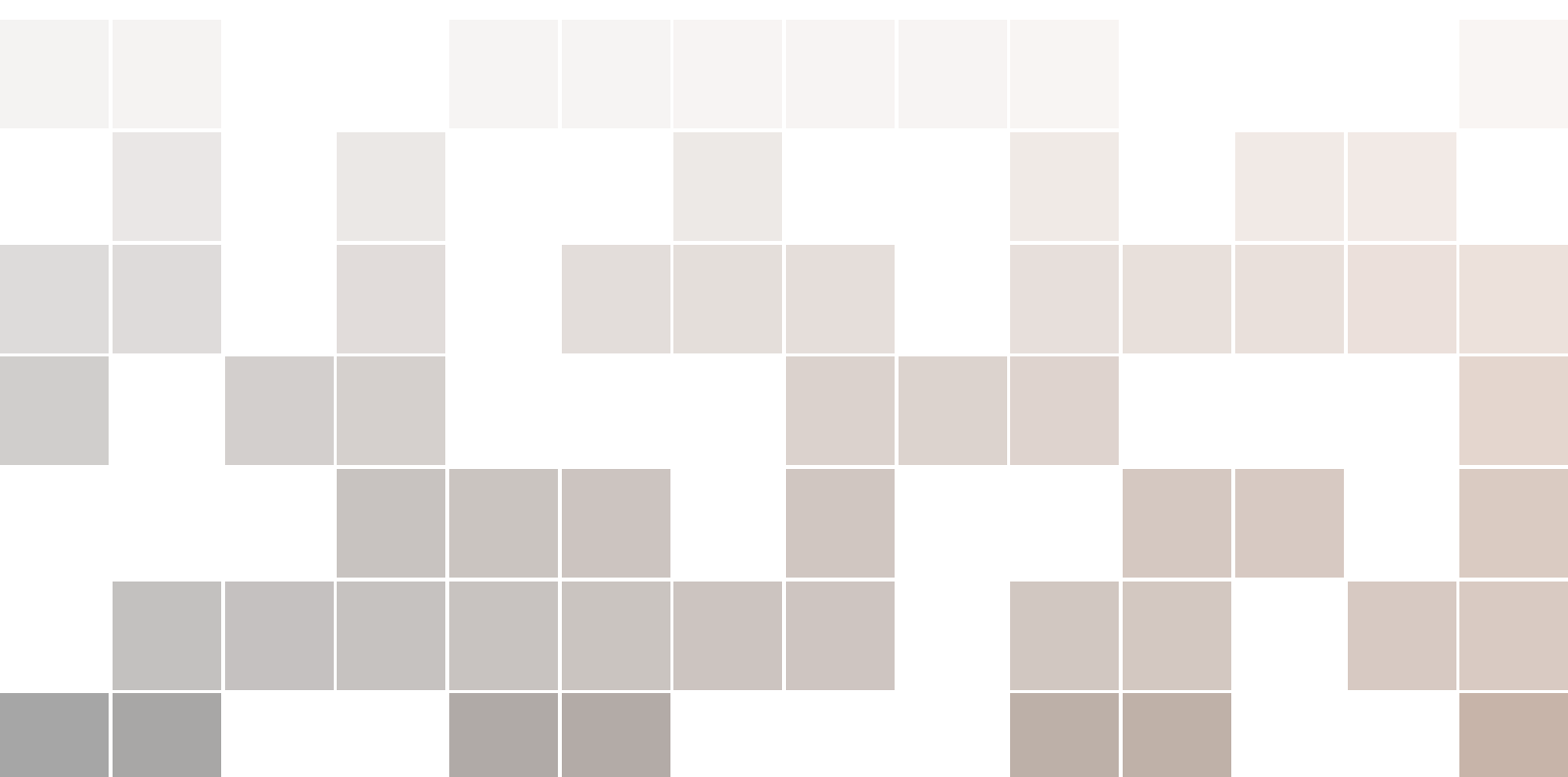




Hands-on Activities using National Instruments myRIO

EE4455: Embedded Systems

Dr. Shi Dongyuan, Mr. Kenneth Ooi, Dr. Lam Bhan and Prof. Gan Woon Seng



Copyright © 2019-2021 DSP Lab

First printing, Jan 2019

Contents

I	Document A: Introduction to myRIO	
1	Introduction of myRIO	7
1.1	Motivation	7
1.2	Introduction	7
1.3	NI myRIO Hardware Overview	9
1.4	Connector Pinouts	9
2	Setting Up Your NI myRIO	11
2.1	NI myRIO Box Contents	11
2.2	Software Installation	11
2.3	Hardware Setup	13
2.4	Additional Resources	14
II	Document B: Getting familiar with myRIO	
3	Getting Started with myRIO	19
3.1	Initial Setup	19
3.2	LabVIEW Environment for myRIO	23
3.3	Analog Input and Analog Output (Express VI)	26
3.4	Accessing OnBoard Devices	29
3.4.1	Accelerometer Express VI	29
3.4.2	Button Express VI	31

3.4.3	LED Express VI	32
3.4.4	Concept of a Shared Variable	33

4	Some Examples on myRIO	37
4.1	Accelerometer	37
4.2	Motion Sensor	38
4.3	Clap Sensor	40
4.4	Microphone Sensor	41
4.5	References	43

