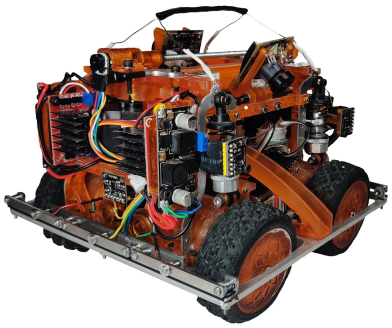
RoboCupJunior Rescue 2021

Team Description Paper

Salbeghi

Abstract

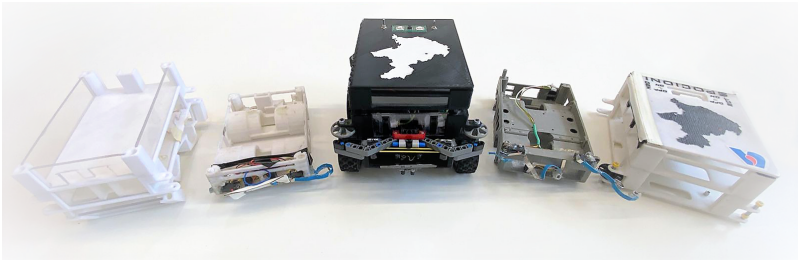
- Our robot has been designed and built with the aim not only to carry out the tests, but to do that in the best possible way, optimizing every movement performed by the robot and overcoming any obstacles in the best possible way. The machine has been completely designed in CAD to make the best use of all volumes. A suspension system and dual-engine kit release system have been implemented. The robot is designed to be built on a laser cut aluminum platform. Two cameras are mounted, one on each side that through OpenCV allow us to recognize visual victims. While exploring the maze the robot maps the maze and through the Dijkstra algorithm reaches the closest square by taking the shortest path.



1. Introduction

a. Team

- Our team is made up of three people, we have divided the tasks according to our passions and skills developed over time.
    a. *Pogetta Matteo* (Captain) : Software - Algorithms - Mapping - 3D Design
    b. *Berlato Matteo* : Software - Sensors - Test
    c. *Grendene Marco* : Hardware - Assembly - Test
- This is the third year we have participated in this competition, always in the Rescue Maze category. With our first robot we took part in the national championships, qualifying for the European Championships in Hannover and winning the seventh place (in 2019).
- This first experience allowed us to acquire the necessary experience for the creation of the new robot, as well as to experiment with numerous innovative hardware and software solutions.



2. Project Planning

a. Overall Project Plan

- The main purpose for which we participate in the competition is to develop our technical-IT skills, but above all the transversal skills, at an interpersonal level. Furthermore, we wanted to be able to build a machine that could not only compete in these competitions, but be somehow an inspiration for rescue applications in real dangerous environments where a robot can save lives without endangering others.
- Thanks to the first experience, we learned which limits and difficulties the robot encounters due to its structure and programming; therefore we have developed this robot trying to solve the problems of the previous one (at a structural level), also implementing new solutions.
  These solutions derive both from personal experiments and from the experiences lived by the previous teams of our school, but also from those at an international level.
- With this robot, we have tried to introduce innovations or improve functions already seen. We wanted to create a rigid structure that surrounds the robot; to do this we laser-cut a 4mm aluminum platform onto which all the other structures are hooked. In addition, we have implemented a suspension system, making the movement of each motor independent and hooking it to the main structure via a 60mm suspension and two bearings with inner diameter of 8 mm and outer diameter of 16mm. In addition to this, we have created limit switches in direct contact between its two parts, the external laser-cut part closes the circuit by touching the base brought to ground. Thanks to the use of resin as printing material (ABS like) and to the complete design in a 3D CAD virtual environment (Autodesk) we have made the robot modular. The complete design has allowed us to create a double exit rescue kit release: kits are conveyed by two motors (DC and 6V servo) and then ejected by two ramps. On the software side, we have implemented various algorithms, including two PID negative feedback controls, mapping via Dijkstra and movement via encoder and gyroscope.

