RoboCupJunior Rescue
Team Description Paper (TDP)
RoboCup Asia Pacific 2018 – Kish Island

Team Name: F6

Participants Name: Nargess Seifi, nargessseifi@outlook.com

Sanaz Motie, s.motie81@gmail.com

Mentor Name: Azade Dulabi, paj.f6@yahoo.com

Institution: Farzanegan 6 high school

Country: Iran

Contact Person: Nargess Seifi

Date: September 2018

RoboCup Asia Pacific 2018 – Kish Island
Team Description Paper (TDP)

Sanaz Motie, Nargess Seifi

F6

Iran

Preface

Our team began its work in the rescue maze league from 2017 October and made an appropriate robot. the advantages of our robot are having servo motors instead of DC motors which most of the teams use, openmv m7 for image processing because of its advantages that we will describe them later, how we sketch the two-sided rescue kit dropper in small size, low weight and volume that help us to increase the accuracy and saving the time. we used eight sharp sensors, to measure the distance between the robot and the walls. These sensors detect the obstacles, but to detect the transparent obstacles we implemented two ultrasonic sensors.

Keywords: Openmv, Servo Motor, Two-Sided Rescue Kit Dropper

Introduction

Nowadays the robotic science is so popular and practical and develops every second so it's great to be familiar with it. Last year we chose the rescue maze league between the other robotic junior leagues, and we started working in this league because of our interested on programming, electronic and mechanic also in this league we can use creative ideas. We won the second place in IranOpen 2018, earned valuable experiences and got some ideas from other teams' robots, and used them to improve our robot and we decided to participate in RCAP 2018 too.

Honors:

- Second place in IranOpen 2018
- Best technical challenge in IranOpen 2018
- Best technical presentation in IranOpen 2018
- Qualified for participating in RoboCup 2018

Rank	Team	Round 1	Round 2	Round 3	Round 4	Round 5	Round 6	Round 7	Round 8	Round 9	Technical Challenge	Educational Videos	Sum
1	helli 1	200	0	305	165	265	250	515	415	350	120	270	2855
2	f6	135	260	245	175	350	135	145	155	465	300	300	2530
3	HEXAGONAL	120	270	180	145	145	290	295	375	245	180	300	2425
4	Allame Tabatabaei	185	65	175	0	270	255	175	470	410	140	250	2395
5	A.T.R.T	40	95	105	200	75	210	195	230	115	0	300	1525
6	POOYESH-B	155	105	105	180	200	125	0	150	0	60	270	1350
7	KavoshDanesh	0	0	90	125	180	55	145	205	110	100	250	1260
8	raymand	0	0	0	195	135	75	130	85	150	120	285	1175
9	Minotaur	0	120	30	30	90	70	80	170	240	0	300	1130
10	Sadrarobot4	50	55	95	90	105	100	0	115	10	40	220	880
11	Salam Zeynoddin	40	35	0	40	60	70	55	105	20	20	250	695

Table 1- rescue maze score table of IranOpen 2018

RoboCup Asia Pacific 2018 – Kish Island
Team Description Paper (TDP)

1. Locomotion system

1.1. Mechanics of locomotion system

1.1.1. Body

To design the body of the robot, we used AutoCad and SolidWorks softwares. We used Plexiglass material for the body of the robot, which makes it very light and solid. The body is designed to fit the other components and the electronic board.

1.1.1.1 Two-sided rescue kit dropper

One of the advantages of our robot is the rescue kit dropper, which includes a moving plexiglass plate, a two-sided ramp (40 degrees), a tank containing 12 cylindrical kits and a servo motor. The servo motor is dynamixel xl-320. It is very small and has a high resolution of 0.29°, which are very important to improve the robot's performance. We took this idea from RoM group which won the first place in the 2016 Robocop.

