URBI Server for Aibo ERS2xx ERS7 Introduction Manual

v 1.0

Jean-Christophe Baillie

January 2005

Contents

Introduction

URBI (Universal Robotic Body Interface) is a scripted language designed to work over a client/server architecture in order to remotely control a robot or, in a broader definition, any kind of device that has actuators and sensors. The main characteristics of URBI, which make it different from other existing solutions are:

- URBI is a low level command language. Motors and sensors are directly read and set. Although complex high level commands and functions can be written with URBI, the raw kernel of the system is low level by essence.
- URBI includes powerful time oriented control mechanisms to chain commands, serialize them or build complex motor trajectories.
- URBI is designed to be independant from both the robot and the client system. It relies on TCP/IP or Inter-Process Communication if the client and the server are both running onboard.
- URBI is designed with a constant care for simplicity. There is no "philosophy" or "complex architecture" to be familiar with. It is understandable in a few minutes and can be used immediately.

URBI Server is released under the GNU General Public License (LICENSE) URBI Language is released under a specific License included in this distribution (LICENSE-URBI-LANGUAGE).

Chapter 1

Installation

You don't have to recompile the server to use it on your Aibo robot. Simply use the MS directory in the 'server-aibo' directory where there is a pre-compiled version available.

1.1 Installing the MS directory

- Prepare your memory stick with the system files for WCONSOLE/memprot/ERSxxx. It should be located in:

  `/usr/local/OPEN_R_SDK/OPEN_R/MS_ERSxxx/WCONSOLE/memprot`

- copy the OPEN-R directory that you find there in the root of the empty memorystick.
- Then, go to the `OPEN-R/SYSTEM/CONF` on the memorystick and edit (or create) the `WLANCONF.TXT` file to suit your network configuration (see the `WLANDFLT.TXT` for a default setting that you might update).
- Untar the `urbiserver.tar.gz` file that you downloaded and do a `cd urbiserver` to move to the main directory, called the 'root' directory in the following.
- Then, `cd` to the `server-aibo` directory in the 'root' and go to the `MS` directory. Copy (and overwrite if necessary) the content of this MS directory in the root of the memorystick.
- Unmount the memorystick and put it in the robot. It should work. You can telnet the robot on port 54000 to check if everything is OK (you should get a URBI Header at start).

1.2 Recompile the server

If you plan to rebuild the server, simply go into the `server-aibo` directory in the 'root' directory and type `make install`. This will update the content of the MS directory with the newly compiled server called 'URBI.BIN'. Then, simply proceed through the installation step above to update your server.

Note: there is a kernel part in the 'root' directory. It contains the kernel of the URBI server. You should not modify this part of the code since it might be upgraded later simply by overwriting this directory with the new kernel. If you want to make changes to the kernel or correct bugs, the best is to contact the person responsible of it on sourceforge and see if a new release can be done with your modifications.

Chapter 2

Quick start

The best way to start with URBI is to open a telnet client on the robot, port 54000. You should get something like this:

```
[00139464:notag]
[00139464:start] ************************************************************
[00139464:start]  URBI Language specif xxx   -   Copyright (C) 2005 JC Baillie
[00139464:start]  URBI Kernel version yyy
[00139464:start]
[00139464:start]    URBI Server version zzz for Aibo ERS2xx/ERS7 Robots
[00139464:start]    (C) 2004 Jean-Christophe Baillie
[00139464:start]
[00139464:start]  URBI comes with ABSOLUTELY NO WARRANTY;
[00139464:start]  This is free software, and you are welcome to redistribute
[00139464:start]  it under certain conditions; see GNU GPL for details.
[00139464:start]
[00139464:start]  See http://urbi.sourceforge.net for news and updates.
[00139464:start] ************************************************************
[00139464:ident] ID: U597766392
```

You can start to enter URBI commands.