- The robot was developed entirely to compete in this category, also relying on the experience of other past groups. Its size, shape, materials, and even the software have been studied according to international regulations and the competition field.

b. Integration Plan

- The robot has been entirely designed and assembled in a virtual environment; this has allowed us to create an assembly in which all the parts interact with each other in the best possible way. In addition, it should not be forgotten that at the time of the development we were already aware of the hardware and software needs, thanks to the past experiences.

c. Testing

- To test the robot's performance and reliability, we first tested each part individually to understand its performance and then created the overall assembly. Once all the parts had worked individually, we

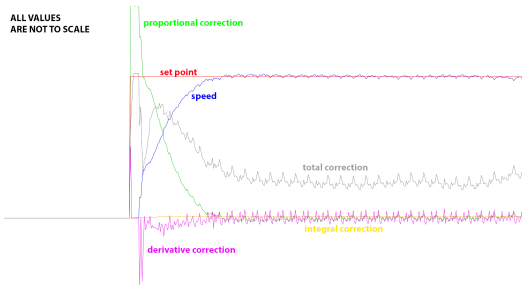
began to merge them, to gradually verify the performance. Once the robot was completed with all functions, we began to test the whole system in the competition-like fields, also adding unexpected obstacles and difficulties.

- For the mapping algorithms, we have also created a simulator to be able to analyze the performance of the various programs. The simulator takes in a map of any size and shows us the various steps and in the end how effective the exploration was.
- The tests of the hardware part were performed both in a virtual 3D environment and in real conditions through tests carried out to test the limits of the robot (mazes with many speed bumps, or with victims, obstacles in unusual positions, etc.).
  As for the software tests, in particular to verify the functioning of the mapping system, as mentioned before, we created a simulator (using the PyGame library) that would execute the code loaded in the robot SD-cards to see, even before bringing them into reality, if the added features could be valid.

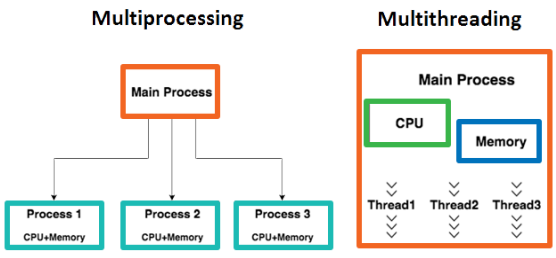
3. Software

a. General software architecture

- The software of our robot is mainly divided into two parts, the one relating to the PID control of the motors carried out in the ESP32 board and the remaining part carried out in the Raspberry PI board.
  a. *ESP32*: programmed in C with its ESP-IDF framework, it is used for motor control and sensor readings with analog output. It communicates via serial to the raspberry pi. This board is connected to a 74HC595 shift register for the expansion of the digital output port; in particular the shift register outputs control the digital signals of the drivers for the 4 main motors. To make the robot as controlled as possible we needed to implement various controls, including a speed control of the traction motors. To do this we have mounted an encoder for speed measurement on all motors and implemented a PID control. The four motors are controlled by a Proportional-Integral-Derivative (PID) software controller, which keeps the desired speed constant. It is performed on the ESP32 board, the 4 motors are independent, i.e. a PID control is calculated for every single motor. The PID is a negative feedback control that acts as a corrector network; the process acquires the instantaneous speed of the motor at the input and compares it with the reference value; it obtains the error as a subtraction of the two and proceeds to calculate the three corrections, proportional, derivative and integral.



