Webots Robots Guide

release 6.0.0

`www.cyberbotics.com`

January 3, 2013

Trademark information

Foreword

The purpose of this guide is to give some clues for starting up programming some specific robots such as the AIBO^{TM}robot.

Disclaimer

These examples are not maintained. It is possible that this document contains obsolete information in comparison with the last Webots version.

Thanks

Cyberbotics is grateful to all the people who contributed to the development of Webots, Webots sample applications, the Webots User Guide, the Webots Reference Manual, and the Webots web site, including Yvan Bourquin, Fabien Rohrer, Jean-Christophe Fillion-Robin, Jordi Porta, Emanuele Ornella, Yuri Lopez de Meneses, Sébastien Hugues, Auke-Jan Ispeert, Jonas Buchli, Alessandro Crespi, Ludovic Righetti, Julien Gagnet, Lukas Hohl, Pascal Cominoli, Stéphane Mojon, Jérôme Braure, Sergei Poskriakov, Anthony Truchet, Alcherio Martinoli, Chris Cianci, Nikolaus Correll, Jim Pugh, Yizhen Zhang, Anne-Elisabeth Tran Qui, Grégory Mermoud, Lucien Epinet, Jean-Christophe Zufferey, Laurent Lessieux, Aude Billiard, Ricardo Tellez, Gerald Foliot, Allen Johnson, Michael Kertesz, Simon Garnieri, Simon Blanchoud, Manuel João Ferreira, Rui Picas, José Afonso Pires, Cristina Santos, Michal Pytasz and many others.

Moreover, many thanks are due to Cyberbotics's Mentors: Prof. Jean-Daniel Nicoud (LAMI-EPFL), Dr. Francesco Mondada (EPFL), Dr. Takashi Gomi (Applied AI, Inc.).

Finally, thanks to Skye Legon and Nathan Yawn, who proofread this guide.

Contents

Chapter 1

Using the IPR^{TM}robots

The goal of this chapter is to explain how to use the Webots model of the IPR robot. The IPR is a light-weight grip arm robot developed by Neuronics (www.neuronics.ch), which has 6 degrees of freedom. You can use the Webots model exactly as you would use the real robot, in order to try your IPR applications without risk.

1.1 Introduction

We will assume that you already have a good knowledge of Webots and have followed the tutorials for modelling and programming.

For additional IPR documentation such as the robot specifications, the communication protocol specification or the use of the control programs, please refer to the documentation provided on the Neuronics website and in particular to the IPR user manual.

1.2 Running the simulation

Launch Webots and open the `ipr_cube.wbt` world, which contains a model of the IPR robot (figure 1.1) on a table. All of the IPR worlds are located in the `projects/robots/ipr/worlds` subdirectory. If you run the simulation, the robot will move the red cube on the top of the black box.

There are other worlds related to the IPR robot that you can use:

- In `ipr_collaboration.wbt` you will find a demo using two IPR robots which collaborate to move cubes over a longer distance.
- In `ipr_factory.wbt` you will find another demo using two IPR robots which grab industrial pieces from a conveyor belt and place them into slots.

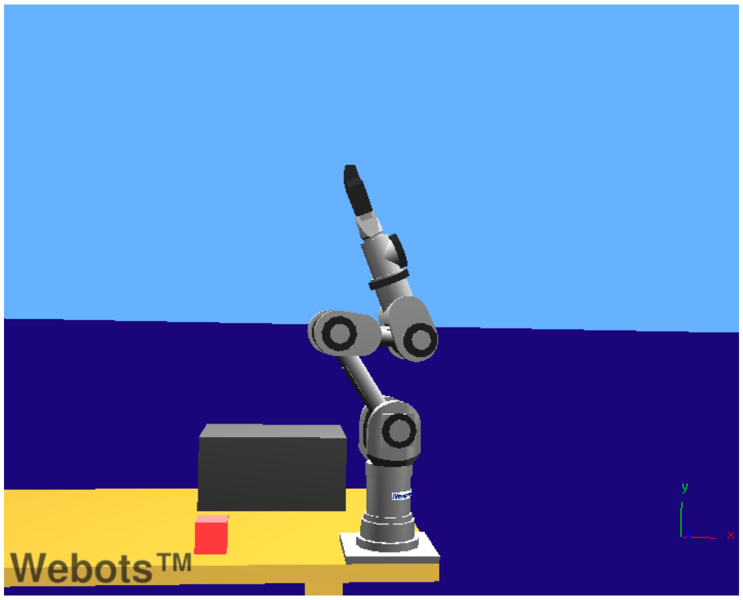

Figure 1.1: The ipr_cube.wbt world

- The `ipr_models.wbt` world contains the four different IPR models available in Webots, so that you can reuse them in your own simulations.

For each model of the IPR robot, you can access its control panel () by selecting the robot and choosing **Show robot window** in the **Simulation** menu, or simply by double-clicking the robot (if the robot window is already open, double-clicking the robot opens the scene tree window instead). The control panel contains two different parts:

1. At the top you have the controller associated with the robot (which should be `ipr_serial` for the standard worlds).
2. The lower part contains a slider for each motor of the IPR robot, next to the motor's name. You can use these sliders to manually move the IPR robot. Please note that you will need to be running the simulation for the motors to be able to move when you change the slider's position.