III

Document C: Hands-on CA using myRIO

5	CA1:myRIO AIR MUSIC	49
5.1	Reading the acceleration values	50
5.2	Sharing the values of the acceleration with the HOST VI	50
5.3	LabVIEW Environment for myRIO	51
5.4	Sound Synthesizer	51
5.5	More (Optional) Tasks	52
6	CA 2: Ambient Sensing	53
6.1	MEMS Microphone	54
6.2	USB Flash Drive	56
6.3	Logging MEMS microphone audio data on USB Flash drive	56
6.4	Analyzing the sound file	56
6.5	Optional Tasks	56



Document A: Introduction to myRIO

1	Introduction of myRIO	7
1.1	Motivation	
1.2	Introduction	
1.3	NI myRIO Hardware Overview	
1.4	Connector Pinouts	
2	Setting Up Your NI myRIO	11
2.1	NI myRIO Box Contents	
2.2	Software Installation	
2.3	Hardware Setup	
2.4	Additional Resources	



1. Introduction of myRIO

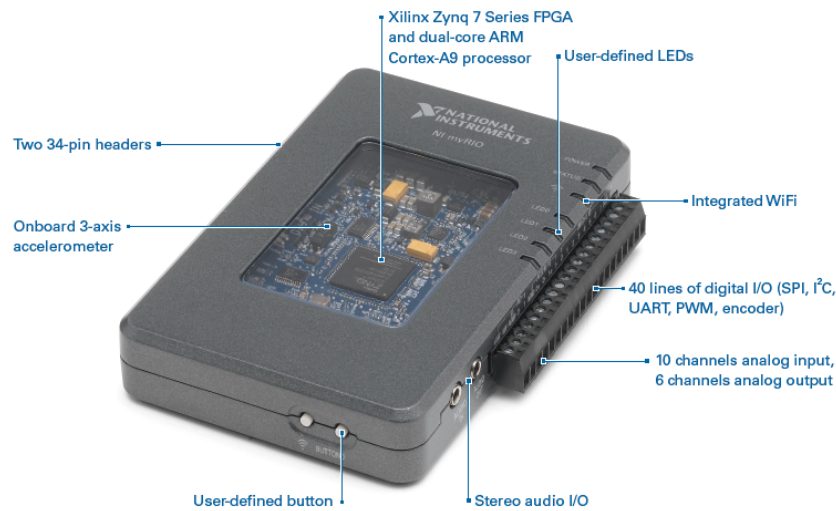
1.1 Motivation

In this document, students will be introduced to the myRIO hardware and the NI software programming language, known as the LabVIEW. This document is a self-paced document and students can go through this document at their own time. It is recommended to complete this document before the second week of the semester. There are several videos and documents available in the website: www.ni.com.

1.2 Introduction

Designed for hands-on experimentation outside the lab, NI myRIO is a data acquisition device for collecting and processing data, which combines portability with a comprehensive set of features. The National Instruments myRIO-1900 is a portable reconfigurable I/O (RIO) device that students can use to design control, robotics, and mechatronics systems. This document contains pinouts, connectivity information, dimensions, mounting instructions, and specifications for the NI myRIO-1900. NI myRIO places dual-core ARM Cortex-A9 real-time processing and Xilinx FPGA customizable I/O into the hands of students. With its onboard devices, seamless software experience, and library of courseware and tutorials, NI myRIO provides an affordable tool that students can use to do real engineering in one semester. It can break down the walls of the traditional lab and lecture by giving the students the ability to perform short lab experiments anywhere and anytime. It can be applied in dealing with circuits, mechanical measurements and signals and systems. The NI myRIO is shown in Figure 1.1 below. **Features:**

- Single device provides eight commonly used plug-and-play computer-based lab instruments.
- Designed by NI with high-quality TI analog ICs.
- Compact and portable for use anywhere, anytime.
- Extend the capabilities with LabVIEW.
- iPod-compatible 3.5mm audio input and output for mixing and manipulating audio.
- In-Built Accelerometer.
- Wifi Compatibility.



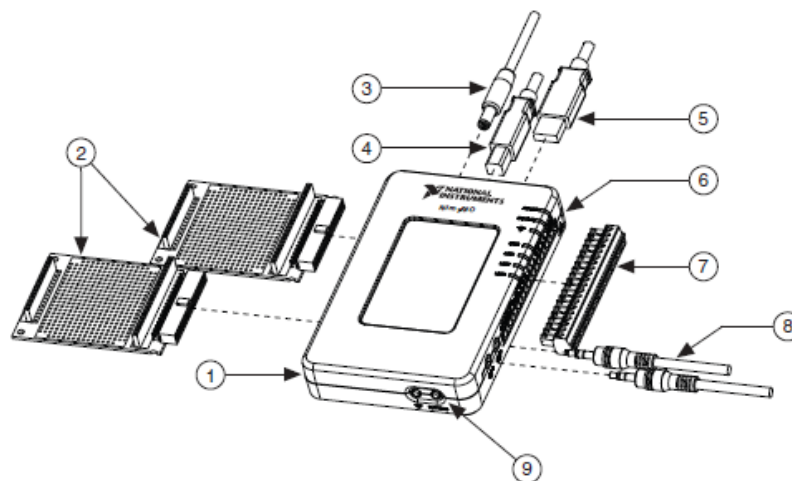
The cutting-edge features of the new NI myRIO allow engineering students to complete complex senior design projects in just one semester.

