

# K-JUNIOR smart EVO

user manual



version 1.0  
november 2013

## Documentation Author

Julien Tharin  
K-Team S.A.  
Z.I. Plans-Praz  
1337 Vallorbe  
Switzerland

Email: [info@k-team.com](mailto:info@k-team.com)

Url: [www.k-team.com](http://www.k-team.com)

## Documentation version

| Version | Date       | Author    | Description |
|---------|------------|-----------|-------------|
| 1.0     | 14.11.2013 | J. Tharin | First draft |
|         |            |           |             |
|         |            |           |             |

## Trademark Acknowledgements:

|                  |   |                                       |
|------------------|---|---------------------------------------|
| <b>IBM PC</b>    | : | International Business Machines Corp. |
| <b>Macintosh</b> | : | Apple Corp.                           |
| <b>LabVIEW</b>   | : | National Instruments Corp.            |
| <b>Matlab</b>    | : | MathWorks Corp.                       |
| <b>Logitech</b>  | : | Logitech Int. SA                      |
| <b>Gumstix</b>   | : | Gumstix Inc.                          |
| <b>Khepera</b>   | : | K-Team SA                             |
| <b>K-Junior</b>  | : | K-Team SA                             |

## LEGAL NOTICE:

- The contents of this manual are subject to change without notice
- All efforts have been made to ensure the accuracy of the content of this manual. However, should any error be detected, please inform K-Team.
- The above notwithstanding, K-Team can assume no responsibility for any error in this manual.

## TABLE OF CONTENTS

|  |           |
|--|-----------|
| <b>1. INTRODUCTION .....</b>                     | <b>1</b>  |
| 1.1 HOW TO USE THIS HANDBOOK .....               | 1         |
| 1.2 SAFETY PRECAUTIONS .....                     | 2         |
| 1.3 RECYCLING .....                              | 2         |
| 1.4 SPECIFICATIONS .....                         | 3         |
| <b>2. UNPACKING AND INSPECTION.....</b>          | <b>4</b>  |
| 2.1 PACKAGE CONTENTS.....                        | 4         |
| 2.2 INSPECTION .....                             | 5         |
| <b>3. DESCRIPTION .....</b>                      | <b>6</b>  |
| 3.1 OVERVIEW .....                               | 6         |
| 3.2 KJ-EVO HARDWARE .....                        | 8         |
| 3.2.1 USER CONNECTOR .....                       | 8         |
| 3.2.2 RESET.....                                 | 8         |
| 3.2.3 USB HOST .....                             | 8         |
| 3.2.4 $\mu$ -SD CONNECTOR.....                   | 9         |
| 3.2.5 LEDS .....                                 | 9         |
| 3.2.6 USB SLAVE .....                            | 9         |
| 3.2.7 MICROPHONES .....                          | 9         |
| 3.2.8 SPEAKERS.....                              | 9         |
| 3.2.9 CAMERA.....                                | 9         |
| 3.2.10 CONNECTORS TO THE ROBOT.....              | 9         |
| 3.2.11 WIFI AND BLUETOOTH .....                  | 10        |
| 3.3 KJ-EVO SOFTWARE.....                         | 11        |
| <b>4. USAGE .....</b>                            | <b>12</b> |
| 4.1 REQUIRED HARDWARE / SOFTWARE .....           | 12        |
| 4.1.1 REQUIRED HARDWARE .....                    | 12        |
| 4.1.2 REQUIRED SOFTWARE .....                    | 12        |
| 4.2 ASSEMBLY .....                               | 13        |
| 4.3 POWER-UP AND CONSOLE ACCESS .....            | 14        |
| 4.4 SHUTDOWN / REBOOT / RESET .....              | 17        |
| 4.5 SOFTWARE .....                               | 18        |
| 4.5.1 INSTALLATION OF LIGHT TOOLCHAIN .....      | 18        |
| 4.5.2 PROGRAMMING AND LIGHT TOOLCHAIN USAGE..... | 22        |
| 4.6 FULL TOOLCHAIN AND SOURCES.....              | 26        |
| 4.6.1 REQUIRED SOFTWARE .....                    | 26        |
| 4.6.2 INSTALLATION.....                          | 27        |
| 4.6.3 FULL TOOLCHAIN USAGE .....                 | 28        |
| 4.6.4 PACKAGES INSTALLATIONS .....               | 31        |

|  |           |
|--|-----------|
| <b>5. ANNEXES .....</b>                        | <b>32</b> |
| 5.1 TOOLS AND COMMANDS .....                   | 32        |
| 5.1.1 USING THE SERIAL PORT AND MINICOM .....  | 32        |
| 5.1.2 USING WIFI .....                         | 36        |
| 5.1.3 TRANSFERRING FILES USING SCP (SSH) ..... | 39        |
| 5.1.4 NETWORKING OVER USB SLAVE .....          | 39        |
| 5.1.5 REMOTE ACCESS.....                       | 40        |
| 5.1.6 USING THE CAMERA MODULE .....            | 40        |
| 5.1.7 NFS CONFIGURATION .....                  | 42        |
| 5.1.8 USING MICROPHONES AND SPEAKERS .....     | 43        |
| 5.1.9 DEVELOPMENT WITH A VIRTUAL MACHINE.....  | 46        |
| 5.1.10 DEBUGGING .....                         | 47        |
| <b>6. WARRANTY .....</b>                       | <b>51</b> |

## 1. INTRODUCTION

---

Thank you for buying the K-Junior Smart Evo (abbreviated KJ-EVO in this document)!

With this card, you will be able to create many new embedded systems by interfacing standard devices or enhance the K-Junior V2 robot by expanding its computing power.

### 1.1 How to use this handbook

This handbook introduces the KJ-EVO and its various operating modes. For a quick start, jump to chapter 4 "*Usage*".

If this handbook does not answer one of the problems you wish to solve, please consult the K-Team web site (<http://www.k-team.com>) and especially the Forum and the FAQs.

- **Unpacking and Inspection** : KJ-EVO package description and first use
- **Description** : KJ-EVO description
- **Usage** : KJ-EVO usage descriptions.
- **Annexes** : Detailed descriptions of several helpful tools and commands are explained.

### 1.2 Safety precautions

Here are some recommendations on how to correctly use the KJ-EVO:

- **Keep the board away from wet area.** Contact with water could cause malfunction and/or breakdown.
- **Store your board in a stable position.** This will avoid the risks of falling, which could break it or cause damage to a person.
- **Do not plug any connectors while the board is powered on.** To avoid any damage, make all connections when the board power is off.
- **Never leave the KJ-EVO powered when it is unused.** When you have finished working with KJ-EVO, turn it off. It will save the battery life.

### 1.3 Recycling

Think about the end of life of your product! Parts of the board can be recycled and it is important to do so. By recycling you can help to create a cleaner and safer environment for generations to come. For those reasons please take care to the recycling of your product at the end of its life cycle, for instance sending back the product to the manufacturer or to your local dealer.

**Thanks for your contribution to a cleaner environment!**

## 1.4 Specifications

The main specifications of the KJ-EVO card are listed below:

- Processor : OMAP 3503 ARM Cortex-A8 CPU @ 600MHz
- Performance : Up to 1200 Dhrystone MIPS
- Memory : 512 MB RAM  
512 MB Flash
- Features :
  - USB host
  - Micro SD Connector
  - Mini USB OTG connector for USB slave
  - I<sup>2</sup>C
  - Serial port
  - Wifi B/G, Bluetooth v2.0 + EDR
  - 3-axis accelerometer
  - Sound: 2 speakers and microphones
  - 2 PWM
  - 2 GPIO
  - 2 ADC
  - Color camera 752x480, 30 fps
- OS : Linux OS, Angström distribution (OpenEmbedded tools), kernel 2.6.34
- Power consumption : 410 mA @ 5 V
- Size : 105 width x 125 depth x 110 height\* [mm]
- Mass : 110 g

\* with antennas, without lower pins

## 2. UNPACKING AND INSPECTION

---

### 2.1 Package Contents



*Figure 2-1: Contents of the KJ-EVO Pack*

Your package should contain the following items:

1. KJ-EVO board
2. DVD with software and this User Manual



## 2.2 Inspection

The KJ-EVO board basic functions should be tested after unpacking. A Complete Linux system is installed in the KJ-EVO flash memory. The system can be started as a standalone Linux box, with the initial console displayed on the serial line or network connection (Bluetooth, Wifi, USB). No application is started except some standard daemons and the initial shell on the console.

The Bluetooth port should be linked to a terminal, such as *minicom*, on the host computer. The basic configuration for the serial line should be (see chapter 4.3 "Power-Up"):

- 115200 Bps
- 8 data bits
- No parity
- 2 stop bits

Then when switching power on of the K-Junior robot, the Linux login message should be displayed on the terminal (see chapter 4.3). If no character is received, the Bluetooth connection and terminal settings should be checked. The boot can be accessed with a serial cable and level shifter adapter (see chapter 5.1.1). If the boot is interrupted at some point, especially during kernel uncompress step, then the system is probably corrupted. Chapter 4.6.3.3: “*Uploading the kernel and the file system*” describes how to upload a new system to the board. If the boot process is completed to the login prompt, then the boot messages should be checked for errors, please refer to chapter 3.3: “*KJ-EVO Software*” for further information on the KJ-EVO Linux system.

