

Cyberbotics' Robot Curriculum

by Wikibooks contributors



*Created on Wikibooks,
the open content textbooks collection.*

Copyright © 2009 Wikibooks contributors.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

Contents

1	About this book	5
	Further reading	6
2	What is Artificial Intelligence?	7
	GOFAI versus New AI	7
	History	8
	The Turing test	9
	Cognitive Benchmarks	13
	Further reading	13
3	What are Robots?	15
	Robots in our every Day's Life	15
	Robots as Artificial Animals	15
4	E-puck and Webots	19
	E-puck	19
	Webots	21
5	Cyberbotics' Robot Curriculum/Getting started	27
	Explanations about the Practical Part	27
	Get Webots and install it	28
	Bluetooth Installation and Configuration	28
	Open Webots	31
	E-puck Prerequisites	35
6	Cyberbotics' Robot Curriculum/Beginner programming Exercises	39
	Discovery of the e-puck [Beginner]	39
	Robot Controller [Beginner]	41
	Move your e-puck [Beginner]	45
	Simple Behavior: Finite State Machine (FSM) [Beginner]	47
	Better Collision avoidance Algorithm [Beginner]	51
	The blinking e-puck [Beginner]	52
	E-puck Dance [Beginner]	53
	Line following [Beginner]	53

Rally [Beginner] [Challenge]	55
A Document Information	57
History	57
PDF Information & History	57
Authors	57
B GNU Free Documentation License	59
1. APPLICABILITY AND DEFINITIONS	59
2. VERBATIM COPYING	61
3. COPYING IN QUANTITY	61
4. MODIFICATIONS	61
5. COMBINING DOCUMENTS	63
6. COLLECTIONS OF DOCUMENTS	63
7. AGGREGATION WITH INDEPENDENT WORKS	64
8. TRANSLATION	64
9. TERMINATION	64
10. FUTURE REVISIONS OF THIS LICENSE	64
ADDENDUM: How to use this License for your documents	65

Chapter 1

About this book

Learning about Intelligent Robots

This book is intended to students, teachers, hobbyists and researchers interested in intelligent robots. It will help you understanding what robots are, what they can do for you, and most interestingly how to program them. It includes two parts: a short theoretical part and a longer practical part. Practical part is decomposed in one chapter about the computer configuration and five chapters of exercises corresponding to five level of difficulty (see the next section). After reading this book, you should be able to design your own intelligent robots.

From Beginners to Robotics Experts

Even if you never wrote a computer program before, you will learn easily how to graphically program the behavior of a simple robot. From this first experience, you will be smoothly introduced to higher level computer programming and discover more possibilities of intelligent robots. This practical investigation is organized in projects for which a difficulty level is associated. You are free to stop at any level if the projects suddenly become too difficult to handle, but if you reach the latest levels successfully, you should consider yourself as a genuine robotics researcher! Here are the levels of difficulty:

- beginner: no prior knowledge needed, suitable for children from 8 years old and people without a scientific background (see [Beginner programming Exercises](#))
- novice: scientific or technological interest needed, suitable for children from 8 years old (see [Novice programming Exercises](#))
- intermediate: general computer science background needed, intended to student from 12 years old with some interest in computer science (see [Intermediate programming Exercises](#))
- advanced: programming skills needed, intended to post-graduate students and researchers (see [Advanced Programming Exercises](#))
- expert: research spirit needed, intended to post-graduate student and researchers (see [Cognitive Benchmarks](#))

Important: The code to which we refer in the exercises is freely available at sourceforge.net. You can download it directly from the SVN at this address:

`http://robotcurriculum.svn.sourceforge.net/svnroot/robotcurriculum`

Easy-to-use robotics Tools

The practical part of this book relies on a couple of software and hardware tools that will allow you to practice intelligent robot programming for real. These tools are the e-puck robot and the Webots software. They are both widely used for education and research in Universities worldwide and are commercially available and well supported. These tools will be described in chapter [E-puck and Webots](#).

Enjoy Robot Competitions

Several exercises are provided along this book. Starting from very simple introductory exercises in chapter [Beginner programming Exercises](#), the reader will learn progressively how to create more and more advanced robotics controllers throughout the following chapters. Finally, the chapter [Cognitive Benchmarks](#) will introduce the reader into the realm of robot competitions through a cognitive benchmark: Rat's Life ¹.

Further reading

- [Cyberbotics Official Webpage](#)
- [E-puck website](#)

¹See their website, [Rat's Life Programming Contest](#)

Chapter 2

What is Artificial Intelligence?

Artificial Intelligence (AI) is an interdisciplinary field of study that includes computer science, engineering, philosophy and psychology. There is no widely accepted precise definition of Artificial Intelligence, because Intelligence is very difficult to define. John McCarthy defined Artificial Intelligence as “the science and engineering of making intelligent machine”¹ which does not explain what intelligent machines are. Hence, it does not help either to answer the question “Is a chess playing program an intelligent machine?”.

GOFAI versus New AI

AI divides roughly into two schools of thought: GOFAI (Good Old Fashioned Artificial Intelligence) and New AI. GOFAI mostly involves methods now classified as machine learning, characterized by formalism and statistical analysis. This is also known as conventional AI, symbolic AI, logical AI or neat AI. Methods include:

- *Expert Systems* apply reasoning capabilities to reach a conclusion. An Expert System can process large amounts of known information and provide conclusions based on them.
- *Case Based Reasoning* stores a set of problems and answers in an organized data structure called cases. A Case Based Reasoning system upon being presented with a problem finds a case in its knowledge base that is most closely related to the new problem and presents its solutions as an output with suitable modifications.
- *Bayesian Networks* are probabilistic graphical models that represent a set of variables and their probabilistic dependencies.
- *Behavior Based AI* is a modular method building AI systems by hand.

New AI involves iterative development or learning. It is often bio-inspired and provides models of biological intelligence, like the Artificial Neural Networks. Learning is based on empirical data and is associated with non-symbolic AI. Methods mainly include:

¹ See [John McCarthy, What is Artificial Intelligence?](#)

- *Artificial Neural Networks* are bio-inspired systems with very strong pattern recognition capabilities.
- *Fuzzy Systems* are techniques for reasoning under uncertainty; they have been widely used in modern industrial and consumer product control systems.
- *Evolutionary computation* applies biologically inspired concepts such as populations, mutation and survival of the fittest to generate increasingly better solutions to a problem. These methods most notably divide into *Evolutionary Algorithms* (including *Genetic Algorithms*) and *Swarm Intelligence* (including *Ant Algorithms*).

Hybrid Intelligent Systems attempt to combine these two groups. Expert Inference Rules can be generated through Artificial Neural Network or Production Rules from Statistical Learning.

History

Early in the 17th century, René Descartes envisioned the bodies of animals as complex but reducible machines, thus formulating the mechanistic theory, also known as the “clockwork paradigm”. Wilhelm Schickard created the first mechanical digital calculating machine in 1623, followed by machines of Blaise Pascal (1643) and Gottfried Wilhelm von Leibniz (1671), who also invented the binary system. In the 19th century, Charles Babbage and Ada Lovelace worked on programmable mechanical calculating machines.

Bertrand Russell and Alfred North Whitehead published Principia Mathematica in 1910-1913, which revolutionized formal logic. In 1931 Kurt Gödel showed that sufficiently powerful consistent formal systems contain true theorems unprovable by any theorem-proving AI that is systematically deriving all possible theorems from the axioms. In 1941 Konrad Zuse built the first working mechanical program-controlled computers. Warren McCulloch and Walter Pitts published A Logical Calculus of the Ideas Immanent in Nervous Activity (1943), laying the foundations for neural networks. Norbert Wiener’s Cybernetics or Control and Communication in the Animal and the Machine (MIT Press, 1948) popularized the term “cybernetics”.

Game theory which would prove invaluable in the progress of AI was introduced with the paper, Theory of Games and Economic Behavior by mathematician John von Neumann and economist Oskar Morgenstern ².

1950’s

The 1950s were a period of active efforts in AI. In 1950, Alan Turing introduced the “Turing test” as a way of creating a test of intelligent behavior. The first working AI programs were written in 1951 to run on the Ferranti Mark I machine of the University of Manchester: a checkers-playing program written by Christopher Strachey and a chess-playing program written by Dietrich Prinz. John McCarthy coined the term “artificial intelligence” at the first conference devoted to the subject, in 1956. He also invented the Lisp programming language. Joseph Weizenbaum built ELIZA, a chatter-bot implementing Rogerian psychotherapy. The birth date of AI is generally considered to be July 1956 at the Dartmouth Conference, where many of these people met and exchanged ideas.

²Von Neumann, J.; Morgenstern, O. (1953), “Theory of Games and Economic Behavior”, New York

1960s-1970s

During the 1960s and 1970s, Joel Moses demonstrated the power of symbolic reasoning for integration problems in the Maccsma program, the first successful knowledge-based program in mathematics. Leonard Uhr and Charles Vossler published “A Pattern Recognition Program That Generates, Evaluates, and Adjusts Its Own Operators” in 1963, which described one of the first machine learning programs that could adaptively acquire and modify features and thereby overcome the limitations of simple perceptrons of Rosenblatt. Marvin Minsky and Seymour Papert published *Perceptrons*, which demonstrated the limits of simple Artificial Neural Networks. Alain Colmerauer developed the Prolog computer language. Ted Shortliffe demonstrated the power of rule-based systems for knowledge representation and inference in medical diagnosis and therapy in what is sometimes called the first expert system. Hans Moravec developed the first computer-controlled vehicle to autonomously negotiate cluttered obstacle courses.

1980s

In the 1980s, Artificial Neural Networks became widely used due to the back-propagation algorithm, first described by Paul Werbos in 1974. The team of Ernst Dickmanns built the first robot cars, driving up to 55 mph on empty streets.

1990s & Turn of the Millennium

The 1990s marked major achievements in many areas of AI and demonstrations of various applications. In 1995, one of Ernst Dickmanns’ robot cars drove more than 1000 miles in traffic at up to 110 mph, tracking and passing other cars (simultaneously Dean Pomerleau of Carnegie Mellon tested a semi-autonomous car with human-controlled throttle and brakes). Deep Blue, a chess-playing computer, beat Garry Kasparov in a famous six-game match in 1997. Honda built the first prototypes of humanoid robots (see picture of the Asimo Robot).

During the 1990s and 2000s AI has become very influenced by probability theory and statistics. Bayesian networks are the focus of this movement, providing links to more rigorous topics in statistics and engineering such as Markov models and Kalman filters, and bridging the divide between GOFAI and New AI. This new school of AI is sometimes called ‘machine learning’. The last few years have also seen a big interest in game theory applied to AI decision making.

The Turing test

Artificial Intelligence is implemented in machines (i.e., computers or robots), that are observed by ”Natural Intelligence” beings (i.e., humans). These human beings are questioning whether or not these machines are intelligent. To give an answer to this question, they evidently compare the behavior of the machine to the behavior of another intelligent being they know. If both are similar, then, they can conclude that the machine appears to be intelligent.

Alan Turing developed a very interesting test that allows the observer to formally say whether or not a machine is intelligent. To understand this test, it is first necessary to understand that intelligence, just like beauty, is a concept relative to an observer. There is no absolute intelligence, like there is no absolute beauty. Hence it is not correct to say that a machine is more or less intelligent. Rather, we should say that a machine is more or less intelligent for a given observer.

See in Wikipedia: [Turing test](#)



Figure 2.1: Asimo: Honda's humanoid robot

Starting from this point of view, the Turing test makes it possible to evaluate whether or not a machine qualifies for artificial intelligence relatively to an observer.

The test consists in a simple setup where the observer is facing a machine. The machine could be a computer or a robot, it does not matter. The machine however, should have the possibility to be remote controlled by a human being (the remote controller) which is not visible by the observer. The remote controller may be in another room than the observer. He should be able to communicate with the observer through the machine, using the available inputs and outputs of the machine. In the case of a computer, the inputs and outputs may be a keyboard, a mouse and computer screen. In the case of a robot, it may be a camera, a speaker (with synthetic voice), a microphone, motors, etc. The observer doesn't know if the machine is remote controlled by someone else or if it behaves on its own. He has to guess it. Hence, he will interact with the machine, for example by chatting using the keyboard and the screen to try to understand whether or not there is a human intelligence behind this machine writing the answers to his questions. Hence he will want to ask very complicated questions and see what the machine answers and try to determine if the answers are generated by an AI program or if they come from a real human being. If the observer believes he is interacting with a human being while he is actually interacting with a computer program, then this means the machine is intelligent for him. He was bluffed by the machine. The table below summarizes all the possible results coming out of a Turing test.

