MECHTRON/SFWRENG 4AA4 - Lab 3

C Support for NI myRIO

Introduction:

NI myRIO is based on Xilinx 7000 series SoC which integrates the software programmability of a processor (dual core ARM Cortex A9) with the hardware programmability of an FPGA resulting in high levels of system performance, flexibility, and scalability. Graphical programming of LabView/Simulink was used in MECHTRON/SFWRENG 3DX4 to implement different projects on NI myRIO. NI also provides C Support for myRIO in order to use the processing and logic capabilities of myRIO by directly coding in C programming language. We shall take this approach for most of the labs in this course. This requires a set of tools particularly a development environment and a cross compiler, which compiles code developed on a Microsoft Windows machine for NI Linux Real-Time target. This lab introduces you to such tools.

Goals:

Learn how to build and run real time programs written in C programming language on NI myRIO and get conversant with C/C++ Development Tools for NI Linux Real-Time, Eclipse Edition.

Note: Before you go for your lab sessions, please read the following documents at your own convenience, in addition to the class notes:

- C_Support_for_myRIO_User_Guide6.0.pdf. Most of the software components mentioned in this document have already been installed and configured on lab computers.
- Getting Started with C Development Tool(Eclipse).pdf
- Document at this link: http://www.drdobbs.com/soft-real-time-programming-with-linux/184402031. Or, a PDF version can be found in folder of "ref" for Lab 3 .
- Document at this link: https://hpc-tutorials.llnl.gov/posix/pthreads_api/

Activities in the lab:

NOTE 1: The "C/C++ Development Tools for NI Linux Real-Time, Eclipse Edition" has been installed on the lab computers. It can be found under "National Instruments" program group, or by searching for it in the Windows system.

NOTE 2: The myRIOs used in the lab have been configured properly and are ready to use.

Part 1: Create a "Hello World" program to run on myRIO [25]

1. Launch the "C/C++ Development Tools for NI Linux Real-Time, Eclipse Edition".

2. When prompted for a workplace, select a folder in which to store Eclipse projects and click **OK**. (Tip: Enable **Use this as the default and do not ask again** to save a project folder as your default workspace)

3. Create your first project to be run on myRIO:

   - Switch to the C/C++ perspective (If Eclipse is not in the C/C++ perspective, select **Window>>Open Perspective>>Other** to display the **Open Perspective** dialog box. Then select **C/C++ (default)** and click **OK**). Select **File>>New>>Project...** to open the **New Project Wizard**.
   - Expand the **C/C++** folder and select **C Project** or **C++ Project**. Click Next to open the C Project page.
   - Enter a project name in the **Project name** text box. Select the **Hello World ANSI C Project** or **Hello World C++ Project** project type. Select **Cross GCC** in the **Toolchains** list box to enable cross-compilation, which configures the compiler to create executable code for embedded systems, such as your NI Linux Real-Time target.
   - Click **Next** to open the **Basic Settings** page. Enter the basic properties of your project in the **Author, Copyright notice, Hello world greeting** and **Source** text boxes.
   - Click **Next** to open the **Select Configurations** page. Enable **Debug** to configure the project to allow debugging your executable, and/or enable **Release** to configure the project to allow building a smaller, faster executable optimized for release. (Note: For purposes of this tutorial, ensure you enable **Debug**.)
   - Click **Next** to open the **Cross GCC Command** page. In the **Cross compiler prefix** text box, enter (include the hyphen at the end): `arm-nilrt-linux-gnueabi-`

     In the **Cross compiler path** text box, browse (**Do Not Copy!**) to the location of the correct compiler for the target that you will use: `C:\build\17.0\arm\sysroots\i686-nilrtsdk-mingw32\usr\bin\arm-nilrt-linux-gnueabi`
   - Click **Finish** to create your project and return to the workbench view.

   NOTE:: After the project is created, errors `'Program "g++" not found in PATH'`, `'Program "gcc" not found in PATH'`, or `'symbol "EXIT_SUCCESS" could not be resolved'` may appear in the **Problems** tab in the lower part the Eclipse window. These errors can be safely ignored and removed by right-clicking the errors and selecting **Delete**.

4. Build the C/C++ Project:

   Before you can run your project, you need to test that your source code compiles by creating an executable build of your project. Complete the following steps to create an executable build of a C/C++ project.

   - Switch to the C/C++ perspective.