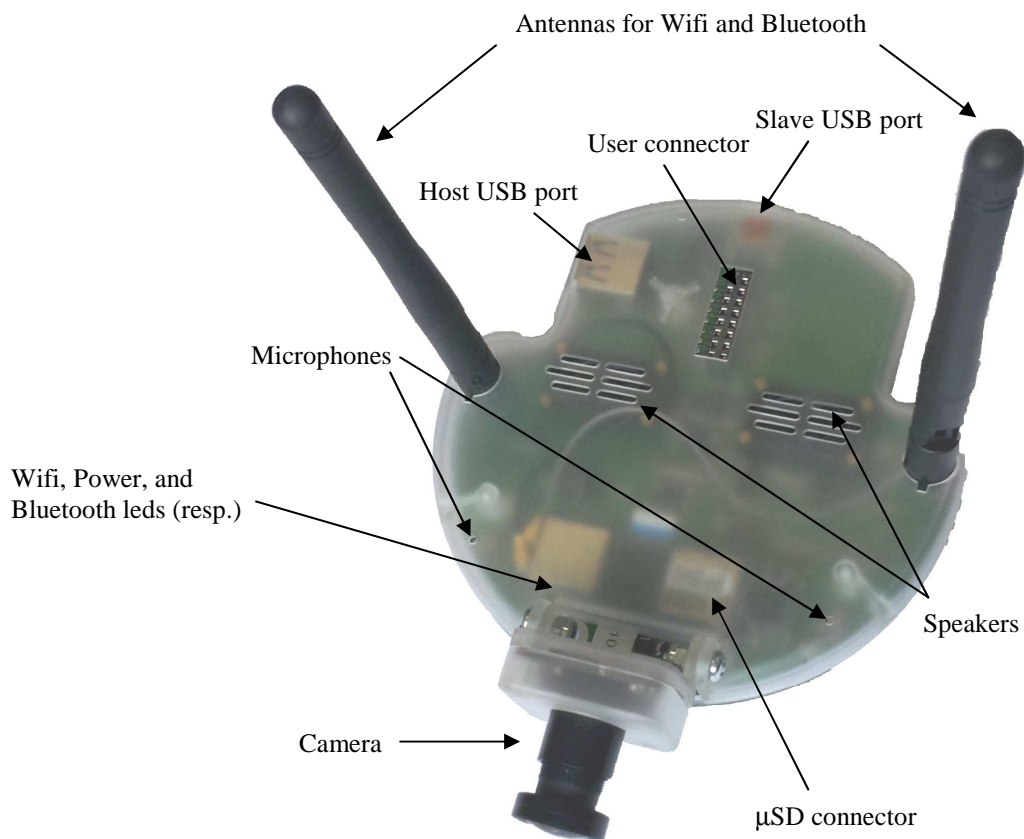
## 3. Description

---

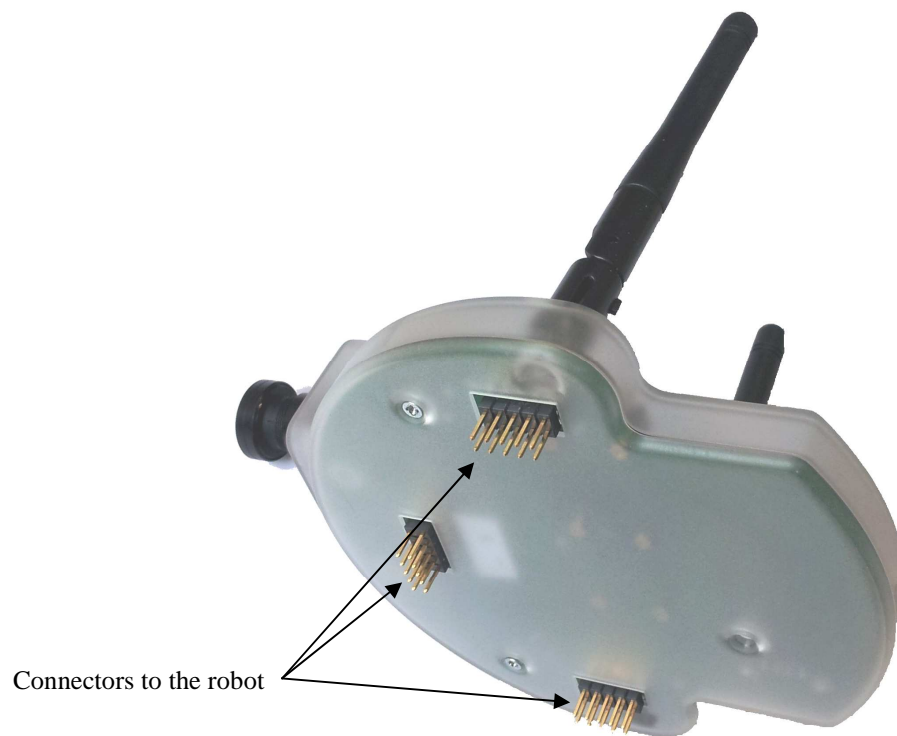
### 3.1 Overview

An overview of the KJ-EVO hardware is depicted in the Figure 3.1 and 3.2. The locations of various key elements are indicated for later references.

All the connectors are described in the subsequent part of this chapter.



*Figure 3-1: KJ-EVO overview – top view*



*Figure 3-2: KJ-EVO overview – bottom view*

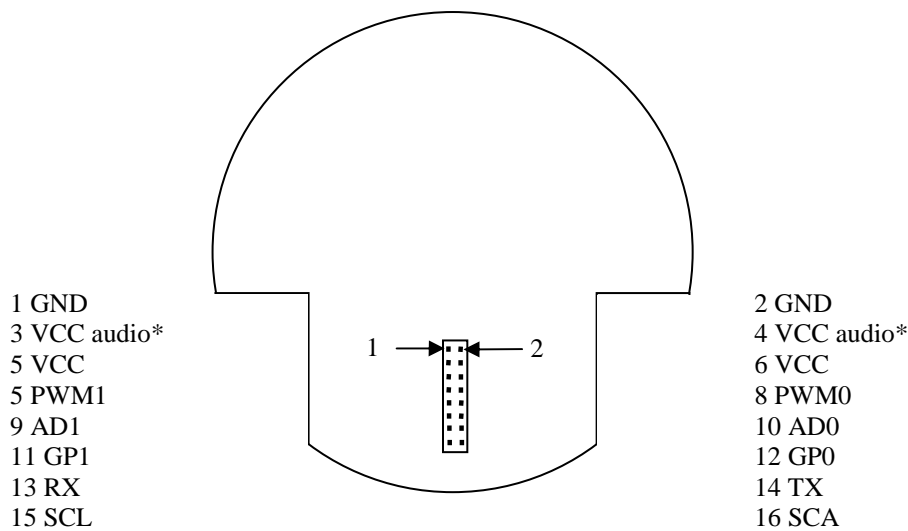
## 3.2 KJ-EVO Hardware

The hardware of the KJ-EVO is described in the sub-chapters thereafter.

### 3.2.1 User connector

The KJ-EVO user connector provides signals for several interfaces. All input-output are in the range of 0-5 V. There are 1 serial, 1 i2c, 2 PWM, 2 GPIO, 2 AD, 2 5V and 2 ground pins as described in figure 3.3 below. For specific usage, see source code examples on chapter 4.5.2.1.

For using a standard RS232 cable, you should use a level shifter for connecting to the serial TX and RX pins.



*Figure 3-3: User connector and pins descriptions*

\*voltage of the battery (3.7-4.2 V): for the old K-Junior ( revision A) coming from KJ-EVO battery; for the new K-Junior (revision B), coming from the robot battery

### 3.2.2 Reset

The KJ-EVO reset can be triggered by pressing the reset button (see Figure 3-1). When pressing the reset button, the board core and the robot are reset.

### 3.2.3 USB Host

The KJ-EVO provides two different USB interfaces. One is the PXA270 USB device interface and the other one is a complete USB Host interface. The USB Host interface provides a way to connect USB devices to the KJ-EVO, such as webcams or external disks. The space was designed specially for an USB memory stick.

The connected USB memory stick will be mounted autonomously on */media/sda1*. For removing it, execute command ***umount /media/sda1*** and finally unplug the stick.

### 3.2.4 $\mu$ -SD connector

With the  $\mu$ -SD connector and a  $\mu$ -SD card, you can easily transfer files to and from the KJ-EVO.

Power off the KJ-EVO; insert the uSD card; power on the KJ-EVO.

Typing mount shows the mounted card location:

...

/dev/mmcblk0p1 on /media/mmcblk0p1 type vfat ...

=> You can copy files to and from the card at the location /media/mmcblk0p1

### 3.2.5 Leds

There are 3 leds indicating the power status (green for ON), the Wifi network connection and the Bluetooth connection, both blue for connection (see Figure 3-1).

### 3.2.6 USB slave

With the USB slave connector of type mini-B, you can plug an USB cable to a computer and control the console of the KJ-EVO board. See chapter 5.1.4 for usage.

### 3.2.7 Microphones

There are 2 microphones for recording ambient sound. See chapter for 5.1.8 usage.

### 3.2.8 Speakers

There are 2 speakers for playing stereo sound. See chapter 5.1.8 for usage.

### 3.2.9 Camera

There is a color camera integrated to the board. Its resolution is 752x480 pixels and can take images up to 60 fps. See chapter 5.1.6 for usage.

### 3.2.10 Connectors to the robot

There are 3 groups of pins holes to be connected to the robot. See figure 3.4, chapter 4.2 and K-Junior user's manual for usage.

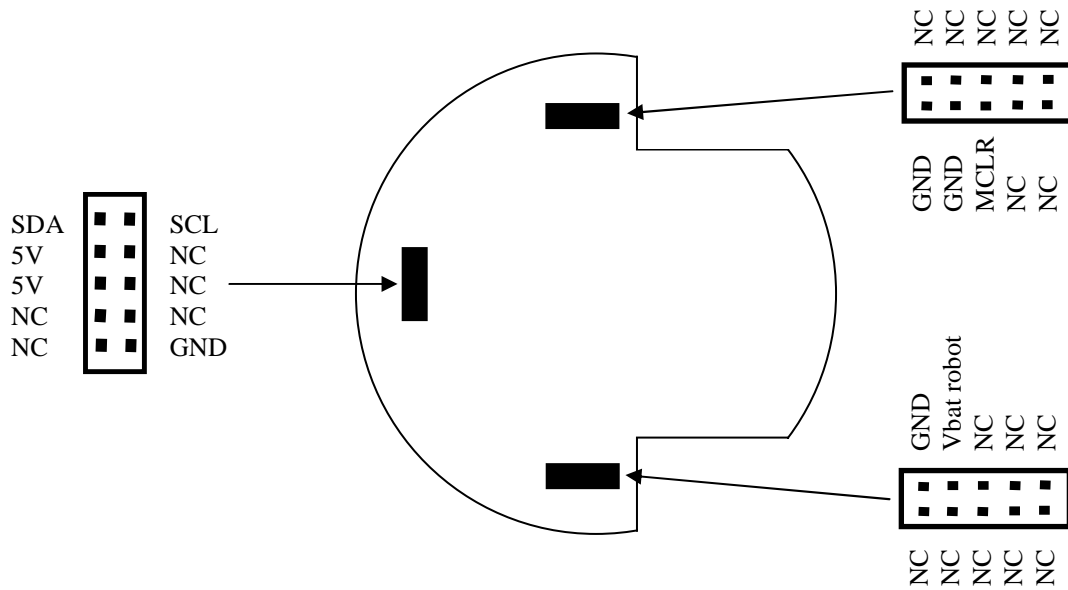


Figure 3-4: robot connectors of the KJ-EVO (bottom view)

### 3.2.11 Wifi and Bluetooth

The Wifi G and Bluetooth connections are available. See chapter 4.3 for Bluetooth and chapter 5.1.2 for Wifi.

### 3.3 KJ-EVO Software

A KJ-EVO embedded system is based on two main software components. One is the Linux Operating System kernel, and the second is a set of software packages that is called a distribution.

The KJ-EVO uses the Linux distribution called "Angström OpenEmbedded".

The Linux kernel is based on standard Linux kernel sources, with adaptation made by Gumstix. Further, exhaustive, information about Linux kernel is available from many sources, especially on the web, starting from <http://www.kernel.org>. The installed distribution is based on the handhelds familiar distribution, which is specially designed for embedded systems. Please visit the "Angström" website <http://www.angstrom-distribution.org>, for further information about this project. The main components within this distribution are described in the following sections.

Using the KJ-EVO, Linux system should be pretty straightforward for users with a Linux or Unix background considering that all components are standard.