The Turing test helps a lot to answer the question "can we build intelligent machines?". It demonstrates that some machines are indeed already intelligent for some people. Although these people are currently a minority, including mostly children but also adults, this minority is growing as AI programs improve.

Although the original Turing test is often described as a computer chat session (see picture), the interaction between the observer and the machine may take very various forms, including a chess game, playing a virtual reality video game, interacting with a mobile robot, etc.

	The machine is remote controlled by a human	The machine runs an Artificial Intelligence program
The observer believes he faces a human intelligence	undetermined: the observer is good at recognizing human intelligence	successful: the machine is intelligent for this observer
The observer believes he faces a computer program	undetermined: the observer has troubles recognizing human intelligence	failed: the machine is not intelligent for this observer

Table 2.1: All possible outcomes for a Turing test

Similar experiments involve children observing two mobile robots performing a prey predator game and describing what is happening. Unlike adults who will generally say that the robots were programmed in some way to perform this behavior, possibly mentioning the sensors, actuators and micro-processor of the robot, the children will describe the behavior of the robots using the same words they would use to describe the behavior of a cat running after a mouse. They will grant feelings to the robots like "he is afraid of", "he is angry", "he is excited", "he is quiet", "he wants to...", etc. This leads us to think that for a child, there is little difference between the intelligence of such robots and animal intelligence.

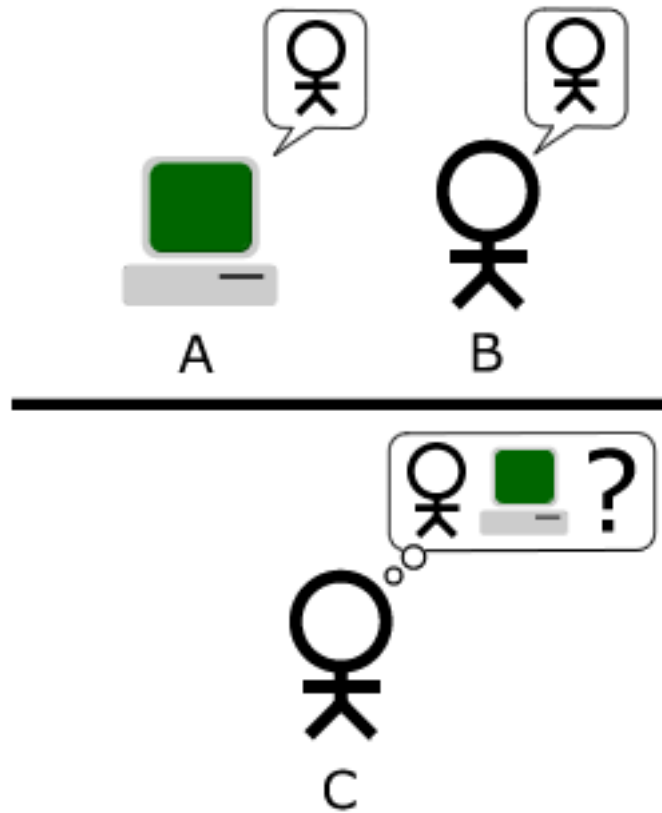


Figure 2.2: The Turing test

Cognitive Benchmarks

Another way to measure whether or not a machine is intelligent is to establish cognitive (or intelligence) benchmarks. A benchmark is a problem definition associated with a performance metrics allowing evaluating the performance of a system. For example in the car industry, some benchmarks measure the time necessary for a car to accelerate from 0 km/h to 100 km/h. Cognitive benchmarks address problems where intelligence is necessary to achieve a good performance.

Again, since intelligence is relative to an observer, the cognitive aspect of a benchmark is also relative to an observer. For example if a benchmark consists in playing chess against the Deep Blue program, some observers may think that this requires some intelligence and hence it is a cognitive benchmark, whereas some other observers may object that it doesn't require intelligence and hence it is not a cognitive benchmark.

Some cognitive benchmarks have been established by people outside computer science and robotics. They include IQ tests developed by psychologists as well as animal intelligence tests developed by biologists to evaluate for example how well rats remember the path to a food source in a maze, or how do monkeys learn to press a lever to get food.

AI and robotics benchmarks have also been established mostly throughout programming or robotics competitions. The most famous examples are the AAAI Robot Competition, the FIRST Robot Competition, the DARPA Grand Challenge, the Eurobot Competition, the RoboCup competition (see picture), the Roboka Programming Contest. All these competitions define a precise scenario and a performance metrics based either on an absolute individual performance evaluation or a ranking between the different competitors. They are very well referenced on the Internet so that it should be easy to reach their official web site for more information.

The last chapter of this book will introduce you to a series of robotics cognitive benchmarks (especially the Rat's Life benchmark) for which you will be able to design your own intelligent systems and compare them to others.

Further reading

- [Artificial Intelligence](#)
- [Embedded Control Systems Design/RoboCup](#)

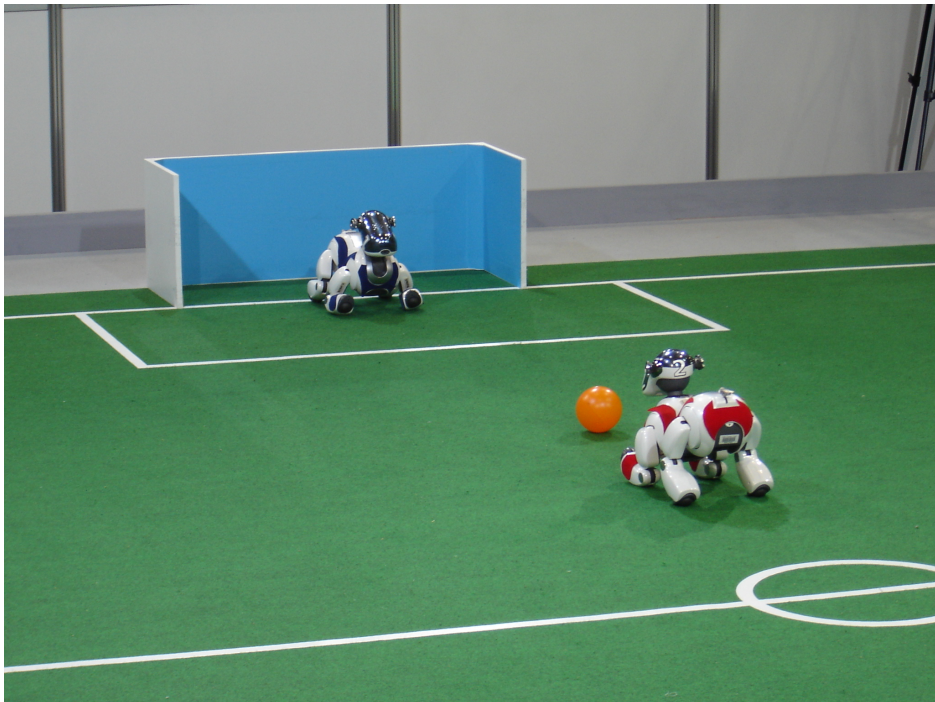


Figure 2.3: Aibo Robocup competition

Chapter 3

What are Robots?

Robots are electro-mechanical machines, interacting autonomously with their environment. They include sensors allowing them to perceive the environment. They also include actuators allowing them to modify their environment. Finally, they include a micro-processor allowing them to process the sensory information and control their actuators accordingly.

Robots in our every Day's Life

There exist few applications of robots in our every days' life. The most well known applications are probably toys and autonomous vacuum cleaners (see figure with toy robots), but there are also grass mower robots, mobile robots in factories, robots for space exploration, surveillance robots, etc. These devices are becoming increasingly complex in term of sensors, actuators and information processing.

Robots as Artificial Animals

Like animals, robots can move, perceive their environment and act. Like animals, they need energy to be able to operate. This is probably why several examples of animal robots were developed for toy applications. This includes the Sony Aibo dog robot (see figure), the Furby toy and later the Pleo dinosaur robot. From the mechanical and electronic points of view, these robots are very advanced. They are equipped with many sensors (distance sensors, cameras, touch sensors, position sensors, temperature sensors, battery level sensors, accelerometers, microphones, wireless communication, etc.) and actuators (motors, speakers, LEDs, etc.). They also include a significant processing power with powerful onboard micro-controllers or micro-processors. Moreover, the latest Aibo robots and several vacuum cleaner robots are able to search their recharging station, to dock on it, recharge their batteries and move on once the battery is charged. This makes them even more autonomous. However, their learning capabilities and ability to adapt to unknown situations is often still very limited. Hence, this affect to comparison with real animals in term of intelligence. When observing an Aibo robot and a real dog, there is no doubt for most observers that the dog is more intelligent than the robot. The same could probably apply if you compare the Pleo toy robot with a real



Figure 3.1: Two Pleo robots



Figure 3.2: Roomba of first generation: a vacuum cleaner

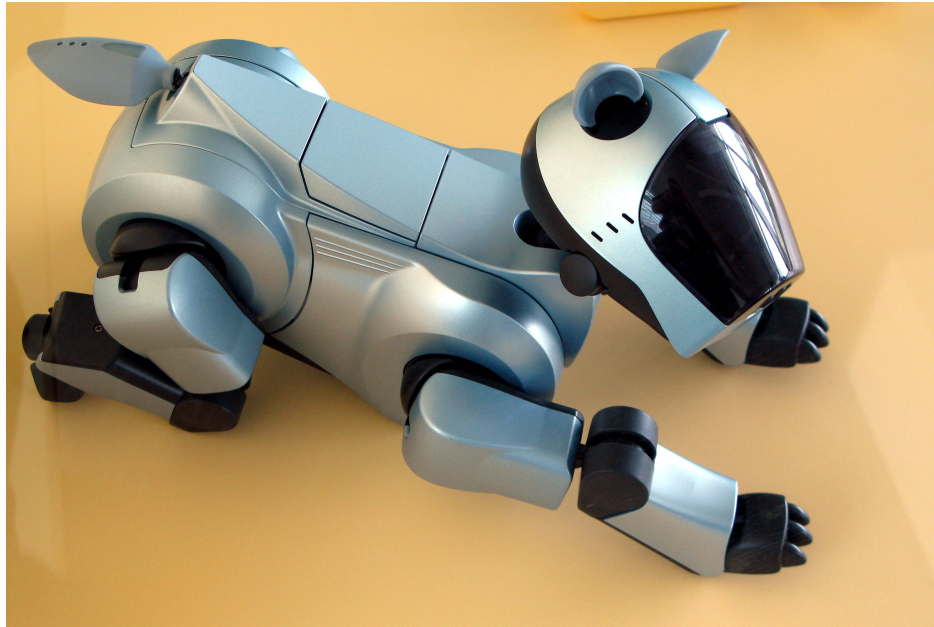


Figure 3.3: Asimo: Honda's humanoid robot

reptile. However, since reptiles appear to be more primitive than dogs, the difference of intelligence in the Pleo / reptile case may not be as evident as in the Aibo / dog case.

The conclusion we can draw from the above paragraph is that the hardware technology for intelligent robots is currently available. However, we still need to invent a better software technology to drive these robots. In other words, we currently have the bodies of our intelligent robots, but we lack their minds. This is probably the reason why most of the toy and vacuum cleaner robots described here are still provided with a remote control...

Hence this book will not focus on robot hardware, but rather on robot software because robot software is the greatest research challenge to overcome to be able to design more and more intelligent robots.

Chapter 4

E-puck and Webots

This chapter introduces you to a couple of useful robotics tools: e-puck, a mini mobile robot and Webots, a robotics CAD software. In the rest of this book, you will use both of them to practice hands-on robotics. Hopefully, this practical approach will make you understand what robots are and what you can do with them.

E-puck

Introduction

The e-puck robot was designed by Dr. Francesco Mondada and Michael Bonani in 2006 at EPFL, the Swiss Federal Institute of Technology in Lausanne (see Figure). It was intended to be a tool for university education, but is actually also used for research. To help the creation of a community inside and outside EPFL, the project is based on an open hardware concept, where all documents are distributed and submitted to a license allowing everyone to use and develop for it. Similarly, the e-puck software is fully open source, providing low level access to every electronic device and offering unlimited extension possibilities. The e-puck robots are now produced industrially by GCTronic S.à.r.l. (Switzerland) and Applied AI, Inc. (Japan) and are available for purchase from various distributors. You can order your own e-puck robot for about 950 Swiss Francs (CHF) from Cyberbotics Ltd. <http://www.cyberbotics.com>.

