

# Distributed Intelligent Systems

## Lab 2 Tutorial

Cyrill Baumann

02.10.2019

# 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.)



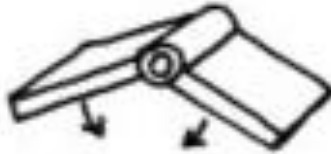
- ◆ Robot prototyping and simulation software
- ◆ Can model practically any type of robot:
  - ◆ Wheeled, legged, flying, swimming, etc.
- ◆ Programming interface to C, C++, Java, Matlab
- ◆ Accelerated OpenGL graphics
- ◆ Physics simulation with Open Dynamics Engine (ODE)

# Physics-based simulation (ODE)

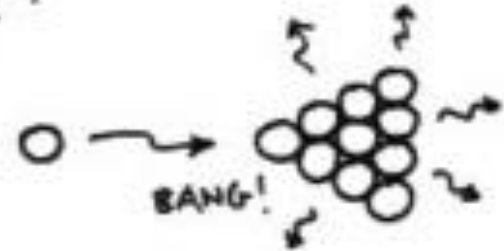
Mechanical systems that have :



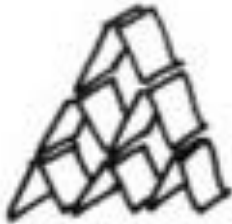
Rigid bodies  
(solid objects)



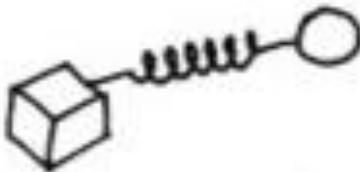
Joints  
(like hinges)



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 16.04, and 18,04.
  - ◆ Support for the lab is only guaranteed for the computers in this room
- ◆ Download Webots installation package from
  - ◆ <http://www.cyberbotics.com/>
  - ◆ 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.