In the next chapter 4 "*Usage*

", the installation and usage of the KJ-EVO software is described. And in the Annexes, tools for customizing, configuring and using specific tools are detailed.

## 4. Usage

---

### 4.1 Required hardware / software

The required hardware and software to use the board and develop programs are described below.

#### 4.1.1 Required hardware

- Computer with Bluetooth or USB port with Linux or Linux emulation\* (not included)
- K-Junior robot (version V2) with its USB cable

#### 4.1.2 Required software

Required free space:

- 3 GB on */usr/local* (for light toolchain)
- 15 MB on user account (~/)

Required files:

- Linux OS\* (kernel 2.6.x) on the computer with the following packages installed:
  - *gcc* : GNU C compiler
  - *minicom* : terminal emulation
  - *lrzsz* : communication package
- from the Internet: <http://ftp.k-team.com/K-JuniorV2/extensions/kj-evo/software> :
  - Cross-compiler light : *kjunior-oetools-1.0-light-kb1.tar.bz2*
  - Environment script : *env.sh*
  - Board library sources : *kj-evo\_library\_1.0.tar.bz2*
  - K-Junior slave firmware : *kjOS\_Slave\_B-01.hex*

Remarks: you may find updated version of these software at:

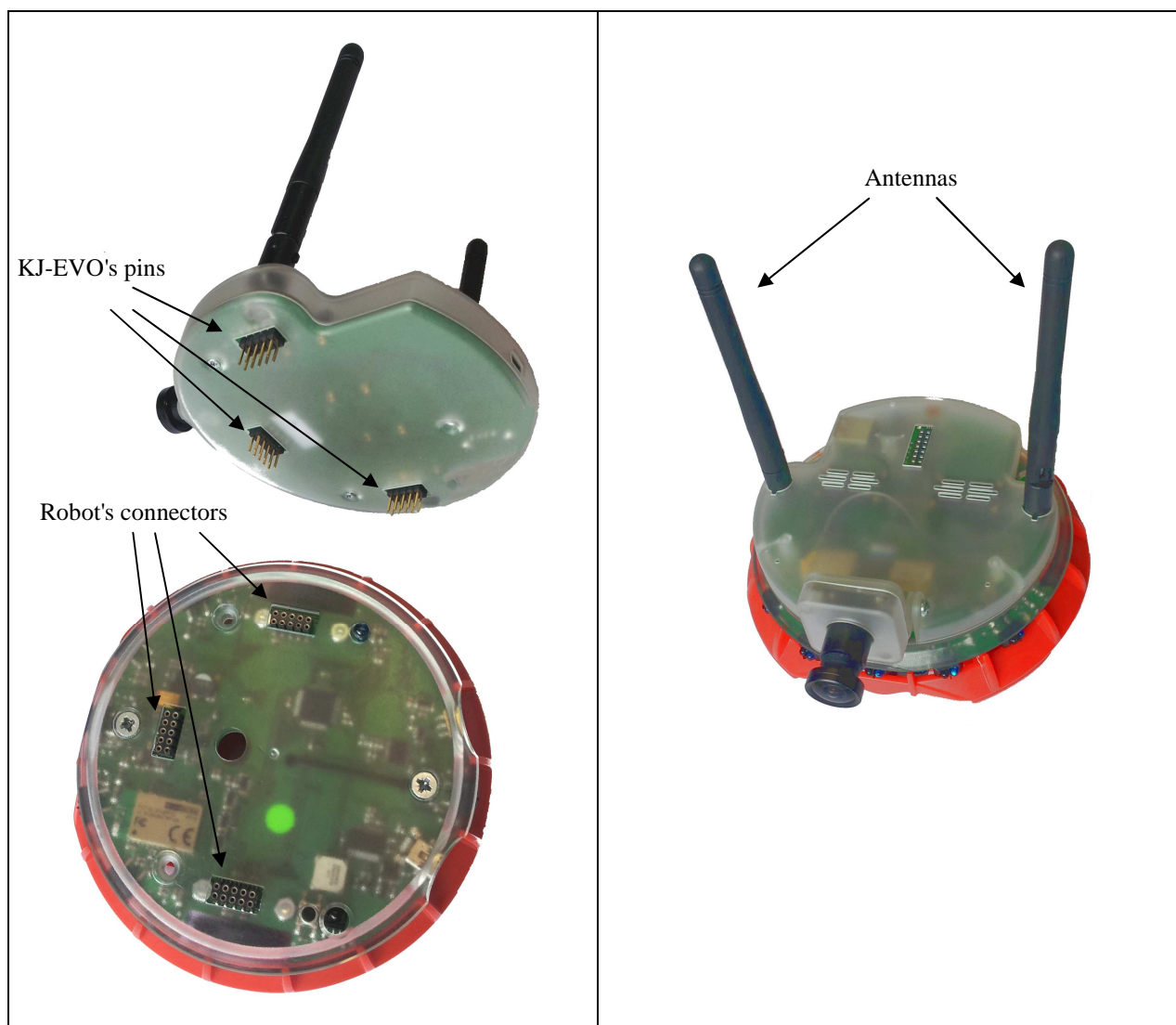
<http://ftp.k-team.com/K-JuniorV2/extensions/kj-evo/software>

\*With a virtual machine, you can also use the toolchain. See chapter 5.1.9.



## 4.2 Assembly

The assembly of the board with the robot is depicted in Figure 4-1. Insert the 3 groups of pins of the KJ-EVO into the 3 connectors of the robot. Deploy the 2 antennas.



*Figure 4-1: Basic assembly*

### 4.3 Power-up and console access

You can have console access through different ways (serial port: chapter 5.1.1, USB: chapter 5.1.4, Wifi: chapter 5.1.2). The following explanation will be for the Bluetooth:

- 1) Install the Linux package **lrzsz** containing communications programs. If your Linux distribution is Ubuntu:

***sudo apt-get install lrzsz***

- 2) On the Linux computer, run the emulation terminal Minicom:

***sudo minicom***

If Minicom is not installed you have to install this package. For Linux Distribution “Ubuntu”, the command is:

***sudo apt-get install minicom***

Install blueteman , a Bluetooth manager for Linux:

***sudo apt-get install blueteman***

- 3) Change the firmware version of your K-Junior v2 robot. The robot needs to be reflashed in slave mode with the special firmware ***kjOS\_Slave\_B-01.hex***. See the **K-Junior V2 User's Manual**, chapter 5.1 entitled **K-Junior Uploader** for uploading that file.

You can find a normal version of the firmware if you want to revert it at:

<http://ftp.k-team.com/K-JuniorV2/Software/Firmware/>

- 4) After having assembled the KJ-EVO board to the K-Junior robot (chapter 4.2), switch the robot on.
- 5) Go to the Bluetooth configuration of your computer and search for a new device (with ***blueteman***). The KJ-EVO board will appear as ***KJ-EVO-ABCD***, where ***ABCD*** is the serial number (Don't connect to ***K-Junior EFGH***, which is the robot itself). Pair the robot with the access key: ***0000*** (four zeros). With ***blueteman***, right click on the paired ***KJ-EVO-ABCD***, choose "***Connect to: Serial Port***".
- 6) Run minicom with the command: ***sudo minicom -o***
- 7) Set its parameters with the sub-menu “Serial port setup” of the menu [configuration] (keys Ctrl-a + o) as described in Figure 4-2. Push RETURN key to validate each time.

```
+-----+
| A -   Serial Device       : /dev/rfcomm0 |
| B - Lockfile Location     : /var/lock      |
| C -   Callin Program      :                |
| D -   Callout Program     :                |
| E -   Bps/Par/Bits        : 115200 8N1     |
| F - Hardware Flow Control : No              |
| G - Software Flow Control : No              |
|                                     |
|      Change which setting?              |
+-----+
```

*Figure 4-2: Minicom serial parameters*

- 8) Save the settings with the command “*Save setup as ...*” of the menu [configuration] (cf Figure 4-3) and choose *bluetooth*. You will be able to run again the program and load this configuration with: *sudo minicom -o bluetooth*

```
+-----[configuration]-----+
| Filenames and paths          |
| File transfer protocols      |
| Serial port setup           |
| Modem and dialing           |
| Screen and keyboard         |
| Save setup as dfl           |
| Save setup as..             |
| Exit                         |
+-----+-----+-----+-----+
```

*Figure 4-3: Minicom configuration menu*

- 9) Push RETURN key and the prompt of the KJ-EVO board will be available (Figure 4-4):

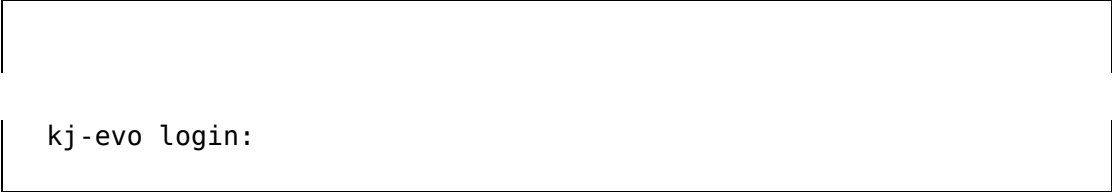
# Angstrom

The Angstrom Distribution kj-evo rfcomm0

Angstrom 2010.7-test-20110202 kj-evo rfcomm0

The Angstrom Distribution kj-evo rfcomm0

Angstrom 2010.7-test-20110202 kj-evo rfcomm0



```
kj-evo login:
```

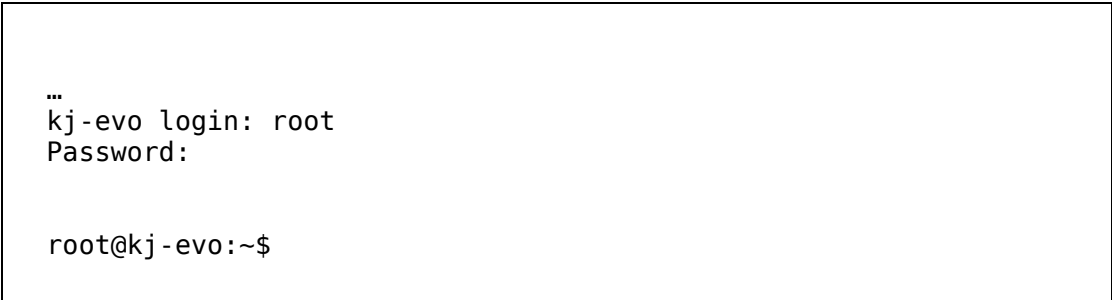
*Figure 4-4: KJ-EVO prompt*

10) Login to the KJ-EVO with the following parameters:

**Login:** *root*

**Password:** *(none, press “Return” key)*

=> You are at the prompt of the Linux console of the KJ-EVO (see Figure 4-5).



```
...  
kj-evo login: root  
Password:  
  
root@kj-evo:~$
```

*Figure 4-5: KJ-EVO prompt: logged*

11) In the directory **examples** (*~/examples*), you will have the library examples compiled. You can execute each by preceding its name with **./** :

**cd examples**

**./adc\_example**

You can add a password to the login of your KJ-EVO board with the command **passwd** . You can upload or upload a file with the KJ-EVO at chapters 5.1.1.2 and 5.1.1.3 .

