Navigation support systems for autonomous vehicles: explorer

David Martín Jiménez

Resumen—El objetivo de este trabajo es presentar un algoritmo de exploración para reducir el tiempo inicial de instalación y costes de un sistema de exploración semiautomático, con fines educativos. El objetivo principal es obtener un mapa del área de trabajo del robot usando un algoritmo de exploración determinado, creando un modelo virtual del explorador. Este algoritmo será también testeado en un robot físico real, y sus resultados serán medidos en términos de eficiencia y eficacia [1].

Palabras clave—Robots, Vehículos autónomos, Sensores

Abstract—The aim of this work is to present an exploration method which can reduce initial installation time and costs of a semiautomated exploration system, but for educational purposes. The main goal is to obtain a map of the robot work area using a determinated scan algorithm, by creating a simulated model of the explorer. This algorithm is also tested in a real physic robot, and its results are measured in terms of efficiency and effectiveness [1].

Index Terms—Robotics, Autonomous vehicles, Sensors

1 Introduction And State Of The Art

AUTOMATED vehicles run by internal logistics can navigate industrial plants or auxiliary warehouses using a localization system. Since the less complementary infrastructure the cheaper (and the more efficient) transport, localization methods that work with on-board systems are preferred. Especially, the methods based on detection of characteristics of the environment. Thus, the location of the vehicle depends on its position with respect to the significant sites it identifies. Each significant place in the work environment must correspond to a unique set of features.

Apart from localization, the recognition of the environment also serves to create a map. In fact, one of the tasks that the vehicles must do is to update the map of the work area based on the changes they detect in the environment.

These tasks can be done simultaneously, and in fact, they are a typical problem in mobile robotics area: Simultaneous Localization and Mapping (SLAM) [2]. A solution to the SLAM problem provides the robots a true autonomy, and it has been one of the most important successes of the robotics community over the last years.

The SLAM has been implemented in different situations and environments, from indoor to outdoor, underwater and airborne, being satisfactory in all cases. Despite of this, there is still work to do, and perceptually rich maps should be improved.

The problem of the SLAM must be resolved independently of the mission that has a specific vehicle at a specific time. The objective of the SLAM solution is no other than help the vehicle to achieve its originals assignments in an easier and more effective way.

In this work, the mission will be precisely the exploration of the work environment. Starting from an unknown area, the robot would be able to explore it.

2 Objectives

The main objective is to make a demonstrator of a basic "explorer". The task to do is design a series of experiments in which an elemental robot equipped with a sonar or lidar can generate the map of a specific work space in an effective and efficient way.

This project is inspired in Embedded Systems practices, a 3rd course subject the student took part last course. The goal is to go further and to achieve more knowledge and a more functional robot than the current one, giving it the "explorer" functionality. It is also inspired in the real world AGV (Automated Guided Vehicles) intralogistics, which provide a series of advantages such as improvement of profitability and assurance of punctuality, reduction of transport damage and failure rates and short reaction times to changing order situations (https://ek-automation.com/en/technology/what-is-an-agvs/). In most real situations the mapping is already done, and the navigation takes place in a known environment, but the robots can have the ability to update the map while they are circulating. This is an important feature which makes them more sophisticated and autonomous and turns them in a kind of carriers-browsers.

Despite of that, in our work we are focused on the mapping due to the initial map could be done manually or with an explorer robot. In this sense, the work is aimed at the construction of one of these explorers. The first and basic objective is to achieve an effective robot (that works). An experiment is prepared, and the robot shows its functionality in a known circuit, all being recorded. A demonstration of the robot efficience is pending and a measure of quality should be determined. What is more, it would be nice if this exploration method was compared to others such as random, regarding to time, distance travelled or both, considering that the comparative could

be different depending on the surface and the environment among others.

The final objectives could be summarized in the following:

- Having the robot with the full stack of controllers
- Having an experiment to show the correct robot functionality
- A final video showing the robot working

Finally, as a requirement, this project should have been done in 150 hours of work plus an extra of 150 hours for "paper work", including writing the final document and recording a final video where the complete functionality of the explorer robot is shown.

3 METHODOLOGY

The methodology chosen for this project has been Agile. Every 15 days a different challenge has been solved or completed, and the student has been the responsible for stating these tasks and for prioritizing them.

During the evolution of this project all the documents, executables and files have been uploaded to a Dropbox folder the teacher and the student have access to.

4 EXPLORATION STRATEGY

The very first is to know that an explorer is a mobile robot which covers an unknown area with the objective of creating a map of it. This requires creating routes to unknown places.

In our case, the robot explores its environment thanks to an ultrasound sensor that can cover an angle of 180 degrees. This sensor returns the distance to the nearest object at a certain angle, and a virtual area delimited by the objects detected is created. Then, the software calculates the center of mass of this virtual polygon, and the robot advances into it. Using this method consecutively we achieve the exploration we are looking for.

It is very important to know how the algorithm our work is based (the center of mass algorithm) works. Here we have the mathematics [3][4]:

- Get the cloud of (α, r) points from sonar
  - Ordered by angle so each pair forms an edge
  - With conversion to Cartesian coordinates $(x, y) = (r \cdot sin\ \alpha, r \cdot cos\ \alpha)$
- Compute the centre of mass polygon defined by edges (x, y) $_{CoM}$
  - Also $(\alpha, r)_{CoM} = \{\frac{\Pi}{2} - atan2(x_{CoM}, y_{CoM}), \sqrt{[(x_{CoM})^2 + (y_{CoM})^2]}\}$
- Determinate final distance to advance
  - Initial $(\alpha, r)_{goal} = (\alpha, r)_{CoM}$
  - Rotate points $(-\alpha_{goal})$
  - $x' = x \cdot cos(-\alpha_{goal}) + y \cdot sin(-\alpha_{goal})$
  - $y' = y \cdot cos(-\alpha_{goal}) - x \cdot sin(-\alpha_{goal})$
  - $(\alpha, r)_{goal} = min\ \{y'|_{|x'|<w/2}, r_{goal}\}$
  - Final distance, $r'_{goal} = r_{goal} - r_{safety}$, (to avoid contact with detected points)