2.1 First contact

The command you might want to try is `motoron` to activate your robot motors.
Let's move the head by typing the following command in the telnet client:

```
headPan.val = 15;
headTilt.val = 20;
```

Switch a few LEDs on (these LEDs exist only on ERS7):

```
ledF12.val = 1;
ledF13.val = 0.8;
ledF1.val = 1;
```

What was the head pan value already? Ask for it:

```
headPan.val;
```

It says:

```
[136901543:notag] 15.1030265089
```

Let's play a sound (the wav file is supposed to be on the memorystick root)

```
speaker.play ("welcome.wav");
```

Let's enter a program that will do the ball tracking head:

```
whenever (camera.ballx != -1) {
  headPan.val  = headPan.val  + camera.xfov * ( 0.5 - camera.ballx ) &
  headTilt.val = headTilt.val + camera.yfov * ( 0.5 - camera.bally )
};
```

2.2 Review of some features

The URBI server receives *commands* from a client and returns *messages* to this client. The normal way of using a URBI controlled robot is to send commands using TCP/IP on the URBI port (54000) and wait for messages in return. A simple telnet client is enough to do that for simple applications, otherwise libraries (liburbi) are available in most programming languages to wrap the TCP/IP sending/receiving job in simple functions.

Any URBI command can be prefixed by a *tag* followed by a colon. If no tag is specified, `notag` is assumed to be the default tag.

Examples:

```
x = 12; // notag
my_tag: y = x+12; // the tag is 'my_tag'
```

When a command returns a value or when it fails, the URBI server returns a message. The format of a message is the following:

```
[timestamp:tag] message
```

The time stamp is the uptime of the server in milliseconds when the message has been sent. The tag is the tag of the associated command or `notag` if there was no tag specified. The message can be a value (float, string or binary) or a system message prefixed by three stars: `***`. Error messages or information messages are system messages.

Here is a typical example of commands with their messages in return:

```
1+1;
[136901543:notag] 2.0000000000

my_tag:6*6;
[136904711:my_tag] 36.0000000000

impossible:1/0;
[136471768:impossible] *** Division by zero
[136471768:impossible] *** EXPR evaluation failed
```

The ability to tag commands is a key feature of URBI since it allows the client to retrieve the results of specific commands in a flow of messages.

Every sensor, motor or controllable element of the robot is a *device*. It has a device name and can be entirely accessed using this name as prefix. Every device is associated with a set of *fields* and *methods* that can be accessed using the syntax `device.field` or `device.method(...)`. Fields are device specific variables and method are device specific functions. The most useful field to start with is `val` which gives access to a motor angular value, a led illumination or a sensor output.

To know what you can do with your robot, simply check the list of available devices (see at the end) and their fields and methods.

This is already enough to control your robot. But URBI is more than that. You can give complex motor commands with complex trajectories, you can set several commands in series or in parallel, you can do some usual C programming using while, for, if. There are also some advanced features for event catching (at, whenever).

These features are described in extension in the URBI Language Specification (located in the `doc` directory, at the root of the `urbiserver` directory).

Chapter 3

Liburbi

Using URBI with a telnet is too limited. You need to be able to send commands and receive messages based on tag filter, using a programming language of your choice. This is what liburbi is made for. There is currently a C++ and java version of liburbi if you want to control your robot using C++ or Java and a liburbi-OPENR version if you want to recompile a liburbi-C++ based program to let it run on the robot. See the liburbi documentation for more details.

Chapter 4

Devices List

You can access the list of devices with the `devices` command. Any device can be queried in URBI using the `info` operator:

```
info headPan;
[04297385:notag] *** description: Head pan
[04297385:notag] *** device: headPan
[04297385:notag] *** current value: 28.486434
[04297385:notag] *** current load: 1.000000
[04297385:notag] *** rangemin: -91.000000
[04297385:notag] *** rangemax: 91.000000
[04297385:notag] *** unit: deg
```

The unit is accessible with the `unit` operator, the min range and max range with the `rangemin` and `rangemax` operators.

The variable `global.nbdevices` contains the number of devices. The variables `device[i]` are the device names.

