

Introduction to liburbi Java for Urbi 1.x

(book compiled from 682M)

Remi Humbert

Introduction to liburbi Java for Urbi 1.x: (book compiled from 682M)

by Remi Humbert

Publication date

Copyright © 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. Getting started	2
Loading liburbijava shared library	2
Connecting	2
Sending URBI commands	3
Sending binary data.	3
Sending a sound	3
3. Receiving	4
Tags and callbacks	4
UMessage	4
UValue	5
UBinary	6
USound	6
UImage	7
A few examples	8
4. Develop with Eclipse	11
Import the liburbi java project	11
Run the examples	14
5. Programming hints	16
A. Copyright	17

Chapter 1. Introduction

Liburbi-java is a library designed to encapsulate an URBI connection. It handles the TCP connection with the URBI server, and the dispatching of messages it sends. The library is thread-safe and reentrant. This library is generated from the Liburbi-c++ source with swig.

The library consists of two main Java classes, `liburbi.UClient` and `liburbi.USyncClient`, and a few helpful functions.

We expect the reader to be a bit familiar with the URBI syntax.

Chapter 2. Getting started

Loading liburbijava shared library

Liburbi Java is a library generated from the liburbi C++ with the tool called swig [<http://www.swig.org/>]. Swig generates wrapper classes in java for all the C++ classes we want to export. It also generates JNI code to make the link between Java and the C++ library. With this in mind, you understand that in order to use the Liburbi Java, you have to somehow load the C++ JNI code in your application. We provide this C++ code in a shared library called urbijava.dll under Windows, liburbijava.so under Linux, and liburbijava.dylib under MacOS. To load the library in your java code, use the function `System.loadLibrary`. For example place this call in your Main class as showed bellow:

```
// All liburbi classes are located in the package liburbi
import liburbi.main.UClient;
import liburbi.main.liburbi;

public class Main {

    static {
        System.loadLibrary("urbijava");
    }
}
```

NB: The path to the library you are loading 'urbijava' must be in your path for your program to find it. On Windows make sure the 'path' environment variable contains the path to the urbijava library. On Unix make sure that your LD_LIBRARY_PATH contains the path to the urbijava library.

Connecting

To connect to an URBI server, simply create a new instance of `liburbi.main.UClient` (or `liburbi.main.USyncClient` if you want to use the synchronous functions described bellow), passing the name or the address of the server as the first parameter, and optionnaly the port as the second parameter:

```
public static void main (String[] arg) {

    UClient client = new UClient("myrobot.ensta.fr");

    // a wrapper is also available in the liburbi.main.liburbi class:
    // UClient client = liburbi.connect("myrobot.ensta.fr");
}
```

The constructor will start an independant thread that will listen for incoming messages from the URBI server.

You can check if the connection was successfully established by calling the error function, which returns a zero value on success, or a nonzero error code in case of failure.

```
if (client.error() != 0)
{
    System.err.println("Couldn't connect to the URBI server.");
}
```

```
return;  
}
```

Sending URBI commands

The method `send` is the simplest way to send commands to the URBI server. It takes a `String` as parameter:

```
int sleeptime = 50;  
  
client.send("motoron;");  
  
for (float val=0; val<=1; val+=0.05)  
    client->send("neck.val = " + val + ";wait (" + sleeptime + ");");
```

Some static functions are defined in the class `liburbi.liburbi`: `'getComma ()'`, `'getSemicolon()'`, `'getPipe()'` and `'getParallel()'` and they return the characters `'.'`, `'.'`, `'|'` and `'&'` respectively.

Sending binary data.

To send binary data to the robot, the method `sendBin` must be used. It takes as parameters the buffer to send (we only support `byte[]` type for now) and its size, and optionally a header.

```
client->sendBin(soundData, soundDataSize,  
    "speaker.val = BIN " + soundDataSize + " raw 2 16000 16 1;")
```

Sending a sound

Although you could use `sendBin` to play a sound on the robot, a specific and efficient method has been written for this purpose: `sendSound`.

```
client->sendSound("speaker", sound, "endsound");
```

The first parameter is the sound device to which we send the sound, the second parameter is an `liburbi.main.USound` class describing the sound to send. The third is an optional tag that will be used by the server to issue a "stop" system message when the sound has finished playing. The function `convert` (in `liburbi.liburbi` class) can be used to convert between various sound formats.

There is no limit to the size of the sound buffer, since it will be automatically cut into small chunks by the library. The data is copied by the library: the `USound` parameter and its associated data can be safely freed as soon as the function returns.