console

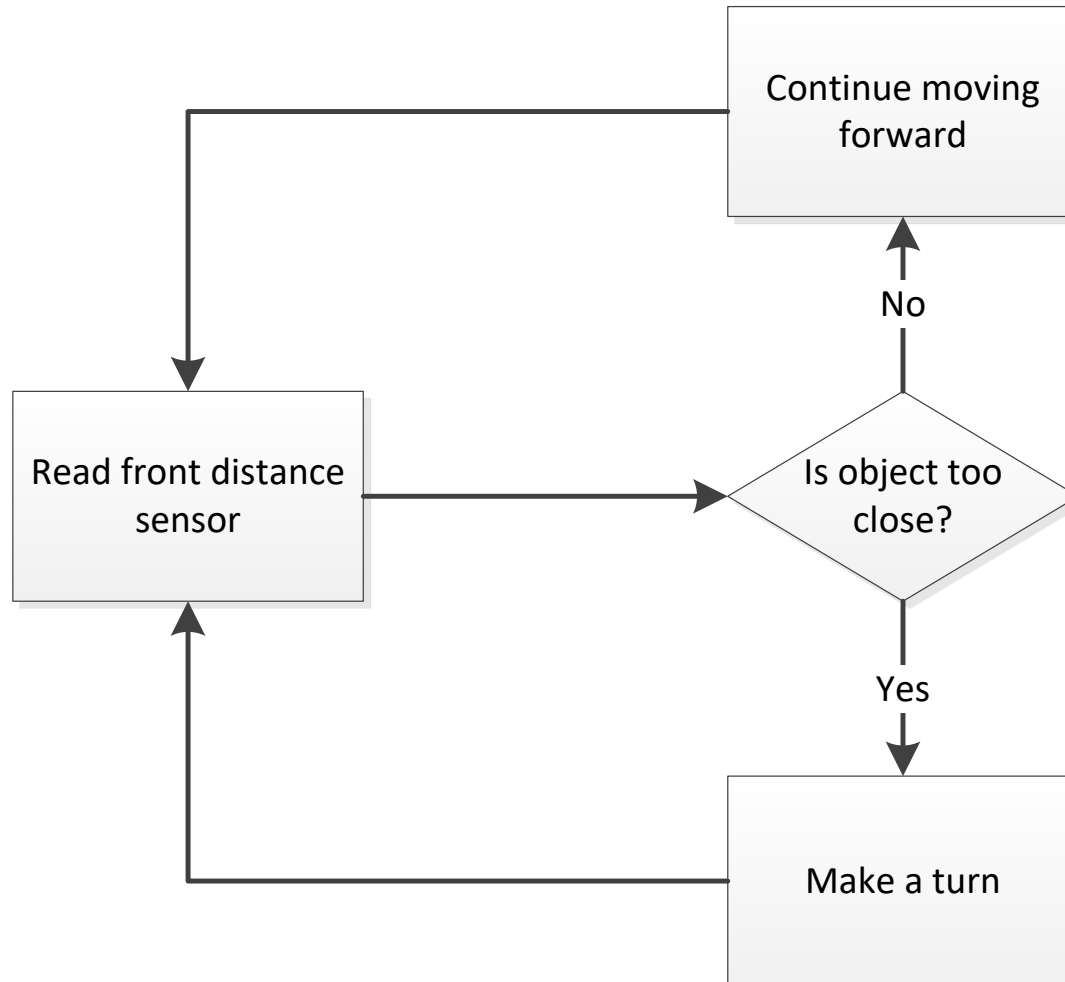


# e-puck robot

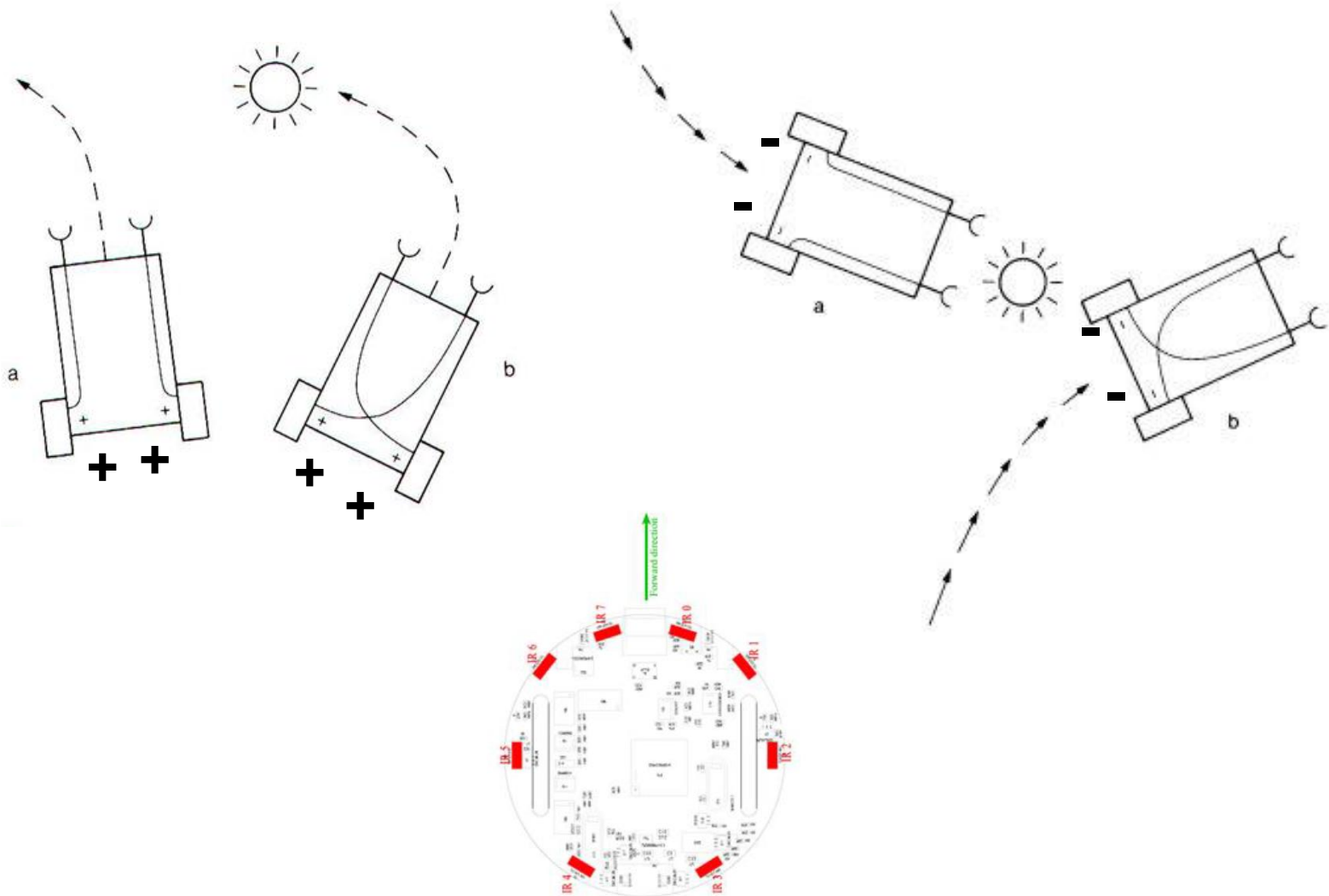
- 8 proximity sensors
- 8 light sensors
- 1 color camera
- 3 microphones
- 1 speaker
- 3 axis accelerometers
- and more ...