Continue with next chapter 4.5 for starting programming.

## 4.4 Shutdown / Reboot / Reset

### **Shutdown:**

While logged into the console of the KJ-EVO, it is better to do a proper shutdown than just switching off the K-Junior robot. This will ensure that the open files are correctly closed and settings saved.

- 1) Run the following command in the console:

***shutdown -h now***

- 2) Wait 15s (if you are logged through the serial cable, you wait until *System halted* appears). Then you can switch off the robot.

### **Reboot:**

If you want to reboot, you can use:

***shutdown -r now***

### **Reset :**

The KJ-EVO reset can be triggered by pressing the reset button (see Figure 3-1). When pressing the reset button, the board core and the robot are reset.

You may need the reset while the software is blocked and you cannot access anymore by another connection with *ssh* (see chapter 5.1.5) for killing the annoying process or doing a soft reboot.

## 4.5 Software

The following sub-chapters explain the software installation and the application development with the board.

Two development packages are available:

- Light toolchain
- Full toolchain and sources: for advanced users; kernel modifications; packages creation/addition

In the subsequent paragraphs, only the light tool chain is explained. The full toolchain is described in the following chapter 4.6.3. With a virtual machine, you can also use the toolchain from another operating system (see chapter 5.1.9).

### 4.5.1 Installation of light toolchain

The installation of the software required to use the board and the development tool is described in the next sub-chapters.

Below are listed the files needed to install the light toolchain. These files are available in the DVD or from the Internet url: <http://ftp.k-team.com/K-JuniorV2/extensions/kj-evo/software/>

From *light\_toolchain/* folder:

- Light toolchain (cross-compiler only):  
*kjunior-oetools-1.0-light-kb1.0.tar.bz2* \*

From *common\_files/* folder:

- Environment script: *env.sh*

From *library/* folder:

- Board library sources: *kj-evo\_library\_1.0.tar.bz2* \*

\* where 1.0 is the release version (may change  
without notice)

#### 4.5.1.1 Installation of the development directory

The development directory will be the base folder for your development. It contains links and scripts to easily use the cross-compiler to make your programs.

- 1) Create a new development folder *~/kj-evo\_development* in your home directory and enter into it. You can use the following commands, assuming you are in a console.

```
mkdir ~/kj-evo_development
```

```
cd ~/kj-evo_development
```

- 2) Obtain the environment script file *env.sh* from the folder above and put it in your new development directory *~/kj-evo\_development* .
- 3) If you gave any other name or path to the development folder you will have to modify the **KTEAM\_HOME** variable in the file **env.sh** to point to your development directory:

Default:

```
KTEAM_HOME=~/kj-evo_development
```

To be modified to:

```
KTEAM_HOME=FULL_PATH_OF_YOUR_NEW_DEV_FOLDER
```

#### 4.5.1.2 Installation of the cross-compiler (light toolchain)

- 1) Extract the cross compiler *kjunior-oetools-1.0-light-kb1.0.tar.bz2* in */usr/local* with the command:

```
sudo tar -xjf kjunior-oetools-1.0-light-kb1.0.tar.bz2 -C /usr/local
```

Remarks: You must be root or use *sudo*. You must put the tools there because there are hardcoded links.

- 2) You can check if the installation is correct by running the cross-compiler. Firstly make the environment variables available then check the version of the cross-compiler:

```
cd ~/kj-evo_development
```

```
source env.sh
```

```
arm-angstrom-linux-gnueabi-gcc --version
```

=> The last command should return:

```
arm-angstrom-linux-gnueabi-gcc (GCC) 4.3.3
```

...

#### 4.5.1.3 Installation of the board library

The library is already installed on the KJ-EVO. To install the library on your development system, follow the following instructions:

- 1) Extract the library *kj-evo\_library\_1.0.tar.bz2* in your development folder:

```
tar -xjf kj-evo_library_1.0.tar.bz2 -C ~/kj-evo_development
```

- 2) You can recompile the whole library by running the following commands in the *kj-evo\_library* folder:

```
cd ~/kj-evo_development/kj-evo_library
```

```
source ../env.sh (if not already sourced in this console before)
```

```
make clean
```

```
make
```

- 3) If you modified the library (any file in *src/*), you will have to transfer the file *build-overo/lib/libkj-evo-1.0.so* to your KJ-EVO board, overwriting */usr/lib/libkj-evo.so* .



The board library contains these files and directories:

|                               |  |
|-------------------------------|--|
| build-overo/                  | compiled library and headers                                     |
| doc/                          | documentation (API: start in <b><i>doc/html/index.html</i></b> ) |
| examples/                     | source code examples of programs                                 |
| adc_example.c                 | adc and accelerometer examples                                   |
| camera_example.c              | camera example   |
| gpio_example.c                | Digital input/output (GPIO) example                              |
| gripper_example.c             | K-Junior gripper example   |
| i2c_example.c                 | i2c usage example  |
| kjunior_and_gripper_example.c | K-Junior and gripper example                                     |
| kjunior_example.c             | K-Junior example   |
| pwm_example.c                 | PWM example  |
| sound_example.c               | Microphones/Speakers example                                     |
| src/                          | source code of the library                                       |
| template/                     | template program   |
| Makefile                      | Makefile for all   |
| README.kteam                  | readme file  |

You can find an updated version of the library from the following ftp site:

<http://ftp.k-team.com/K-JuniorV2/extensions/kj-evo/software/library/>

## 4.5.2 Programming and light toolchain usage

### 4.5.2.1 Application development

A template program *prog-template.c* is available in the board library in the folder *kj-evo\_library/template*.

You can start your code into the template program and use the following commands to build it. The first one makes the environment variables (path to the cross-compiler, libraries) available to the system and the second run the **Makefile** script to compile and build the executable program. Enter in the *kj-evo\_library/template* folder and type in a console to build the template program:

```
cd ~/kj-evo_development/kj-evo_library/template  
  
source ~/kj-evo_development/env.sh  
  
make
```

=> The “**template**” file is the executable output file.

You can transfer the program to the KJ-EVO by Bluetooth or Wifi (see chapters 5.1.2 and 5.1.3).

Then execute it on the board by running:

```
./template
```

Application Programming Interface documentation of the library is available at:

```
~/kj-evo_development/kj-evo_library/doc/html/index.html
```

#### Remarks:

If you modify the program name, you will have to modify its occurrences in the **Makefile** file.

You can find many examples of source code in *~/kj-evo\_development/kj-evo\_library/examples* (see list end of 4.5.1.3). They are compiled with the command below. To compile only the examples, run:

```
make clean_examples  
  
make kj-evo_examples
```

See next sub-chapter for using Eclipse guide for editing source code.

See chapter 5.1.10 for debugging your program.

#### 4.5.2.2 C/C++ Programming: using GUI source code editor Eclipse

You can use also Eclipse as source code editor. See instructions below how to install and use it.

Install the Java Runtime Environment (JRE) if not already installed.

Find the version corresponding to your OS here:

<http://www.eclipse.org/downloads/moreinfo/jre.php>

and the JRE file there: <http://java.com/en/download/index.jsp>

- 4) Download the linux version of "Eclipse IDE for C/C++ Developers (includes Incubating components)" from (for Ubuntu users, don't install with apt-get because, the apt-get version is older):

<http://www.eclipse.org/cdt/>

- 5) Extract the Eclipse program file:

***sudo tar -xzf eclipse-cpp-indigo-SR2-incubation-linux-gtk.tar.gz***

You can also create a link to the start file: ***sudo ln -s eclipse/eclipse /usr/bin/eclipse***

- 6) You should have installed the latest version of the kj-evo toolchain (light or full). Copy the debugger program file ***arm-angstrom-linux-gnueabi-gdb*** to ***/usr/local/kjunior-oetools-1.0/tmp/cross/bin*** on your computer.

You can check by opening a terminal and sourcing the kj-evo environment with "source env.sh" (chapter 4.5.2.1), command:

***arm-angstrom-linux-gnueabi-gdb --version***

which should return: ***GNU gdb (GDB) 7.1***

- 7) Run eclipse and at the "Workspace launcher" window, choose where you would like to put your project.
- 8) Go to file menu "File => C Project" (or C++; for running a C++ on the kj-evo, you must install the standard C++ library:

with the command ***ipkg install libstdc++6\_4.3.3-r22.1.6\_armv7a.ipk*** on the kj-evo) and choose a Project Name (ex: test).

- 9) In the next window "C Project", push the "Advanced Settings" button and on the "C/C++ Build => Discovery Options",

for c: On 'GCC C Compiler' change '***Compiler invocation command***' to ***'/usr/local/kjunior-oetools-1.0/tmp/sysroots/i686-linux/usr/armv7a/bin/arm-angstrom-linux-gnueabi-gcc'***

or (depending if your project is C or C++)

for c++: On 'GCC C++ Compiler' change '***Compiler invocation command***' to ***'/usr/local/kjunior-oetools-1.0/tmp/sysroots/i686-linux/usr/armv7a/bin/arm-angstrom-linux-gnueabi-g++'***

- 10) On C/C++ “Build => Settings”, “Cross GCC Compiler” change **'Command'** to ***'/usr/local/kjunior-oetools-1.0/tmp/sysroots/i686-linux/usr/armv7a/bin/arm-angstrom-linux-gnueabi-gcc'***
- 11) On “Cross GCC Compiler” Includes => Include paths, add:  
***/usr/local/kjunior-oetools-1.0/tmp/sysroots/i686-linux/usr/include***
- 12) On Miscellaneous, replace "Other flags" with ***"-c -march=armv7a -mtune=xscale -Wa,-mcpu=xscale"***
- 13) On the Cross GCC Linker, change **'Command'** to ***'/usr/local/kjunior-oetools-1.0/tmp/sysroots/i686-linux/usr/armv7a/bin/arm-angstrom-linux-gnueabi-gcc'***
- 14) On “Cross GCC Linker => Libraries” at "Libraries (-l), add ***kj-evo*** with the + button on the upper right.
- 15) At "Libraries search path", add ***'/usr/local/kjunior-oetools-1.0/tmp/sysroots/armv7a-angstrom-linux-gnueabi/lib'***
- 16) On "Cross GCC Assembler", in Command field, modify to ***: /usr/local/kjunior-oetools-1.0/tmp/sysroots/i686-linux/usr/armv7a/bin/arm-angstrom-linux-gnueabi-as***
- 17) Choose “C/C++” Build menu at the left, “Discovery options” repeat the steps above from 7) for the Release configuration.
- 18) push “Finish” button.
- 19) Go to menu “File => New => Source file” choose ***test.c*** as filename
- 20) Close the Welcome window with its cross at the upper left.