Chapter 3. Receiving

Tags and callbacks

Most of the messages received from the URBI server are the results of a previously sent command. The mechanism of URBI tags enables to link a message to its reply: with each command is associated a tag, and this tag is repeated in the reply message. The `liburbi.main.UClient` class handles the reception of those messages in the independant thread created by the constructor, parses them and fills a `UMessage` structure.

In your Java program, you can have classes extending `liburbi.main.UCallbackInterface` and redefining its method `onMessage`. You can register these classes with the `UClient` method `setCallback` and associate them with a tag. Then each time a message with this tag is sent by the server, the redefined `onMessage` function will be called with a `liburbi.main.UMessage` class as a parameter.

The `liburbi.main.UCallbackInterface` class is defined as follow:

```
package liburbi.main;

public class UCallbackInterface {
    public UCallbackAction onMessage(UMessage msg);
}
```

The `onMessage` function take as parameter a `liburbi.main.UMessage` and return an enum: `liburbi.main.UCallbackAction` which can have two values: `liburbi.main.UCallbackAction.URBI_CONTINUE` or `liburbi.main.UCallbackAction.URBI_REMOVE` (in which case the callback is removed, and wont be called anymore, even if you receive a message with the same tag).

The `setCallback` function is defined as follow:

```
package liburbi.main;

public class UClient {

    [...]
    public long setCallback(UCallbackInterface ref, String tag);
}
```

Give as first parameter your class redefining `liburbi.main.UCallbackInterface`, and give as second argument a `String` corresponding to the tag you want to associated the callback with.

UMessage

The `UMessage` structure is capable of storing the informations contained in any kind of URBI message by using a "getType" function and an `UValue` (union of classes). These two classes are defined as follows:

```
package liburbi.main;

public enum UMessageType {
    MESSAGE_SYSTEM, /// System message
```



```
MESSAGE_ERROR,    /// Error message
MESSAGE_DATA;     /// Message containing data in the UValue
}

public class UMessage {

    /// Server-side timestamp.
    public int getTimestamp();

    /// Associated tag.
    public String getTag();

    /// Type of the UValue contained in the UMessage
    public UMessageType getType();

    /// UValue contained in the UMessage
    public UValue getValue();

    public String getMessage();

    /// Raw message without the binary data.
    public String getRawMessage();

    /// Client from which originated the message.
    public UAbstractClient getClient();
}
```

The type field `UMessageType` can be `MESSAGE_SYSTEM`, `MESSAGE_ERROR` or `MESSAGE_DATA`. If the type is `MESSAGE_DATA`, the message contains an `UValue`.

UValue

```
package liburbi.main;

public enum UDataType {
    DATA_DOUBLE,
    DATA_STRING,
    DATA_BINARY,
    DATA_LIST,
    DATA_OBJECT,
    DATA_VOID;
}

public class UValue {

    public UDataType getType();

    /// value if of type UDataType.DATA_STRING
    /// check that string is not null (it can be in the C++ side,
    /// remember liburbijava is build on top of liburbic++)
    /// with 'isStringNull ()'
    public String getString();
    public boolean isStringNull();

    /// value if of type UDataType.DATA_DOUBLE
    public double getDouble();
}
```

```
/// value if of type UDataType.DATA_BINARY
public UBinary getUBinary();

/// value if of type UDataType.DATA_LIST
public UList getUList();

/// Used to print the UValue conveniently
public String toString();
}
```

The UValue contains an UDataType which can take the values: DATA_DOUBLE, DATA_STRING, DATA_BINARY, DATA_LIST, DATA_OBJECT, DATA_VOID. Depending of this field, the corresponding value in the UValue will be set. If the UValue is of binary type, it contains an UBinary class defined hereafter. The UBinaryType in the UBinary structure will give additional informations on the type of data (BINARY_NONE, BINARY_UNKNOWN, BINARY_IMAGE, BINARY_SOUND), and the appropriate sound or image structure will be filled.

