Embodied reactive agents

UNIVERSITAT JAUME·I

EI1028 Intelligent Systems

David Dag Mora Zapata
Castellón De La Plana, España
al152047@uji.es

Mihaita Alexandru Lupoiu
La Pobla Tornesa, España
al204332@uji.es

***Abstract* -- In this article we are going to explain the basic study about Embodied and Reactive Agents, as well as explaining the different types of tests we've done with the robot Lego Mindstorm NXT. The tests were made in do environments, physical and virtual.**

Keywords—virtual; physical; embodied; reactive; robot, lego

I. INTRODUCTION (*Heading 1*)

The LEGO Mindstorms series of kits contain software and hardware to create small, customizable and programmable robots. They include a programmable *brick* computer that controls the system, a set of modular sensors and motors, and LEGO parts from the Technics line to create the mechanical systems.

LEGO Mindstorms NXT was released by Lego in July 2006, replacing the first-generation LEGO Mindstorms kit. The main component in the kit is a brick-shaped computer called the NXT Intelligent Brick. It has a 32-bit ARM7 processor, 256kb of FLASH memory, and 64kb of RAM.

Very simple programs can be created using the NXT Intelligent Brick itself. In order to create larger, more complex programs, programming software on a PC is required.

RobotC is an Integrated development environment targeted towards students that is used to program and control LEGO NXT, VEX, and RCX robots using a programming language based on the C programming language. It aims to allow code to be ported from one robotics platform to another with little or no change in code. It is used for this project not only for it's simplicity, but also for the option of testing our programs in virtual worlds but selecting one of the challenges that we have. If we want to test our algorithm from the virtual robot in real life, we just have to download the program in the robot and run it.

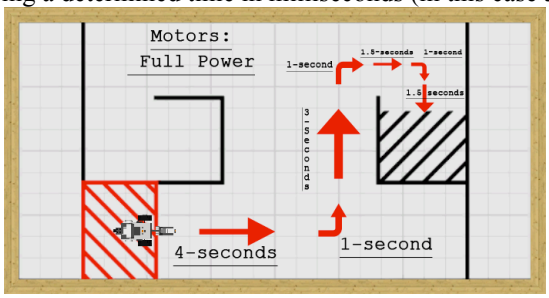
During the lab sessions we mounted and used different types of sensors to complete the task that was proposed. We've been experimenting with different algorithms to perform the simplest tasks, and sometimes made more than two algorithms to perform the same task. The reason we more algorithms was to improve the efficiency of our robot, or by any of the random events that gave us the virtual or physical environment and forced us to adjust the algorithm for more accuracy.

II. THE WORK DONE IN THE IN LABORATORY SESSIONS AND THE EXPERIMENTAL RESULTS

A. *Introduction to Lego Mindstorms NXT and RobotC*

In the first session we used for the first time RobotC. So before we start to write any code, we need to configure our environment and know the basics like for example what structure and syntax uses RobotC, how to compile, and how to use the virtual world.

The first task we had to do was a simple labyrinth, to experience how the robot works. The goal was to move our robot from the red square to the black square, and by doing that we learned how to move the robot. Because the robot has two motors, left and right, we can make the robot go in any direction we want by stopping one motor and the other make it move. But to achieve that purpose we had to also use the wait1Msec(3000) function, that make the robot do the last task during a determined time in milliseconds (in this case 3000).



After a few tests and rectifications, the result we achieved was the one we wanted. The robot arrived to the finish line, but after trying a few times we saw that it didn't always finished. Sometimes, the robot was doing some strange things, and it was because of a random function of the virtual world so it could be more realistic to the real life.

The robot can not interact with the world if we only use motors and the waith1Msec(). So for that purpose we have to mount different types of sensors, that can be:

- Touch Sensor: reacts to touch and release and generates "1" when the sensor is pressed and "0" when not.