4.1 ERS-7

4.1.1 Motors

The following devices are joints, with the corresponding range indicated:

```
legRF1          range=[-134.000000,120.000000]   unit=deg : Right fore legJ1
legRF2          range=[-9.000000,91.000000]      unit=deg : Right fore legJ2
legRF3          range=[-29.000000,119.000000]    unit=deg : Right fore legJ3
legRH1          range=[-134.000000,120.000000]   unit=deg : Right hind legJ1
legRH2          range=[-9.000000,91.000000]      unit=deg : Right hind legJ2
legRH3          range=[-29.000000,119.000000]    unit=deg : Right hind legJ3
legLF1          range=[-120.000000,134.000000]   unit=deg : Left fore legJ1
legLF2          range=[-9.000000,91.000000]      unit=deg : Left fore legJ2
legLF3          range=[-29.000000,119.000000]    unit=deg : Left fore legJ3
legLH1          range=[-120.000000,134.000000]   unit=deg : Left hind legJ1
legLH2          range=[-9.000000,91.000000]      unit=deg : Left hind legJ2
legLH3          range=[-29.000000,119.000000]    unit=deg : Left hind legJ3
neck            range=[-79.000000,2.000000]      unit=deg : Neck tilt1
headTilt        range=[-16.000000,44.000000]     unit=deg : Neck tilt2
headPan         range=[-91.000000,91.000000]     unit=deg : Head pan
tailPan         range=[-59.000000,59.000000]     unit=deg : Tail pan
tailTilt        range=[2.000000,63.000000]       unit=deg : Tail tilt
mouth           range=[-58.000000,-3.000000]     unit=deg : Mouth
```

For all joints, you have the following fields:

- `val` : the value of the joint
- `valn` : the normalized value of the joint between 0 and 1
- `load` : the load of the joint. 0 means “loose”, and 1 means “blocked”. Values in between give intermediary results.
- `PGain` : the P gain of the joint
- `IGain` : the I gain of the joint
- `DGain` : the D gain of the joint
- `PShift` : the P shift of the joint
- `IShift` : the I shift of the joint
- `DShift` : the D shift of the joint

4.1.2 Leds, switches, ...

```
ledF1           range=[0.000000,1.000000] : Face light1
ledF2           range=[0.000000,1.000000] : Face light2
ledF3           range=[0.000000,1.000000] : Face light3
ledF4           range=[0.000000,1.000000] : Face light4
ledF5           range=[0.000000,1.000000] : Face light5
ledF6           range=[0.000000,1.000000] : Face light6
```

```
ledF7          range=[0.000000,1.000000] : Face light7
ledF8          range=[0.000000,1.000000] : Face light8
ledF9          range=[0.000000,1.000000] : Face light9
ledF10         range=[0.000000,1.000000] : Face light10
ledF11         range=[0.000000,1.000000] : Face light11
ledF12         range=[0.000000,1.000000] : Face light12
ledF13         range=[0.000000,1.000000] : Face light13
ledF14         range=[0.000000,1.000000] : Face light14
ledBFC         range=[0.000000,1.000000] : Back light(front,color)
ledBFW         range=[0.000000,1.000000] : Back light(front,white)
ledBMC         range=[0.000000,1.000000] : Back light(middle,color)
ledBMW         range=[0.000000,1.000000] : Back light(middle,white)
ledBRC         range=[0.000000,1.000000] : Back light(rear,color)
ledBRW         range=[0.000000,1.000000] : Back light(rear,white)
ledHC          range=[0.000000,1.000000] : Head light(color)
ledHW          range=[0.000000,1.000000] : Head light(white)
modeB          range=[0.000000,1.000000] : Mode Indicator(blue)
modeG          range=[0.000000,1.000000] : Mode Indicator(green)
modeR          range=[0.000000,1.000000] : Mode Indicator(red)
earL           range=[0.000000,1.000000] : Left ear
earR           range=[0.000000,1.000000] : Right ear
WIFIswitch     range=[0.000000,1.000000] : Wireless LAN switch
ledWIFI        range=[0.000000,1.000000] : Wireless light
```

For all these devices, you have the following fields:

- `val` : the value of the device
- `valn` : the normalized value of the device

There is a global "led mode" available on ERS7. You can change it via the global variable `robot.ledMode` (default value is 0). The lights that will be switched on (on the face), and the colors of those lights depends on the mode. Because of this, there is no real one-to-one correspondance between a device name and a physical led on the ERS-7 robot.