The e-puck robot was designed to meet a number of requirements:

- Neat Design: the simple mechanical structure, electronics design and software of e-puck is an example of a clean and modern system.
- Flexibility: e-puck covers a wide range of educational activities, offering many possibilities with its sensors, processing power and extensions.
- Simulation software: e-puck is integrated in the Webots simulation software for easy programming, simulation and remote control of real robot.
- User friendly: e-puck is small and easy to setup on a table top next to a computer. It doesn't need any cable (rely on Bluetooth) and provides optimal working comfort.

- Robustness and maintenance: e-puck resists to student use and is simple to repair.
- Affordable: the price tag of e-puck is friendly to university budgets.

The e-puck robot has already been used in a wide range of applications, including mobile robotics engineering, real-time programming, embedded systems, signal processing, image processing, sound and image feature extraction, human-machine interaction, inter-robot communication, collective systems, evolutionary robotics, bio-inspired robotics, etc.



Figure 4.1: The e-puck mobile robot

Overview

The e-puck robot is powered by a dsPIC processor, i.e., a Digital Signal Programmable Integrated Circuit. It is a micro-controller processor produced by the Microchip company which is able to

perform efficient signal processing. This feature is very useful in the case of a mobile robot, because extensive signal processing is often needed to extract useful information from the raw values measured by the sensors.

The e-puck robot also features a large number of sensors and actuators as depicted on the pictures with devices and described in the table. The electronic layout can be obtained at this address: [E-puck electronic layout](#) Each of these sensors will be studied in detail during the practical investigations later in this book.

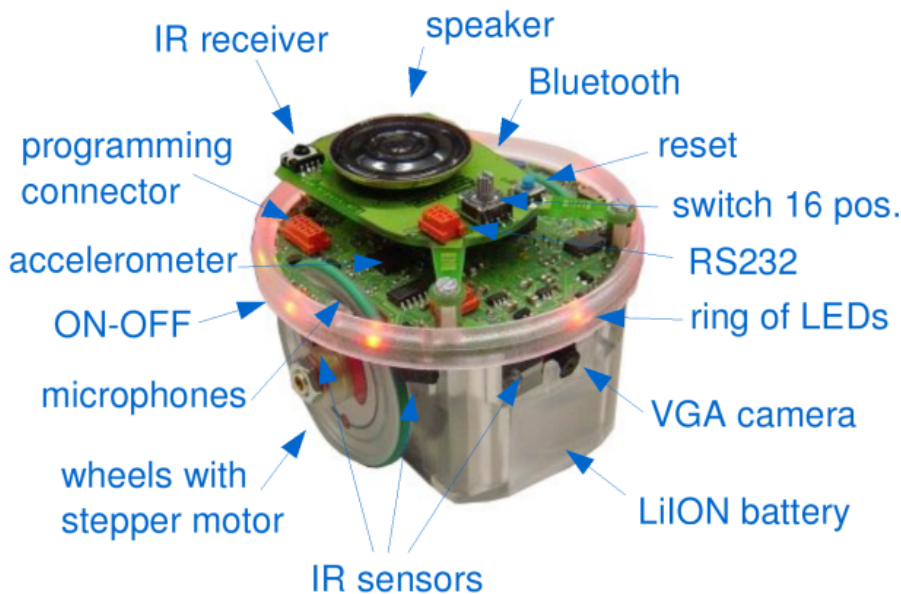


Figure 4.2: Sensors and actuators of the e-puck robot

Webots

Introduction

Webots is a software for fast prototyping and simulation of mobile robots. It has been developed since 1996 and was originally designed by Dr. Olivier Michel at EPFL, the Swiss Federal Institute of Technology in Lausanne, Switzerland, in the lab of Prof. Jean-Daniel Nicoud. Since 1998, Webots is a commercial product and is developed by Cyberbotics Ltd. User licenses of this software have been sold to over 400 universities and research centers world wide. It is mostly used for research and education in robotics. Besides universities, Webots is also used by research organizations and corporate research centers, including Toyota, Honda, Sony, Panasonic, Pioneer, NTT, Samsung, NASA, Stanford Research Institute, Tanner research, BAE systems, Vorverk, etc.

The use of a fast prototyping and simulation software is really useful for the development of most advanced robotics project. It actually allows the designers to visualize rapidly their ideas, to

check whether they meet the requirements of the application, to develop the intelligent control of the robots, and eventually, to transfer the simulation results into a real robot. Using such software tools saves a lot of time while developing new robotics projects and allows the designers to explore more possibilities than they would if they were limited to using only hardware. Hence both the development time and the quality of the results are improved by using a rapid prototyping and simulation software.

Overview

Webots allows you to perform 4 basic stages in the development of a robotic project as depicted on the figure.

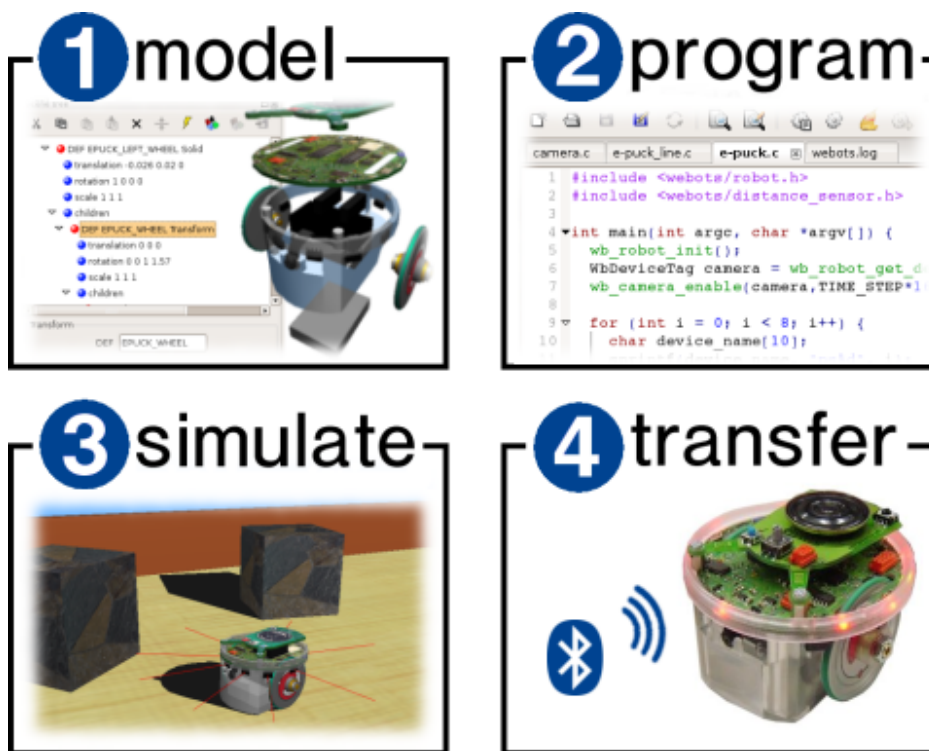


Figure 4.3: Webots development stages

The first stage is the modeling stage. It consists in designing the physical body of the robots, including their sensors and actuators and also the physical model of the environment of the robots. It is a bit like a virtual LEGO set where you can assemble building blocks and configure them by changing their properties (color, shape, technical properties of sensors and actuators, etc.). This way, any kind of robot can be created, including wheeled robots, four legged robots, humanoid robots, even swimming and flying robots! The environment of the robots is created the same way, by populating the space with objects like walls, doors, steps, balls, obstacles, etc. All the

physical parameters of the object can be defined, like the mass distribution, the bounding objects, the friction, the bounce parameters, etc. so that the simulation engine in Webots can simulate their physics. The figure with the simulation illustrates the model of an e-puck robot exploring an environment populated with stones. Once the virtual robots and virtual environment are created, you can move on to the second stage.

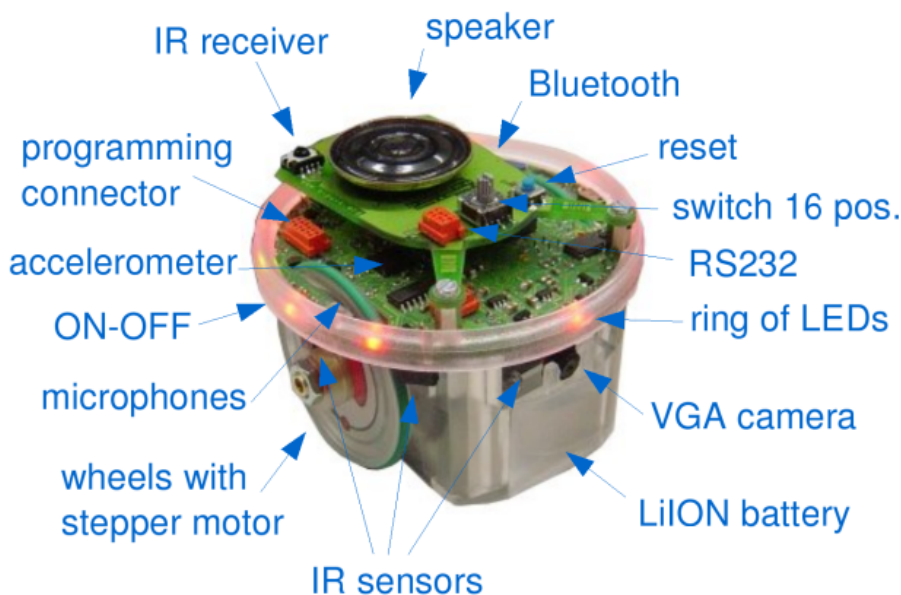


Figure 4.4: Model of an e-puck robot in Webots

The second stage is the programming stage. You will have to program the behavior of each robot. In order to achieve this, different programming tools are available. They include graphical programming tools which are easy to use for beginners and programming languages (like C, C++ or Java) which are more powerful and enable the development of more complex behaviors. The program controlling a robot is generally an endless loop which is divided into three parts: (1) read the values measured by the sensors of the robot, (2) compute what should be the next action(s) of the robot and (3) send actuators commands to perform these actions. The easiest parts are parts (1) and (3). The most difficult one is part (2) as this is here that lie all the Artificial Intelligence. Part (2) can be divided into sub-parts such as sensor data processing, learning, motor pattern generation, etc.

The third stage is the simulation stage. It allows you to test if your program behaves correctly. By running the simulation, you will see your robot executing your program. You will be able to play interactively with your robot, by moving obstacles using the mouse, moving the robot itself, etc. You will also be able to visualize the values measured by the sensors, the results of the processing of your program, etc. It is likely you will return several times to the second stage to fix or improve your program and test it again in the simulation stage.

Finally, the fourth stage is the transfer to a real robot. Your control program will be transferred

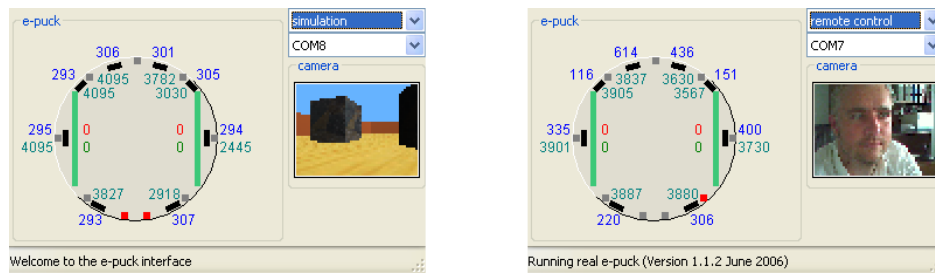


Figure 4.5: Transfer from the simulation to the real robot

into the real robot running in the real world. You could then see if your control program behaves the same as in simulation. If the simulation model of your robot was performed carefully and was calibrated against its real counterpart, the real robot should behave roughly the same as the simulated robot. If the real robot doesn't behave the same, then it is necessary to come back to the first stage and refine the model of the robot, so that the simulated robot will behave like the real one. In this case, you will have to go through the second and third stages again, but mostly for some little tuning, rather than redesigning your program. The figure with two windows shows the e-puck control window allowing the transfer from the simulation to the real robot. On the left hand side, you can see the point of view of the simulated camera of the e-puck robot. On the right hand side, you can see the point of view of the real camera of the robot.

