

Distributed Intelligent Systems

Lab 2 Tutorial

Ali Marjovi

04.10.2017

Feedback forms

To help us improve, please fill them in.
They are anonymous.



- ◆ Why robotics simulation software ?
 - ◆ Hardware prototyping is time consuming and expensive
 - ◆ Real commercial robots are expensive
 - ◆ Ability to quickly change the experimental set-up
 - ◆ Sometimes easier to measure physical quantities
 - ◆ Sometimes faster than real-time
 - ◆ Numerical optimization methods (GA, PSO, etc.)



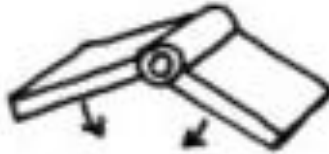
- ◆ Robot prototyping and simulation software
- ◆ Can model practically any type of robot:
 - ◆ Wheeled, legged, flying, swimming, etc.
- ◆ Programming interface to C, C++, Java, Matlab
- ◆ Accelerated OpenGL graphics
- ◆ Physics simulation with Open Dynamics Engine (ODE)

Physics-based simulation (ODE)

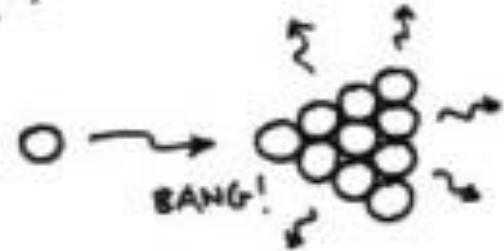
Mechanical systems that have :



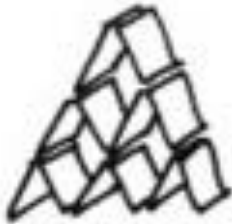
Rigid bodies
(solid objects)



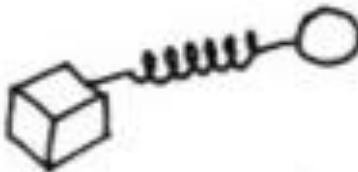
Joints
(like hinges)



Contact and
collisions



Friction
(keeps a tower
of cards steady)



Gadgets
(like springs)



- ◆ Many robot models: Khepera, E-Puck, Aibo, Pioneer 3DX, DARwIn-OP, etc.
- ◆ Sensors: distance sensors, light sensors, cameras, touch sensors, GPSs, force sensors
- ◆ Actuators: servo-motors, grippers, LEDs, connectors, etc.
- ◆ Emitters and receivers (multi-agent systems)
- ◆ And more ...

Using Webots @ Home

- ◆ Available for Linux Ubuntu 12.04, 14.04 and 14.10 (64 bit)
 - ◆ Support for the lab is only guaranteed for the computers in this room
- ◆ Download Webots installation package from
 - ◆ <http://www.cyberbotics.com/linux>
- ◆ See installation instructions on Moodle
- ◆ Windows and Mac versions also available, we cannot offer support

