ITESM-Campus Querétaro
Ana Laura González Ramírez - A01201959
Gregory Villicaña Hodges - A01062735
Ingeniería en Sistemas Computacionales
2nd/September/2015

Sistemas Inteligentes
Professor Ruben Stranders

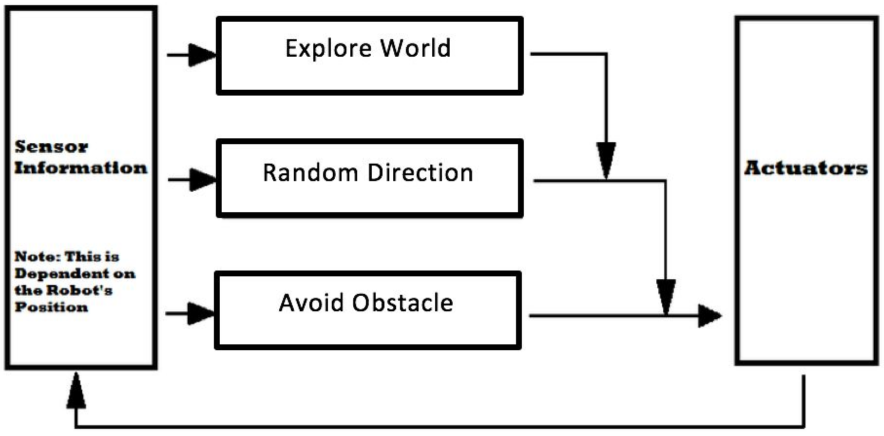
5. Programming Reactive Agents

Lab

Each team will create an architecture for a simple agent and will program its behaviours in a robot.

Basic Architecture

1) Implement a reactive agent that follows these rules:



2) Modify the architecture: you can include other behaviours or change all of them.

Report

1. Describe the behaviours you changed and the reasons why you chose it in 100-200 words.
2. Include your agent final architecture.
3. Based on what you saw in this lab, what are the advantages and disadvantages of reactive agents? Can they achieve complex tasks? (Explain your answer in 100 - 200 words).

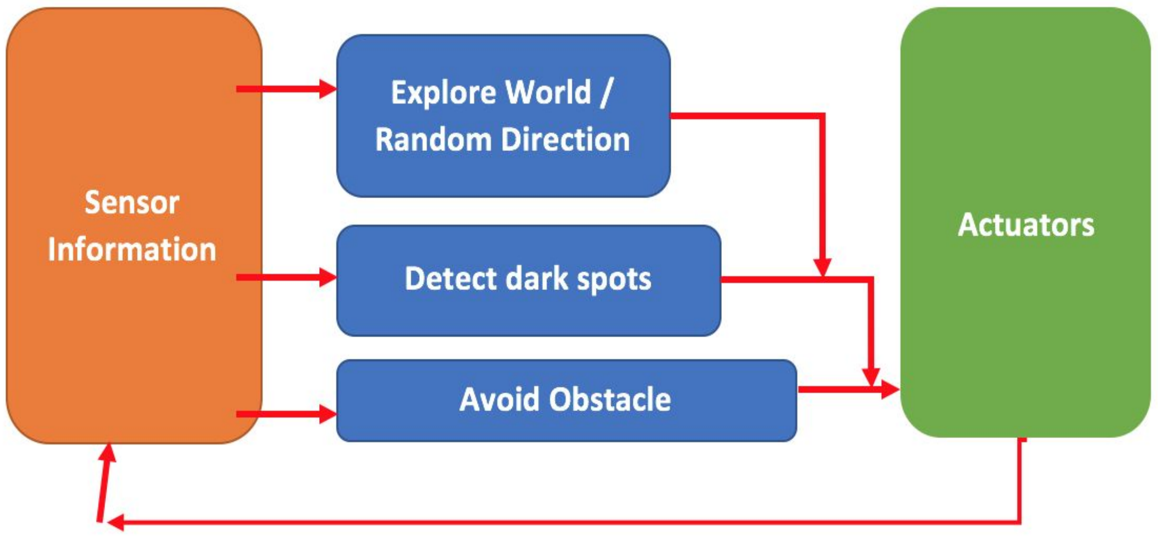
Hands-on work

Our agent uses two sensors in order to perceive the environment: the light sensor (darks or brights) and the object detector sensor (Ultrasonic Sensor). The lowest priority behaviour in our agent is the "***explore behaviour***". When the robot executes this, it moves forward a random amount of seconds and rotates a random angle. We chose this to simulate a more natural movement, rather than just following a straight line path. This is the behaviour with the lowest priority, because we want our agent to follow it on a regular basis, until other behaviours are triggered; after the other behaviours finish, the agent should continue in this state.

We used also the light sensor to trigger a new behaviour called "***detect behaviour***", which will make a sound when it has found a dark spot and then move backwards. We chose this behaviour because we think in the real world something similar but more complex can be applied in an agent to perceive better the environment and report its location to help track specific items.

The last behaviour with the highest priority is "***avoid obstacle***", which will take control once the Ultrasonic Sensor detects something in front the agent.Once this happens, our agent will make a buzz, stop and turn to a random side to avoid the obstacle, and it will remain in this mode until it is obstacle-free and able to move forward. We decided to use the avoiding obstacle behaviour with some improvements (tweaks), because once we develop an effective collision avoidance algorithm, the agent will be able to be more autonomous on the ground. This is something we consider must be one of the most important ones, due to without it navigation would be impossible unless the environment was controlled.

Architecture



Advantages of reactive agents:

1. Easier implementation than a cognitive agent.
2. Easy to make them for simple tasks.
3. Move in fast(real time) responses but not most optimal.
4. Very simple representation of the environment.

Disadvantages of reactive agents:

1. They may not find the most optimal solution to problems.
2. Agents can't do complex representations or consider about alternative plans.
3. Every wanted solution must be coded in advance by programmer, therefore the agent has no autonomy.
4. Complex programs can get incredibly large.
5. They are subject to loops.

Reactive agents are not able to achieve very complex tasks due to its architecture and their autonomy issue. they are designed for simple but useful activities, but they won't be able to tackle very complex issues due to lack of understanding possible changes in external variables, variables that weren't considered at the beginning or were not precise.

Learnings

- We learned how to use and implement the JAVA NXT Subsumption Architecture library, which divides agent's actions into different areas, overall it helped us structure better our code for future challenges and behaviours.
- We learned how to create a more rational agent, by adding more relevant conditions that will help our agent change between behaviours only when it is safe/possible to do so.
- Using the leJOS Subsumption Architecture library made our code a lot more readable and better to manage than our first lab, where we hard coded the behavior priorities with control statements and used a separate thread for each sensor.