Features	Thecnical information
Size, weight	70 mm diameter, 55 mm height, 150 g
Battery autonomy	5Wh LiION rechargeable and removable battery providing about 3 hours autonomy
Processor	dsPIC 30F6014A @ 60 Mhz (~15 MIPS) 16 bit microcontroller with DSP core
Memory	RAM: 8 KB; FLASH: 144 KB
Motors	2 stepper motors with a 50:1 reduction gear, resolution: 0.13 mm
Speed	Max: 15 cm/s
Mechanical structure	Transparent plastic body supporting PCBs, battery and motors
IR sensors	8 infra-red sensors measuring ambient light and proximity of objects up to 6 cm
Camera	VGA color camera with resolution of 480x640 (typical use: 52x39 or 480x1)
Microphones	3 omni-directional microphones for sound localization
Accelerometer	3D accelerometer along the X, Y and Z axis
LEDs	8 independent red LEDs on the ring, green LEDs in the body, 1 strong red LED in front
Speaker	On-board speaker capable of WAV and tone sound playback
Switch	16 position rotating switch on the top of the robot
PC connection	Standard serial port up to 115 kbps
Wireless	Bluetooth for robot-computer and robot-robot wireless communication
Remote control	Infra-red receiver for standard remote control commands
Expansion bus	Large expansion bus designed to add new capabilities
Programming	C programming with free GNU GCC compiler. Graphical IDE (integrated development environment) provided in Webots
Simulation	Webots facilitates the use of the e-puck robot: powerful simulation, remote control, graphical and C programming systems

Table 4.1: Features of the e-puck robot

Chapter 5

Getting started

The first section of this chapter (section [Explanations about the Practical Part](#)) explains how to use this document. It presents the formalism of the practical part, i.e., the terminology, the used icons, etc.

The following sections will help you to configure your environment. For profiting as much as possible of this document, you need Webots, an e-puck and a Bluetooth connection between both of them. Nevertheless, if you haven't any e-puck, you can still practice a lot of exercises. Before starting the exercises you need to setup these systems. So please refer to the following sections:

- Section [Get Webots and install it](#) describes how to install Webots on your computer.
- Section [Bluetooth Installation and Configuration](#) describes how to create a Bluetooth connection between your computer and your e-puck.
- Section [Open Webots](#) describes how to launch Webots.
- Section [E-puck Prerequisites](#) describes how to update your e-puck's firmware.
- [Chapter 4](#) in the [User Guide](#) describes how to model your own world with Webots.

If you want to go further with Webots you can consider the online user guide [User Guide](#) or the [Reference Manual](#).

Explanations about the Practical Part

Throughout the practical part, you will find different symbols. They have the following meaning:



: When this symbol occurs, you are invited to answer a question. The questions are related either to the current exercise or to a more general topic. They are referenced by a number which has the following form: "[Q." + question number + "]". For example, the third question of the exercise will have the [Q.3](#) number.



: When this symbol occurs, you will be invited to practice. For example you will have to program your robot to obtain a specific behavior. They are referenced by a number which has the following form: "[P." + question number + "]".



: When this symbol occurs, only the users who work with a Linux operating system are invited to read what follows. Note that this curriculum was written using Ubuntu Linux.



: Ibid for the Windows operating system. Note that this curriculum was also written using Windows XP.



: Ibid for Mac OS X operating system.

Each section of this document corresponds to an exercise. Each exercise title finishes with its level between square brackets (for example : [Novice]). When an exercise title, a question number or a practical part number is bounded by the star character (for example: *[Q.5]*), it means that this part is optional, i.e., this part is not essential for the global understanding of the problem but is recommended for accruing your knowledge. They can also be followed by the **Challenge** tag. This tag means that this part is more difficult than the others, and that it is optional.

Get Webots and install it

The easiest way to obtain Webots is to visit the following website:

<http://www.cyberbotics.com>

There, you will find all the information about Webots and its installation.

Bluetooth Installation and Configuration

First of all, your computer needs a Bluetooth device to communicate with your e-puck. This kind of devices is often integrated in modern laptops. The installation of this device is out of the scope of this document. However, its correct installation is required. So, refer to its installation manual or to the website of its constructor. This document explains only the configuration of the Bluetooth connection between your computer and the e-puck. This connection emulates a serial connection. Refer to your corresponding operating system:



First of all, your Linux operating system needs a recent kernel. Moreover, the following packets have to be installed: **bluez-firmware**, **bluez-pin** and **bluez-utils**¹

The commands **lsusb** (or **lspci** according to your Bluetooth hardware) and **hciconfig** inform about the success of the installation.

¹This part is inspired by the "Bluetooth and e-puck" article written by Bonani Michael on the official e-puck website.

Switch on your e-puck (with the ON-OFF switch) and execute the following command:

```
> hcitool scan
```

```
Scanning ...
```

```
00:13:11:52:DE:A8 PowerBook G4 12"
```

```
08:00:17:2C:E0:88 e-puck_0202
```

The last line corresponds to your e-puck. It shows its MAC address (08:00:17:2C:E0:88) and its name (e-puck_0202). The number of the e-puck (0202) should correspond with its sticker.

Edit the `/etc/bluetooth/hcid.conf` and change the security parameter from "auto" to "user".

Edit the `/etc/bluetooth/rfcomm.conf` configuration file and add the following entree (or modify the existing `rfcomm0` entree):

```
rfcomm0 {

    bind yes;

    device 08:00:17:2C:E0:88;

    channel 1;

    comment "e-puck_0202";

}
```

`rfcomm0` is the name of the connection. If more than one e-puck is used, enter as entrees (`rfcomm0`, `rfcomm1`, etc.) as there are robots. The device tag must correspond to the e-puck's MAC address and the comment tag must correspond to the e-puck name. `rfcomm0` is the name this connection.

Execute the following commands:

```
> /etc/init.d/bluez-utils restart
```

```
> rfcomm bind rfcomm0
```

A PIN (Personal Identification Number) will be asked to you (by `bluez-pin`) when trying to establish the connection. This PIN is a 4 digits number corresponding to the name (or ID) of your e-puck, i.e., if your e-puck is called "e-puck_0202", then, the PIN is 0202.

Your connection will be named "`rfcomm0`" in Webots.



This part² was written using Windows XP. There are probably some differences with other versions of Windows.

²This part is inspired by the third practical work of the EPFL's Microinformatique course.

After the installation of your Bluetooth device, an icon named "My Bluetooth Places" is appeared on your desktop. If it is not the case, right click on the Bluetooth icon in the system tray and select "Start using Bluetooth". Double-click on the "My Bluetooth Places" icon. If you use "My Bluetooth Places" for the first time, this action will open a wizard. Follow the instructions of this wizard up to arrive at the window depicted in the first figure of the wizard. If you already used "My Bluetooth Places", click on the "Bluetooth Setup Wizard" item. This action will open this window.



Figure 5.1: The first window of the wizard

In this first window, select the second item: "I want to find a specific Bluetooth device and configure how this computer will use its services.". Switch on your e-puck by using the ON/OFF switch. A green LED on the e-puck should be alight. Click on the Next button.

The second window searches all the visible Bluetooth devices. After a time, an icon representing your e-puck must appear. Select it and click on the Next button.

This action opens the security window. Here you have to choose four digits for securing the connection. Choose the same number as your e-puck (if your e-puck is called "e-puck_0202", choose 0202 as PIN) and click on the Initiate Paring button.

The opened window (on a figure too) enables you to choose which service you want to use. Select COM1 (add a tick). If there isn't any service, it's maybe because the battery is too low. This action opens a new window (see the next figure). Here you can select which port is used for the communication. Select for example "COM6".

To finish, click on the Finish button.

Finally, in the "My Bluetooth Places" window (also shown on figure), right click on the "e-puck_0202 COM1" icon and select the "Connect" item.

Your connection will be named "COM6" in Webots.

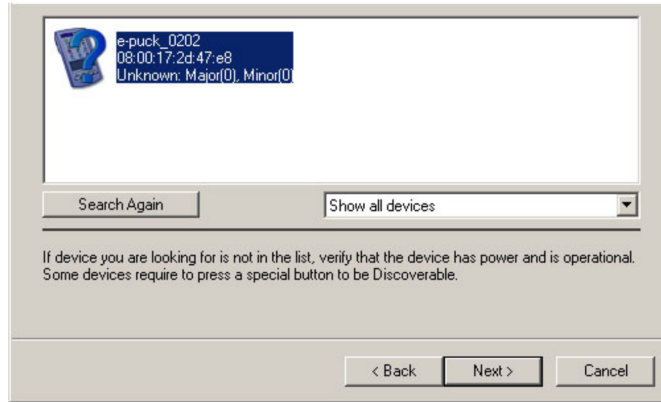


Figure 5.2: Research the Bluetooth devices

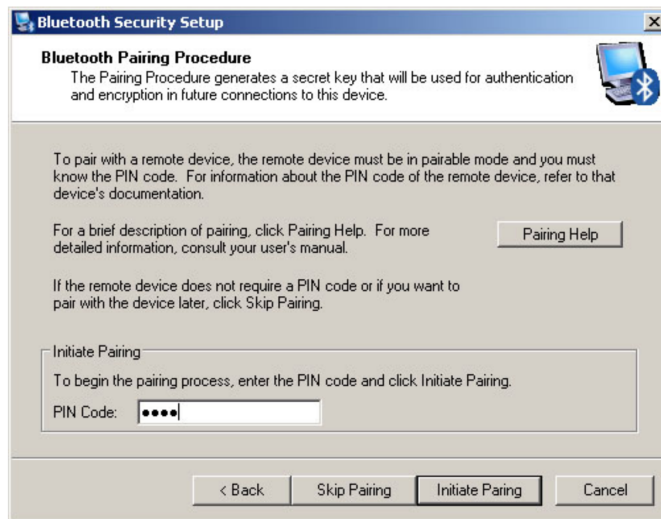


Figure 5.3: The security window

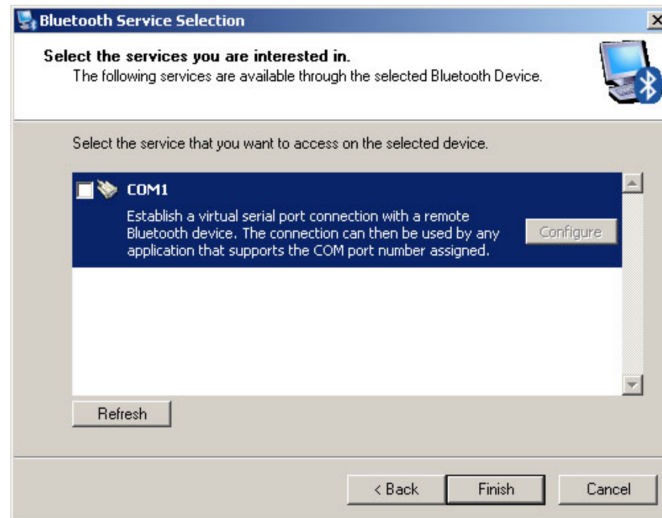


Figure 5.4: Selection of the services

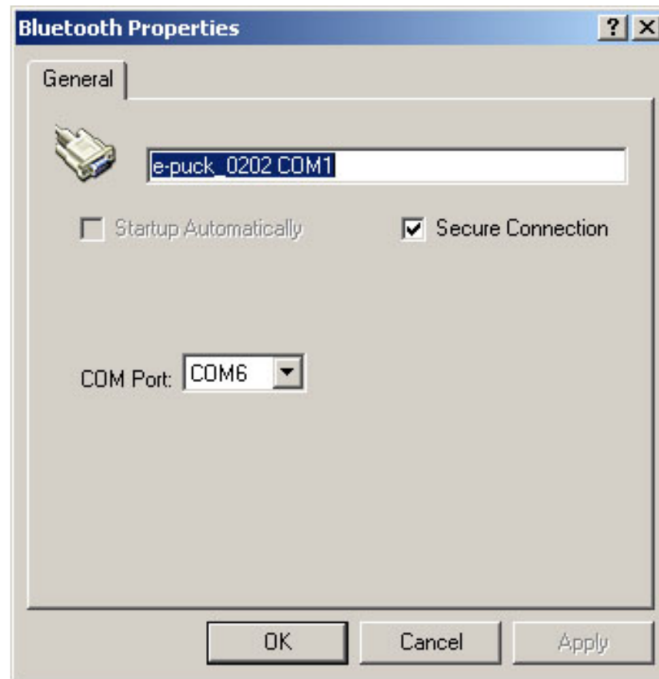


Figure 5.5: Configure the COM port

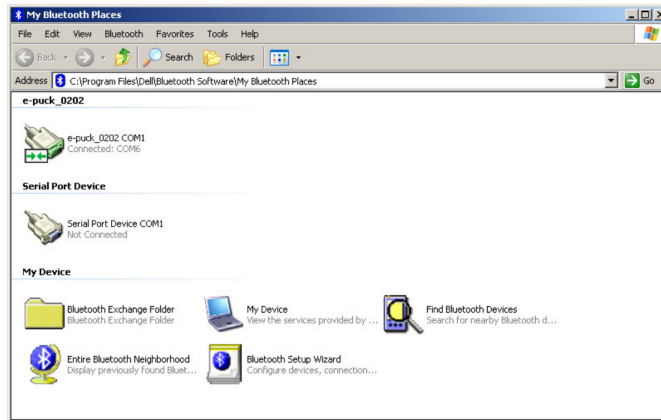


Figure 5.6: My Bluetooth places



If your Bluetooth device is correctly installed, a Bluetooth icon should appear in your System Preferences. Click on this icon and on the Paired Devices tab. Switch on your e-puck. A green LED on the e-puck should be alight. Then, click on the New... button. It should open a new window which scans the visible Bluetooth devices. After a while, the name of your e-puck should appear in this list. Select the e-puck in the list and click on the Pair button. A pass key is asked. It is the number of your e-puck coded on 4 digits. For example, if your e-puck has the number 43 on its stickers, the pass key is 0043. Enter the pass key and click on the OK button.

