

# **URBI Doc for URBI-engine-bioloid**

## **Devices documentation**

**(book compiled from Revision 425M)**

**Gommard Teddy  
Guillaume Deslandes**

---

# **URBI Doc for URBI-engine-bioloid: Devices documentation: (book compiled from Revision 425M)**

by Gommard Teddy and Guillaume Deslandes

Published

Copyright © 2006-2008 Gostai

This document is released under the Attribution-NonCommercial-NoDerivs 2.0 Creative Commons licence (<http://creativecommons.org/licenses/by-nc-nd/2.0/deed.en>).

---

---

---

# Table of Contents

1. Introduction .....	1
2. Quickstart .....	2
.....	2
How to flash the URBI firmware? .....	2
Installation and first use of the engine .....	2
3. First commands .....	6
Forewords .....	6
Device quick list .....	6
Simple commands .....	6
4. Devices .....	8
Bioloid .....	8
.....	8
.....	8
Motors .....	9
Sensors .....	12
5. bioloid scripting .....	15
Forewords .....	15
Basic motor .....	15
Music .....	15
Bumper .....	16
Navigator .....	16
6. Advanced use - URBI SDK ( <i>write in progress...</i> ) .....	18
.....	18
Advantage of plugged UObjects .....	18
Make your own .....	18
Download UObjects .....	19
7. Troubleshooting and FAQ .....	20
Troubleshooting .....	20
Frequently Asked Questions .....	20
Q: I have a message <code>./rs232/serial-unix.hxx:113: read timeout, got 0 chars on 4</code> when I send commands? .....	20
Q: Error message happens on URBI-server console? .....	20
Q: How to charge batteries? .....	20
Q: What do all the buttons? .....	20
A. Copyright .....	21
.....	21

---

## List of Tables

3.1. Devices characteristics .....	6
4.1. Motors characteristics .....	9
4.2. AX-12 Motor attributes .....	11
4.3. AX-S1 Sensor attributes .....	13

---

# Chapter 1. Introduction

This documentation contains informations about the bioloid URBI engine.

You may be interested in reading information about the bioloid at [www.bioloid.info](http://www.bioloid.info) [<http://www.bioloid.info/tiki/tiki-index.php>] if you are not familiar with it yet.

The first chapters provide quicklaunch instructions and a short tutorial for using URBI on the bioloid.

The remainder of the documentation contains an exhaustive reference for the bioloid URBI engine and more advanced URBI script examples.

---

# Chapter 2. Quickstart

First, make sure your bioloid CM-5 module is operational.

You may have to flash the URBI firmware in the module. This operation is needed because the format of the URBI-engine send data is different from the Robotis protocol.

You can find the hex file here: in `bin/firmware/` folder of your bioloid installation folder or latest version on <http://www.gostai.com/bioloid>

## How to flash the URBI firmware?

1. Use the "robot terminal" software from Robotis (installed with your bioloid softwares).
2. Make sure your robot have batteries fully charged or is sinked by a power supply.
3. Connect your CM-5 brick to the serial comPort of your computer.
4. CM-5 shutoff, press '#' key on keyboard (Alt Gr + 3 on azerty keyboard, shift + 3 on qwerty), stay '#' pushed and switch on the CM-5 brick, (see bioloid documentation for more precision on how to enter bootstrap mode) you must see this:

```
SYSTEM O.K. (CM5 boot loader V1.xx)
-#####
```

5. Press enter key and type load and press enter key again:

```
SYSTEM O.K. (CM5 boot loader V1.xx)
-#####
-load
Write Address: 00000000
Ready..
```

6. In menu bar select Files -> Transmit file (shortcut Ctrl+T) and select the URBI bioloid firmware (hex file), click open.
7. After a few seconds, loading ended:

```
Ready..Success
(and other information)
```

If procedure fails, try again! In case of 'Rewriting:0x0002' message: you already have the good firmware flashed.