21) Insert in the *test.c* file the following C source code:

```
#include <stdlib.h>
#include <stdio.h>
#include <kj_evo/kj_evo.h>

int main(int argc, char *argv[]) {
    int rc;

    // initialise kj-evo library
    if((kj_evo_init( 0 , NULL )) < 0 )
    {
        fprintf(stderr,"Error: Unable to initialize the kj-evo
library!\r\n");
        return -1;
    }

    printf("Library kj-evo Template Program\r\n");
}
```

22) Then cross-compile the project with menu “Menu Project => Build-All”

=> The output file will be in the subdirectory Debug or Release of the project.

23) Transfer the file "test" to your kj-evo (see chapter 4.5.2.3).

24) And execute the program file with command: *./test*

#### 4.5.2.3 Transferring files

You can transfer files to and from the KJ-EVO board by different means. The default one is Bluetooth (see chapter 4.3). You have also serial (chapter 5.1.1). But USB slave (chapter 5.1.4) is faster and Wifi the fastest way (chapter 5.1.3).

## 4.6 Full toolchain and sources

The full toolchain is for advanced users who would like to modify the kernel, rebuild the image system or develop new packages for the KJ-EVO.

Remarks: Prior knowledge of Linux, its kernel and Open Embedded tools is highly recommended.

### 4.6.1 Required software

Required free space:

- 38 GB on */usr/local* (51 GB including the packed tools)
- 15 MB on user account ( *~/* )

Required files:

- Linux packages
  - *g++* : GNU C++ compiler
  - *patch* : patch software
  - *help2man* : manual converter
  - *diffstat* : reads the output of *diff* and displays a histogram
  - *texi2html* : convert to html
  - *makeinfo* : produce doc (*texinfo* on *Ubuntu*)
  - *ncurses-dev* : library allowing the programmer to write (*libncurses5-dev* on *Ubuntu*)
  - *cvs* : revision control system
  - *gawk* : programming language designed for processing text-based data
  - *python-dev* : dynamic object-oriented programming language
  - *python-pysqlite2* : Python sql interface
  - *subversion* : version control system
  - *git* : fast version control system
  - *chrpath* : allows you to change the rpath (where the application looks for libraries) in an application

On *Ubuntu* Linux distribution, you can use the following command to install all the above packages in one time:

```
sudo apt-get install g++ patch help2man diffstat texi2html texinfo libncurses5-dev  
cvs gawk python-dev python-pysqlite2 subversion git-core chrpath
```

- Included in the DVD-ROM of the package:
  - Cross-compiler and  
“Open Embedded” tools sources : ***kjunior-oetools-1.0-kb1.0.tar.bz2 \****
  - Environment script : ***env.sh***
  - Board library sources : ***kj-evo\_library\_1.0.tar.bz2 \****
  - Script and files for uploading kernel  
and file system : ***flash, MLO, u-boot.img***

*\* where 1.0 is the release version (may change without notice)*

#### **4.6.2 Installation**

- 1) Install the development directory as explained in chapter 4.5.1.1: “*Installation of the development directory*” if not already done.
- 2) Extract the cross compiler sources ***kjunior-oetools-1.0-kb1.0.tar.bz2*** in ***/usr/local*** with the command:

```
sudo tar -xjf kjunior-oetools-1.0-kb1.0.tar.bz2 -C /usr/local
```

**Remark:** you must put the tools there because there are hardcoded links. You have to overwrite the light tools if they were already installed.

- 3) You can check if the installation is correct by running the cross-compiler. Firstly make the environment variables available then check the version of the cross-compiler:

```
source ~/kj-evo_development/env.sh
```

```
arm-angstrom-linux-gnueabi-gcc -version
```

=> The last command should return:

```
arm-angstrom-linux-gnueabi-gcc (GCC) 4.3.3
```

- 4) Install the board library as explained in chapter 4.5.1.3: “*Installation of the board library*”.

### 4.6.3 Full toolchain usage

#### 4.6.3.1 Rebuilding the whole system

To rebuild the toolchain system, the image file and the kernel, execute the following instructions:

- 1) In a console in the directory */usr/local/kjunior-oetools-1.0*, source the following file to have access to the environment setup:

*source extras/profile*

- 2) With the following commands you can rebuild the whole system:

- Cleaning : *bitbake -c clean kj-evo-image*
- Rebuild : *bitbake kj-evo-image*

=> The output files will be stored in:

*/usr/local/kjunior-oetools-1.0/tmp/deploy/glibc/images/overo*

Three main files will then be built:

the kernel : *uImage-2.6.34-r97-overo.bin*

its modules : *modules-2.6.34-r97-overo.tgz*

the file system :

*Angstrom-kj-evo-image-glibc-ipk-2010.7-test-20110718-overo.rootfs.tar.bz2 \**

*\* where 20110718 is the release version*

#### Remarks:

You may find updated version of these software at:

<http://ftp.k-team.com/K-JuniorV2/extensions/kj-evo>

With the chapter 4.6.3.3: “*Uploading the kernel and the file system*”, you can upload these files to the KJ-EVO.

You can find more information about *bitbake*, the tool for executing task and managing metadata here: <http://docs.openembedded.org/bitbake/html>

And information about the OEtools is available here:

- Cross-compiler and tools (OpenEmbedded):  
[http://www.openembedded.org/wiki/Main\\_Page](http://www.openembedded.org/wiki/Main_Page)
- Linux distribution of the KJ-EVO: <http://www.angstrom-distribution.org>
- Gumstix documentation: <http://www.gumstix.org>



#### 4.6.3.2 Kernel modification

You can modify the kernel here by accessing to its menu with these commands:

```
cd /usr/local/kjunior-oetools-1.0/tmp/work/overo-angstrom-linux-gnueabi/linux-omap3-2.6.34-r97/git/
```

and configure the kernel with:

```
make ARCH=arm menuconfig
```

copy ".config" file to **/usr/local/kjunior-oetools-1.0/user.collection/packages/linux/linux-omap3-2.6.34/overo/defconfig** (defconfig is the new filename)

Then you rebuild it and the file system by executing these commands at the root of the *KJ-EVO oetools*:

```
cd /usr/local/kjunior-oetools-1.0
```

```
source build/profile
```

```
bitbake -c clean virtual/kernel
```

```
bitbake virtual/kernel (may take up to 30 min depending of your computer)
```

⇒ The new kernel **uImage** file will be located at:  
**/usr/local/kjunior-oetools-1.0/tmp/deploy/glibc/images/overo/uImage-overo.bin**

Build also the *filesystem* image:

```
bitbake -c clean kj-evo-image
```

```
bitbake kj-evo-image
```

⇒ The new *filesystem* will be located at:  
**/usr/local/kjunior-oetools-1.0/tmp/deploy/glibc/images/overo/kj-evo-image-overo.tar.bz2**

You can now upload the files to the KJ-EVO with chapter 4.6.3.3: “*Uploading the kernel and the file system*”.

### 4.6.3.3 Uploading the kernel and the file system

You may need to reflash your KJ-EVO board by uploading the kernel and filesystem if you want to upgrade or if something went wrong.



**WARNING:** All the data on the KJ-EVO will be lost after running the following instructions!

Material needed:

- uSD card with uSD reader
- level shifter for serial port and modified serial cable (see chapter 5.1.1)
- *flash*, *MLO* and *u-boot.img* script from DVD or K-Team's ftp

- 1) Firstly you will need to make 2 partitions on your uSD card. Backup all the data on your uSD card before.

Follow the chapters "Partitioning the Card", "Formatting the New Partitions" and "Installing the Boot Files" the instructions there:

<http://www.gumstix.org/create-a-bootable-microsd-card.html>

- 2) You will take the *uImage* and *Angstrom-kj-evo-image-glibc-ipk-2010.7-test-20110718-overo.rootfs.tar.bz2* \*

\* where 20110718 is the release version

from K-Team ftp or from your full oetools directory (see chapter 4.6.3.2):

*/usr/local/kjunior-oetools-1.0/tmp/deploy/glibc/images/overo/*

- 3) Copy also the *kj-evo-image-overo.tar.bz2* and *flash* into your *uSD* in the first partition.  
The names of the kernel *uImage* and the filesystem *kj-evo-image-overo.tar.bz2* must match with the ones in the *flash* file.
- 4) You will need a serial connection to the KJ-EVO (see chapter 5.1.1).  
Insert the uSD created above and power-up your KJ-EVO.

- 5) Login and run the flash script with the commands:

*cd /media/mmcblk0p1/flash*

*./flash*

- 6) Shutdown with the command :

*shutdown -h now*

- 7) Wait until *System halted* appears, unplug the power or shutdown the K-Junior robot if mounted and remove the  $\mu$ SD card.

#### 4.6.4 Packages installations

With OpenEmbedded, you can easily cross-compile existing packages or add your own. You can list installed packages on the board with the command: *opkg list\_installed*

##### 4.6.4.1 Existing packages

Many packages are available for Open-Embedded. Here below are the instructions to add an existing package:

- 1) Check if there is already a recipe package is already in */usr/local/kjunior-oetools-1.0/org.openembedded.dev/recipes*
- 2) If it is present, you can compile it by running in the */usr/local/kjunior-oetools-1.0* directory:

*source extras/profile*

*bitbake PACKAGE\_NAME*

- 3) The package will be created in one of the folders into:

*/usr/local/kjunio-oetools-1.0/tmp/deploy/glibc/ipk/*

- 4) Transfer the package to the KJ-EVO (with Minicom or ssh: see chapter 5.1.1.2 "*To send a file to the KJ-EVO (upload)*" or chapter 5.1.3 "*Transferring files using scp*").
- 5) Then install it:

*opkg install PACKAGE\_NAME.ipk*

##### 4.6.4.2 Removing packages

You can remove a package with the command:

*opkg remove PACKAGE\_NAME*

You may have to add the command parameter *--force-depends* if the package depends on others.

##### 4.6.4.3 Creating new package:

You can create new packages for the KJ-EVO, following the instructions there:

<http://www.gumstix.org/software-development/open-embedded/160-bitbake-tutorial.html>

And if you installed the full toolchain, an example is available into */usr/local/kjunior-oetools-1.0/user.collection/recipes/helloworld* .

## 5. Annexes

### 5.1 Tools and commands

In this part, the detailed descriptions of several tools and helpful commands are explained.

#### 5.1.1 Using the serial port and Minicom

The User connector (chapter 3.2.1) has a serial port that can be used to connect the KJ-EVO board to a computer.



**Warning:** Pay attention to the Voltage 0-5V of the User connector. It is not the standard RS232 voltage level! You will need a voltage level shifter.

- 1) Install the Linux package **lrzsz** containing communications programs. If your Linux distribution is Ubuntu:

```
sudo apt-get install lrzsz
```

- 2) On the Linux computer, run the emulation terminal Minicom:

```
minicom
```

If Minicom is not installed you have to install this package. For Linux Distribution “Ubuntu”, the command is:

```
sudo apt-get install minicom
```

- 3) Set its parameters with the sub-menu “Serial port” setup of the menu [configuration] (keys Ctrl-a + o) as described in Figure 5-1.

You may modify the serial port device name depending where you plugged your serial port (see chapter 5.1.1 “*Using the serial port and Minicom*”).

