

# 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

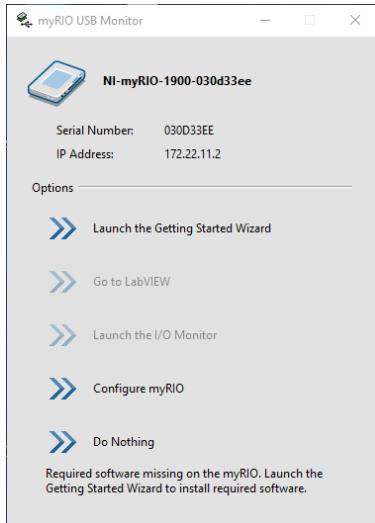
<https://www.java.com/fr/download/>



## 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.

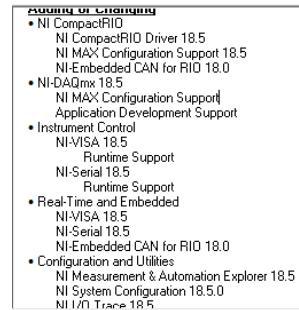


Choose « Launch the Getting Started Wizard » and follow the instructions. The firmware will be flashed and updated on your myRIO.

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.



The screenshot shows the 'NI MAX - Measurement & Automation Explorer' application window. The left sidebar shows a tree structure with nodes like 'Système', 'Voisinage de données', 'Pérophériques et interfaces', 'Échelles', 'Logiciels', 'Systèmes déportés', and a specific entry for 'NI-myRIO-1900-030d33ee'. The main panel has tabs for 'Paramètres système', 'Paramètres réseau', and 'Paramètres de temps'. The 'Paramètres système' tab is currently active, displaying various system parameters such as 'Nom d'hôte' (NI-myRIO-1900-030d33ee), 'Adresse IP' (172.22.11.2 (Ethernet)), 'Nom DNS' (NI-myRIO-1900-030d33ee.local), 'Fournisseur' (National Instruments), 'Modèle' (myRIO-1900), 'Numéro de série' (030D33EE), 'Version du firmware' (6.0.0f1), 'Système d'exploitation' (NI Linux Real-Time ARMv7-A 4.9.47-rt37-ni-6.0.0f1), 'État' (Connecté - Démarré), 'Heure de démarrage du système' (27/02/2019 10:46), and 'Commentaires'. The 'Paramètres de démarrage' tab shows checkboxes for 'Forcer le mode sans échec', 'Activer la sortie console', 'Désactiver l'application de démarrage RT', 'Désactiver l'application de démarrage FPGA', 'Activer le serveur Secure Shell (sshd)' (which is checked), and 'Accès du projet LabVIEW'. A sidebar on the right provides a step-by-step guide for configuring the target, starting with 'Démarrez dans LabVIEW Real-Time', followed by 'Configurez les paramètres réseau', 'Installez les logiciels', 'Configurez les E/S', 'Configurez les paramètres système', 'Configurez les paramètres de temps', and 'Transférez les fichiers utilisateur'. It also includes a note about enabling the target for LabVIEW Real-Time.



### 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<br>(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).

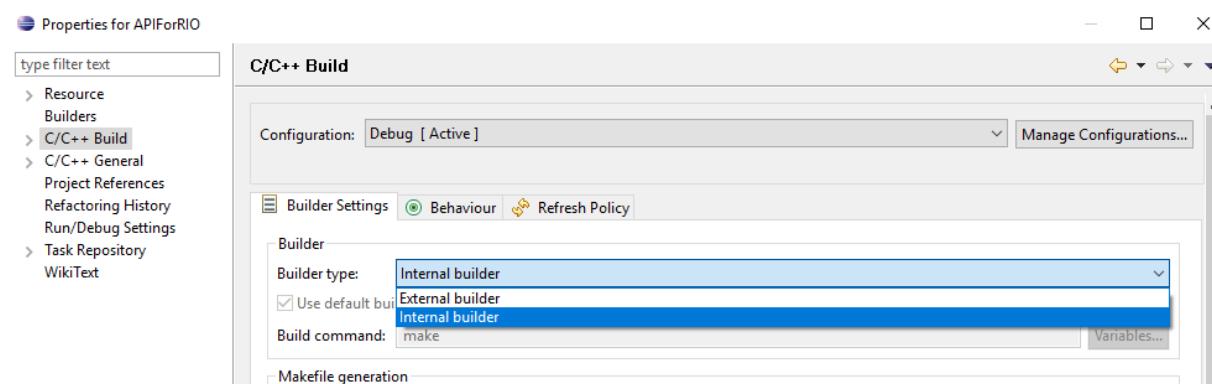
Extract « C\_Support\_for\_myRIO\_v6.0 » in a different directory.