UBinary

```
package liburbi.main;

public enum UBinaryType {
    BINARY_NONE,
    BINARY_UNKNOWN,
    BINARY_IMAGE,
    BINARY_SOUND;
}

public class UBinary {

    public UBinaryType getType();

    /// value if of type BINARY_IMAGE
    public UImage getUImage();

    /// value if of type BINARY_SOUND
    public USound getUSound();

    /// Size of the data
    public int getSize();

    public String getMessage();
    public String getExtraHeader();
}
```

USound

```
package liburbi.main;

public enum USoundFormat {
    SOUND_RAW,
    SOUND_WAV,
    SOUND_MP3,
```

```
SOUND_OGG,  
SOUND_UNKNOWN;  
}  
  
public enum USoundSampleFormat {  
    SAMPLE_SIGNED(1),  
    SAMPLE_UNSIGNED(2);  
}  
  
public class USound {  
  
    /// total size in byte  
    public long getSize();  
  
    /// number of audio channels  
    public int getChannels();  
  
    /// rate in Hertz  
    public int getRate();  
  
    /// sample size in bit  
    public int getSampleSize();  
  
    /// format of the sound data  
    /// (SOUND_RAW, SOUND_WAV, SOUND_MP3...)  
    public USoundFormat getSoundFormat();  
  
    /// sample format  
    public USoundSampleFormat getSampleFormat();  
  
    /// sound data in an array of byte  
    public byte[] getDataAsByte();  
}
```

UImage

```
package liburbi.main;  
  
public enum UImageFormat {  
    IMAGE_RGB(1),  
    IMAGE_YCbCr(2),  
    IMAGE_JPEG(3),  
    IMAGE_PPM(4),  
    IMAGE_UNKNOWN;  
}  
  
public class UImage {  
  
    /// image size in byte  
    public long getSize();  
  
    /// image width  
    public long getWidth();  
}
```

```
    /// image height
    public long getHeight();

    /// IMAGE_RGB, IMAGE_YCbCr, IMAGE_JPEG...
    public UImageFormat getImageFormat();

    /// image data in an array of byte
    public byte[] getDataAsByte();
}
```

A few examples

```
import liburbi.main.*;

/// Callback for images
class onImage extends UCallbackInterface {

    public UCallbackAction onMessage(UMessage msg) {
        if (!(UMessageType.MESSAGE_DATA == msg.getType ()))
            return UCallbackAction.URBI_CONTINUE;
        UValue value = msg.getValue ();

        if (!(UDataType.DATA_BINARY == value.getType ()))
            return UCallbackAction.URBI_CONTINUE;
        UBinary bin = value.getUBinary ();

        if (!(UBinaryType.BINARY_IMAGE == bin.getType ()))
            return UCallbackAction.URBI_CONTINUE;
        UImage img = bin.getUImage ();
        byte[] data = img.getDataAsByte ();

        System.out.println("Image of size " + img.getWidth () + "x"
                           + img.height + " received");

        /// myDisplay*Image are custom functions not part of liburbi-java
        /// replace by your own code
        if (img.imageFormat == UImageFormat.IMAGE_JPEG)
            myDisplayJPEGImage (data, img.getWidth (), img.getHeight ());
        else if (img.imageFormat == UImageFormat.IMAGE_YCbCr)
            myDisplayYCbCrImage (data, img.getWidth (), img.getHeight ());
        else if (img.imageFormat == UImageFormat.IMAGE_RGB)
            myDisplayRGBImage (data, img.getWidth (), img.getHeight ());

        return UCallbackAction.URBI_CONTINUE;
    }
}

/// Callback for sound
class onSound extends UCallbackInterface {

    public UCallbackAction onMessage(UMessage msg) {
        if (!(UMessageType.MESSAGE_DATA == msg.getType ()))
            return UCallbackAction.URBI_CONTINUE;
        UValue value = msg.getValue ();
    }
}
```

```
        if (!(UDatatype.DATA_BINARY == value.getType ()))
            return UCallbackAction.URBI_CONTINUE;
        UBinary bin = value.getUBinary ();

        if (!(UBinaryType.BINARY_SOUND == bin.getType ()))
            return UCallbackAction.URBI_CONTINUE;
        USound sound = bin.getUSound ();

        byte[] data = img.getDataAsByte ();

        System.out.println("Sound received");
        return UCallbackAction.URBI_CONTINUE;
    }
}

/// Callback for joint value
class onJoint extends UCallbackInterface {

    public UCallbackAction onMessage(UMessage msg) {
        if (!(UMessageType.MESSAGE_DATA == msg.getType ()))
            return UCallbackAction.URBI_CONTINUE;
        UValue value = msg.getValue ();

        if (!(UDatatype.DATA_DOUBLE == value.getType ()))
            return UCallbackAction.URBI_CONTINUE;

        System.out.println("The joint value is " + value.getDouble ());
        return UCallbackAction.URBI_CONTINUE;
    }
}

public class test {
    static {
        /// Load the C++ liburbi library and the jni code.
        System.loadLibrary("urbijava");
    }

    public static void main(String argv[]) {

        UClient c = new UClient(argv[1]);

        if (c.error() != 0)
        {
            System.out.println("Couldn't connect to the URBI server.");
            System.exit(1);
        }

        onImage oi = new onImage ();
        onJoint oj = new onJoint ();
        onSound os = new onSound ();

        c.setCallback (oi, "img");
        c.setCallback (oj, "snd");
        c.setCallback (os, "joint");

        c.send("img: camera.val;");
        c.send("loop snd: micro.val;");
        c.send("joint: headPan.val;");
    }
}
```