Where:

- x = center of mass angle coordinate
- y = center of mass distance coordinate
- α = goal angle coordinate
- r = goal distance coordinate
- w = max width / 2

We also present the states machine this algorithm has implemented:

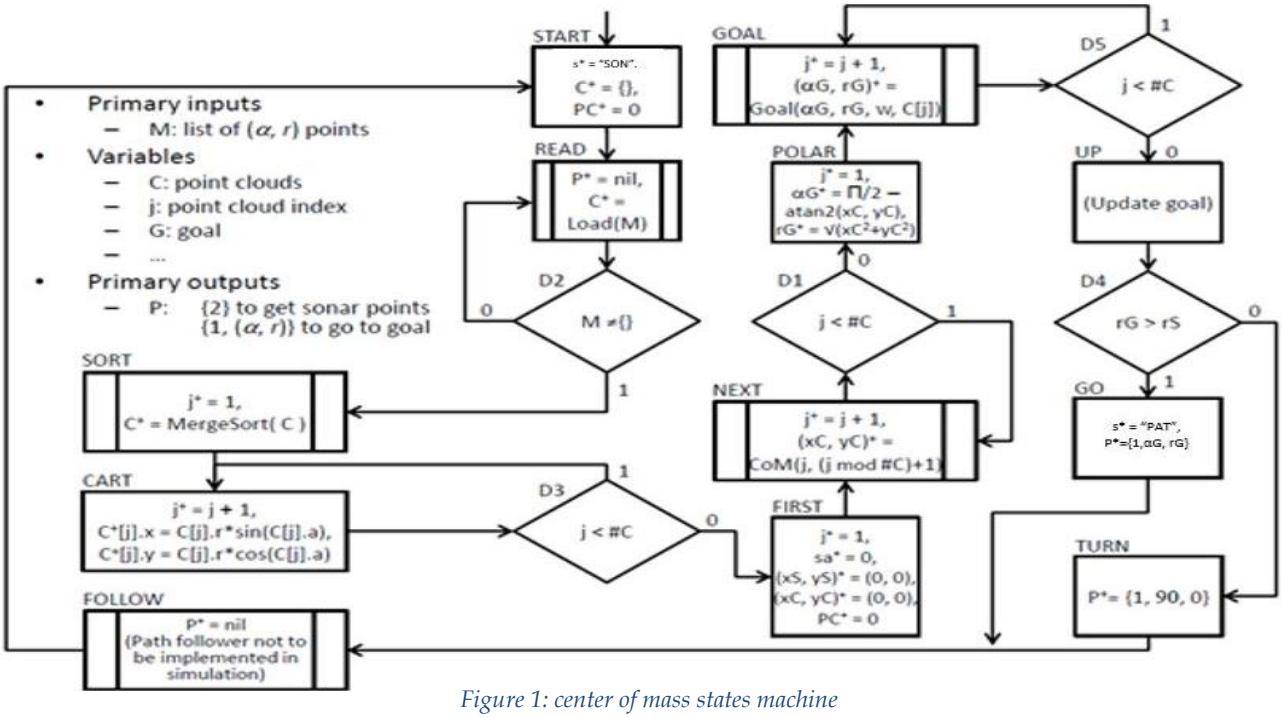

Figure 1: center of mass states machine

The code is divided in three different layers: L2, L1 and L0.

- L2: top layer. Is the one with the center of mass algorithm.
- L1: intermediate layer. Is the one who intercommunicates L2 and L0.
- L0: lower layer. Is the one who communicates with the virtual robot.

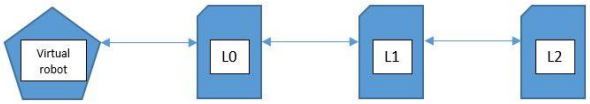

Figure 2: layers levels

The operation is as follows: L2 is the responsible of sending the variable "s" to L1. This variable can contain the value "SON" that indicates that a lecture of all sonar points should be done, or "PAT" which indicates that the robot should follow the route the variable "path" contains.

The logic order is that in the first loop a lecture of sonar points is requested. When L1 receives "SON" it sends a "SONAR" command to L0 that means go to the first angle and calculate the distance to the nearest object. After that, L1 sends "RESUME" commands to L0 meaning going to the next angle and calculating the distance to the nearest object again. The list of angles is a local variable located on L0, and when it arrives to the last angle of the list the robot returns a code "0008". When this code is read by L1, all the angles and its respective distance to their nearest objects are sent to L2, where the center of mass algorithm is executed. Finally, when L2 has the position where the robot must go, it is sent to L1 and this layer sends a "GO" command plus the goal angle and distance to L0, that executes the movement. After this, all the loop is repeated making exploration possible.

As a conclusion we have 3 possible commands (plus one control command):

- GO: move to the distance and angle indicated.
- SONAR: calculate the distance to the first angle.
- RESUME: calculate the distance to the next angle.
- HALT: abort mission.

On the other hand, the operation of the center of mass algorithm is the following: when L2 has all the list of angles and the distance to their respective objects, a Merge Sort algorithm is executed to order from smallest to largest angle. After that these coordinates are transformed to Cartesian (or rectangular) coordinates, and when all the point clouds have been transformed the center of mass is calculated. Then this point is transformed again to Polar coordinates (the ones the robot understands), and it is calculated the rotation angle and the distance to advance needed to reach this point. Finally, this point is sent to L1 and L1 sends to L0 the respective "GO" command.