If you use a serial to USB adapter, the serial device may be */dev/ttyUSB0*.

```
+-----+
| A -   Serial Device       : /dev/ttyS0 |
| B - Lockfile Location    : /var/lock   |
| C -   Callin Program     :             |
| D -   Callout Program    :             |
| E -   Bps/Par/Bits       : 115200 8N1  |
| F - Hardware Flow Control : No         |
| G - Software Flow Control : No         |
|                               |
|   Change which setting?   |
+-----+
```

Figure 5-1: Minicom serial parameters

Save the settings with the command “*Save setup as dfl*” of the menu [configuration] (cf Figure 5-2).

```
+-----[configuration]-----+
| Filenames and paths          |
| File transfer protocols      |
| Serial port setup           |
| Modem and dialing           |
| Screen and keyboard         |
| Save setup as dfl           |
| Save setup as..             |
| Exit                        |
+-----+
```

*Figure 5-2: Minicom configuration menu*

#### 5.1.1.1 Establish the serial connection

Set its parameters with the sub-menu “Serial port” setup of the menu [configuration] (Figure 5-4) accessed with keys “Ctrl-a + o” as described in Figure 5-3.

You may modify the serial port device name depending where you plugged your serial port. If you use a serial to USB adapter, the serial device may be /dev/ttyUSB0.

```
+-----+
| A -   Serial Device       : /dev/ttyS0      |
| B - Lockfile Location    : /var/lock        |
| C -   Callin Program      :                 |
| D -   Callout Program     :                 |
| E -   Bps/Par/Bits        : 115200 8N1      |
| F - Hardware Flow Control : No              |
| G - Software Flow Control : No              |
|                                     |
|      Change which setting?              |
+-----+
```

*Figure 5-3: Minicom serial parameters*

Save the settings with the sub-menu “*Save setup as dfl*” (Figure 5-4).

```

+-----[configuration]-----+
| Filenames and paths         |
| File transfer protocols     |
| Serial port setup          |
| Modem and dialing           |
| Screen and keyboard         |
| Save setup as dfl           |
| Save setup as..             |
| Exit                        |
+-----+

```

*Figure 5-4: Minicom configuration menu*

```

Instruments X-Loader 1.4.4ss (Oct 20 2010 - 10:10:28)
OMAP3503-GP ES3.1
Board revision: 0
Loading u-boot.bin from nand

U-Boot 2010.09 (Oct 20 2010 - 10:11:49)

OMAP3503-GP ES3.1, CPU-OPP2, L3-165MHz, Max CPU Clock 600 mHz
Gumstix Overo board + LPDDR/NAND
I2C:  ready
DRAM:  256 MiB
NAND:  256 MiB
*** Warning - bad CRC or NAND, using default environment
.
.
.

NAND read: device 0 offset 0x280000, size 0x400000
4194304 bytes read: OK
## Booting kernel from Legacy Image at 82000000 ...
   Image Name:   Angstrom/2.6.34/overo
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3255012 Bytes = 3.1 MiB
   Load Address: 80008000
   Entry Point:  80008000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK

OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
Linux version 2.6.34 (jtharin@KHEPERA04) (gcc version 4.3.3
(GCC) ) #1 Wed Jun 29 14:46:53 CEST 2011.
.
.
.

```

```
The Angstrom Distribution kj-evo ttyS2
```

```
Angstrom 2010.7-test-20110202 kj-evo ttyS2
```

```
kj-evo login:
```

*Figure 5-5: part of the KJ-EVO Boot log*

#### **5.1.1.2 To send a file to the KJ-EVO (upload)**

- 1) In the Minicom console, hold the keys “**Ctrl + a**” and press “**s**” and select “Z-Modem”.
- 2) Select the file you would like to upload to the KJ-EVO (navigate with the arrows keys, 2x “**spacebar**” to change directory and “**spacebar**” to select the file).

Select [Okay] to send it.

#### **5.1.1.3 To send a file to the computer (download)**

In the Minicom console at the prompt of the KJ-EVO, type the following command, where FILENAME is the file you would like to send.

***lsz FILENAME***

=> The file FILENAME is sent to the last directory Minicom used (or if not changed, where it started).

### 5.1.2 Using Wifi

There are different ways described below for configuring the Wifi network, depending on the given security.

You can transfer file with the *scp* command (see chapter 5.1.3). You can have remote access to the KJ-EVO with chapter 5.1.5.

#### 5.1.2.1 Without any encryption for security:

- Modify the file */etc/network/interfaces* with your wireless network settings:

```
YOUR_SSID_OF_NETWORK the SSID of your wireless network
YOUR_IP_ADDRESS      the ip address you would like for the KJ-EVO
YOUR_NETMASK         the netmask of your wireless network
YOUR_GATEWAY_IP      the gateway of your wireless network

/***** /etc/network/interfaces file *****/
.
.
.
auto wlan0
iface wlan0 inet dhcp
iface wlan0 inet static
    wireless_mode managed
    wireless_essid YOUR_SSID_OF_NETWORK
    address YOUR_IP_ADDRESS
    netmask YOUR_NETMASK
    gateway YOUR_GATEWAY_IP
.
.
.
/*****/
```

The 3 last following steps are optional but will get you Internet connection:

- delete the link */etc/resolv.conf* : *rm /etc/resolv.conf*
- insert the local domain name in */etc/resolv.conf*

```
echo search YOUR_LOCAL_DOMAIN_NAME >>/etc/resolv.conf
```

- and the dns server

```
echo nameserver YOUR_DNS_SERVER_IP_ADDRESS >>/etc/resolv.conf
```



### 5.1.2.2 WEP encryption support

a) for configuring the wifi connection, type:

```
iwconfig wlan0 essid YOUR_SSID_OF_NETWORK
```

b) if the network is secured, enter the key by typing :

```
iwconfig wlan0 key YOUR_KEY
```

c) then set an ip address to the KJ-EVO:

```
ifconfig wlan0 YOUR_IP_ADDRESS
```

d) configure the gateway by entering the gateway ip:

```
route add default gw YOUR_GATEWAY_IP wlan0
```

e) insert the local domain name in */etc/resolv.conf*

```
echo search YOUR_LOCAL_DOMAIN_NAME >>etc/resolv.conf
```

f) and the dns server

```
echo nameserver YOUR_DNS_SERVER_IP_ADDRESS >>/etc/resolv.conf
```

You can also create a file in */etc/network/if-pre-up.d* named wireless to have these settings saved.

Put the following into it:

```
#!/bin/sh  
ifconfig wlan0 up  
iwconfig wlan0 essid YOUR_SSID_OF_NETWORK  
iwconfig wlan0 key s:YOUR_KEY  
ifconfig wlan0 YOUR_IP_ADDRESS  
route add default gw YOUR_GATEWAY_IP wlan0
```

And the following in a file named */etc/resolv.conf*:

```
search YOUR_LOCAL_DOMAIN_NAME  
nameserver YOUR_DNS_SERVER_IP_ADDRESS
```

### 5.1.2.3 WEP, WPA and other encryptions:

a) Create a file named */etc/wpa\_supplicant/wpa\_supplicant.conf* and insert your selected wireless encryption:

WEP:

```
#Shared WEP key connection (no WPA):
network={
    ssid="YOUR_SSID"
    key_mgmt=NONE
    wep_key0="YOUR_WEP_KEY"
    auth_alg=SHARED
    wep_tx_keyidx=0
    priority=5
}
```

WPA-TKIP:

```
#/etc/wpa_supplicant/wpa_supplicant.conf
#with WPA-PSK TKIP:
network={
    ssid="YOUR_SSID"
    psk="YOUR_PASS_KEY"
    key_mgmt=WPA-PSK
    group=TKIP
    pairwise=TKIP
    proto=WPA
    priority=5
}
```

You can check the following link for other encryptions:

[http://hostap.epitest.fi/wpa\\_supplicant/](http://hostap.epitest.fi/wpa_supplicant/)

b) run the daemon controlling the wireless connection with the following command:

```
wpa_supplicant -c /etc/wpa_supplicant/wpa_supplicant.conf -i wlan0 -Dwext -B
```

c) In */etc/network/if-pre-up.d* named wireless, add the following commands:

```
#!/bin/sh
ifconfig wlan0 up
ifconfig wlan0 YOUR_IP_ADDRESS
route add default gw YOUR_GATEWAY wlan0
wpa_supplicant -c /etc/wpa_supplicant/wpa_supplicant.conf -i wlan0 -Dwext -B
```

d) reboot the system or restart the network with the following command:

```
/etc/init.d/networking restart
```

### 5.1.3 Transferring files using scp (ssh)

- 1) Establish a network connection between the computer and the KJ-EVO, either with Wifi or with USB slave.
- 2) Execute the following command:

```
scp FILE root@KJ-EVO_IP:/home/root
```

where **FILE** : is the file to transfer,

**KJ-EVO\_IP** : is the KJ-EVO ip address.

Push the Return key when a password is asked.

### 5.1.4 Networking over USB slave

Connect an USB cable to the mini USB slave port of the KJ-EVO board (chapter 3.2.6). By default the KJ-EVO USB slave port is configured into file */etc/network/interfaces* with:

```
Interface name: usb0  
ip address: 192.168.1.2  
netmask: 255.255.255.0  
gateway: 192.168.1.1 (commented with #)
```

At your computer console (not the robot console), execute the following for loading the drivers and setting the network connection:

```
sudo modprobe usbnet
```

```
sudo modprobe cdc_ether
```

```
sudo ifconfig usb0 192.168.1.3 netmask 255.255.255.0
```

You can transfer file with the *scp* command (see chapter 5.1.3). You can have remote access to the KJ-EVO with chapter 5.1.5.

On Microsoft Windows, you may need to install a driver if the version is lower than Windows 7 (for Windows 7: <http://developer.toradex.com/knowledge-base/how-to-install-microsoft-rndis-driver-for-windows-7>).

The driver is located here:

<http://www.davehylands.com/linux/gumstix/usbnet/linux.inf>

and the installation instructions there:

[http://docwiki.gumstix.org/index.php/Windows\\_XP\\_usbnet](http://docwiki.gumstix.org/index.php/Windows_XP_usbnet)

Then you configure the ip address 192.168.1.3 with netmask 255.255.255.0 and gateway 192.168.1.1 for your new network connection on the computer.

### 5.1.5 Remote access

You can have remote access to the KJ-EVO board console while using Wifi or USB slave.

- 1) Establish a connection between the KJ-EVO and the computer with one of the connections Wifi or USB slave method described in the chapters above.
- 2) In a console of the computer, launch the ssh command, where KJ-EVO\_IP\_ADDRESS is the network address of your KJ-EVO board:

***ssh root@KJ-EVO\_IP\_ADDRESS***

- 3) Accept the *host authenticity* by answering yes in pushing Return key.

=> ***root@kj-evo:~#*** appears; you are logged in.

### 5.1.6 Using the camera module

The driver of the camera (*mt9v032.so*) is loaded by default on start-up.

