Tutorial for programming the e-puck robot using the bootloader via bluetooth

Francesco Mondada and Michael Bonani
Autonomous Systems Laboratory
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Project web page:`http://www.e-puck.org`

February 24, 2006

1 Introduction

To run this tutorial you need to have an e-puck with a bootloader on it and a computer equipped with Bluetooth (or a USB Bluetooth adapter). If you do not have the bootloader on your e-puck robot, you have to program it using for instance the ICD2 programmer (please look at the corresponding tutorial).

The goal of this tutorial is to learn how to program the e-puck miniature mobile robot using the bootloader via a Bluetooth connection from a Windows environment. It is based on version 7.21 of MPLAB IDE software and on version 1.33 of the C30 GNU C compiler, both running under Windows. Source files are associated to this tutorial and should be available at the same place where you downloaded this file.

Before running this tutorial you have to download and install the MPLAB IDE software and the C30 GNU C compiler, both from microchip. MPLAB IDE is a free program. The C30 compiler is a commercial program but can be used with very few restrictions in student mode for free. You can download these programs in the section `Development Tools` of the web site `http://www.microchip.com`.

You have also to download the `Tiny Bootloader` program on the web site

`http://www.etc.ugal.ro/cchiculita/software/picbootloader.htm`.

2 Tutorial A: FlashLed

This tutorial is built around the program `ledtest.c` written in C. The program just flashes a LED. The source file is used with a linker script file (p30f6014a.gld), an initialization file and its definition files (`init_port.c`, `init_port.h` and `epuck-ports.h`) to form a complete project. The tutorial is a simple project. More complex projects might use multiple assembler and C source files as well as library files and precompiled object files. For simplicity, this tutorial uses only two source files. There are four steps to this tutorial:

1. Create a project in MPLAB IDE.
2. Compile and link the code.
3. Program the robot via Bluetooth.

2.1 Creating the project

The first step is to create a project and a workspace in MPLAB IDE. Usually, you will have one project in one workspace.

A project contains the files needed to build an application (source code, linker script files, etc.) along with their associations to various build tools and build options.

A workspace contains one or more projects and information on the selected device, debug tool and/or programmer, open windows and their location, and other IDE configuration settings.

MPLAB IDE contains a Project Wizard to help create new projects. Before starting, create a folder for the project files for this tutorial (`C:\Tutorial` is assumed in the instructions that follow). From the TP web pages, copy the zipped file into your `C:\Tutorial` folder and decompress it.

2.1.1 Select a Device

1. Start MPLAB IDE.
2. Close any workspace that might be open (File>Close Workspace).
3. From the Project menu, select Project Wizard.
4. In the Welcome screen, click Next> to go to the Project Wizard Step One dialog (Figure 1)

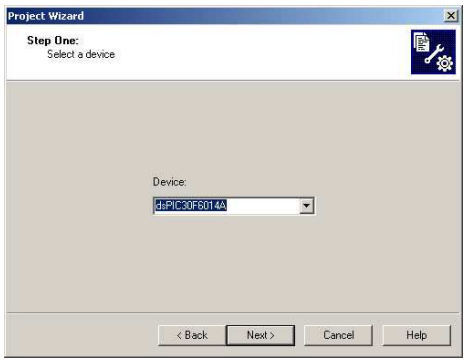

Figure 1: Project wizard, step 1, select a device.

5. From the Device: pull-down list, select dsPIC30F6014A and click Next>. The Project Wizard Step Two dialog displays (see Figure 2).

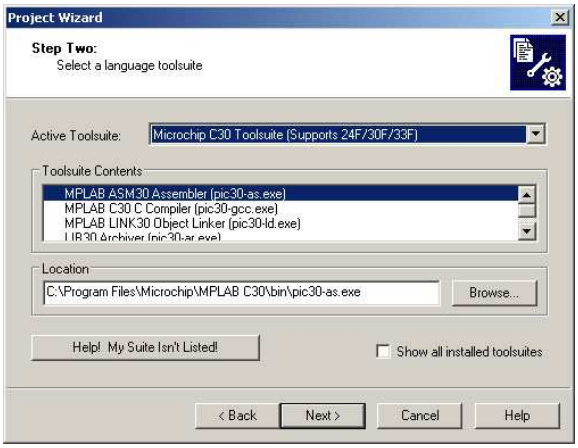

Figure 2: Project wizard, step 2, select language toolsuite.