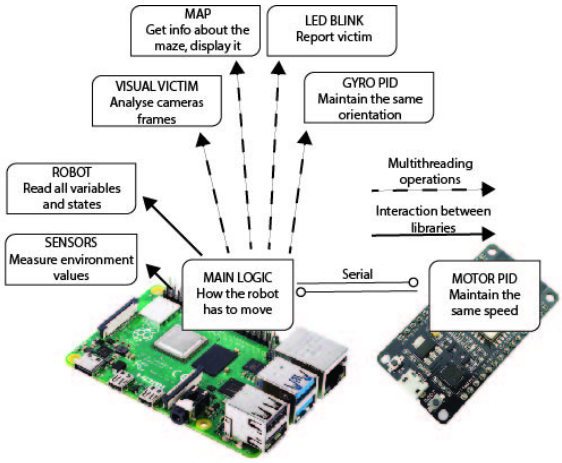
  b. *SERIAL COMMUNICATION*: Communication between Esp32 and Raspberry Pi takes place via an appropriately managed RS-232 serial bus, it is a 1,000,000 baud protocol, 8 data bits for transmission, 1 parity control bit and 1 bit stop. The communication is bidirectional, but only the Raspberry Pi makes the requests to the ESP, and the ESP responds with the desired values. The data protocol has been structured exactly on the desired data, it allows little expansion but reduces the bytes sent. The ESP32 is programmed in such a way that if it receives nothing for more than a second it blocks the motors by setting their speed to 0. The communication is also completed by a digital signal shared between the two boards: this is brought high by the ESP32 when ready (if it is turned on and started correctly), so the Raspberry Pi can know when the ESP is turned off, and also when it has not started



correctly, so that it can be reset via another digital signal connected on the reset pin of the ESP.

c. *RASPBERRY PI*: The code structure of the Raspberry Pi is organized according to different files, containing different classes (sensors, map, movements, image interpretation and main logic). Moreover, the Raspberry Pi has to run numerous programs in parallel, including the main logic, two image recognitions and the map display. We also decided to divide the programs into several parallel processes: they use multiprocessing, which, at the cost of isolating the various data structures of the various sections, allows greater usability of the raspberry resources, including CPU and Ram. This is because processes are considered to be completely separate, as opposed to multithreading which executes multiple pieces of code in "parallel", but these are considered to be the same process. The communication between the various processes takes place through the use of special queues that create bridges between one process and another. Various multithreading is then used within the various general sections, in particular the logic part consists of multithreading for the PID control of the gyroscope and one for the activation of the victim signaling LEDs, each code for the recognition of the cameras makes use of a thread for the continuous taking of the frames from the camera.

d. *FINITE STATE LOGIC*: The main logic of the robot has been programmed following a finite state logic that allows to fragment the code into many sections and make them collaborate. These small sections of code can be executed one after the other continuously so that multiple logical lines can be followed at the same time. This type of programming allows us not to have to repeat many parts of the code, for example, communication with the ESP card occurs once per cycle, this allows us to be sure of the operation of the ESP itself, that the motors have the exact desired speed, and to have the distance and the photoresistor value always updated. This also applies to the laser and temperature sensors, to the multiprocessing recognition of the visual victims, to have the map display always updated. The data structure that has been created for this logic is divided into numerous classes, in which all the variables corresponding to the assigned function are saved. The command of all the code sections is managed through a very complex logic that makes use of many lists and levels.

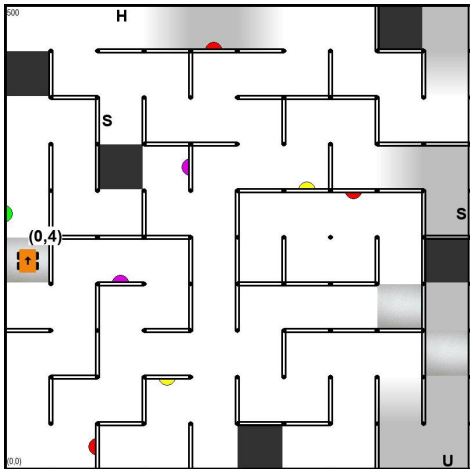


e. *MAP AND ORIENTATION*: The creation of the map requires a particular logic of orientation, the aim is to be able to make the robot move within the map exactly as it moves inside the maze. The task is quite complex given the very nature of the labyrinth: ramps, obstacles and speed bumps are elements that make the robot movements difficult to control and detect. For this we have implemented a method to transform the distance and gyroscope data into the values of X, Y, Z (coordinates considering the maze in a Cartesian plane), G (rotation of the robot) in the map plane. For the calculation of the three Cartesian components, the displacement completed cycle by cycle is added to each cycle. This is given by the difference in distance traveled, the current one minus the one detected by the last cycle, for the trigonometric components.

