

Eclipse

and

myRIO



in

C++

I. Necessary files

Eclipse

<http://www.ni.com/download/labview-real-time-module-2014/4846/en/>

C Support for myRIO

<http://www.ni.com/download/labview-myrio-toolkit-2018/7583/en/>

NiRIO Drivers

<http://www.ni.com/download/compactrio-device-drivers-january-2019/7833/en/>

Java

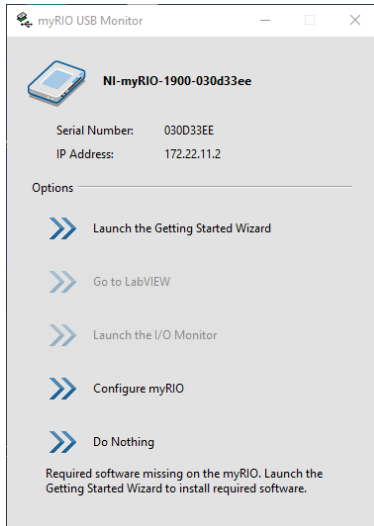
<https://www.java.com/fr/download/>




II. Configuration of myRIO



Install « NiRIO Driver » leaving the default installation options. Restart the computer when asked after the installation finishes.

When the installation is complete, connect your myRIO to your computer. The window below should appear.

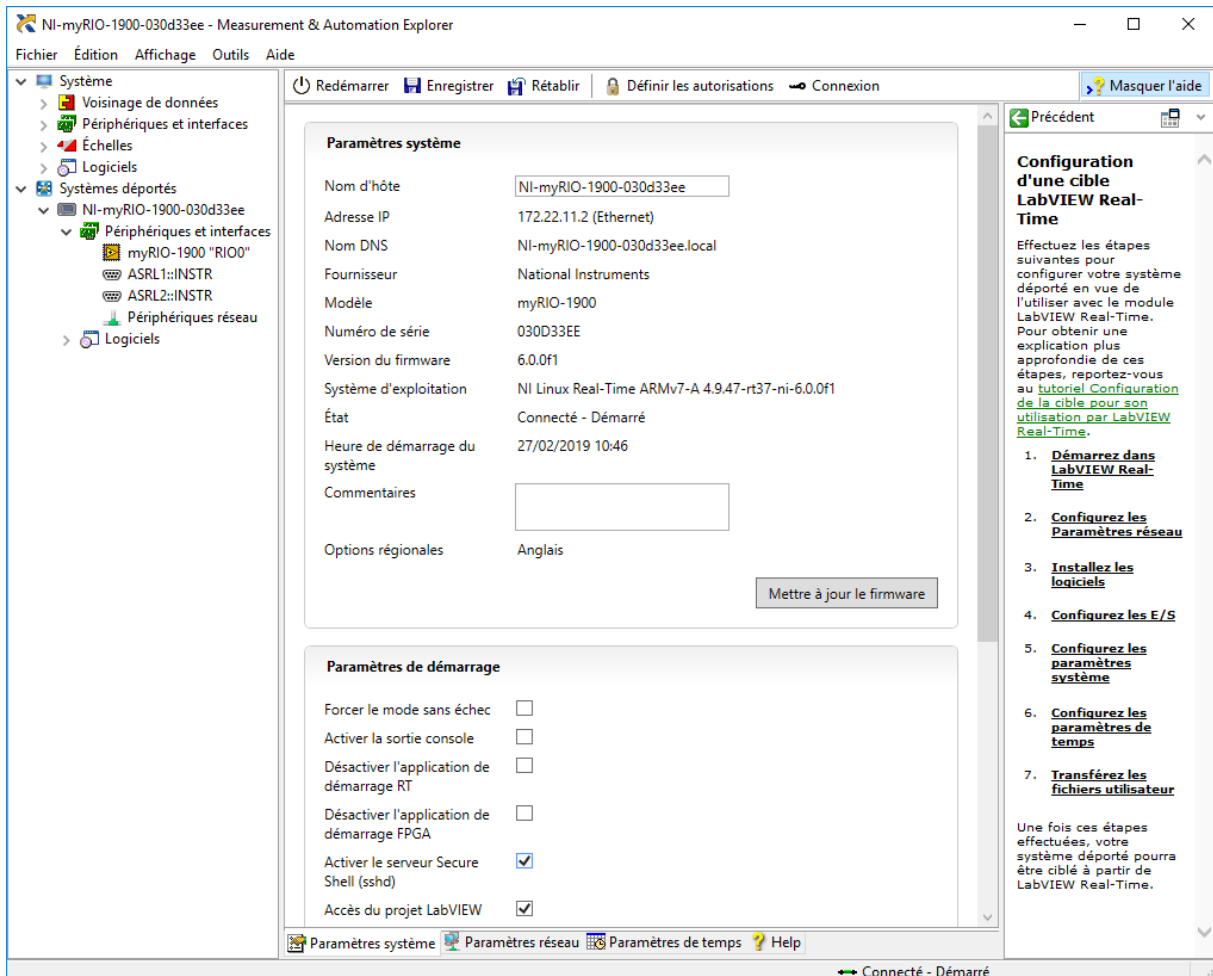
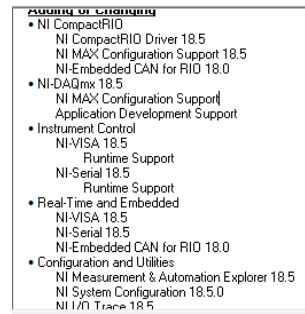