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 ».

The screenshot shows the 'Settings' window with the 'Tool Settings' tab selected. On the left, a tree view shows 'Cross Settings' expanded, with 'Cross GCC Compiler' and 'Cross G++ Compiler' collapsed. Under 'Cross G++ Compiler', 'Preprocessor' is selected, and the 'Defined symbols (-D)' section contains the entry '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 | <code>-c -fmessage-length=0 -mfpu=vfpv3 -mfloat-abi=softfp</code> |
|-------------|---|

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

Add the following libraries and path of libraries:

|                     |  |
|---------------------|--|
| Libraries           | <code>dl</code><br><code>visa</code><br><code>pthread</code> |
| Library search path | <code>"\${workspace_loc:/\${ProjName}/src/CAPI}"</code>      |

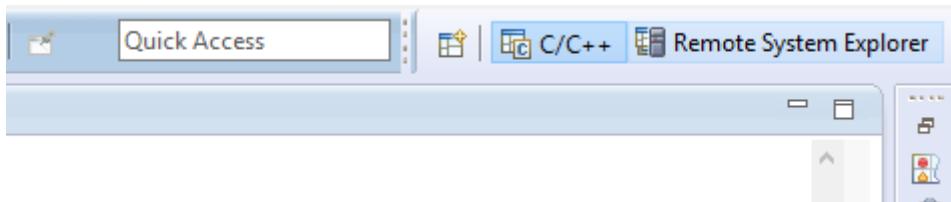
The screenshot shows the 'Settings' window with the 'Tool Settings' tab selected. On the left, a tree view shows 'Cross Settings' expanded, with 'Cross GCC Compiler' and 'Cross G++ Compiler' collapsed. Under 'Cross G++ Compiler', 'Libraries' is selected. The 'Libraries (-l)' field contains 'dl', 'visa', and 'pthread'. Below it, the 'Library search path (-L)' field contains the value '`"${workspace_loc:/${ProjName}/src/CAPI}"`'.

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

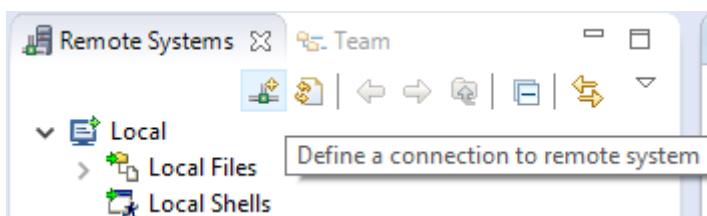


## IV. Connection to myRIO

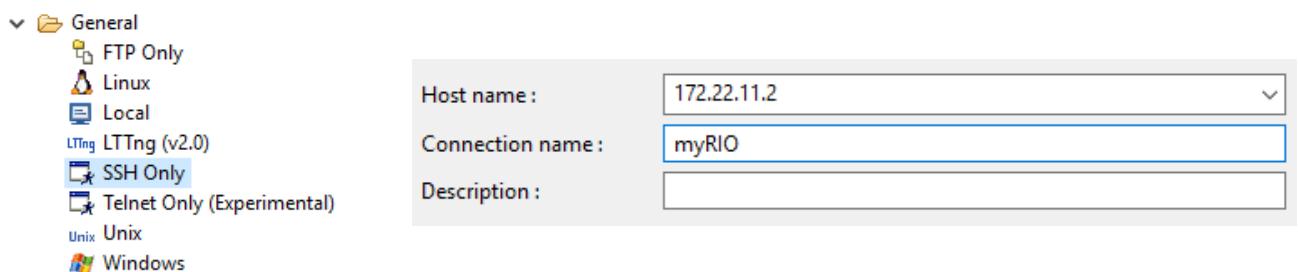
On Eclipse, choose «  Remote System Explorer »



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 ».