- Modify the template source code if you want to adapt the template to your application needs, or enter the C/C++ code if you selected a blank project. For purposes of this tutorial, you do not need to make changes to the Hello Word ANSI Project source code.
- Right-click your project in the **Project Explorer** tab and select **Properties**.
- Select **C/C++ Build** in the left pane of the **Properties** dialog box.
- Select **Internal builder** from the **Builder type** pull-down menu to build the executable. Selecting the **Internal builder** is to use C/C++ Development Tools for NI Linux Real-Time, Eclipse Edition, instead of an external build file, to build the executable.
- Select **Settings** under **C/C++ Build** in the left pane of the **Properties** dialog box.
- Select **Miscellaneous** under **Cross GCC Compiler** in the **Tool Settings** tab.
- In the **Other flags** text box, add a space after the existing text, and then enter the following settings to improve the performance of floating-point operations:
  `-mfpu=vfpv3 -mfloat-abi=softfp --sysroot=c:\build\17.0\arm\sysroots\cortexa9-vfpv3-nilrt-linux-gnueabi`
- Select **Miscellaneous** under **Cross GCC Linker** in the **Tools Settings** tab. In the **Linker Flags** text box, enter:
  `--sysroot=C:\build\17.0\arm\sysroots\cortexa9-vfpv3-nilrt-linux-gnueabi`
- In the left pane of the **Properties** dialog box, select **Paths and Symbols** under **C/C++ General**.
- Under the tab of **include**, for **GNU C** or **GNU C++**, add the following three paths:
  `C:\build\17.0\arm\sysroots\cortexa9-vfpv3-nilrt-linux-gnueabi\usr\include\c++\4.9.2\`
  `C:\build\17.0\arm\sysroots\cortexa9-vfpv3-nilrt-linux-gnueabi\usr\include\c++\4.9.2\arm-nilrt-linux-gnueabi`
  `C:\build\17.0\arm\sysroots\cortexa9-vfpv3-nilrt-linux-gnueabi\usr\include`
- Click **Apply** and then **OK** to close the **Properties** dialog box.
- Select **Project>>Build Project** in the workbench view to create an executable of your project. If the build completes successfully, the **Console** tab will display **Build Finished** without any errors, and a `'Binaries'` folder will be created under the project in the Project Explorer panel. with a binary file in the folder.

5. Configure a Remote System:

   Before you can run the executable you created in the previous section on your NI Linux Real-Time target, you need to add your target to the project. Complete the following steps to configure your target as a remote system in C/C++ Development Tools for Linux Real-Time, Eclipse Edition.

   - Select **Window>>Open Perspective>>Other** to open the **Open Perspective** dialog box. Select **Remote System Explorer**. Click **OK** to add the Remote System Explorer perspective to the workbench.

