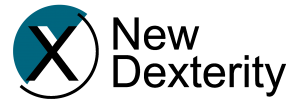
Test Rig Manual





**Table of Contents**

[1. Introduction 1](#_Toc73721589)

[2. Hardware Setup 2](#_Toc73721590)

[2.1 Overview 2](#_Toc73721591)

[2.2 myRIO 2](#_Toc73721592)

[2.2.1 MXP Port A Pinouts 4](#_Toc73721593)

[2.2.2 MXP Port B Pinouts 4](#_Toc73721594)

[2.2.3 MSP Connector Pinouts (Port C) 5](#_Toc73721595)

[2.3 Load Cell Amplifier 6](#_Toc73721596)

[2.4 Current Sensor 6](#_Toc73721597)

[2.5 Voltage Divider Circuit (Voltmeter) 7](#_Toc73721598)

[2.6 Light Sensor Circuit 7](#_Toc73721599)

[3. Software 8](#_Toc73721600)

[3.1 Overview 8](#_Toc73721601)

[3.2 Setup 8](#_Toc73721602)

[3.3 Main Program 8](#_Toc73721603)

# Introduction

This test rig manual covers the hardware setup (including the wiring) and the usage section of the test rig.

# Hardware Setup

## 2.1 Overview

Looking at Figure 1, we can see that the PC connects to three myRIO’s and the Dynamixel motors. These hardware are connected to the PC via USB connections. Within the myRIO, the PC accesses the myRIO’s FPGA boards for readout values for each component connected to the myRIO (except PWM out, where the PC sends an output instead). The left and right motors are powered using an external power supply.

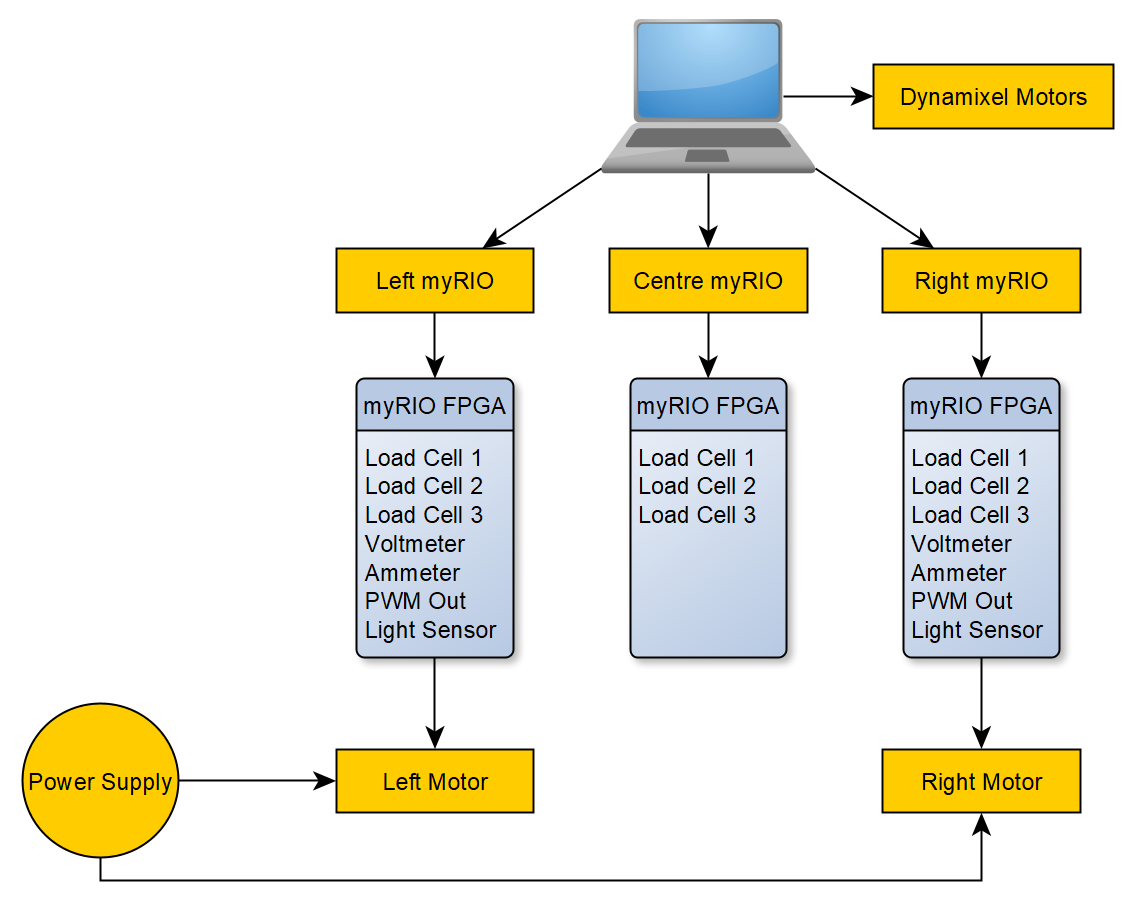


Figure Current Rig Setup

## 2.2 myRIO

The NI myRIO is a microcontroller and an FPGA board all in one package. For the test rig, we do not really need the microcontroller, as the PC will do all the processing work, but the FPGA board is what is useful. The FPGA board provides a proper real-time interface to the hardware that the PC cannot do especially when its operating system is Windows. The myRIO has two MXP connectors shown in Figure 2 and one MSP connector shown in Figure 3. The rest of this subsection will detail the pinouts for the myRIO