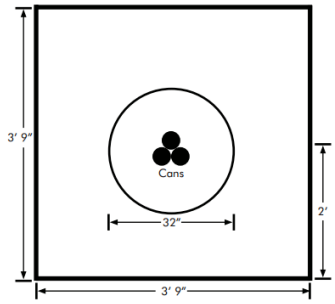
- Light Sensor: it enables are robot to distinguish between light and dark, as well as determine the light intensity in a room or the light intensity of different colors. The values return are between 0 and 100. The darker the environment the more close the value return is to 0, and otherwise.

- Sonar Sensor: detects an object and measure its proximity in inches or centimeters by calculating the time it last in receiving the pulses of sound that has emitted. The value that return is the distance to the object.

- Compass Sensor: measure the earth's magnetic field and calculates a magnetic heading to tell which direction your robot is facing. The compass has a built-in calibration to help reduce magnetic interference from other sources.

In the second and third task we had learned to use some of the sensors, more exactly the sonar and light sensors. We used the sonar sensor to detect the walls of a room at a certain distance, distance that we specified. The robot stops and turns right or left so the robot does not collides. To test the virtual robot, we put it in a rom that was surrounded by walls and let it free. The robot moved randomly around the room, but when it reached a wall, it stopt and avoided the wall. After that, we used the light sensor, that was pointing to the floor to avoid dropping down from a table. The table had a black border so that when the sensor detects the color black, changes que value of the light detected and when it has a certain value that we specificated the robot will go backwards and avoid the edge.

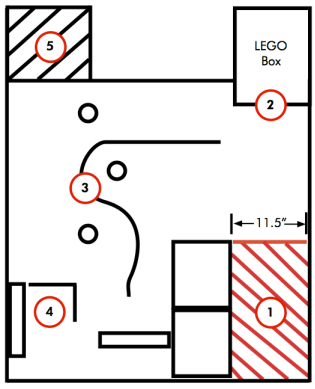
The last task of this session was to use all the sensors and knowledge learned from the other tasks to program the robot to push three full soft drink cans out of a sumo ring. The light sensor must detect the edge of the ring and stop. The ultrasonic sensor allows the robot to detect the cans and push the cans. When we start, the robot starts from an unknown position and must complete the challenge. The problem we had, was the priority of the tasks, so that the robot don't get out of the ring and push the cans.



B. Programming a reactive agent with RobotC Virtual Worlds (I)

In the second session we had develop and combine simple reactive behaviors using sensors, in order to make the robot attain a goal through an obstacle course. The goal was to get the robot to go from the first position to the fifth position, passing thru positions two, three and four. The robot had to go forward until it touches the wall from position two. After that, it goes back a bit so it can start to detect the line, and follow the line a determined time. When the time is over, the robot has to turn 90º right and go to the fourth position until the sonar sensor stops the robot. The last thing to do is to turn 90º right and go to the finish.

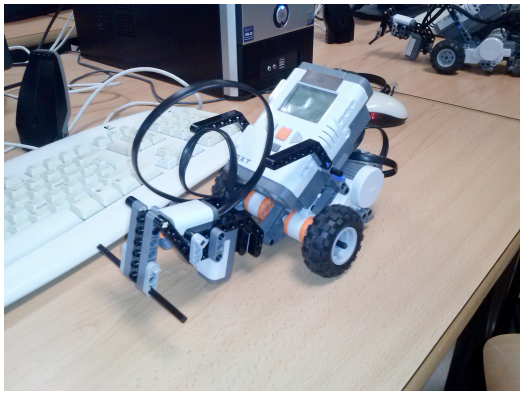
The problems we encountered here were related to the precision when we had to turn, because of the random algorithm of the virtual world and the error that the sensor poses. But at the end the robot finished the challenge a considerate amount of times and decided not rewrite any more code.



C. An embodied agent with Lego Mindstorms NXT (I)

The third session we had to use for the first time the an robot, so we can experience which is the difference between virtual worlds and real robots. Also we learn how to use functions in RobotC, so that if certain circumstances happen satisfying the requirements of the function, it will run at a higher priority than the main program, being able to stop the robot from a fatal error.

