Robotics Cheat Sheet

Dr. Julien Nembrini, Manuel Mondal

Contents

1 Installation

All installations procedures provided here are targeted at Ubuntu 18.04.

1.1 Webots (required)

```
wget -q https://www.cyberbotics.com/Cyberbotics.asc -O- | sudo apt-key add -
sudo apt-add-repository 'deb http://www.cyberbotics.com/debian/ binary-amd64/'
sudo apt-get update
sudo apt-get install webots
```

And run the command `export WEBOTS_HOME=<installation_path>`, where your installation path can be found out using the `whereis webots` command.
Example:

```
run: whereis webots
returns: /usr/local/bin/webots /usr/local/webots

run:  export WEBOTS_HOME=/usr/local/webots
```

Note: to avoid having to retype the export command on every system login, add the command to your .bashrc file using the commands:

```
cp ~/.bashrc ~/.bashrc.BACKUP
echo >> ~/.bashrc
echo "export␣WEBOTS_HOME=/usr/local/webots" >> ~/.bashrc
```

1.2 LaTeX and TexMaker (recommended)

```
sudo apt-get install texlive-full
sudo apt-get install texmaker
```

LaTeX templates for both Report 1 and Report 2 are available on moodle.

2 Compile and run your robot controller

The procedure below shows how to compile code in the simulation environment (through Webots) and on the real e-pucks (command line).

When creating a new robot controller:

- Add a new directory in the `controllers` directory, making sure that the new directory and the .c file share the same name.
- Copy the Makefile into the new directory.

Note: when switching between simulation and real world, make sure to either run the command `make clean`, or to press 'Remove intermediate build files' button in Webots.

2.1 Requirements

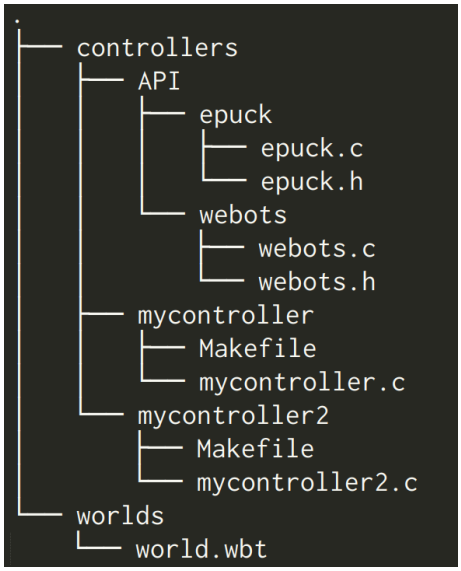
Clone the epuckAPI repository to your disk:

```
git clone https://github.com/nembrinj/epuckAPI.git
```

or get the zip file `https://github.com/nembrinj/epuckAPI/master.zip`. The API might be extended during the semester. If so, you will be notified and will need to `git pull` to update it locally (or re-download the zip).

2.1.1 Folder structure

The code must be placed according to this specific folder structure:



A `world` directory contains the Webots world you are using for the simulation. A `controllers` directory must contain:

- The provided APIs.
- Provided controllers, or any controller you might wish to add. Note that (1) a controller directory and its C file must share the same name and (2) the directory must contain a copy of the provided Makefile.

2.2 Compile and run in simulation

1. Launch Webots.
2. Open a world file (*.wbt), by going to Webots > File > Open World.
3. Expand the node E-Puck (nodes can be found in box to the left of the simulation). And click on the "controller ..." entry. The name of the controller should appear in the box below the node box, accompanied by two buttons "Select . . . " and "Edit". Here you can choose any controller from the directory `controllers` (and from the default controllers), and you can edit their source code.
4. Click "Edit", which opens the source code of the controller.
5. Click on the gear-shaped 'Build the current project' button to compile your code.
6. Click the play-shaped 'Run the simulation in real-time.' button to run it.