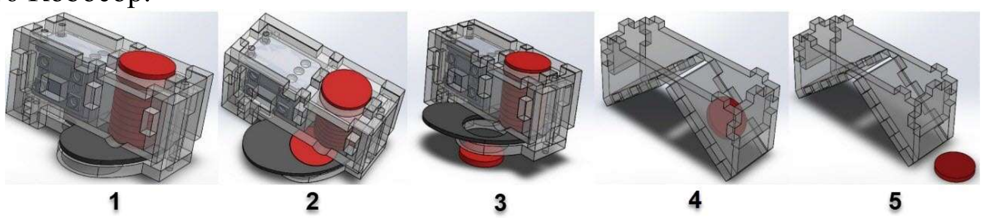


Figure 1- work steps of dropping a rescue kit

1.1.2. Motors and Wheels

We used Robotis servo motors instead of DC motors, which have PID controllers and are able to provide a constant velocity in every situation. In addition, the implemented driver motor in these motors, make the electric circuit simpler. We used 4 motors of this kind (xm430-w210 model) with the rotational velocity of 77 RPM [1], and connect them to the microcontroller by using UART and RS485 serial protocols.

Figure 2– dynamixel xm430-w210

We used four aluminum wheels with a 7 cm diameter. This height is appropriate to prevent any collision of the robot's bottom with speed bumps.

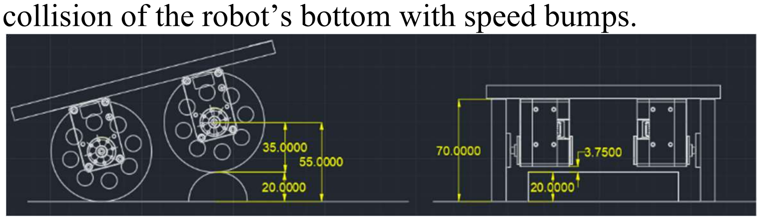


Figure 3– design of wheels and motors hitting the speedbump

1.1.3. Circuits

Servo motors have internal drivers, so we don't need any driver on the main board. We used a 3 cells lithium polymer battery which provides 12V and is appropriate for xm430-w210 motors that we used. We supply the power of xl320 motor, using the balance wire of the battery, because it supports at most 8V input so that it uses only 2 cells, but to supply the 5V and 3V for the other components we used miw3011 and miwi 10-24s05 voltage converters.

We used the industrial dspic33ep512mu810 microcontroller provided by the microchip company. We chose this microcontroller because of having high noise damping properties, 12bits ADC, different connection protocols and 60MHz processing speed. [2]

2. Maze solving algorithm

2.1. Field Mapping

Last year we used lidar lite v3 in IranOpen 2018 but in some situations, it gave us incorrect value; So this year we decided to use the timer and servo motor's encoder. First, we calculated the required time to reach a go from the center of a tile to the center of next tile; and using the internal timer of the microcontroller and according to the heading of the robot, the position will be updated. using this method, we could do the things like rotation in the center of tiles to make the collision with the walls minimum possible. in some situations, like when detecting a victim, we keep the time also while crossing a speedbump we ignore the time is running out and round of motors.

2.2. Moving Straight

We implement two gp2y0a41sk sensors from the Sharp company in each side of the robot and 4 cm from its edges to avoid any problems due to the blind spots. The sensors output is an analog voltage and we used an ADC module to read it. We also used two srf08 ultrasonic sensors in front of the robot to detect the transparent obstacles in the maze. This sensor has better accuracy among the other ultrasonic sensors and its measurement range is from 3cm up to 6m. We used an i2c protocol to connect to this sensor.

The cmps11 is an electronic compass and goniometer, which includes a compass, an accelerometer, and a gyroscope sensor. This sensor is not very accurate so we just used it to detect the direction (north, east, south, west). We used the accelerometer to detect the speed bumps and ramp. we connect this sensor to the microcontroller using the i2c protocol.