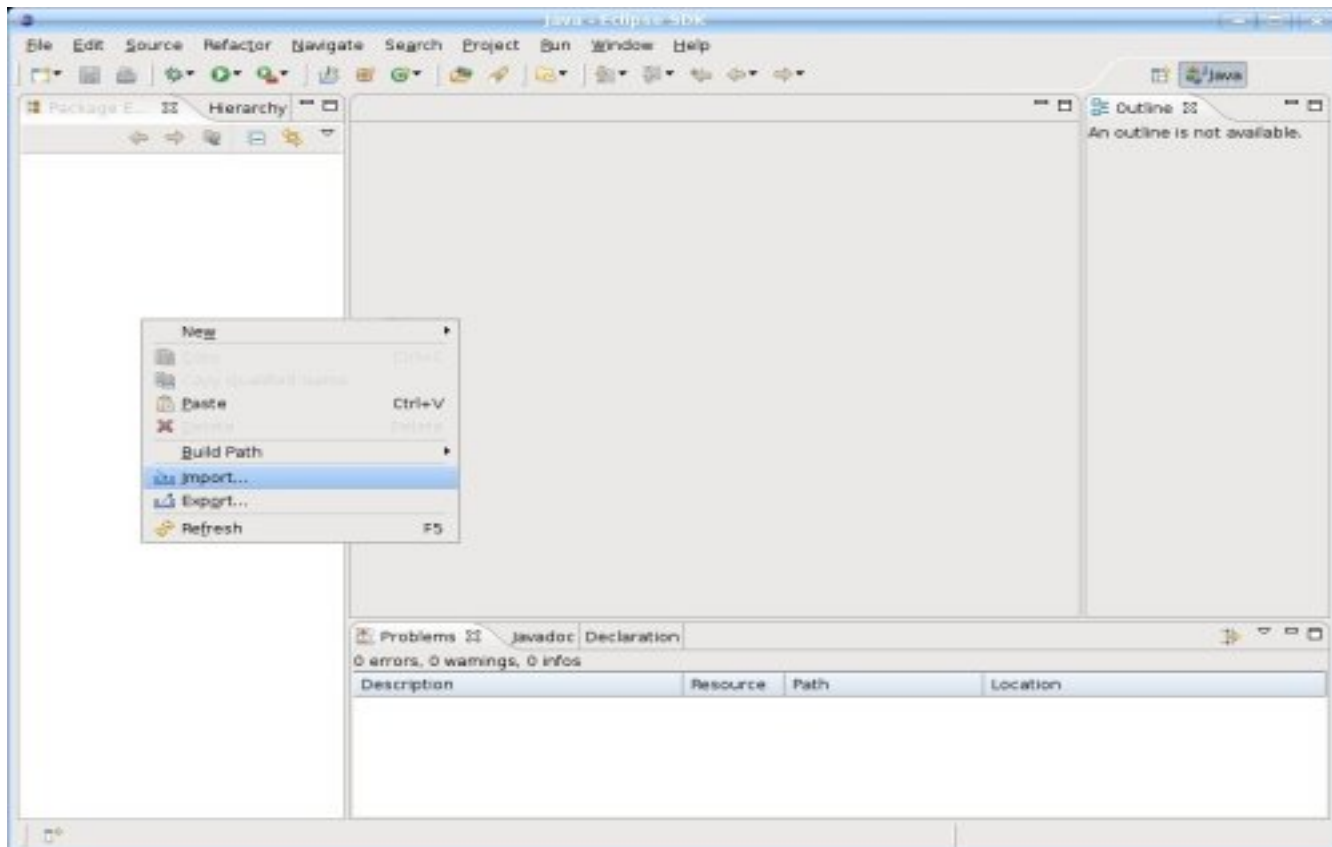
```
        liburbi.execute (); /// function that do a infinite sleep  
    }  
}
```

