e-puck Lab Induction 2017/18

1 Preparation Prior to the Lab

You will be working in groups of up to four students (the allocation is available on MOLE). Before coming to the lab, each group member should have completed the following steps. Depending on your system, this may take various amount of time. In the event that you encounter a problem that you cannot solve, the demonstrators can help you with it during the lab.

1.1 Downloading the Software

The e-puck's microcontroller is a dsPIC from Microchip. Microchip provides the MPLAB X © Integrated Development Environment for programming PICs. It also provides a plug-in C compiler for MPLAB X, which allows us to program the microcontroller in C rather than in assembly language.

You need to install the *MPLAB XC16* compiler and *MPLAB X IDE* (freely available for Windows, MacOS and Linux from https://www.microchip.com/development-tools/downloads-archive):

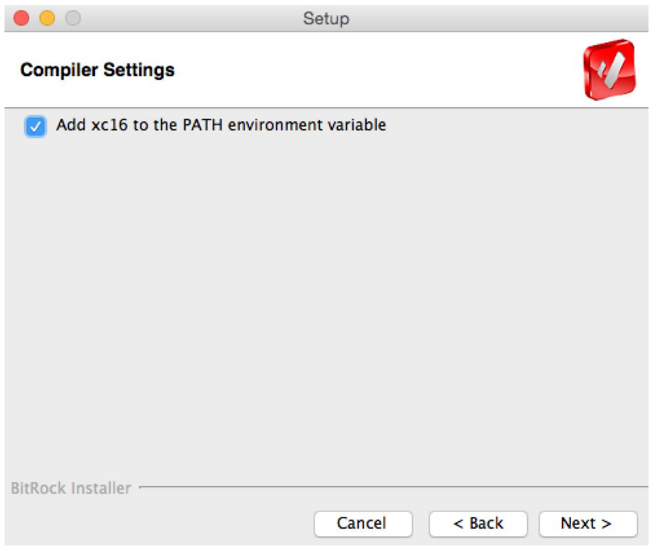
- Download **MPLAB XC16 v1.30** compiler (**later versions may be incompatible with the e-puck**)
- Download **MPLAB X IDE v4.10**

Before you can install MPLAB or the compiler, please make sure that you have the current Java Runtime Environment (JRE).

1.2 Installing the MPLAB XC16 Compiler and MPLAB X IDE

First, you need to run the MPLAB XC16 compiler installer with administrator rights.

- While installing the compiler, choose *Free* as the **license type**.
- Make sure that you check the box "**Add xc16 to the PATH environment variable**":



Setup

Compiler Settings

✓ Add xc16 to the PATH environment variable

Next, you need to run the MPLAB X IDE installer with administrator rights.

- If the setup wizard asks about ***Proxy Settings***, choose "Use System Proxy Settings".
- The setup wizard may ask which **programs to install**. Check both **MPLAB X IDE and MPLAB IPE**.

- Once the setup wizard has finished installing MPLAB, it will ask you whether you wish to be directed to some websites for installing additional components. The first one is already installed, and the other two are not needed. You can thus uncheck all 3 options.

1.3 Opening and Building an MPLAB X Project

1. Download the **e-puck_Lab.zip** folder from MOLE and extract it. This folder includes a version of the e-puck library, and a demo application as an MPLAB X project.
2. Open MPLAB X IDE.
3. Click on the **File** menu and select **Open Project...**.
4. Browse to the folder where you extracted the archive and open **Example.X**.
5. Set **Example** as the main project, by right clicking on the project folder and then clicking on **Set as Main Project.**
6. Select **Clean and Build Main Project** from **Production (or Run)** menu.
7. Check the *Output* window. It should read "**BUILD SUCCESSFUL** (total time: ..)"

1.4 Familiarising yourself with the e-puck

Take a look at Fig. 1 and familiarise yourself with the different parts of the e-puck. In addition, take a look at the example project and try to understand the code in **main.c**.