f. *MOVEMENTS*: The robot movements have been designed to be functions of the main logic that can be recalled when needed; being part of a state logic they are more complex but very efficient. The programmed functions are for example straight, turn, limit switch, align,

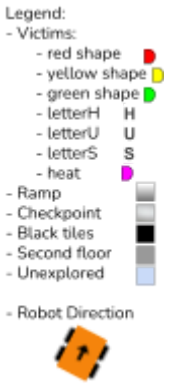
center, victims, etc. These can be activated by changing flags and setting the desired settings on specific variables, as in the case of turning the speed and distance to travel.

g. *MAPPING*: The robot must be able to autonomously explore the whole maze. To do this we had to implement a mapping algorithm, that is a process that saves the maze, as it is explored, within a specific structure of data, and calculates the best route to explore the maze. The robot cannot know the map in advance, so dynamic rules must be implemented. The basic rule used is that of the "right hand" according to which the robot explores the maze always giving precedence to its right. In practice the robot always follows the wall to its right, this allows in perfect mazes to explore all the zones and to return to the starting point. But for mazes that are not perfect, with rooms and cycles as in our case, something more is needed. We have solved this and implemented together with the right-hand rule an algorithm to find the shortest paths. The algorithm chosen is called “Dijkstra”, deriving from the theory of graphs, allows the calculation of the shortest paths in a graph with positive weights. The maze is a particular type of graph, where each square corresponds to a node and the weight of each arc is equal to 1. Given the starting square, that is the current position of the robot, the algorithm calculates the shortest path to travel to get to any other unexplored square, or the starting square. Therefore the total exploration algorithm is based on two planes: if an unexplored square is directly available next to the robot's current square, it follows the right-hand rule, otherwise it executes Dijkstra's algorithm to get to the nearest unexplored square, or if there are no more, to return to the initial box. The advantage of this solution is that it works without a "memory" of the previously performed actions, every time the robot enters a new box, if there are no unexplored boxes available next to it, it executes Dijkstra from the beginning without taking into account the previous actions.

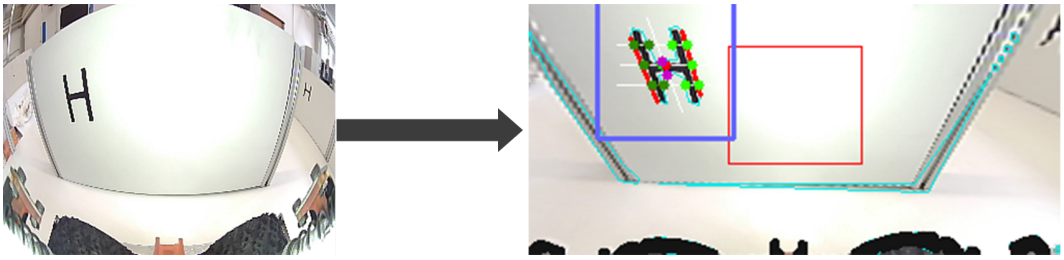


h. *SIMULATOR*: The map code was programmed and then tested on a PC and then transferred to the Raspberry Pi and integrated with the rest of the code. To simulate the operation of the robot we have created a simulator based on the PyGame library. The viewer is not only used in the simulation phase on the PC, but also in the real functioning of the robot, showing what the robot knows, where it thinks it is, the decisions it makes, all this also for a debug.

