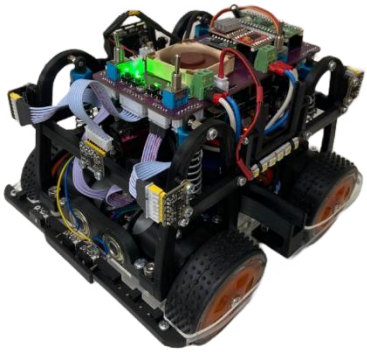
ROBOCUPJUNIOR RESCUE MAZE 2023

TEAM DESCRIPTION PAPER

Sacagnà

Abstract

- Our robot has been designed not only to complete every task in the competition, but also to do it as fast and in the best way possible. The machine, which has been completely designed in CAD to optimize the space and have an idea of how the robot would be, has fully independent suspension and 2 cameras, one for each side, that through OpenCV allow us to recognize victims. The robot is composed of a laser cut plexiglass base, on which is mounted a modular 3d printed structure, so that we can easily make changes. Our idea was to minimize processes and optimize every movement to make the robot faster, less sensible to different processing power changes and less error prone.



1. Introduction

a. Team

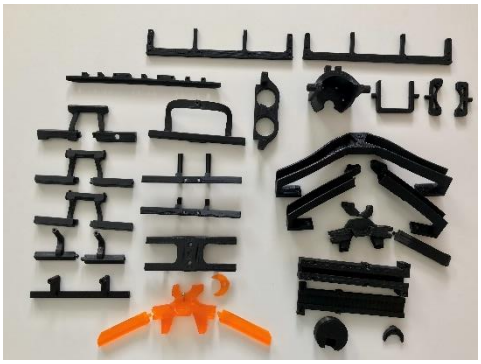
- The team consists of two people, each with its own tasks, but we have worked together on multiple occasions, in order to have a better understanding of how all the robot works.
- From the left:
    a. *Filippo Cracco* (Captain) : PCB Design, CAD, Sensors, Software, Hardware
    b. *Alessandro Calli* : Hardware, Mapping, Exit strategy, Test
- This is our second year of participation in this competition, in the previous year we participated in the Rescue Line Italy, finishing in 30th place out of 60 participants. This year we finished 5th in the regional and 6th in the national championship.

- Our first experience has led to the decision of changing league, which is based on the harder challenges that the Rescue Maze has, with the need of more sensors and more precise movements. Even if we have changed league we still carried on some experience, that has led to the creation of the current machine, and has allowed us to experiment new hardware and software solutions.

2. Project Planning

a. Overall Project Plan

- The main reason we participate in this competition is to develop our technical and IT skills, but above all to develop soft skills, such as teamwork, problem solving and responsibility. 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 our previous experience and past students experience we were able to identify the main challenges and limits the machine would encounter from a structural and programming perspective, we tried to develop a robot that would solve these problems and also overcome new challenges by implementing new solutions.
- With this robot we have tried to implement some new solutions, but also to improve some that we had seen. We created a rigid structure that surrounds the whole robot and prevents it from getting stuck, it is a 5mm Plexiglas sheet cut with a high precision laser onto which all the 3d printed parts are mounted, we have an independent suspension system that pivots on 8/14 bearings. We have inverted the position of the PCB and the kit release that has allowed us to an easier access to the PCB and has the benefit of a more controlled kit release, because the kits have less potential energy. As the robot is mostly 3d printed, it is fully modular and modifying things has been quite easy. On the software side the implemented a modified Dijkstra algorithm that is a bit lighter and we slimmed the programs to 1/3 of the previous robots in our school.



b. Integration Plan

- The robot has been entirely designed in 3D before building it, 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. The sensors and actuators have been positioned in order to make it easy to access and change them if needed, on the PCB everything is on connectors, so we can swap components in a matter of seconds, and the motors can be changed without disassembling all the robot.