2.1.2 Select Language Toolsuite

1. From the Active Toolsuite: pull-down menu, select Microchip C30 Toolsuite. This toolsuite includes the compiler, assembler and linker that will be used.
2. In the Toolsuite Contents block, select MPLAB ASM30 Assembler (pic30-as.exe).
3. In the Location block, click Browse... and navigate to:
   `C:\Program Files\Microchip\MPLAB C30\bin\pic30-as.exe`
4. In the Toolsuite Contents block, select MPLAB C30 C Compiler (pic30-gcc.exe).
5. In the Location block, click Browse... and navigate to:
   `C:\Program Files\Microchip\MPLAB C30\bin\pic30-gcc.exe`
6. With MPLAB LINK 30 Object Linker (pic30-ld.exe) selected in Toolsuite Contents, click Browse... and navigate to: `C:\Program Files\Microchip\MPLAB C30\bin\pic30-ld.exe`
7. Click Next> to continue. The Project Wizard Step Three dialog displays (see Figure 3).

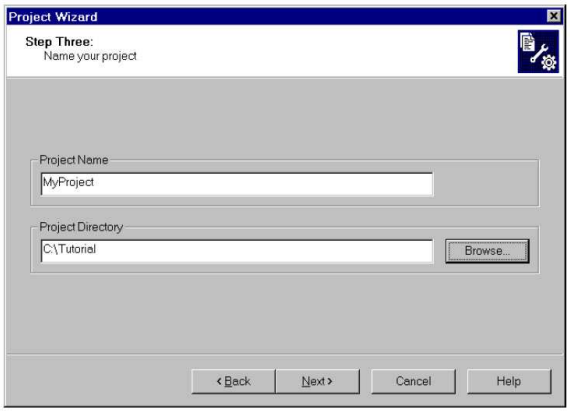

Figure 3: Project wizard, step 3, name your project.

2.1.3 Name Your Project

1. In the Project Name text box, type a name for your project, for instance `MyProject`.
2. Click Browse... and navigate to `C:\Tutorial` (or the directory you ahve chosen) to place your project in the Tutorial folder.
3. Click Next> to continue. The Project Wizard Step Four dialog displays (see Figure 4).

2.1.4 Add Files to Project

1. Locate the `C:\Tutorial` folder and select the `testled.c` file.
2. Click Add≫ to include the file in the project.
3. Add in the same manner the files `init_port.c`, `init_port.h` and `epuck-ports.h`.
4. Go to the `C:\Program Files\Microchip\MPLAB C30\support\gld` folder and select the `p30f6014a.gld` file.

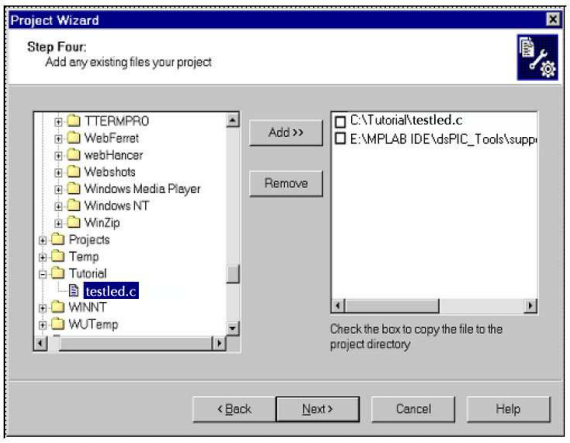

Figure 4: Project wizard, step 4, add files to project.

5. Click Add≫ to include the file in the project. There should now be two files in the project.
6. Click Next> to continue.
7. When the summary screen displays, click Finish.