#### 5.1.6.1 Taking images

With the *v4l2grab* program, you can take snapshots pictures with the camera:

***v4l2grab -o image.jpg -W 752 -H 480 -q 85***

where:

***image.jpg*** is the output image file in JPEG format

***752*** is the width of the image in pixel (max)

***480*** is the height of the image in pixel (max)

***85*** is the quality jpg compression in %

#### 5.1.6.2 Video streaming

There is a mjpeg streamer with web server installed by default on the KJ-EVO board. Firstly, set a network connection as explained in chapters 5.1.2 or 5.1.4.

Then launch the mjpeg streamer with the following command on the KJ-EVO (in one line command only:

***mjpg\_streamer -i "input\_uvc.so -yuv -f 15" -o "output\_http.so -w /usr/local/mjpg-streamer/www"***

On a computer connected to the same network, you can have access to the video streaming at [http://KJ-EVO\\_IP:8080/?action=stream](http://KJ-EVO_IP:8080/?action=stream), where *KJ-EVO\_IP* is the KJ-EVO network address.

You can stop with the ***CTRL-c*** keys.

You can configure the web server in the folder */usr/local/mjpg-streamer/www* . Some help is available in the file */usr/local/mjpg-streamer/www/README* .

### 5.1.6.3 Programming Image processing

In the chapter 4.5.2.1 "Application development", you will find how to modify and compile the examples source code. Select the following example for the camera:

`~/kj-evo_development/kj-evo_library/examples/camera_example.c`

**Driver source code:** you can access to the driver if you installed the full toolchain. The driver *bitbake* recipe is located in:

`/usr/local/kjunior-oetools-1.0/user.collection/recipes/mt9v032-module`

It is useful if you would like to change the *autoexposure*, *hdr*, *autogain* or *low\_light* parameters. You can test with the right combination of these parameters in unloading the driver and reloading it with these parameters in argument; example:

`rmmod mt9v032`

`modprobe mt9v032 hdr=1`

After compiling the module with *bitbake*, copy the file to the KJ-EVO and install it. You need to remove the kernel module before:

`opkg remove --force-depends kernel-module-mt9v032`

`opkg install mt9v032-module_1.0-r97.6_overo.ipk`

### 5.1.6.4 Changing colors levels (whitebalance)

You need the full toolchain (chapter 4.6) to change the colors levels.

For testing, modify on your computer in file:

`/usr/local/kjunior-oetools-1.0/tmp/work/overo-angstrom-linux-gnueabi/linux-omap3-2.6.34-r97/git/drivers/media/video/isp/isppreview.c`

the structure *ispprev\_rgbtorgb* containing the RGB blending, specially the parameters **RR**, **GG** and **BB** with the matrix equation linking input to output colors:

|  |   |
|--|---|
| <pre>static struct ispprev_rgbtorgb flr_rgb2rgb = {     {          // RGB-RGB Matrix gains         {RR, RG, RB },         {GR, GG, GB },         {BR, BG, BB}     },          // RGB Offset     {Rof, Gof, Bof} };</pre> <p style="text-align: center;">C code</p> | $\begin{pmatrix} R_{out} \\ G_{out} \\ B_{out} \end{pmatrix} = \begin{pmatrix} RR_f & RG_f & RB_f \\ GR_f & GG_f & GB_f \\ BR_f & BG_f & BB_f \end{pmatrix} \cdot \begin{pmatrix} R_{in} \\ G_{in} \\ B_{in} \end{pmatrix} + \begin{pmatrix} Rof_f \\ Gof_f \\ Bof_f \end{pmatrix}$ <p style="text-align: center;">Equation</p> |
|--|---|

These parameters are integer values. They are coded in fixed-point numbers: S12Q8 for the 9 gains and S10Q0 for the 3 offsets.

For converting the gains for the array, multiply by 256 and round the value. For the offsets, just round the value:

Example:

$RR_f = 0.8$   $GG_f = 1.0$   $BB_f = 1.5$   $Rof_f = 10.5$

=>

$RR = 205$   $GG = 256$   $BB = 384$   $Rof = 11$

By default, all the parameters are zeros except  $RR = 256$ ,  $GG = 314$  and  $BB = 498$ , which is for an incandescent light source.

Then recompile the kernel:

- source the *env.sh* file from your development folder
- install the needed program *uboot-mkimage* with : *sudo apt-get install boot-mkimage*
- go to folder  
*/usr/local/kjunior-oetools-1.0/tmp/work/overo-angstrom-linux-gnueabi/linux-omap3-2.6.34-r97/git/*
- compile the kernel with the command  
*make clean*  
*make -j8 ARCH=arm CROSS\_COMPILE=arm-angstrom-linux-gnueabi- uImage*
- in */usr/local/kjunior-oetools-1.0/tmp/work/overo-angstrom-linux-gnueabi/linux-omap3-2.6.34-r97/git/arch/arm/boot/* you will have the kernel file *uImage* . Use this file for the point 2) of chapter 4.6.3.3.

For future recompilations of the kernel and creations of the filesystem:

When you have found the right parameters, insert them in the existing patch file

*/usr/local/kjunior-oetools-1.0/user.collection/recipes/linux/linux-omap3-2.6.34/overo/isppreview.patch*

and rebuild the kernel and filesystem as explained in chapter 4.6.3.1.

### 5.1.7 nfs configuration

The first service to set up should be transparent file sharing using NFS. Most Linux distributions include NFS support by default, and the KJ-EVO system is ready to be connected. The directory to be shared between the computer and the board must be declared to the NFS service in the */etc/exports* configuration file. Please refer to NFS documentation, or man exports, for a detailed syntax description. Basically, the following line should be added to the file */etc/exports* on the computer:

*/mnt/nfsarm KJ-EVO\_IP/255.255.255.0(rw,no\_root\_squash,sync)*

And make this folder on the computer:

*mkdir /mnt/nfsarm*

The next step is to mount the shared directory to the KJ-EVO file system. Mounting a local hard drive partition or a network directory is exactly the same from the user point of view, the mount commands should be on the KJ-EVO, where `COMPUTER_IP`, is the IP address of the computer which the KJ-EVO will be connected:

```
mkdir /mnt/nfs
```

```
mount -t nfs -o nolock COMPUTER_IP:/mnt/nfsarm /mnt/nfs
```

If the NFS service is not started on the PC, the mount command will issue the following message:

```
mount: RPC: Unable to receive; errno = Connection refused
```

```
NFS: mount program did not respond!
```

```
mount: nfsmount failed: Bad file descriptor
```

The NFS service is usually started using a startup script for which location and name depend on you distribution (for example `/etc/init.d/nfs`).

Documentation for the distribution should detail the method to start and stop services.

Caution: For the nfs service to work properly, the portmap service should be started as well and if a firewall is active on the host machine, it should be configured to allow the nfs port access from the KJ-EVO.

Once the directory is successfully mounted, it can be accessed from the board exactly as if it was a local directory. Files on the PC can be read or written, new files can be created, and programs can be executed, as long as they are ARM executables. If required, the shared directory can be unmounted using the command:

```
umount myMountPoint
```

### **5.1.8 Using microphones and speakers**

#### **Playing:**

You must switch ON the speakers with the commands:

```
echo 64 >/sys/class/gpio/export
```

```
echo out >/sys/class/gpio/gpio64/direction
```

```
echo 1 >/sys/class/gpio/gpio64/value
```

You can play PCM WAV files with ***aplay*** and the syntax:

```
aplay FILE.wav
```

and MPEG audio files with ***madplay*** and the syntax:

```
madplay FILENAME -o cdda:- / aplay -f cdr
```

You can then switch the speakers OFF with:

```
echo 0 >/sys/class/gpio/gpio64/value
```

### **Recording:**

You can record PCM WAV with *arecord* and the syntax:

```
arecord -f cd FILENAME.wav
```

The *-f cd* parameter sets the recording in stereo mode in CD quality. See with the command *arecord -h* for all the parameters and help.



**Controlling volume:**

You can modify the playback volume by launching the volume mixer with the command:

*alsamixer*

You change channel with the keyboard left and right arrows and volume with the up/down arrows. The channels are for speakers of the KJU-EVO:

*"Dac2 Analog"* for standard control

*"Dac2 Digital Coarse"* for rough control

*"Dac2 Fine Coarse"* for fine control

You can change left and right with while a channel above is selected with:

*Q and Z for left*

*W and X for both*

*E and C for right*

And mute:

*left with < key*

*right with > key*

Pushing *h* will get you more help.

For modifying the record volume, push the Tab key while in *alsamixer*. You change channel with the keyboard left and right arrows and volume with the up/down arrows. The channels are for the KJU-EVO:

*"Analog Left Main Mic"* for mute settings of left micro

*"Analog Right Sub Mic"* for mute settings of right micro

By default the record channels are muted. While a channel is selected, pushing spacebar will toggle it.

You can change left and right while the *"Analog"* is selected with:

*Q and Z for left*

*W and X for both*

*E and C for right*

**Programming playback and record:**

In the chapter 4.5.2.1 "Application development", you will find how to modify and compile the examples source code. Select the following example:

*~/kj-evo\_development/kj-evo\_library/examples/sound\_example.c*

### 5.1.9 Development with a virtual machine

You can install the development tools on a virtual machine if you don't have access to a Linux machine. This is

You can even find a virtual machine image already configured with the development tools there:

[http://ftp.k-team.com/K-JuniorV2/extensions/kj-evo/virtual\\_machine](http://ftp.k-team.com/K-JuniorV2/extensions/kj-evo/virtual_machine)

- for the light tools (~2.1 GB): `Ubuntu_LTS_KJ-EVO_light_tools.ova`
- for the full tools (~15 GB): `Ubuntu_LTS_KJ-EVO_full_tools.ova` (available on request)

- 1) Download and install VirtualBox, version 4.1.18 or newer for your computer from:

<https://www.virtualbox.org/wiki/Downloads>

If you choose a more recent version, you will have to reinstall the Guest Additions; see VirtualBox user's manual.

- 2) Import on of the image file above with "File" menu, "Import Appliance" in the VirtualBox Manager.

This will take some time and hard disk space (~10 GB for the light tools and ~40 GB for the full tools).

- 3) Create the directory `C:\virtual_machine_shared` on your computer. This will be the shared directory for transferring data between the virtual machine and your host computer. It corresponds to the directory `/media/sf_virtual_machine_shared` of the virtual machine.

- 4) Start the virtual machine with your imported image:

Its login is:

username: user

password: root2011

- 5) The development tools are already installed. The development folder `kj-evo_development` is in `/home/user/kj-evo_development`.
- 6) You may need to update the toolchain. Remove the older one and follow instructions in chapter 4.5.1 or 4.6.2 depending if this is the light or full toolchain.

**Stopping the virtual machine:**

When going to the menu "Machine" and "Close, you will have 3 choices:

- "Save the machine state": saves the current state, which will be reloaded next time you restart it.
- "Send the shutdown signal", or press the shutdown button into the machine: power off the machine and save only the work done.
- "Power off the machine": WARNING: it doesn't save anything; any work done will be discarded!