L1 states machine is as follows: it is waiting until it receives the value "SON" or "PAT". In the first case it collects all the values L0 sends, and when the sonar sends the last value code it sends the values collected to L2 and goes to init again. Then it waits in this state until it receives the "PAT" signal, which indicates that the new coordinates (received from L2) should be sent to LO. When the movement is done, the machine returns to the init point ready to do a new cycle.

LO states machine is a little more complex. The reason is that all the values sent to the robot to control the movement velocity, the rotation velocity, etc, are managed by this layer. In the appendix can be seen both the L1 and the L0 states machine

This layer structure makes very important the order in which the values are passed during the simulation between them. In V-REP, if one layer has two reading functions, it will read first the first value passed. This means that if the values are sent to the corresponding layer in a random order, the execution will fail. In our case, the communication has been studied in a very exhaustive way, and the next timeline is the one we are working with:

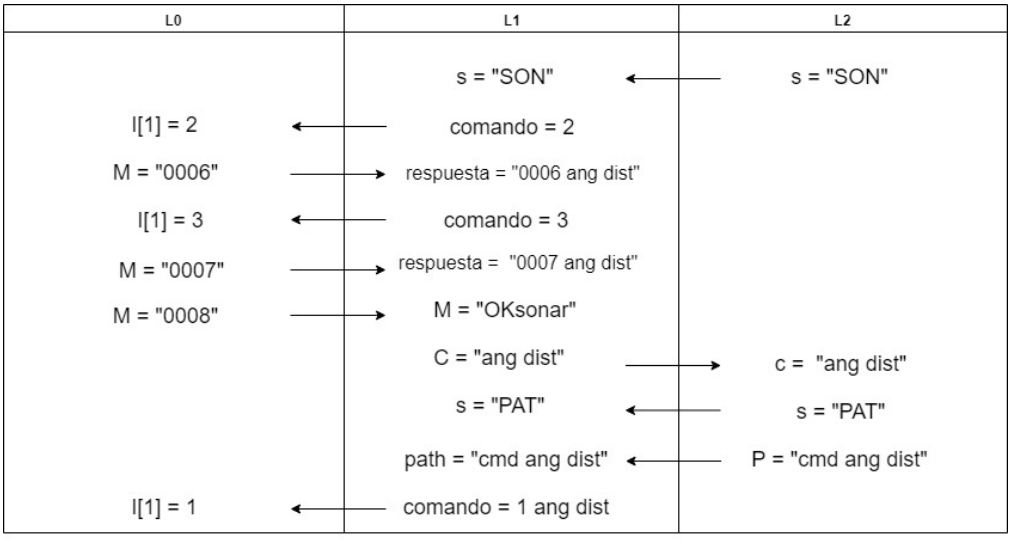

Figure 3: layers communication timeline

5 Experiment Setup

Three different layouts have been tested. We should consider that many factors can influence the robot navigation, such as the density of the objects, its shape, etc. Each layout has its own peculiarities and have been deeply thought to imitate real situations, and they are the following ones:

- Layout 1: In this layout we are trying to imitate a real factory storage room. With a certain pattern it is the best way to store the maximum number of products and allows an efficient organization of them.

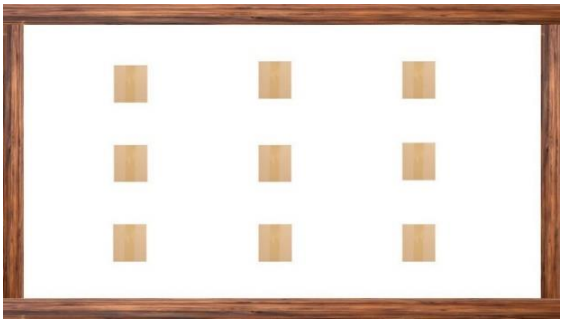

Figure 4: layout 1

- Layout 2: In this layout we are trying to imitate a real factory workspace. The machines are placed in a certain way, trying to maximize the productivity. For this reason, it is impossible to find a certain pattern.

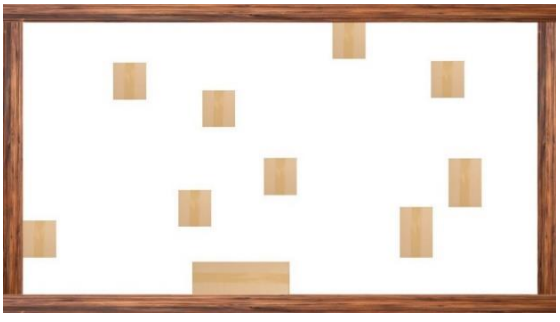

Figure 5: layout 2

- Layout 3: In this layout we are trying to imitate a tricky situation. We are trying to see which the robot behavior is when it has small space to circulate. For example, we can find this situation in the corridor of a company, between the factory room space and the factory workspace.

Figure 6: layout 3

6 Experiment Simulations

Once we had defined the three layouts we are going to use, we started a process to define which was the best explorer algorithm we can use in our case. The original idea was to use the center of mass algorithm, but maybe another algorithm has more benefits.

Tests go from configurations in which the robot goes straight until it finds an obstacle, up to configurations in which the robot analyses the environment and then goes to the virtual centre of mass of a polygon built from the objects detected. All this to discover which is the best technique to follow.

These simulations were run by hand. The reason is that we only wanted some approximations to discard quickly the exploration configurations less effective, and a few centimetres of difference will not invalidate these tests. Furthermore, the robot starts always at the same position and orientation, in the top right pointing down. In this way we make sure that no configuration starts with advantage over other.