Once pairing is completed, you need to specify a serial port to use in order to communicate with Webots. So, click the Serial Ports tab. Thanks to the New... button, create an outgoing port called COM1. Finally, quit the Bluetooth window.

Your connection will be named “COM1” in Webots.

Open Webots

This section explains how to launch Webots. Naturally it depends on your environment. So please refer to your corresponding operating system:



Open a terminal and execute the following command:

```
> webots &
```

You should see the simulation window appear on the screen.



From the Start menu, go to the Program Files | Cyberbotics menu and click on the Webots (+ version) menu item. You should see the simulation window appear on the screen.



Open the directory in which you uncompressed the Webots package and double-click on the webots icon. You should see the simulation window appear on the screen.

E-puck Prerequisites

An e-puck has a computer program (called firmware) embedded in its hardware. This program defines the behavior of the robot at startup.

There are three possible ways to use Webots and an e-puck:

- The simulation: By using the Webots libraries, you can write a program, compile it and run it in a virtual 3D environment.
- The remote-control session: You can write the same program, compile it as before and run it on the real e-puck through a Bluetooth connection.
- The cross-compilation: You can write the same program, cross-compile it for the e-puck processor and upload it on the real robot. In this case, the previous firmware is substituted by your program. In this case, your program is not dependent on Webots and can survive after the rebooting of the e-puck.

In the case of a remote-control session, your robot needs a specific firmware for having talks to Webots.

For uploading the latest firmware (or other programs) on the real e-puck, select the menu `Tool | Upload to e-puck robot...` as depicted in the figure. Then, a message box asks you to choose which Bluetooth connection you want to use. Select the connection which is linked to the e-puck and click on the Ok button. The orange LED on the e-puck will switch on. Then, a new message box asks you to choose which file you want to upload. Select the following file and click on the Ok button:

```
...webots_root/transfer/e-puck/firmware/firmware-X.Y.Z.hex
```

Where X.Y.Z is the version number of the firmware. Then, if the firmware (or an older version) isn't already installed on the e-puck, the e-puck must be reseted when the window depicted in the figure is displayed.

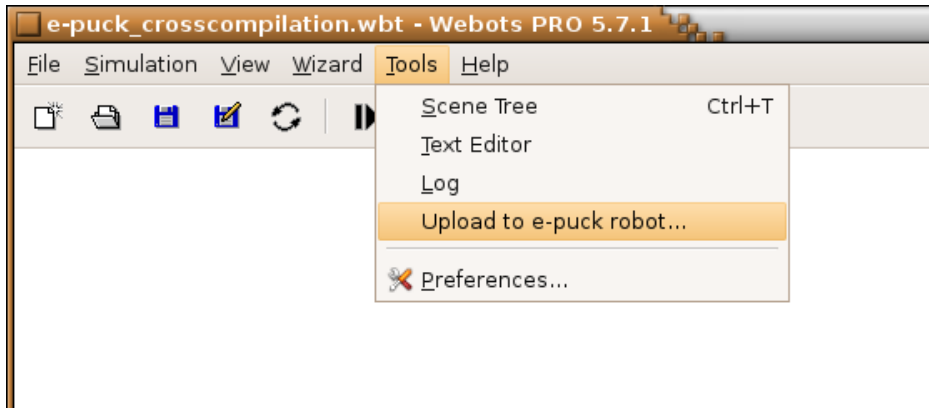


Figure 5.7: The location of the tool for uploading a program on the e-puck

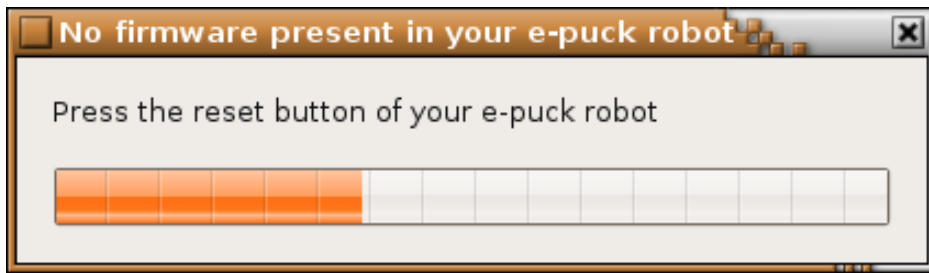


Figure 5.8: When this window occurs, the e-puck must be reset by pushing the blue button on its top

Chapter 6

Beginner programming Exercises

This chapter is composed of a series of exercises for beginners. You don't need prior knowledge to go through these exercises. The aim is to learn the basics of mobile robotics by manipulating both your e-puck and Webots. First, you will discover some e-puck devices and their utility. Then, you will acquire the concept of a robot controller. And finally, you will program a simple robot behavior by using a Webots module: BotStudio. This module enables to program an e-puck robot using a graphical interface. You will discover how to use it and what are the notions related to it.

Discovery of the e-puck [Beginner]

As explained in the chapter [E-puck and Webots](#), an e-puck has different devices. Through this document, you will use some of them: the stepper motors, the LEDs, the accelerometer, the infrared sensors and the camera. In this exercise, you will discover the utility of each of them. The following list gives you a quick definition of these devices. You will see in the next chapter all these devices in more details.

- Stepper motor: A stepper motor¹ is an electrical motor which breaks up a full rotation into a large number of steps. An e-puck possesses two stepper motors of 1000 steps. They can achieve a speed of about one rotation per second. The wheels of the e-puck are fixed to these motors. They are used to move the robot. They can move independently. Moreover, for knowing the position of the wheels, an incremental encoder can be used. The e-puck encoder returns the number of steps since the last reset of the encoder. For example, this "device" can be used for turning the wheel of one turn precisely.
- LED: A LED² (Light-Emitting Diode) is a small device which can emit light by using few energy. An e-puck possesses several LEDs. Notably, 8 around it, 4 in the e-puck body and 1 in front of it. The front LED is more powerful than the others. The aim of these LEDs is mainly to have a feedback on the state of the robot. They can also be used for illuminating the environment.

¹More information on: [Stepper motor](#)

²More information on: [Led](#)

- Accelerometer: An accelerometer³ is a device which measures the total force applied on it as a 3D vector. An e-puck has a single accelerometer. If your e-puck is at rest, the accelerometer indicates at least the gravitational vector. The accelerometer can be used for detecting a collision with a wall or for detecting the fall of the robot.
- Infrared (IR) sensor: An e-puck possesses 8 infrared (IR) sensors. An IR sensor is a device which can produce an infrared light (a light which is out the range of the visible light) and which can measure the amount of the received light. It has two kind of use. First, only the received light is measured. In this configuration, the IR sensor measures the light of the nearby environment. The e-puck can detect for example from where a light illuminates it. Second, the IR sensor emits infrared light and measures the received light. If there is an obstacle in front of the IR sensor, the light will bounce on it. The light difference is bigger. So, the e-puck can estimate the distance between its IR sensors and an obstacle.
- Camera: In front of the e-puck, there is also a VGA camera. The e-puck uses it to discover its direct front environment. It can for example follow a line, detect a blob, recognize objects, etc.

Note that the stepper motors and the LEDs are actuators. This device have an effect on the environment. To the contrary, the IR sensors and the camera are sensors. They measure specific information of the environment. On the following page you can see photos of the [mechanical design](#).

To successfully go through the following exercises, you have to know the existence of other devices. The e-puck is alimented with a Li-ION battery. It has a running life of about 3 hours. You can switch on or off your e-puck with the ON/OFF switch which is located near the right wheel. The robot has also a Bluetooth interface which allows a communication with your computer or with other e-pucks.

Finally, the e-puck has other devices (like the microphones and the speaker) that you will not use in this document because the current version of Webots doesn't support them yet.



[Q.1] What is the maximal speed of an e-puck? Give your answer in cm/s. (Hint: The wheel radius is about 2.1 cm. Look at the definition of a stepper motor above.)



[Q.2] Compare your e-puck with an actual mobile phone. Which e-puck devices were influenced by this industry?



[Q.3] Sort the following devices either into the actuator category or into the sensor category: a LED, a stepper motor, an IR sensor, a camera, a microphone, an accelerometer and a speaker.



[P.1] Find where these devices are located on your real e-puck. (Hint: look at the figure [Epuck devices.png](#))

³More information on: [Accelerometer](#)

Robot Controller [Beginner]

In order to understand the concept of a robot controller you will play the role of the robot controller. You will perceive the sensory information coming from the sensors of the robot and you will be able to control the actuators of the robot. In this exercise, you will not actually program the behavior of the robot, but you will nevertheless control the robot.

Open the World File

First of all, you need to open the world file of this exercise. A world file contains the entire environment of the simulation, i.e., the robot shape, the ground shape, the obstacles shape and some general information like the position of the camera and even the direction of gravitational vector. In the simulation window (window (1) in figure below), click on File | Open menu and open:



```
.../worlds/beginner_robot_controller.wbt
```

You can also open the world file by clicking on the open button on the tool box of the simulation window. The e-puck model and its environment are loaded in Webots. In the simulation window, you can see an e-puck on a green board.

The Webots Windows and the simulation Camera

Webots can display several windows. Some of them were already introduced. You will focus especially on two of them (which are depicted on a figure):

- The simulation window (1): This window is probably the most important one. It shows a 3D representation of the simulation. In our case, you can see a virtual e-puck and its virtual environment. If you want to modify the camera orientation, just click and drag with the left button of the mouse where you want in the panel. Similarly you can modify the position of the camera by using the right button (Note for the Mac OS X users : if you have a mouse with a single button, hold down the Ctrl key and click for emulating the right click.). Finally you can also set the zoom by moving the mouse wheel. There are also two important buttons in this window: the play/stop button and the revert button. With the first one, the simulation can be either played or stopped, and with the second one, the entire simulation can be reset.
- The robot window (2): This window shows a 2D representation of the e-puck. The purpose of this window is to visualize the sensor values and the actuators values in real-time during a simulation. The figure with the robot window shows the meaning of the values that can be seen. The red integers correspond to the speed of the motors. They should be initially null. The green values below correspond to the encoders. The light measured by the IR sensor is represented by the green integers. While the distance between a IR sensor and an obstacle is represented by blue integers. So, note that the green and the blue values represent the same device. The red or black rectangles correspond to the LEDs which are respectively switched on or off. Finally, the accelerometer is represented both by a 2D vector which corresponds to the inclination of the e-puck, and by a slider which represents the norm of the acceleration. This window contains also a drop-down menu to configure the Bluetooth connection.

-  [P.1] By using the camera, identify where is the front and the back of your virtual e-puck.
(Hint: the camera is placed in the front of the e-puck)
-  [P.2] Try to place the camera of the simulation window on the e-puck roof in order to see in front of it. Then, use the revert button.

