ME 477 — Winter 2022
Setting up the C Development Tool for myRIO on your laptop computer

Complete all of the steps below to set up the development environment on your laptop.
Parts 1, 2, and 3 can be performed *without connecting* your laptop to one of the myRIO microcomputer.

Do Parts 1, 2, and 3 just once, in the order shown.

Part 1. Setting up the Software Environment

Follow these instructions to set up the C Development Tool for myRIO.
Clicking on a blue hyperlinks opens the appropriate websites in your browser.[1]

1. Download and install LabVIEW 2015 myRIO Toolkit. (2700 Mb)

   For Native Windows 8 or 10: Download from myRIOToolkit2015. Mount disk image. Then run `setup.exe`.

   For Native Windows 7: Download from myRIOToolkit2015. You may need a means of mounting the .iso disk image. For example, use Virtual CloneDrive to mount the .iso disk image files as a virtual CD-ROM drive. Then run `setup.exe`.

   For Virtual Windows 7, 8, or 10 under Parallels: Download from myRIOToolkit2015 *under OS X*. From Parallels, **Devices→CD/DVD→Connect Image...** mount the disk image. Then run `setup.exe`.

2. Install Java. Visit the Java website GetJava to download Java. (17 Mb) Use Internet Explorer, not Microsoft Edge.

3. Install the C/C++ Development Tools for NI Linux Real-Time 2014, Eclipse Edition.
   Visit this link Eclipse2014 to download and install Eclipse. (260 Mb)

4. Project templates have been prepared for each of the nine ME 477 laboratory exercises.
   Visit the ME 477 website Resources Page to download the ME477 myRIO support 2020 archive.
   Remember where you put this archive, but **do not unzip.** (2 Mb)

5. Eclipse uses Launch Configurations to specify how the project will be deployed and run on the myRIO.
   Visit the ME 477 website Resources Page to download the ME 477 Launch Configurations archive.
   Unzip into folder LaunchConfig477 (40 kb). Remember where you put the folder.

6. Add the compiler path to the system environment variables.

   a. Visit the ME 477 website Resources Page. Open the 64-bit compiler path file, select and copy (ctrl-c) the contents.

   b. Open the Start Search, type in "env", and choose **Edit the system environment variables**.

   c. Click the **Environment Variables** button.

   d. Select **PATH** in the **User variables** group box and click **Edit**.
   If **PATH** does not exist, click **New** to create it.

   e. Click **New** and paste (ctrl-v) the compiler path to **Variable value**. Be certain that there are **no extra spaces** in the path.

7. Click **OK** to close the dialog box and save changes.

[1] This document may found on-line at: http://courses.washington.edu/mengr477. Look for the link: C Development Tool Setup for Laptop

Part 2. Define a connection to the myRIO

Complete the following steps to define a connection in Eclipse from your laptop to the myRIO target:

1. Launch Eclipse, specify a workspace, and click **OK** to display the **C/C++** perspective (default). **Note:** The name of your workspace must not contain a space.

   Two other perspective views, **Remote Systems Explorer** and **Debug**, will also be useful. To make these available, select **Window→Open Perspective→Other** to display the **Open Perspective** dialog box.