Figure 4 - cmps11

2.3.Algorithms

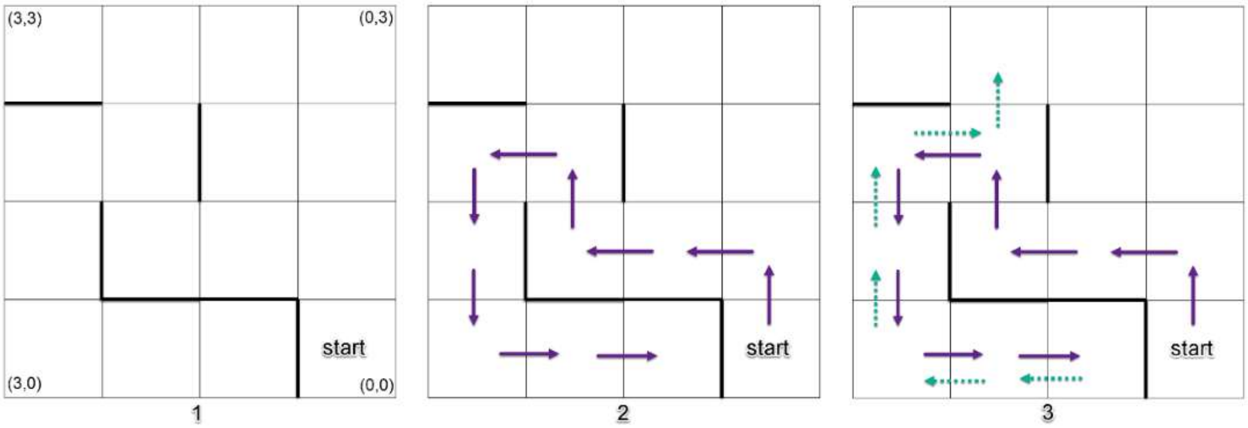
After passing each tile, the robot saves the properties of that tile in a 3D data structure array. For example, table 2 shows all the stored information related to the tile [1,1] (figure 3).

We use the left-handed algorithm with some modifications to search the whole maze. We define a 3D Boolean array including the length and width of the tile and the room number. The Boolean becomes 1 if the relevant tile is already visited, and consequently the algorithm prevents the robot to enter it again.

When the robot enters in a tile which has more than one unvisited neighbor tile (for example [1,1] in figure 3), it chooses one of them and pushes the coordinates of the others to a stack. The first coordinate pushed to the stack is the start tile, and the robot will come back home at the end. The robot goes ahead following this algorithm until it becomes surrounded with walls or visited tiles (tile [1,0] in figure 4 for instance). Then, according to the stored information, it finds the shortest path to the tile which is on the top of the stack. Thereafter, it will change the determined path to unvisited tiles in the Boolean array and automatically goes there.

<table>
<tr><td>0</td><td>North</td><td rowspan="4">walls</td></tr>
<tr><td>1</td><td>South</td></tr>
<tr><td>0</td><td>East</td></tr>
<tr><td>0</td><td>West</td></tr>
<tr><td>0</td><td colspan="2">Black</td></tr>
<tr><td>0</td><td colspan="2">Silver</td></tr>
<tr><td>0</td><td colspan="2">Victim</td></tr>
<tr><td>0</td><td colspan="2">Obstacle</td></tr>
</table>

Table 2- structure of the (1,1) tile



0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

0	0	0	0
1	1	0	0
1	1	1	1
1	1	1	1

0	0	0	0
0	0	0	0
0	1	1	1
0	0	0	1

Figure 5- movement of the robot in a hypothetical maze

(2،2)
(1،1)
(0،1)
(0،0)

Figure 6- tiles which stored in the stack

3. Victim detection

3.1. Sensors

The common sensor that most of the teams use too, is mlx90614 (figure 7), which is an infrared thermometer sensor that measures the temperature with 0.02°C accuracy without contact [3]. We put two of them on each side (left/right) of the robot. Also, we connect these sensors using i2c protocol. To detect the visual victims, we use openmv m7.

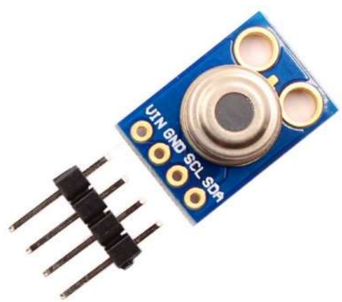


Figure 7- the mlx90614

3.2. Algorithms

To detect the heated victims, we compare the mlx's value with the value we found in the setup time according to the temperature of victims and the environment. If it was bigger, then the robot stops and checks the mlx's value again to be sure that its diagnosis is right. So after that, the robot drops a rescue kit and turns on a LED for 5 seconds.