## Installation and first use of the engine

Once you have a working robot you can install the bioloid engine. The installation procedure may vary depending on your system, but it should not be unfamiliar to you. Once it is completed, you will be provided with a `bin` directory in which you will find the `urbi-server-bioloid` binary and a script file `urbi.sh` or `urbi.bat` depending of your Operating System: this is the script you need to run to launch the engine.

Other important files are located in the `Bioloid.data` directory.

- `URBI.INI` contains all the scripts that need to be launched at the end of engine startup. This is the entry point.
- `CLIENT.INI` contains all the scripts that need to be launched at each client connection.
- `*.u` which are provided or custom scripts.

Before launching the engine, you may need to change the connection port value in `Bioloid.data/URBI.INI` file with a simple text editor as notepad, gedit, vi, etc... At the same time you can check that `URBI.INI` contains a line like `load( " " );`.

Files between the brackets of `load( )` command can be `bioloid_scan.u` file (default file) if you are just starting with urbi. This file scan your bioloid network and create all necessary objects.

You can obviously create your own `.u` files with custom behaviours. To be successfully loaded at engine startup, these files will need to be :

- in the `Bioloid.data/` folder
- loaded by `URBI.INI` ( `load( "<YOUR-CUSTOM-U-FILE>.u" )` )

Check this file to see how to do this yourself if result is not appropriated to your bioloid network. (in particular, order of AX-12 motors, depending of your assembly).

```
Bioloid.connect(port, speed);
```

Connect to bioloid on port `port` at speed `speed`. `COM1` or `/dev/ttyS0` and `57600` are good usual values. This function return 0 for a successful port open. Warning: This does not mean a bioloid is really well connected.

Under Windows example:

```
Bioloid.connect("COM1", 57600);
```

For Windows users, basic launch of the engine can be achieved by the quicklaunch program menu, under the Gostai category.

Go to your bioloid install directory, change to the `bin/` directory and launch `urbi-server-bioloid --help`.

The following help message should appear :

```
usage: ./urbi-server-bioloid [OPTIONS] [PATHS...]

  PATHS    absolute or relative path elements searched
           in order for files when 'load' is called.

Options:
  -p PORT  specify the tcp port URBI will listen to.
  -b ADDR  bind to a specific ip address.
```

```
-w FILE    write port number to specified file.
-n         disable networking.
-r         report the time taken by URBI loop to execute
-i         treat stdin as an urbi connection
-P PERIOD  base URBI interval in milliseconds
-s PERIOD  shell-mode (no network) with given period
-t         use high priority scheduling
-f         fast mode: the server will run as fast
           as possible and emulate the period specified
-v, --version  print version information and exit
```

You can use script file can have a different extension, depending of your system:

- .bat for windows users.
- .sh for linux users.

to quickly launch the bioloid engine, run the urbi .bat file (resp. .sh) in the bin/ directory of the installation.

Just edit this file if you want to customize the way you start your engine. Here is some examples :

```
./urbi-server-bioloid -P 40
```

in urbi .sh for unix systems, or

```
cmd.exe /C urbi-server-bioloid -P 40 ../gostai/Bioloid.data
```

in a urbi .bat file for Windows users.

The engine will display a Gostai header. Next, there can be messages from the different scripts which are launched in the URBI.INI. If everything's all right, nothing more should happen.

At this time, your engine should be running. To test the connection, use either a tool specially provided by Gostai (like URBI\_console, see <http://www.urbiforge.com/>), or a simple telnet client (please note that all URBI engine ports default to 54000).

```
telnet localhost 54000
```

If no error occurs, your telnet client should receive the same kind of header you saw in the engine window. The last line should give your connection Id and be of the following form :

```
[65000000:ident] *** ID: U135766920
```

If this is not the case, please make sure you correctly installed the engine and followed the previous steps. Please refer to Troubleshooting and FAQ if you still have problems launching the server afterwards.