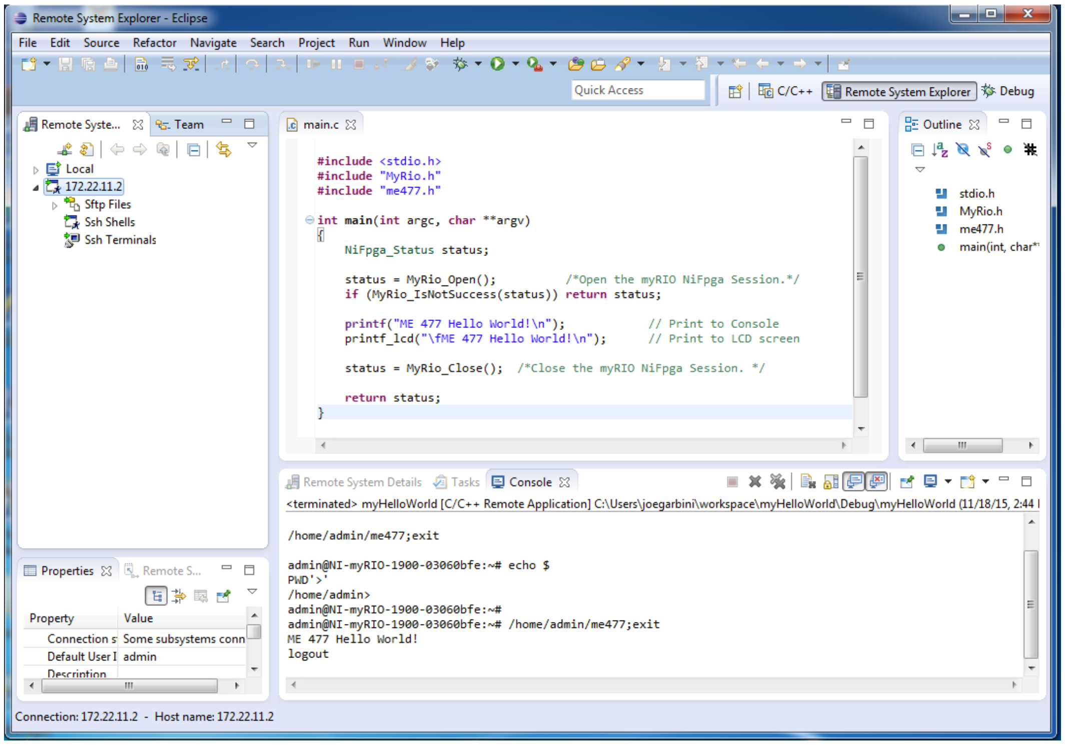
   Then select **Remote Systems Explorer** and click **OK** to display the Remote Systems Explorer perspective. Repeat this process to display the **Debug** perspective. Buttons for all three perspective should appear in the upper right corner of your eclipse window, and can be used at any time to switch perspectives.

2. Select the **Remote Systems Explorer** perspective to display the **Remote Systems** pane at left.

3. Click the **Define a connection to remote system** icon to display the **New Connection** dialog box.

4. Select **General→SSH Only**.

5. Enter the IP address `172.22.11.2` in the **Host name** textbox and click **Finish**. Your target displays in the Remote Systems tab in the Remote System Explorer pane



```c
#include <stdio.h>
#include "MyRio.h"
#include "me477.h"

int main(int argc, char **argv)
{
    NiFpga_Status status;

    status = MyRio_Open();          /*Open the myRIO NiFpga Session.*/
    if (MyRio_IsNotSuccess(status)) return status;

    printf("ME 477 Hello World!\n");             // Print to Console
    printf_lcd("\fME 477 Hello World!\n");       // Print to LCD screen

    status = MyRio_Close();  /*Close the myRIO NiFpga Session. */

    return status;
}
```

```
<terminated> myHelloWorld [C/C++ Remote Application] C:\Users\joegarbini\workspace\myHelloWorld\Debug\myHelloWorld (11/18/15, 2:44 

/home/admin/me477;exit

admin@NI-myRIO-1900-03060bfe:~# echo $
PWD'>'
/home/admin>
admin@NI-myRIO-1900-03060bfe:~#
admin@NI-myRIO-1900-03060bfe:~# /home/admin/me477;exit
ME 477 Hello World!
logout
```

