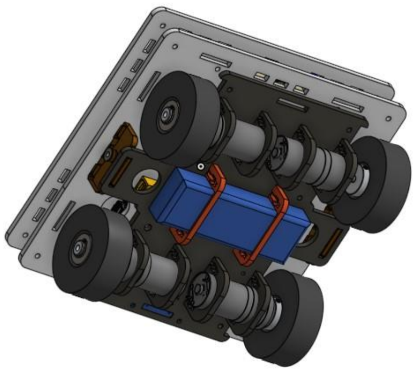


Image 7 and 8: Robot

C. Rescue Kits

The launcher is composed of a box with adjacent sides hollowed to the exit of the rescue kit, which will be stacked in the box, for release, the servo will move a piece on one of the leaked sides making it come out on the other side and preventing the next drop to fall out.

In Onshape, also, kits were designed to be used for this year, which will be printed in PLA, respecting the rules and metrics of the competition

Image 9: Rescue Kit

D. Hardware

For the exploration of the robot we'll use 9 ultrasonic HC-05 sensors distributed through its structure, in this way we can measure the distances of the robot in relation to the walls and obstacles in his surrounds, the data will be read and preprocessed by an Arduino Mega, then the Raspberry Pi that will finalize the process applying the D * Lite algorithm. In addition, we will also have a gyroscope sensor MPU 6050, which will help to keep the robot centralized in the maze (parallel to the walls), and motors with encoder, that will allow us to obtain a greater precision on the movement. For the detection of different soil colors, we will use a TCS230 color sensor, and for the victims, we are using a Webcam Logitech, and 3 MIX90615 IR temperature sensors. The mechanical part, motors - encoder, bridge, and sensors will be controlled by Arduino Nano as well as MEGA, which will be responsible for the sensors, these will be connected to the Raspberry Pi via USB, through Serial communication. Finally, the Raspberry will be responsible for the processing of the images received by the camera and the LCD that will help us to follow the routing of the processes. The servos will be used to rotate the camera and dispense the rescue kits.

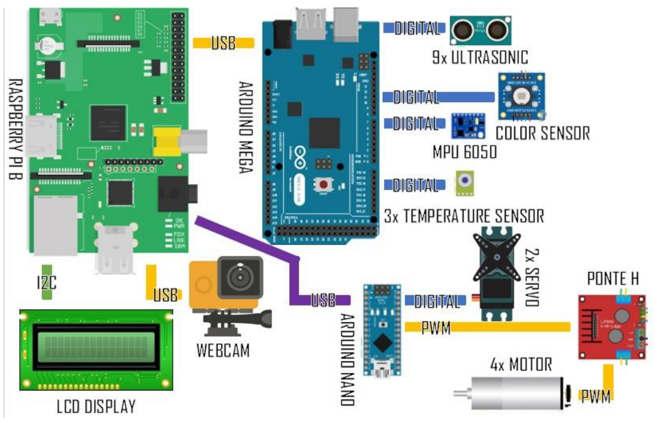


Image 10: Hardware

E. Batteries

The power of the robot is made by two Turnigy Lipo Battery of 2200mAh batteries and a 1000mAh battery: Rhino, connected in parallel, and connected to a general power switch. This configuration was decided by its greater efficiency and its greater duration, which helps in the time of its exchange during the competition

Image 11: Batteries

F. Statistical Noise

To make the robot obtain a better performance, we use in all readings of values obtained by the sensors a filtering method, known as the Kalman filter, which is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, producing estimates of variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe, allowing the robot to approximate the measurements to numbers similar to actual values

G. Programming

Our code is created in C ++ and Python, structured in classes, using the object-oriented features. The code is separated into three distinct layers: two lower level, with direct interaction with the hardware, one referring to the Arduino Mega, commanding the ultrasonic sensors, temperature and color sensors and the gyroscope to maintain the stability of the robot, another referring to the Arduino Nano that controls the servomotors of the release of the rescue kits, these two layers offer aid to the code of the second layer, the one of higher level, focused in researching and mapping the arena and identification of the visual victims.

• Layers 1 and 2: Layers of action

The first and second layers serve only to receive and execute commands from the above layer, communicating between the physical components of the robot (sensors, motors, etc.) and the logic itself, both of which are made entirely in C ++ in the Arduino IDE.

• Layer 3: Layer of Logic and Strategy

The algorithm is fundamentally composed of a depth search merged with a D * Lite. The search in-depth scans graphs, always trying to reach points more "distant" and returning to previously visited points in cases of non-existent points, connected by only one edge. The graph in the project represents the map of the maze, the graph vertices, and the connections between tiles the edges, when the connection is not possible (there is a wall) the weight given to the vertex and associated with infinity, preventing from reaching that point, however, when there is no wall, it receives the unit weight. The D * Lite is an optimum heuristic based path algorithm, widely used for its high performance, is considered more efficient than many others, such as A *. During execution, a map of the environment is made. With this map is possible to navigate to any point on the track of the most efficient way possible using D * Lite. The program initially uses the in-depth search algorithm until arriving at a point where there is no exit, where all the paths around it have already been explored or are blocked by obstacles, at this points the D * Lite is used to finding the shortest path between the current cell and the closest cell to be explored. Obstacles are detected by abrupt differences in the reading of the distance sensors and marked as a subsequence region, whenever the robot enters in a new cell, using its sensors, it detects possible walls, victims, but, mainly, adjusts to keep its sides parallel the walls. Detection is done in a second processing line, which verifies the temperature detected and the image of the camera. Then the program goes on, exploring the maze using the methods described and delivering the rescue kits when some victim is found. When the map is fully exploited, the robot can assume that all the victims were found and saved, thus, the D * Lite algorithm is used by last time, with the objective of returning to the starting.