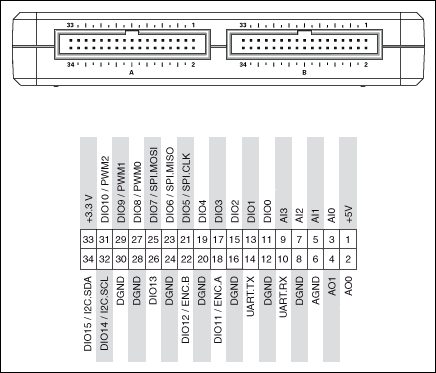


Figure myRIO MXP connectors and their pinouts

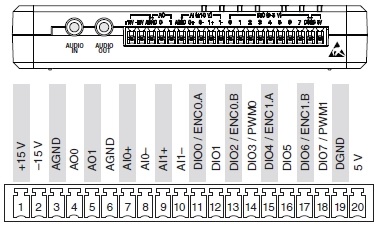


Figure myRIO MSP connector and its pinout

### 2.2.1 MXP Port A Pinouts

The following table provides a mapping for the myRIO to components. Please note pins 3, 6, 27 and 28 are only used for the left and right myRIO’s. Except for the wires used in pins 27 and 28 which use female-to-male wires, all other wires use female-to-female wires.

Table MXP Connector Port A Pinouts

|  |  |  |  |
| --- | --- | --- | --- |
| **From** | **To** | **Colour** | **Function** |
| Pin 1 | VCC of Load Cell Amplifiers (via parallel branch) | Red | Provides power to the amplifiers |
| Pin 8 | GND of Load Cell Amplifiers (via parallel branch) | Black | Provides the reference line to the amplifiers |
| Pin 11 | DAT of Load Cell 1 | Blue | Provides bit by bit data on this line |
| Pin 13 | CLK of Load Cell 1 | White | Provides the sync clock in which to receive the data |
| Pin 15 | DAT of Load Cell 2 | Blue | Provides bit by bit data on this line |
| Pin 17 | CLK of Load Cell 2 | White | Provides the sync clock in which to receive the data |
| Pin 19 | DAT of Load Cell 3 | Blue | Provides bit by bit data on this line |
| Pin 21 | CLK of Load Cell 3 | White | Provides the sync clock in which to receive the data |
| Pin 33 | VDD of Load Cell 1 (via parallel branch) | Yellow | Provides power line for measurement |
| Pin 3 | IOut of KR Sense 90A | Green | Measures the current |
| Pin 6 | GND of KR Sense 90A | Black | Provides a reference line |
| Pin 27 | Signal line of motor PWM input | Yellow | Sends a PWM signal to the motor |
| Pin 28 | Ground line of motor PWM input | Black | Sends a reference line with the PWM signal to the motor |

### 2.2.2 MXP Port B Pinouts

The following table provides a mapping for the myRIO to components. Please note that this port is only used for the left and right myRIO’s. The connection for pin 6 uses a female-to-female wire. Pins 3 and 6 use a female-to-multiple-female wires.

Table MXP Connector Port B Pinouts

|  |  |  |  |
| --- | --- | --- | --- |
| **From** | **To** | **Colour** | **Function** |
| Pin 1 | 5V lines of light sensor circuit | Yellow | Provides power to the light sensor |
| Pin 3 | Non-ground end of the measurement resistor | Green | Measures the voltage across the measurement resistor of the light sensor |
| Pin 6 | Ground lines of light sensor circuit | White | Provides a reference line to the light sensor |

### 2.2.3 MSP Connector Pinouts (Port C)

The following table provides a mapping for the myRIO to components. Please note that this port is only used for the left and right myRIO’s. The wires that are used to connect pins 6 and 7 to the voltage divider circuit just use plain wires. Also note that there is a resistor that is used to connect Pin 6 and Pin 8 together. If Pin 6 and Pin 8 aren’t connected by a resistor, then there will be a significant drift in the voltmeter readings.

|  |  |  |  |
| --- | --- | --- | --- |
| **From** | **To** | **Colour** | **Function** |
| Pin 6 (AGND) | Ground end of the measurement resistor in the voltage divider circuit | Black | Provides a reference line to the light sensor |
| Pin 7 (AI0+) | Non-ground end of the measurement resistor in the voltage divider circuit | Yellow | Provides a scaled down voltage measurement |
| Pin 8 (AI0-) | Pin 6 (AGND) using a 10k resistor | 10k resistor | Ensures that the AIO- end is virtually earthed. 10k resistor is used to dissipate any charge buildup that occurs due to a floating pin. |

## 2.3 Load Cell Amplifier

As of writing this manual, the load cell amplifier used was the HX-711. The wiring to the myRIO can be found in 2.2.1 MXP Port A Pinouts. The wiring to the load cell should be self-explanatory as the colour of the wires from the load cell amplifier should match that of the HX-711’s colour pinouts, except that the yellow pin is not used. See Figure 4 as a reference.

If using more HX-711 amplifiers, one should solder male pin headers onto the board, and connect with the myRIO using female-to-female wires.

