Tutorial setting up linux-port

Tutorial for setting linux-port on the e-puck

IAS Laboratory-Bristol & Supméca-Toulon
Jean-Charles Antonioli

The theory is when it does not work, but we know why. Practice is when it works, but we do not know why.

Tutorial setting up linux-port

Contents

1 Introduction 2
1.1 Presentation 2
1.2 Where can I buy it? 2
1.3 How can I connect my e-puck to my computer? 2
2 What must be done first with windows 4
2.1 The tool you need to download&install: SAM-BA 4
2.2 Initialize connection with SAM-BA 5
2.3 Sending AT91BOOTSTRAP and U-Boot 6
3 Then, use linux to flash your linux-port 7
3.1 Install ckermit 7
3.2 Install Linux on your linux-port 7
3.3 Set u-boot environment 8
4 Install Linux on SD-card 9
4.1 Save an already installed SD-card 9
4.2 Prepare file-system 10
4.3 Maintenance/restore linux on SD-card 11
5 Using your Linux-port (with linux) 12
5.1 Connect to the wi-fi 12
5.2 Connect your computer to your e-puck linux-board 12
5.3 Change IP of your e-puck 12
5.4 Sending your first program to your e-puck 12
5.4.1 Easy-steps 12
5.4.2 Commentaries about this program 13
6 Conclusion and thanks 14

1 Introduction

In this tutorial, using linux, programming in C, understanding the meaning of binary and how actuators and sensors work will not be explained. You should learn how-to use the e-puck too (I have written another how-to for that before). You should learn these first, before following these steps to use the linux-port on the e-puck.

You will need both Windows (XP) and Linux (Ubuntu 8) to use this tutorial.

This tutorial is a step-by-step document: you only need to read it entirely and follow the steps to use the linux-port.

1.1 Presentation

The linux-port will permit you to use the e-puck to make stronger work than with the original e-puck. Thanks to it, you can save variables in live, communicate easier with others robots, all of that using linux!

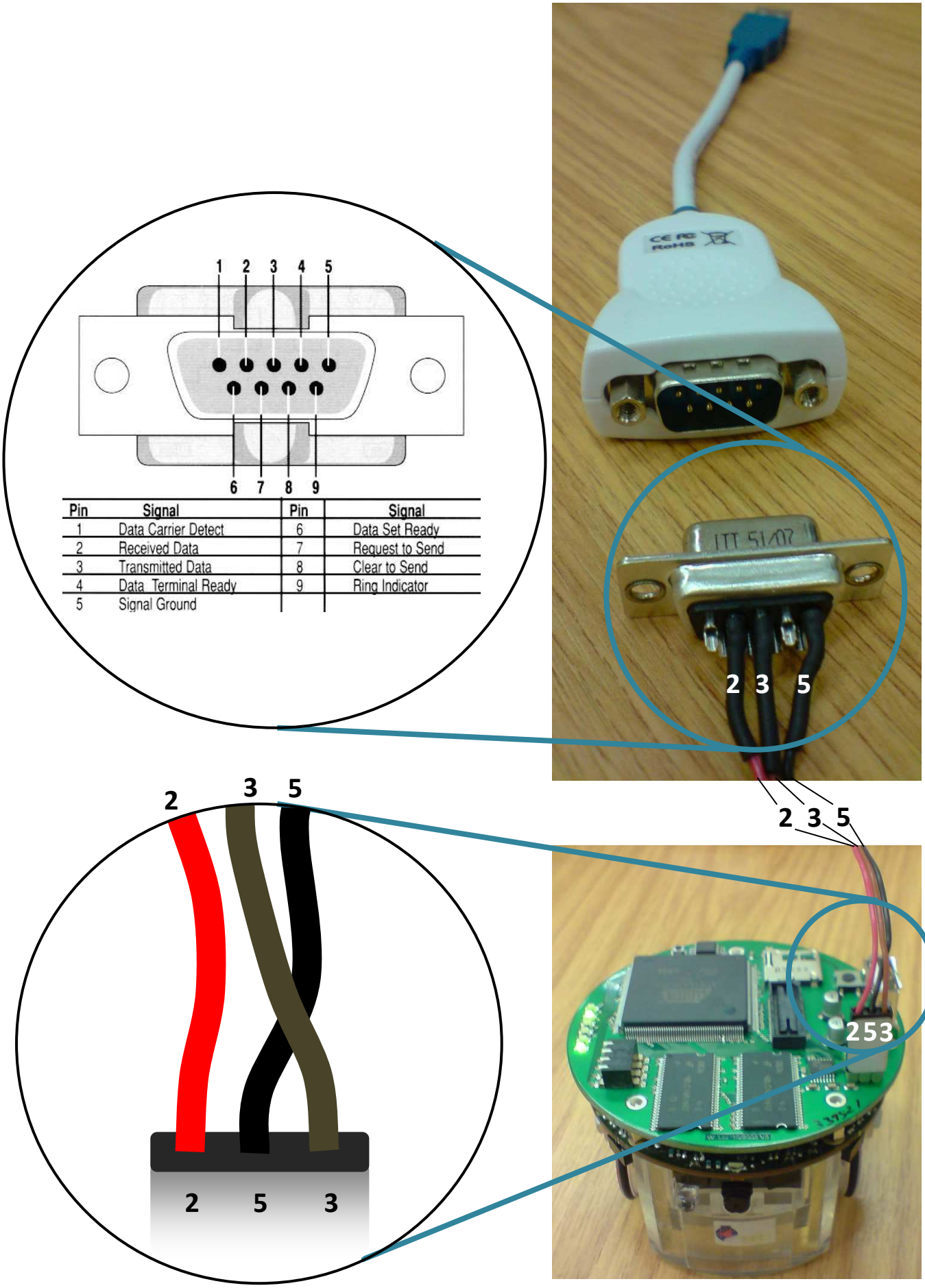
This port is based on an aTmel AT81SAM9260 chip.