You are now ready to send commands to your robot through the bioloid URBI engine. For exemple try :

```
vars;
...
[00004004] *** Bioloid = OBJ
```

...

The first line asks for all variables known by the server and the following, which you don't have to take care of, list symbols and their respective values.

There is some special files like `URBI.INI` which is read at start of the engine, if this file exist. There is a `URBI.INI` file in the Bioloid standard package, you can run your engine with this file by entering this:

```
./urbi-server-bioloid -P 40 ../gostai/Bioloid.data
```

The folder `../gostai/Bioloid.data/` contain a `URBI.INI` file and some files.u.

Note: before entering previous command, make sure you have edited `bioloid_scan.u` and uncommented the line `Bioloid.connect` adapted to your system.

---

# Chapter 3. First commands

## Forewords

This chapter will introduce basic concepts for the use of the bioloid URBI engine. It may be skipped if you are already familiar with URBI engines.

For further instructions and informations on URBI commands, please read the first chapters of the URBI tutorial [<http://www.gostai.com/doc/en/urbi-tutorial-1.5>].

Before anything else, make sure the steps presented in the Quickstart chapter succeeded.

## Device quick list

Here is a quick list of the devices available on the bioloid which you can control through the engine. This are the default devices created with no .u file or .ini file

**Table 3.1. Devices characteristics**

Name	Type	Description
Bioloid	Actor	Robot root
Motor	Actor	Motor device: parent, create one by AX-12
Sensor	Actor	Sensor device: parent, create one by AX-S1 sensor

This devices are parent devices, it is needed to create devices from them to use with your bioloid robot: this is explained next.

## Simple commands

Try this first command in your telnet session :

```
sensor = new Sensor(100);  
[00000124] 0.000000
```

If not already done, this line create the UObject linked to your real AX-S1 dynamixel with ID 100, change this value to fit your configuration.

```
sensor.irC;  
[00000476] 0.000000
```

The first line asks for the value of the center IR emitter. The second is the answer from the bioloid engine. The first integer value is a timestamp for the answer (in miliseconds from the engine startup). It is followed by a tag. Please refer to the URBI tutorial [<http://www.gostai.com/doc/en/urbi-tutorial-1.0>] for more details. What is important here is the floating point value ending the line. It is the value stored in the front IR emitter. Here *0.000000* indicates nothing is in front of the robot. With your hand right in front of the robot, you should get a value close to *255.000000*.

Please note that the next commands will be enclosed in `+end: { . . . };`. This is to make the engine warn you when the command comes to an end.

Try the following :

```
+end:{motor__1.speed = 100; sleep (2s) | motor__1.speed = 0;};  
[00002622] *** end
```

Your robot should have the 1st motor rotating for 2 second, and then stop. Now try to make it come back to its initial position with a similar command.

Next try :

```
+end:{motor__1.speed = -100; sleep (2s) | motor__1.speed = 0;};  
[00005095] *** end
```

Your motor should come back to its original position.

The last simple command will be :

```
sensor on;  
irCDetect:at (sensor.irC > 100) echo "Front obstacle : " + sensor.irC;
```

Now try to put your hand close to the front of your bioloid AX-S1 module. You should get a message like :

```
[00008215] *** Front obstacle : 230
```

The `at` keyword allows you to detect events and make your bioloid act accordingly.

The bioloid interface is close to the one specified in the manual of the AX-12 and of the AX-S1. Type `"myMotor; mySensor;"` in a connection (where `myMotor` is an instantiated `Motor`, and `mySensor` a `Sensor`) to get the full list of their attributes.

You should now be able to play with the bioloid URBI engine. The next chapters will give you complete details about the engine, but what we saw here is enough to get used with the bioloid and URBI.

---

# Chapter 4. Devices

## Bioloid

This object is the master of all other objects for Bioloid. It has only three methods available for user.

