Tutorial for programming the e-puck robot in linux

Xavier Raemy
SWIS group
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Project web page: `http://www.e-puck.org`

November 22, 2006

1 Introduction

The goal of this tutorial is to learn how to program the e-puck robot using the bootloader script in a Linux environment.

Before running this tutorial you have to install the Bluetooth packages for your Linux distribution, the Linux C30 GNU C compiler and the `epuckupload` script. These are all available on the e-puck website (`http://www.e-puck.org`) with the necessary information for installing them. These installations are not covered in this tutorial.

2 Tutorial A: FlashLed

This tutorial is built around the program `ledtest.c` written in C and using the standard e-puck library. The program lets one of the LEDs flash. The source file is used with a Makefile and the e-puck developement tree (see below) to form a complete project. There are four steps in this tutorial:

1. Get the developement tree.
2. Create the project and edit the code.
3. Compile and link the code.
4. Program the robot's microcontroller using `epuckupload`.

2.1 Getting the E-puck Developpement Tree

On Windows, users have a IDE[1] that takes care of linker scripts, paths, and so on. Under Linux, everything is managed by `make` with the associated Makefile. Editing a Makefile is not recommended for those unfamiliar with the syntax. Thus to avoid unnecessary complications, we strongly reccommend that you use the provided Makefile and directory structure, which we will refer to as "the development tree". The tree may be obtained from the developement repository (if you have access to it), the e-puck website, or with the files provided for this TP on the internal course website.

Download the archive and extract it on a location of your choice. We will refer to this location as `${epuckroot}`. The tutorial project is at

```
${epuckroot}/e_puck/program/tutorial/linux_tutorial
```

[1] Integrated Development Environment

2.2 Create the project and edit the code

The first step is to create a new project.

A project contains the files needed to build an application (principally the source code and the Makefile).

Go to `${epuckroot}/e_puck/program/` and copy the `./tutorial/linux_tutorial` directory to a new directory `./my_first_tutorial` in the `program` directory.

The first line of the provided Makefile is `PROG = linux_tutorial`. This is the name of the project and will be the executable file name. Change it to `my_first_tutorial`.

The second line of the Makefile is `EPUCKLIBROOT = ../../../library`. This line informs `make` of the location of all the necessary libraries and required linker files. Because you have just copied the project to a new directory that is one level higher, you need now to remove one `../` (so that it becomes `EPUCKLIBROOT = ../../library`).

Save the file. Now your project is ready for compilation.

2.3 Building the Code

The code (`main.c`) contains the lines `#include "./motor_led/e_init_port.h"` and `#include "./motor_led/e_led.h"` which means that our code uses two parts of the e-puck library:

- `e_init_port.h` contains the API for a set of functions that configure the I/O ports of the microcontroller.
- `e_led.h` contains the API for a set of functions facilitate control of the LEDs.

Feel free to inspect these files. All of the most common usage questions can answered by reading the comments included in these files.

In this project, building the code consists of compiling the file `testled.c` to create an object file, `testled.o` and then linking that file with the library to create the output file `my_first_tutorial.hex`. The `.hex` file contains the compiled program in a format that can be uploaded on the e-puck.

Before building the project, you need to generate object files (*.o) for each part of the library you use. Go to `${epuckroot}/e_puck/library/motor_led` and type `make`. This should produce the object files in that directory, notably `e_init_port.o` and `e_led.o` which contain the implementations of the functions we have referenced from `e_init_port.h` and `e_led.h` in our project. As we will not be modifying the library, this compilation is only necessary once. Now that it has been done (and the object files are available) we will not need to recompile them for future projects.

Go to your project directory and type `make`. This should generate a file named: `my_first_tutorial.hex`. This is the raw machine code you will upload to the microcontroller on the e-puck robot.

2.4 Programming the dsPIC using epuckupload

Now that you have the `.hex` file, you are ready to upload it to the robot.

The e-puck robot uses a bootloader to program itself. A bootloader is a small program located in a protected area of the microcontroller's program memory. Just after power up or reset, the microcontroller will first execute the bootloader code regardeless of the user code or the robot configuration. It will listen on the Bluetooth channel for a specific command telling it to reprogam the robot. If this command is not received before a (short) timeout, the bootloader will instead load the user code. For those familiar with lilo or grub, this procedure is essentially the same. If we want to upload a new firmware, we need to be connected to the robot and to send the "reprogram" command during this timeout period. In practice, we achieve this by sending the "reprogram"

command continuously and pressing the "reset" button on the robot. This is what is done by `epuckupload`.

To upload a program on robot 121, go to your project directory and type

```
epuckupload -f my_first_tutorial.hex 121
```

After a couple of seconds, you should see an orange LED that lights up on the robot (see Figure 1) which indicates that you are connected to it via Bluetooth. Then, a series of dots will appear in the terminal, indicating that we are sending the "reprogram" command to the robot. You may now reset the robot, and the upload should begin (the dots become stars in the terminal).

After several seconds, the script indicates success (or error) and exits. You robot is programmed and will immediately start executing your code.

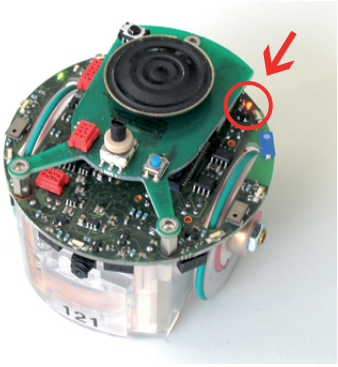

Figure 1: the e-puck bluetooth LED