4. Black/Silver tile detection

4.1. Sensors

For this part, we use laser class 2 and tempt6000 as a receiver. they are the appropriate selection for our requirement. For using the laser and the receiver we designed a PCB to connect them to the microcontroller.

Figure 8- laser color sensors

4.2. Arrangement and Design

We sketch the PCB so that connect the body of the robot with two screws and using two spacers we make the color sensors closer to the ground to be able to work correctly. Under the robot, we made a location to connect the sensor to the main board and made the possibility of separating the sensor's wire from the board, minimum in the competition or because of collision hand with it. Also if the sensor needs any repairing we don't need to open the whole robot and with opening the two screw we can separate the sensor easily.

4.3. Algorithms

To detecting the color of the tile, first, we saved the sensor's value in black, white and silver tiles. We found that using the sensor's original value makes many bugs because of the different value in different light conditions. So each time, we check the sensor's value for thirty times and for each color count how many times it is detected; the color of the tile is which its counter is bigger. We tried many numbers and found that thirty is the best.

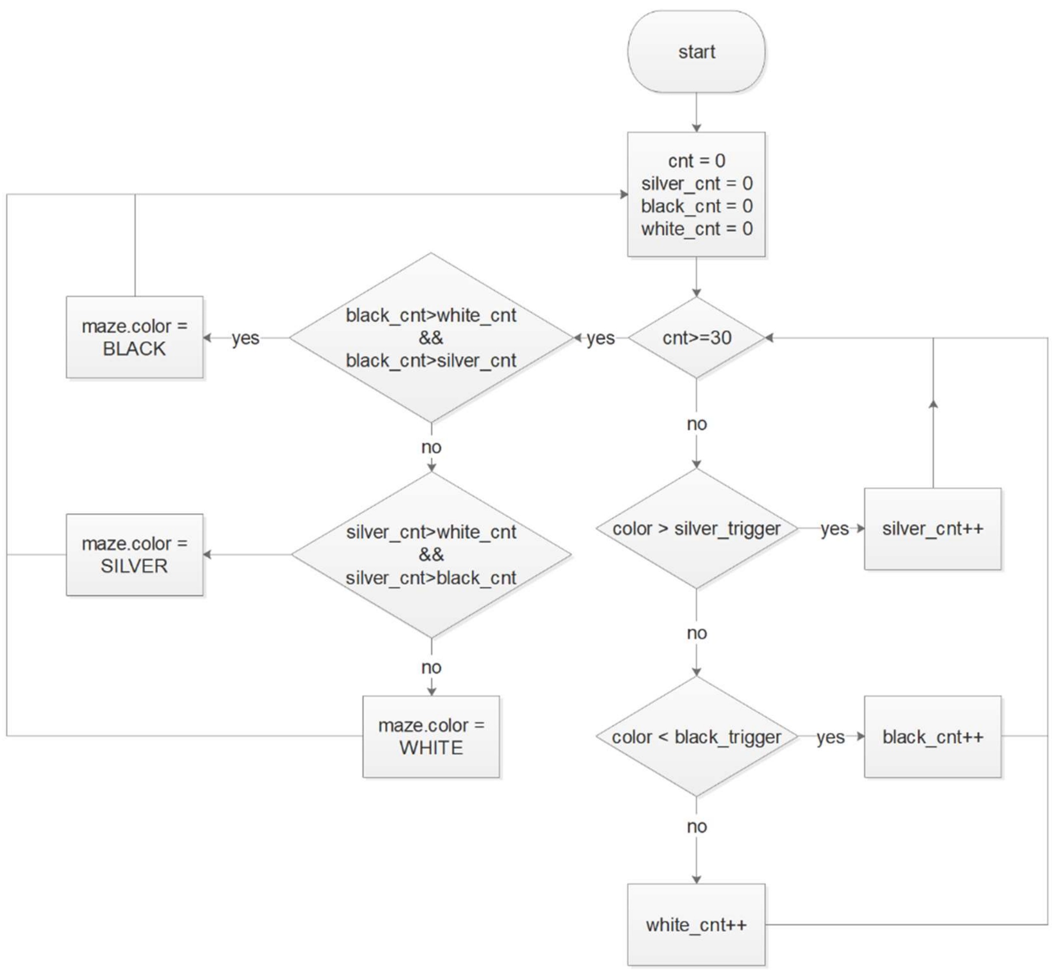


Figure 9- flowchart of determining the maze color

Although the robot does not enter to the black tiles, they will be considered as visited ones.
Each time we pass a silver tile, we store data of different variables related to position, data of each tile like the walls, stack etc. in the flash memory of the microcontroller so if we get lack of progress and when the robot’s power is gone, it turns on again and detects that it is on the silver tile so it will read all of the flash memory's data and beginning to search rest of the maze.