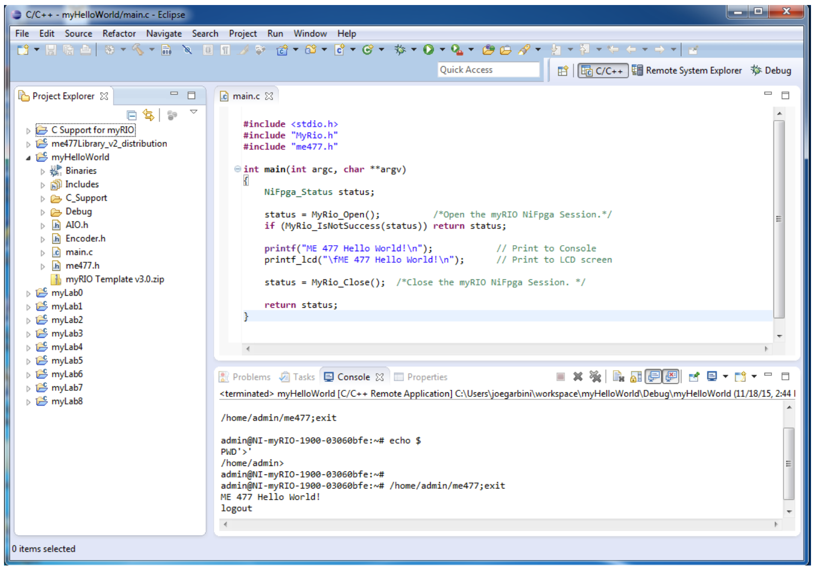
Property	Value
Connection s	Some subsystems conn
Default User I	admin
Description	

Connection: 172.22.11.2 - Host name: 172.22.11.2

Part 3. Importing C Support and Launch Configurations

Complete the following steps to import C Support and Launch Configurations to Eclipse:

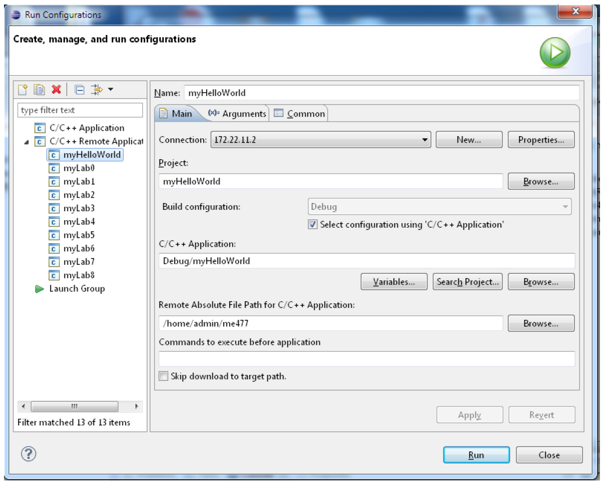
1. From the **C/C++** perspective, select **File→Import** to display the **Import** dialog box.
2. Select **General→Existing Projects into Workspace** and click **Next** to display the **Import Projects** page.
3. Select **Select archive file**, click **Browse** and select the **ME 477 C Support for myRIO** that you downloaded in step 4. of Part 1.
4. Ensure that all items are checked and click **Finish** to import ME 477 C Support for myRIO. See Figure 1.
5. Under the **Project** menus select **Build All**.
6. Again, from the **C/C++**, select **File→Import** to display the **Import** dialog box.
7. Select **Run/Debug→Launch Configurations** and click **Next** to display the **Import Launch Configurations** page.
8. Click **Browse** and select the **LaunchConfig477** folder that you downloaded in step 5. of Part 1.
9. Ensure that all items are checked and click **Finish**. To check that the Import of the Launch Configurations was successful, pull down **Run→Run Configurations. . .** See Figure 2.



```c
#include <stdio.h>
#include "MyRio.h"
#include "me477.h"

int main(int argc, char **argv)
{
    NiFpga_Status status;

    status = MyRio_Open();          /*Open the myRIO NiFpga Session.*/
    if (MyRio_IsNotSuccess(status)) return status;

    printf("ME 477 Hello World!\n");            // Print to Console
    printf_lcd("\fME 477 Hello World!\n");      // Print to LCD screen

    status = MyRio_Close();  /*Close the myRIO NiFpga Session. */

    return status;
}
```

```
<terminated> myHelloWorld [C/C++ Remote Application] C:\Users\joegarbini\workspace\myHelloWorld\Debug\myHelloWorld (11/18/15, 2:44 I

/home/admin/me477;exit

admin@NI-myRIO-1900-03060bfe:~# echo $
PWD'>'
/home/admin>
admin@NI-myRIO-1900-03060bfe:~#
admin@NI-myRIO-1900-03060bfe:~# /home/admin/me477;exit
ME 477 Hello World!
logout
```

Figure 1. ME 477 Project Templates



Create, manage, and run configurations

Name: myHelloWorld

Connection: 172.22.11.2

Project: myHelloWorld

Build configuration: Debug

[✓] Select configuration using 'C/C++ Application'

C/C++ Application: Debug/myHelloWorld

Remote Absolute File Path for C/C++ Application: /home/admin/me477

Commands to execute before application

[ ] Skip download to target path.