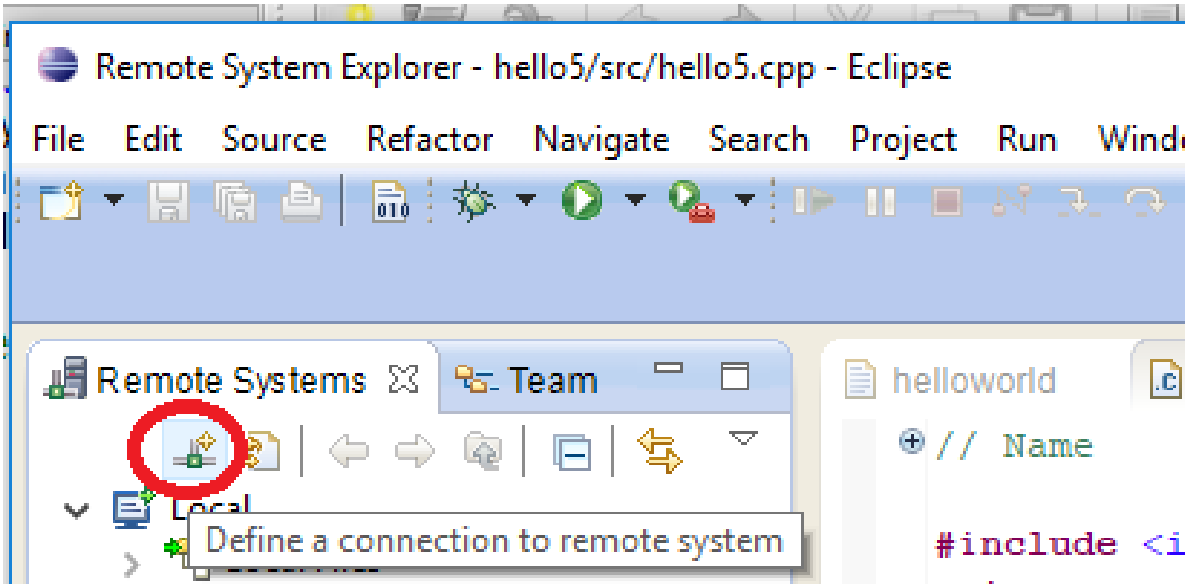

Figure 1: Define a Connection to Remote System

- Click the **Define a connection to remote system** button, circled in Figure 1, to open the **New Connection** window.
- Select **SSH Only** under the **General** folder. Click **Next** to open the **Remote SSH Only System Connection** window.
- Enter the IP address of your myRIO in the **Host name** text box. The assigned IP address can be found on the sticker on the myRIO.
- (Optional) Change the connection name in the **Connection name** text box or enter a description in the **Description** text box to help you identify your target when it appears in the Remote System Explorer perspective.
- Click **Finish**. Your target is displayed in the **Remote Systems** tab in the **Remote System Explorer** perspective.
- Right click the connection name or the IP address of the target that you want to connect in the **Remote System Explorer**, and select the **Connect** command from the context menu.
- When prompted, enter the user name and password assigned to your target and click OK. The login name is **"admin"** and the password is **"4aa4"**.
  At this stage, you can let the system remember your credentials by checking the checkboxes for saving your user ID and your password.
- With a successful login, you establish an SSH connection to your target and enable transferring files to it.

6. Run the C/C++ Executable on Your NI Linux Real-Time Target:

   At this point, your project contains a target and an executable. Complete the following steps to run your C/C++ executable on your target.

- Go back to the C/C++ Perspective. Select **Run>>Run Configurations**, or right click a project in the Project Explorer and select **Run as | Run Configurations...** from the context menu, to open the Run Configurations dialog box. Select **C/C++ Remote Application** in the left pane. Click the **New launch configuration** button, circled in Figure 2, to specify settings for running an executable on your target.

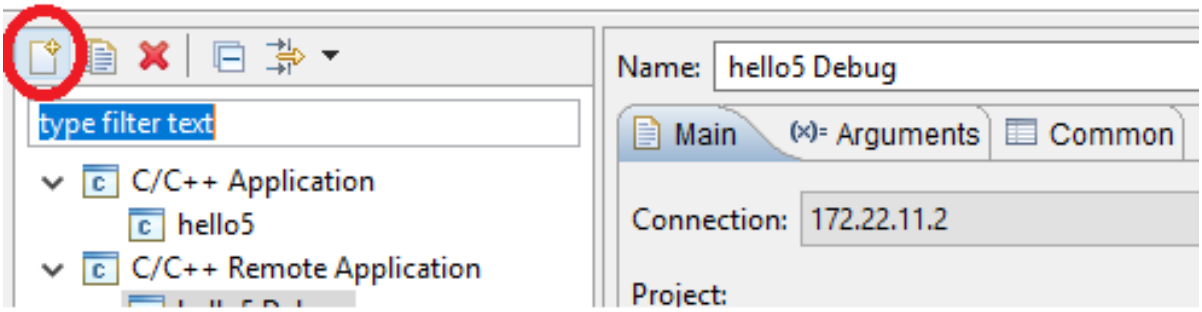

Figure 2: Create a New Run Configuration

- Select your connection name or the IP address of your target from the **Connection** pull-down menu.
- Click the **Browse** button to the right of the **Project** box to select the project you will run, Click the **Search Project...** button below the **C/C++ Application** box to choose a **C/C++ Application** to run.
- Click the **Browse** button beside the **Remote Absolute File Path for C/C++ Applications** text box to open the **Select Remote C/C++ Application File** dialog box. This will set the path and the executable file name on the remote MyRIO target.
- Right-click the **My Home** directory in the list box and select **New>>Folder** to create a folder on the target, or select an existing folder under the **My Home** directory, in which to place a copy of the executable.
- Return to the **Run Configuration** window. Append your project name (or whatever name you wish to assign to the executable that will be transferred to the target) to the path populated in the **Remote Absolute File Path for C/C++ Applications** text box, as shown in Figure 3.