To identify the visual victims we used a Webcam Logitech with an algorithm of recognition of letters, utilizing TensorFlow and machine learning

It is crucial that the robot is aligned at all times with the walls so that, with his movements, he does not get lost in the maze. With this fact in mind, we implemented PID controllers (Proportional, Integral, Derivative), which, by providing a continuous variation of the output within a control loop feedback to control accurately the process and the data obtained by our gyroscope sensor this removes the oscillation and increases the efficiency of our robot.

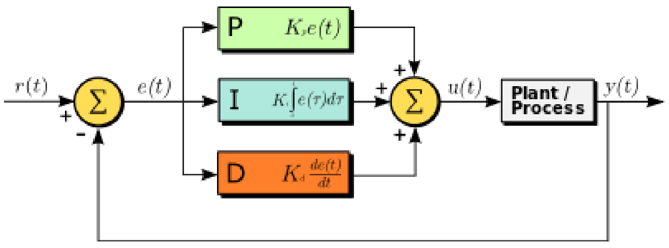


Image 12: PID

H. Computational Simulation

To test the robot's programming and verify its efficiency, a Python simulator was created that receives and includes the robot code in a random environment. Then you can see an example, which can also be found in the link below: https://www.youtube.com/watch?v=jB3hP3f2PwQ&feature=youtu.be

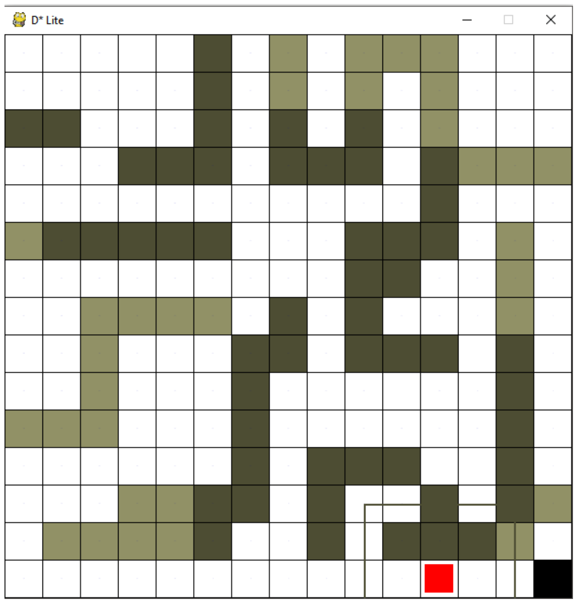


Image 13: Simulation

VI. Conclusion

In conclusion, it's possible to evidence the evolution of the development of our robot, using CAD software like Onshape and Solidworks it was possible to have total control of the displacement of the mechanical and electrical parts on the robot and the creation of the project in an online platform played an important role, because we didn't have to restrict this work to only one computer. Was added to the project sensitive and fast engines, which have a lower error and higher performance, in addition to a triggering system simple and fast. In respect to the sensors, we dedicated a good amount of time calibrating them to achieve an almost perfect prosecution. With the use of a simulator, we could ensure the accuracy of our functional algorithm, thanks to its creation it was possible to minimize pre-existing programming errors. In the end, with the increase in performance, it is possible in some years to be able to create robots with real application in the rescue of victims.

Acknowledgment

We thank the support of the Colegio Ari de Sá Cavalcante and the coordinators Rafael Brito Arruda and Kessly Araújo, to all our teachers and staff members of our school who helped us on this journey, an honorable mention for the professional responsible for our team, Josué Caleb Gurgel Saraiva , as well as to our parents and friends who contributed to this project, especially Luís Eduardo de Medeiros.

References

[1] Arduino Software (IDE). https://www.arduino.cc/en/Main/Software H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.

[2] S. Koenig. (2002). Fast Replanning for Navigation in UnknownTerrain"[PDF].http//pub1.willowgarage.com/konolige/cs225b/dlitetro05.pdf / E. H. Miller, "A note on reflector arrays (Periodical style—Accepted for publication)," *IEEE Trans. Antennas Propagation.*, to be published.

[3] H. Reddy. (2013, December 13). "PATH FINDING - C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995. Dijkstra's and A* Algorithm's"[PDF].http//cs.indstate.edu/hgopireddy/algor.pdf

[4] Advanced 1. Incremental Path Planning. https://www.youtube.com/watch?v=_4u9W1xOuts

[5] Onshape Product Design Platform. https://cad.onshape.com/help/PDF/Onshape.pdf

[6] R. Bonilla et al., Robocup junior rescue maze – rules 2015, 2015. [Online]. Available: http://rcj.robocup.org/rcj2015/rescue maze 2015.pdf

[7] Robocup Junior Rules TDP rules. http://www.cbrobotica.org/?page_id=27

[8] Instructions on installing and using the Tesseract library for Optical Character Recognition (OCR) with Python and OpenCV: www.pyimagesearch.com/