4.1.3 Sensors

```
pawLF          range=[0.000000,1.000000]     : Left fore leg,paw sensor
pawLH          range=[0.000000,1.000000]     : Left hind leg,paw sensor
pawRF          range=[0.000000,1.000000]     : Right fore leg,paw sensor
pawRH          range=[0.000000,1.000000]     : Right hind leg,paw sensor
accelX         range=[-19.613300,19.613300] : Acceleration sensor(front-back)
accelY         range=[-19.613300,19.613300] : Acceleration sensor(right-left)
accelZ         range=[-19.613300,19.613300] : Acceleration sensor(up-down)
chinsensor     range=[0.000000,1.000000]     : Chin sensor
backsensorF    range=[0.000000,60.000000]    : Back sensor(front)
backsensorM    range=[0.000000,60.000000]    : Back sensor(middle)
backsensorR    range=[0.000000,60.000000]    : Back sensor(rear)
headsensor     range=[0.000000,35.000000]    : Head sensor
distanceChest range=[19.000000,90.000000]  : Chest distance sensor
distanceNear   range=[5.700000,50.000000]    : Head distance sensor(near)
distanceFar    range=[20.000000,150.000000] : Head distance sensor(far)
```

For all these devices, you have the following fields:

- `val` : the value of the device
- `valn` : the normalized value of the device

4.1.4 Camera

The camera device on Aibo is called `camera`. The available fields are:

- `val` : the image (binary)
- `shutter` : the camera shutter speed: 1=SLOW (default), 2=MID, 3=FAST
- `gain` : the camera gain: 1=LOW, 2=MID, 3=HIGH (default)
- `wb` : the camera white balance: 1=INDOOR (default), 2=OUTDOOR, 3=FLUO
- `format` : the camera image format: 0=YCbCr 1=jpeg (default)
- `jpegfactor` : the jpeg compression factor (0 to 100). Default=80
- `resolution` : the image resolution: 0:208x160 (default) 1:104x80 2:52x40
- `reconstruct` : reconstruction of the high resolution image(slow): 0:no (default) 1:yes
- `width` : image width
- `height` : image height
- `xfov` : camera x FOV (degrees)
- `yfov` : camera y FOV (degrees)
- `ballx` : normalized X position of the red ball in the image (between 0 and 1) or -1 when there is no ball visible.
- `bally` : normalized Y position of the red ball in the image (between 0 and 1) or -1 when there is no ball visible.

4.1.5 Speaker

The speaker device on Aibo is called `speaker`. The available fields are:

- `val` : the sound to play (binary)
- `playing` : equal 1 when there is a sound playing, 0 otherwise
- `remain` : number of milliseconds of sound to play, 0 when the buffer is empty

There is also a method available:

- `play (file)` : plays the wav file $file$ which is stored on the root of the memorystick.

4.1.6 Micro

The micro device on Aibo is called `micro`. The available fields are:

- `val` : contains always a buffer of 2048 bytes of the latest sound heard by the robot.