  Note: You must have a **valid remote absolute path** and **an application name for the executable binary** to run your project!!!
- Click **Apply** and then **Run** to transfer the executable binary to your target and to run it.

  Note: The first time when you run the application, the system may give you error message "Permission Denied". This may be a bug of the Eclipse, which may try to run the executable binary before the system get the binary ready to run. In this case, just try to run it again.

7. Change the output to your names **(consistent with Avenue)**, your student number, your group number, and the myRIO IP address. **Take a screenshot of your result and show it to the TA at the end of the lab session**.

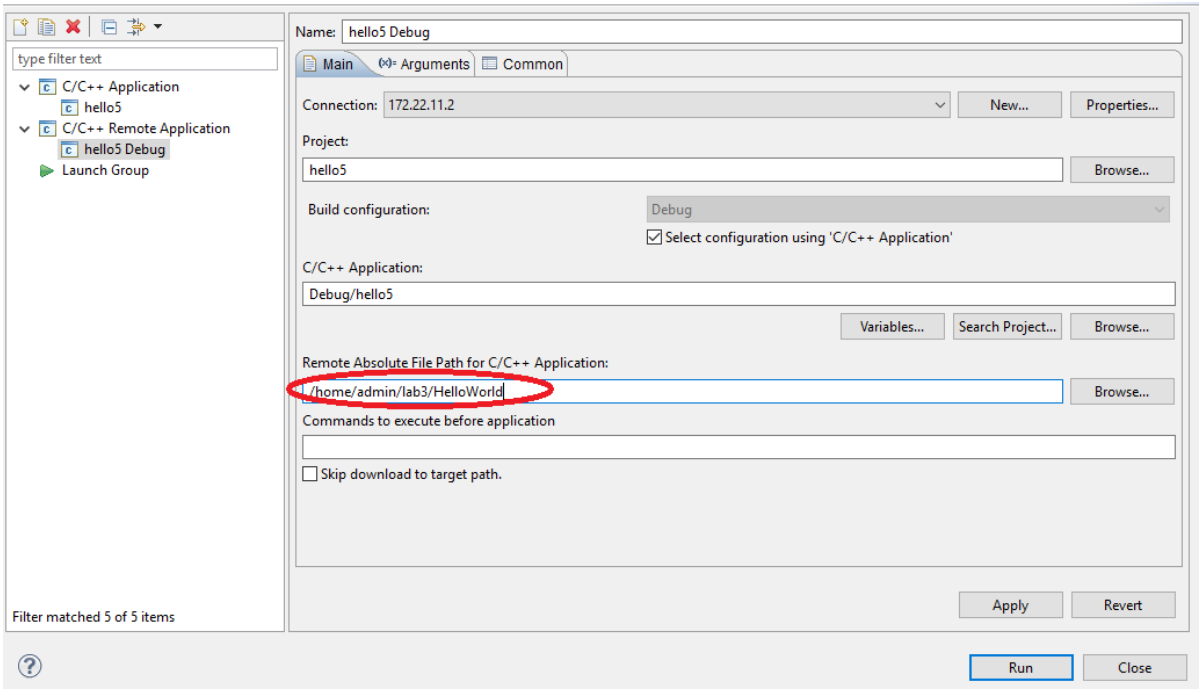

Figure 3: Specify a Name and a Remote Path for the Application

Part 2: Compile and run an example myRIO project [25]

1. Download the `C_Support_for_myRIO1900_v6.0.zip` from the course web page, or copy it from folder `C:\Program Files(x86)\National Instrument\Eclipse` on the lab computers.

2. Launch Eclipse, specify a workspace, and click **OK** to display the C/C++ perspective, if necessary.

3. Select **File** >> **Import** to display the **Import** dialog box.

4. Select **General** >> **Existing Projects into Workspace** and click **Next** to display the **Import Projects** page.

5. Select **Select archive file**, click **Browse** and select the `C_Support_for_myRIO1900_v6.0.zip` that you download.

6. Ensure that all items are checked and click **Finish** to import C Support for myRIO to Eclipse.

7. In your workspace you will have twelve folders containing sample projects for myRIO and a folder named 'C Support for myRIO' containing a sub-folder named 'template project', (along with other sub-folders). In the lab part, we will use the example project `'myRIO Example - Accelerometer'`.