1.2 Where can I buy it?

1.3 How can I connect my e-puck to my computer?

You will need to crate yourself a cable to connect the linux-port to your computer.

So you may buy an USB-RS232 (UC232R has been used for this tutorial) adapter and create a three-links wire (2, 3 and 5) between this adapter and your e-puck, as shown on the pictures on the next page.



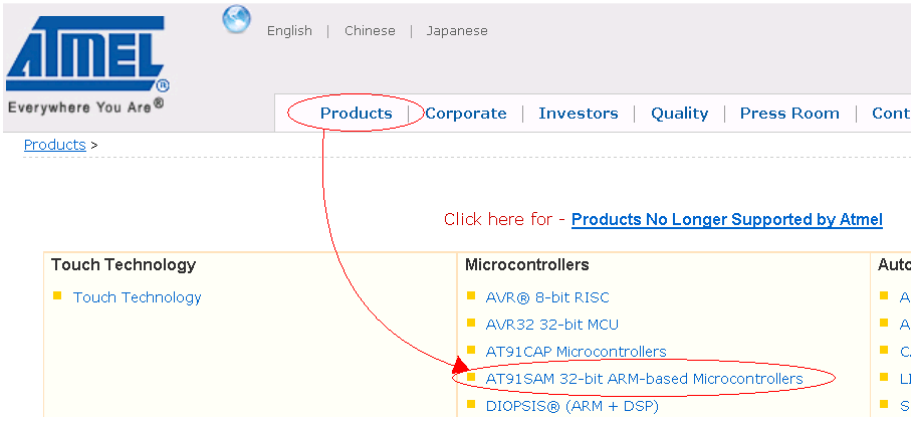
Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