4.2 ERS-210

4.2.1 Motors

The following devices are joints, with the corresponding range indicated:

```
legRF1        range=[-116.000000,116.000000]   unit=deg : Right fore legJ1
legRF2        range=[-9.000000,92.000000]      unit=deg : Right fore legJ2
legRF3        range=[-26.000000,146.000000]    unit=deg : Right fore legJ3
legRH1        range=[-116.000000,116.000000]   unit=deg : Right hind legJ1
legRH2        range=[-9.000000,92.000000]      unit=deg : Right hind legJ2
legRH3        range=[-26.000000,146.000000]    unit=deg : Right hind legJ3
legLF1        range=[-116.000000,116.000000]   unit=deg : Left fore legJ1
legLF2        range=[-9.000000,92.000000]      unit=deg : Left fore legJ2
legLF3        range=[-26.000000,146.000000]    unit=deg : Left fore legJ3
legLH1        range=[-116.000000,116.000000]   unit=deg : Left hind legJ1
legLH2        range=[-9.000000,92.000000]      unit=deg : Left hind legJ2
legLH3        range=[-26.000000,146.000000]    unit=deg : Left hind legJ3
tailPan       range=[-21.000000,21.000000]     unit=deg : Tail tilt
tailTilt      range=[-21.000000,21.000000]     unit=deg : Tail pan
headTilt      range=[-87.000000,41.000000]     unit=deg : Head tilt
headPan       range=[-89.000000,89.000000]     unit=deg : Head pan
headRoll      range=[-27.000000,27.000000]     unit=deg : Head roll
mouth         range=[-46.000000,-3.000000]     unit=deg : Mouth
```

For all joints, you have the following fields:

- `val` : the value of the joint
- `valn` : the normalized value of the joint between 0 and 1
- `load` : the load of the joint. 0 means “loose”, and 1 means “blocked”. Values in between give intermediary results.
- `PGain` : the P gain of the joint
- `IGain` : the I gain of the joint
- `DGain` : the D gain of the joint
- `PShift` : the P shift of the joint
- `IShift` : the I shift of the joint
- `DShift` : the D shift of the joint

4.2.2 Leds, switches, ...

ERS210

```
ledEUL        range=[0.000000,1.000000]        unit=bool : Eye light(Upper left)
mode          range=[0.000000,1.000000]        unit=bool : Mode indicator
ledEUR        range=[0.000000,1.000000]        unit=bool : Eye light(Upper right)
earL          range=[0.000000,1.000000]        unit=bool : Left ear
earR          range=[0.000000,1.000000]        unit=bool : Right ear
ledTB         range=[0.000000,1.000000]        unit=bool : Tail light(Blue)
```

```
ledTO          range=[0.000000,1.000000]        unit=bool : Tail light(Orange)
ledELL         range=[0.000000,1.000000]        unit=bool : Eye light(Lower left)
ledEML         range=[0.000000,1.000000]        unit=bool : Eye light(Middle left)
ledELR         range=[0.000000,1.000000]        unit=bool : Eye light(Lower right)
ledEMR         range=[0.000000,1.000000]        unit=bool : Eye light(Middle right)
```

For all these devices, you have the following fields:

- `val` : the value of the device
- `valn` : the normalized value of the device

There is a global "led mode" available on ERS7. You can change it via the global variable `robot.ledMode`.

4.2.3 Sensors

```
pawLF          range=[0.000000,1.000000]        unit=bool : Left fore leg,paw sensor
pawLH          range=[0.000000,1.000000]        unit=bool : Left hind leg,paw sensor
pawRF          range=[0.000000,1.000000]        unit=bool : Right fore leg,paw sensor
pawRH          range=[0.000000,1.000000]        unit=bool : Right hind leg,paw sensor
backSensor     range=[0.000000,1.000000]        unit=bool : Back sensor
chinSensor     range=[0.000000,1.000000]        unit=bool : Chin sensor
headSensorB    range=[0.000000,98.000000]       unit=uPa  : Head sensor(back)
headSensorF    range=[0.000000,98.000000]       unit=uPa  : Head sensor(front)
psd            range=[0.000000,90.000000]       unit=m    : Position Sensing Device
thermoSensor   range=[0.000000,60.000000]       unit=C    : Thermo sensor
accelX         range=[-19.613300,19.613300]     unit=m/s2 : Acceleration sensor(right-l
accelY         range=[-19.613300,19.613300]     unit=m/s2 : Acceleration sensor(front-b
accelZ         range=[-19.613300,19.613300]     unit=m/s2 : Acceleration sensor(up-down
```