5. Visual Victim

5.1. Positioning Sensors and Circuits

To detect visual victims, there is a openmv m7 on each side of the robot, between the wheels. we use openmv m7 because of the high process speed, easy programming, possibility to see the image alive by connecting to the laptop using USB cable, distinguishing letters easier than other cameras.

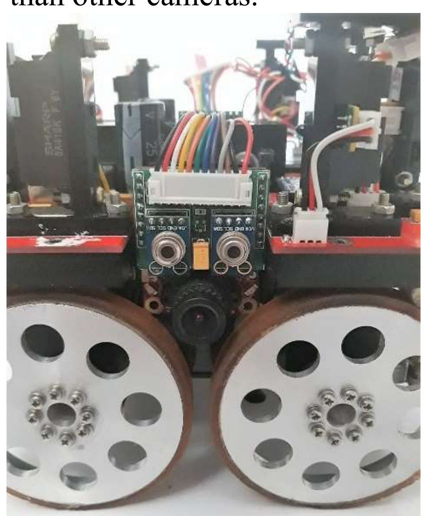


Figure 10- openmv position

5.2. Algorithms

First, we used a binary filter on the image. When the robot is searching through the maze, if the openmv detects a black object on the wall, draws a rectangle around it, the robot will stop moving. The openmv separates the object into simple lines (figure 11).

Figure 11- letters convert to simple lines

Then the openmv will compare it with H, S and U letters, according to some features like pixel density, Length to width ratio and the angle between lines. Then it sends the result to the microcontroller using two I/O pins. We wrote the code of openmv in python and in the different light conditions, we must to set the threshold of openmv and program it again.

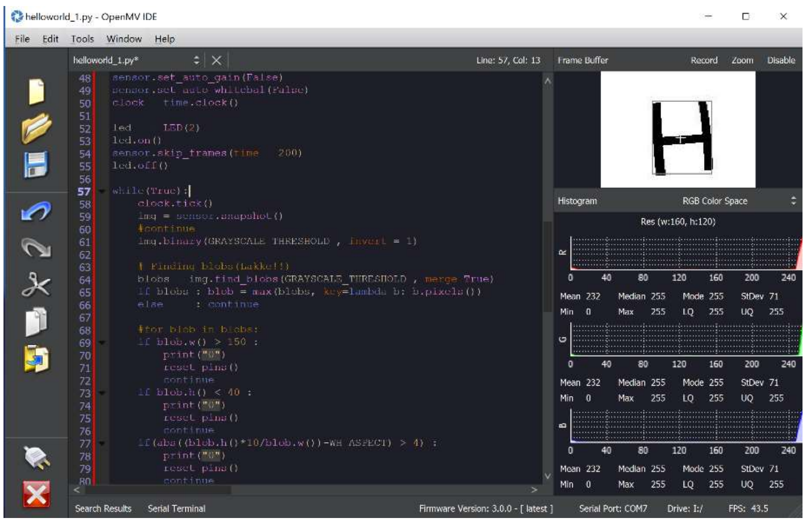


Figure 12- the openmv IDE

6. Innovations

6.1. Mechanical Designs

6.1.1. Sliding on a ramp

Sliding while passing the ramps is one of the challenges. We first used foam to address this issue; however, it was not the best solution. Looking carefully at the other competing teams, we understood that they usually use a mouse pad to increase the friction. However, we found that using a Gel (used in stamp Gel kit) on the wheel surface works much better to avoid sliding the robot on ramps.