As a conclusion of these tests (available on Youtube: https://www.youtube.com/watch?v=sOHaxtaNg20&list=PLvN8o0Ha-R2jESwhRm_Jxgehw4I0fBStI), we have observed how our first idea of using the centre of mass as an algorithm represents exploring a bigger area compared to the other configurations tested. It has been the unique algorithm that has succeed in all three layouts, which is a very important point. For this reason, for now on we are going to be focused on this method and we are not trying other configurations.

7 Implementation

We are using V-REP, a virtual robot experimentation platform (http://www.coppeliarobotics.com/). Working in Linux, Windows and OS, it is open source and based on a distributed control architecture in which each object can be individually controlled using C/C++, Python, Java, Lua, Matlab or Octave. We have used V-REP PRO EDU version 3.4.0, and our programming language is Lua.

Internally the simulation state diagram and the simulation loop are the ones in Figures 7 and 8:

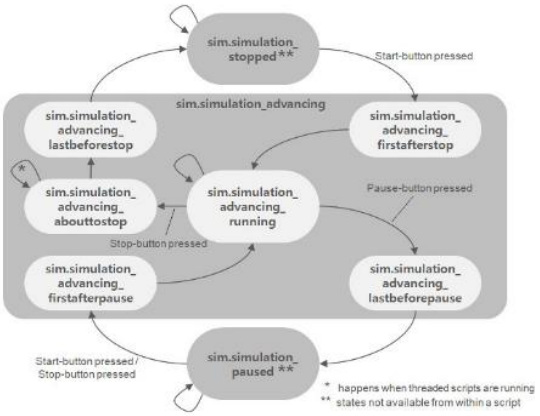

Figure 7: main simulation loop

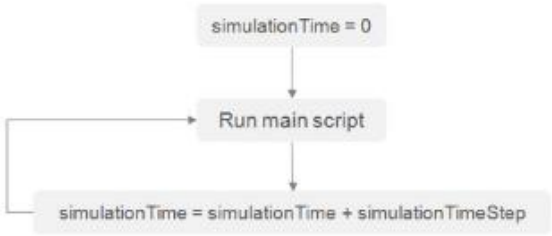

Figure 8: simulation state diagram

The control code is divided into four system call functions that are the next ones:

- Initialization (before simulation): called only when the script is initialized.
- Actuation (during simulation): called when an actuation should happen.
- Sensing (during simulation): called when a sensing should happen.
- Cleanup (after simulation): called only when the simulation is finished.

So, after the initialization the actuation and the sensing are repeated one after one to measure all the variables involved and to do the correct movements. When a movement or movements are done, all the measures are taken again to have all the necessary data updated. When everything is finished, the cleanup function is executed.

We have built a virtual robot with an ultrasound sensor to detect the objects, and two wheels to do the movements. The layouts are the previous one presented, and the center of mass algorithm is implemented.

We are simulating all the previous layouts using V-REP. Thanks to a script we have created a 9x8 matrix where we can put and delete boxes as needed. The plane is 1,8m x 1,6m and every box is 0,2m x 0,2m as Figure 9 indicates.

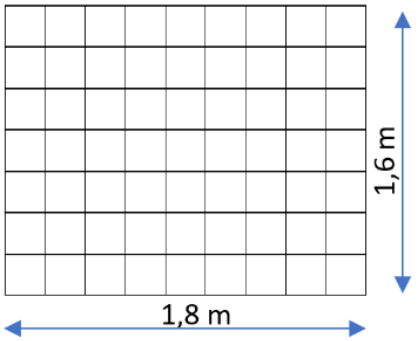

Figure 9: virtual area

The real simulation screen is the following:

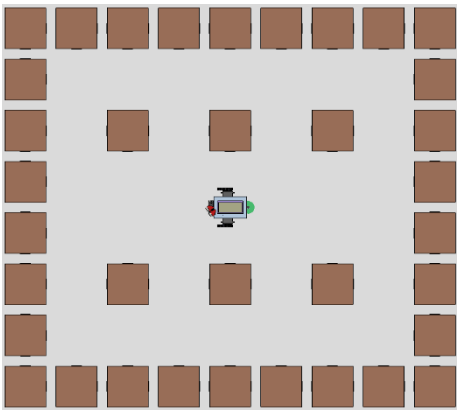

Figure 10: layers levels with Bluetooth

Once the virtual simulation is finished, we are doing the real simulation. In this case, the layers structure is a little bit different: between L0 and L1 we add the L0_L1_bluetooth_link file. This new layer has all the necessary code to connect via Bluetooth the L0 layer (now located on the physical robot) and L1.

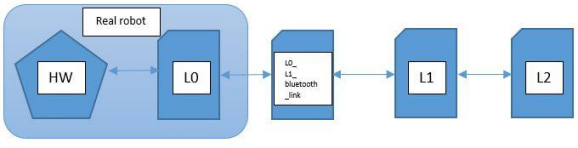

Figure 11: simulation view

8 RESULTS

Once we are arriving to the end of this work, it is important to remember the reason why all the work has been done: we wanted to demonstrate a basic "explorer". For this pourpose we need a robot working with full stack of controllers, an experiment to show the correct robot funcionallity and a video showing all the previous points. This is what we are going to see next.

The first results we are presenting are the ones obtained thanks to a virtual simulation. We have created the three layouts presented in section five in V-REP and we have run using this software the center of mass algorithm. It is convenient to say that the robot has been configured to scan (to calculate the distance to the nearest object) in the next angles and with the next order: {15º, 30º, 45º, 60º, 75º, 90º, 40º, 0º, -15º, -30º, -45º, -60º, -75º, -90º, -40º}.

For the first layout (the regular area) we have confirmed that the robot can explore the center area despite of starting from the outskirts. For this experiment, the safe distance the robot uses to do not hit a box has been reduced from 15cm to 10cm, because in the V-REP layout the boxes are very near one over the other. The video of the experiment can be seen in the following link: https://youtu.be/mXNLMYSfiAk .