For all these devices, you have the following fields:

- `val` : the value of the device
- `valn` : the normalized value of the device

4.2.4 Camera

The camera device on Aibo is called `camera`. The available fields are:

- `val` : the image (binary)
- `shutter` : the camera shutter speed: 1=SLOW (default), 2=MID, 3=FAST
- `gain` : the camera gain: 1=LOW, 2=MID, 3=HIGH (default)
- `wb` : the camera white balance: 1=INDOOR (default), 2=OUTDOOR, 3=FLUO
- `format` : the camera image format: 0=YCbCr 1=jpeg (default)
- `jpegfactor` : the jpeg compression factor (0 to 100). Default=80
- `resolution` : the image resolution: 0:208x160 (default) 1:104x80 2:52x40
- `reconstruct` : reconstruction of the high resolution image(slow): 0:no (default) 1:yes
- `width` : image width

- `height` : image height
- `xfov` : camera x FOV (degrees)
- `yfov` : camera y FOV (degrees)
- `ballx` : normalized X position of the red ball in the image (between 0 and 1) or -1 when there is no ball visible.
- `bally` : normalized Y position of the red ball in the image (between 0 and 1) or -1 when there is no ball visible.

4.2.5 Speaker

The speaker device on Aibo is called `speaker`. The available fields are:

- `val` : the sound to play (binary)
- `playing` : equal 1 when there is a sound playing, 0 otherwise
- `remain` : number of milliseconds of sound to play, 0 when the buffer is empty

There is also a method available:

- `play (file)` : plays the wav file $file$ which is stored on the root of the memorystick.

4.2.6 Micro

The micro device on Aibo is called `micro`. The available fields are:

- `val` : contains always a buffer of 2048 bytes of the latest sound heard by the robot.

4.3 ERS-220

4.3.1 Motors

The following devices are joints, with the corresponding range indicated:

```
legRF1        range=[-116.000000,116.000000]   unit=deg : Right fore legJ1
legRF2        range=[-10.000000,92.000000]     unit=deg : Right fore legJ2
legRF3        range=[-26.000000,146.000000]    unit=deg : Right fore legJ3
legRH1        range=[-116.000000,116.000000]   unit=deg : Right hind legJ1
legRH2        range=[-10.000000,92.000000]     unit=deg : Right hind legJ2
legRH3        range=[-26.000000,146.000000]    unit=deg : Right hind legJ3
legLF1        range=[-116.000000,116.000000]   unit=deg : Left fore legJ1
legLF2        range=[-10.000000,92.000000]     unit=deg : Left fore legJ2
legLF3        range=[-26.000000,146.000000]    unit=deg : Left fore legJ3
legLH1        range=[-116.000000,116.000000]   unit=deg : Left hind legJ1
legLH2        range=[-10.000000,92.000000]     unit=deg : Left hind legJ2
legLH3        range=[-26.000000,146.000000]    unit=deg : Left hind legJ3
headPan       range=[-89.000000,89.000000]     unit=deg : Head pan
headRoll      range=[-27.000000,27.000000]     unit=deg : Head roll
headTilt      range=[-87.000000,41.000000]     unit=deg : Head tilt
```

For all joints, you have the following fields:

- `val` : the value of the joint
- `valn` : the normalized value of the joint between 0 and 1
- `load` : the load of the joint. 0 means “loose”, and 1 means “blocked”. Values in between give intermediary results.
- `PGain` : the P gain of the joint
- `IGain` : the I gain of the joint
- `DGain` : the D gain of the joint
- `PShift` : the P shift of the joint
- `IShift` : the I shift of the joint
- `DShift` : the D shift of the joint