Chapter 4. Develop with Eclipse

Import the liburbi java project

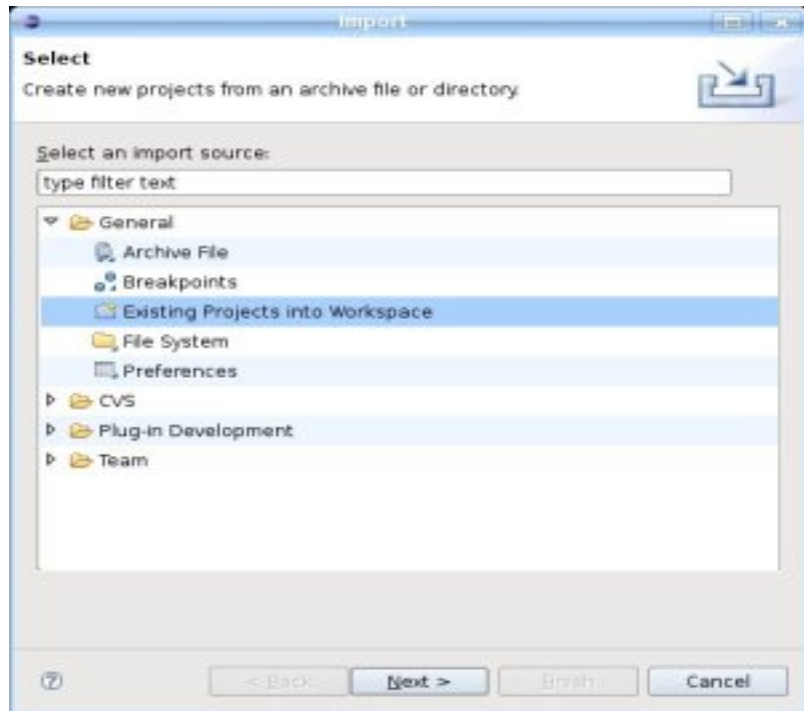
We provide a sample Eclipse project configuration that you can import in Eclipse and use to create your project using Liburbi Java. We illustrate here how you can do this:

- Open Eclipse



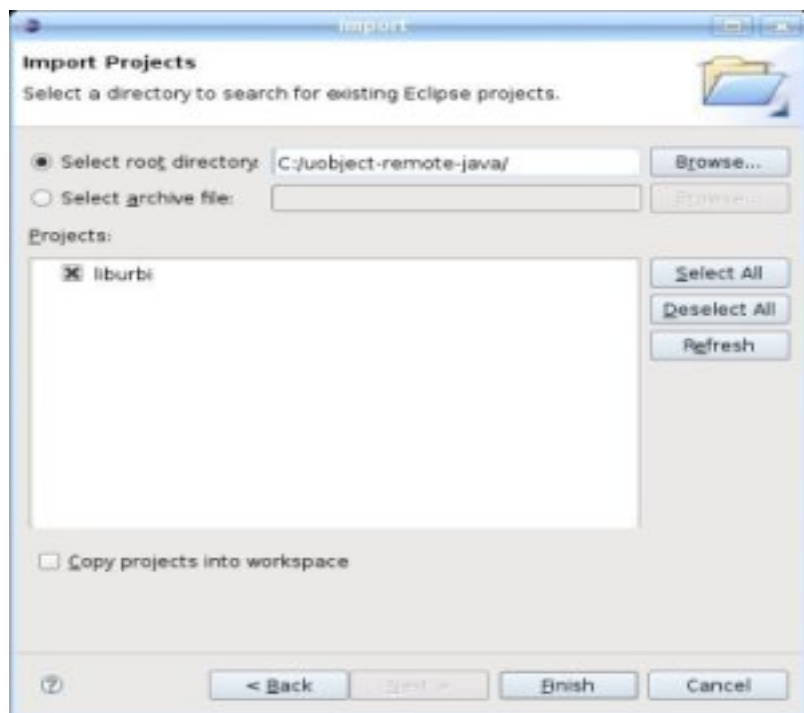
[images/eclipse-import.jpg]

- Right click in the Package Explorer panel and select 'import' (or go in File/import)