You may notice that even though in the real robot there is only one motor for the gripper, in the control panel (and scene tree as well) there are two motors associated with it (`left_gripper` and `right_gripper`). This is due to Webots' modelling of the gripper as two servos; this is totally transparent when controlling this motor with the controller.

1.3 Understanding the simulation model

As you can see on the Neuronics web site, there are many different types of IPR robots. Webots provides you with models of four of these types. If your desired type is not implemented, you can easily modify one of these to model it.

1.3.1 The different types of models

The IPR HD6M180 model

This model, which is equipped with a ”motor 1” of type ”normal” and a gripper, is the one used in the `ipr_cube.wbt` world (figure 1.1). This is the standard IPR robot.

The IPR HD6Ms180 model

This model, which is equipped with a ”motor 1” of type ”strong” and a gripper, is the one used in the `ipr_collaboration.wbt` world (). The differences between the normal and the strong versions of the same model are mainly the torque of the motor 1 and the size of the pedestal.

The IPR HD6M90 model

This model, which is equipped with a ”motor 1” of type ”normal” and a gripper, is the one used in the `ipr_factory.wbt` world ().

The IPR HD6Ms90 model

This model, which is equipped with a ”motor 1” of type ”strong” and a gripper, is one of the models contained in the `ipr_models.wbt` world ().

1.3.2 The model parameters

Open the scene tree window, then open the `Robot` node. This node represents the IPR robot. One of the last fields of this node is called `controllerArgs`, and is used to pass arguments to the robot controller.

The `ipr_serial` controller accepts several arguments, separated by blank spaces:

- The first one can be either ”normal” or ”strong”, in order to determine the type of motor 1. This argument is mandatory.
- The second argument is optional, and is used to specify a port number on which the TCP/IP server will listen for an incoming connection. If this argument is omitted, the TCP/IP server will connect using the standard port number. To find this number, you can either look at the `ipr_tcpip.c` file or launch a simulation in which you omit this argument, and look at the server’s messages in the Webots log window.

- The third argument is also optional; it specifies the name of a client program which should be launched by the controller. Note that the name of the program must be preceded by its relative path to the `ipr_serial` controller directory.
- The rest of the arguments are also optional, and are used only after the third argument (the name of the client program). They will be passed as arguments to the client program. All arguments placed after the name of the client program will be its arguments.

Please note that it is important to respect this order for the arguments.

For an example of the use of these arguments, please have a look at the corresponding node in one of the demo worlds (either `ipr_cube.wbt`, `ipr_collaboration.wbt` or `ipr_factory.wbt`).

1.3.3 Modifying the model

As the real IPR robot may show some small differences from the given models, it could be useful to tune the given models to your needs.

Most of the parameters that you would need to change are in the code of the controller. You can access the corresponding code sections easily by searching for the modification tag in the code. These modification tags are shown in parenthesis to the right of the titles of the following subsections.

Modifying the sensors (SENSORS_MODIFICATION)

If you want to change the way a particular sensor is responding, you will need to change the corresponding lookup table of the sensor node in the scene tree. For this modification, you will not have to change the source code of the controller.

If you want to add, remove or change the type of a sensor, you will first have to change the corresponding sensor node in the scene tree. Then you will have to make the corresponding changes in the source code of the controller.

For more information on how to modify the source code of the controller, please refer to the code sections identified by the `SENSORS_MODIFICATION` tag in `ipr_serial.c`.

For an example on how to add a sensor, please see the `ipr_factory.wbt` world.

Modifying the torque values (TORQUE_MODIFICATION)

All the models are initialized using the optimal values, both for the normal and the strong models. Nevertheless, if you want to modify the torque of the motors you will have to change the corresponding values in the controller source code.

For more information on the values used and on how to modify the torque values, please refer to the source code section identified by the `TORQUE_MODIFICATION` tag in `ipr_serial.c`. For more information on the standard values for torque, please refer to the IPR user manual.

Adapting the support of the model

Affixing the robot to the ground with a big plate of metal, as in the models, is probably not the most common way to do things with the real robot. As you can see in the demo worlds, the models can be placed on a table (). If you want to place a model on a table, the easiest way to do it is to copy an entire `Robot` node from a provided world (either `ipr_cube.wbt`, `ipr_collaboration.wbt` or `ipr_factory.wbt`) into a new world, and then replace the whole robot with the model you need.

Please note that when we talk of the ”whole robot”, this means the `PEDESTAL`, `LOGO` and `ROBOT` nodes from the children node, and also all the unnamed nodes from the `boundingObject` field of the `Robot`. All the other nodes are part of the support of the model. Please also note that in order to avoid DEF/USE problems, it is highly recommended to do this modification by editing the corresponding Webots worlds as text files, and to perform copy and paste operations.

If you want to adapt the table of your model, it is quite simple. The `Robot` position is defined as the center of the base of the model ().