**Serial port sharing:**

You may configure your virtual machine to share the serial port to connect to the KJ-EVO (for using Bluetooth ...).

- Power off your virtual machine.
- Select your machine image and press "Settings" button.
- Go to "Serial Ports" settings;
- On the tab "Port 1", enable the checkbox "Enable Serial Port" and choose:
  - Port Number: COM1 (=> this will be the Linux serial port /dev/ttyS0)
  - Port Mode: Host Device
  - Port/File Path: COM1 if you have a standard serial port and using COM1.  
Or change to adapt to your computer serial port or Bluetooth emulated serial port.

**5.1.10 Debugging**

You can debug a program you made directly in console on the KJ-EVO or remotely. You will find on the DVD or online the needed files:

<http://ftp.k-team.com/K-JuniorV2/extensions/KJ-EVO/software/debugger>

**5.1.10.1 Debugging on the KJ-EVO**

- You need firstly to install the following packages with the command ***opkg install PACKAGE.IPK***:
  - libthread-db1\_2.9-r37.4.6\_armv7a.ipk***
  - gdb\_7.1-r8.4.6\_armv7a.ipk***
- Then your program must be cross-compiled with the option ***-g*** . Edit your Makefile and replace the flag ***-O2*** by ***-g*** .
- Copy to the KJ-EVO the compiled program and its source file. And run for debugging it:  
***gdb YOUR\_PROGRAM***

The basic commands are:

|                               |   |
|-------------------------------|---|
| <b><i>r</i></b>               | : run   |
| <b><i>n</i></b>               | : next: one step in the program; enter in subroutines |
| <b><i>s</i></b>               | : one step in the program                             |
| <b><i>b</i></b> line/function | : break at line/function                              |
| <b><i>delete</i></b> break    | : delete breakpoint number                            |
| <b><i>until</i></b> line      | : continue until line                                 |
| <b><i>c</i></b>               | : continue  |
| <b><i>l</i></b>               | : list code   |
| <b><i>q</i></b>               | : quit gdb  |
| <b><i>h</i></b>               | : help; you can have all the commands here            |

A common sequence of debugging can be:

- to list the code with ***l***
- set a breakpoint at the beginning of main with ***b main***
- run the program with ***r***
- execute a step with ***n***
- display a variable with ***p VAR\_NAME***
- execute next step with ***n***
- continue to the end with ***c***
- Quit with ***q***.

#### 5.1.10.2 Remote debugging

You need firstly to install the following package with the command ***opkg install PACKAGE.IPK***:

***gdbserver\_7.1-r8.0.6\_armv7a.ipk***

- Then run the gdbserver with your cross-compiled program and an unused port number as argument (here ***1234***):

***gdbserver --multi :1234 YOUR\_PROGRAM***

- Source your env.sh (see chapter 4.5.2.1) and run:

***source ~/kj-evo\_development/env.sh***

- Go to the folder where you cross-compiled your program

- You can use the debugger in command line or with **ddd** GUI:

#### Command line:

- Execute for running the debugger on the computer:  
***arm-angstrom-linux-gnueabi-gdb YOUR\_PROGRAM***
- In this debugger, set parameter and connect to the remote gdb with these 2 commands:  
***set sysroot /usr/local/kjunior-oetools-1.0/tmp/sysroots/armv7a-angstrom-linux-gnueabi***  
***target extended-remote YOUR\_KJU\_IP\_ADDRESS:1234***

#### With GUI:

- On your computer, install ddd the debugger GUI with :  
***sudo apt-get install ddd***
- Launch ddd with:  
***ddd --debugger arm-angstrom-linux-gnueabi-gdb YOUR\_PROGRAM***
- In the window at the bottom where the (gdb) prompt is, run these 2 commands:  
***set sysroot /usr/local/kjunior-oetools-1.0/tmp/sysroots/armv7a-angstrom-linux-gnueabi***  
***target extended-remote YOUR\_KJU\_IP\_ADDRESS:1234***

The sequence of debugging and also the commands are the same as described above in chapter 5.1.10.1.

You can even send your program to the remote (KJ\_evo) with the gdb commands:

***remote put hostfile targetfile***  
***set remote exec-file targetfile***

where *hostfile* and *targetfile* is the same name of your new program to be debugged.

For quitting the debugger and remote, execute commands ***monitor exit***, then ***disconnect*** and finally ***quit*** .

#### Tips for using the advantage of the GUI:

- For setting a breakpoint, double-click on the line you want to put it, just before the line number or right click at the same place or choose “Set Breakpoint”.
- With the floating menu, you have the different commands for debugging.
- You can add a watch on a variable with clicking with the right mouse button on the variable and choose “Display VARIABLE\_NAME”.

### 5.1.10.3 Remote debugging with Eclipse

You can also debug remotely your program with Eclipse (see chapter 4.5.2.2 for a proper installation). Follow instructions below:

- 1) Install ***gdbserver*** on your kj-evo as explained in chapter 5.1.10.2 above.
- 2) On Eclipse menu 'Run' => 'Debug Configurations', double click on 'C++ Remote Application'  
=> test Debug is created.
- 3) In the 'Main' tab => "C/C++ application", enter: Debug/test
- 4) On "Connection": push "New" button and choose "SSH", in the next window, ***Hostname***: ip address of your kj-evo and push "Finish" button.
- 5) On "Remote Absolute File Path for C/C++ Application" modify to ***/home/root/test***
- 6) In the Debugger tab, choose  
***/usr/local/kjunior-oetools-1.0/tmp/sysroots/i686-linux/usr/armv7a/bin/arm-angstrom-linux-gnueabi-gdb***
- 7) On 'Main' tab change 'GDB Debugger' to ***/usr/local/kjunior-oetools-1.0/tmp/sysroots/i686-linux/usr/armv7a/bin/arm-angstrom-linux-gnueabi-gdb***  
On 'Shared Libraries' tab add ***/usr/local/kjunior-oetools-1.0/tmp/sysroots/armv7a-angstrom-linux-gnueabi/lib***
- 8) Then Push Debug, enter ***root*** as "User ID" and no password.
- 9) Debug step by step.

You can go again into the settings, and select "Release" configuration when your program has been fully debugged.

You can run the program from the kj-evo (with a remote terminal) or with Eclipse GUI. After having set the Debug configuration, the parameters are also available for the run.

Just go the menu Run, then choose "Run".

In the lower part of the Eclipse window, you will see the output of your program in the Console tab.

## 6. WARRANTY

---

K-TEAM warrants that the KJ-EVO is free from defects in materials and workmanship and in conformity with the respective specifications of the product for the minimal legal duration, respectively one year from the date of delivery.

Upon discovery of a defect in materials, workmanship or failure to meet the specifications in the Product during the afore mentioned period, Customer must request help on K-Team Internet forum on <http://www.k-team.com/forum/> by detailing:

- the type of KJ-EVO used (version)
- the kernel version of the KJ-EVO
- the programming environment of the KJ-EVO/robot (standard, version, OS)
- the standard use of Product before the appearance of the problem
- the description of the problem.

If no answers have been received within two working days, Customer can contact K-TEAM support by phone or by electronic mail with the full reference of its order and KJ-EVO serial number.

K-TEAM shall then, at K-TEAM's sole discretion, either repair such Product or replace it with the equivalent product without charging any technical labor fee and repair parts cost to Customer, on the condition that Customer brings such Product to K-TEAM within the period mentioned before. In case of repair or replacement, K-TEAM may own all the parts removed from the defective Product. K-TEAM may use new and/or reconditioned parts made by various manufacturers in performing warranty repairs and replacement of the Product. Even if K-TEAM repairs or replaces the Product, its original warranty term is not extended.

This limited warranty is invalid if the factory-applied serial number has been altered or removed from the Product.

This limited warranty covers only the hardware and software components contained in the Product. It does not cover technical assistance for hardware or software usage and it does not cover any software products contained in the Product. K-TEAM excludes all warranties expressed or implied in respect of any additional software provided with Product and any such software is provided "AS IS" unless expressly provided for in any enclosed software limited warranty. Please refer to the End User License Agreements included with the Product for your rights with regard to the licensor or supplier of the software parts of the Product and the parties' respective obligations with respect to the software.

This limited warranty is non-transferable.

It is likely that the contents of Customer's flash memory will be lost or reformatted in the course of the service and K-TEAM will not be responsible for any damage to or loss of any programs, data or other information stored on any media or any part of the Product serviced hereunder or damage or loss arising from the Product not being available for use before, during or after the period of service provided or any indirect or consequential damages resulting therefore.

IF DURING THE REPAIR OF THE PRODUCT THE CONTENTS OF THE FLASH MEMORY ARE ALTERED, DELETED, OR IN ANY WAY MODIFIED, K-TEAM IS NOT RESPONSIBLE WHATEVER. CUSTOMER'S PRODUCT WILL BE RETURNED TO CUSTOMER CONFIGURED AS ORIGINALLY PURCHASED (SUBJECT TO AVAILABILITY OF SOFTWARE).

Be sure to remove all third parties' hardware, software, features, parts, options, alterations, and attachments not warranted by K-TEAM prior to Product service. K-TEAM is not responsible for any loss or damage to these items.

This warranty is limited as set out herein and does not cover, any consumable items (such as batteries) supplied with the Product; any accessory products which is not contained in the Product; cosmetic damages; damage or loss to any software programs, data, or removable storage media; or damage due to (1) acts of God, accident, misuse, abuse, negligence, commercial use or modifications of the Product; (2) improper operation or maintenance of the Product; (3) connection to improper voltage supply; or (4) attempted repair by any party other than a K-TEAM authorized module service facility.

This limited warranty does not apply when the malfunction results from the use of the Product in conjunction with any accessories, products or ancillary or peripheral equipment, or where it is determined by K-Team that there is no fault with the Product itself.



K-TEAM EXPRESSLY DISCLAIMS ALL OTHER WARRANTIES THAN STATED HEREINBEFORE, EXPRESSED OR IMPLIED, INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE TO THE FULLEST EXTENT PERMITTED BY LAW.

Limitation of Liability: IN NO EVENT SHALL EITHER PARTY BE LIABLE TO THE OTHER FOR ANY INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM PERFORMANCE OR FAILURE TO PERFORM UNDER THE CONTRACT, OR FROM THE FURNISHING, PERFORMANCE OR USE OF ANY GOODS OR SERVICE SOLD OR PROVIDED PURSUANT HERETO, WHETHER DUE TO A BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, OR OTHERWISE. SAVE THAT NOTHING HEREIN SHALL LIMIT EITHER PARTY'S LIABILITY FOR DEATH OR PERSONAL INJURY ARISING FROM ITS NEGLIGENCE, NEITHER PARTY SHALL HAVE ANY LIABILITY TO THE OTHER FOR INDIRECT OR PUNITIVE DAMAGES OR FOR ANY CLAIM BY ANY THIRD PARTY EXCEPT AS EXPRESSLY PROVIDED HEREIN.







K-Team S.A.  
Z.I Plans-Praz  
1337 Vallorbe  
Switzerland

---