**connect("device",int speed)** Connect to RS232 Bioloid network. This method need to be call first of all other commands. it returns 0 on connection success.

under Unix system:

```
Bioloid.connect("/dev/ttyS0", 57600);  
[00000066] 0
```

under Windows system:

```
Bioloid.connect("COM1", 57600);  
[00000076] 0
```

Please, note that the firmware flashed in Bioloid CM-5 is programmed for working at 57600 bauds for the moment; Giving 57600 as parameter in all cases for the second parameter is a good practise.

**ping(int Id)** Ping ID of a dynamixel to test if dynamixel is present on network. This method return the ID if dynamixel is present, 255 if not present or ID unavailable

```
Bioloid.ping(12);  
[00004993] 12  
Bioloid.ping(29);  
[00012900] 255
```

A dynamixel with ID 12 was on network, but no one with ID 29.

**scan()** Scan all the 253 available IDs (0~252, no need to test broadcast ID 254 and **#253 is reserved for CM-5 management**), and return list of all dynamixels on the network. This is a blocking function that can take a bit time (2~3 seconds). In most cases, you need just to call it once at start to discover network (just after doing a Bioloid.connect ).

```
Bioloid.scan();  
scanning network, may take some time...  
[00099415] [12, 18, 100]  
Bioloid;  
[00109621] OBJ [listAXS1:[100],load:1,listAX12:[12, 18]]
```

This function automatically fill the Bioloid.listAXS1 and Bioloid.listAX12 lists with the number of the present nodes ID on the network.

*Note: How scan() works: it does a ping on all IDs and wait for reply.*

After calling this functions, two lists are available in Bioloid object: `Bioloid.listAXS1` and `Bioloid.listAX12`. The lists contain IDs (sorted) of respectively AX-S1 and AX-12 dynamixels. You can use these lists to create all your motors and sensors objects. (see `<bioloid-dir>/Bioloid.data/bioloid_scan.u` in your installation directory for more details and examples).

## Motors

The bioloid motors are accessed through the `Motor` objects and have the following characteristics :

**Table 4.1. Motors characteristics**

Name	Rangemin	Rangemax	Description
motor__0	-1023	1023	motor with ID #0
motor__1	-1023	1023	motor with ID #1
...	-1023	1023	...
motor__x	-1023	1023	motor with ID #x

If your engine doesn't load any filename.u at start (declared in `CLIENT.INI` or in `URBI.INI`), by default no motors are created because of the structure of the bioloid, we don't know how many motors are in your robot. You must declare them before sending any command to them

```
motor[i] = new Motor(j);
```

This method can be used to create an array for all your motors. Remember the array begins at index 0. Moreover, the parameter you give to `Motor` is the ID of the motor you want to instantiate.

```
var i = 0;
for j in [10,12,16,18] {
  motor[i] = new Motor(j);
  i = i + 1;
};
```

or for a full automated creation of all available motors:

```
var i = 0;
for j in Bioloid.listAX12 {
  motor[i] = new Motor(j);
  i = i + 1;
};
```

this sample create the 4 motors of the bioloid car:

- Motor with #ID10 is assigned to `motor__0` (same as `motor[0]`)
- Motor with #ID12 is assigned to `motor__1` (same as `motor[1]`)
- Motor with #ID16 is assigned to `motor__2` (same as `motor[2]`)

- Motor with #ID18 is assigned to motor\_\_3 (same as motor[3])

Motors are available by the motor\_\_X, with X between 0 and your max\_numbered\_motor. You can (need to) adapt the #ID of motors in your scripts to the configuration of your robot (it's easier to rename each motor)!

**Naming Standard:** Depending the form of your bioloid robot (humanoid, car, quaduped, etc ...) \*.u files exists for creating aliases and groups for this motors, to have a common naming standard for all robot. see Bioloid.data folder.