i. *CAMERA RECOGNITION*: The visual victims are detected by the robot through two cameras, one on the right and one on the left. These cameras are read by the raspberry pi with the necessary computer vision procedures. The library used is OpenCV, which allows both image capture and processing. To be able to "see" as much as possible, possibly the entire wall of the maze, to be able to detect the victims only in the center of the box and not continuously, freeing up computing power for the rest of the programs, 180 ° cameras were used. These cameras have a very high fisheye effect, which makes it very difficult to recognize the elements, since their shape varies greatly depending on the distance from the center of the lens. To solve this problem we have mapped the camera lens with the necessary procedures and developed the correction matrices. These matrices, calculated only once, are applied to each input image to 'straighten' the image and have an undistorted view. The recognition of letters and colored shapes has been experimented in several ways, however, it is always a question of binarizing the image (bringing the image to black and white),



detecting the edges, then the shape and color. Once the binarization has been performed, the image contours are identified: this is given by the fine contours function of OpenCV, which outputs a list of all vertices of the detected edges, this contour hierarchy is analyzed contour by contour. Every single closed contour is then approximated by a function contained in OpenCV, which, depending on the tolerance entered, significantly decreases the number of vertices. The controls applied to the colors mainly involve the calculation of the rectangle circumscribed to the outline, that is, which includes all the edges, and its dimensions in relation with the image ones, that is, it must not be too small or too large, it must be as square as possible be too elongated, it must not touch the top or bottom of the frame. The number of sides is not taken into account as according to the latest versions of the regulation colored visual victims can have any shape. The checks for the victims, on the other hand, are made up in a first phase of the same colored forms, and then move on to understand what letter it is. To do this, the two longer sides of the approximate contour are made explicit, which for the H and the U must be on the vertical sides of the circumscribed rectangle and have approximately the same height as it, this does not apply to the S. Another control sees the tracing of two lines respectively horizontally and vertically which is cut into 4 equal parts in a circumscribed rectangle. These guidelines are followed, and the points where the color changes from white to black are identified, their position and their number together determine what the first thing is. Recognition through the cameras is not only able to recognize the victims but also to see the black boxes, and change the behavior of the robot accordingly.



Original COLOR MASKS
Yellow Green Red Black

b. Innovative solutions

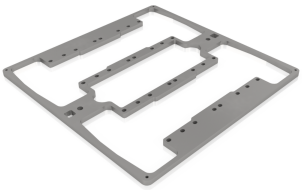
- We have experimented with numerous unconventional solutions for the code of this robot, from the creation of parallel processes to the creation of a completely state logic, the latter of which includes very onerous and complicated management.
- Also for the kit release, we have implemented a system able to always understand how many rescue kits are present in the loader and if a possible release was successful, all of this thanks to a servo motor and a DC motor with incremental encoder.
- Also for the cameras we have studied and developed the transformations for the correction of the fisheye lens. In addition to this, we are able to recognize the letters without any particular function

but only by the simple analysis of the binarized image. We also just learned the ability to recognize black boxes via cameras.

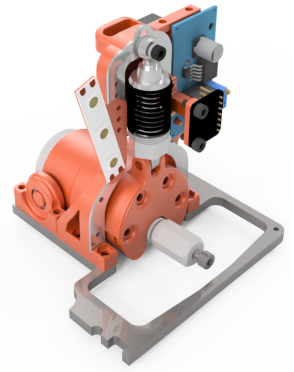
4. Hardware

- This robot was entirely designed in 3d before being built: this allowed us to experiment with various solutions and to have a vision of the robot in its entirety. Although numerous changes have been made in the course of construction, the initial project has never changed. The development in 3D has allowed us to calculate all the dimensions, the positioning of the sensors and actuators to ensure maximum use of each space, and to guarantee comfort and accessibility to the various components. The design was made with the awareness of the printing material: the resin, as we will see later, and of the printing volumes; we have created a modular structure, with many pieces but small in size. This modularity helped us to be able to modify parts even later and to add modules not considered before. As you can see from the various renderings it is not a classic robot, we decided to experiment with new solutions, such as suspensions, or the double-motor kit deployment system.

