RoboCupJunior Rescue Maze 2023

Team Description Paper

Laberinto

Ahmed Wael, Ahmad Zaki, Abdullah Ayman, Adam Mohamed

Mentors: Adham Amr, Amr El Shabacy

Sidi Gaber Language School for Boys, Alexandria, Egypt

Innova Stem Education

laberinto.team@gmail.com

Abstract: This paper describes the details of hardware and software that Laberinto team developed to compete in RoboCup Junior 2023 Rescue Maze Category. It showcases our robot's mechanism to deploy rescue kits, the algorithm used to navigate through the maze and the algorithm used to detect and identify the victims in the maze.

1. Introduction

1.1 Team Background

Team Laberinto was first formed in 2016 and this is the second generation of this team which started in 2022. Our team consists of 4 students, 2 of which are university undergraduates and the other 2 are high school students. Ahmad Zaki is the team leader and is responsible for the software development of the robot together with Ahmed Wael. Abdullah is responsible for the electrical system of the robot. And finally, Adam is responsible for the mechanical design of the robot. All the team members have already participated multiple times in robotics competitions such as RoboCupJunior Egypt in the Rescue Line Category, World Robot Olympiad, and Mate ROV.

Figure 1: Team Photo

2. Robot

2.1 Hardware

2.1.1 Mechanical Design

Overview: Our aim while developing this design was to make as little wiring as possible to make it easier to connect all the electrical components with each other or disconnect them in the least possible time, and to help us debug any electrical issues that might face us. For that reason, we decided to make the top plate a PCB plate.

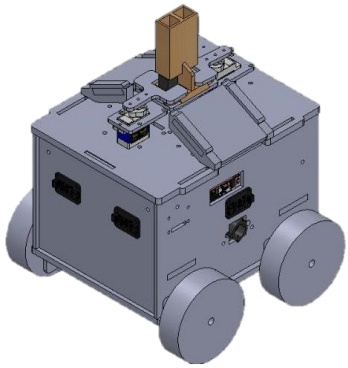

Figure 2: Robot design on CAD software

PCB plate: The top plate (3rd plate) is considered a PCB (Printed Circuit Board) which connects our two micro-controllers together without the need of any external wiring and connects them with all the sensors and cameras with data cables to make all the connections traceable. In addition to that, all the components that are connected to it are connected from the bottom side as to leave space for the rescue kits mechanism to be mounted on it.

Mechanism: Our rescue kits mechanism consists of three main sections: 2 slides, kits holder and 2 kits hitter. The kits holder is a 3D printed part that can hold up to 12 rescue kits (6 on each side). The rescue kit is a 1 * 1 * 1 cm cube that is 3D printed too. We decided to develop this design for the kits holder as to make the robot shorter and at the same time be able to deploy the rescue kits either to the right or left side according to the victim's position.

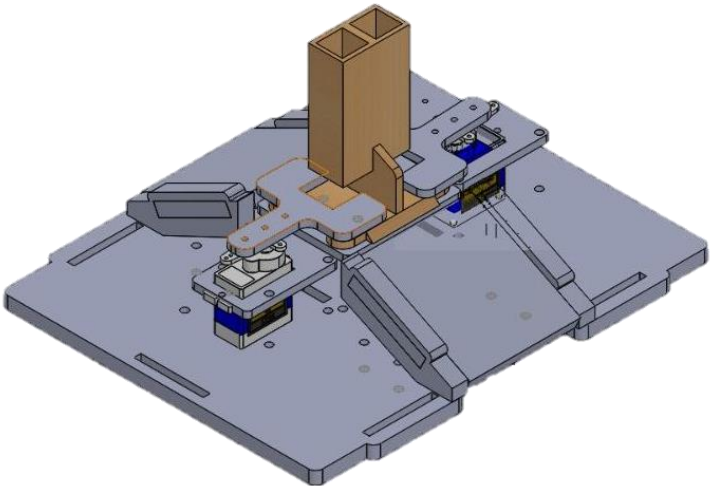

Figure 3: Top plate (PCB)

2.1.2 Electrical Design

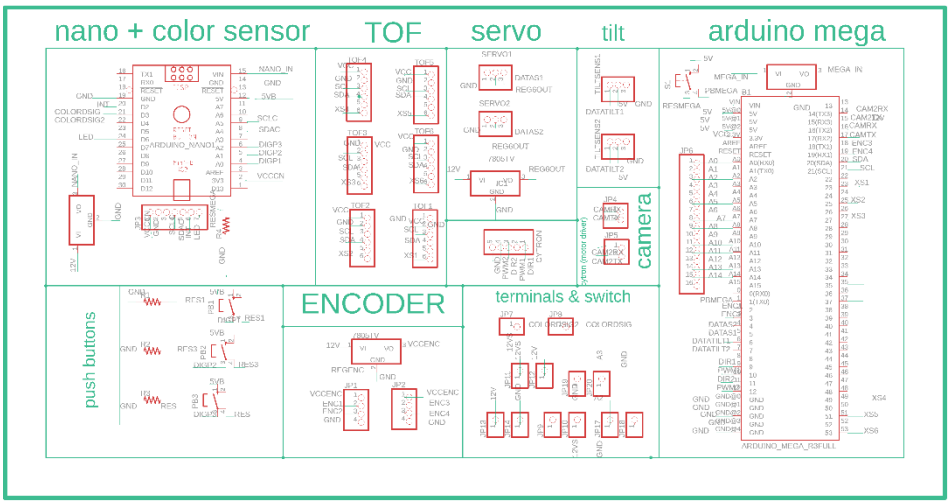

Figure 4: Electrical system schematic

PCB: Our electrical system is mainly based on a plate PCB which is our 3rd plate as mentioned before. This PCB connects all our micro-controllers together, supplies power for all the components, each with its suitable voltage and has an LED and a buzzer for kit identification notification. It also connects 4 pushbuttons to the Arduino Nano for color sensor calibration.