4.3.2 Leds, switches, ...

```
ledTailC      range=[0.000000,1.000000]        unit=bool : Tail light (Center)
ledBL         range=[0.000000,1.000000]        unit=bool : Head Face side light(Back l
ledTailL      range=[0.000000,1.000000]        unit=bool : Tail light (Left)
ledCL         range=[0.000000,1.000000]        unit=bool : Head Face side light(Center
ledBR         range=[0.000000,1.000000]        unit=bool : Head Face side light(Back r
ledTailR      range=[0.000000,1.000000]        unit=bool : Tail light (Right)
ledCR         range=[0.000000,1.000000]        unit=bool : Head Face side light(Center
ledFL         range=[0.000000,1.000000]        unit=bool : Head Face side light(Front
ledL1         range=[0.000000,1.000000]        unit=bool : Back multi indic (1st from
ledL2         range=[0.000000,1.000000]        unit=bool : Back multi indic (2nd from
ledRetractHead range=[0.000000,1.000000]       unit=bool : Retractable head light
```

```
ledL3          range=[0.000000,1.000000]        unit=bool : Back multi indic (3rd from
ledFR          range=[0.000000,1.000000]        unit=bool : Head Face side light(Front
ledHead        range=[0.000000,1.000000]        unit=bool : Head indicator
ledR1          range=[0.000000,1.000000]        unit=bool : Back multi indic (1st from
ledR2          range=[0.000000,1.000000]        unit=bool : Back multi indic (2nd from
ledR3          range=[0.000000,1.000000]        unit=bool : Back multi indic (3rd from
ledA           range=[0.000000,1.000000]        unit=bool : Face front light A
ledB           range=[0.000000,1.000000]        unit=bool : Face front light B
ledC           range=[0.000000,1.000000]        unit=bool : Face front light C
```

For all these devices, you have the following fields:

- `val` : the value of the device
- `valn` : the normalized value of the device

There is a global "led mode" available on ERS7. You can change it via the global variable `robot.ledMode`.

4.3.3 Sensors

```
pawLF          range=[0.000000,1.000000]        unit=bool : Left fore leg,paw sensor
pawLH          range=[0.000000,1.000000]        unit=bool : Left hind leg,paw sensor
pawRF          range=[0.000000,1.000000]        unit=bool : Right fore leg,paw sensor
pawRH          range=[0.000000,1.000000]        unit=bool : Right hind leg,paw sensor
thermoSensor   range=[0.000000,60.000000]       unit=uPa  : Thermo sensor
tailSensorC    range=[0.000000,1.000000]        unit=uPa  : Tail sensor (Center from bel
backSensor     range=[0.000000,99.000000]       unit=uPa  : Back sensor
tailSensorL    range=[0.000000,1.000000]        unit=uPa  : Tail sensor (Left from behi
tailSensorR    range=[0.000000,1.000000]        unit=uPa  : Tail sensor (Right from beh
headSensorB    range=[0.000000,35.000000]       unit=uPa  : Head sensor(back)
headSensorF    range=[0.000000,60.000000]       unit=uPa  : Head sensor(front)
faceSensor     range=[0.000000,60.000000]       unit=uPa  : Face sensor
psd            range=[0.000000,90.000000]       unit=cm   : Position Sensing Device
accelX         range=[-19.613300,19.613300]     unit=m/s2 : Acceleration sensor(right-l
accelY         range=[-19.613300,19.613300]     unit=m/s2 : Acceleration sensor(front-b
accelZ         range=[-19.613300,19.613300]     unit=m/s2 : Acceleration sensor(up-down
```

For all these devices, you have the following fields:

- `val` : the value of the device
- `valn` : the normalized value of the device

4.3.4 Camera

The camera device on Aibo is called `camera`. The available fields are:

- `val` : the image (binary)
- `shutter` : the camera shutter speed: 1=SLOW (default), 2=MID, 3=FAST
- `gain` : the camera gain: 1=LOW, 2=MID, 3=HIGH (default)
- `wb` : the camera white balance: 1=INDOOR (default), 2=OUTDOOR, 3=FLUO
- `format` : the camera image format: 0=YCbCr 1=jpeg (default)