2.3 Compile and run on the E-puck2 (wifi remote)

1. Open a terminal in the `mycontroller` directory (or any other controller directory).
2. Run:

```
make API=epuck
```

3. If you have no error, run:

```
./mycontroller 192.168.2.XYZ
```

where XYZ stands for the 3 last digits of the robot number (e.g. 202 for robot n. 4202)

2.4 E-puck code (advanced mode)

Feel free at any point to extend the e-puck API, or event to write code directly for the e-pucks. To start, have a look at the current code in the `API/epuck` directory and consult the e-puck2 GCtronic wiki (see section 6).

Note that debugging robot code might be more difficult or frustrating than using the provided API.

3 Lab information

3.1 Network connection

In order to control the e-pucks via WiFi you need to be connected to the same network as the epucks.

SSID: robotics_lab

Key: 73943477

Very important: to connect your laptop to the network, your MAC address needs to be whitelisted. Before your first visit of the lab, send the MAC address of your wifi card to manuel.mondal@unifr.ch and wait for the confirmation reply.

3.2 E-puck IP

Each E-puck has a sticker with and ID of the form 4XYZ. The e-puck IP address is 192.168.2.XYZ

3.3 Computer login

Each device has an identifier A (whiteboard side), B (center) or C (book case side).

Username: robotics_a / robotics_b / robotics_c

Password: robotics

3.4 Lab availability

Workdays The lab's door is locked at night. If you are the first to arrive, or the last to leave in the evening, ask one of the researchers in an office of the same corridor to lock or unlock the lab.

Weekends Uni and therefore the lab is always closed on Sundays. On Saturdays, the lab can sometimes be opened on appointment. To request access, write an e-mail to manuel.mondal@unifr.ch, **at least 1 day in advance**, also specifying when you want to use the lab. You will be notified if the lab can be unlocked at that time.

Coordinate e-puck use Since there is a limited number of robots and limited space in the lab, you can use the following Google sheet to indicate your group number and reserve your space and timeslots: goo.gl/YYMFRm.

One reservation corresponds to one (of the three desktops) and up to three (of the 9) e-pucks.

The rule is simple: you can go to the lab without reservation, however groups who have reserved the material will have priority to use it. Be mindful and fair to each other.

4 Bug reporting / defective material

When identifying bugs in the API, issues with compiling, etc. or when noticing defects in the lab (e-puck defective, loading station not working, lab desktop not starting, etc.), follow these steps:

1. Describe the bug precisely (specifying e-puck ID, affected desktop, etc.).
2. Describe how to reproduce the bug (e.g. "When enabling the camera, the ground sensors systematically become unresponsive on e-puck 4200. Use the attached code to reproduce the situation").
3. Post it on the Moodle forum.

5 Series hand-in guidelines

The following guidelines must be strictly respected for your homework assignement to be accepted:
Zip and PDF naming convention: `gg_nn_ff.zip`, where `gg` stands for your group number (00 if none), `nn` for your last name and `ff` for your first name.

- Hand-in a single zip file using the naming convention specified above.
  Example: 99_mustermann_max.zip
- The zip file must include exactly:
  - One directory `controllers` containing **only** the controllers relevant for the assignment
  - One directory `worlds` containing **only** the worlds relevant for the assignment
  - One PDF for your report **(named using the same naming convention)**

The zip can contain one additional directory `misc` if you have other files you want to send us.
A python 3 script is available to help you check if you respected the guidelines (see on Moodle). Put the script on the same folder as your zip file and run it with:

```
python student_script.py 99_mustermann_max.zip
```

The output of the script will indicate if the zip passes the requirements.

If your homework submission does not pass the script, it will be refused.

6 References and guides

Webots e-puck API: https://cyberbotics.com/doc/guide/epuck#e-puck2

Getting started with Webots: https://cyberbotics.com/doc/guide/getting-started-with-webots

E-puck2 GCtronic full wiki: http://www.gctronic.com/doc/index.php?title=e-puck2_PC_side_development#WiFi_2

E-puck2 Firmware: https://github.com/e-puck2

Webots source code: https://github.com/omichel/webots