2 What must be done first with windows

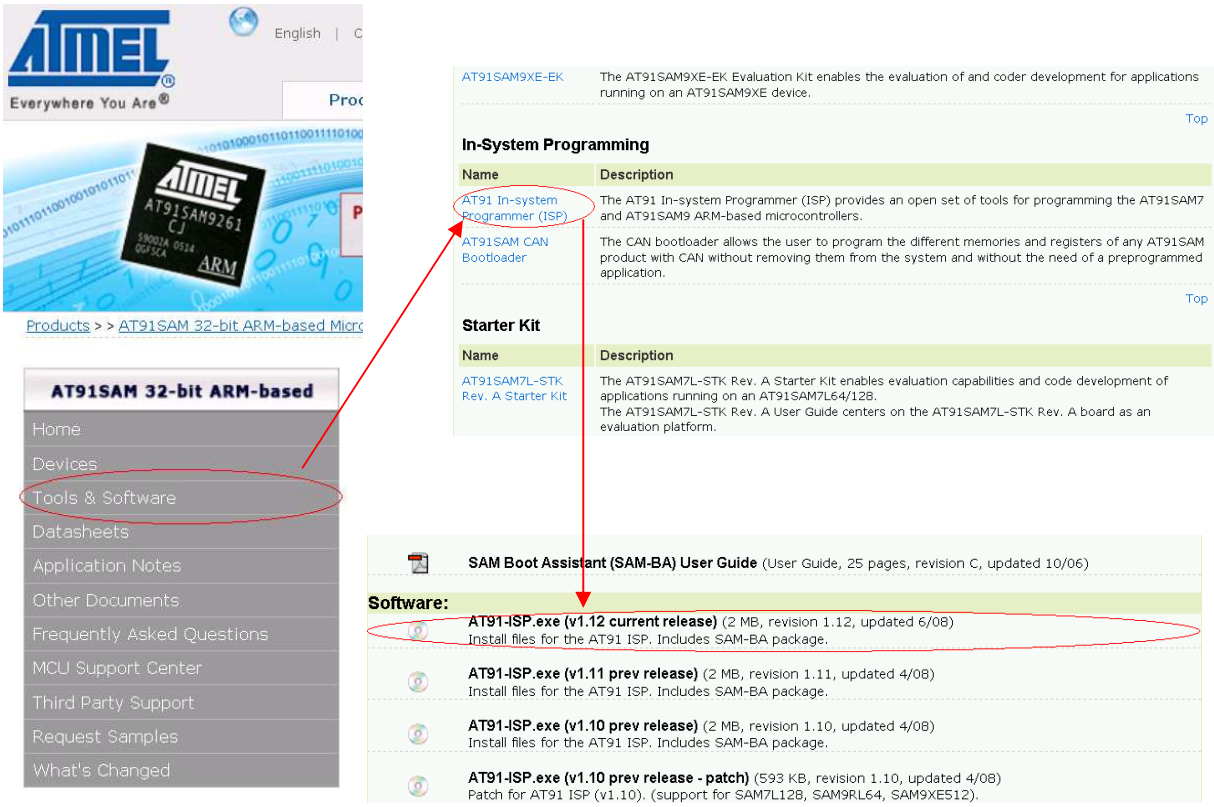
2.1 The tool you need to download&install: SAM-BA

Go to http://www.atmel.com/

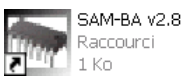
Click on « Products », then click on the « 32-bit ARM-based Microcontrollers » link:



Click on « Tools & Software », then click on « AT91 In-System Programmer » and download and install the latest Software « AT91-ISP.exe »:



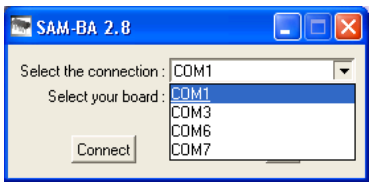
Once this is done, you should have at least this program on your desktop:



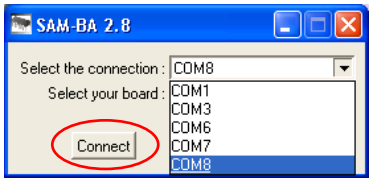
2.2 Initialize connection with SAM-BA

Once this is done, you have all the file you will need therefore to flash your linux-port. Plug-in your linux-port to your e-puck and connect it to your computer via the cable. If a driver is asked to be installed, and is not recognized by windows, you can download it from internet and install it by selecting « choose a device from a directory » and select the folder in which you have unzipped the files. Otherwise, install the driver.

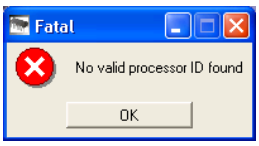
Disconnect your USB cable from your computer and start SAM-BA. Choose the right chip on « Select your board ». Then have a look on the list of ports available in the list « Select the connection ». You should see something like:
Remember the list and close SAM-BA.