After the project wizard completes, the MPLAB IDE project window shows the `testled.c` and `init_port.c` files in the Source Files folder, the `init_port.h` and `epuck-ports.h` files in the Header folder and the p30f6014a.gld file in the Linker Scripts folder.

Check that you have the correct include path under Project>Build Options ...>Project as illustrated in figure 5.

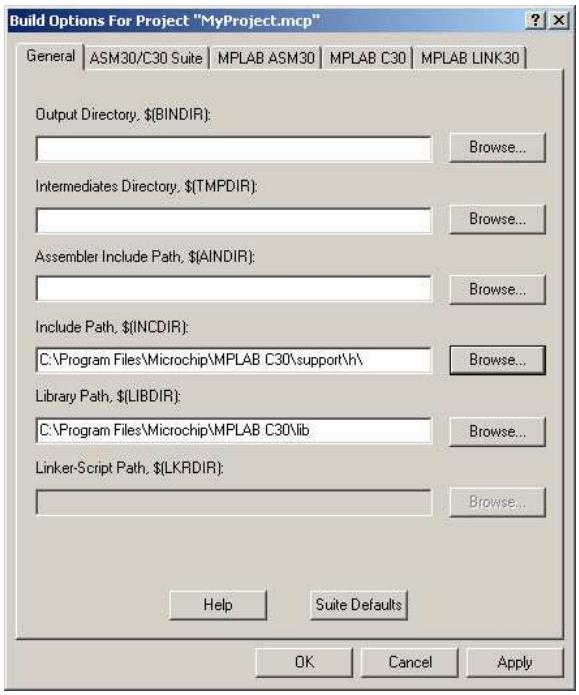

Figure 5: Path settings.

A project and workspace has now been created in MPLAB IDE. MyProject.mcw is the workspace file and MyProject.mcp is the project file. Double-click the `testled.c` file in the project window to open the file.

2.2 Building the Code

In this project, building the code consists of compiling the `testled.c` file to create an object file, `testled.o`, compiling the `init_port.c` file to create an object file, `init_port.o`, and then linking the object files to create the `testled.hex` and `testled.cof` output files.

2.2.1 Compiler parameters

1. Under Project>Build Options ...>Project please select the MPLAB LINK30 tab to view the linker settings (see Figure 6 left).

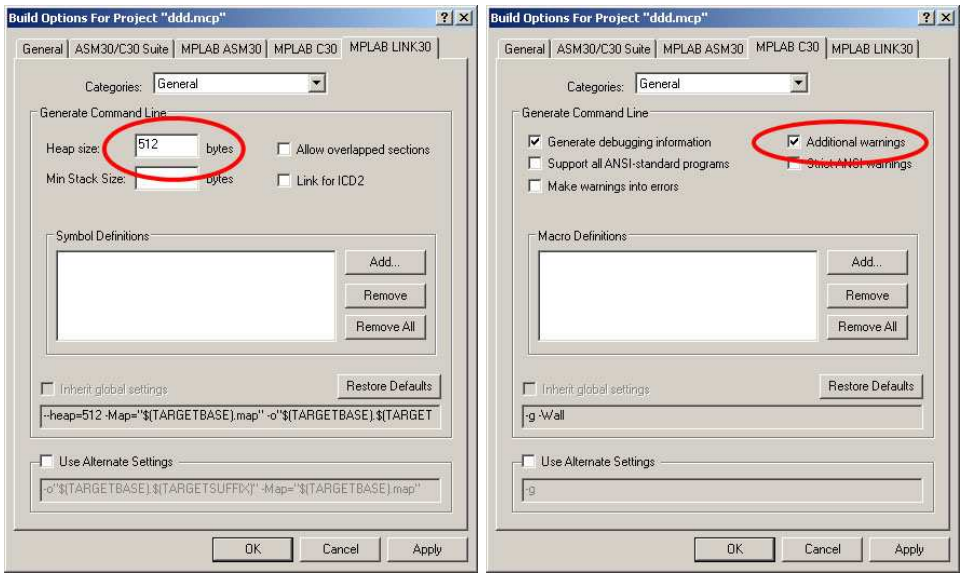

Figure 6: Left: MPLAB LINK30 build options. Right: C30 compiler options.