a. Mechanical Design and Manufacturing



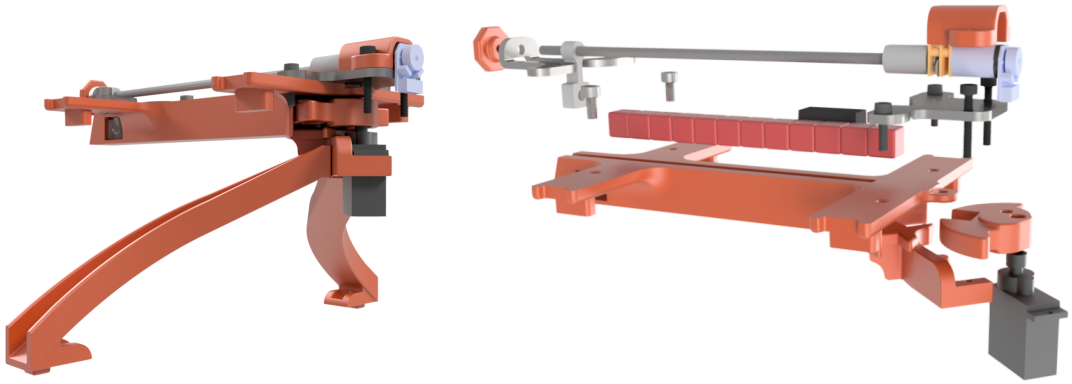
- *FLOOR*: The platform of the robot, where all the other parts are fixed, is made of a 4mm aluminum sheet, with a maximum size of 200x200 mm, designed in 3d and laser-cut by an affiliated company. This allowed us to have a smooth outer edge of the robot, in which it was difficult to get stuck, and great structural solidity. All the other components of the robot have been screwed onto the base, from the suspension supports to the limit switches.



- *SUSPENSIONS*: Suspensions are an experiment that we wanted to introduce to bring something new. They are hooked on one side to the upper end of the "arches" and on the other to the motors with resin supports that covers them almost entirely, this then goes to hook to the sheet through two mini bearings. They are useful for effectively overcoming obstacles, especially in collaboration with the PID speed control performed on each individual motor.
- *ACTUATORS*: The robot is moved by four 12V DC motors, each with a reduction gear that brings it to 111 rpm, they are operated by two L298N H-bridge drivers controlled in turn for each motor by a pair of digital signals, for the direction of travel, and a PWM signal, for speed. An incremental encoder is coupled to each motor in order to determine the exact speed. The kit release, on the other hand, is operated by a 6V DC motor with reduction gear that brings it to 750 rpm. The motor is driven by an L293D H-bridge driver always controlled by a pair of digital signals, for the direction of travel, and a PWM signal, for speed. Moreover, this motor is coupled to an incremental encoder to allow us to detect the speed, but above all the rotations for the calculation of the kits present on the loader. The movement of the kit to select the right or left chute is then driven by a TowerPro MG92B servomotor.

- *LIMIT SWITCH*: limit switches have been built by us with laser-cut and folded sheets, with springs and screws.

- *KIT DEPLOYMENT MECHANISM*: The mechanism for releasing rescue kits is composed of a central magazine that can contain up to 12 kits of 10x10x10 mm dimensions. The kits are pushed towards the deployment via a slider that advances by turning a screw without end. This screw is a threaded rod rotated by a 6V mini DC motor equipped with an incremental encoder. The drain is formed by a properly shaped selector, this drain can rotate thanks to a servo motor to the right or left so that the kit can fall inside one of the two slides. The upper release part is mechanically independent from the rest of the robot so that it can be easily removed to access the PCB and the underlying components. Mainly it consists of resin molded elements, which allows great precision, especially for the ramps that must be perfect and very smooth due to the low slope. There are also 3mm sheets for fixing the motor and the threaded rod, these also give structural support to the main element. The cursor is always made up of laser-cut 3mm sheet metal and subsequently threaded to be able to slide when the screw is rotated. The system allows a quick and simple management of the kits in fact, thanks to the encoder it is possible to calculate the number of kits inside the magazine and if any release has been completed.