For the second layout (the irregular area) we can see how the robot can explore a big area going through narrow gaps. For the robot is not a problem to travel long distances and this random layout seems not to be a blocker for the robot. The video of the experiment can be seen in the following link: https://youtu.be/N7TpP21gv4Y .

For the third layout (the tricky aera) we can see how the robot has escaped of the trap, and that has been able to continue his exploration all over the map. This is quite important because it shows that the robot will not be stuck in narrow places such as the hall of a factory. The video of the experiment can be seen in the following link: https://youtu.be/hAys2fkLKZ4 .

The results obtained in these three layouts confirm that using the center of mass algorithm is a good option to achieve an explorer. Despite different environments, the robot is capable to explore a big area without being stuck or enter a loop.

The second results we are presenting are the ones obtained from the real simulation.

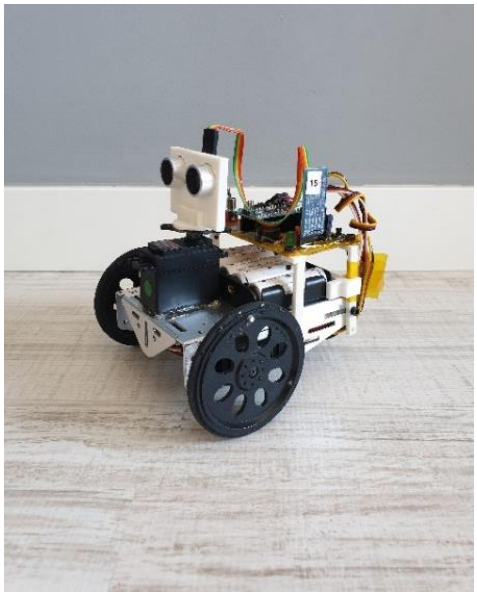

Figure 12: real robot

Before starting these tests, we have verified that the robot works as expected making it do some basic movements. Also, we have tested if the ultrasound sensor was well calibrated and returning realistic data. In Table 1 we have the values the sensor has returned from two executions in the same position:

Table 1: distance sonar returns

Angle	Distance (cm) execution 1	Distance (cm) execution 2	% difference
15º	158	159	0,00633%
30º	204	204	0,00000%
45º	121	124	0,02479%
60º	37	37	0,00000%
75º	36	36	0,00000%
90º	35	36	0,02857%
40º	128	122	0,04688%
0º	237	235	0,00844%
-15º	162	164	0,01235%
-30º	158	143	0,09494%
-45º	130	126	0,03077%
-60º	43	42	0,02326%
-75º	42	42	0,00000%
-90º	42	42	0,00000%
-40º	144	142	0,01389%

As we can see the returned values are very similar between simulations, therefore the percentatge difference between both executions is smaller than 1% in all cases.

We have done one more comprobation before starting with the real test executions. This time the test will be to determine if the sensor is accurate, and to see what happens if the object detected is curved.

Table 2: sonar accuracy

Angle	Distance (cm) sensor returned	Real distance (cm)	% difference
0º	37	37	0,00000%
15º	38	37	0,02632%
30º	39	39	0,00000%
45º	50	50	0,00000%
60º	33	33	0,00000%
75º	33	32	0,03030%
90º	32	32	0,00000%
40º	50	49	0,02000%

As with the previous test, the percentatge difference is smaller than 1% in all cases. So, with these results we can conclude that the ultrasound sensor is accurated and working well, so we can start the tests with confidence in all parts involved.

We have prepared two different tricky layouts: in one of them the boxes are placed in a uniform way forming 90 degrees angles between its different segments. In the other one, these boxes are placed in a way that makes a small tie. It will be interesting to see if the robot can successfully escape from the trap in both cases. The images of the circuits can be seen in Figures 13 and 14:

Figure 13: test one

The videos of the experiment one can be seen in the following links: https://youtu.be/-DgbW6fJaVM and https://youtu.be/0VRWrt91Jdc .

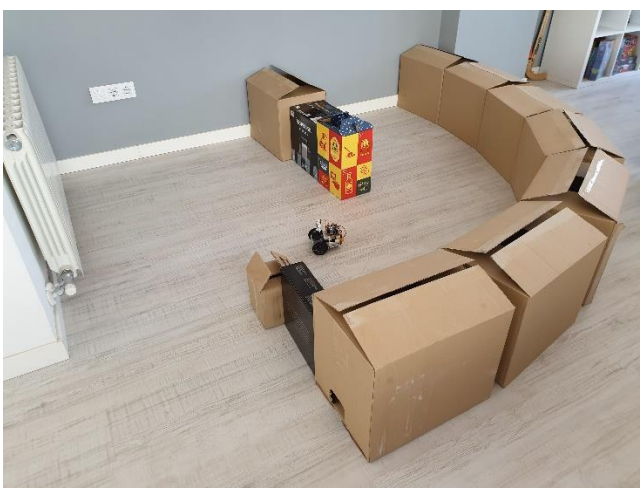

Figure 14: test two

The video of the experiment two can be seen in the following link: https://youtu.be/AeHmo-MlwEc .

In both cases the robot manages to exit the trap in a successful way. The code implemented to do these simulations has worked successfully, and the results are the expected. We can conclude that we have achieved our goals, the robot is working with all the software and the hardware implemented, and we have recorded it. But we want to go further, and we are testing which is the robot behavior in two more situations.

The first one is using a very similar layout as in test one, but with a narrower path. We wanted to test if the robot was able to go through this narrow street, and to continue the exploration for the rest of the circuit. As it can be seen in the video (https://youtu.be/siITMit86XY) the robot manages successfully the situation and the exploration can be completed.