Figure 1.1: NI MyRIO

Specifications 1 NI MyRIO:

- Two differential analog input and analog output channels (200 ks/s, 16 bit, +/- 10 volts).
- +5, +15, and -15 volt power supply outputs (up to 500m Watts of power).
- Eight digital input and digital output lines (3.3 volt TTL-compatible).
- 60 Volt digital multimeter (DMM) for measuring voltage, current, and resistance.
- Reusable storage box with storage tray, DMM probes, and audio cable.

More information about myRIO can be found on the NI website.



- | | |
|--|---|
| 1 NI myRIO-1900 | 6 LEDs |
| 2 myRIO Expansion Port (MXP) Breakouts (One Included in Kit) | 7 Mini System Port (MSP) Screw-Terminal Connector |
| 3 Power Input Cable | 8 Audio In/Out Cables (One Included in Kit) |
| 4 USB Device Cable | 9 Button0 |
| 5 USB Host Cable (Not Included in Kit) | |

Figure 1.2: NI myRIO parts

1.3 NI myRIO Hardware Overview

The NI myRIO-1900 provides analog input (AI), analog output (AO), digital input and output (DIO), audio, and power output in a compact embedded device. The NI myRIO-1900 connects to a host computer over USB and wireless 802.11b,g,n.

The following figure shows the arrangement and functions of NI myRIO-1900 components.

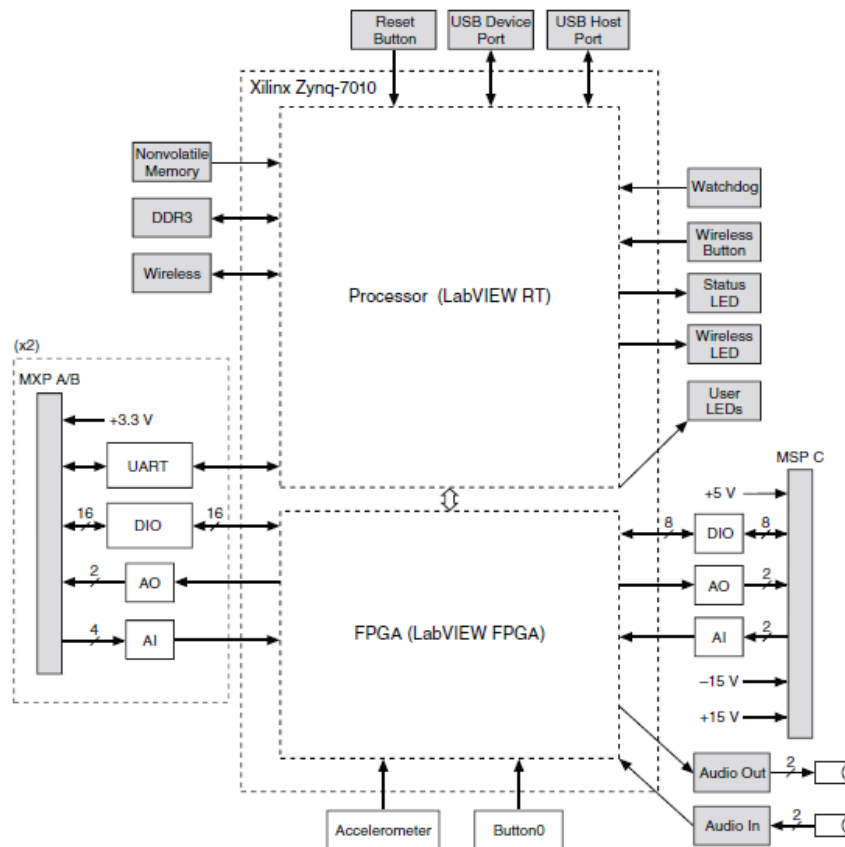


Figure 1.3: NI myRIO hardware block diagram

Students are encouraged to check the various components from www.ni.com and understand their functionalities and specifications. Some of this information will be asked in the Quiz CA at the end of the semester.

NOTE: More information on the specifications of myRIO can be found from the document 'NI myRIO User Guide and Specifications.pdf' on NTUlearn.

1.4 Connector Pinouts

NI myRIO-1900 Expansion Port (MXP) connectors A and B carry identical sets of signals. The signals are distinguished in software by the connector name, as in ConnectorA/DIO1 and ConnectorB/DIO1. Refer to the software documentation for information about configuring and using signals. The following figure and table show the signals on MXP connectors A and B. Note that some pins carry secondary functions as well as primary functions.

The following figure and table show the signals on Mini System Port (MSP) connector C. Note that some pins carry secondary functions as well as primary functions. The NI myRIO-1900 has