2. Add `512` in the `Heap size` box.
3. Please select the MPLAB C30 tab to view the compiler settings.
4. Check the box for `Additional warnings` (see Figure 6 right).
5. Click OK again to save these changes. The project is now ready to build.

2.2.2 Set Up the Device Configuration

1. Use the Configure>Configuration Bits menu to display the configuration settings.
2. Set up the configuration bits as shown in Figure **??**. The highlighted configuration settings may need to change to the these values:

   Oscillator: **XT w/PLL 8x**

   Watchdog Timer: **Disabled**

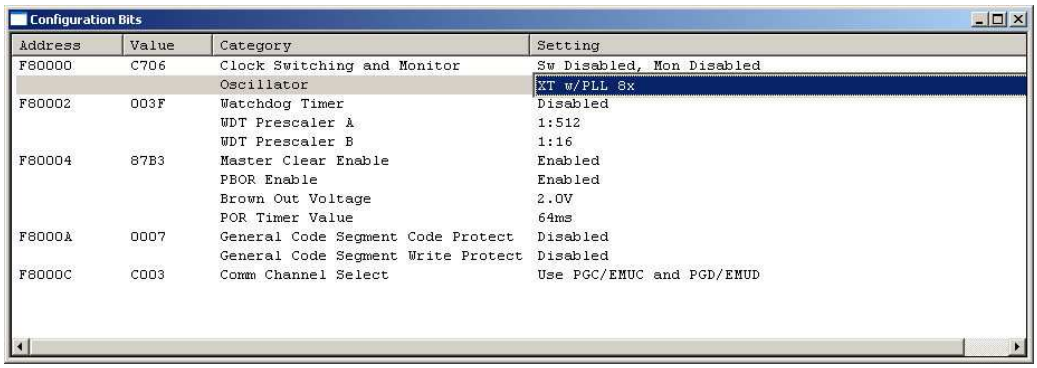

Figure 7: Configuration settings.

2.2.3 Build the Project

1. Select Make>Project menu to display the Build Output window (Figure 7).
2. Observe the progress of the build.
3. When BUILD SUCCEEDED displays you are ready to program the robot.

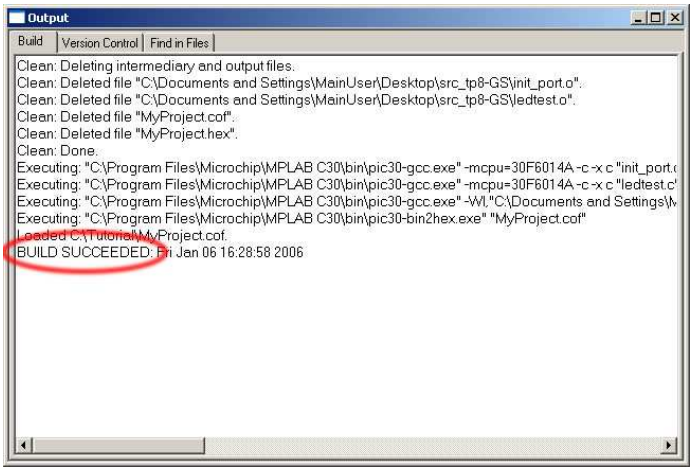

Figure 8: Build output window.

2.3 Programming the e-Puck robot

The e-puck robot will be programmed via a bluetooth connection. To do so, the first step is to pair the e-puck robot with your computer.