The second one is an execution done in a different type of floor, one in which the joints have relief. As a result, we have seen how the robot changes its direction when it passes over the joint (https://youtu.be/NkQe_oq6yoI).

The conclusions we get from these last two tests are that the robot can manage difficult and narrow situations, but with certain ground conditions.

But one observation must be done: when the ultrasound sensor returns a 0, it means that no object has been detected. In other words, the sensor can detect objects up to 400 centimeters, so if the sensor returns 0 it means that there is no object in at least 4 metres from the sensor. This causes a problem because the algorithm uses the distance to the nearest object as the denominator of a division, and for the mathematics principles a denominator never can be 0. The algorithm fails, and what is more, if a virtual polygon is constructed using a distance 0 instead of a distance bigger than 400 centimeters, of course the polygon and therefore the center of mass will be wrong. So, we have two options: avoid this angle and its respective distance or add to this angle a determined distance. Our solution has been the second one, we are adding a value, but it disagrees with the real distance value, and the mathematics give a result a little bit away from reality.

9 CONCLUSIONS

The main goal of this work was to have a working model of an explorer agent that can be used to map a building or a factory among others.

For this reason, we have succeeded in the elaboration of this work: the robot is able to explore a determined area following the explorer algorithm coded. What is more, not only we have tested this software in a virtual environment, but also in a real one using a real robot.

But it is only the capable amount of work that can be done with the time programmed for that educational purpose. Other interesting purposes an explorer of this kind can be used for are industrial situations, such as for moving materials, tools and products in a factory and for investigate one place not accessible for humans, for example the moon.

This work can be further expanded by creating a visual interface in which the user could see the area the robot is exploring, and the objects detected. Furthermore, the software can also be improved using some memory: if we already have been in one place, the robot can avoid this area and explore a new one. What the hardware refers to, we can use a better ultrasound sensor to get more exact distances to objects and in all conditions (luminics, temperature, 0 gravity, etc). Also, we can use others wheels to be able to navigate in all surfaces, and other materials to achieve a more resistant robot.

But despite the material and time we had, the elaboration of this work has been a very interesting journey. We have learnt about different exploration methods, about state machines and we have discovered some interesting coding tricks. We have worked with a virtual and with a real robot, we have practiced our English and we have enjoyed.

As a conclusion, we are proud of the work done and for the results obtained, but we think that with more time some upgrades to this version could have been done. Anyway, the code is public in our Dropbox folder (https://www.dropbox.com/s/om7irofiuxnhusc/ExplorerCoM%20-%20real%20simulation.ttt?dl=0) and everybody can download it and do some tests, improvements or whatever desired.

ACKNOWLEDGMENT

During the elaboration of this project many people has helped me.

My dad, Francisco Martín Hernández, who recommend me not to study computer engineering (like him), but now I think he is so proud of me.

My mum, Ana María Jiménez López, who gives me all the support and love necessary to carry out all the heavy work of the last 4 years.

My sister, Anna Martín Jiménez, who is the best that has happened to me.

All my family, Victor Martín, Conxi Martín, Jesús Martín, Conxa Hernández, Cristina Cañamares, Encarnación López, Miguel Jiménez, Oscar Ortiz-Villajos, Chesca and Elena. For being there, for encouraging me every day and for helping in all necessary. And with a special mention to my uncle Victor who has helped me with the preparation of this work.

My friends which are also studying computing engineering, for their help and for the good moments lived together during these four years.

My friends which are not studying computer engineering, but with which I have enjoyed very good moments.

My work colleagues in Adevinta, Kimberly Lorenzo, David Lozano, Samanta Cebrian, Iraitz Zabala, Aïda Soler, Luisa Terzulli, for letting me combine my studies with work, and for teaching me everything they know.

My teacher, Lluís Ribas Xirgo, for all his help, his patience, his fun facts and his willingness to embark with me on this project.

My teacher, Joaquín Saiz Alcaine, for introducing me in the robotics world and for the help in the real simulation of this work.

All the teachers that during these four years have taught me everything they knew, fot their kindness in class and in tutorials.

Without all of them, everything would have been different (but worse), so thanks a lot.

For sure I am forgetting someone, so I apologize.

REFERENCES

[1] K. Fuerstenberg, P. Beinschob, C. Reinke, L. Sabattini, V. Digani, E. Cardarelli. Plant Exploration, PAN-ROBOTS: Plug&Navigate robots for smart factories. Executive summary, Oct. 2014. [http://www.pan-robots.eu/wp-content/uploads/2015/08/D5.3-exc-summ.pdf]

[2] Hugh Durrant-Whyte, Fellow, IEEE, and Tim Bailey (2006). Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. Retrieved from https://people.eecs.berkeley.edu/~pabbeel/cs287-fa09/readings/Durrant-Whyte_Bailey_SLAM-tutorial-I.pdf

[3] Lluís Ribas Xirgo (2017). Embedded systems theory slides, Univ. Autònoma de Barcelona. Retrieved from: https://www.dropbox.com/s/1uk84iiaoz3ahjh/Sessi%C3%B3%204.pdf

[4] Joaquín Saiz Alcaine (2017). Embedded systems lab practice slides, Univ. Autònoma de Barcelona. Retrieved from: https://www.dropbox.com/s/lu12ffd7els4soq/P11_CDFG.pdf