The first task we had to do was to build the robot and attach to it the touch and light sensor by following the instructions from the aulavirtual.



The second task, we had to create a algorithm so that the robot could start a random exploration. The robot had to move an random amount of time forward, then turn left or right an indeterminate amount of time. All that had to be done in an infinite loop, so that the robot never stops.

The third task was to add an exception to the robot, and was that if it touches anything, by using the touch sensor, the robot has to stop. Go backwards a bit and turn right or left and retake the exploration algorithm.

The fourth task was the same thing but, this time instead of using a touch sensor we had to use the light sensor. So that when the light sensor detected a black line on the floor, the robot had to do the same instructions from the touch sensor. Go backwards a bit and turn right or left and retake the exploration algorithm.

The last task was to follow a line circuit, by starting over it. The code was not so difficult, the only thing we had to do is calibrate the robot so that it didn't go too slow and pases all the curves and pased a small intersection.

The problems we encountered here were that the values we use to follow the line in the virtual world were not of any use in the real world.

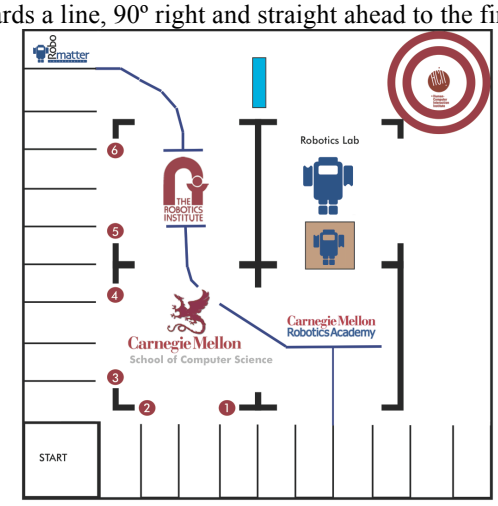
D. Programming a reactive agent with RobotC Virtual Worlds (II)

In the fourth session we had to make a robot capable of doing several tasks in this environment of the task is described in the "RVW Grand Challenge" document. This challenge was modeled after the DARPA Grand Challenge where robots were designed and programmed to travel autonomously through a city. Our scenario consisted of a building with four labs (the Robotics Institute, the School of Computer Science, the Robotics Labm and the Robotics Academy) and two locations (Robomatter Inc. and the Human Computer Interaction Institute).

The goal for this task was to 3 of this list of tasks:

- Travel to Robomatter, wait for one second and return to Start.
- Go to the Human Computer Interaction Institute and then return home.
- Push the robot into the Robotics Lab.
- Go into either entrance of the School of Computer Science without knocking down any balls and then return back to start, returning through any of the other doors.
- Travel to the Robotics Academy doorway and follow the line to the Robotics Institute. Leave the Robotics Institute by traveling through either doorway and returning home.

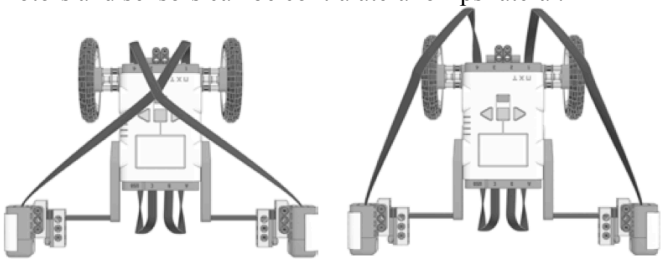
But what we wanted is to use all the sensors, and do all 5 tasks as a on single big task so the first thing we have done was to make the robot go ahead until we got to count 10 lines, after that we go backwards using the same method. When got back to the starting line, turn 90º right using the compass sensor. The robot started to count another time 10 lines. Turn left until que light sensor detected the black line, and in that moment it has to move forward. When the sonar sensor detects that it is too close to the wall it has to stop for 1 second. Then turn 180º with the compass sensor and go down until the sonar sensor stops the robot again and turns 90º right using time approximation. After turning right it counts 4 lines, starts to turn right 90º. The robot moves forward for a determined time, so it can move a box to the robotic lab. After moving the box, the robot gone backwards until it detects a blue line. When detected the blue line, the robot start to follow the and passes through The Robotic Institute and stoping to the Robot Matter. The last thing the robot had to do to do before going to the finish line, was to pass between the balls 5 and 6 without touching them, and get out between balls 1 and 2 the same way. In this case it also turn 90º left, counted 3 lines, because the one we were on doesn't count. Turn left 90º advance a little, turn right 90º, down until we touch the wall, gone backwards a line, 90º right and straight ahead to the finish.