Not all config are already defined, don't hesitate to create and share your own .u files on [www.urbiforge.org](http://www.urbiforge.org)

A motor has the following attributes:

			10.0V)
lowVoltageLimit	50	250	0x0C set 10 x volatge threshold to set alarm
highVoltageLimit	50	250	0x0D set 10 x volatge threshold to set alarm
<b>Table 4.2: AX-12 Motor attributes</b>			
		1023	0x30 minimum current supplied to motor (32 is better as minimum)
<b>LoopBack</b>			
ccwAsservMargin	0	254	0x1B
cwAsservMargin	0	254	0x1A
cwAsservGain	0	254	0x1C
ccwAsservGain	0	254	0x1D
<b>Led</b>			
LED	0	1	0x19 switch Off/On red led on this motor; 1 is ON
alarmLED	0	127	0x11 if set bits to 1 led blink: there is a selected error
alarmShutdown	0	127	0x12 if selected bit is set, torque is set to 0 on selected error(s)
<b>Identifier</b>			
ID	0	253	0x03 Read ID of AX-12 motor
firmwareVersion	0	255	0x02 Read firmware version
modelName	0	65535	0x00 Read always 12 for an AX-12
baudRate	0	254	0x04 Communication speed. refer bioloid AX-12 manual (1 is 1Mbps)
<b>Other informations</b>			
downCalibration	0	65535	0x14 ReadOnly: delta between potentiometer and position
upCalibration	0	65535	0x16 ReadOnly: delta between potentiometer and position
registeredInstruction	0	1	0x2C ...
returnDelay	0	254	0x05 time delay for return status packet: 2us *returnDelay
returnStatus	0	2	0x10 status of dynamixel, equal 2 in normal mode
lock	1	1	0x2F if set to 1 only speed and val can be written, power down to disable

all this attributes are accessible by calling `myMotor.attribute` like:

- `motor__1.load`; to read the load attribute of `motor[1]` object.
- `motor__2.LED = 1`; to write the LED attribute of `motor[2]` object.
- `motor[2].LED = 1`; to write the LED attribute of `motor[2]` object.

You can drive your motor in two modes, depending of the close-loop control you want:

- Position mode : with `motor__X.val`, the motor turn to a position and stop
- Speed mode : with `motor__X.speed`, the motor turn endless at this speed

To use this feature you need to set `.cwAngleLimit` and `.ccwAngleLimit` to adapted values:

- Position mode : `.cwAngleLimit` and `.ccwAngleLimit` are limiting position values, clockwise and counter-clockwise.
- Speed mode : `.cwAngleLimit` and `.ccwAngleLimit` equal to 0.

Note that every descriptions beginning with a `0xNN` address number is directly associated to AX-12 memory address, see dynamixel AX-12 manual for more informations on these functions : AX-12 Dynamixel [<http://www.robotshop.ca/PDF/robotis-AX-12-specifications.pdf>] or here [[http://www.electronickits.com/robot/BioloidAX-12\(english\).pdf](http://www.electronickits.com/robot/BioloidAX-12(english).pdf)]

## Sensors

When the bioloid engine is ready, you can create `Sensor` objects ( same way as `Motor` objects ). Then you can access to their different members which return all values of an AX-S1 module. To create a new sensor uobject (AX-S1 #ID is 100 in this example):

```
sensor = new Sensor (100);
```

The AX-S1 dynamixel sensor module is accessed through the `Sensor` object and have the following attributes :