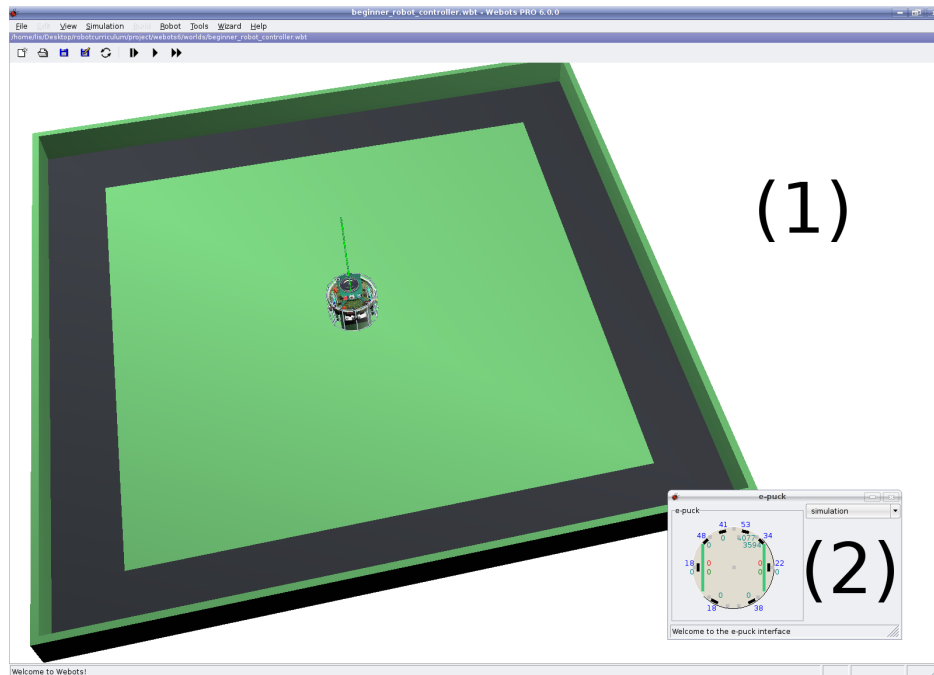


Figure 6.1: The simulation window (1) and the robot window (2)

The e-puck Movements

Check that the simulation is running by clicking on the start/stop button. Then, click on the virtual e-puck in order to select it. When your e-puck is selected, white lines appear. They represent the bounds of your object for the physical simulation. You also remark red lines. They represent the direction of the IR sensors. While the magenta lines correspond to the field of view of the camera. Moreover, you can observe the camera values into a little window in the top left part of the simulation window.

On your keyboard, press the “S” key and the “X” key for respectively increasing or decreasing the speed value of the left motor. Try to press the “D” key and on the “C” key for modifying the speed of the right motor. Now you can move the virtual robot like a remote control toy. Note that only one key can be pressed at the same time.

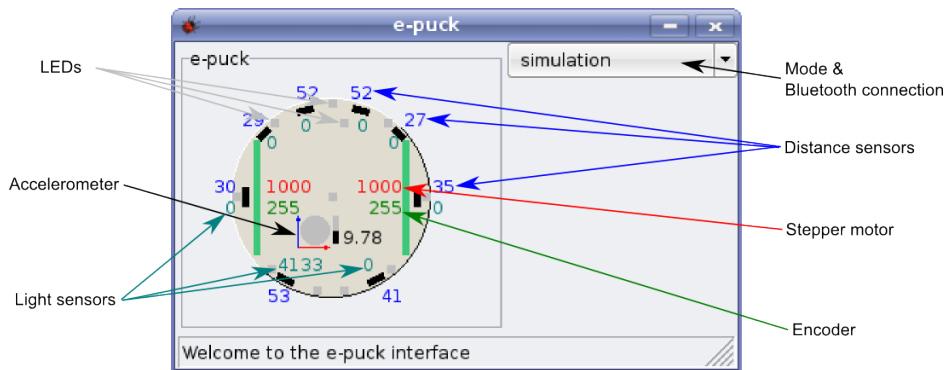


Figure 6.2: A description of the robot window



[P.3] Try to follow the black band around the board by using these four buttons.



[Q.1] Is it easy? What are the difficulties?



[Q.2] There are different kind of movements with an e-puck. Can you list them? (Ex: the e-puck can go forwards)



[Q.3] Try to use the keyboard arrows and the "R" key. What is the utility of these commands? Explain the difference with the first ones. Are they more practical? Why?

Blinded Movement [Challenge]

The aim of this subsection is to play the role of the robot controller. A robot controller perceives only the values measured by the robot sensors, treats them and sends some commands to the robot actuators as depicted in the figure. Note that the sensor values are modified by the environment, and that a robot can modify the environment with its actuators.



[P.4] Hide the simulation window (However, this window has to remain selected so that the keyboard strokes are working. A way to hide it is to move it partially off-screen) and just look at the sensor values. Try now to follow the wall as before only with the IR sensor information.



[Q.4] What information is useful? From which threshold value do you observe that a wall is close to the robot?

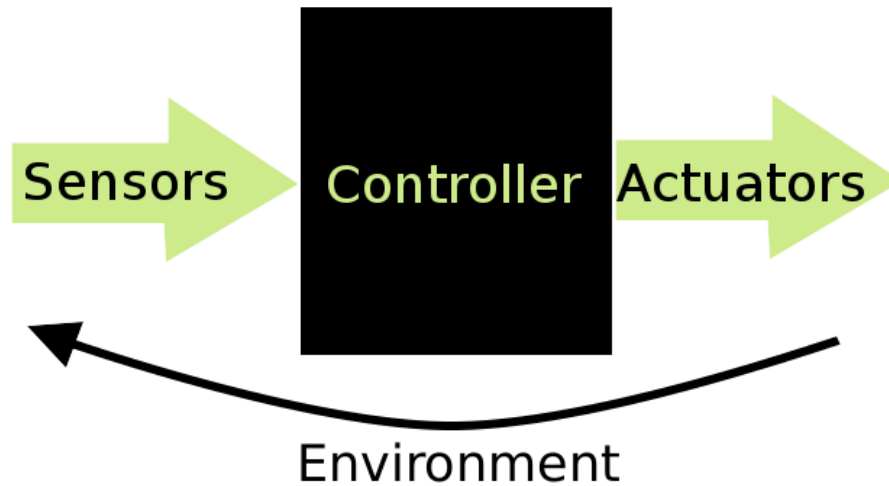


Figure 6.3: The robot controller receives sensor values (ex: IR sensor, camera, etc.) and sends actuator commands (motors, LEDs, etc.)

Let's move your real Robot

You probably have a real e-puck in front of you and you would like to see it moving! Webots can communicate with an e-puck via a Bluetooth connection. It can receive some values from the e-puck sensors and send some values to command the e-puck actuators. So, Webots can play the role of the controller. This mode of operation is called a remote-control session.

In order to proceed, configure first your Bluetooth connection as explained in the section [Bluetooth configuration](#). Stop the simulation with the start/stop button. Switch on your e-puck with the ON/OFF switch. Then, in the robot window, select your Bluetooth connection in the drop-down menu. Behind the e-puck, an orange LED should switch on. To finish press the start/stop button in order to run the program. Your e-puck should behave the same as in simulation.



[Q.5] Observe the sensor values from the real e-puck. Are they similar as the virtual ones?



[Q.6] Set the motor speeds to 10|10. When the real e-puck moves slowly, it vibrates. That does not occur in simulation. Could you explain this phenomenon?

Your Progression

Congratulation! You finished the first exercise and stepped into the world of robotics. You already learned a lot:

- What is a sensor, an actuator and a robot controller.
- What kind of problems a robot controller must be able to solve.
- What are the basic devices of the e-puck. In particular, the stepper motors, the LEDs, the IR sensors, the accelerometer and the camera.
- How to run your mobile robot both in simulation and in reality and what is a remote-control session.
- How to perform some basic operations with Webots.

Move your e-puck [Beginner]

You already learned what a robot controller is. In the following exercises you will create simple behaviors by using a graphical programming interface: BotStudio. This module is integrated in Webots. The aim of this exercise is to introduce BotStudio by discovering the e-puck's movement possibilities.

Open the World File

Similarly to the first exercise, open the following world file:

```
.../worlds/beginner_move_your_epuck.wbt
```

Two windows are opened. The first one is the simulation window that you know. You should observe a similar world as before except that the size of the board is twice as big. This is because an e-puck needs room for moving. The second window is the BotStudio window (see figure).

The “forward” State

A BotStudio window is composed of two main parts. The left part is a graphical representation of an automaton. You will learn to use this part and understand the automaton concept in the next exercise. The right part represents an e-puck in 2 dimensions. On this representation, you can observe the e-puck sensors values in real-time. Moreover, you can set the actuators commands. This interface has also a drop-down menu for choosing a Bluetooth connection in order to create a remote-control session. This menu is similar to the two drop-down menu of the robot window that you saw above. In top, there is a tool menu. This menu enables you to create, to load, to save or to modify an automaton. The last button (the upload button) executes your automaton on the e-puck.

In the BotStudio window, select the “forward” state (blue rectangle in the middle of the white area) just by clicking on it. A selected rectangle becomes yellow. In the right part of the BotStudio window, you can modify the actuator commands, i.e., the motors speed and the LEDs state. If you want to change the motors speed, click and drag the two yellow sliders. You can set this value between -100 and 100. 0 corresponds to a null speed, i.e., the wheel won't turn. A positive value should turn the wheel forward, and a negative one backwards. If you want to change the state of a LED, click on its corresponding gray circle (red -> on, black -> off, gray -> no modification).

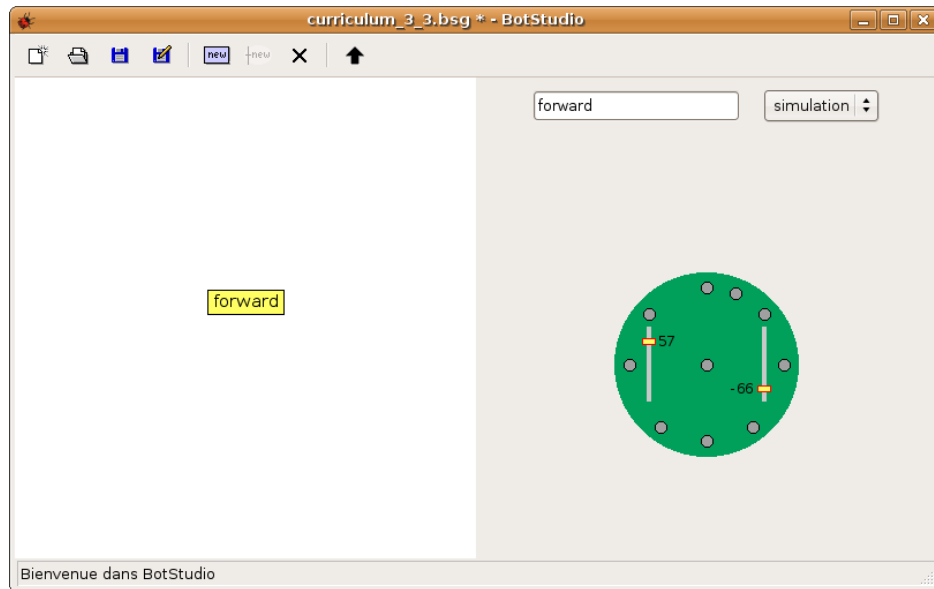


Figure 6.4: The BotStudio interface

Configure the “forward” state as follows: all the LEDs are alight, and the motors speeds are -30|30. Upload it on the virtual e-puck by clicking on the upload button. If the simulation is running, the virtual e-puck should change its actuators values accordingly. Note that when the simulation is launched, the right part of BotStudio displays the IR sensors.



[P.1] Set the actuators of your virtual e-puck in order to go forward, to go backwards, to follow a curve and to spin on itself.