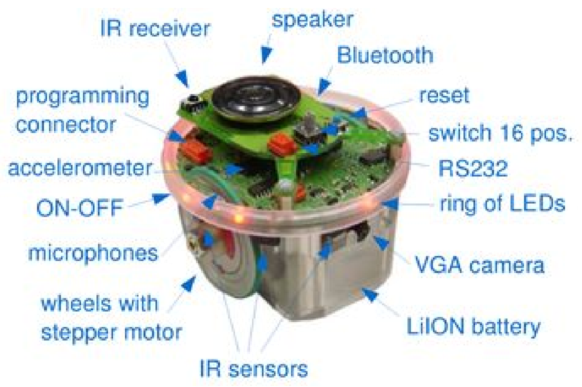


Figure 1. An e-puck and its components. Image reprinted from http://en.wikibooks.org/wiki/Cyberbotics'_Robot_Curriculum/E-puck_and_Webots

Websites that you may find useful for programming the e-puck:

- EPFL (the developer of the robot) maintain an e-puck website: http://www.e-puck.org/
- In this lab, we use the e-puck library, an embedded system library for e-puck. It is already included in e-puck_Lab.zip. For more information: https://github.com/gctronic/e-puck-library/
- GCtronic (the manufacturer of the robot) provides a mini-doc and maintain a very resourceful Wiki: http://www.gctronic.com/e-puck.php

Lab Preparation Checklist

Tick the tasks that you have completed before the lab and **make a note of any problems** you have encountered.

- ❏ MPLAB XC16 compiler is successfully installed.
- ❏ MPLAB X IDE is successfully installed..
- ❏ **e-puck_Lab.zip** is downloaded from MOLE and extracted.
- ❏ The project **Example.X** is successfully built on MPLAB X IDE.

- ❏ Familiarise yourself with the e-puck and the library/code.
- ❏ Check if your computer has Bluetooth (if you are using Windows or MacOS you can learn how to check it here http://www.wikihow.com/Check-if-Your-Computer-Has-Bluetooth). If your laptop does not have Bluetooth, we will provide you with a Bluetooth dongle during the lab.

2 First Lab Session: e-puck Tutorial

If you have completed the lab preparation you can go ahead. You will be mostly using MPLAB X IDE, the integrated development environment. This software is used to link the library and your C code, then convert it to machine code in HEX code format. In particular, when one builds an MPLAB project, one gets a hexadecimal file (**.hex**), which contains the machine code to be executed. The file is located in directory **Example***/dist/default/production/*.

Once you successfully built the HEX code, it is ready to be uploaded into an e-puck. You need to use **MPLAB IPE**, an integrated programming environment to upload the code using the PICkit3™ In-Circuit Debugger (hereafter, **PICkit3 programmer**). This is explained in Section 2.1.

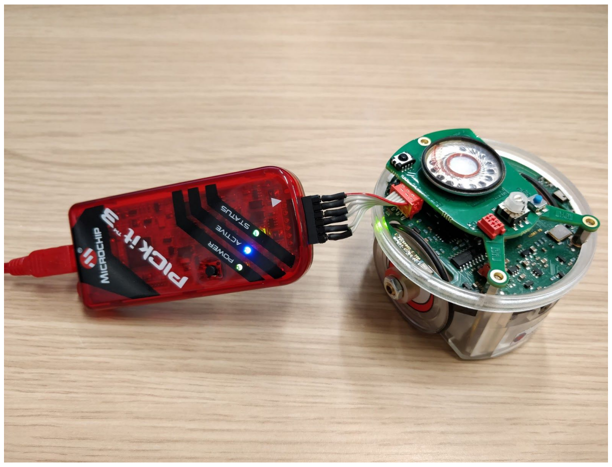
To debug your program (e.g. get sensor readings), you need to communicate with e-puck via Bluetooth. This is explained in Section 2.2.

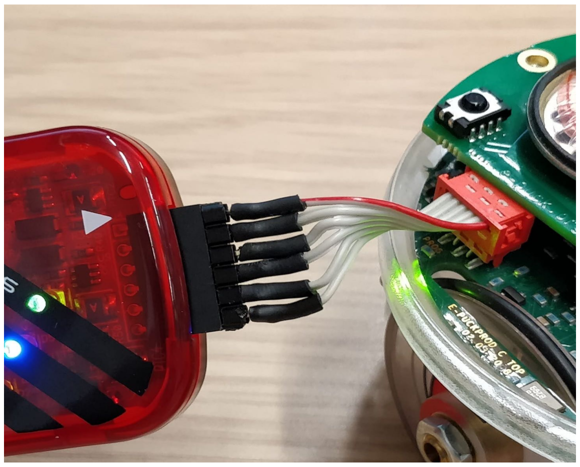
2.1 Using PICkit3 Programmer

Each group will be handed a PICkit 3 programmer and an e-puck at the beginning of each session. For detailed information check
http://www.microchip.com/Developmenttools/ProductDetails.aspx?PartNO=PG164130

1. Open MPLAB IPE (this software is already installed together with MPLAB X IDE).
2. Carefully plug the PICkit3 programmer to the computer and to the e-puck. **The orientation of the cable is very important**:

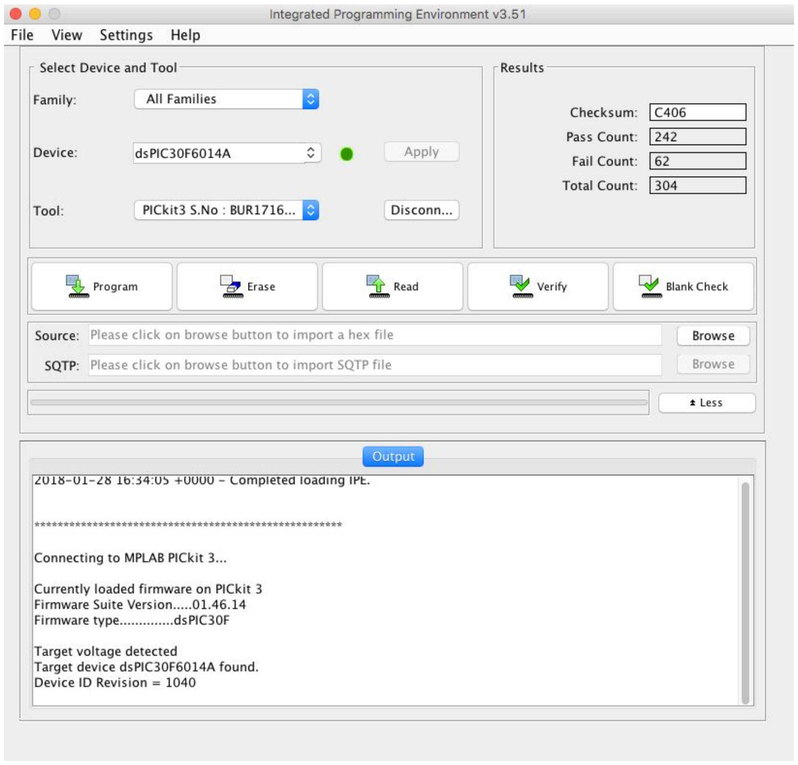
CAUTION: Check the orientation of the cable is correct. The red thread should go into the position indicated by the white arrow on the programmer. See below images.





3. Select the **device** model as **dsPIC30F6014A**, then press **Apply**.
4. Select the programmer model under the **Tool** (there should be only one option).

5. Make sure that the e-puck is **switched on** (OK if it already was).
6. Connect to the PICkit3 programmer using **Connect** button.
7. Check if the connection is confirmed. Please see the screenshot below:



Family: All Families
Device: dsPIC30F6014A
Tool: PICkit3 S.No : BUR1716...

Checksum: C406
Pass Count: 242
Fail Count: 62
Total Count: 304

Source: Please click on browse button to import a hex file
SQTP: Please click on browse button to import SQTP file

```
2018-01-28 16:34:05 +0000 - Completed loading IPE.

*****************************************************

Connecting to MPLAB PICkit 3...

Currently loaded firmware on PICkit 3
Firmware Suite Version.....01.46.14
Firmware type.............dsPIC30F

Target voltage detected
Target device dsPIC30F6014A found.
Device ID Revision = 1040
```

8. Select the **Source**, by clicking on the **Browse** button and choosing the generated hex code (in **Example/dist/default/production**).
9. Click the **Program** button to upload your code onto the e-puck (at this stage the software may ask you if you confirm a **Caution** message, press **OK** button).
10. Unplug the PICkit3 programmer carefully (the cable attached to the e-puck should remain).

CAUTION: Once the upload has finished, the e-puck immediately starts executing the program. Make sure that the e-puck is not at the edge of a table, or anywhere where it may fall over if starting to move. If the robot moves in an uncontrolled way, switch it off.

2.2 Bluetooth Communication

You can use Bluetooth for **receiving data** from e-puck and displaying it in your computer. If your computer does not already have Bluetooth capabilities ask the demonstrators for a Bluetooth dongle. Plug it into your computer and install the drivers if necessary.

Communicating with e-puck via Bluetooth is platform specific. You need to configure the e-puck as a Bluetooth device first. To configure the e-puck as a Bluetooth device please follow the steps in Section 2.3 specific to your operating system.

Windows users can use Tiny PIC Bootloader which is explained in Section 2.4.
MacOS/Linux users can use built-in shell commands, which is explained in Section 2.5.