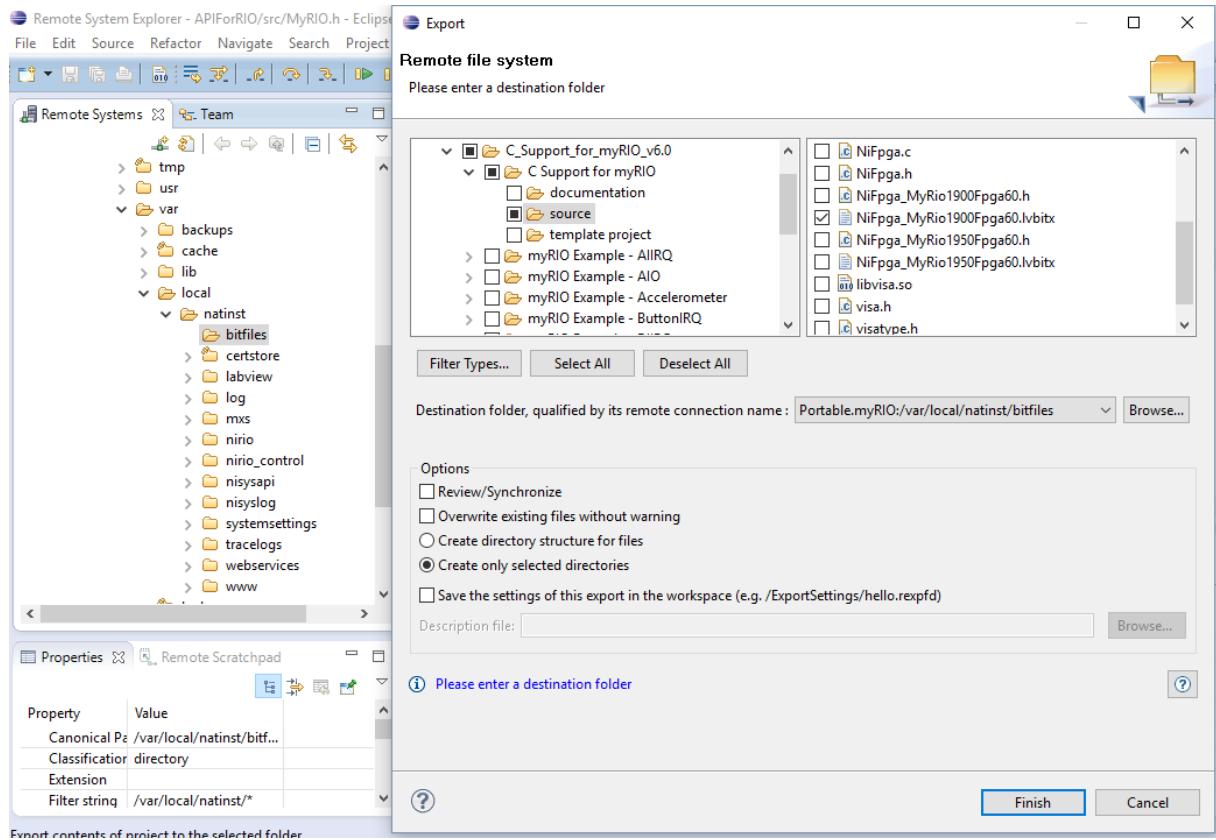
A new connection appeared. Expand it, open «  Sftp Files » then «  Root ».

|   |  |
|---|--|
| ▼  myRIO       | A window will pop up and ask for the user's name and password: enter   |
| ▼  Sftp Files  | « admin » as a user name and leave the field of the password empty.  |
| >  My Home     | Then confirm.  |
| >  Root        |  |
|  Ssh Shells    | Go to «  /var/local/natinst » and create a new folder named |
|  Ssh Terminals | « bitfiles ». Right click on this folder, then « Export from project... »  |

Go to the left window, into the « CAPI » folder. Check on the right window

«  NiFpga\_MyRio1900Fpga60.lvbitx » then click « Finish ».