Choose « Launch the Getting Started Wizard » and follow the instructions. The firmware will be flashed and updated on your myRIO.

When the installation is finished, launch «  NI MAX » (a shortcut should be on the start menu).

Expand «  Remote systems » and wait until myRIO is detected. Select it, then check « Enable Secure Shell Server (sshd) » then «  Save ».

Close NI Max and myRIO USB Monitor.




III. Installation

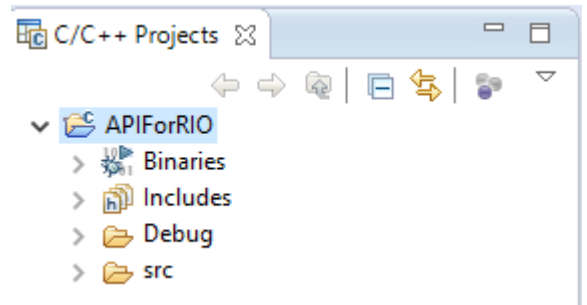
Install Java then Eclipse.

After installing, launch Eclipse (Start/National Instruments/C & C++ Development Tools for NI Linux Real-Time 2014, Eclipse Edition).




Close the « Welcome » window so you can access your project.

Create a new  C/C++ project, and name it, for example, « APIForRIO ».

Choose « Empty Project » and « Cross GCC ». Click « Next » two times.





Cross compiler prefix	arm-nilrt-linux-gnueabi-
Cross compiler path (On a x64 computer)	C:\Program Files (x86)\National Instruments\Eclipse\14.0\arm\sysroots\i686-nilrtsdk-mingw32\usr\bin\armv7a-vfp-neon-nilrt-linux-gnueabi

On your project create a folder named «  src »; inside it create a file named «  main.cpp » and lastly create another folder named «  CAPI » (so its path will be APIForRIO/src/CAPI).

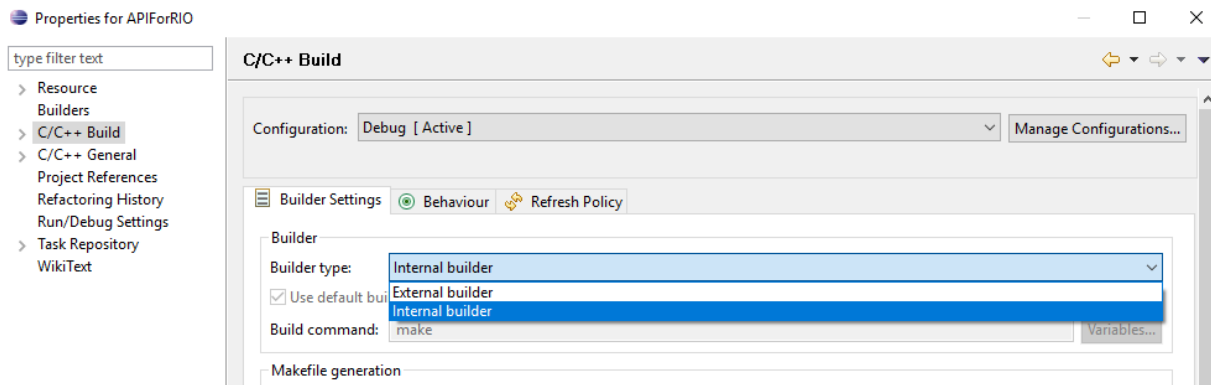
Extract « C_Support_for_myRIO_v6.0 » in a different directory.