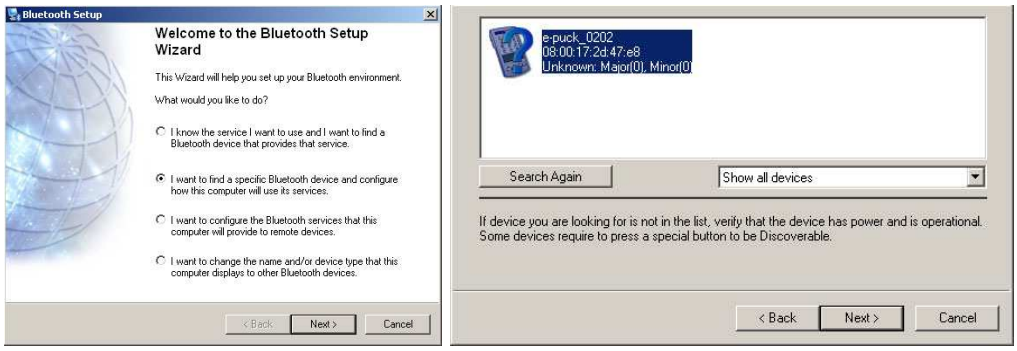

Figure 9: Left: Initial choice on the Bluetooth Setup Wizard. Right: Select your e-puck robot in the bluetooth devices found by your computer.

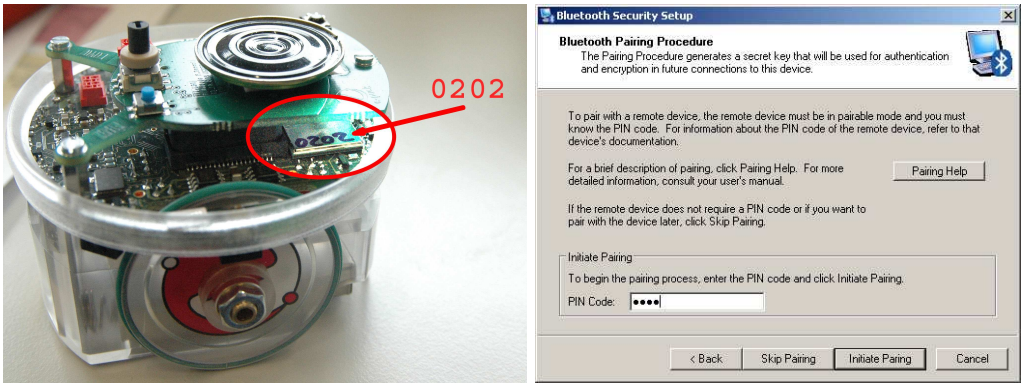

Figure 10: Left: The identification number (here 0202) of each e-puck is written on the metallic plate under the speaker extension. Right: Enter the pin code which is identical to the e-puck number (here 0202).

2.3.1 Pairing the e-Puck robot on bluetooth

1. Switch On your e-puck robot. The green power LED should become on.
2. On your computer, please start the `Bluetooth Setup Wizard`, as shown in figure 8 left.
3. As illustrated in figure 8 right, please choose the `e-puck_XXXX` device where XXXX is the number of your e-puck. The number is written on the metallic plate under the speaker extension. On figure 8 *XXXX* means *0202*.
4. As illustrated in figure 9 right, please enter the pin code to access to your e-puck. The default pin code is the same number as the number you have in the e-puck identificator. In the example given here, for the e-puck number *0202* the pin code is *0202*.
5. As illustrated in figure 10 left, select the COM service of the e-puck robot.
6. As illustrated in figure 10 right, select the WIndows COM port where your want to have your e-puck robot. In the example the choosen port is COM6. You will have to give this port to the program accessing to e-puck.
7. The pairing process should be terminated.

Depending on the Bluetooth drivers you have, the activation of the Bluetooth connection can be made by the program accessing it or has to be made manually by accessing to `My Bluetooth places` and double-clicking on the icon corresponding to the connection, as shown on figure 11.