8. In the **Project Explorer** pane, right click project `'myRIO Example - Accelerometer'` and select **Build Project** from the context menu to build the example project.

   Note 1: Check if there is a `'Binaries'` folder under the project in the Project Explorer panel. If so, it means the project was built successfully.

   Note 2: After Part 1, the Eclipse may still remember your compiler settings so you can build your

project readily. Otherwise, you need to follow the Step 4 in Part 1 to configure the compiler first to build your project.

9. In the **Project Explorer** pane, right click the project and select **Run as >> Run configurations** to display the **Run Configurations** dialog box.

10. Follow the Step 6 in Part 1 to set up the Run Configurations and to deploy the project.

11. The output for this project is the x, y and z coordinates of the position of the myRIO box. Move myRIO box and observe how the output changes.

12. **Take a screenshot of your result and show it to the TA at the end of the lab session.**

Part 3: Create a real-time task [25]

1. Find the `myRio Template` project, then rename it to a suitable name such as `lab3part3`.

2. Double click on the project name in the **Project Explorer** or use the arrow sign to the left of the project name to open the folder. There is a `main.c` file in addition to other folders/files.

3. Double click `main.c` which opens in an editor. Notice that the template contains basic code to open the myRIO FPGA session. You will not use the FPGA for this lab and can replace the code with your own code.

4. Replace the code in `main.c` with that given in file named `task1RT.c`, which can be found in the A2L, and build the project. You may see some error messages in the **Problems** tab, saying that certain symbols could not be resolved, however if the messages in the **Console** tab indicate that the project compiled successfully, these error messages can be deleted. Rebuild the project and no error will be displayed!

5. Run the project on your myRIO by following the steps similar to those in part 1.

6. Add your name in the output. **Take a screenshot of the messages displayed by the program and show the output to the TA at the end of the lab session**.

Part 4: Create a task as a thread [25]

A real time task can also be created as a thread. This is particularly useful when you want to run a number of tasks simultaneously. In this part of the lab you will create a task similar to that in part 3 but as a thread. Each thread needs a function that is required to execute after the thread is created in the main() function.

1. Because the `myRIO Template` project has been renamed to `lab3part3` in Part 3, we need to create a new `myRIO Template` project:

   - In Eclipse with C/C++ Perspective, select **File>>Import** to display the **Import** dialog box.
   - Select **General>>Existing Projects into Workspace** and click **Next** to display the **Import Projects** page.

- Select **Select archive file**, navigate to <`workspace`>\`C Support for myRIO\template project`, where `<workspace>` is the workspace directory that you have specified in Eclipse, and select the `myRIO Template v6.0.zip` file. The `myRIO Template` now appears in the **Projects** list.
- Click **Finish**. A new `myRIO Template` project will appear in the **Project Explorer**.

2. Rename the `myRIO Template` project to a suitable name, such as `lab3part4`.

3. Replace the code in `main.c` in this project with that given in file named `threadedTask.c`, which can be found in the A2L.

4. To build this project, we need to modify the settings for the Cross GCC compiler:

   Right click on the project name in the **Project Explorer** and choose **Properties**. Under **C/C++ Build** >> **Settings** > **Cross GCC Compiler** > **Symbols**, click add button to add:

   `_GNU_SOURCE`

   in the **Defined symbols (-D)** window, then click **Apply** and **OK**. Now the project can be built.

5. Build and run the project on your myRIO.

6. Add your name in the output. **Take a screenshot of your result and show the output to the TA.**

FAQ

Q: the console window shows "No build find" when I build the project in Part 1.
A: You may need to do Step 3 again, make sure the Cross Compiler prefix and "Cross-compiler Path" are exactly the same as instructed. Make sure the value for the "Other flags" in Step 4 is correct.

Q: Cannot run my project in myRIO.
A: 1. Look at Figure 3 in the lab manual, you need to make sure the file paths are correct. Don't forget to append **a file name** after the folder name for the `Remote Absolute File Path for C/C++ Application`.
2. Make sure you run the project on myRIO, not at the local computer.

Q: When I run the project, why do I get the "Permission denied" error?
A: Many times, the first time you run the project you will get this error. Try to run the project for the second time. If you still get this error, ask your TA for help or to change a myRIO.