Copy the files from « .../C Support/source/ » in the folder that you just created (CAPI).

Right click «  CAPI », then «  Refresh » to update the modifications that you just did to your workspace.


Right click on the name of the project, then « Properties ».

In « C/C++ Build » option, choose « Internal builder » instead of « External builder ».



Expand « C/C++ Build » then click on « Settings ».

- In the «  Cross GCC Compiler » section

In «  Symbols → Defined symbols (-D) »: Add a new symbol named « MyRio_1900 ».


- In the «  Miscellaneous » sections

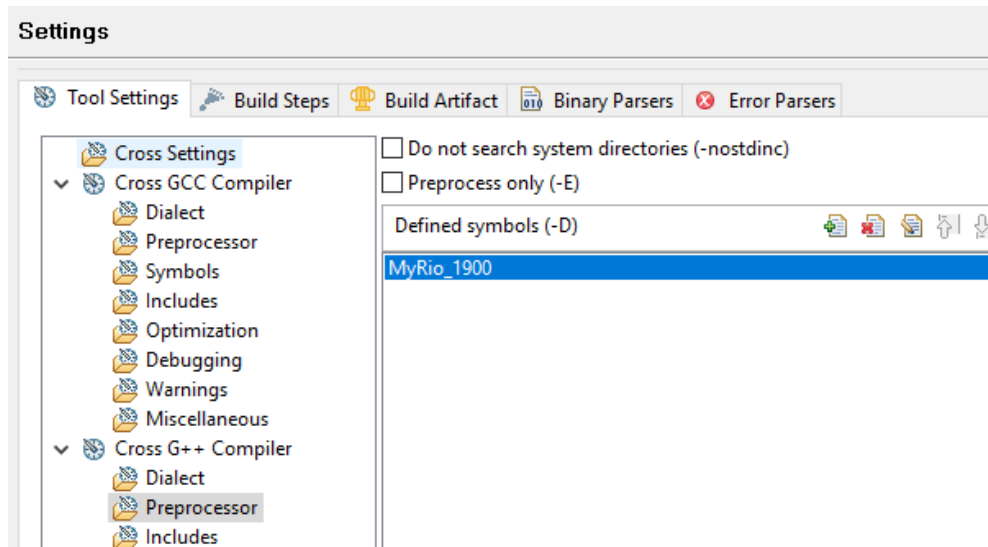
In the field « Other flags », add « -mfpu=vfpv3 -mfloat-abi=softfp » for more precision with floating numbers.

Other flags	-c -fmessage-length=0 -mfpu=vfpv3 -mfloat-abi=softfp
-------------	--



- In the «  Cross G++ Compiler » section

In «  Preprocessor »: add a symbol named « MyRio_1900 ».



- In the «  Miscellaneous » section

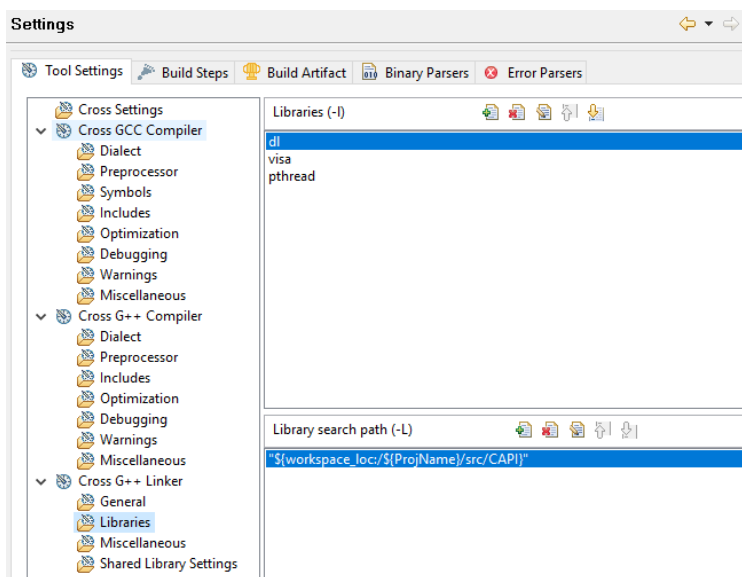
In the field « Other flags », add « -mfpu=vfpv3 -mfloat-abi=softfp » for more precision with floating numbers.