[Q.1] For each of these moves, what are the links between the two speeds? (Example: forward : $\text{right_speed} = \text{left_speed}$ and $\text{right_speed} > 0$ and $\text{left_speed} > 0$)



[Q.2] There are 17 LEDs on an e-puck. 9 red LEDs around the e-puck (the back LED is doubled), 1 front red LED, 4 intern green LEDs, 2 LEDs (green and red) for the power supply and 1 orange LED for the Bluetooth connection. Find where they are and with which button or operation you can switch them on. (Hint: some of them are not under your control and some of them are linked together, i.e. they cannot be switched on or off independently)

The real e-puck's IR Sensors

The aim of this subsection is to create a remote-control session with your real e-puck. This part is similar to the subsection in previous exercise where you used the real robot. There are just two

differences due to the fact that the BotStudio window is used instead of the robot window. For choosing the Bluetooth connection, there is just one drop-down menu instead of two. So, please select your Bluetooth connection instead of the simulation item on the top right part of the window. Then, you have to click on the upload button for starting the remote-control session.



[P.2] Set the actuators such that the e-puck doesn't move. Try this configuration on your real e-puck by creating a remote-control session. Put your hands around your real e-puck and observe the modifications of the IR sensor values in the BotStudio window.



[Q.3] What are the values of the front left IR sensor when there is an obstacle (example: a white piece of paper) at 1 cm ? At 3 cm ? At 5 cm ? At 10 cm ? Starting from which distance is it difficult to distinguish an obstacle from the noise⁴ ?

Simple Behavior: Finite State Machine (FSM) [Beginner]

In the precedent exercise, you learned to configure a single state. One cannot speak about behavior yet because your robot doesn't interact with its environment, i.e., it moves but it hasn't any reaction. The goal of this exercise is to create a simple behavior. You will discover what an automaton is, how it is related to the robot controller concept and how to construct an automaton using BotStudio.

Finite State Automaton

A finite state automaton (FSM)⁵ is a model of behavior. It's a possible way to program a robot controller. It's composed of a finite number of states and of some transitions between them. In our case, the states correspond to a configuration of the robot actuators (the wheels speed and the LEDs state), while the transitions correspond to a condition over the sensor values (the IR sensors and the camera), i.e., under which condition the automaton can pass from one state to another. One state is particularly important: the initial state. It's the state from where the simulation begins. During this curriculum, an automaton will have the same signification as an FSM.

BotStudio enables you to create graphically an automaton. When an automaton is created, you can test it on your virtual or real e-puck. You will observe in the following exercises that this simple way to program enables to create a large range of behaviors.

Open the World File and move Objects

Open the following world file:

```
.../worlds/beginner_finite_state_machine.wbt
```

This time, there are two obstacles. You can move an object (obstacle, e-puck or even walls) by selecting the desired object, and drag and drop it by pressing the shift key. The reverse button can be pressed when you want to reset the simulation.

⁴Noise is an unwanted perturbation. More information on : [Noise](#)

⁵Source and more information on : [Finite state automaton](#)

Creation of a Transition

In the BotStudio window, create two states with the new state button. Name the first state "forward" and the second one "stop" by using the text box at right. You can change the position of a state by dragging and dropping the corresponding rectangle. Change the motors speed of these states (forward state -> motors speed: 45|45, stop state -> motors speed: 0|0). Now, you will create your first transition. Click on the new transition button. Create a link from the "forward" state to the "stop" state (the direction is important!). In this transition, you can specify under which condition the automaton can pass from the "forward" state to the "stop" state. Select this transition by clicking on its text field. It becomes yellow. Rename it to "front obstacle". By dragging the two highest red sliders, change the conditions values over the front IR sensors to have ">5" for each of them. You should obtain an automaton as this which is depicted in the figure called "First automaton". Select the initial state (the "forward" state) and test this automaton on your virtual e-puck by clicking on the upload button.

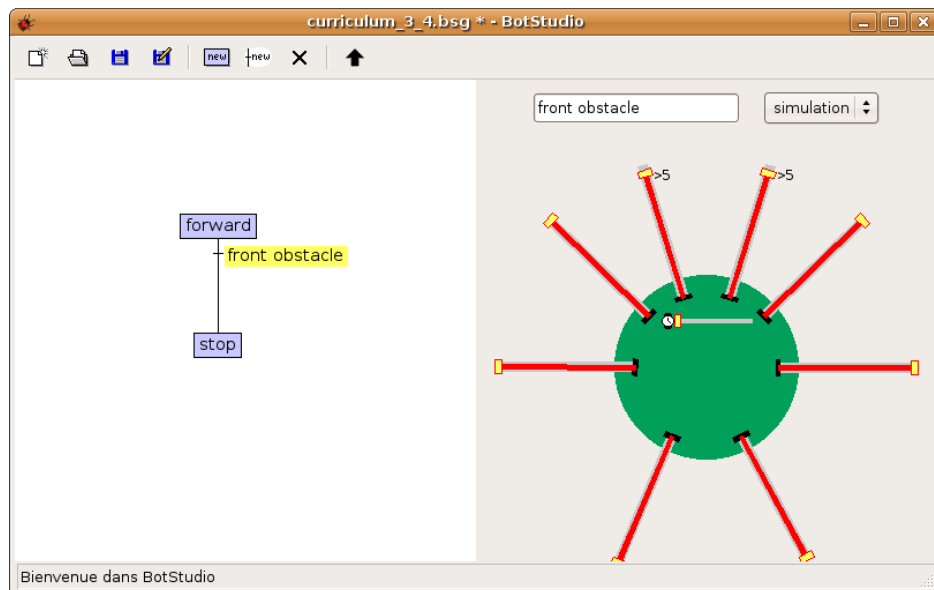


Figure 6.5: First automaton



[Q.1] What is the e-puck behavior?



[Q.2] In the "forward" state, which actuator command is used? Which condition over the IR sensors values are tested in the "front obstacle" transition?



[P.1] Execute the same automaton on the real e-puck.

You finished your first collision avoidance algorithm, i.e., your e-puck doesn't touch any wall.

This kind of algorithm is a good alternative to collision detection algorithm because a collision can engender a robot's degradation. Of course it isn't perfect, you can think about a lot of situations where your robot would still touch something.

U-turn

In this subsection, you will extend your automaton in order to perform a U-turn (a spin on itself of 180 degrees) after an obstacle's detection. Add a new state called "U-turn" to your automaton. In this state, set the motors speed to 30|-30. Add a transition called "timer1" from the "stop" state to the "U-turn" state. Select this transition. This time, don't change the conditions of the IR sensors but add a delay (1 s) to this condition by moving the yellow slider in the middle of the green circle. The figure called "The timer condition" depicts what you should obtain.

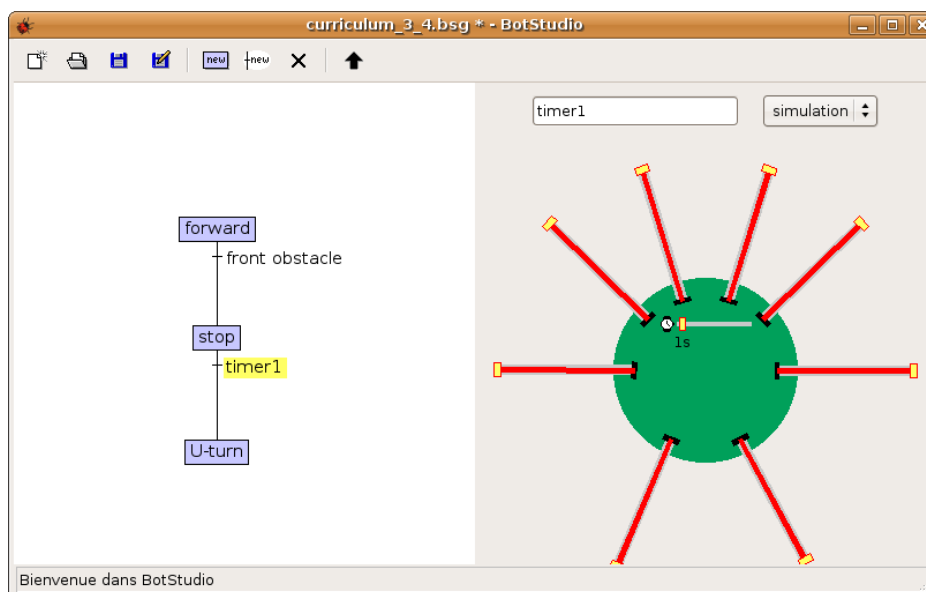




Figure 6.6: The timer condition

 [P.2] Run the simulation (always with "forward" state as initial state) both on the virtual and on the real e-puck.

For performing a perfect U-turn, you still have to stop the e-puck when it turned enough. Add a new timer ("timer2") transition from "U-turn" state to "forward" state (see next figure).

 [Q.3] With which delay for the "timer2" transition does the robot perform a perfect U-turn? Is it the same for the real robot? Why?

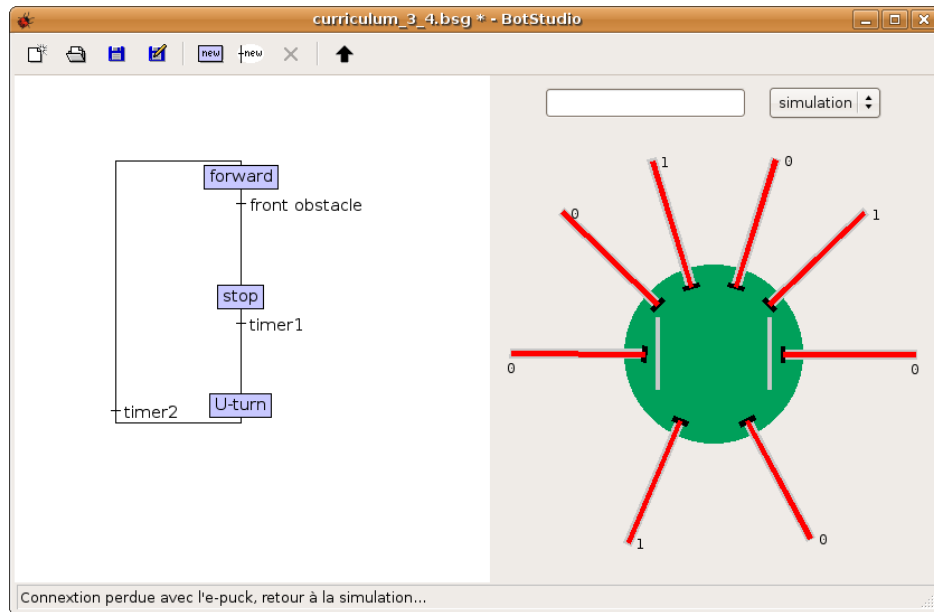


Figure 6.7: A loop in the automaton



[Q.4] You created an automaton which contains a loop. What are the advantage of this kind of structure?



[Q.5] Imagine the two following states: the "forward" state and the "stop" state. If you want to test front IR sensors values for passing from the "forward" state to the "stop" state, you will have two possibilities: either you create one transition in which you test the two front IR sensors values together or you create two transitions in which you test independently the two IR sensors values. What is the difference between these two solutions?

During this exercise you created an automaton step by step. But there are still several BotStudio tricks that are not mentioned above:

- For switching from "bigger than" to "smaller than" condition (or inversely) for an IR sensor, click on the gray part of an IR sensor slider.
- If you don't want to change the speed of a motor in a state, click on the yellow rectangle of the motor slider. The rectangle should disappear and the motors speed will keep its precedent value.
- If you want to remove a condition of a transition, set it to 0. The value should disappear.
- There is a slider which wasn't mentioned. This slider is related to the camera. You will learn more about this topic in exercise [sec:Line-following].

Your Progression

Thanks to the two previous exercises, you learned:

- What is an FSM and how it's related with a robot behavior
- How to use BotStudio
- How to design a simple FSM

The following exercises will train you to create an FSM by yourself.

Better Collision avoidance Algorithm [Beginner]

At this step, you know what an automaton is. Now, you will reinforce this knowledge through an exercise of parameters estimation. The structure of the automaton is given (the states and the transitions) but it doesn't contain any parameter, i.e., the actuators commands and the conditions over the sensors values aren't set. You will set these parameters empirically.

Open the World File

Open the following world file:

```
.../beginner_better_collision_avoidance_algorithm.wbt
```

You may have to increase the size of the BotStudio window to see the entire automaton.

Collision Avoidance Automaton

Note that you can store your automaton by clicking on the save as button in the BotStudio window. You can also load it by clicking on the load button.



[P.1] Start with the given automaton (see the figure). There is only its structure, i.e., there are states and transitions but their parameters aren't set. Find the parameters of each state and each transition such that the e-puck avoids obstacles.