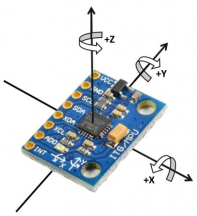


b. Electronic Design and Manufacturing

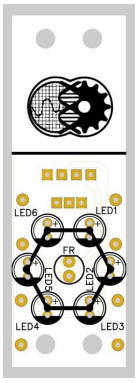
- Sensors:
  - The robot is equipped with advanced sensors, which allows us to have very precise measurements from the surrounding environment.
    - As for the orientation in space, the detection of walls or any obstacles, five *VL6180X*, laser proximity sensors with a range up to 255mm that communicate via I2C protocol to the Raspberry Pi 4 board were used; these sensors were chosen for the high degree of accuracy and speed in reading (modified as indicated in the datasheet), as well as for the fact that other IR sensors are highly influenced by the temperature conditions of the environment, which does not happen with these;

    - *TPA81* temperature sensors. These are sensors composed of an eight-pixel matrix that detects the temperature in a range of about 120 degrees and therefore allows to detect heated victim from a distance, after using other sensors we focused on these because they had a much higher accuracy than other commercial sensors, and the use via I2C is very simple.

- The orientation in space is given by the *BNO055* module, a triaxial gyroscope, accelerometer with 9-axis magnetometer, used in the IMU (Inertial Measurement Unit) mode which calculates the orientation in space, as Euler angles, starting from the acceleration data of the gyroscope and accelerometer. We have chosen this gyroscope for its ease of use through I2C communication and the vastness of data that it can provide us. Through this gyroscope we understand how we are positioned in the maze and if we are moving or not.

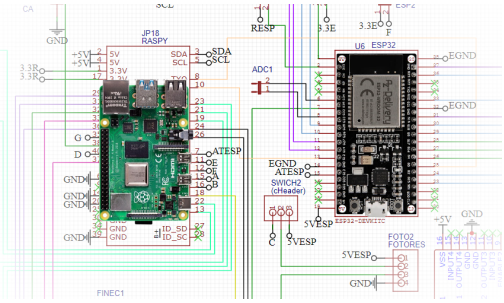


- The reading of the color is performed through a photoresistor placed in series with a fixed resistance, welded in a *PCB* created specifically by us, this allows us to choose which voltage value we will read based on the color of the box.
- All *DC motors* (*type JGB37-520 - 111 rpm - 12V*) used on the robot are equipped with an incremental rotary encoder with two phases out of phase by 90 degrees. The encoder of the main motors performs about 990 rotations for 1 rotation of the wheel; this allows us to calculate not only the speed with the sign of the motor but also the distance traveled by the robot.