Other flags	-c -fmessage-length=0 -mfpu=vfpv3 -mfloat-abi=softfp
-------------	--

- In the «  Cross G++ Linker /  Libraries » section

Add the following libraries and path of libraries:

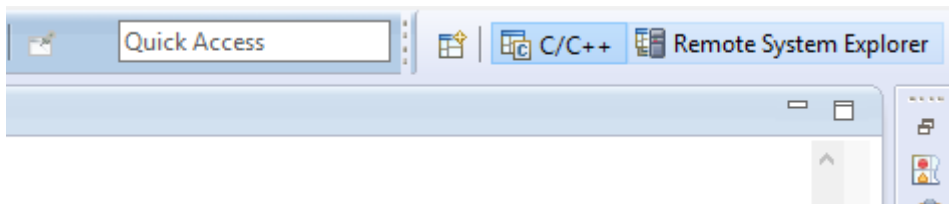
Libraries	dl
	visa
	pthread
Library search path	"\${workspace_loc}/\${ProjName}/src/CAP1"




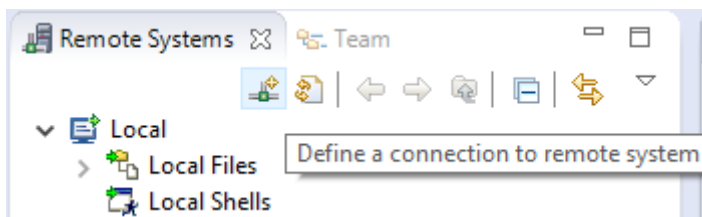
Make sure that your window matches this window and validate the new properties by clicking OK.


IV. Connection to myRIO

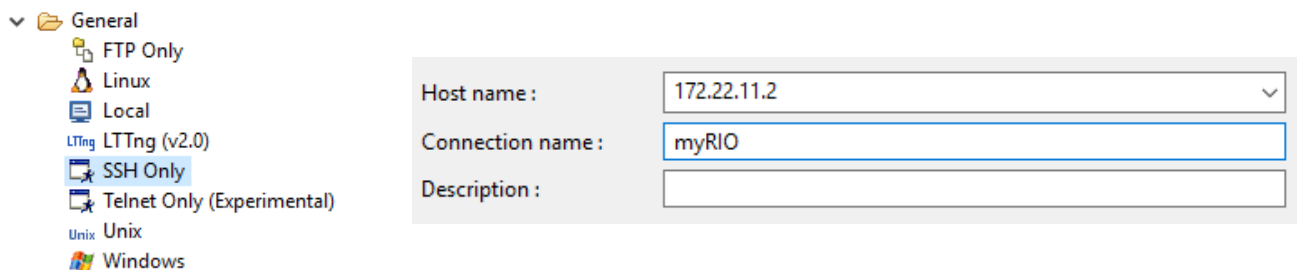
On Eclipse, choose «  Remote System Explorer »





Choose «  Define a connection to remote system »

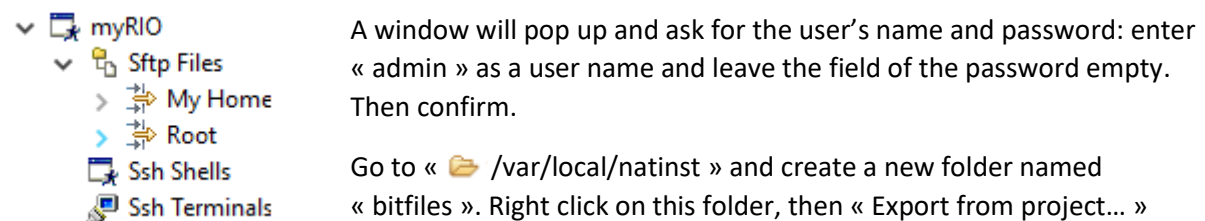



Click on «  SSH Only », then fill the fields « Host name » and « Connection name » with « 172.22.11.2 » and « myRIO » respectively.