If using more load cells, one should note the order of the colour of the pins of the HX-711 and arrange the load cell wires accordingly. Crimp a female pin header off the wires.

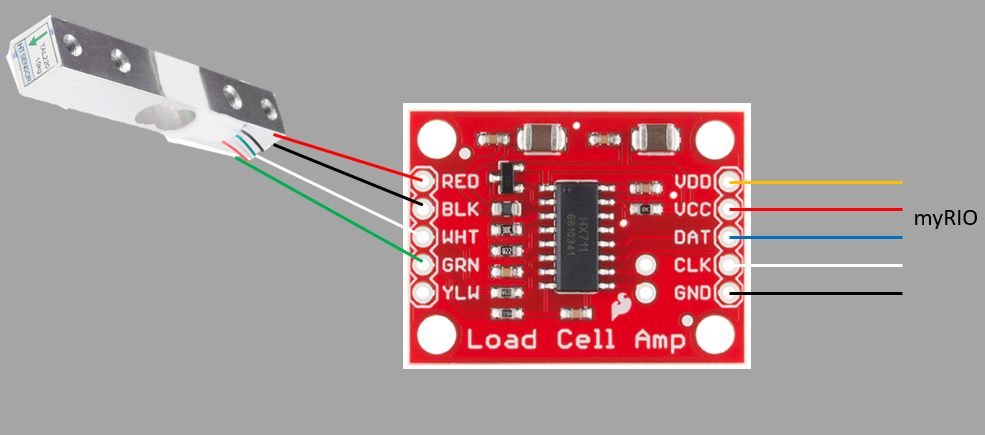


Figure HX-711 wiring

## 2.4 Current Sensor

The current sensor we use is the SEN-16408 (KR Sense Current and Voltage Sensor – 90A). The wiring can be found in Figure 5. Please note that the +IN must go to the power supply and the +OUT must go to the motor, as the direction of current is important for this sensor and only reads in the direction defined by the three arrows seen on the chip. The Iout and ground pins are to have male pin headers and connected to the myRIO by female-female wires.

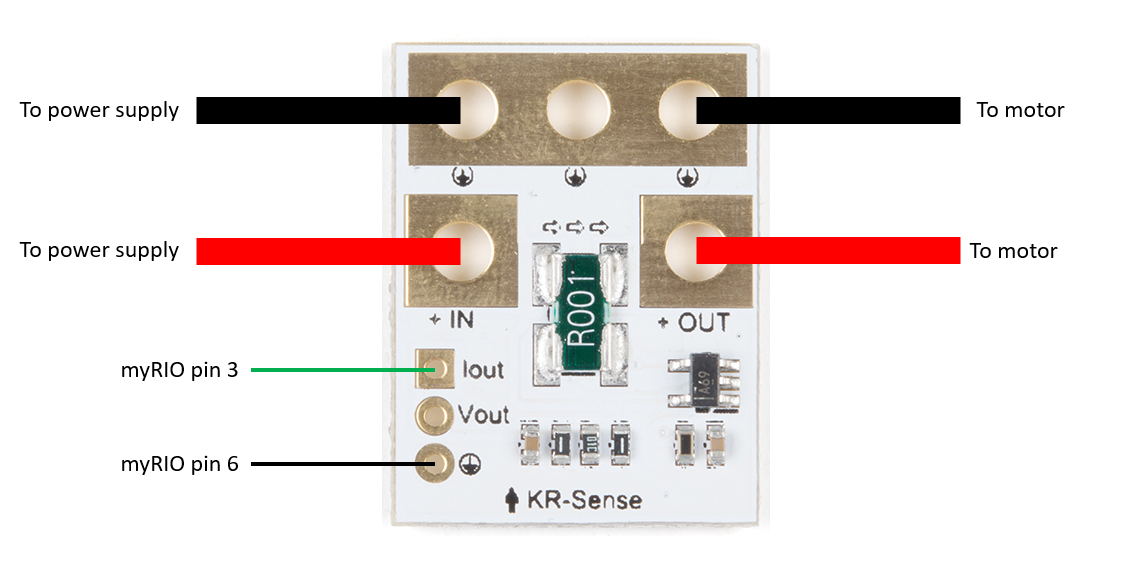


Figure Current sensor circuit wiring.

## 2.5 Voltage Divider Circuit (Voltmeter)

For the voltage measurement system, a voltage divider is used as the voltage ranges that are required to measure the voltage across the motors are not in the range of the myRIO’s voltage ranges of 0-10V (technically -10V to 10V but requires a double differential circuit).

The working principle of the voltage divider circuit is simple. You first connect two resistors between the positive end of the voltage to measure and the ground, and then connect the myRIO’s positive input pin between the two resistors and the same ground of the voltage to measure. This is shown in Figure 6.

In terms of selecting the resistances for the resistors, the sum of the resistances of R1 and R2 should be large enough so that this circuit draws minimal current from the source (the MΩ range is generally good). The resistance of R2 should be lower than that of R1. However, if R2 is far too low, then the voltage range will be very large but at the same time, the resolution (step-size) of the voltage read will also be large as well due to the myRIO’s ADC. So ideally, the resistances should be chosen to closely fit the voltage range of Vin. The equation for calculating how many times the Vin you are scaling down to is defined as the following:

As an example, if R1 = 5 MΩ and R2 = 1 MΩ, then Vin will be 6 times Vout. Therefore, the range of Vin can be from 0V to 60V.