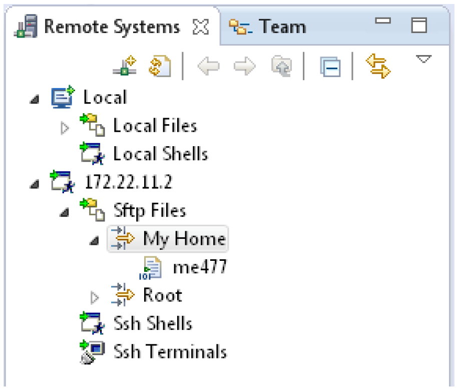
Figure 2. Launch Configurations

Your laptop *must be connected through a USB cable* to the myRIO microcomputer. Each time you connect, a **myRIO USB Monitor** dialog box will appear indicating myRIO IP Address: 172.22.11.2.
Always select **Do Nothing**.

Part 4. Connect to the myRIO target

Complete the following steps to establish a connection between Eclipse and the myRIO target:

1. In the **Remote Systems** pane, right-click the target and select **Connect** from the shortcut menu to display the **Enter Password** dialog box.
2. Enter the user ID: `admin` and password: `me477` and click **OK**.
3. Click **OK** in the **Info** dialog box.
4. If the **Keyboard Interactive authentication** dialog box appears, leave the password blank, and click **OK**. As shown below, green arrow appears on the target icon when the myRIO is connected.



In Parts 5 and 6 you will run and debug a project. Here, the myHelloWorld project is used as example.

Part 5. Running the myHelloWorld project

Eclipse uses a "Run Configuration" to specify how the project will be deployed and run on the myRIO.
Run Configurations for ME 477 projects were downloaded in step 5 of Part 1.

Complete the following steps to run the myHelloWorld example project.

1. In Eclipse, switch to the **C/C++** perspective.
2. You can view and edit the C source code by double clicking on the myHelloWorld project in the left pane, and then double clicking on `main.c`
3. In the **Project Explorer** pane, right-click the myHelloWorld project, and select **Build Project** from the shortcut menu to build the project. Any build errors will be noted in the **Problems** pane.
4. Right-click the myHelloWorld project and select **Run As→Run Configurations** to display the **Run Configurations** dialog box.
5. Select the myHelloWorld project in the left pane. Be sure that the **Connection:** box is set to 172.22.11.2.
6. Click **Run**. The project runs on the myRIO target.
   You can find the result in the **Console** pane, and on the LDC screen.