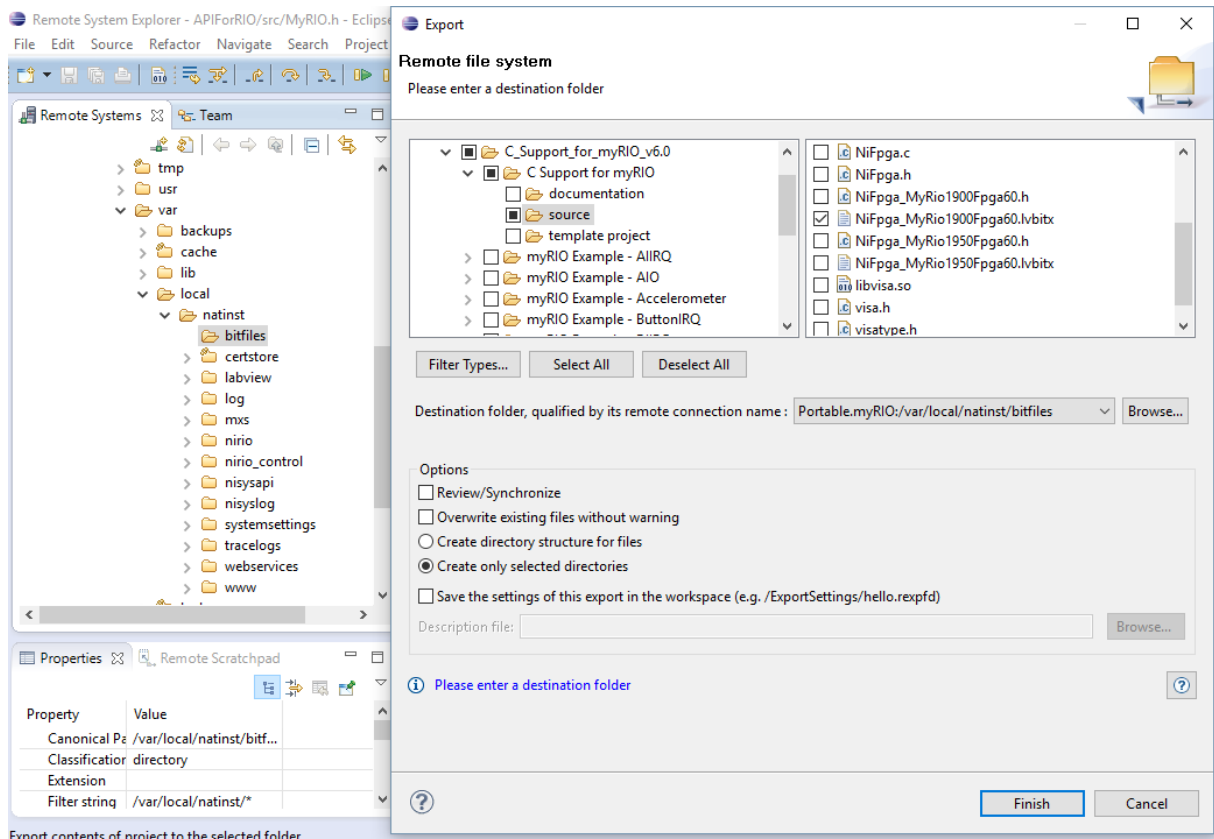


Then, click on « Next », « Next », « Next » then « Finish ».

A new connection appeared. Expand it, open «  Sftp Files » then «  Root ».



Go to the left window, into the « CAPI » folder. Check on the right window «  NiFpga_MyRio1900Fpga60.lvbitx » then click « Finish ».