APPENDIX

A1. L0 STATES MACHINE

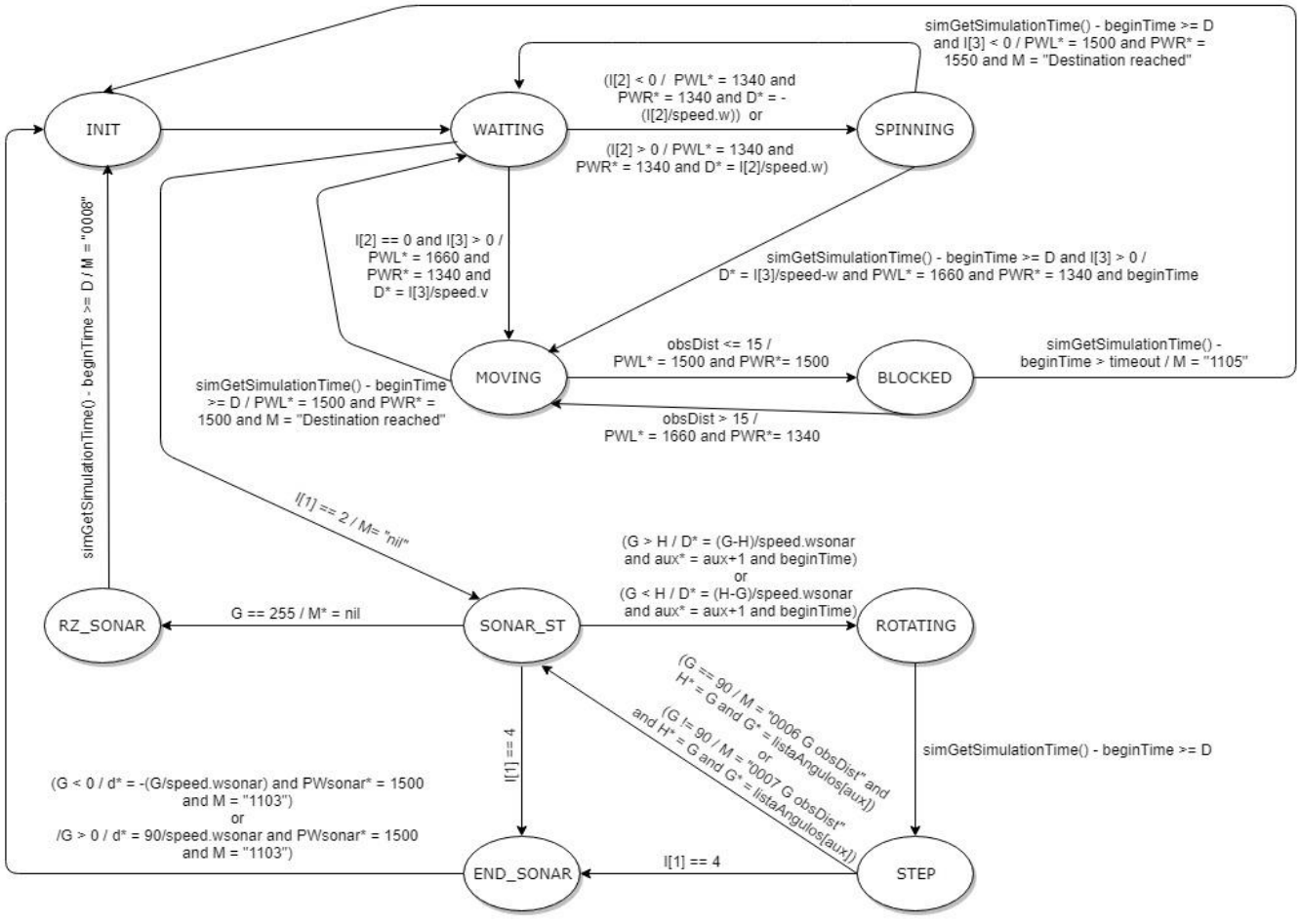

Figure 15: L0 states machine

A2. L1 STATES MACHINE

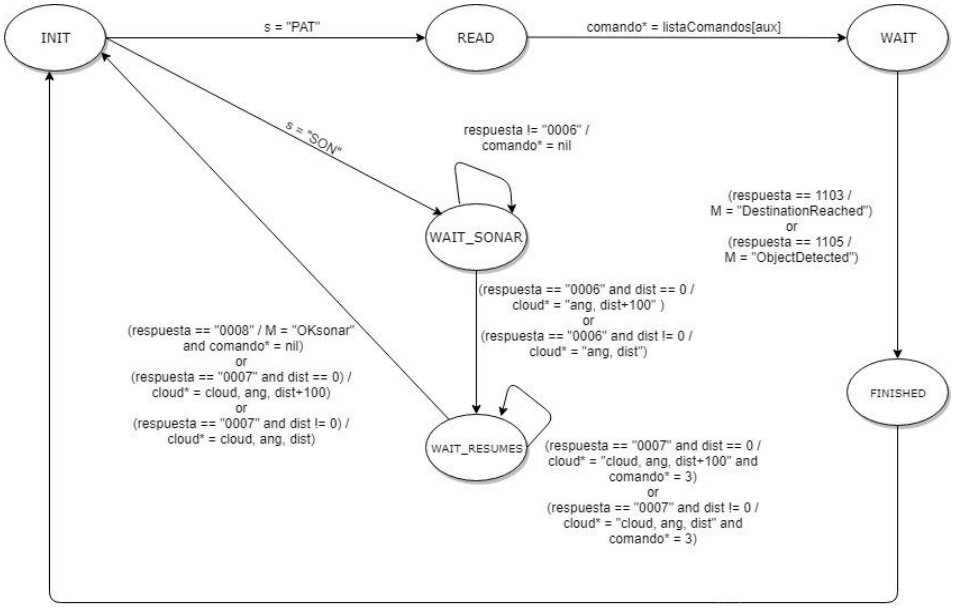

Figure 16: L1 states machine

A3. How To Connect A Windows 10 Computer With The Real Robot Using Bluetooth

1 – Search for "Bluetooth and other devices settings" in the search box.