Reminder: Webots GUI

scene tree

world view

editor

The screenshot displays the Webots GUI interface. On the left, the scene tree lists various objects including WorldInfo, Viewpoint, Background, PointLight, DirectionalLight, DEF ground Solid, Solid, and DEF MAZE_WALL_SHORT Solid. The central world view shows a 3D simulation of a robot in a maze environment with a wooden floor and black walls. On the right, the editor window shows the source code for fsm.c, which includes headers, global defines, auxiliary functions, and robot variables. At the bottom, the console window shows the output of the 'make clean' command.

```
File Edit View Simulation Build Robot Tools Wizards Help
E-puck /mnt/z/DIS/14-15/Exercices/Lab02/Code/controllers/fsm/fsm.c
0:00:00:000 - 0.00x
WorldInfo
Viewpoint
Background
PointLight
DirectionalLight
PointLight
DEF ground Solid
Solid
Solid
Solid
DEF MAZE_WALL_SHORT Solid
DEF MAZE_WALL_SHORT Solid
DEF MAZE_WALL_SHORT Solid
DEF MAZE_WALL_SHORT Solid
DEF MAZE_WALL_SHORT Solid
DEF MAZE_WALL_SHORT Solid
Supervisor
DEF E_PUCK_1 EPuck
  translation 0 0 0.15
  rotation 0 1 0 0
  controller "fsm"
  controllerArgs ""
  name "001"
  camera_windowPosition 0 0
  camera_fieldOfView 0.84
  camera_width 52
  camera_height 39
  camera_pixelSize 0
  camera_antialiasing FALSE
fsm.c
1
2
3#include <string.h>
4#include <stdio.h>
5#include <stdlib.h>
6#include <time.h>
7#include <webots/robot.h>
8#include <webots/differential_wheels.h>
9#include <webots/distance_sensor.h>
10#include <webots/light_sensor.h>
11
12 // Global Defines
13
14#define TIME_STEP          64
15
16 // AUXILIARY
17#define NB_SENSORS        8           // number of sensors
18#define BIAS_SPEED        400        // robot bias speed
19#define MAXSPEED          800        // maximum robot speed
20#define DISTRANGE         512
21#define LIGHTRANGE        4095
22
23 // STATES OF FINITE STATE MACHINE
24#define STATE1            1
25#define STATE2            2
26
27 /**
28  ** Auxiliary
29  **
30  **
31  **
32  */
33
34 //robot variables
35 static WbDeviceTag dist_sensors[NB_SENSORS]; // Device variables for
36 static WbDeviceTag light_sensors[NB_SENSORS]; // Device variables for
37 const char *robot_name;
38
39 int state; // current state of the robot in the Finite State Machine
40
41 //AUXILIARY
42
43 int ds[NB_SENSORS]; // distance sensor readings stored here
```

Console
make clean
done.