- If you want to move the robot and the table position, simply change the `Robot` position.
- If you want to move the robot position on the table, simply move the model of the table around the robot.
- If you want to change the properties of the table (like its height, its length, etc.), simply adapt the model of the table to your needs and adapt the `Robot` position to these changes.

Please note that in order for the model to work properly, you must apply the same changes both to the table model and to its corresponding `boundingObject`.

Finally, there are a few things you must be aware of when creating a new support for your model:

1. As the table is part of the `Robot`, and as a `Robot` node can represent only one IPR robot, it is not possible to put more than one model on the same table. Nevertheless, you can create as many tables of varying size as you want.
2. As both the pedestal of the IPR robot and its support are part of the same node, they have the same `Physics` node too. So, when setting values for this node, you will have to take these two parts into account. Please note, for the `weight` parameter, that the weight of the pedestal alone is 1270/1430 [g] for the normal/strong models respectively.

3. You can use the USE functionality of Webots to create copies of your IPR model. This allows you to easily create exact copies in only a few clicks, in a flexible manner (the modifications applied to the original model are also applied to the copy).

For more information on working with multiple IPR robots, please see the `ipr_collaboration.wbt` world and the `ipr_collaboration.c` client file.

Modifying the node fields

The IPR models were already given the optimal values for these parameters, and we advise you not to change them. Nevertheless, it may be useful to do so, for example if you want to model a different robot.

Modifying the model more deeply (MODEL_MODIFICATION)

If you want to make more significant changes to the models, like changing the gripper's type or creating a new model, you will have to change the scene tree organization, and various parts of the code. This is a complex task that will require you to understand the model, and most of the controller source code as well. During the modification of the model you will probably also modify some parts which have been discussed in the previous subsections.

For more information on the controller source code, please start by reading the header files, especially `ipr_protocol.h`, then refer to the parts of the code in `ipr_serial.c` identified by the `MODEL_MODIFICATION` tag.

1.4 Different applications

There are various ways of using the IPR models within Webots. Depending on what you want to do, you will have to choose the most suitable one.

1.4.1 Controlling the IPR model using Neuronics software

You can use the same software, for example Katana4D, that you are using to control your real IPR robot to control the Webots model. To do this, you only need a version of this software with the capability to connect through TCP/IP. As we have seen previously, to configure Webots you only have to choose the correct port number for the server. To configure the Neuronics software, you only need specify the same port number and the IP address corresponding to the computer running Webots. Then, controlling your model is as easy as launching your world in Webots and running the simulation. Now you can use the Neuronics software exactly as you would do when controlling a real IPR robot.

Please note that if you are running both programs on the same computer, the IP address will be 127.0.0.1 (the address of the ”local host”).

In order to find out if your version of the Neuronics software has the TCP/IP option, please refer to your software specifications. For more information on how to use the Neuronics software, please refer to the corresponding User Manual.

1.4.2 Controlling the IPR model using the Neuronics programming library

Neuronics also provides programming libraries, for example the KNI library. If you want to use them to control your Webots model, it is as simple as linking in these libraries at compilation time, either dynamically or statically, and using the desired functions directly in your Webots controller code. Please note that in order to use a given library, your program must be written in a compatible language. This means that depending on the language used by the library you want to link, you may have to change the language used by your Webots controller program (e.g. C to C++). Please also note that these libraries are not provided with Webots, and you will have to install them separately before using them.

1.4.3 Controlling the IPR model using a TCP/IP client

You can also use a TCP/IP client program to directly control the Webots model. This is exactly what has been done to create the included demonstrations. If you want to use this method to control your model, you will have to write a client controller, put it in the clients subfolder of the ipr_serial controller and compile it. You will also have to send the name of your client to the controller using `controllerArgs`, as seen earlier in this chapter. Then, simply run your Webots world and the client will be launched.

1.4.4 Controlling the IPR model using a standard Webots controller

The `ipr_serial` controller provided by Webots implements a TCP/IP server. This is just one of the many possible ways to use the interface provided by `ipr_serial.h` to control the IPR model. You can easily replace the TCP/IP server by your own control code and use this interface to suit your needs.

For more information on the interface provided, please see `ipr_serial.h`. For an example of how to use this interface, please see `ipr_tcpip.c`.

1.5 Troubleshooting

The model is unable to grab a given object

If your are not able to grab an object because it is too ”slippery”, you will have to increase its `coulombFriction` value, which can be found in its `Physics` node.

The model and base tip over when the robot moves

Webots works with a physics simulator, so consider the stability of your base when creating it. To improve the stability of your base, you can either increase its mass value (in the `Physics` node), add a weight to compensate for the weight of the IPR robot (as in `ipr_cube.wbt` or `ipr_factory.wbt`) or change the positions of the legs of the table.

The model crashes when it should not

The crash detection used in the IPR model relies on the crash detection used by the real IPR robot. The sensitivity of the crash detection is given by the parameters `crashlimit` and `crashlimit_linear`. If your Webots model crashes too often, please try to change these parameters. For instructions on how to modify these parameters, please refer to the section on the ”`S` command” in the IPR user manual. For more precise information on crash detection, please refer to the corresponding chapter of the IPR user manual.