TUTORIAL: CALLING YARP FROM MATLAB (WINDOWS)

Prerequisites

First you need to have the following programs installed on your computer:

- Matlab
- CMake
- YARP

To install CMake, go to http://eris.liralab.it/wiki/CMake and follow the instructions.

To install YARP for Windows, go to http://eris.liralab.it/wiki/Getting_YARPed and follow the instructions. To download YARP as a source package, go to the following webpage instead of the one given on the wiki:

http://eris.liralab.it/yarp/specs/dox/download.html

You will then be able to get the latest version of YARP.

If you already have YARP on your computer, check if you find the ./example/swig folder in your YARP directory. You will need this in order to use YARP with Matlab, but in the oldest versions, the folder does not exist. If you don't find it, download and install a newer release of YARP.

Add to your Windows PATH (Control Panel -> System –> Advanced Tab -> Environment Variables to edit it) the path to the ./bin folder of your YARP directory.

Now you need to install the Java development environment and SWIG.

Installing Java on Windows

Download the latest release of the Java Development Kit (JDK) fromthe official Java webpage and install it:

http://java.sun.com/javase/downloads/index.jsp

Add to your Windows PATH the path to the ./bin folder ofyour Java JDK directory.

Installing SWIG on Windows

Download SWIG from http://www.swig.org/download.html.
You should get a ZIP folder: you just need to unpack it.