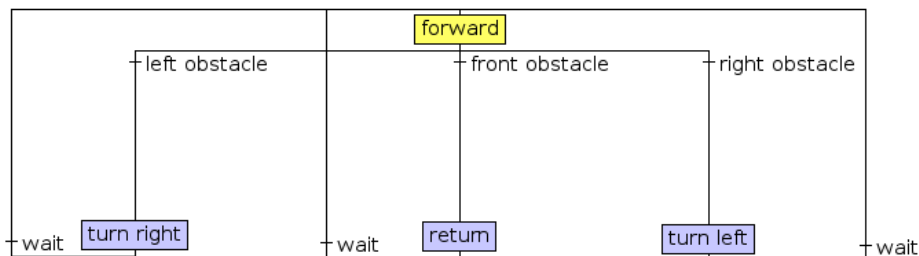


Figure 6.8: A better collision avoidance automaton



[P.2] Repeat the operation for the real e-puck.



[Q.1] Describe your method of research.

The blinking e-puck [Beginner]

Until now, you just have modified existing automata. It is time for you to create them. In this practical exercise, you will design your own automaton by manipulating LEDs.

Open the World File

Open the following world file:

```
.../worlds/beginner_blinking_epuck.wbt
```

Maybe you should increase the size of the BotStudio window for seeing the entire automaton. In the simulation window, if you don't see all the LEDs, you can move the camera around the robot by left clicking. For this exercise, working directly on the real e-puck is more convenient.

Modify an Automaton



[Q.1] Without launching the automaton, describe the e-puck behavior. Verify your theory by running the simulation.



[P.1] Modify the current automaton in order to add the following behavior: the 8 LEDs will light on and switch off clockwise and they will stay on for 0.2 s.



[P.2] Modify the current automaton in order to add the following behavior: when you cover up the real e-puck with your hands, the LEDs turn in the other direction. Use only 4 LEDs for this one (front, back, left and right).

Create your own Automaton

A way to design an automaton is first to identify the possible actuators configurations, to create a state for each of these configurations, to set the parameters of these states, to establish the conditions to pass from a state to another, to create a transition for each of these conditions and finally to set the parameters of these conditions. Unfortunately it is not always so easy. For example, in an automaton, it's possible to have two states with identical actuators commands. Indeed, if you see somebody who is running across a street without any context, you don't know if he's running to catch a bus or if he's leaving a building in fire. He seems identical but it's internal state is different.



[P.3] Create a new automaton (press the new graph button in BotStudio). Choose only the four following LEDs: the front LED, the back one, the left one and the right one. The goal is to switch on the LED corresponding to the side of the obstacle. Note that if there are obstacles on two sides of the robot, two LEDs should be on ! Don't do the case with an obstacle on three or four sides.



[Q.2] If I proposed to repeat the exercise by using the 8 LEDs around the e-puck and with all the cases up to 8 obstacles, would you do it? Why? Do you find a limitation in BotStudio?

***E-puck Dance* [Beginner]**

The goal of this exercise is to put the fire on the dance floor with your virtual e-puck by creating the e-puck dance. You can imagine dance as a succession of movements with the same rhythm. You will be able to model that easily in a finite state automaton.

Open the World File

Open the following world file:

```
.../worlds/beginner_epuck_dance.wbt
```

This opens a disco dance floor. Moreover, there is already a very little example of what you can achieve. I hope you will find a better e-puck dance than the existing one.

Imagine your Dance



[P.1] Observe the existing dance. The automaton has a loop shape. The time of every transition is identical. Create a new automaton or modify the existing one. First of all, choose a rhythm. The chosen rhythm of the example is a movement every second. It implies that every timer is set to 1s. Then, you should create a state for each movement you want to see during the global loop (note that if you want to have twice the same movement during the main loop, you have to create two states). Then, you have to set the states parameters according to your rhythm. Finally, link each state with a timer transition. (Hints: for producing a beautiful dance, LEDs are welcome. You can also perform semi-movements)

Line following [Beginner]

The goal of this exercise is to explore the last device available in BotStudio: the camera. With the e-puck camera, you can obtain information about the ground in front of it. BotStudio computes "in real time" the center of the black line in front of the e-puck. The camera is another e-puck sensor and the center of the front line is the sensor value of this camera.

A linear Camera

An e-puck has a camera in front of it. Its resolution is 480x640. For technical issues it has a width of 480 and a height of 640. The most important information for following a line is the last line of the camera (see the figure called “The field of view of the linear camera”). For this reason, only the last line of the camera is sent from the e-puck to Webots. Finally an algorithm is applied on this last line in order to find the center of the black line.

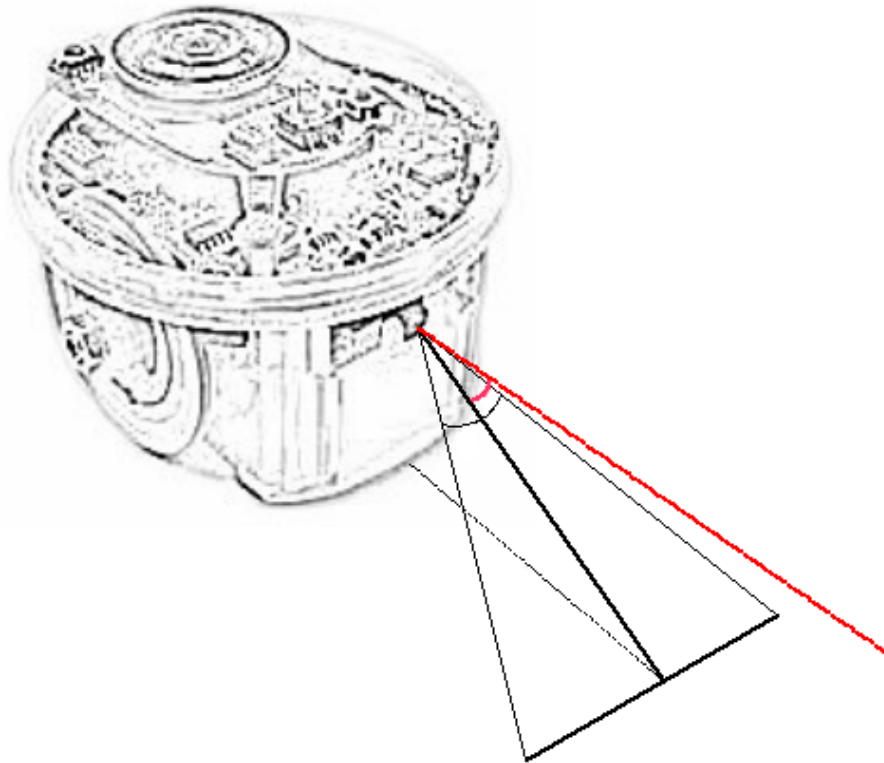


Figure 6.9: The field of view of the linear camera

The problem is that the e-puck sees only at about 5.5 cm in front of it, and sees a line of about 4.5 cm of width. Moreover, this information is refreshed about 2 times per seconds. It is little information!



[Q.1] Imagine that, every 5 s, at 4 m in front of you, you can only see a line of 3 m wide with always the same angle. What will be your strategy for following the line? Imagine that the line is intricate. What is the most important factor for following the line?

Open the World File

Open the following world file:

```
.../worlds/beginner_linear_camera.wbt
```

This opens a long world. A black line is drawn on the ground. Remark that there is a grain (Gaussian noise) on the e-puck camera values in order to be more realistic. Indeed, a real camera doesn't acquire perfect values. Some noise is always present on a real camera, it comes from several factors.

Line following Automaton

In the BotStudio interface, you will also find a condition over the camera (figure called "The linear camera condition in BotStudio") represented by a slider. The value represents the center of the black line in front of the robot. When a transition is selected, you can change the condition over the camera by dragging the slider, and you can change the direction of the test by clicking on the text field (ex: "<5" becomes ">5"). Remark that if there is no line in front of the robot, the center value can be wrong.



[P.1] Run the given automaton both in the simulation and in the reality. For creating a real environment, you can draw a line with a large black pen on a big white piece of paper (ex: A2 format). The black line must pass far from the paper bounds.



[P.2] Observe the direction (bigger than or smaller than) of the two conditions.



[P.3] Try to let go the e-puck as fast as possible by changing parameters of the states and of the transitions.

Rally [Beginner] [Challenge]

Open the following world file:

```
.../worlds/beginner_rally.wbt
```



[P.1] Create an automaton which can perform a complete turn of this path. (Hint: adapt your speed!)

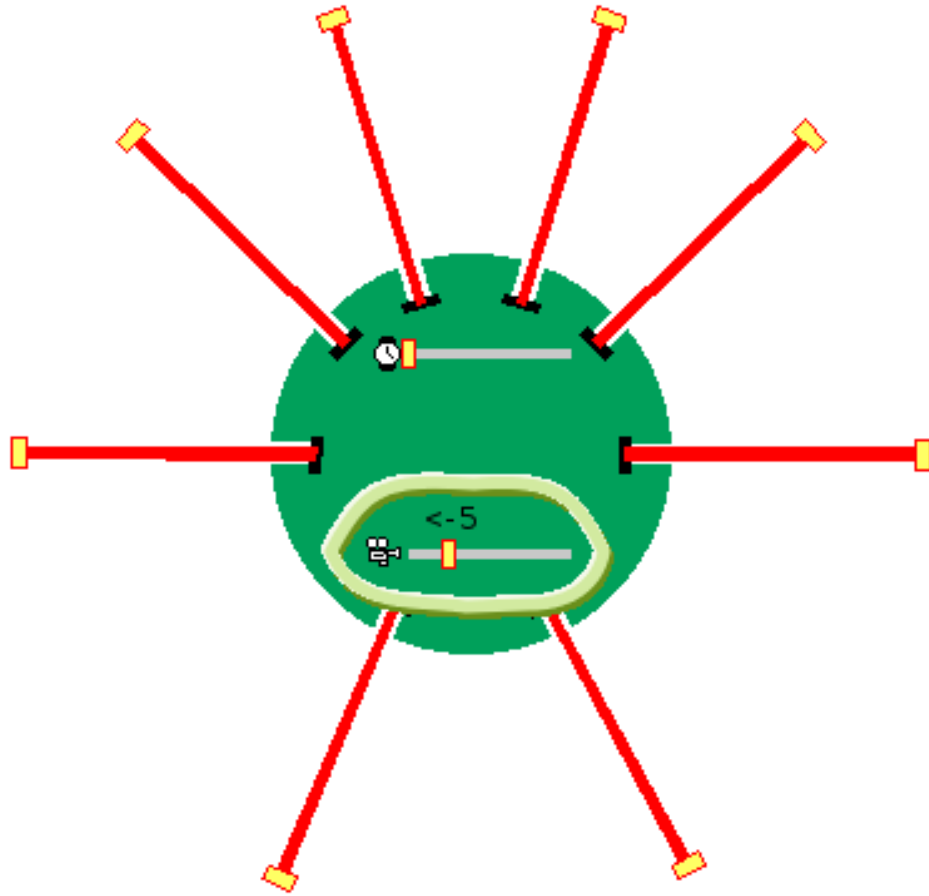


Figure 6.10: The linear camera condition in BotStudio

Appendix A

Document Information & History

History

This book was created on the [Wikibooks](#) project and developed on the project by the contributors listed in Appendix A, page 57. For convenience, this PDF was created for download from the project. The latest Wikibooks version may be found at http://en.wikibooks.org/wiki/Cyberbotics'_Robot_Curriculum.

PDF Information & History

This PDF was compiled from [L^AT_EX](#) on January 14, 2009, based on the 14 January 2009 [Wikibooks textbook](#). The latest version of the PDF may be found at http://en.wikibooks.org/wiki/Image:Cyberbotics'_Robot_Curriculum.pdf.

Authors

[Cyberbotics](#), Olivier Michel, Fabien Rohrer, Nicolas Heiniger, [DavidCary](#), [Trolli101](#), and anonymous contributors.

Appendix B

GNU Free Documentation License

Version 1.2, November 2002
Copyright © 2000,2001,2002 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “**Document**”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “**you**”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “**Modified Version**” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “**Secondary Section**” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “**Invariant Sections**” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “**Cover Texts**” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “**Transparent**” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “**Opaque**”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “**Title Page**” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “**Entitled XYZ**” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “**Acknowledgements**”, “**Dedications**”, “**Endorsements**”, or “**History**”.) To “**Preserve the Title**” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that

these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with ... Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.