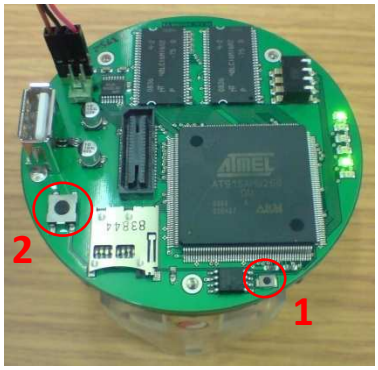


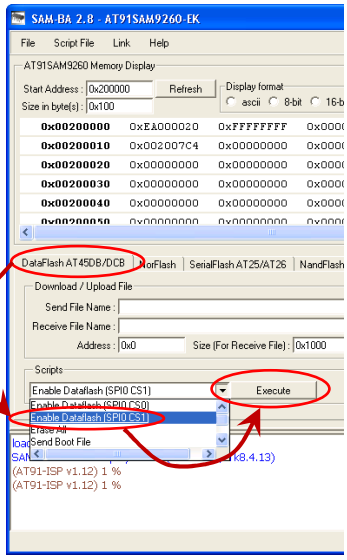
Now connect your USB cable, and start SAM-BA again. A new port COM should have appeared in the list. Select it and click on « Connect ».



If it does not work (a window appears few seconds after with « No valid processor ID found »), it is certainly because your chip is already flashed. If you still want to flash it, you can do it. Press the button 1, then, while still pressing the button 1, press the button 2. Now you can unpress the two buttons and you chip is ready to receive data to be flashed.







Once this is done, you must initialize direct connection to your chip. To do that, select « DataFlash AT56DB/DCB » tab, and select the script « Enable Dataflash (SPI0 CS1) ». Then click on « Execute ». If it does not work, try to enable another script to enable Dataflash like « SPI0 CS# » (where # is another number).

You should see something like «-I- Applet initialization done » in the terminal. Keep this window opened.

2.3 Sending AT91BOOTSTRAP and U-Boot

The on-board tools needed to prepare your e-puck to receive its linux Operating System must be flashed first. To do that, go to http://www.linux4sam.org/ to download AT91BOOTSTRAP and U-Boot (you can find further informations on this same website):

- AT91BOOTSTRAP: click on AT91BOOTSTRAP on the left side, and download the « DataFlash » bin for at91sam9260ek (the chip used on the e-puck port):

Pre-built binaries

DataFlash

BOARD	DESCRIPTION	BINARY
at91sam9260ek	AT91Bootstrap binary with 2nd level application in **Dataflash chip** (SPI 0 CS 1)	dataflash_at91sam9260ek.bin
at91sam9xeek	AT91Bootstrap binary with 2nd level application in **Dataflash chip** (SPI 0 CS 1)	dataflash_at91sam9xeek.bin

- U-Boot: click on U-Boot on the left side, and download the « U-Boot » bin by Dataflash for at91sam9260ek:

U-boot pre-built images

A revision of u-boot for AT91 boards is built from the official u-boot website and a patch against this revision of u-boot.
This patch is regularly updated.

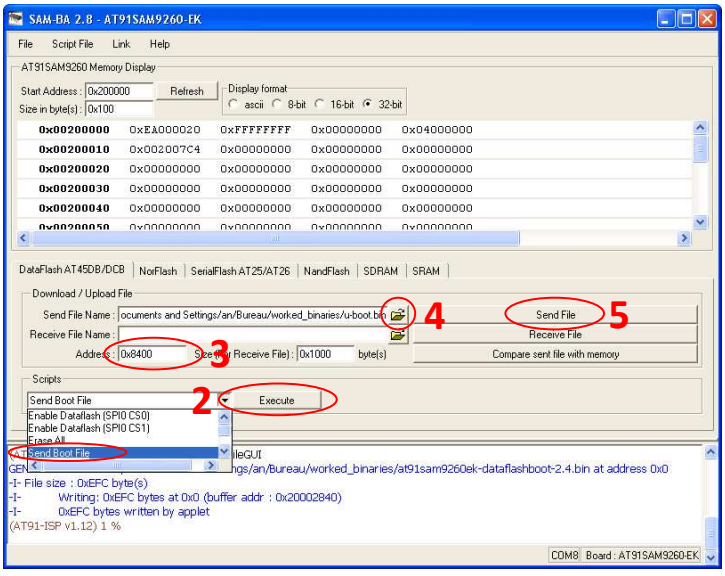
U-boot environment

The u-boot environment is a little read/write persistent space that stores variables needed by the bootloader to configure itself properly and to adapt to its environment (network configuration, boot arguments, storage location, etc.). It is located in the same media that it has booted from. Check Boot capabilities matrix to know where the u-boot environment is located for the pre-built binary.

