**Universidade de Aveiro**
**Departamento de Eletrónica, Telecomunicações e Informática**

# Robótica Móvel e Inteligente / Intelligent Mobile Robotics

**(Academic year of 2023-2024)**

# Assignment 2

## Robotic challenge solver using the CiberRato simulation environment

November, 2023

## 1 Objectives

In this assignment each group should develop a robotic agent to command a simulated mobile robot in order to implement a set of robotic tasks, involving different navigation skills.

The list of tasks is the following:

1. Localization: The agent needs to navigate and **localize itself in an unknown maze**, defined by line strips drawn on the floor, using the movement model, the line sensor and the compass. GPS is not available. Motors and compass are noisy. You can consult the `C4-config.xml` file to get the noise parameters. Every cycle that the center of the robot is too far away from strip a penalty is applied.
2. Mapping: The agent needs to explore an unknown maze in order to **extract its map** . At the same time, the agent must localize the target spots placed in the maze. **The number of target spots is available at the beginning, and can change between mazes.** The map generated by the agent must show the identification number of each target spot. After completing the mapping task, the agent **should return to the starting spot**.
3. Planning: The agent needs to **compute a closed path with minimal cost that allows to visit all target spots**, starting and **ending at the starting spot**.

Figure 1 depicts an example of a maze with 3 target spots. Target 0 is always the starting spot.

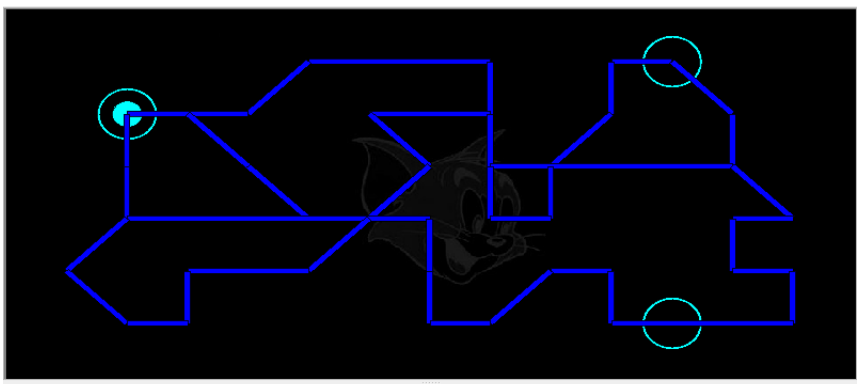


Figure 1: Example of a maze.

## 2 The CiberRato environment

The CiberRato simulation environment will be used to assess the agent developed to overcome the different tasks. The simulated robot (see figure 2) is equipped with 2 motors (left and right) and 3 leds (visiting, returning and finish). In terms of sensors, it includes a compass, four obstacle sensors, a ground sensor, a collision sensor and a line sensor. The available sensors depend on the challenge to be solved.

The simulated robot navigates in a delimited rectangular arena, 14-units tall and 28-units wide, being the diameter of the robot the unit of measure. There is a 1-unit wide not used area around the arena, making the navigable area 12 by 26 units. This navigable area can be seen as a bi-dimensional array of fixed size cells, each cell being a square with side length equal to twice the diameter of the robot (2-units). So, the maximum size of the competing arena is 6-cells tall and 13-cells wide. A maze is defined by putting thin strips connecting the middle of cells (horizontally, vertically or diagonally), which can be detected using the line sensor. The target spots are detectable by the ground sensor.