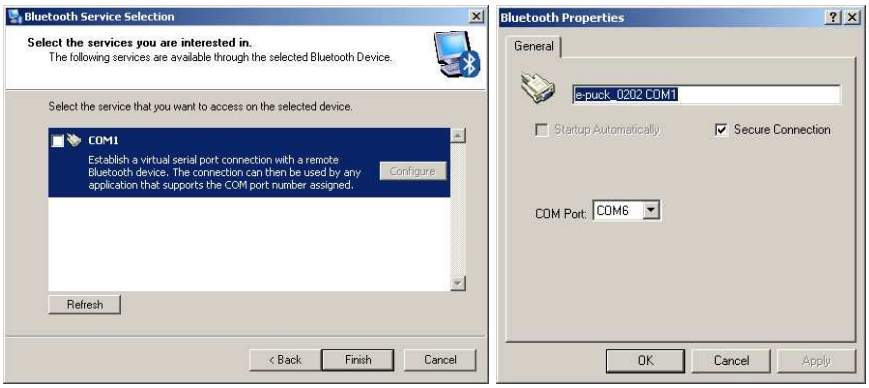

Figure 11: Left: Select the COM service of the e-puck robot. Right: Select the COM port where you want to have the e-puck robot.

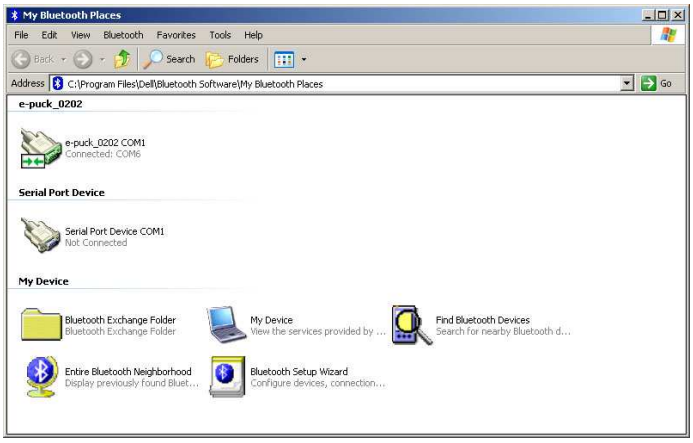

Figure 12: Activate the bluetooth connection by double-clicking on the icon corresponding to the connection.

2.3.2 Programmin the e-puck flash

1. Start the `Tiny Bootloader` program. You should get the window of figure 12.

2. Click on `Browse` to look for your `.hex` file. You should get the window of figure 13.

3. Select the COM port corresponding to your e-puck. If the COM port is not in the list, type it.

4. Click on the `Write Flash` button. When the program display the connection, the orange Bluetooth led on the e-puck should be on and you have five second to push on the blue reset button on the top of your e-puck. You can change this timing in the options.

5. When programmed, the LED0 should start blinking.

2.4 Modify and test new code

Change the code to:

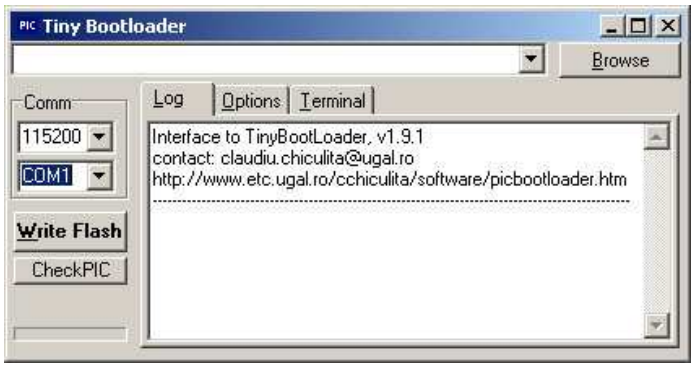

Figure 13: Initial window on the Tiny Bootloader.