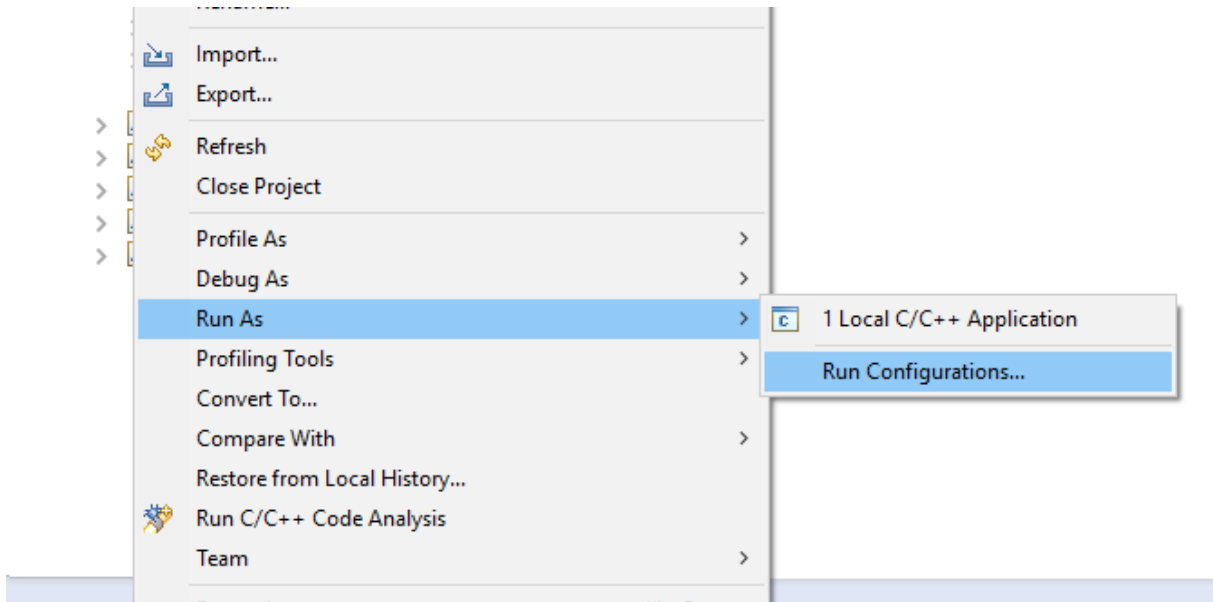


MyRIO is now configured to support interactions on C/C++ language.

To return to your project, click «  C/C++ » on the top right side.