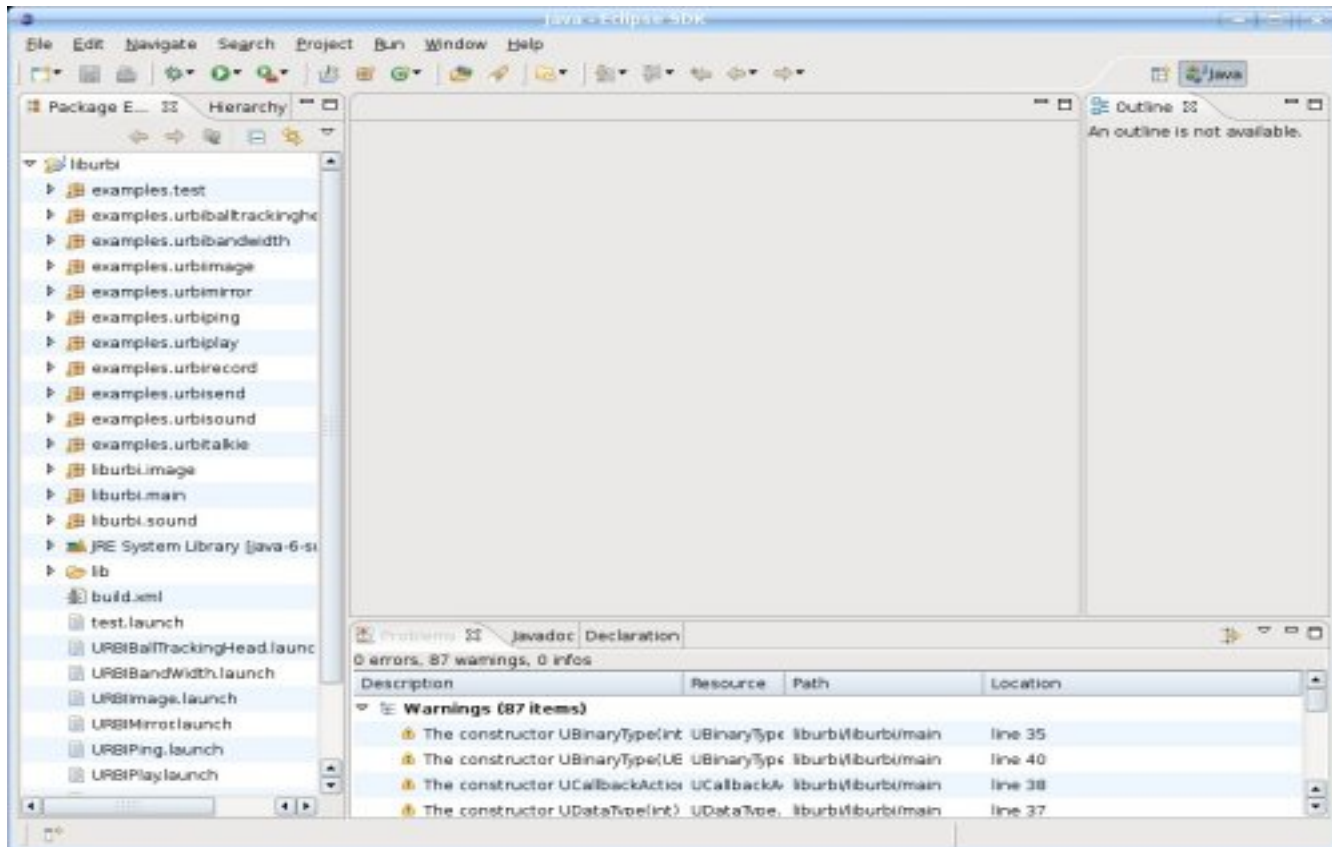
[images/select-import.jpg]

- Select 'Existing Projects into Workspace' in the opened windows
- Click 'Next'



[images/select-proj.jpg]