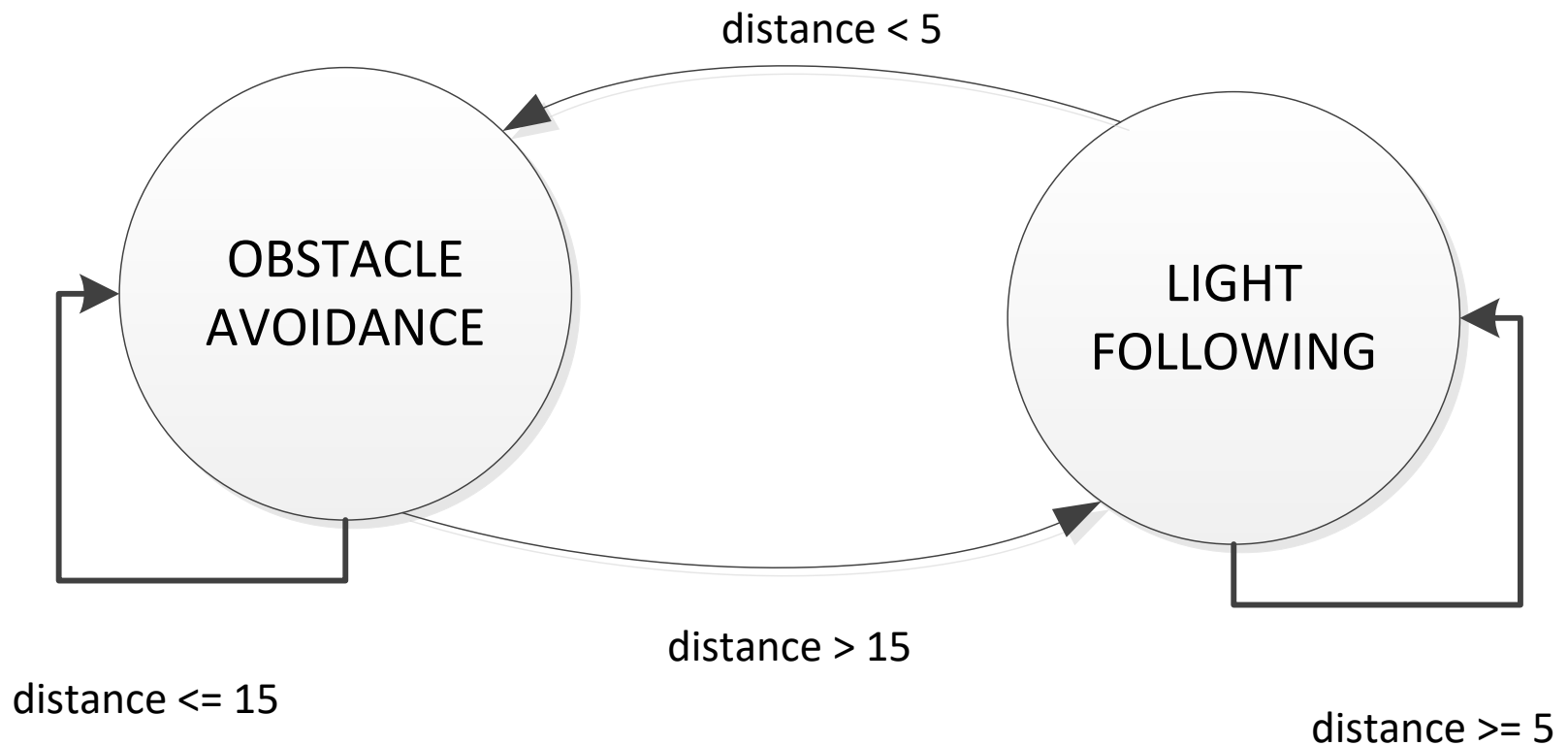
# Robot control – Rule based



# Robot control – Braitenberg



# Finite State Machine (FSM)



# And now: let's start ...