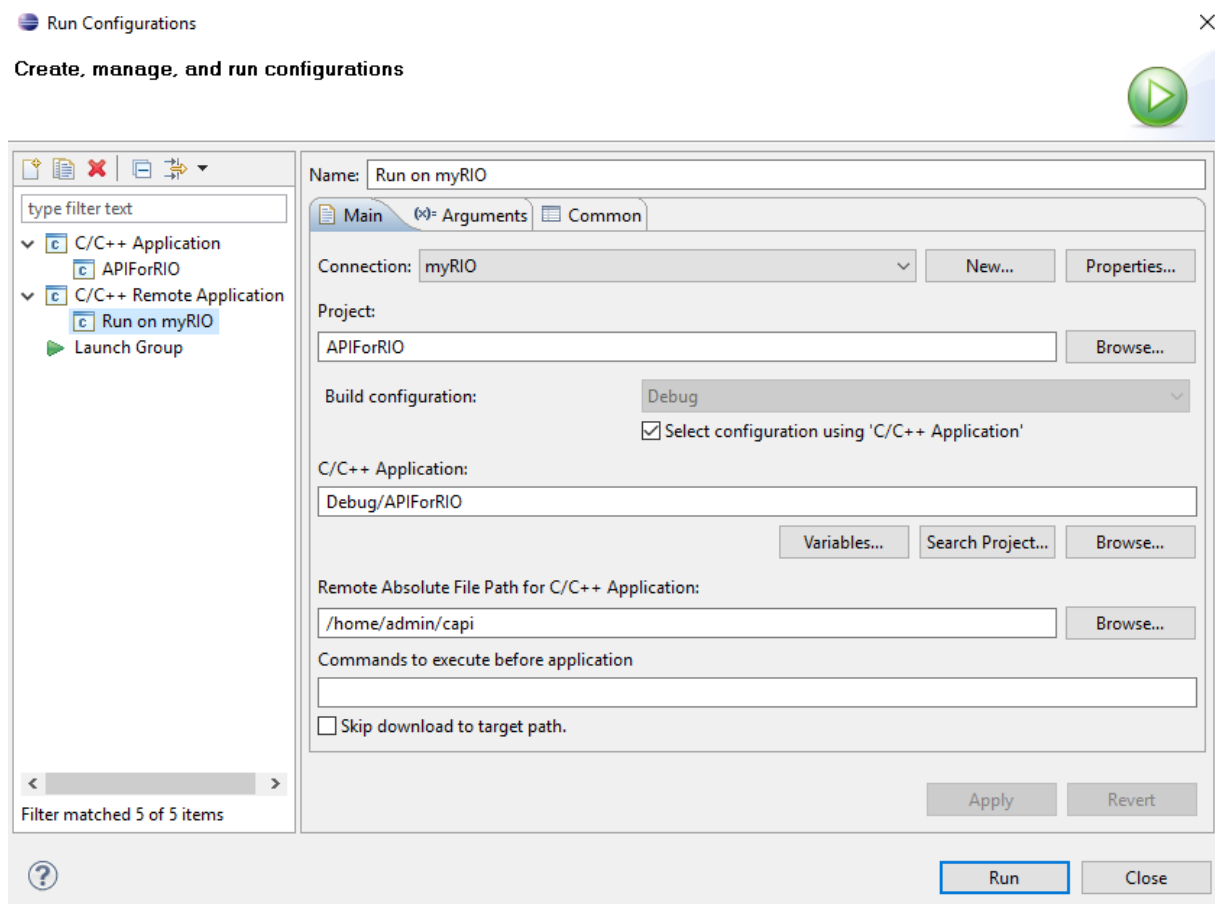
V. Configuration of the uploading

Right click on the project on Eclipse, then « Run As », then « Run configurations... »



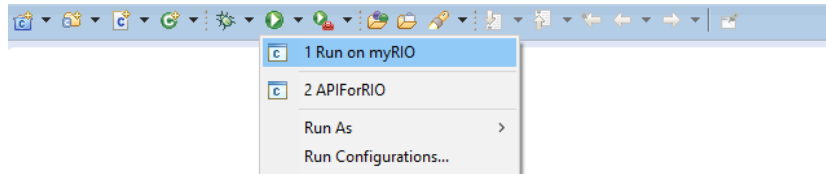
Right click on « Remote Application », then click on «  New ».

Name your configuration, choose the connection (myRIO previously connected), the project (APIForRIO), the path of the application to send and the path of the distant implantation. Verify the concordance with the following screenshot:

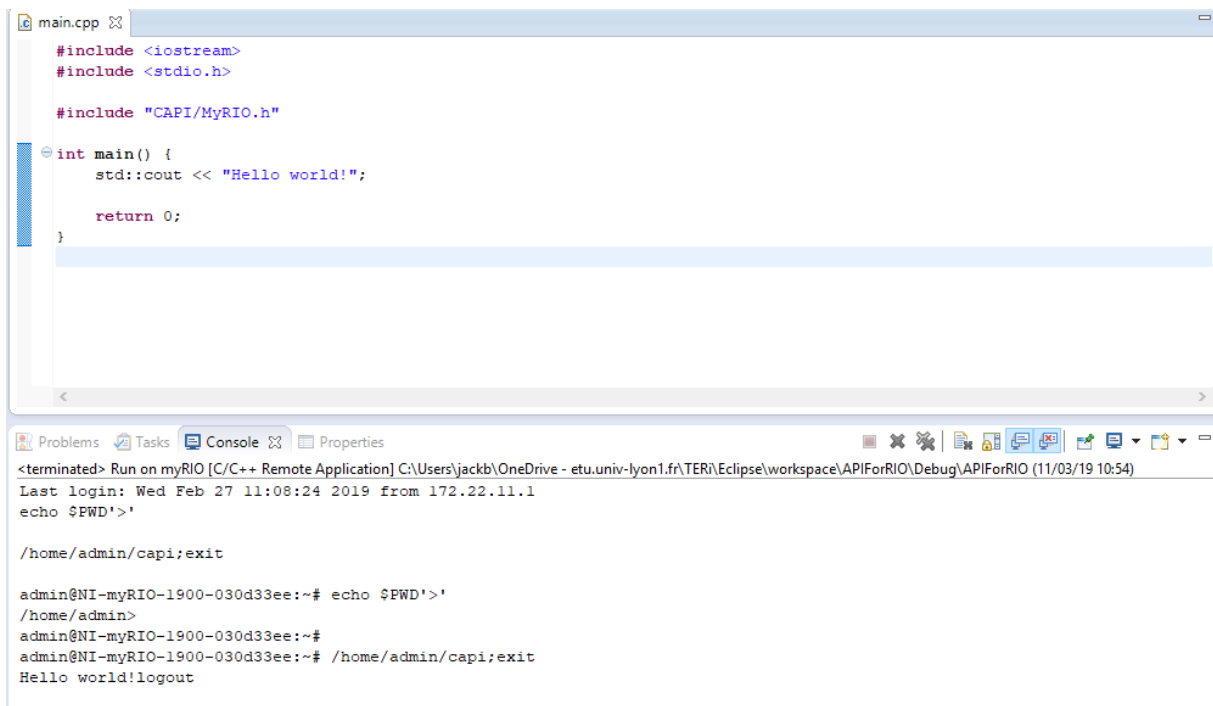


VI. Functional check

In the « `main.cpp` » file, `#include` « `CAPI/MyRIO.h` » then click on the name that you gave to your configuration, in our case « `Run on myRIO` ».



If the configuration is correct, the console should give the output of myRIO.

A screenshot of the Eclipse IDE showing the console output. The top part shows the source code of `main.cpp` with the following content:

```
#include <iostream>
#include <stdio.h>

#include "CAPI/MyRIO.h"

int main() {
    std::cout << "Hello world!";

    return 0;
}
```

The bottom part shows the console output, which includes the following text:

```
<terminated> Run on myRIO [C/C++ Remote Application] C:\Users\jackb\OneDrive - etu.univ-lyon1.fr\TERA\Eclipse\workspace\APIForRIO\Debug\APIForRIO (11/03/19 10:54)
Last login: Wed Feb 27 11:08:24 2019 from 172.22.11.1
echo $PWD'>'

/home/admin/capi;exit

admin@NI-myRIO-1900-030d33ee:~# echo $PWD'>'
/home/admin>
admin@NI-myRIO-1900-030d33ee:~#
admin@NI-myRIO-1900-030d33ee:~# /home/admin/capi;exit
Hello world!logout
```

To verify the good functioning of the registers, here is a little program to turn on the LEDs of myRIO:

```
#include <iostream>
#include <stdio.h>

#include "CAPI/MyRIO.h"

extern NiFpga_Session myrio_session;

int main() {
    // ouvre une session myRIO
    NiFpga_Status status = MyRio_Open();
    if(MyRio_IsNotSuccess(status))
        std::cout << "MyRio not created : " << status << std::endl;

    // sélectionne les DigitalOutput LED 3:0, et envoie 0b1111 pour toutes les allumer
    NiFpga_MergeStatus(&status,
        NiFpga_WriteU8(myrio_session, DOLED30, 15));
    // attend un arrêt forcé
    while(true);
}
```





VII. Useful links to go further

- **Default FPGA personalities**

http://zone.ni.com/reference/en-XX/help/373925B-01/myriohelp/myrio_fpga_personalities/

Documentation (associated registers...):

<http://www.ni.com/product-documentation/14655/en/>

myRIO Shipping Personality 6.0 Reference

This document contains reference information about the myRIO shipping personality.

Contents

Introduction	4
Register Naming Convention	4
Peripheral Type	5
Channel Name	5
Property Names	6
System Control / Function Select	8
Host Synchronization Registers (SYS.x.RDY)	8
Function Select Registers (SYS.SELECTx)	9
myRIO Expansion Ports (MXP)	9
Mini Systems Port (MSP)	10
Onboard Device Registers	11
LEDs (DO.LED3:0)	11
Button (DI.BTN)	11
Accelerometer Value Registers (ACC.x.VAL)	12
AI/AO/Audio	13

- **Extension board documentation**

<https://learn.ni.com/teach/resources/808/robot-builder-s-guide-pitsco-tetrix-prime-for-ni-myrio>

Used ports descriptions (motors, gyroscope...) from page 38 to page 44.

Motor Number (noted on motor board)	Pin Name (from left to right based on the image above)	Wire color (if using provided DC motor)	MXP Pin number	Name in software (based on MXP A)
1	Encoder B	Purple	22	A/ENC.B
1	Encoder A	Blue	18	A/ENC.A
1	Encoder Ground	Green	n/a	n/a
1	Encoder VCC	Brown	n/a	n/a
1	Motor +	Red	27 – PWM speed control, 15 – DIO direction control	A/PWM0 A/DIO2
	Motor -	Black		
2	Encoder B	Purple	22 (on opposite MXP port)	B/ENC.B
2	Encoder A	Blue	18 (on opposite MXP port)	B/ENC.A
2	Encoder Ground	Green	n/a	n/a