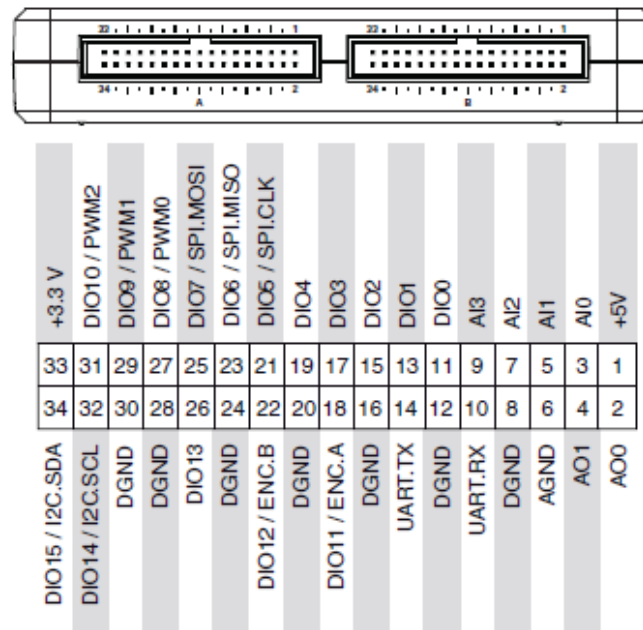


Figure 1.4: NI myRIO MXP Connectors A and B

analog input channels on myRIO Expansion Port (MXP) connectors A and B, Mini System Port (MSP) connector C, and a stereo audio input connector. The analog inputs are multiplexed to a single analog-to-digital converter (ADC) that samples all channels. MXP connectors A and B have four single-ended analog input channels per connector, AI0-AI3, which you can use to measure 0-5 V signals. MSP connector C has two high-impedance, differential analog input channels, AI0 and AI1, which you can use to measure signals up to ± 10 V. The audio inputs are left and right stereo line-level inputs with a ± 2.5 V full-scale range.

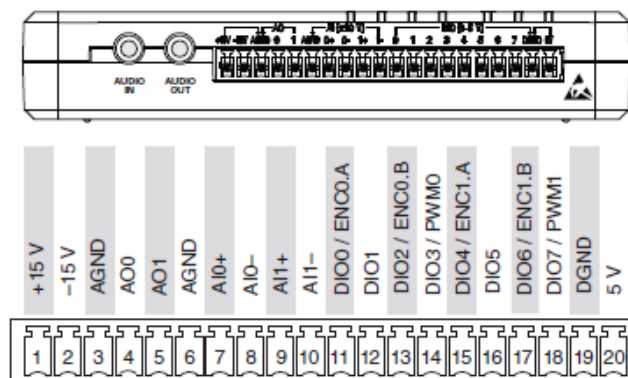



Figure 1.5: NI myRIO MSP Connector C

The NI myRIO-1900 has analog output channels on myRIO Expansion Port (MXP) connectors A and B, Mini System Port (MSP) connector C, and a stereo audio output connector. Each analog output channel has a dedicated digital-to-analog converter (DAC), so they can all update simultaneously. The DACs for the analog output channels are controlled by two serial communication buses from the FPGA. MXP connectors A and B share one bus, and MSP connector C and the audio outputs share a second bus.



2. Setting Up Your NI myRIO

This chapter covers the installation of NI myRIO software and setting up your NI myRIO.

2.1 NI myRIO Box Contents

The first item you see when you open your NI myDAQ box is the Getting Started place mat. This place mat walks you through installing software, setting up your NI myRIO, and running your first VI.

Next, locate your NI myRIO in the box. Keep the tray and box to store your NI myRIO and accessories.

1 Take the tray out of the box and make sure you have these items:

1. NI myRIO embedded student design device.
2. Driver and Software Installation DVDs (NI LabVIEW Student Edition, NI myRIO Module).
3. USB Cable.
4. Power supply with international adapters.
5. MXP accessory.
6. NI Screw Driver and MSP Screw Terminal Connector.

2.2 Software Installation

WARNING: This step requires a fast internet connect and may take a few hours and quite a far bit of disk space (10 – 15 GB). Do not run anything in the background to speed up the process.

Before you can start using your NI myRIO, you need to install the required software from the NI myRIO Software Bundle. For your convenience, the software bundle (5.9 GB) has been downloaded from the NI website and uploaded to Google Drive. Download the zipped file from this link: <https://goo.gl/Ttp4eb>.

NOTE: The NI software required is only officially supported on Windows 8.1/8/7/Vista/XP SP3. It should be able to run on Windows 10. If you have a Mac, install Windows 8.1 with

Bootcamp (<https://support.apple.com/en-gb/HT201468>). You can get windows 8.1 from Microsoft DreamSpark for free using your NTU EEE account here: <https://goo.gl/205RDy>. Press “Register” then “request an account” if you do not already have the NTU EEE DreamSpark account.

Unzip the file and navigate to the “2015 (web-based)/Installer/setup”. Install the products as shown in Figure 2.1.