- `jpegfactor` : the jpeg compression factor (0 to 100). Default=80
- `resolution` : the image resolution: 0:208x160 (default) 1:104x80 2:52x40
- `reconstruct` : reconstruction of the high resolution image(slow): 0:no (default) 1:yes
- `width` : image width
- `height` : image height
- `xfov` : camera x FOV (degrees)
- `yfov` : camera y FOV (degrees)
- `ballx` : normalized X position of the red ball in the image (between 0 and 1) or -1 when there is no ball visible.
- `bally` : normalized Y position of the red ball in the image (between 0 and 1) or -1 when there is no ball visible.

4.3.5 Speaker

The speaker device on Aibo is called `speaker`. The available fields are:

- `val` : the sound to play (binary)
- `playing` : equal 1 when there is a sound playing, 0 otherwise
- `remain` : number of milliseconds of sound to play, 0 when the buffer is empty

There is also a method available:

- `play (file)` : plays the wav file $file$ which is stored on the root of the memorystick.

4.3.6 Micro

The micro device on Aibo is called `micro`. The available fields are:

- `val` : contains always a buffer of 2048 bytes of the latest sound heard by the robot.

Chapter 5

Default URBI.INI

Here is the default URBI.INI file which sets the standard grouping hierarchy for aibo and starts a nice animation (tagged with `anim`, so that you can stop it with `stop anim`).

```
speaker.play("start.wav");
tps = 4000;

group legRF {legRF1,legRF2,legRF3},
group legLF {legLF1,legLF2,legLF3},
group legRH {legRH1,legRH2,legRH3},
group legLH {legLH1,legLH2,legLH3},
group legs {legRF,legLF,legRH,legLH},
group leg1 {legRF1,legLF1,legRH1,legLH1},
group leg2 {legRF2,legLF2,legRH2,legLH2},
group leg3 {legRF3,legLF3,legRH3,legLH3},

// ERS7
if (global.name == "ERS-7") {
  anim:ledF12.val = 1,
  anim:ledBFW.valn = 0.2 sin:tps ampli:0.5  &
  anim:ledBMW.valn = 0.2 sin:tps ampli:0.5 phase:(pi/3)  &
  anim:ledBRW.valn = 0.2 sin:tps ampli:0.5 phase:(2*pi/3),

  group head {neck,headPan,headTilt,mouth},
  group tail {tailPan,tailTilt},
  group ears {earR,earL},
  group robot {legs,head,tail},

  group ledF {ledF1,ledF2,ledF3,ledF4,ledF5,ledF6,ledF7,
              ledF8,ledF9,ledF10,ledF11,ledF12,ledF13,ledF14},
  group ledHead {modeR,modeG,modeB,ledHC,ledHW},
  group ledBW {ledBFW,ledBMW,ledBRW},
  group ledBC {ledBFC,ledBMC,ledBRC},
  group leds {ledF,ledHead,ledBW,ledBC,ledWIFI}
},

//ERS210
if (global.name == "ERS-210") {
  anim: ledEML.val = 1,
```

```
  anim: ledEMR.val = 1,
  anim: ledTB.val = 0.5 sin:tps ampli:0.6,

  group head {headRoll,headPan,headTilt,mouth},
  group tail {tailPan,tailTilt},
  group robot {legs,head,tail},
  group ears {earR,earL},

  group ledT {ledTB,ledTO},
  group ledE {ledELL,ledEML,ledEUL,ledELR,ledEMR,ledEUR},
  group leds {mode,ledT,ledE}
},
```

Chapter 6

Default CLIENT.INI

Here is the default URBI.INI file which plays a sound and start a battery monitor:

```
speaker.play("client.wav");

//Power Monitoring
current_power=inf;
power:at ( current_power - power() >= 0.01 ) {
  current_power=power();
  power:echo "Battery at "+string(current_power*100)+" %"
};
```