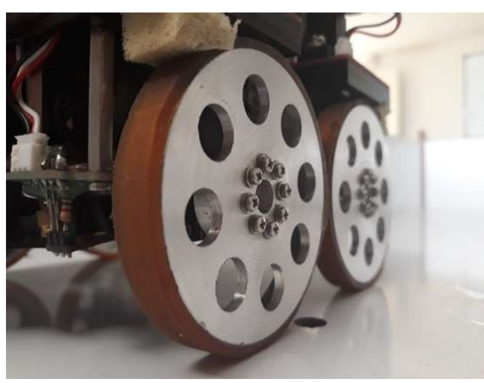


Figure 13- the Gel on the wheel surface

6.2. Circuits Design and Sensors

6.2.1. Transparent obstacles

Sharp sensors are not able to detect transparent objects and obstacles. To overcome this issue, two srf08 ultrasound sensors were used as supporting sensors.

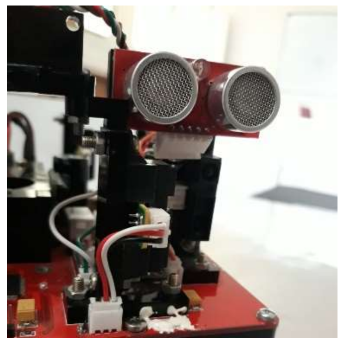


Figure 14- srf08

6.2.2. Repair feasibility

To facilitate and reduce the reparation time, the main board was designed in a way that all the sensors were set up on the main board; so that reparation makes less harm to sensors and parts.