2.3 Configuring the e-puck as a Bluetooth Device

Add the e-puck to your PC as a Bluetooth device:

On Windows 7/8

1. Make sure that the **e-puck is switched on**.
2. Go to **Devices and Printers** in the Control Panel.
3. Click **Add a Device**.
4. Click on the e-puck device when it appears.
5. Select *Enter the device's pairing code*.
6. Enter the **4-digit pairing code written on the e-puck's sticker**.
7. When the e-puck has been connected, right click on its icon, go to **Properties** and make a note of **which COM port it is connected to**.

On Windows 10

1. Make sure that the **e-puck is switched on**.
2. Press the Windows key and type in "*Bluetooth*" and click on **Bluetooth Devices**.
3. Click on the e-puck device when it appears.
4. Select *Enter the device's pairing code*.
5. Enter the **4-digit pairing code written on the e-puck's sticker**.
6. When the e-puck has been connected, press the Windows key again and type in *"Devices and Printers"* and right-click on your e-puck under unspecified devices. Now, go to **Properties** and make a note of **which COM port it is connected to** in the **COM Ports** tab.

On MacOS

1. Make sure that the **e-puck is switched on**.
2. Open **System Preferences** then select **Bluetooth**.
3. Make sure that Bluetooth is on.
4. Click on the e-puck device when it appears.
5. Select **Options**.
6. Enter the **4-digit pairing code written on the e-puck's sticker**.
7. Check for Bluetooth device files:
   [Terminal] `ls -l /dev/ | grep puck`
8. Give the user permission to use the device connected to the COM1 port:
   [Terminal] `sudo chown <Your User Name> /dev/tty.e-puck_####-COM1`

On Linux

1. Install the **bluez** and **blueman packages**:
   [Terminal] `sudo apt-get install bluez blueman`
   NB: The `apt-get` command is used for Debian and Debian distributions like Ubuntu flavours (e.g. Ubuntu, Xubuntu, Kubuntu, etc.). Other Linux distributions may use another package manager, for example Arch linux uses `pacman`.
2. Make sure that the **e-puck is switched on**.
3. Find the e-puck's Bluetooth address:
   [Terminal] `hcitool scan`
4. You will see something like:
   `10:00:E8:AD:79:EA e-puck_####` (where `####` is your e-puck's 4-digit code)
5. Make a note of the Bluetooth address.

6. Release any other Bluetooth connection:
   [Terminal] `sudo rfcomm release 0`
   NB: If the port has been already released, you will get "Can't release device: No such device". This is not a problem.
7. Bind your **COM1** port to the e-puck:
   [Terminal] `sudo rfcomm bind 0 <e-puck's Bluetooth Address>`
8. Give the user permission to use the device connected to the **rfcomm0** port:
   [Terminal] `sudo chown <Your User Name> /dev/rfcomm0`

NOTE: To prevent the draining of the battery of the e-puck please switch the robot off. Remember that every time you switch the robot on you may have to reestablish the Bluetooth connection (which may not require to input the pairing code again).

2.4 Windows users: Tiny PIC Bootloader

To find out which COM port the e-puck is connected to; open **Bluetooth settings**, click **More Bluetooth Options**. Under **COM Ports** tab**,** you should see the name of your e-puck: **e-puck_#### 'COM1',** the **COM Port** corresponding to the **Outgoing** direction is the one you will select in Tiny PIC Bootloader.

1. Download **Tiny PIC Bootloader** (TinyBld_1_10_6_pc_beta.zip) from MOLE and extract it.
2. Open the Tiny PIC Bootloader (click on **tinybldWin.exe**).
3. Under **Search** button (left side), select the **COM port** that your e-puck is connected to.
4. Click **Terminal** section, and click **Open** button.
5. The terminal should display the data sent by the e-puck.

2.5 MacOS/Linux users: Built-in Shell Commands

It is easier and more reliable to use built-in shell commands for MacOS and Linux users. Make sure that the e-puck is paired with your computer.

1. Open Terminal.
   a. On MacOS, type `cat /dev/tty.e-puck_####-COM1` (where `####` is your e-puck's 4-digit code), then hit Enter.
   b. On Linux, type `cat /dev/rfcomm0`, then hit Enter.
2. The computer may ask you to enter the 4-digit code of the e-puck. Enter the code and press Enter.
3. The terminal should display the data sent by the e-puck.

3 Example programs

Now the purpose is for you to familiarise yourself with coding the e-puck. The example code provided in **Example.X** turns on the front LED. The code snippet below contains only the front LED function. In the `main.c` file you can see the header files, commented functions and other function declarations.

```
// Initialisation.
e_init_port(); // Initializes the ports.
e_init_ad_scan(ALL_ADC); // Initializes the analogue to digital converters.
e_led_clear(); // Turn off all the LEDs in the initialisation stage.
// Main Loop.
while(1) {
   e_set_front_led(1); // Turn the front LED on.
}
```

Following tasks are not compulsory, although, can help you to understand how to program the e-puck.