As of writing R1 ~ 13.6 MΩ using two 6.8 MΩ resistors and R2 ~ 1.64 MΩ using two 0.82 MΩ resistors. This means that the range of Vin can be from 0V to approximately 93V.

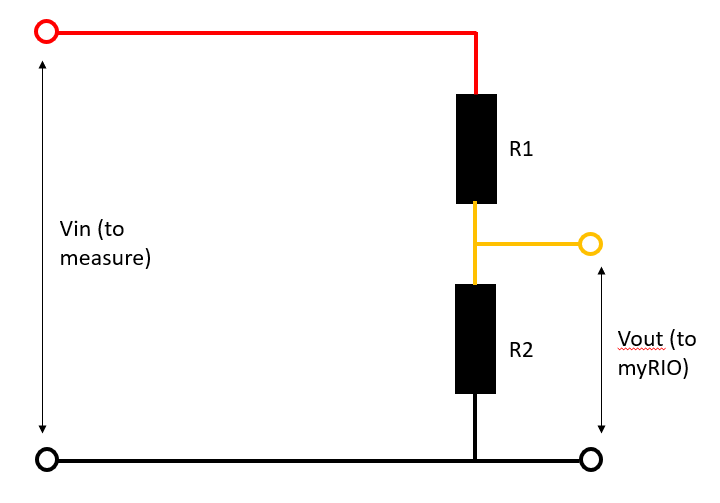


Figure Voltage divider circuit

## 2.6 Light Sensor Circuit

The light sensor is used to measure the speed of the motors, by applying a small piece reflective tape on the rotors of the motors, and getting the light sensor to detect every time this tape passes the light sensor as the rotor of the motor is rotating.

The light sensor uses an LED to emit light and a collector-emitter circuit to collect the light emitted by the LED via reflection. If there is enough light reflected, then the collector-emitter turns on and allows current to flow through it, and as a result, the resistor connected to the emitter end has a voltage across it, and this voltage is measured by the myRIO which then measures when the voltage was dropped across it, and determines the speed of the motor based on the time intervals of each time the voltage goes high.

The light sensor used for the test rig is the BPR-301-ND, and the circuit diagram can be shown in Figure 7. As the power supplied from the myRIO is at 5V (with the ground also connected to the myRIO), a 56Ω and 15Ω resistor was used in series to create an equivalent resistance of 71Ω (which is closest to 70Ω) for the LED. This is because the LED, when turned on, requires 50mA of current flowing through it. When there is 50mA of current flowing through it, there is 1.5V dropped across it. Given the myRIO can supply 5V, this means a resistor needs to drop 3.5V as well as keep 50mA of current to allow this current to flow through the diode. Therefore, a 70Ω or so resistance is required, thus a 56Ω and 15Ω was used. Although there is a single 68Ω resistor available, it was safer to choose 71Ω as it is better to have slight under current than over current in this case.

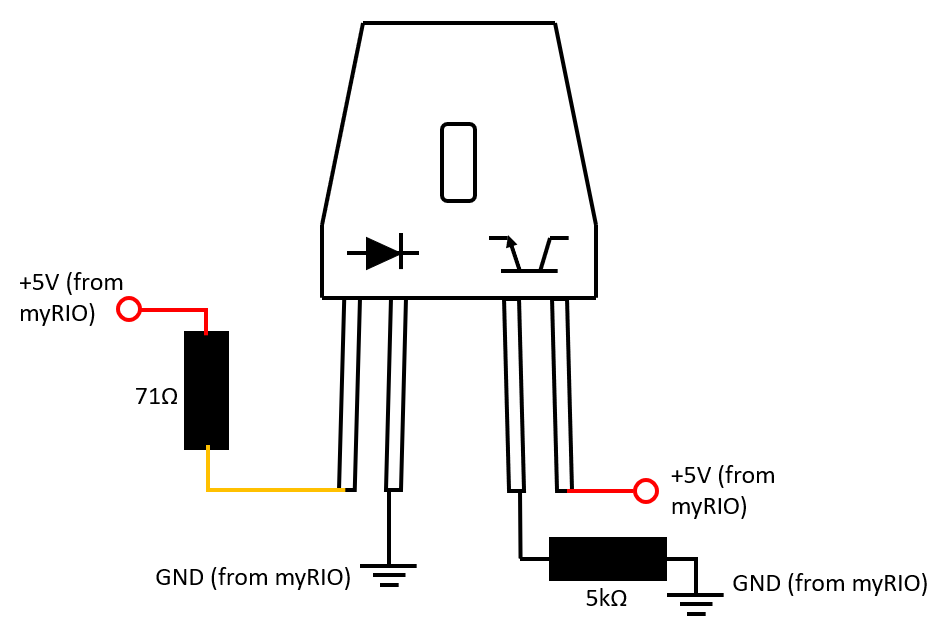


Figure Light sensor circuit, the left two pins are for the LED and the right two pins are for the collector-emitter junction. Of the right two pins, the emitter pin is on the left and the collector is on the right.