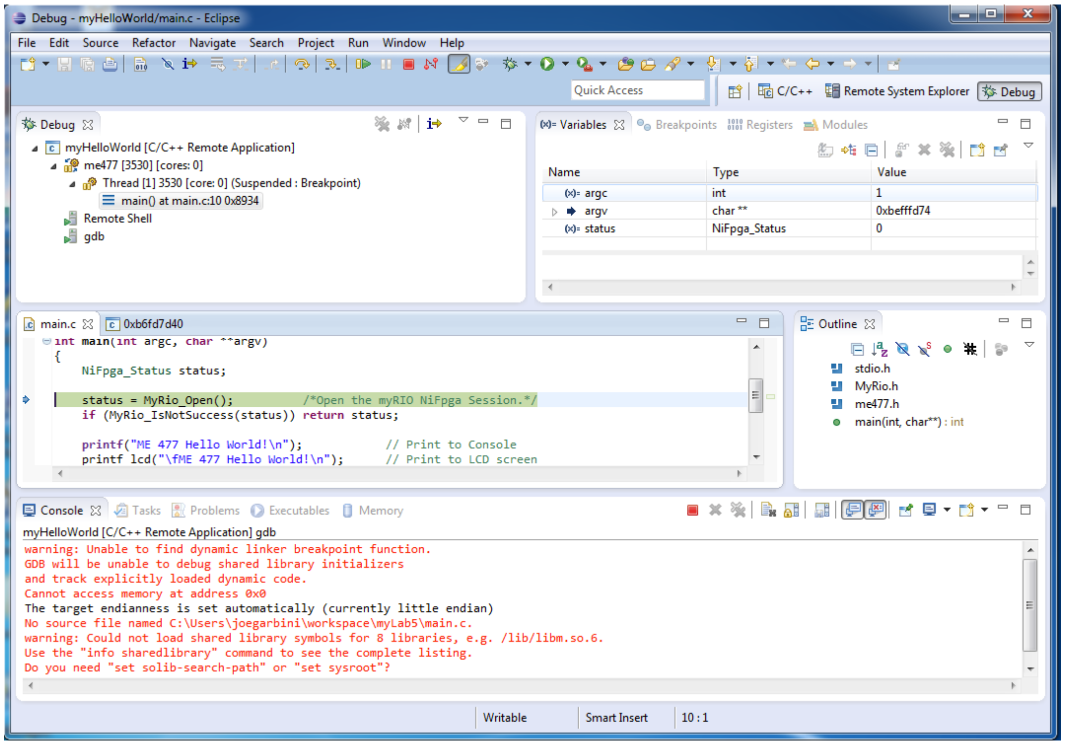
Part 6. Debugging the myHelloWorld project

Similarly, Eclipse uses a “Debug Configuration” to specify how the program will be debugged on the myRIO. Once the Debug Configuration for a project is set up, debugging the program requires just a single click.

Complete the following steps to set up the Debug Configuration for the myHelloWorld project. Step 5. includes building, deploying, and debugging the project:

1. In Eclipse, switch to the **C/C++** perspective.
2. In the **Project Explorer** pane, right-click the myHelloWorld project and select **Debug As→Debug Configurations** to display the **Debug Configurations** dialog box.
3. Select the myHelloWorld project in the left pane.
4. Click **Debug**. The project runs on the myRIO target within the debugger. Some warnings may appear in the Console pane. Under normal circumstances, these warnings are not a problem. You can find the debug tools on the toolbar of Eclipse. There will be more about this in the first laboratory exercise.
5. For now, try setting a breakpoint at the `printf()` statement by double-clicking in the margin at left of that statement. A blue dot *with a small checkmark* should appear in the margin. The blue dot indicates that the breakpoint is enabled, and the checkmark indicates that the breakpoint is installed.

   If you `resume` (green arrow) from the beginning of the program, execution should pause at the breakpoint.



```
myHelloWorld [C/C++ Remote Application]
  me477 [3530] [cores: 0]
    Thread [1] 3530 [core: 0] (Suspended : Breakpoint)
      main() at main.c:10 0x8934
  Remote Shell
  gdb
```

Name	Type	Value
argc	int	1
argv	char **	0xbefffd74
status	NiFpga_Status	0

```c
int main(int argc, char **argv)
{
    NiFpga_Status status;

    status = MyRio_Open();              /*Open the myRIO NiFpga Session.*/
    if (MyRio_IsNotSuccess(status)) return status;

    printf("ME 477 Hello World!\n");                // Print to Console
    printf_lcd("\fME 477 Hello World!\n");          // Print to LCD screen
```

```
myHelloWorld [C/C++ Remote Application] gdb
warning: Unable to find dynamic linker breakpoint function.
GDB will be unable to debug shared library initializers
and track explicitly loaded dynamic code.
Cannot access memory at address 0x0
The target endianness is set automatically (currently little endian)
No source file named C:\Users\joegarbini\workspace\myLab5\main.c.
warning: Could not load shared library symbols for 8 libraries, e.g. /lib/libm.so.6.
Use the "info sharedlibrary" command to see the complete listing.
Do you need "set solib-search-path" or "set sysroot"?
```