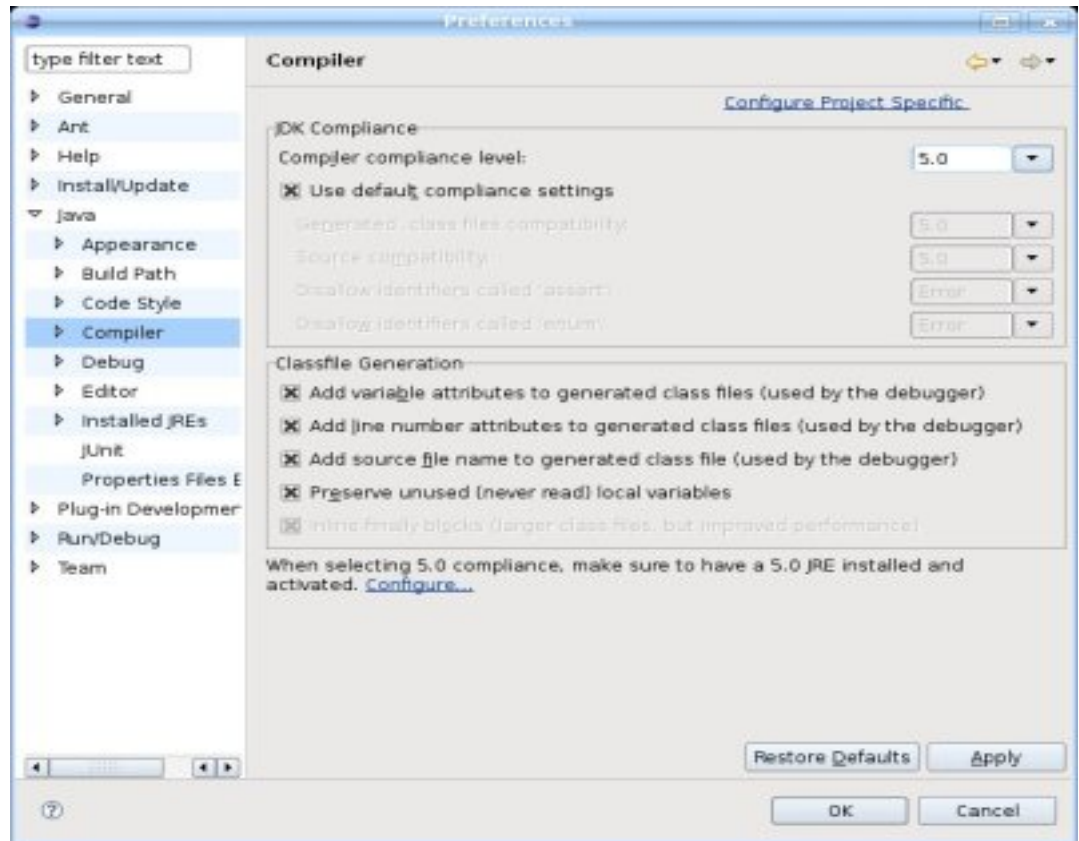
- Enter the path of the sdk-remote java on your computer
- Eclipse should find .project file in the provided and display the 'liburbi' project
- Select the 'liburbi' project and click 'Finish'



[images/project-liburbi-open.jpg]

The liburbi Java project is loaded. You can see the three packages of liburbi: liburbi.main, liburbi.image and liburbi.sound. From the source in these three packages, we made liburbijava.jar, that you can use in your applications. You can also see the sources of the different examples we provide. We put them in the packages examples.nameofexample. You can inspire yourself from these examples to make your own application. Here, we will see how to compile and run them in eclipse

NB: If Eclipse complains about errors in the source code, it can be that your compiler compliance level is too low. You have to set the compiler compliance level to Java 5 at least (Windows/Preferences/Java/Compiler).

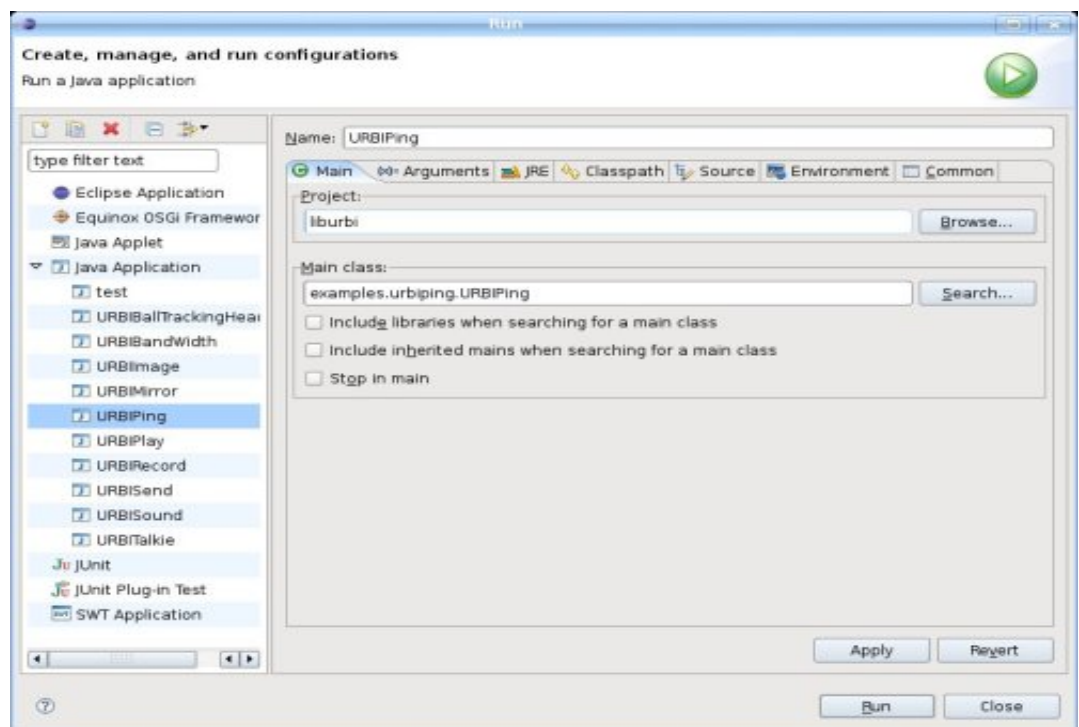


[images/compiler-compliance-level.jpg]