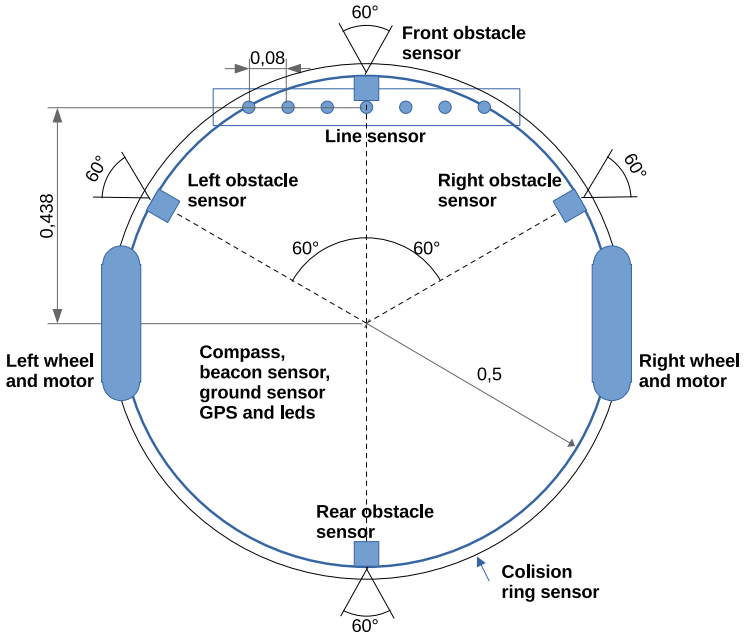



Figure 2: The simulated robot.

The source code of the CiberRato simulation environment can be found at:

`github.com/iris-ua/ciberRatoTools`

Instructions to compile the tools are available at the **README.md** file of the repository.

To start a test of this assignment, execute:

**./startC4**

## 3 Movement model

Consider that the robot's pose is given by $(x, y, \theta)$, where $x$ and $y$ define the robot position and $\theta$ specifies the robot orientation. When the command sent to the simulator at step $t$ is `DriveMotors(`$in_t^l, in_t^r$`)`, then the following equations determine the new robot pose.

An IIR filter is applied to each of the powers provided by the agent ($in_t^l$ and $in_t^r$) that models the inertial characteristics of the motors and generates the effective powers that will be applied

to the motors, corresponding to

$$out_t = \frac{in_i + out_{t-1}}{2} * \mathcal{N}(1, \sigma^2) \tag{1}$$

where $out_t$ is the power applied at time $t$, $out_{t-1}$ the power applied at time $t-1$, and $\mathcal{N}(1, \sigma^2)$ Gaussian noise with mean 1 and standard deviation $\sigma$.

Then, the movement is splitted in a translation of the robot position, considering its current orientation, followed by a the change of the orientation of the robot. For the translation one has

$$lin = \frac{out_t^l + out_t^r}{2} \tag{2}$$

$$x_t = x_{t-1} + lin * cos(\theta_{t-1}) \tag{3}$$

$$y_t = y_{t-1} + lin * sin(\theta_{t-1}) \tag{4}$$

and for the rotation

$$rot = \frac{out_t^r - out_t^l}{D} \tag{5}$$

$$\theta_t = \theta_{t-1} + rot \tag{6}$$

where $D$ is the robot diameter (1 in the CiberRato environment). This provides the new robot pose $(x_t, y_t, \theta_t)$ at the next step, in case no collisions occur. If the new pose involves a collision, the simulator only applies the rotational component.

# 4 Scoring scripts

Each execution of your agent should create a **map file** and a **path file**. These files should have the same base filename and have extensions `.map`, for the map file, and `.path`, for the path file (see `run.sh` for an example on how to create these files).

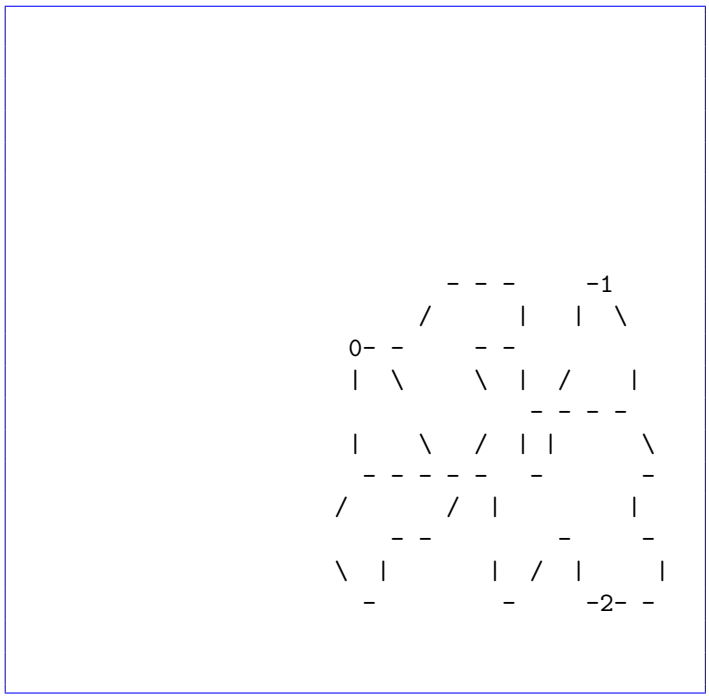
To check if the map file is correct, you can use the `mapping_score.awk` script, executing, in the `simulator` directory:

**gawk -f mapping_score.awk planning.out** ≪***path-to-your-map-file***≫

To get the best possible score, execute:

**gawk -f mapping_score.awk planning.out planning.out**

The map file is a text file that contains 21 lines, each with 49 characters, where the center of the file always represents the starting position, marked as 0, which corresponds to target 0. Additional targets should be marked with their ids. Below you can find an example of a map file (line numbers are not part of the file contents):



To check if the path file is correct, you can use the `planning_score.awk` script, executing, in the `simulator` directory:

**gawk -f planning_score.awk planning.out** ≪***path-to-your-path-file***≫

The path file is a text file that contains, in each line, the $x$ and y coordinates of the center of the cells that are visited by the path. Coordinates are relative to the starting position and are measured in robot diameters. The starting and ending coordinates should always be `0 0`. Below you can find an example of a path file (line numbers are not part of the file contents):

```
0 0
2 0
4 0
6 2
8 2
10 2
12 2
12 0
12 -2
14 -2
16 0
16 2
```

```
18 2
20 0
20 -2
22 -4
20 -4
20 -6
22 -6
22 -8
20 -8
18 -8
16 -8
16 -6
```

```
14 -6
12 -8
10 -8
10 -6
10 -4
8 -4
6 -4
4 -2
2 0
0 0
```

## 5 Assessment

The assessment of this assignment will be composed of 2 components: performance of the agent and a presentation.

- The agents will be tested and graded by teachers, in batch mode, in their own computers. Thus, the format specified for their execution is mandatory. The agent must generate a file representing the map of the maze, generate a file indicating the best path computed and finish its run in the starting spot.
- Each group will make a presentation of its work, consisting of an oral presentation, based

on PDF slides, and a short discussion of the work (maximum of 10 minutes for presentation and 3 minutes for discussion).

# 6 Deliverables and deadline

- Source code of the developed agent.
  - The code should be in a folder called `agent`, regardless of the programming language used.
  - `agent` folder should contain a script file called `build.sh` that allows to build the source code. Even if the code is developed in `Python`, the script should exist (it may do nothing). If you are using non-standard `Python` modules, please use this script to install them in a virtual environment.
  - `agent` folder should contain a script file called `run.sh` that allows to run the agent. This script should accept options `-c`, to define the challenge id (allways 4 in this assignment), `-p`, to set the initial position of the robot in the grid, `-r`, to define the agent name, `-h`, to define the IP of the computer running the simulator and `-f`, to define the basename of map and path files. Example:
    **./run.sh -c 4 -p 0 -r myagent -h 127.0.0.1 -f solution**
    If you set up a virtual environment in `build.sh`, do not forget to activate it in `run.sh`.
- Presentation (in PDF format).

  The presentation should include:
  - Initial slide (names, nmecs, course, assignment, etc.).
  - For each task (localization, mapping, planning) you should present your **approach** for solving each (sub)problem, the experiments carried out and the **results** obtained (use the slides you deem appropriate).
  - Final slide with the conclusions of the work
  - Slide with bibliography/sources used in the work (this slide does not need to be presented).

  All slides should be numbered.

Source code and presentation must be submitted in the Moodle's course page. The following dates apply:

- Source code: December 15$^{\text{th}}$, 2023, by 23:59.
- Presentation: December 20$^{\text{th}}$, 2023, by 23:59.

# 7 Bibliography


- "Principles of Robot Motion: Theory, Algorithms, and Implementations", Howie Choset et al., MIT Press, Boston, 2005.
- "Introduction to Autonomous Mobile Robots", Second Edition, Roland Siegwart et al., MIT Press, 2011.
- "Artificial Intelligence: A Modern Approach", 3rd edition, Russel and Norvig, Pearson, 2009.