			in RAM (0x34) at power ON
<b>Buzzer</b>			
sound	0	Devices	255
			0x23 return max sound amplitude 0 mini <128 null <255 max
<b>Table 4.3. AX-S1 Sensor attributes</b>			
soundMaxHold	0		255
			0x24 low pass filter, 0 for measuring max loudness
soundOccurenceCount	0		255
			0x25 Sound clap counter
soundOccurenceTime	0		65535
			0x26 Time of last detected clap
buzzerIndex (speaker.val)	0		51
			0x28 Note to play. Play note at val assignment 0=1a, 1=1a#,...
buzzerTime	3		50
			0x29, time to play next note x0.1s (from 0.3 to 5s)
<b>Temperature</b>			
temperature (temperature.val)	0		255
			0x2B Read temperature in celcius degrees
temperatureLimit	0		150
			0x0B Temperature Alerte threshold
<b>Power supply</b>			
voltage (voltage.val)	0		255
			0x2A Read 10 x voltage
lowVoltageLimit	50		250
			0x0C
highVoltageLimit	50		250
			0x0D
<b>IR communication</b>			
IRRemoconRX0 (IRRemoconRX0.val)	0		255
			0x30 Read first byte received
IRRemoconRX1 (IRRemoconRX1.val)	0		255
			0x31 Read second byte received
IRRemoconTX0 (IRRemoconTX0.val)	0		255
			0x32 first byte to transmit
IRRemoconTX1 (IRRemoconTX1.val)	0		255
			0x33 second byte to transmit
IRRemoconArrived (IRRemoconArrived.val)	0		255
			0x2E Read
<b>Other Informations</b>			
load	0		1
			..
ID	0		253
			0x03 Read ID of AX-S1
firmwareVersion	0		255
			0x02 Read firmware version
modelName	0		65535
			0x00 Read model number: is 13 for AX-S1
returnStatus	0		2
			0x10
baudRate	0		254
			0x04 baudrate
registeredInstruction	0		1
			0x2C
returnDelay	0		254
			0x05
lock	1		1
			0x2F

Here you can find the AX-S1 Specifications [[http://www.tribotix.info/Downloads/Robotis/Bioloid/AX-S1\(english\).pdf](http://www.tribotix.info/Downloads/Robotis/Bioloid/AX-S1(english).pdf)] for more information.

---

# Chapter 5. bioloid scripting

## Forewords

This chapter will introduce a few simple scripts. To use them type in the console :

```
load ("filename.u");
```

Depending on the version of the bioloid URBI engine you got, some scripts may be available in the `data`, `profiles` or `scripts` folders of your installation directory.

Before anything else, make sure you are familiar with the previous parts of this document.

For further instructions and informations on URBI commands, please read the first chapters of the URBI tutorial [<http://www.gostai.com/doc/en/URBI-tutorial-1.0>].

All these simple scripts assumes URBI.INI is empty, so comment lines with `#` or `//` at start of each lines of URBI . INI to avoid unpredictable behaviour.

## Basic motor

```
/* ----- Basic for bioloid ----- */
/// assume there is nothing in the URBI.INI
/// Connect to bioloid (do not do if already connected!)
Bioloid.connect("COM1", 57600);

/// Create motors
var i = 0;
//declare ID of your connected motors there
for j in [10,12,16,18] {
  motor[i] = new Motor(j);
  i = i + 1;
};

var NumberOfMotor = i;

t: {
  for (j=0; j<3;j++) {
    for (i=0; i<NumberOfMotor;i++) motor[i].speed = 100 smooth:1s;
    sleep(1.5s);
    for (i=0; i<NumberOfMotor;i++) motor[i].speed = 0 smooth:1s;
    sleep(1.5s);
  };
};
/* ----- Basic for bioloid ----- */
```

This should make your bioloid test all motors by a little move, 3 times.

## Music

```

/* ----- Music for bioloid ----- */
/// assume there is nothing in the URBI.INI
/// Connect to bioloid (do not do if already connected!)
Bioloid.connect("COM1", 57600);

/// Create sensor
sensor = new Sensor(100);
music:{
  for (i=0; i<27;i++){
    sensor.buzzerTime=255;
    sensor.buzzerIndex=i;
    sleep(1.5s);
  };
};
/* ----- Music for bioloid ----- */

```

This should make your bioloid AX-S1 module play its 26 pre-recorded melodies one by one with a wait of 1.5 seconds between each.