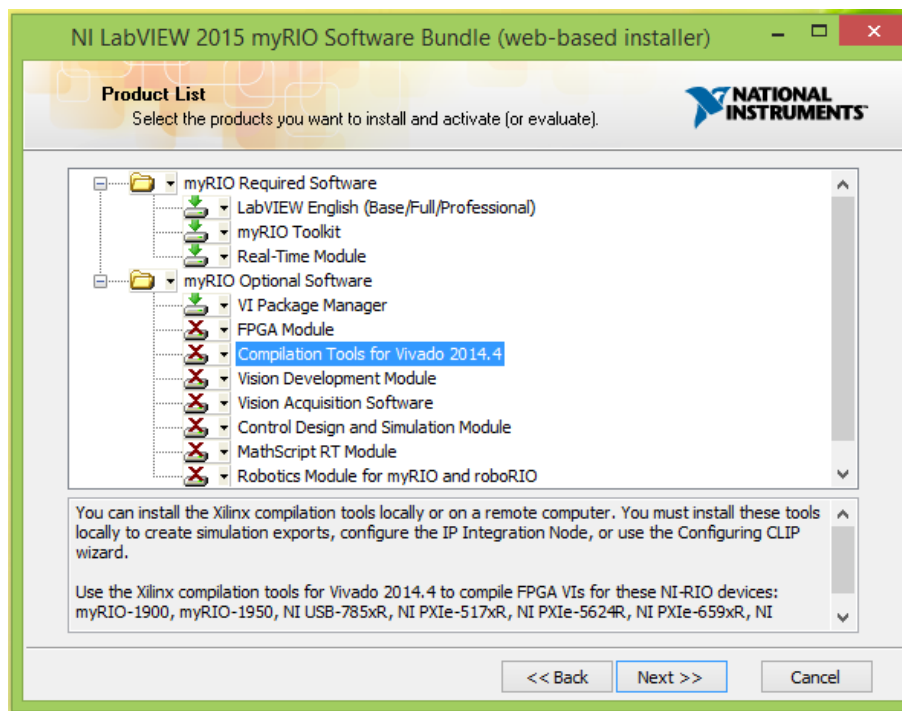


Figure 2.1: Product List to be installed

The NI LabVIEW software and its toolkits/modules must be activated in order for them to work (Figure 2.2). Head to the NI website to request for a 6-month student license: <https://decibel.ni.com/content/docs/DOC-30610>. Once you have requested for a serial, you will see the serial as shown in Figure 2.3.

NOTE: Please store the serial number in a safe place.

You next will be requested to sign in to your NI.com account. Create one if you don’t have an account. The 6-month student license serial number will be tied to this email.

You would be prompted to restart your computer to complete the installation process.

Figure 2.2: Key in Serial Number

Figure 2.3: 6-month Student Evaluation Serial Number

2.3 Hardware Setup

Congratulations on successfully installing the required NI software. Hardware setup of the NI MyRIO is very simple. Once all the myRIO drivers and softwares are installed, connect the NI myRIO with the USB cable given in the box. Make sure you have powered the NI myRIO module and that the POWER Led indicator has turned Blue.

As the NI MyRIO is connected, a NI MyRIO USB monitor screen is displayed as shown in Figure 2.4. Most of the myRIO devices are loaded with older firmware as highlighted by the “Version mismatch” warning, click on “Launch the Getting Started Wizard” to update the firmware. If you do not see the “Version mismatch” warning, the firmware is compatible.

A tutorial on How to start programming the myRIO with LabVIEW with few examples are explained in the myRIO B document.

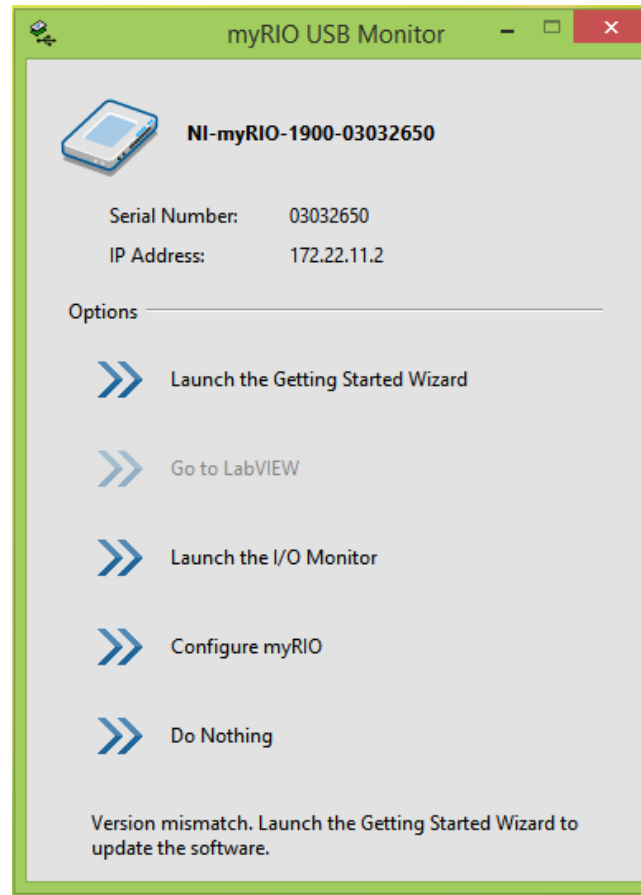


Figure 2.4: myRIO USB Monitor

2.4 Additional Resources

- <http://www.ni.com/myrio/setup/>
- <http://www.ni.com/white-paper/14620/en/>
- https://decibel.ni.com/content/community/academic/products_and_projects/myrio



Document B: Getting familiar with myRIO

3	Getting Started with myRIO	19
3.1	Initial Setup	
3.2	LabVIEW Environment for myRIO	
3.3	Analog Input and Analog Output (Express VI)	
3.4	Accessing OnBoard Devices	
4	Some Examples on myRIO	37
4.1	Accelerometer	
4.2	Motion Sensor	
4.3	Clap Sensor	
4.4	Microphone Sensor	
4.5	References	

In this Document B, students will be going through how to get started with LabVIEW programming for the NI myRIO. This document will also explain the Vis necessary to use the on board accelerometer of the NI myRIO. These examples will be useful for the hands on CA (stated in Document C).

Chapter 1 shows the steps to get started with NI myRIO. Some basic steps in launching the LabVIEW for myRIO, settings will be explained in this chapter.

Chapter 2 will explain some working examples on the NI myRIO. Particularly the accelerometer and audio ports will be explained in this chapter. These exercises are essential before students can embark on the hands-on CA (stated in Document C) to be carried out from Week 6 onward.

More working myRIO examples can also be found in the NI website: <http://www.ni.com/myrio/setup/getting-started/>

3. Getting Started with myRIO

3.1 Initial Setup

Once you have installed all the drivers of the myRIO device, plug the myRIO device to the PC using the USB cable.