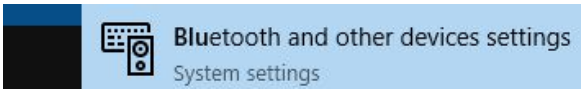

Figure 17: Bluetooth settings icon in the search box

2 – Switch on Bluetooth and click on "Add Bluetooth or other device".

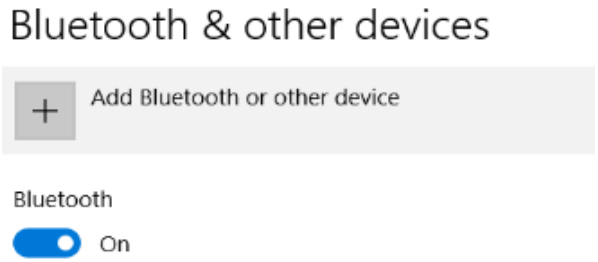

Figure 18: Bluetooth configuration

3 – Click on "Bluetooth".

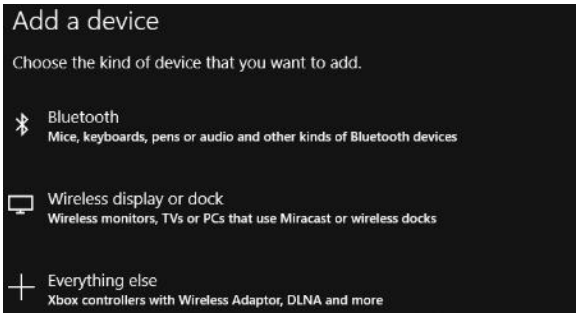

Figure 19: Bluetooth configuration, adding a device

4 – Make sure the robot is on and click on "Servobot_xx", where xx is the number id of the robot.

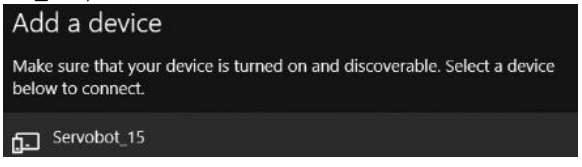

Figure 20: Bluetooth configuration, connecting Servobot

5 – Add the password "00xx", where xx is the number id of the robot.

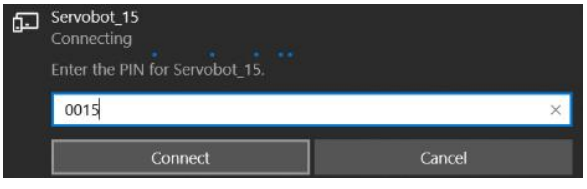

Figure 21: Bluetooth configuration, entering Servobot pin

6 – Search for "Devices and Printers" in the search box.

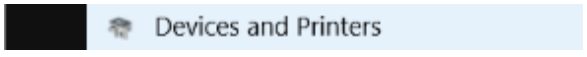

Figure 22: Devices and Printers icon in the search box

7 – Right click on "Servobot_xx" and then click in "Propiedades".

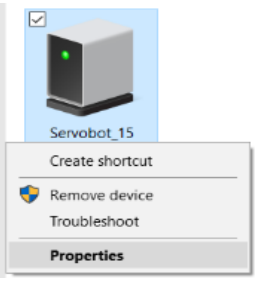

Figure 23: Servobot properties

8 – Click in "Hardware" and copy the port COM number.

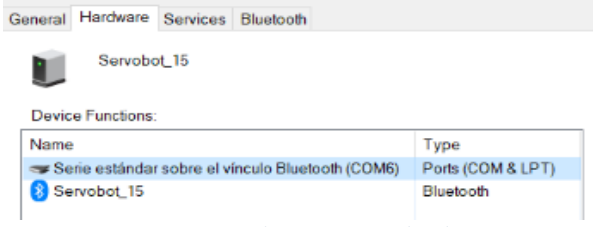

Figure 24: Servobot properties, hardware

9 – In V-REP -> Scene hierarchy, click the right botton in "L0_L1_Bluetooth_Link"

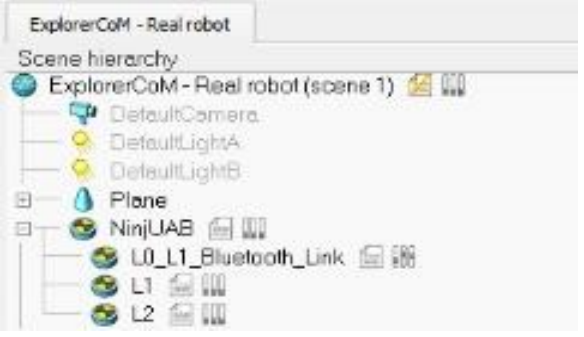

Figure 25: V-REP scene hierarchy menu

10 – In the windows opened click in "serialPortNumber" and paste the port COM value.

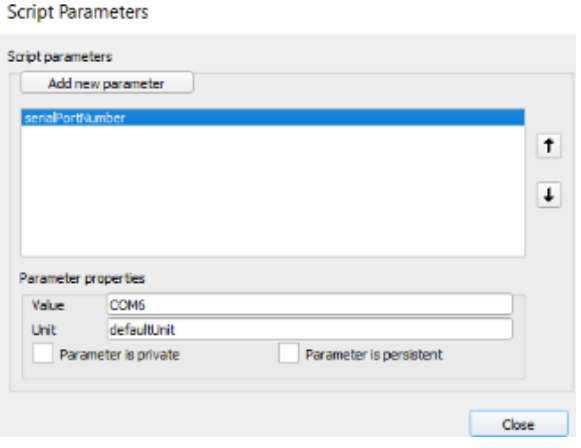

Figure 26: V-REP script parameters menu

11 – The connection has been done successfully!