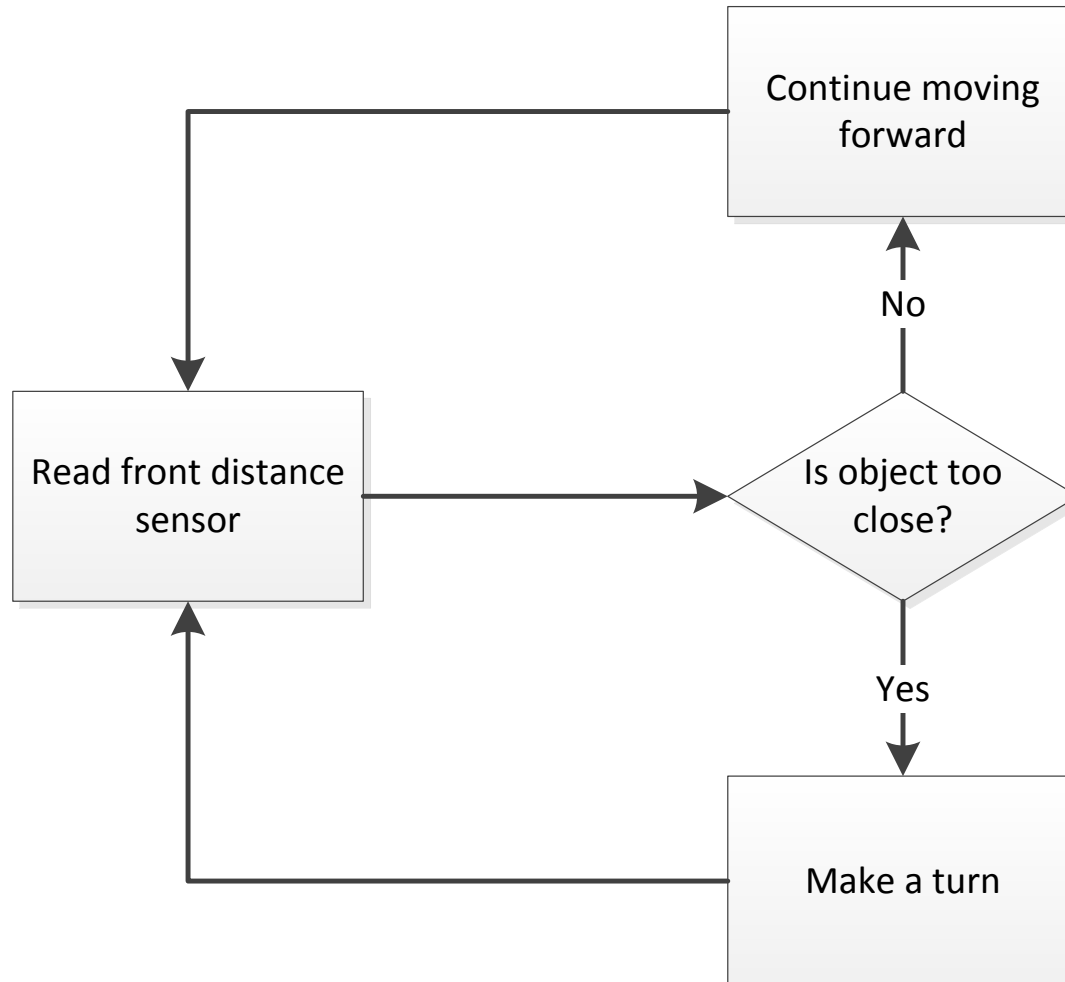
console

e-puck robot

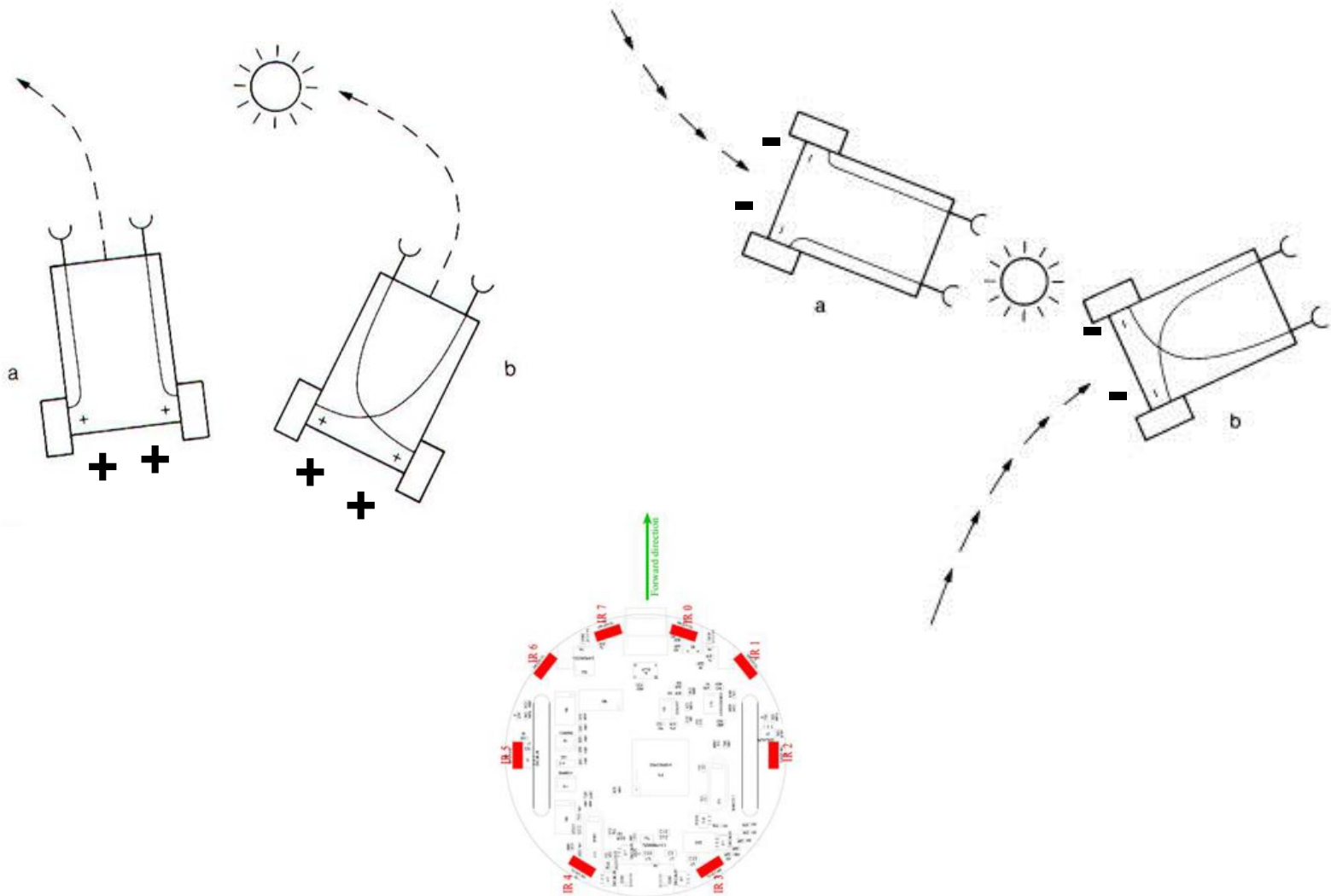
- 8 proximity sensors
- 8 light sensors
- 1 color camera
- 3 microphones
- 1 speaker
- 3 axis accelerometers
- and more ...