1. Launch the Getting Started Wizard to test the on-board devices in the NI myRIO. Make sure that the launch wizard should detect that the NI myRIO device is connected. If it fails to recognize, click the refresh button and try again. On successful device recognition, a following window should appear.

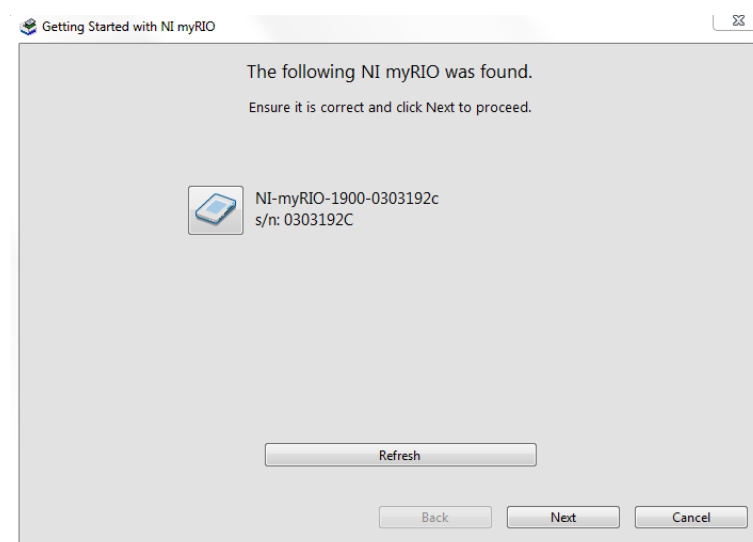


Figure 3.1: Getting Started with NI myRIO

2. Clicking next would then establish connection to the target myRIO device. A prompt window to change the name of the device would appear. Carry on with the default name and click next.

This would then first test the date and time of the NI myRIO device. If all is well, a window to test the onboard devices will appear which looks like the following figure.

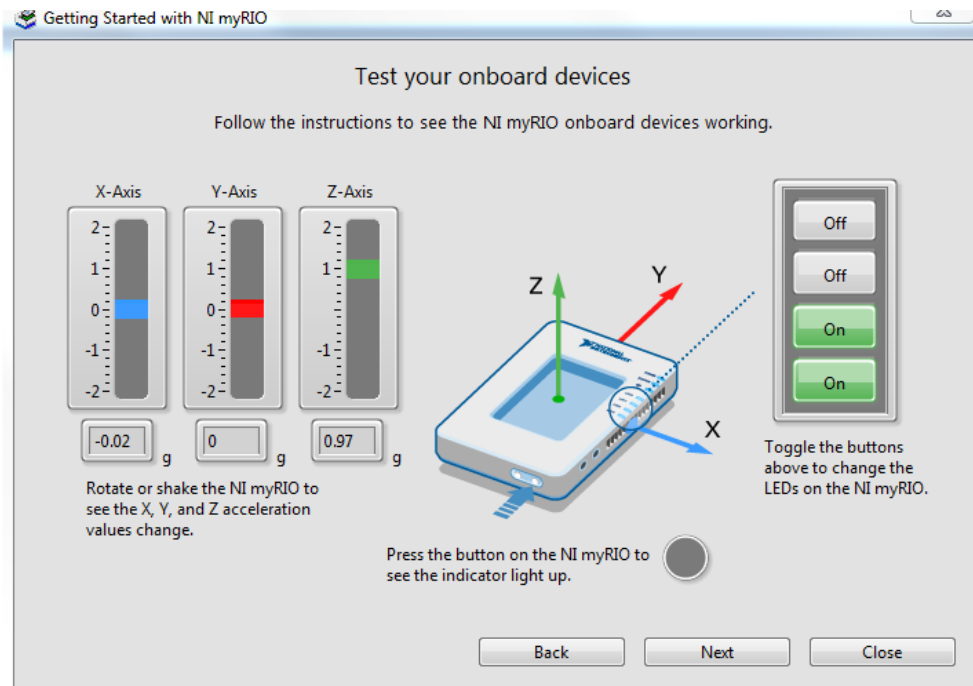


Figure 3.2: Hardware test interface

Shake the NI myRIO to ensure that the acceleration values in the X, Y and Z direction are changing. There are 4 LED's namely LED0, LED1, LED2, LED3, in the myRIO device. The LED's can also be tested by toggling the buttons in the window to ensure that all the LED's are working.

Now that you have configured your NI myRIO and tested the onboard devices, you can create your own applications on the NI myRIO. These tutorials presented in the next section of this document show you how to create your own application to test the NI myRIO onboard devices and how to configure WiFi on the NI myRIO.

3. Its time now to test your own applications to test the onboard devices of MyRIO. Launch LabVIEW and create a new project from the getting started window by proceeding with "Start your first project now". This should launch LabVIEW and the getting started tutorials in your browser.