Micro-Controllers: We use 2 Arduinos to control our robot, one Arduino Mega, that is our main controller and is responsible for controlling the actuators and the rescue kits mechanism. There is also one Arduino Nano which is our secondary controller which is responsible for getting the readings of the color sensor and the built in IMU readings and processing this data and then sending them to the Arduino Mega through SPI communication to use in the algorithm.

TOFs: In the national qualifications we were using only 2 TOF sensors: one on the front and one on the right while that was a simple configuration, it didn't give us the needed accuracy while navigating through the maze especially when there is no wall on the right of the robot to align itself to so we decided to use a 5 TOF configuration: 2 up front, 2 on the right and 1 on the left.

Cameras: We use 2 Openmv H7 Cameras mounted on each side of the robot to detect the victims. If a victim is detected it sends the number of victims to be deployed to the Arduino Mega through UART communication.

Table 1: Components list

Function	Part	Qt.
Control	Arduino Mega	1
	Arduino Nano RP2040	1
Power	3S 2P 9600 mAh Battery	1
	PCB (Top Plate)	1
	DC-DC 12V to 3.3V Output Voltage Module	1
Movement	12V DC Worm Gear Motor with Encoder	4
	Cytron 10A Motor Driver	1
Cameras	Openmv H7	2
TOFs	VL53L0X	5
Orientation	IMU (built in Arduino Nano RP2040)	1
Rescue Kit Mechanism	Micro Servo	2

2.2 Software

Overview: The main idea behind our algorithm is that the robot does one lap around the map using the right-hand algorithm till it reaches the starting point again. Then it tries to visit the remaining unvisited tiles and after all the tiles have been visited, it returns to the starting tile again and stops to finish the round.

Movement: We use encoders to measure the length of every tile that the robot passes over. We already measured how many revolutions the motor makes every 30 cm (tile length), and we use this number to correctly move between tiles. We also use the absolute difference between the right and left encoder signal values to correctly turn 90 degrees. However, we recently opted to use an IMU to accurately measure the yaw angle of the robot and turn 90 degrees accurately. When it comes to the robot moving in a straight line, we use two right TOF sensors to align the robot's orientation when moving forward and two front TOF sensors to align the robot's orientation if there is no near right wall accessible which usually happens when the robot is trying to visit the tiles in the middle of the map. We also use one left TOF to avoid collision with any wall on the left of the robot.

Right-Hand algorithm: To allow the robot to navigate through the maze we start by detecting whether there is a right wall to the robot or not, if not the robot rotates till it finds a right wall then we make the robot move forwards so that there is always a right wall to the robot which will eventually lead it to its starting point again while knowing the edges of the map. While it is moving in this first lap it records data about the walls in each tile and the location of the tile and saves it to a map in the robot. It also detects victims using the cameras while moving and deploys rescue kit(s) if needed and saving it to this specific tile in the map so if the robot passes by this tile again it doesn't deploy a rescue kit again.

Victims: Every frame that we get from the Openmv camera, we try to find any blobs in this frame. A blob is an area of pixels in an image that lies within a certain threshold. We searched for color victims which are red, yellow, and green. If we find any of these blobs, we first check whether it meets certain criteria such as: blob area size and position of the center of the blob in the image, then we accordingly send the number of kits that needs to be deployed to the Arduino Mega. If we didn't find any of these colors and we found a black blob, we first decide whether it meets our very strict criteria; the area, center, density, length, and width of the blob so as to recognize it as a victim or just ignore it and treat it as a misidentification. We then divide the blob into multiple ROIs (Region of Interests), we get the area of the blob in each ROI and if it exceeds a certain number, we know that this ROI contains a part of the letter, then we compare all the ROIs with previously saved configurations of ROIs saved to each letter and sends accordingly the number of victims that needs to be deployed to the Arduino Mega.

Unvisited tiles: After the robot has done one lap around map, the algorithm will start to find a path to the nearest unvisited tile that lies next to a visited tile. The algorithm calculates the Euclidean distance from the current tile to all the unvisited tiles and starts to order them ascendingly. Then it checks whether that tile can be reached by moving in a naive path (moves in X-axis first then Y-axis or vice versa). If there isn't a naive path, then we move according to the right-hand algorithm. Once the robot reaches the new tile it will mark it as visited and this whole process will start again.

IMU: As we are using the built in IMU in the Arduino Nano RP2040, it is only a 6-axis IMU, and we need to calculate the yaw angle for 90 degree turns so to achieve that we fused the accelerometer and gyroscope readings using the Madgwick filter and used a complementary filter to minimize the errors in readings caused by gyroscope drift. Also, we use the pitch angle to detect whether the robot is on a ramp or not and store this information.

3. Performance

3.1 Testing

Hardware testing: We start the hardware testing with a checklist to ensure that all mechanical and electrical subsystems are working properly. This process is usually done after changing anything within the hardware as connecting or disconnecting any components or repairing any hardware damage. There is also a testing phase at the start of the algorithm to test all the sensors and if there are any errors the code is aborted and the sensor with the problem is stated to check it for any wiring issues.

Software testing: Every time we test the algorithm, we create a new map and make a test run each time we modify the algorithm and record the scores of every run. We repeat this process with the same map until the robot scores higher than a certain percentage of the maximum points it can get in this map then we make a new map and repeat this whole process again.

4. References

Our GitHub Repository: https://github.com/AWael8/Laberinto-Maze-Solving-Robot