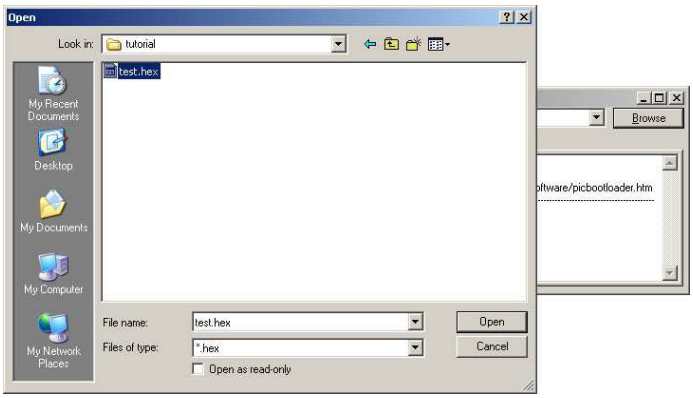

Figure 14: Browse to find the hex file corresponding to your project.

- Make a loop switching ON and OFF the 8 LEDs in a sequential way (LED0 ON and others off, then LED1 ON and others OFF and so on).
- Read the four bits coming from the "running mode selector" and replicate the value on four LEDs.

3 Tutorial B: Timer Interrupt

3.1 Introduction

The management of real time delays by loops of `nop` instructions cannot work with the presence of interrupts that can modify the delay generated. The management of real-time events by interrupt routines under the control of a timer is much more efficient and widely used. It is therefore important to know how to use this mechanism and how to implement it in C.

3.2 LED blinking by interrupt

The blinking of a LED can be implemented in a reliable way by using a timer that generates an interrupt in a regular way. The interrupt routine can then switch on and off the LED, ensuring the blinking effect. The source code for the initialization of the interrupt and the interrupt routine are provided with this document. Please read and understand the code, compile it and test it on the robot.

3.2.1 Sequential blinking LEDs

Make a loop switching ON and OFF the 8 LEDs in a sequential way (LED0 ON and others off, then LED1 ON and others OFF and so on) using the timer interrupt routine.

3.2.2 Blinking of two LEDs

Please use a second timer, for instance the number 4, to blink LED1 at a frequence of 3 Hz.

4 Tutorial C: Motor control by interrupt

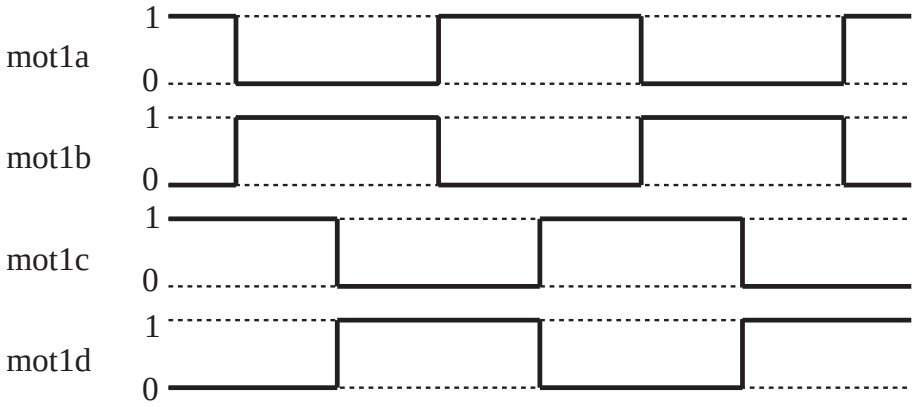

Figure 15: Activation sequence for the motor 1 control signals.

The e-puck robot is equipped with two stepper motors. These motors have two windings. The power in these windings is controlled by the signals `motXa`, `motXb`, `motXc`, and `motXd` (with X being 1 or 2). For motor 1, for instance, the signals `mot1a` and `mot1b` control the power supply of the first winding, `mot1c` and `mot1d` the power supply of the second winding. Figure 14 shows how these four signals have to be activated to make the motor turn. Every signal transition is considered as a step. A rotation of 360^o of the wheel corresponds to 1000 steps (20 steps per revolution of the motor combined with a gear reduction of 50:1.

Try to implement the control of the two motors by interrupt of the timers 4 and 5. Please give to the two motors a fixed speed to make the robot move at 10 mm/s.

Compare your code with the code implemented in `motors.c` provided with the `libraries` of the e-puck robot.

Start looking at the other code you can find in the `libraries` of the e-puck robot.