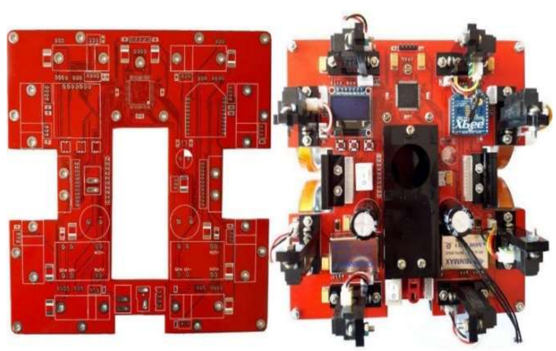


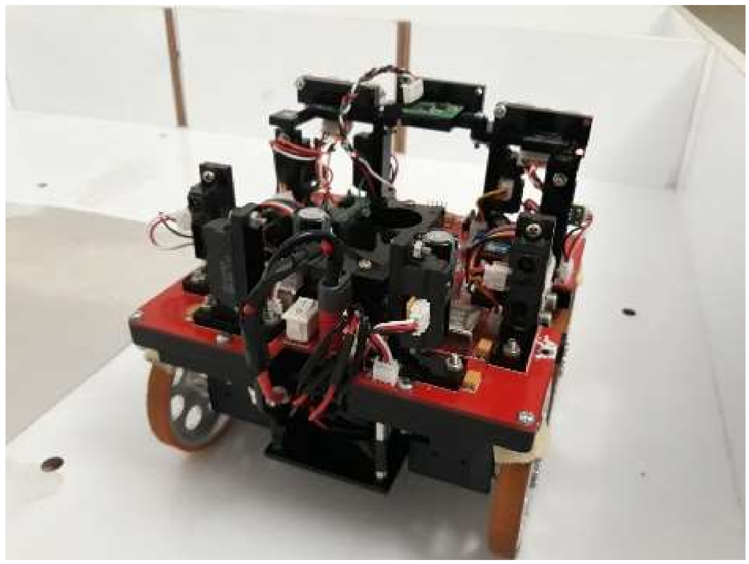
Figure 15- main board

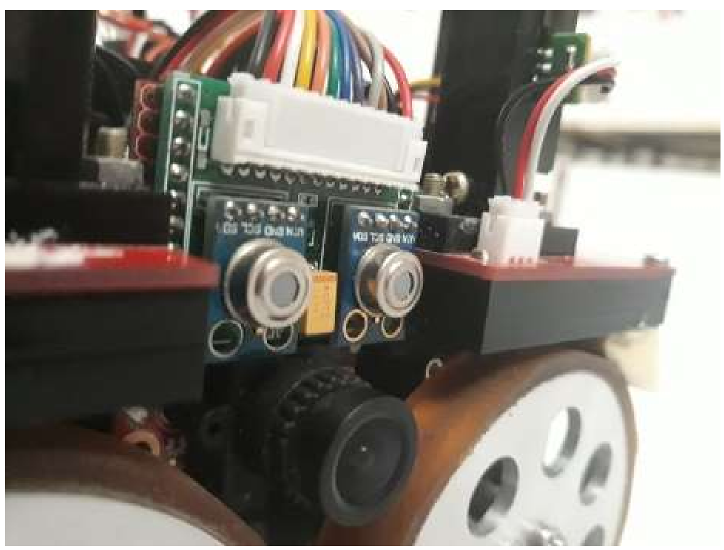
6.3. Programming and Algorithms

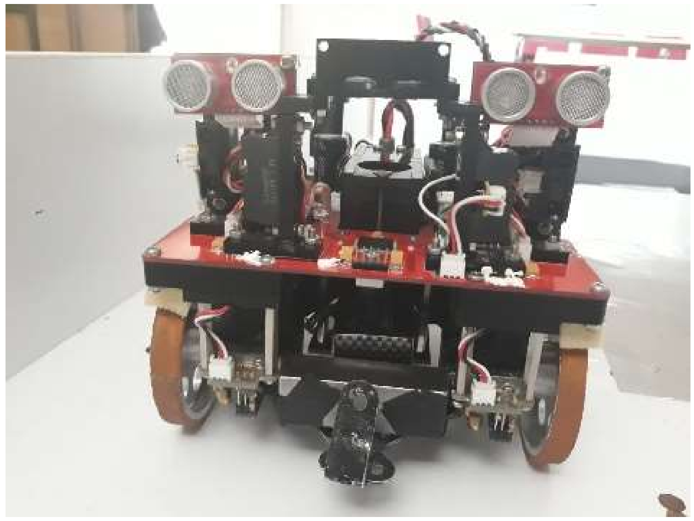
6.3.1. Test and practice facility

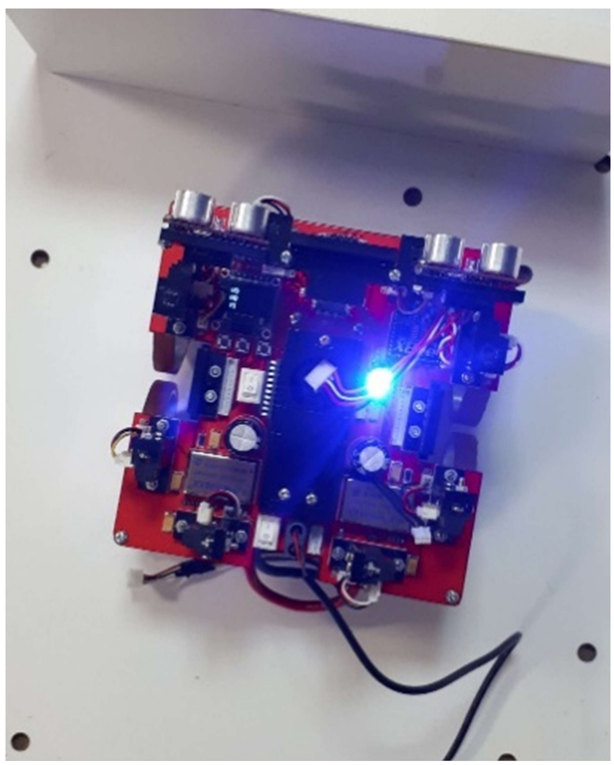
Due to the high number of variables and information, it was not possible to show all the information on the screen. A XBee was used to present the required information on the computer monitor.

7. Final Robot









8. References

1- en.robotis.com
2- microchip.com
3- sparkfun.com