Run the examples

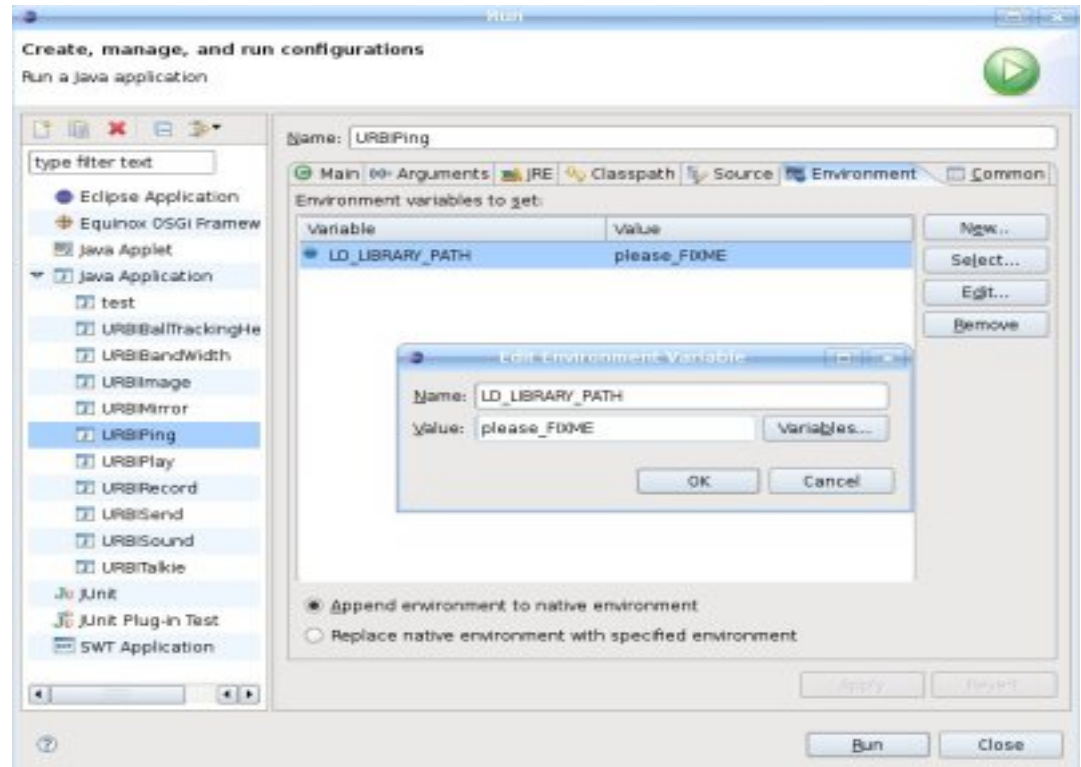
We provide sample .launch files that you can load in eclipse to run the projects.

- Click on Run/Open Run Dialog (or 'Run...' in some versions of Eclipse)



[images/run-urbiping.jpg]

- The launch configurations should be recognized automatically. You can choose one from the list, let's choose 'URBIPing'.
- In the right panel, select 'Environment', depending whether you are under Windows, Linux or MacOS, you will see the variable 'path', 'LD_LIBRARY_PATH' or 'DYLD_LIBRARY_PATH' defined.



[images/edit-ldlibpath.jpg]

- Edit the value of the variable. Set the absolute path of the liburbi java directory 'lib' (the one that contains the (lib)urbijava.{so,dll,dylib}). This is needed by the examples, in order to load the liburbijava library.
- Then click on the 'Arguments' tab, and enter the address of the urbi server you want to ping in the 'program arguments' field. For example if you want to ping the server running in Webots, set 'localhost' as address (without the quotes).
- Click 'Apply'. Click 'Run'.

Chapter 5. Programming hints

- Except if what you are doing is trivial, try not to use the `sync*` functions. They are less efficient than the asynchronous ones.
- The callback functions should return as fast as possible, since all callbacks are called by the same thread. If you have time-consuming operations, you should spawn an other thread and use synchronisation mechanisms such as semaphores or mutexes.

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, MERCHANTIBILITY, 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.