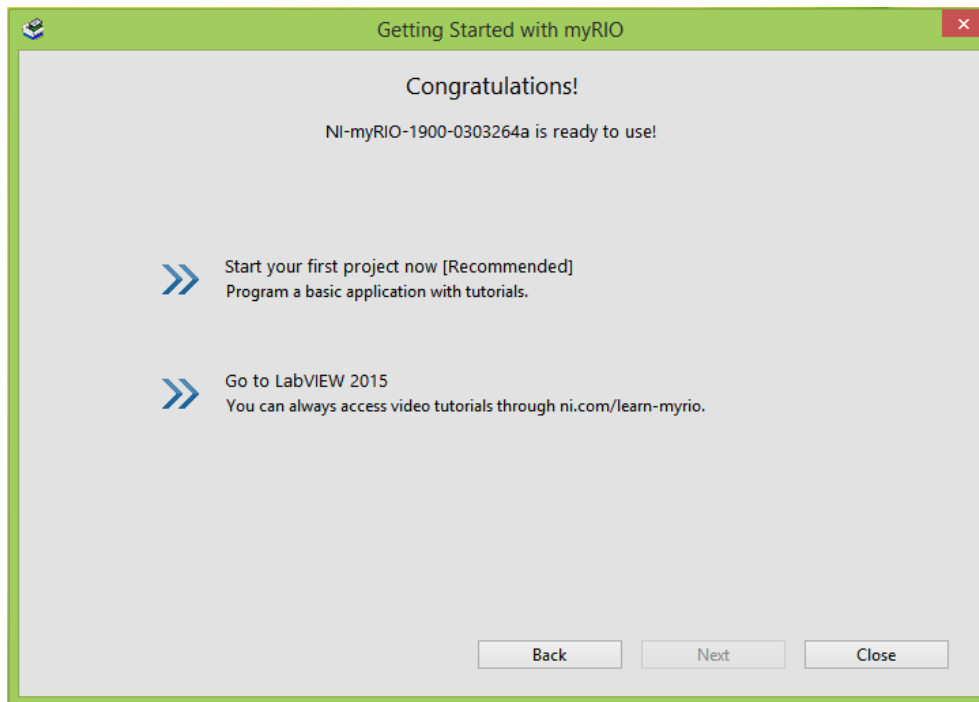


Figure 3.3: Launching LabVIEW and Getting Started Tutorials

NOTE: You can always revisit these tutorials by clicking the **Set Up and Explore** link on the **Getting Started** window in LabVIEW and selecting **Access Getting Started Tutorials** (Figure 3.4).

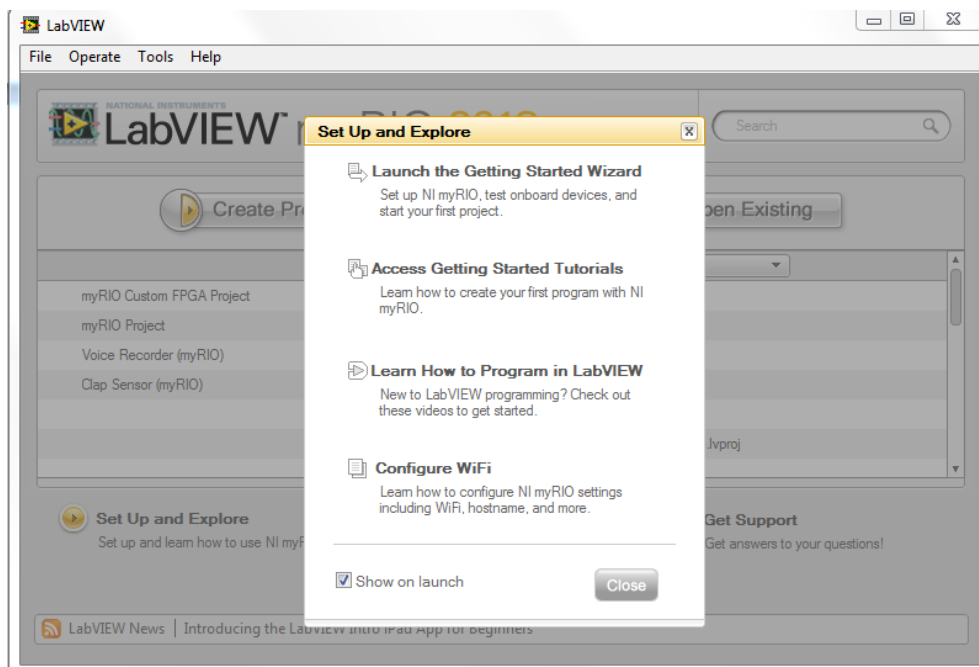


Figure 3.4: LabVIEW “Set Up and Explore” Window

4. **Select Templates >> myRIO** and then select the myRIO Project template in the **Project List**

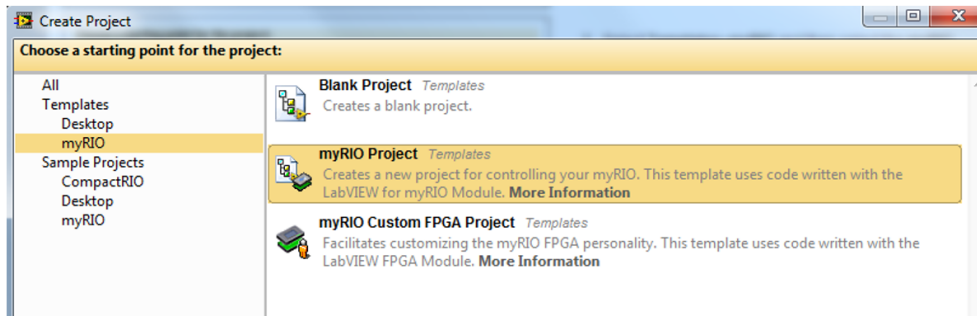


Figure 3.5: Creating myRIO Project

5. Enter the name of the project name under Project Name, e.g. “My First myRIO Application”. Specify the directory into which you want to save the project under Project Root. LabVIEW places all the project files and VIs in this directory. You can even optionally enter a prefix that can distinguish the copy of the template you create from additional copies of the template you may create later under File Name Prefix, e.g., “Tutorial-”. Finally In Select Target, select the NI myRIO that you connected to your computer. Select also the mode of connection to the PC. If the myRIO is connected using a USB, then choose “Plugged into USB”. Or if the myRIO is connected via WIFI, choose “Connected over WiFi”. Click Finish. LabVIEW saves the project and opens the **Project Explorer** window.