In terms of the collector-emitter junction, it also uses the +5V power supply from the myRIO (with the ground also connected to the myRIO). The +5V is connected to the collector (the right-most pin), and the ground and the emitter resistor is connected to the emitter (the immediate left of the right-most pin where the arrow points to). The resistor was placed on the emitter (common collector configuration) rather than on the (common emitter configuration). A common collector configuration means that the voltage is 0V when there is no reflection and around 5V when there is a reflection (some voltage will be dropped across the junction), and this makes it easier to code in the myRIO FPGA board. Refer to this link for more details: <https://www.digikey.com/en/articles/how-to-use-photodiodes-and-phototransistors-most-effectively>. 5kΩ was used as the emitter resistor, as this will minimise the current drawn from the junction (approx. 1mA is drawn).

In terms of the placement of the photodiode circuit, ensure that the photodiode is aimed at the reflective tape approximately 5mm away.

# Software

## 3.1 Overview

This section will cover the software section of the rig. The setup section will go into the details particularly with the software required to setup the main program as well as drivers to install. The main program section will go into the details of using the software.

## 3.2 Setup

### 3.2.1 Installation

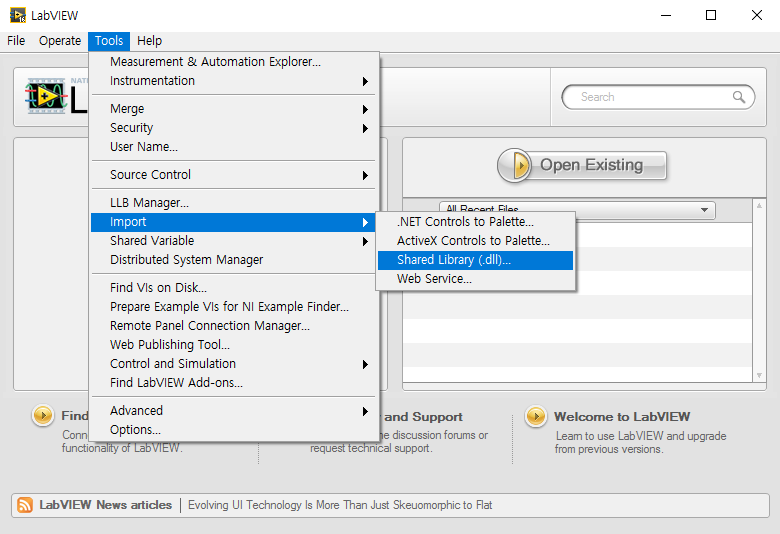
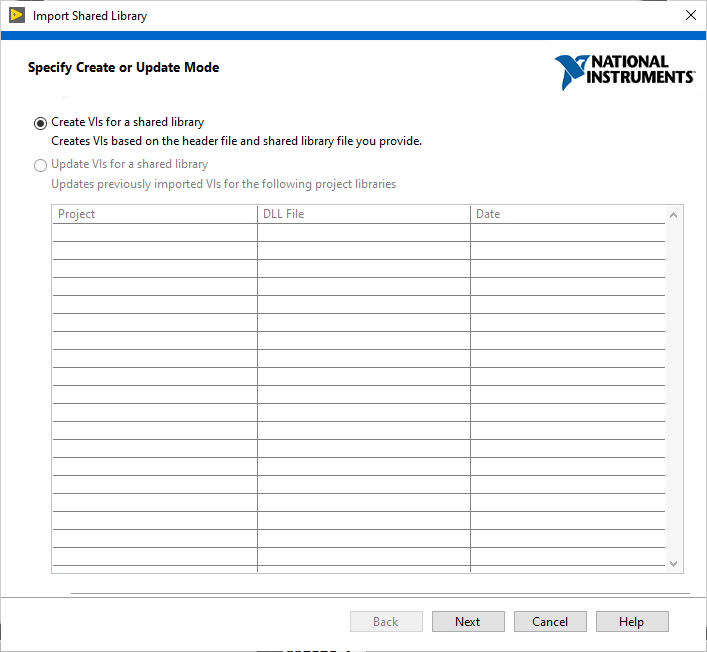
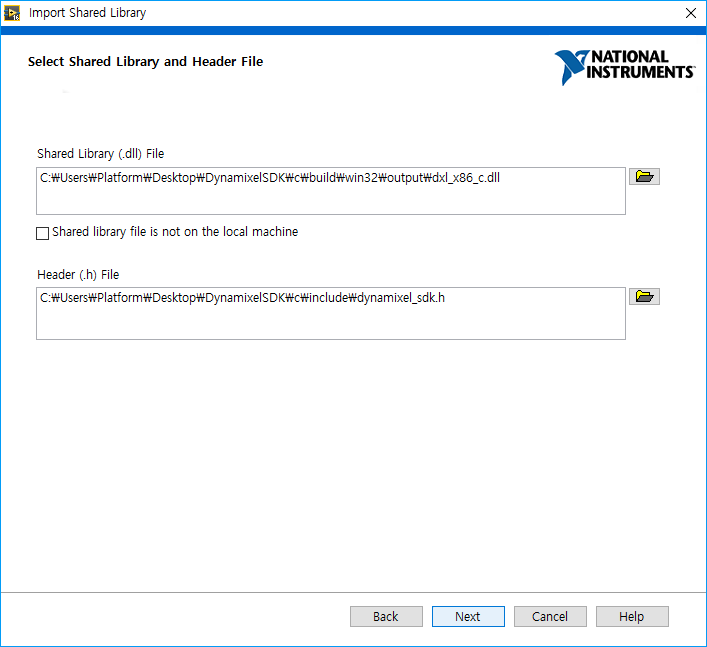
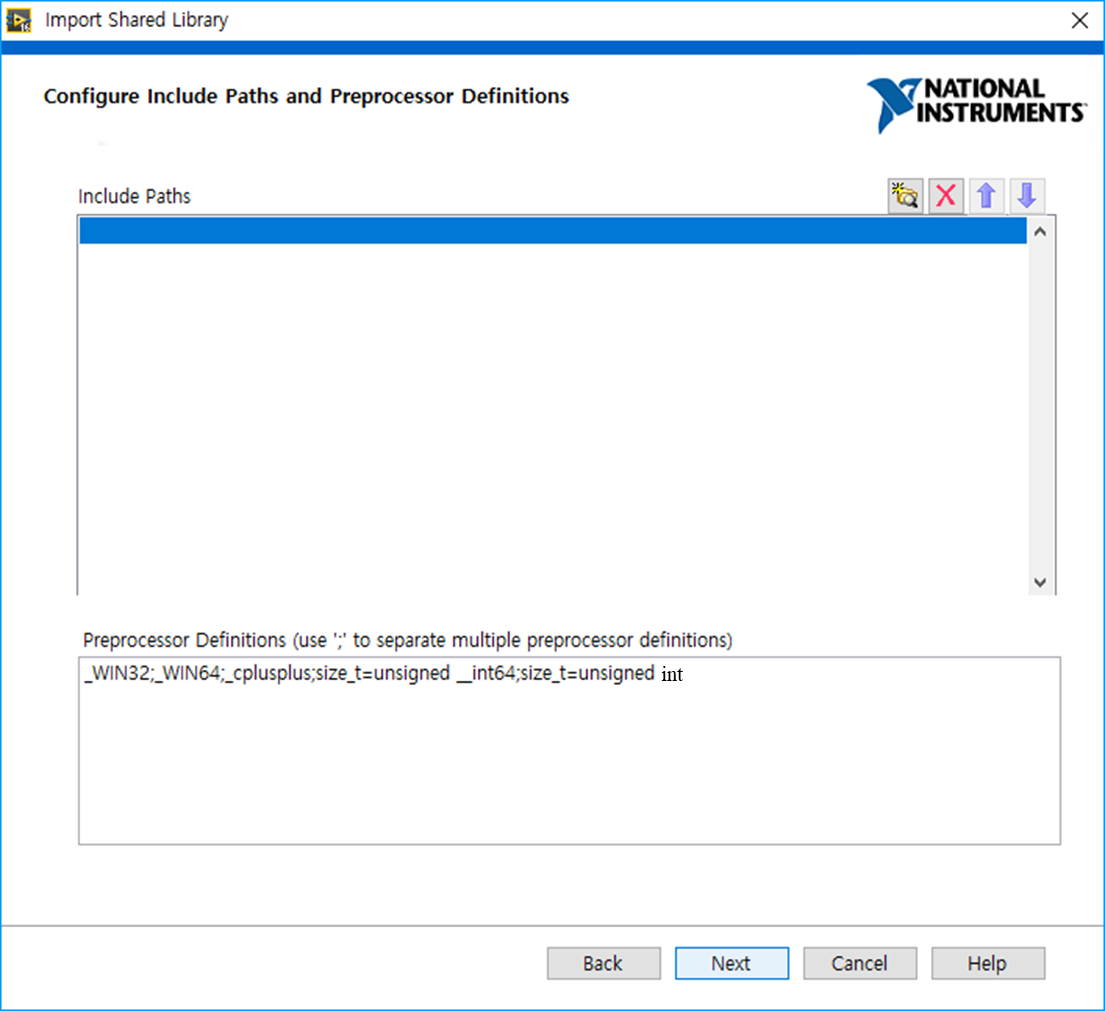
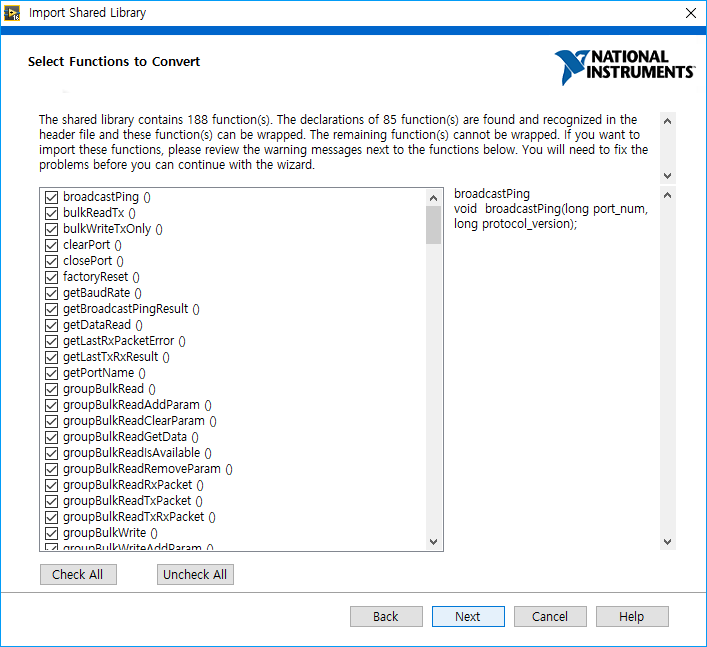
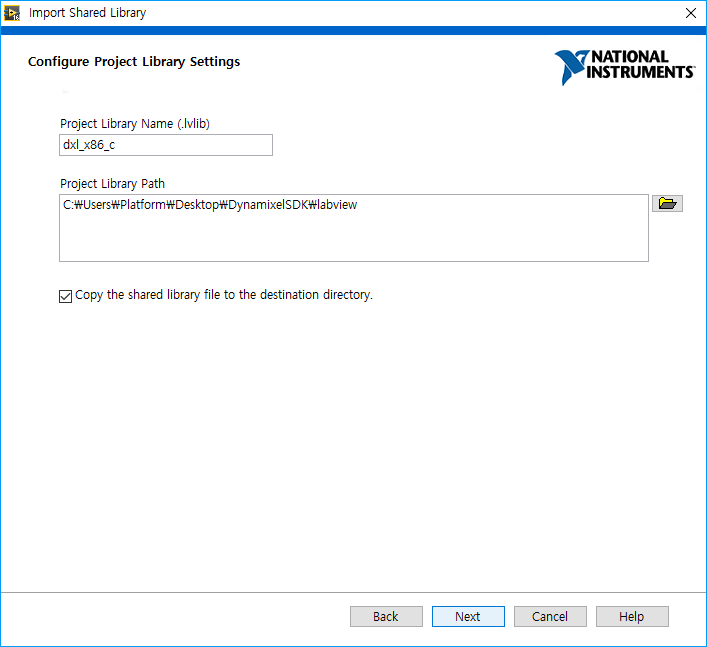
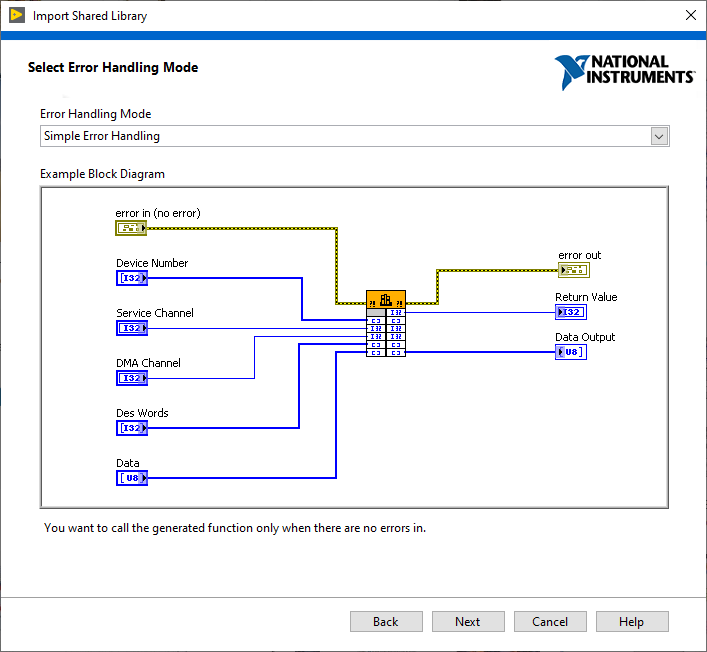
The following list shows the programs that will need to be installed:

* LabVIEW 2018 myRIO software bundle (ask UoA IT to install)
  + LabVIEW English (base/full/professional) 32-bit
  + myRIO toolkit
  + Real-time module
  + VI package manager
  + FPGA module
  + Compilation tool for Vivado 2017.2
  + (the other add-ons are optional)
* Visual Studio Express or better (latest version should be fine, and can be downloaded for free from Microsoft)
* USB FTDI driver (can be found in DynamixelDriversAndWizard -> CDM21236\_Setup.zip)
* Dynamixel Wizard (can be found in DynamixelDriversAndWizard -> DynamixelWizard2Setup\_x86.exe)

The Visual Studio Express is optional and only needed when the libraries found in the DynamixelSDK-3.7.31 folder cannot be added to LabVIEW due to operating system mismatch.

### 3.2.2 Importing Dynamixel libraries into LabVIEW

In order to use Dynamixel motors from LabVIEW, the libraries must be imported into LabVIEW. DO NOT USE the “labview” folder found in the DynamixelSDK-3.7.31 folder as that will be the folder LabVIEW will generate the libraries to, instead, you use the “c” folder. To do so, follow the list of steps with the unlabelled figures used in each step to help the user to install the libraries:

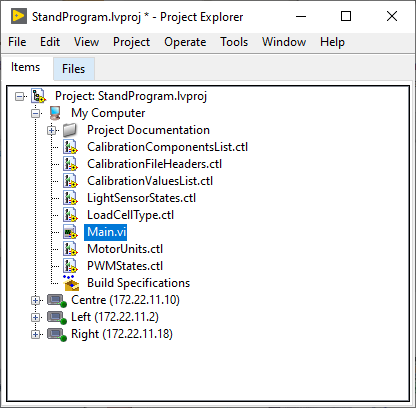
1. Open LabVIEW. On the menu bar, go to Tools -> Import -> Shared Library (.dll)... 
2. Select “Create VIs for a shared library” then click Next 
3. Here you’ll be asked to load the paths of the .dll file and the .h file. The path to the .dll file is “[path\_to\_joaoandthethreestands]\joaoandthethreestands\DynamixelSDK-3.7.31\c\build\win32\output\dxl\_x86\_c.dll”. The path to the .h file is “[path\_to\_joaoandthethreestands]\joaoandthethreestands\DynamixelSDK-3.7.31\c\include\dynamixel\_sdk\dynamixel\_sdk.h”. Please note that regardless of whether you have a 32-bit or a 64-bit Windows OS, you must use win32 as LabVIEW uses 32-bit architecture (at least at the time of writing these instructions). Click Next.
4. In this page you provide include paths and preprocessor definitions. The include paths box is left empty. The preprocessor definitions should have the blue text (excluding the quotation marks) copied and pasted “\_WIN32;\_WIN64;\_cplusplus;size\_t=unsigned \_\_int64;size\_t=unsigned int”. Click Next. 
5. This page should show a list of functions to convert into VI’s. Note that not all the functions from the Dynamixel C libraries can be wrapped into VI’s. So it will show in the text that there are 188 or so functions in the library, but 85 or so can only be wrapped into LabVIEW, this is due to the fact that there are more functions that can be wrapped for LabVIEW, but is not needed for the ones used in this rig. Click Next. 
6. In this page, Change Project Library Name to dxl\_x86\_c and Project Library Path to “[path\_to\_joaoandthethreestands]\joaoandthethreestands\DynamixelSDK-3.7.31\labview”. Then click Next. 
7. In this page, select “Simple Error Handling”. Click Next.
8. Afterwards, leave the settings as they are, and continue clicking Next on the next set of pages until it imports and finishes. Once it finishes, untick “Open the generated library” and click Finish.

## 3.3 Main Program

As of time of writing this manual, due to the ever-changing nature of the LabVIEW program, this manual will not cover all the specifics of the main program but will only cover the general startup and stopping procedures.

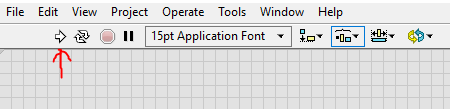
### 3.3.1 Opening the Program

The program must have run through a LabVIEW project. To find this project go to this path “[path\_to\_joaoandthethreestands]\joaoandthethreestands\StandProgram”. Double click on the “StandProgram.lvproj” to open the project. In this project, double click on Main.vi to open the stand program. The following unlabelled figure shows where to find Main.vi in the project.



### 3.3.2 Running the Program

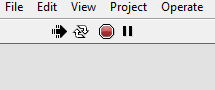
To run the program simply click the run button, which is the white arrow pointing right found near the top-left corner of the window. The following unlabelled figure shows where it is found. The other way to run it is using a shortcut key “Ctrl + r”.



Run button

### 3.3.3 Stopping the Program

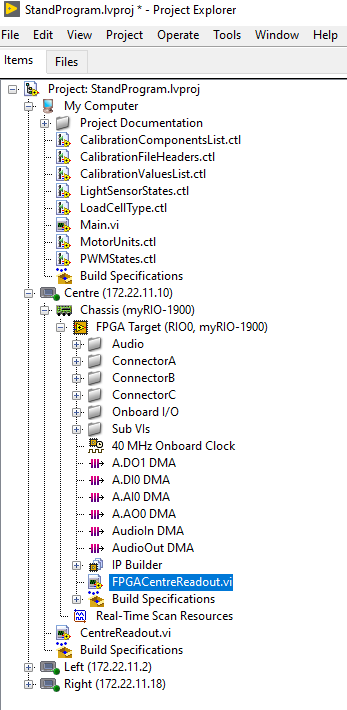
To stop the program, either press on the stop button in the front panel of the main program or click on the abort button (shortcut key for abort is “Ctrl + .”). The stop button is the safer option as it programmatically allows any hardware or channels connected to LabVIEW to be closed or released. The stop button should be located on the front panel, and its location and appearance will be set by the designer of the program. The following unlabelled figure shows where the abort button is found.



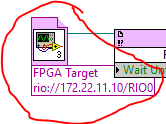
Abort button

### 3.3.4 Compiling the myRIO FPGA Programs

When this rig needs to be run on a new Windows device, then the FPGA codes on the three myRIO’s may need to be recompiled, as the Main.vi may throw an error. In this case, the FPGA VI on each myRIO will need to be forced to recompile. To do so, go into the project, and go into each of the myRIO’s, collapse them, and collapse Chassis (myRIO-1900) -> FPGA Target (RIO0, myRIO-1900). There should be a “FPGA…Readout.vi” as shown in the following unlabelled figure for the Centre myRIO.



Open the VI, and simply add a Boolean control or indicator, save the VI and then run the program which will force a recompile on the myRIO. Once all the FPGA VI’s have been recompiled, then go back into the Main.vi program and see if it runs. If it does not run, then in the block diagram of the Main.vi, make sure the myRIO targets have been configured correctly. The following unlabelled figure shows what the VI looks like in the block diagram. There should be three of these, each for a myRIO.



If the Main.vi still doesn’t run, then try right clicking into the targets, “Configure Open FPGA VI Reference”. There will be a window with a “Build Specification” path. Click on the folder icon next to it, and there should be a “FPGA…Readout\_2”. Click on that and click OK. The Main.vi should run.