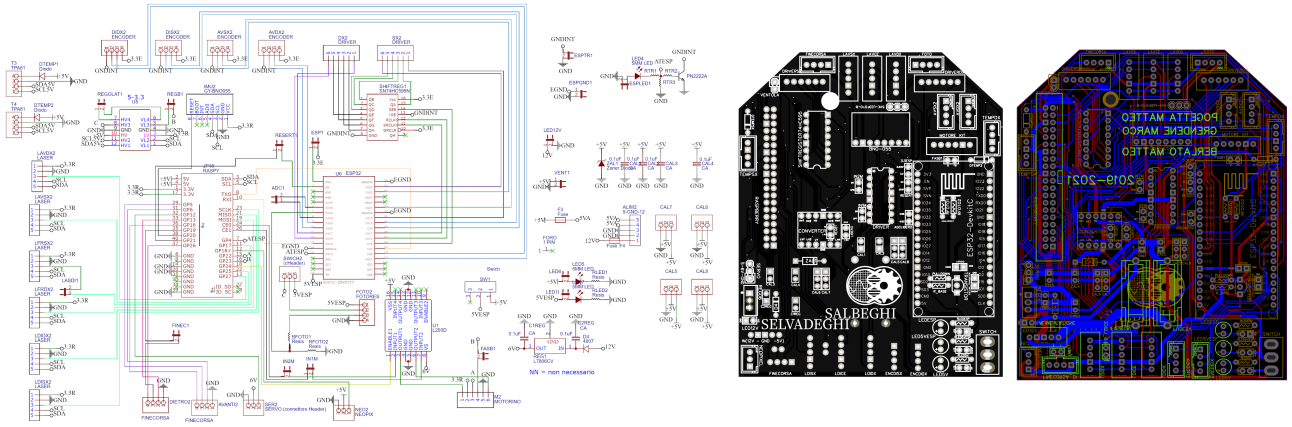


- The *limit switch* was built by us with laser-cut and folded sheets, with springs and screws. It consists of two distinct parts, electrically isolated, one in contact with the base plate brought to GND, and the other movable, and held detached by the springs, connected to a digital pullup input of the raspberry. This ensures that when the parts of a limit switch come into contact, the raspberry pin is brought to GND and therefore detectable at a low level.

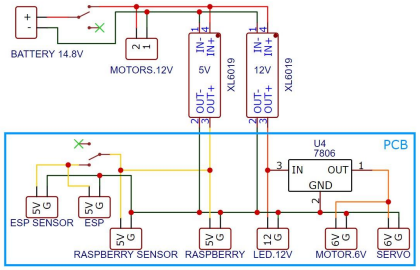
- Boards / microcontrollers:
  - The robot is equipped with two boards, a Raspberry Pi 4, which manages the logic, the analysis of the cameras and the reading of the sensors, and an Esp32 which instead performs the PID motor speed control and the reading of the photoresistor, the two boards communicate via serial communication, through an instruction protocol created specifically by us.
    - The *Raspberry Pi 4* board was chosen because it is one of the best boards that can be used to interface with webcams and has a very small size.
    - Instead, the *Esp32* board was used for its enormous calculation capacity, the possibility of programming through its framework gives complete control over what you are using, and obviously for the fact that it has pins for reading the Hardware inputs.



- PCB:
  - To connect all the boards and sensors together, a customized printed board was designed and built to occupy all the space available in the robot structure and to facilitate the connection of all the electronic components. We had previously created a board by soldering cables, but this became very complex to manage and it could happen that over time the cables were interrupted or detached, making the maintenance phase very complicated.
    The printed circuit instead has a thickness of 1.6mm, it consists of two layers and allows you to mount a series of JST 2.54mm connectors, placed according to the position of the sensors, motors and basic structures of the robot.



- Power supply:
  - The robot is powered by a *14.8V 2300mAh lipo battery*, this is interrupted by the main switch and then connected directly to the drivers. The motors are therefore powered with the same voltage as the battery, but since there is the PID controller the performance is not affected. from the power supply voltage, if above 10V.
    Instead, the circuit (PCB) is powered at 5V and 12V, the power supply of the main switch passes through two *XL6019* buck-boost converters which then lower to 5V and 12V. 5V is not interrupted for the power supply of the raspberry, instead for the Esp32 a switch has been inserted directly into the PCB; this interrupts the 5V at the Esp and therefore also acts as a robot operating switch, only this is used in the race.



5. Performance evaluation

- After these years of experience, continuous research, innovations and hours spent on the development of hardware and software that integrates perfectly with each other, to be able to bring the final project to an ever-higher level, we can now say that we have created a highly performing machine capable of overcoming even the most difficult challenges. As you can see from the videos sent, the robot always follows a precise logic and leaves no room for the most common errors committed. This is because the research and tests carried out have been designed to eliminate these types of errors. Obviously, it can happen that it can make mistakes on some occasions, but we have tried to minimize this, being able to say instead that in the vast majority of cases it is able to fully explore a race field at 100%.

6. Conclusion

- In this report, we couldn't manage to describe all our work accurately, and therefore to answer all your questions exhaustively. To this purpose we leave in the appendix the direct links to the complete documentation, to the source code, to all the images and videos, as well as to our GitHub page and Youtube channel. We are also glad to share our work with the RoboCup community so that it can be useful for future projects.

Appendix and References

- Group GitHub page: https://github.com/robocup-depre/Salbeghi
- Group Youtube channel: https://youtube.com/playlist?list=PL4E3h3ClAotzZJWBBYdVjQWadXayW_QSV
- Group Drive Folder: SALBEGHI_WEB