Optional Task 1:Toggle the green body LED

Modify the example program given above to toggle the body LEDs.

You can look at the cheat sheet for the related functions. Use `Wait` function to give a time delay. `Wait` function accepts time duration in microseconds as input. 1 second is 1000000 microseconds.

Optional Task 2: Control the movement with the selector

The e-puck has a selector on the top of it. The selector provides values from 0 to 15, some of the new e-puck models display these values in hexadecimal number system (0, ..., 9, A, ...,E). You can use `GetSelector()` to read the current selector value.

The e-puck has stepper motors that can move up to **± 12.7 cm/s**. You can use `e_set_speed_left(int motor_speed)` function to set left wheel speed (`e_set_speed_right(int)` for the right motor). The input `motor_speed` is an integer and bounded by **-1000** and **1000**. Thus, you should not set **12.7** to set the maximum wheel speed but **1000** instead.

Program the robot to turn on the spot, by giving both wheels the same speed but different directions, as opposite signs (e.g. 5, -5 cm/s). The body LED should blink as many times as the set selector value, after this, the robot should rotate in the opposite direction.

For instance, if the selector is set to 4, every time the body LED blinks the 4th time, the robot turns direction from clockwise to counterclockwise (or vice versa). If the selector is 0, it should never change direction.

Optional Task 3: Indicate the presence of close objects

Program the robot to indicate that an object is in close proximity. The e-puck robot has **8** IR receivers that can be used as proximity sensors. The proximity sensor readings depend on the ambient light. Thus, it is a good practice to use `e_calibrate_ir()` function before you get sensor readings. The **range** of the proximity sensors can vary in different conditions. If there is an object close to a proximity sensor, the reading is a high number. In general, it is between **0** and **400**. However, if there is an object very close to the proximity sensor, this reading might be up to 4000.

You can send the proximity sensor readings to your computer using Bluetooth communication (UART). Although, you should first implement this. You can use `e_send_uart1_char(char *, int)` function to send buffer array. Do **not** forget to initialise the UART1 module first `e_init_uart1()`, once, before the main loop of your program. Please check the cheat sheet for more information. To display the data sent by the e-puck please check Section 2.4 or 2.5.

Whenever there is an object close to a sensor (you can define your own threshold), turn on the LED that is near that sensor. This would identify the direction of the nearby object in proxy. The range of the proximity sensors are approximately 8 cm.

For instance, assume that there is an object to the left of the robot and you set your threshold to 5 cm. If the object is close e.g. 2 cm, LED6 is switched on. However, if the object is e.g. 7 cm away, LED6 remains off.

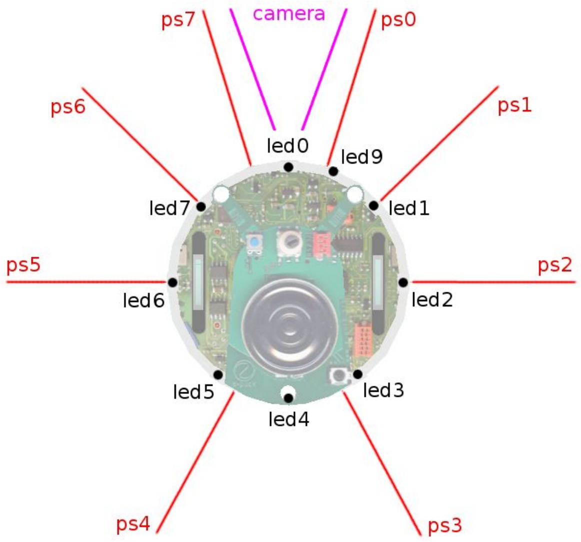


Figure 2. The e-puck components: camera, LEDs, proximity sensors (image reprinted from https://www.cyberbotics.com/doc/guide/using-the-e-puck-robot).

The above examples consider simple tasks for you to familiarise with the robot and programming environment. You can use the robot's accelerometer, UART communication, camera, etc. Please note that the proximity sensors needs to be calibrated or tuned each session you use a robot as the proximity readings are based on the ambient light. For more detailed explanations of the library, please visit: http://www.e-puck.org/

You will find your assessment task in the **e-puck Lab Assessment Briefing** document.
Good luck!