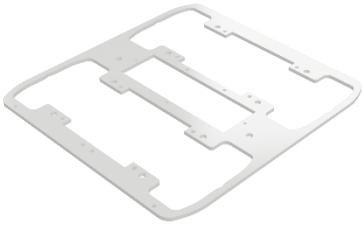
3. Hardware

- This robot, as already said, 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 (we are at version 2.5 of the robot), the main idea 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: PLA, 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

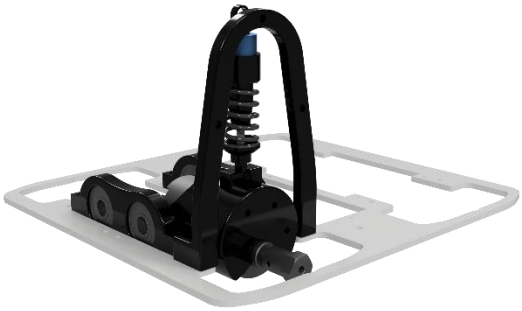
even later and to add modules not considered before. As you can see from the various renderings it is not a classic robot, we decided to experiment with new solutions.

a. Mechanical Design and Manufacturing

- MAIN PLATFORM: The floor of the robot, where all the other parts are fixed, is made of a 5mm Plexiglas sheet, with a maximum size of 190x210 mm, designed in 3d and laser-cut at our school. This allows us to have a smooth outer edge of the robot, in which it is difficult to get stuck, it is a bit bendy but with all the robot assembled it has great structural solidity. All the other components of the robot have been screwed onto the base, from the suspension supports to the limit switches. In the centre we have a hole for the battery, which is positioned as low as possible to maintain stability on speedbumps and ramps



- SUSPENSION: The suspension is an experiment carried out by one of our school past students, we decided to implement it as well since it gives the robot extra stability and overcoming obstacles becomes easier and with constant speed on all the wheels the robot remains perfectly straight.



- ACTUATORS: The robot moves using 4 independent 12V motors, that have a rated speed of 107 rpm, they are driven by two dual channel TB6612FNG H-bridges. The four motors have incremental encoders with a resolution of 990 ppr. The kits are released with a 9g Servo that directs the kit either left or right.

- LIMIT SWITCHES: One of the experiments we did was to heavily modify the concept of limit switch, we have used 2 micro switches to create one, which is actuated by a plastic cover that protects the delicate switches. By using 2 switches, it can be activated by applying force in every direction.

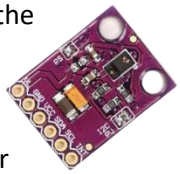
- KIT RELEASE: One of the other experiments was to lower the kit release in order to drop kits close to the victim, without them bouncing around. Initially the kits were designed to be coin shaped, as last year's rules defined the minimum volume, but with the change of the regulations we went back to the cubic kits. The kits in the magazine are pushed by an elastic band, that positions the kit in the selector, and with the mini servo we deploy the kit either left or right. The kit magazine is also a structural part, that connects the back with the centre section and it holds the voltage regulators.

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.

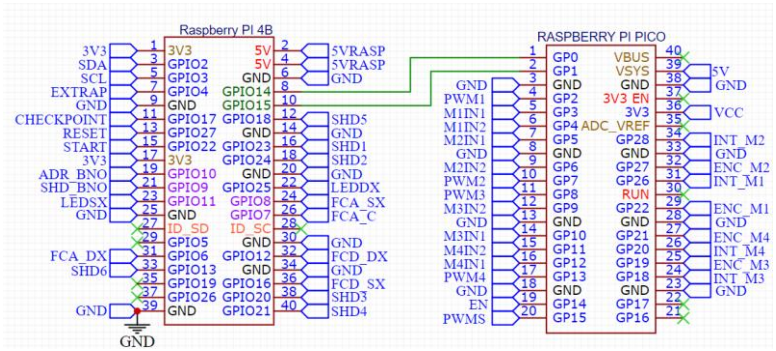
- The orientation in space is given by the BNO055 module, a triaxial gyroscope, accelerometer with 9-axis magnetometer, used in the IMU (Inertial Measurement Unit) mode which calculates the orientation in space, as Euler angles, starting from the acceleration data of the gyroscope and accelerometer. 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.