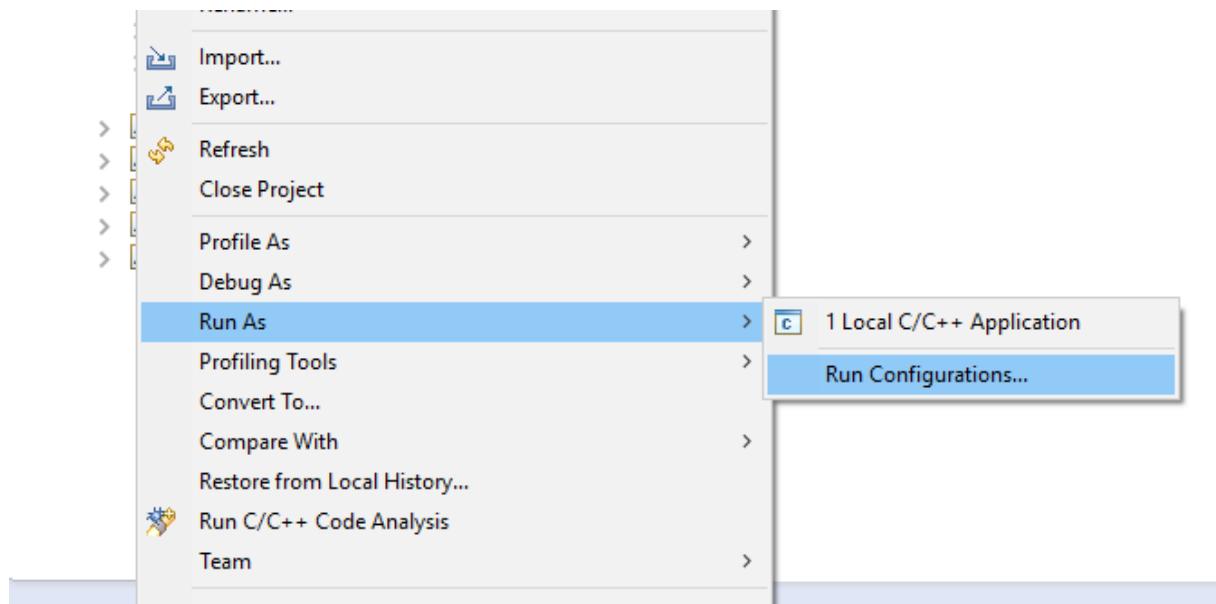
MyRIO is now configured to support interactions on C/C++ language.