E. An embodied agent with Lego Mindstorms NXT (II)

In the fifth session we had performed a second assembly of a physical robot, to explore their behavior in different environments using two light sensors. The robot used was the same as in session three , the Lego Mindstorm NXT.

The first task was the installation of light sensors on the robot, one for each side. The design is based on the REMbot, Which is somewhat different from the robot of the reference book (insert reference). Thus the sensor attachment is different, but the construction of the light sensors is the same. The sensors are connected to ports 1 and 4. As Shown in the figure, the relationship Between motors and sensors can be contralateral or ipsilateral.



The second task consists in to translate a program written in NXC to the syntax of RobotC the program with three parallel tasks. The main task, which states some definitions and starts the other tasks. The DriveLeft task, which runs the left motor according to the left eye and the DriveRight task, which runs the right motor according to the right eye. The speed of the left and right motor depends directly on the detected light of the respected sensors.

As we had all the components and all development was fairly guided, we didn't had any complications when performing tasks.

The most interesting part of this session was the experiments we made with the finish robot, that were these:

1. The first experiment was to test the behaviour of the robot when the light sensors are connected in contralateral and pointed straight. The result was that the robot was attracted by the light.
2. The first experiment was to test the behaviour of the robot when the light sensors are connected in ipsilateral and pointed straight. The result was that the robot was avoiding the light.
3. The first experiment was to test the behaviour of the robot when the light sensors are connected in contralateral and pointed crossed. The result was that the robot was avoiding the light.
4. The first experiment was to test the behaviour of the robot when the light sensors are connected in ipsilateral and pointed crossed. The result was that the robot was attracted by the light.

The experiment was tested with background light and without it, and the result was the same.

F. General Errors and dificulties

The results we had when we run the programs in the robots, we discovered different problems and some errors. The most common error was when we had to make a turn with some determined values, it always has a problem. One of the problems was the angle, because it never stopped where it should. To correct that, we had to make the robot stop some milliseconds and then make the turn.

We also preferred to use the compass sensor to make turns with greater accuracy. Although we had to take into account a small error in the angle that was made in all the turns.

After several runs, we discovered that the robot had to reduce the speed of the robot so that the light sensor could detect correctly the black line for example. Another problem was that the values in the virtual world did not match the ones in the real world.

III. CONCLUSIONS

After doing this study we realized that how a basic robot works, and how we can create effective and useful code to resolve a simple or more complicated task. Also that the robots are not only objects that we program to do a task repeatedly, but also can interact with the environment with the help of the sensors we can attach to it.

The virtual environment is easier to control, but in the real environment, there are also other factors that affect the the precision and actions of the robot. But in our days almost anything can be done with the help of the intelligent systems.

BIBLIOGRAPHY

[1] http://en.wikipedia.org/wiki/RobotcJ. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[2] http://en.wikipedia.org/wiki/DARPA_Grand_ChallengeK. Elissa, "Title of paper if known," unpublished.

[3] aulavirtual.uji.es

[4] http://mindstorms.lego.com/en-us/products/MS1034.aspx