U-boot 1.3.4 experimental binaries

DataFlash

BOARD	DESCRIPTION	BINARY
at91sam9260ek	u-boot binary with environment in **Dataflash chip** (SPI 0 CS 1)	u-boot-1.3.4-exp-at91sam9260ek-dataflash_cs1.bin
at91sam9xeek	u-boot binary with environment in **Dataflash chip** (SPI 0 CS 1)	u-boot-1.3.4-exp-at91sam9xeek-dataflash_cs1.bin



Now, come back to SAM-BA. Select the script « Send Boot File » (1). Click on Execute (2) and in the window that is opened, select your « dataflash_at91sam9260ek.bin ».

Once this is done, you should see something like « -I- 0xEFC bytes written by applet » in the terminal.

Then, change the value of the Adress (3) to « 0x8400 », choose your « u-boot.bin » file as « Send File Name » (4) and click on « Send File » (5).

Once this is done, you should see something like « » in the terminal.

3 Then, use linux to flash your linux-port

3.1 Install ckermit

- In a terminal, execute:

> sudo apt-get install ckermit

- Then, create a configuration file, directly in your home:

> nano .kermrc

- Inside this file, copy these lines:

```
set line /dev/ttyUSB0
set speed 115200
set carrier-watch off
set handshake none
set flow-control none
robust
set file type bin
set file name lit
set rec pack 1000
set send pack 1000
set window 5
```

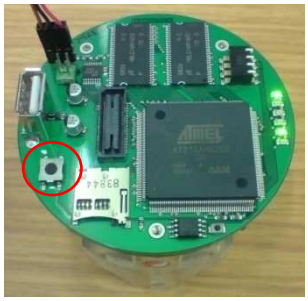
- Quit (ctrl+x) and save (O). You may need to change /dev/ttyUSB0 to propriate name. Find the right name using:

> dmesg | grep "tty*"

3.2 Install Linux on your linux-port

- Connect the target board with host PC by using the terminal:

> kermit -c



- Reset the target board (with the button circled on this picture on the right). Once it has finished to work, put in your terminal:

> loadb

- Press « ctrl + \ » and then press « c » to disconnect, and send the kernel file with (this file should have been enclosed at the same time than this tutorial):

> send uImage.gz

- Connect the board again by typing:

```
> connect
```

- You should have something like this in your terminal:

```
## Total Size       = 0x001377C0 = 136508 Bytes
## Start Addr       = 0x22000000
```

- Copy the binary files from mermory to flash:

```
> cp.b 0x22000000 0xD0042000 0x1377C0
```

- Note that the size of the file used at this step is retrieved from previous step (and will be used in the next step for kernelSize).

3.3 Set u-boot environment

- Now the linux kernel is in the flash (AT45DB321D), we need to set up some u-boot environmental variables in order to let u-boot know how to load the linux kernel into the memory. So put in your terminal:

```
> setenv kernelAddr 0x22000000
> setenv kernelOff 0xD0042000
> setenv kernelSize 0x1377C0
> setenv bootcmd 'cp.b $(kernelOff) $(kernelAddr) $(kernelSize); bootm $(kernelAddr)'
> setenv bootargs 'root=/dev/mmcblk0p2 rootdelay=10'
> saveenv
```

- You can check the environmental variables using:

```
> printenv
bootdelay=3
baudrate=115200
kernelAddr=0x222000000
kernelOff=0xC0042000
bootcmd=cp.b $(kernelOff) $(kernelAddr) $(kernelSize) ;bootm $(kernelAddr)
bootargs=root=/dev/mmcblk0p2 rootdelay=10
kernelSize=0x14d21c
stdin=serial
stdout=serial
stderr=serial
```