## Bumper

```

/* ----- Bumper for bioloid car ----- */
bumper:at (irC > 100) // at obstacle detection
{
  wheels = 0 | sleep (500ms); // stop and wait a bit
  wheels = -50 | sleep (1s); // go backward
  wheels = 0 time:1s | // stop smoothly
  wheels = 100; // go forward
};

wheels = 100; // initial start
/* ----- Bumper for bioloid car ----- */

```

This should make your bioloid car go forward, stop before the first obstacle, go back for a bit and go again. You will have to figure how to make it stop. You must load `bioloid_car.u` before executing this script to have good result. (it's better to have a robot with 4 wheels to have a good look of the behaviour, see description of motors in `bioloid_car.u`).

## Navigator

```

/* ----- Navigator for bioloid car ----- */
var WHEEL_SLOW = 30; // moving speed
var WHEEL_FAST = 80; // turning speed
var NAV.side = 1; // way to turn

navigator:at (irC > 150) // at obstacle detection
{
  wheels = 0 | sleep (500ms); // wait a bit
  wheels = -40 | sleep (500ms); // go back
  NAV.side = NAV.side * -1; // change way
  wheelL = NAV.side * WHEEL_FAST & // turn

```

```
    wheelR = - NAV.side * WHEEL_FAST;
    waituntil (irC < 50) | wheels = WHEEL_SLOW; // wait for clear way
                                                // and go forward
};
```

```
wheels = WHEEL_SLOW; // initial start
```

```
/* ----- Navigator for bioloid car ----- */
```

You must load `bioloid_car.u` before executing this script to have good result.

This script should make your robot turn either left or right upon obstacle detection until it finds a clear way. It then will go forward again until the next obstacle.

---

# Chapter 6. Advanced use - URBI SDK (*write in progress...*)

In this section you will learn how to create your own Bioloid engine with the modules you need for your Bioloid robot.

Binary executable file `urbi-server-bioloid` contain only UObjects to manage CM-5, AX-12 and AX-S1 bricks. If you want your robot to have other features, like a camera, micro, etc ... you need to add UObjects to your engine (and hardware, too): they can be plugged or remote.

## Advantage of plugged UObjects

**Remote components:** UObjects that are self-executable, they need to connect to a running engine. they are compiled alone and they can connect to any engine.

**Plugged components:** UObjects compiled inside the engine, they are automatically launched with the engine as they are fully belonging to it.

More detailed information on this at : *URBI quickstart* [<http://www.gostai.com/doc.php>], in particular the UObject chapter.

By creating your own engine you have only one executable to launch, no need to launch engine and after the remote components. How to obtain this result is explained in the next section.

If you need to add a module to your engine, having a plugged UObjects is more practical when using engine (only one executable to launch). One Disadvantage is you need to compile the whole engine when you do any modification on one plugged UObject. For a remote component, only compiling the UObject which is modified is require on update.

With your URBI Bioloid SDK is provided a `liburbiengine-bioloid.a` file which is a library containing the Bioloid engine; linking this file when compiling your UObject is the best way to create your own Bioloid engine.

## Make your own

Create your own UObjects : generally these are `.cc` files, see *tutorial: UObject documentation* [<http://www.gostai.com/doc.php>] for more details.

How to compile:

Template exists for Visual C++, Dev-CPP, see *Gostai Website* [<http://www.gostai.com>] for downloads

Manual method: create a `Makefile` file with this in it:

```
#name of the final binary
PRG=myBioloidEngine

#where is the folder named "urbiengine-bioloid"
URBIENGINE_BIOLOID_DIR=/  

```

```
LIBURBIENGINE_BIOLOID=\
$(URBIENGINE_BIOLOID_DIR)/usr/local/gostai/core/i686-pc-linux-gnu/\
engine/liburbiengine-bioloid.a
```

```
BIN_BIOLOID=$(URBIENGINE_BIOLOID_DIR)/usr/local/bin
```

all:

```
$(BIN_BIOLOID)/umake-engine --prefix=../usr/local \
-o $(PRG) $(LIBURBIENGINE_BIOLOID) *.cc
```

clean:

```
rm -rf _ubuild-$(PRG) $(PRG)
```

\$PRG is the name of the final binary you generate.

