1 Working with Webots

This guide gives instructions on how to upload code from the Webots environment (fresh installation) to the e-puck robot. Before proceeding, please note that the default installation directory of the Webots program is problematic, and it is important to install the application directly to the C: under windows, or some other controlled directory in linux, at least for the purposes of this guide.

1.1 Installing Webots

Install the webots program to the C: and NOT to the default installation directory. The latest version is always available from their website. If you install webots directly into C: \Webots, the cross-compilation steps detailed below will be drastically easier, since the configuration provided in the e-puck resources folder will already work.

1.1.1 A Note on Dongles

The dongles included with the e-puck robot contain valid Webots licensees. If a dongle is in a USB drive on your computer, you will have complete access to all of the features of Webots .edu edition; if the dongle is not present then you will only be able to use the free version.

1.2 Loading the World

A world has been created for webots, and is located in the e-puck resources folder (The folder where this document is located). It can be found in Webots/my_project/worlds and it is called unmEpuckDevelopment.wbt.

To load the world, simply boot Webots, go to file - open world, and then navigate to the directory that contains the e-puck resources folder.

1.3 The Controllers

There are two controllers loaded into the project you have just loaded, lightFollower.c and movingBlinker.c. For information about what they are capable of doing, please refer to the .c files associated with each controller, as they are well documented. You can plug controllers into epucks in the world by selecting the puck in the Scene Tree, and changing its controller variable to the controller of your choosing.

You can also modify and create controllers usings webots. Please refer to the webots tutorial on their website for more information about how to do this.

1.4 Cross - Compliation

In order to get your code working on the e-puck, your source code MUST be written in C and a couple of steps have to be taken to ensure the process succeeds. Basically, for cross compilation to be possible, your controller must have a file called 'Makefile.e-puck'

stored in the same directory as the controller's source code. In the case of lightFollower and movingBlinker, the Makefile.e-puck already exists. The makefile tells the compiler where the Include list for compiling webots code is located. This file is part of the Webots install, and is always located in the Webots installation directory. If you installed Webots to C: \Webots, then you shouldn't have to do anything else. If, however, it is installed in another location or you're running linux, you will need to edit the Makefile.e-puck file to point to the correct location. It needs to have the following line:
include [webots-directory] \transfer \e-puck \libepuck \Makefile.include
where webots-directory is the path to your webots installation.

If cross compilation is set up correctly, you will see the cross compilation option become available in Webots. If you have a controller's source code open, you will see several icons above the source code, including new, open, save, save as, reload, etc. To the right of these icons, there are several gear and broom icons. The second gear icon (the one with the right arrow), is the cross-compile button. Pressing this button will compile the currently selected controller for the e-puck, and put the resulting .hex file into the same directory that the controller's source code is located in.

1.5 Uploading .hex files to your puck

In order to upload code to the e-puck, you will need to first pair the e-puck with your bluetooth radio. This is different in every operating system, but is relatively straightforward in windows. You can always pair the puck using the 4-digit number attached to the robot with a sticker as the paircode. Make sure the puck is on while you do this.

Once your puck is paired, typically two ports will be opened between the puck and the computer, one COM port for outgoing communication, and one for incoming. You will need to identify which port is the outgoing port. In windows, there should be a bluetooth icon on your system trey, and if you right click on it you can open settings. If you click on the COM Ports tab, the open ports will be enumerated along with their name, and you should be able to identify the port that your puck is using.

Once you know which port your epuck is using for outgoing communications (those communications initiated by the computer), uploading the code to the puck is easy.

1. Double click on the e-puck in your simulator (the actual 3d model of the device).
2. Click the Refresh Ports button.
3. Click on Upload ...
4. Select the COM port your puck is using for outgoing communications.
5. Press Okay.
6. Once communication is established, a select file dialog should appear. Navigate to the folder where you cross-compiled your .hex file to (see the Cross - Compilation section).

7. Press Okay.
8. You may be prompted to press the reset button; if so, do it.
9. The upload should begin and a progress bar will appear.
10. Keep the puck close to the bluetooth transmitter on your computer. It isn't pretty if it gets out of range during the upload.
11. Once complete, the puck should immediately begin running the new code.