4 Install Linux on SD-card

Plug-in your SD-card into your SD-card reader and connect it to your e-puck. Open a terminal and install dump:
> sudo apt-get install dump

4.1 Save an already installed SD-card

To know what is the media used by USB:
> df
Sys. de fich. 1K-blocs Occupé Disponible Capacité Monté sur
/dev/sda3 9107960 6399636 2245652 75% /
tmpfs 1030120 0 1030120 0% /lib/init/rw
varrun 1030120 228 1029892 1% /var/run
varlock 1030120 0 1030120 0% /var/lock
udev 1030120 192 1029928 1% /dev
tmpfs 1030120 468 1029652 1% /dev/shm
lrm 1030120 2392 1027728 1% /lib/modules/2.6.28-11-generic/volatile
/dev/sda5 96735364 77833544 18901820 81% /media/DATA
/dev/sdc2 1826940 866924 867212 50% /media/UUUUUUUUUUUUUUU
/dev/sda1 10231392 9127456 1103936 90% /media/XP

> sudo dump -0 -a -L -u -f - /dev/sdc2 >> linuxepuck.bak

Then, you have to unmount your USB reader of SD-card, so you need to know which one to unplug:
> df
Sys. de fich. 1K-blocs Occupé Disponible Capacité Monté sur
/dev/sda3 9107960 6399636 2245652 75% /
tmpfs 1030120 0 1030120 0% /lib/init/rw
varrun 1030120 228 1029892 1% /var/run
varlock 1030120 0 1030120 0% /var/lock
udev 1030120 192 1029928 1% /dev
tmpfs 1030120 468 1029652 1% /dev/shm
lrm 1030120 2392 1027728 1% /lib/modules/2.6.28-11-generic/volatile
/dev/sda5 96735364 77833544 18901820 81% /media/DATA
/dev/sdc2 1826940 866924 867212 50% **/media/UUUUUUUUUUUUUUU**
/dev/sda1 10231392 9127456 1103936 90% /media/XP

> sudo umount /media/UUUUUUUUUUUUUUU/

4.2 Prepare file-system

Get your SD card and a card reader ready. plug into USB. Using…

```
sudo fdisk -l
```

…to find the device, then…

```
sudo fdisk /dev/sdb
```

You have to create one linux swap (64M) and one linux partitions:

```
Command (m for help): n
Command action
      e extended
      p primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-983, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-983, default 983): +64M

Command (m for help): n
Command action
      e extended
      p primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (33-983, default 33):
Using default value 33
Last cylinder or +size or +sizeM or +sizeK (33-983, default 983):
Using default value 983

Command (m for help): t
Partition number (1-4): 1
Hex code (type L to list codes): 82
Changed system type of partition 1 to 82 (Linux swap / Solaris)

Command (m for help): p

Disk /dev/sdb1: 2031 MB, 2031226368 bytes
64 heads, 63 sectors/track, 983 cylinders
Units = cylinders of 4032 * 512 = 2064384 bytes
Disk identifier: 0x00000000

      Device Boot   Start       End    Blocks    Id   System
/dev/sdb1p1             1        32    64480+   82   Linux swap / Solaris
/dev/sdb1p2            33       983  1917216    83   Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 22: Invalid argument.
The kernel still uses the old table.
The new table will be used at the next reboot.
Syncing disks.
```

Finally, close this terminal and open a new one. Do:

```
> fdisk -l
```

You should find /dev/sdc2. If it is, check again with:

```
> dir /dev/sd <TAB><TAB>
```

If it SDC2 appears correctly (or at another name for your own device), then:

```
> sudo mount
/dev/sda3 on / type ext3 (rw,relatime,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
varrun on /var/run type tmpfs (rw,nosuid,mode=0755)
varlock on /var/lock type tmpfs (rw,noexec,nosuid,nodev,mode=1777)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
fusectl on /sys/fs/fuse/connections type fusectl (rw)
lrm on /lib/modules/2.6.28-11-generic/volatile type tmpfs (rw,mode=755)
securityfs on /sys/kernel/security type securityfs (rw)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,noexec,nosuid,nodev)
gvfs-fuse-daemon on /home/jc/.gvfs type fuse.gvfs-fuse-daemon (rw,nosuid,nodev,user=jc)
/dev/sda5 on /media/DATA type fuseblk (rw,nosuid,nodev,allow_other,blksize=4096)
/dev/sda1 on /media/XP type vfat
(rw,nosuid,nodev,uhelper=hal,shortname=mixed,uid=1000,utf8,umask=077,flush)
/dev/sdc1 on /media/disk type vfat
(rw,nosuid,nodev,uhelper=hal,shortname=mixed,uid=1000,utf8,umask=077,flush)

> sudo umount /media/disk/
```