- In order to read the colour of the floor we have chosen after lots of testing the APDS9960, as it was the only sensor that was able to distinguish blue, black and grey. It is a multi-channel colour sensor that outputs 4 values: Red, Green, Blue and Clear. As all the other sensor we chose it also for the I2C interface, that allows us to easily connect it in parallel with all the other sensors.



- The 4 DC motors are equipped with a two-phase incremental rotary encoder, it has a resolution of around 990 per wheel revolution and it allows us to monitor speeds and distance travelled.
- The limit switches are connected to digital pins on the Raspberry Pi 4, they are connected to ground in the normally closed position, in order to detect also cable and contact failures. The pin on the Raspberry Pi 4 is pulled high, when the limit switch is actuated, the contact opens and we can detect the high 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 a Raspberry Pi Pico which instead performs the PID motor speed control, the two boards communicate via serial communication, through an instruction protocol created specifically by us.



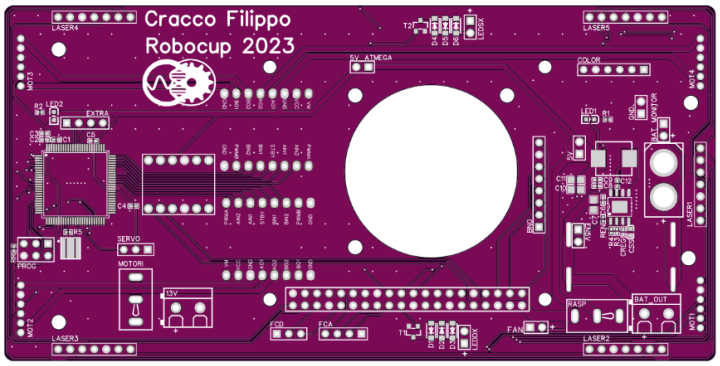
  - The Raspberry Pi 4 board was chosen because it is one of the best boards that can be used to interface with webcams and has a very small size.
  - On the other side the Raspberry Pi Pico is used to control the motors, it has been chosen because of its high versatility and low price, it has 2 cores, a 125MHz clock, 16-bit resolution PWMs and externa hardware interrupts on all pins. We started with using a custom board with an Atmega2560 (the Arduino mega one) but it became clear that it was to slow for our application.

- Power supply: the robot is powered by a 14.8V 2300mAh LIPO battery, that is connected to the main switch. The circuit (PCB) is powered at 5V and 14.5V, the power supply of the main switch passes through two XL6019 buck converters which then lower to 5V and 14.5V. On the board we have 3 more switches, 2 for 5V: one for the Pi 4 and one for the Pico, and one for the 14.5V that goes to the drivers.

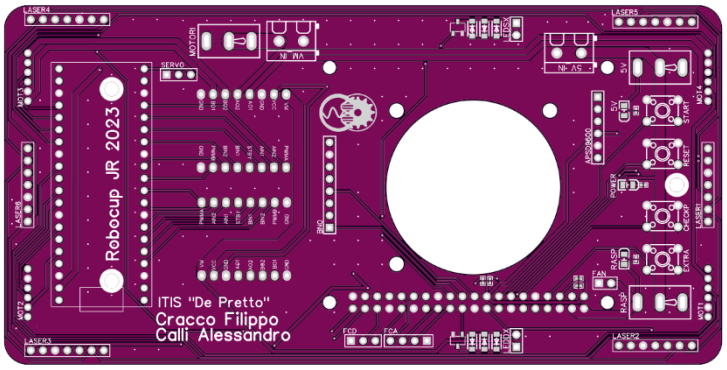
- PCB: to connect all the boards and sensors together, a customized printed board was designed and built to occupy all the space available in the robot structure and to facilitate the connection of all the electronic components. It is the core of our robot, we fitted onto it everything we could, in order to have less cables possible and more stable connections. The printed circuit allows us to mount a series of JST 2.54mm connectors, placed according to the position of the sensors, motors and basic

structures of the robot. On the board we have both the microcontroller and the Raspberry Pi 4, with the hole for a fan, the 2 H-bridge drivers, the BNO055 and some switches and LEDs. We made 3 versions of it:

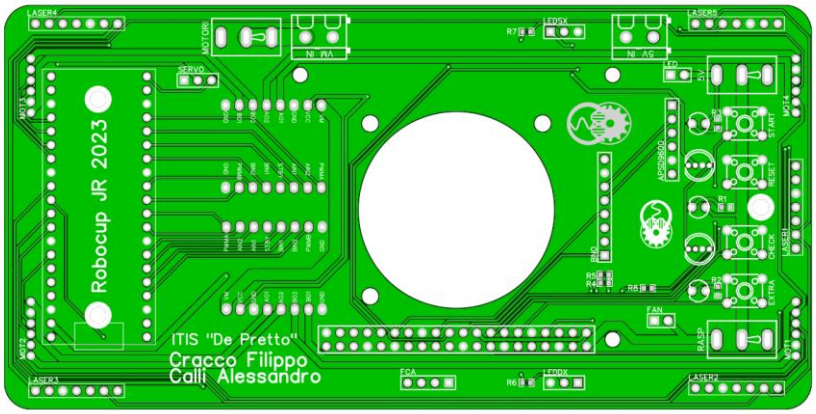
- The first one was an experiment, it included a 5v switching regulator, an Atmega2560 chip used as an Arduino Mega with its dedicated clock, capacitors and programming connector. It ended up being unreliable and had lots of interferences. The voltage regulator would even catch fire sometimes.



- The second version was a complete change, we switched to a more traditional philosophy, but with the Raspberry Pi Pico, we put an external 5v regulator and added some pushbuttons to control the robot during the run. As the Pico runs at 3.3V like the Pi 4, we were able to take out the logic converter for the serial communication



- The third one is just a revision with better component placement and 4 layers that was made possible by two affiliated companies.



4. Software

a. General software architecture

- The software of our robot is divided in two parts, one is for the PID control of the motors and is on the Pico, the rest is on the Pi 4:
    - PICO: it is programmed in C with its own SDK in VS Code, it contains all the actuation part and communicates through UART with the Pi 4. To be able to control the speed on the four motors independently we implemented a PID control loop on each motor, which is fed with the desired speed and outputs the PWM correction. The PICO is also responsible for the kit release, it runs a 50Hz PWM to the servo to control its position.
    - UART: Communication between the Pico and the Pi 4 takes place via an appropriately managed RS-232 serial bus, it is a 1,000,000 baud protocol, 8 data bits for transmission and 1 bit stop. The communication is bidirectional, but only the Pi 4 makes the requests to the Pico, and the Pico responds with the desired values. We have developed our data protocol in order to minimize the number of bytes sent, it is thought in such a way that we can add more requests. The Pico is programmed so that if it doesn't receive anything for a long period of time it automatically stops the motors, the interface is completed with a digital signal that allows the Pi 4 to reset the Pico.
    - RASPBERRY PI 4: the code of the Raspberry Pi is organized in different files, containing different classes (sensors, movements, image interpretation and main logic). Moreover, the Raspberry Pi has to run numerous programs in parallel, including the main logic and two image recognition. 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 signalling LEDs, each code for the recognition of the cameras makes use of a thread for the continuous taking of the frames from the camera.
    - FINITE STATE LOGIC: we have started to implement a finite state logic, it 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 Pico board occurs once per cycle, this allows us to be sure of the operation of the Pico 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.
    - 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, align, centre, 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.

- 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.
- 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 centre 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 centre 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 coloured 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 colour. 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 analysed contour by contour. The colours are detected with a simple mask, the letters are recognized with the function convexity defects, that returns the number of concave defects and all the points of the defect, and after some conditions we get the letter.

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.
- 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.

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. The robot is capable of exploring the maze in its entirety, but the room for growth is enormous.

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 source code, our GitHub page. 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/Flink22/Sacagna