\$URBIENGINE\_BIOLOID\_DIR is the install directory where can be found *.../gostai/core/.../liburbi-engine-bioloid.a*.

To compile your engine, just enter `make` in a shell in your directory.

## Download UObjects

You can find UObjects here for example: *Urbiforge* [<http://www.urbiforge.org>].

---

# Chapter 7. Troubleshooting and FAQ

## Troubleshooting

None known at this time. Make us feedback if any.

## Frequently Asked Questions

### Q: I have a message `./rs232/serial-unix.hxx:113: read timeout, got 0 chars on 4` when I send commands?

A: This happen when your bioloid CM-5 module is power down or when you connect on the bad RS232 port (COMport).

- Check the power supply of the bioloid (some Leds need to light).
- Check the physical connection between your bioloid and computer (Port Number and speed).

### Q: Error message happens on URBI-server console?

```
-----  
| bad response:  
| - error: 8: range error  
| - id: 10  
| - value: 0  
| - raw value: [235, ]  
| - value length: 0  
-----
```

A: This message is specific to the CM-5 module. It's a status packet containing the error. Please check the AX-12 documentation to interpret this error.

In most of case, this happenned when you switched from speed mode to close-loop control mode by setting `ccwAngleLimit` and `cwAngleLimit`. if the position (`.val`) isn't in the range and you call a `motor__X.val=y`; this kind of error is thrown: this is normal.

### Q: How to charge batteries?

A: We have implemented in the firmware the battery management. It is the same procedure as the original firmware : push U ( "up" ) button. Power Led shall blink to the end of the charge. this documentation [Leds-Code-Battery-management.odt](http://www.gostai.com/bioloid) [<http://www.gostai.com/bioloid>] (OpenOffice format) explain in detail all the signification of leds behaviour on CM-5 with URBI firmware.

### Q: What do all the buttons?

A: As we develop a new firmware, the old functionalities from Robotis are unavailalble while Urbi firmware is in the CM-5. Actually, only the U button has a use (see above), Mode button reset the CM-5 module, others are useless (see documentation above for details).

---

# Appendix A. Copyright

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

## 1. Definitions

1. "Collective Work" means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License. 2. "Derivative Work" means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License. 3. "Licensor" means the individual or entity that offers the Work under the terms of this License. 4. "Original Author" means the individual or entity who created the Work. 5. "Work" means the copyrightable work of authorship offered under the terms of this License. 6. "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive,

perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

1. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works; 2. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights to make Derivative Works. All rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Sections 4(d) and 4(e).

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

1. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any reference to such Licensor or the Original Author, as requested.
2. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.
3. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work, You must keep intact all copyright notices for the Work and give the Original Author credit reasonable to the medium or means You are utilizing by conveying the name (or pseudonym if applicable) of the Original Author if supplied; the title of the Work if supplied;

and to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work. Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit. 4.

For the avoidance of doubt, where the Work is a musical composition:

1. Performance Royalties Under Blanket Licenses. Licensor reserves the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work if that performance is primarily intended for or directed toward commercial advantage or private monetary compensation.
2. Mechanical Rights and Statutory Royalties. Licensor reserves the exclusive right to collect, whether individually or via a music rights agency or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions), if Your distribution of such cover version is primarily intended for or directed toward commercial advantage or private monetary compensation.
5. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

#### 5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR

EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

1. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License. 2. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

1. Each time You distribute or publicly digitally perform the Work or a Collective Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License. 2. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable. 3. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent. 4. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.