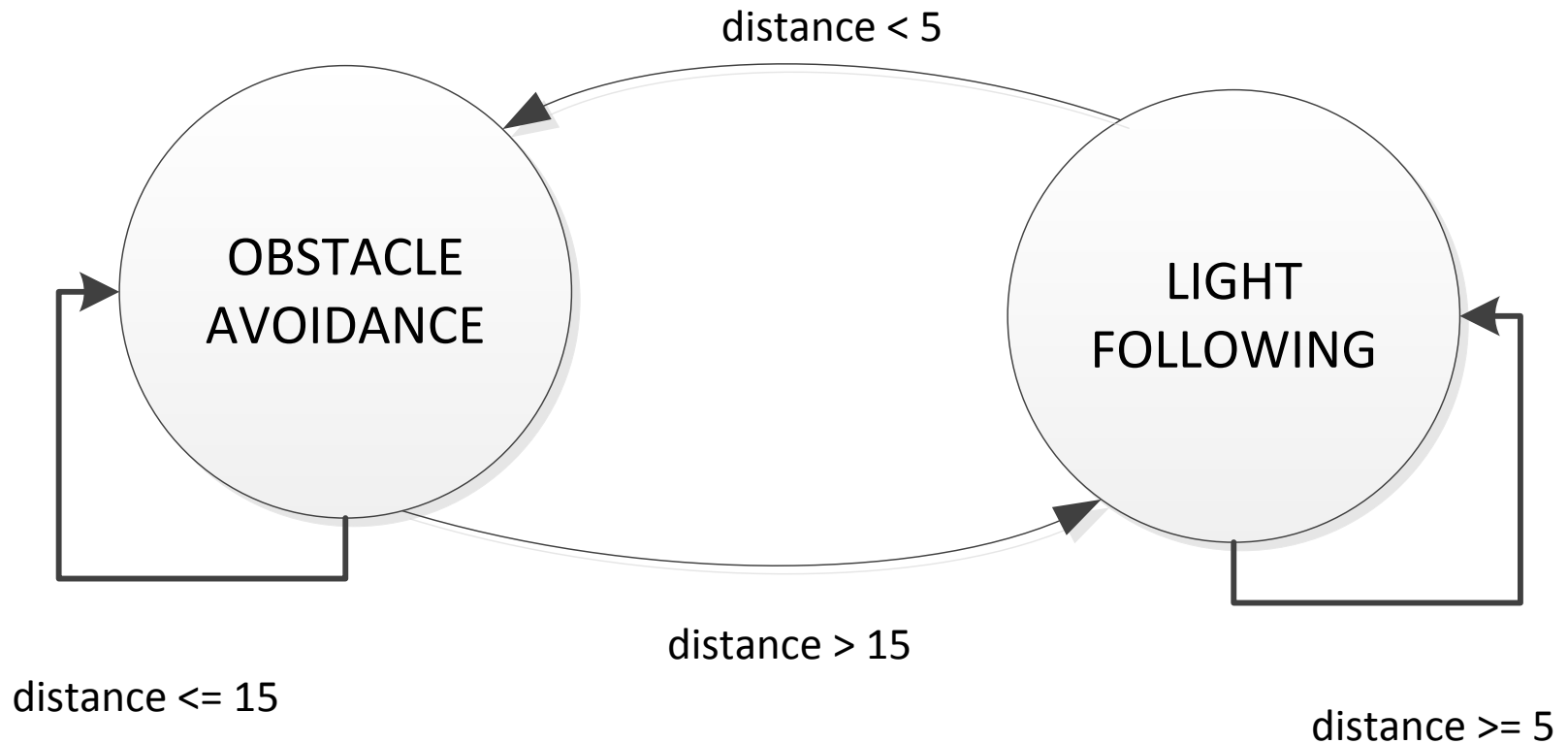
Robot control – Rule based



Robot control – Braitenberg



Finite State Machine (FSM)



And now: let's start ...