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



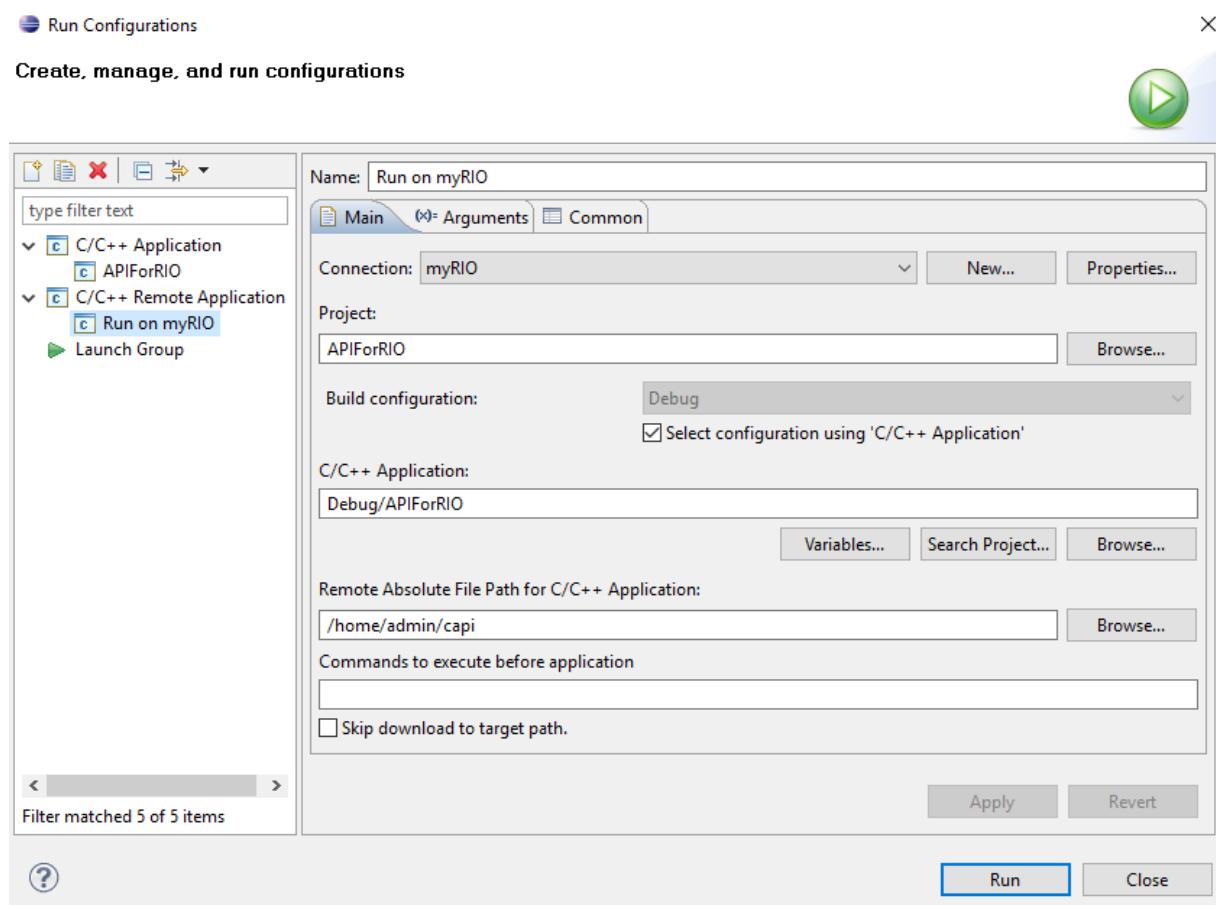
## V. Configuration of the uploading

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



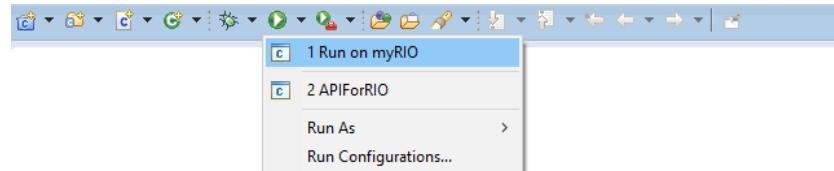
Right click on « Remote Application », then click on « New ».

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 « `.cpp` 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' view. The terminal window displays the output of the 'Hello world!' program. It shows the command 'echo \$PWD' being run, followed by the path '/home/admin/capi'. The program then outputs 'Hello world!' and logs out. The terminal window has tabs for 'Problems', 'Tasks', 'Console', and 'Properties'. The status bar at the bottom indicates the session is 'terminated'.

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/](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<br>(noted on motor<br>board) | Pin Name (from left<br>to right based on the<br>image above) | Wire color (if using<br>provided DC motor) | MXP Pin number  | Name in software<br>(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<br>control,<br>15 – DIO direction<br>control | A/PWM0<br>A/DIO2                     |
|   | Motor -  | Black                                      |   |                                      |
| 2   | Encoder B  | Purple                                     | 22 (on opposite MXP<br>port)                                | B/ENC.B                              |
| 2   | Encoder A  | Blue                                       | 18 (on opposite MXP<br>port)                                | B/ENC.A                              |
| 2   | Encoder Ground   | Green                                      | n/a   | n/a                                  |