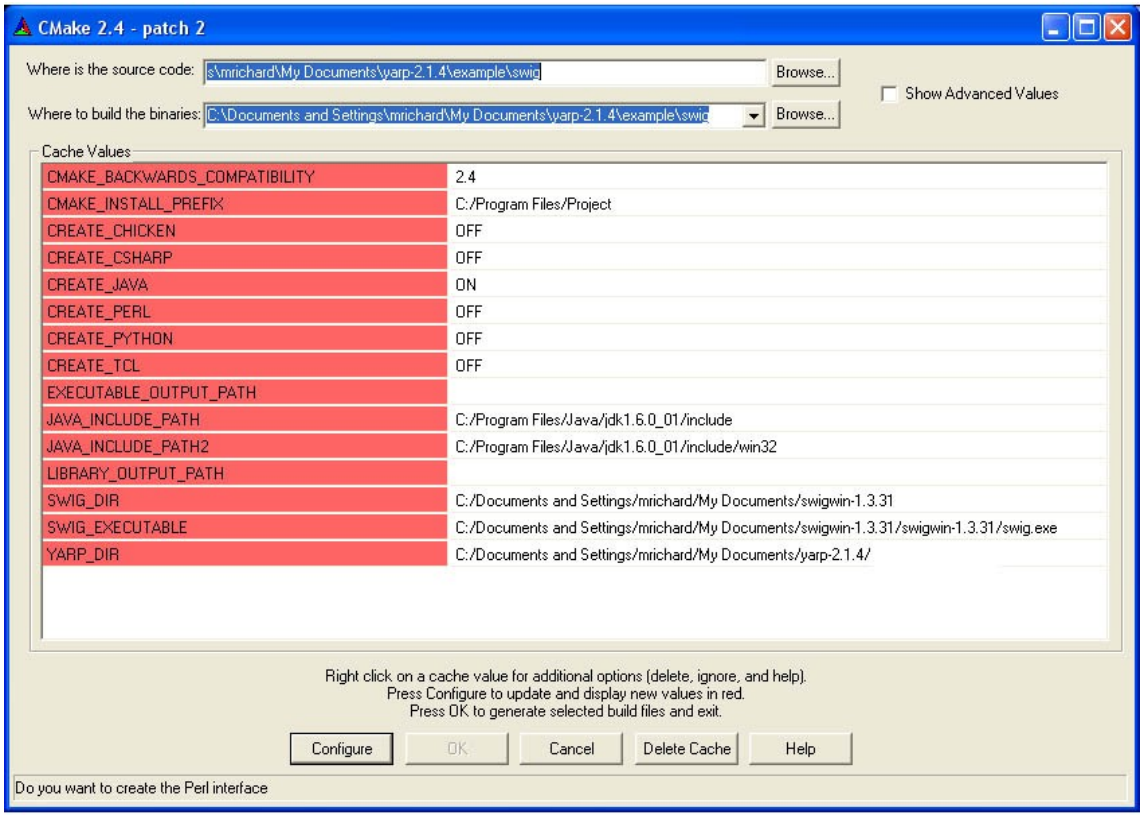
YARP side

Now you need to run CMake in the YARP folder ./example/swig.

In the CMake configuration window, you need to specify the following paths and options:

- enable the CREATE_JAVA option
- JAVA_INCLUDE_PATH is the path to your $JAVA_JDK_ROOT/include folder
- JAVA_INCLUDE_PATH2 is the path to your $JAVA_JDK_ROOT/include/win32 folder
- SWIG_DIR is the path to the unzipped SWIG folder
- SWIG_EXECUTABLE is the path to the *swig.exe* file
- YARP_DIR is the path to your YARP main directory.

You should end with a configuration similar to this one:



You should find a *project.sln* file in the ./example/swig folder. Open it with Visual Studio and build the BUILD_ALL solution (with the Release option if you use Visual Studio 8).

Now a folder named *generated_src* should have been created in the ./example/swig folder. Open it and check if it contains many .java files. You should also find a *release* folder in ./example/swig which must contain a shared library: *jyarp.dll*

If you don't find those folders, it means that the build operation has failed. Try to run again CMake (delete the cache and start again). When opening *project.sln*, do a "Make Clean" before building again the solution.

Now copy ./example/swig/src/*.java to ./example/swig/generated_src and compile all .java classes in /example/swig/generated_src. To do so, type the following command:

```
javac -source 1.3 -target 1.3 *.java
```

If the compilation is successful, no message appears in the terminal window and you should find .class files in the ./example/swig/generated_src folder.

MATLAB side

Find the Matlab *classpath.txt* and *librarypath.txt* files. You can ask Matlab about their location by typing

```
which classpath.txt
```

in the Matlab terminal. Both files are located in the same folder, usually *C:\Program Files\MATLAB\R2006b\toolbox\local*.

In both *classpath.txt* and *librarypath.txt* files, add the following lines:

```
yarp_root/example/swig/generated_src
yarp_root/example/swig/release
```

(yarp_root is the full path to the YARP directory).

The first line tells Matlab the location of the .class files, while the second points to the folder which contains the shared library *jyarp.dll*.

Now if you run Matlab and type the following line in the terminal,

```
LoadYarp;
```

you should get no error. If it does not work at the first try, try to copy the *jyarp.dll* file in a location contained in your Windows PATH. Every time you make a change, you must close Matlab and open it again in order to take into account the modifications.

Testing YARP with MATLAB

The last step is to test YARP with Matlab to check if the installation has been successful. To do so, we will use the .m files contained in the ./example/matlab folder of the YARP directory.

First open a Windows terminal and create a YARP server by typing:

```
yarp server
```

A message appears and tells you that the server has been created.

Now open two Matlab windows: place yourself in the ./example/matlab folder of the YARP directory. In one window, launch the *yarp_write.m* file and in the other, launch *yarp_read.m*. The first program creates a matlab/write YARP port, in which you can write, while the second creates a matlab/read port that can read data sent through YARP. You now need to connect them together.

To do so, open a second Window terminal and type:

```
yarp connect matlab/write matlab/read
```

You should get a message that tells you that YARP has linked an output of matlab/write to matlab/read.

Try to type a message in the Matlab window where you have launched *yarp_write.m.* In the other Matlab window, where you have launched *yarp_read.m*, you should get an echo of what you have typed. If you type 'quit' in the first window, both applications should terminate.

If it works, congratulations, you managed to install YARP for MATLAB !