Now, you can unplug and plug again USB-reader.

```
> dir /dev/sd <TAB><TAB>
sda sda1 sda2 sda3 sda5 sda6 sdb sdc sdc1 sdc2 sdd sde
> sudo mkfs.ext3 /dev/sdc2
> sudo mkswap /dev/sdc1
```

4.3 Maintenance/restore linux on SD-card

```
> sudo mkdir temp
> sudo mount /dev/sdc2 temp
> cd temp
> sudo restore -ruf - < /home/jc/linuxepuck.bak
```

5 Using your Linux-port (with linux)

Open two terminals (one will be called « A » and the other one will be called « B »).

5.1 Connect to the wi-fi

Connect the wi-fi of your computer to « swarm-lab » (password is robotswarm).

5.2 Connect your computer to your e-puck linux-board

Follow these steps on the right terminal:

« A » : ~$ ssh root@192.168.0.9 (ip address of your robot – then wait a while)

« A » : root@192.168.0.9's password: robot (this is the password asked and will not be prompted will writing)

« A » : epuck:~# cd epuck_linux_mod/linux_side/playerplugin/

« A » : …in# modprobe po0630 (this will load drivers for the camera on the board)

« A » : …in# player lpuck.cfg (this will start the player server: now you can send your code to the epuck thanks to terminal B threw the wi-fi)

5.3 Change IP of your e-puck

> cd /etc/network
> vim interfaces

You can find the IP configuration of your epuck inside, so change it as you want.

Then, you have to reboot the network of the e-puck, and reconnect your computer to your epuck using the right IP (section 5.2).

> /etc/init.d/networking restart

5.4 Sending your first program to your e-puck

5.4.1 Easy-steps

« B » : create a file (for example0.cc) in which you put this code:

```
#include <iostream>
#include <libplayerc++/playerc++.h>
int main (int argc, char *argv[])
```

```
{
 using namespace PlayerCc ;
 PlayerClient robot("192.168.0.9"); // change the IP into the one of your e-puck
 IrProxy ir(&robot,0);
 //Position2dProxy pp(&robot,0);
 for(;;)
 {
  double turnrate, speed;
  robot.Read(); // read from proxies
  std::cout << ir << std::endl;
//pp.SetSpeed(speed,turnrate); //command the motors
 }
}
```

« B » : .../example0_folder$ g++ -o example0 `pkg-config –cflags playerc++` example0 `pkg-config –libs playerc++`

Be careful on this previous code. You can not use « ' » nor « ‘ » but only « ` ». This code is made to compile your example0.cc into a script example0, and this is this script that you will direcly execute to send your program to your e-puck thanks to player.

« B » : .../example0_folder$ ./example0

5.4.2 Commentaries about this program

PlayerClient, IrProxy and Position2dProxy are all C++ classes created by player. You can find all of them on that page: http://playerstage.sourceforge.net/ in section Docs on the left banner, click on « Manuals » > « HTML » Link for Player > « Client Libraries » in section User > « libplayerc++ » on the drawing > « Proxies » on the drawing.

But not all of them are used on the e-puck. To know which ones you can use, have a look on ~/epuck_linux_mod/linux_side/support.c on your e-puck.

6 Conclusion and thanks

Now, you can use all the fonctions you have to make your robot move and use its skills much more easily.

I would want to thank some people for the help I have received from them:

- Alan Winfield, professor and my master in the UWE
- Gilles Richard and Cedric Anthierens for their help
- Wenguo who has created the linux-port and helped me a lot to write this tutorial (some things are directly taken from parts of what he had written himself).
- Bristol Robotics Laboratory
- Supméca Toulon