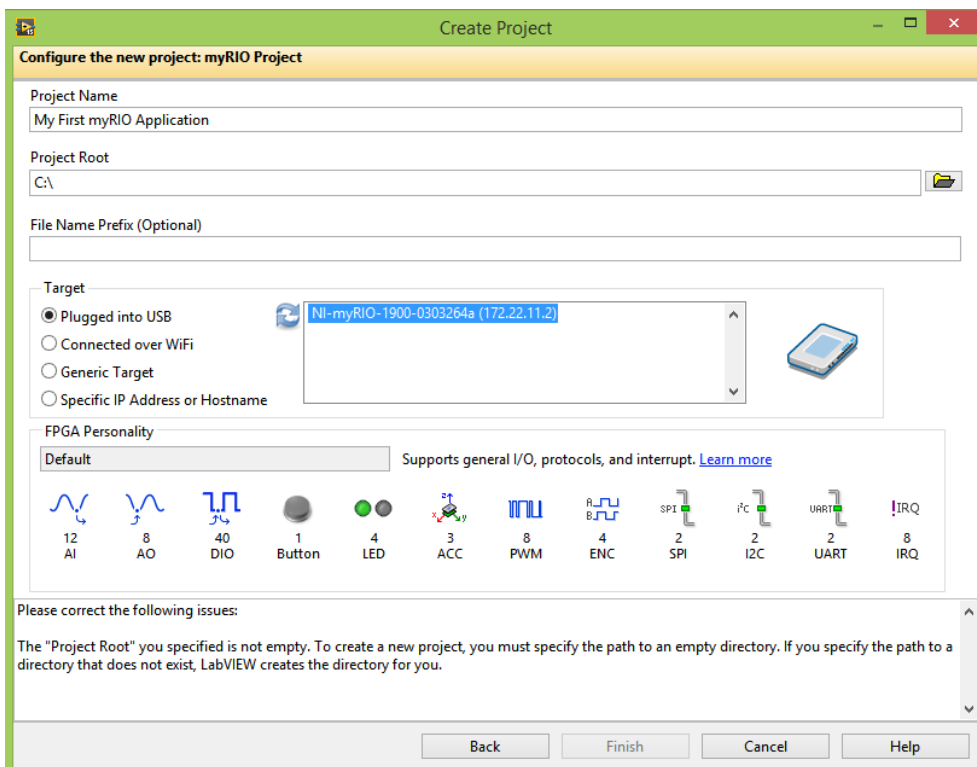


Figure 3.6: Configure new myRIO project

6. Explore the Project Explorer window. If you want to learn more details about this myRIO project, open myRIO Project Documentation.html under Project Documentation.

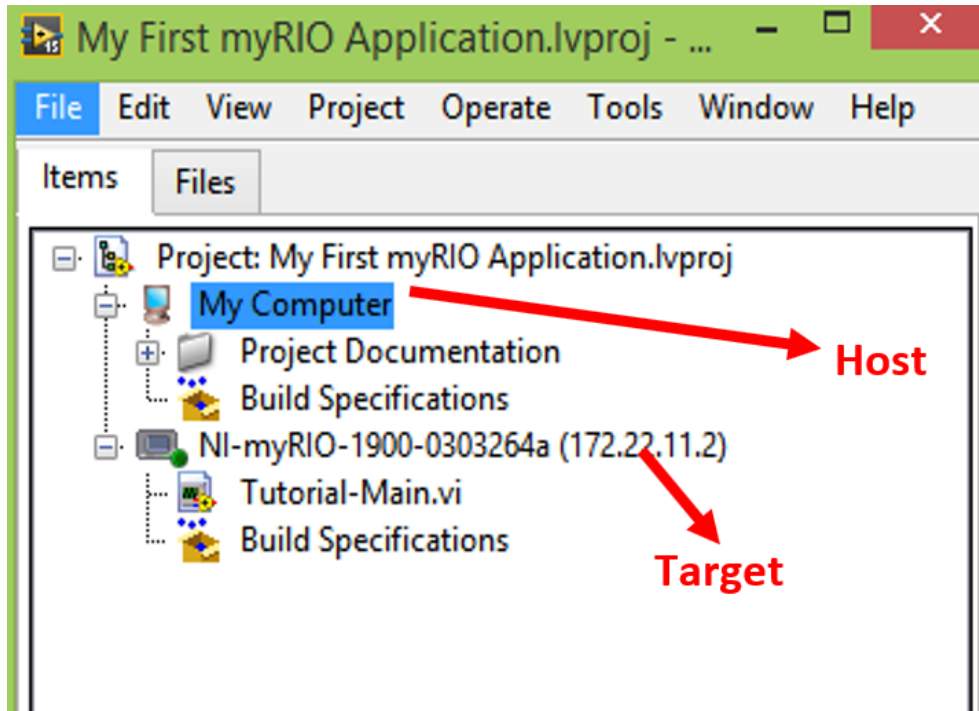


Figure 3.7: Project explorer window

In the project explorer window, shown in Figure 3.7, you can see the host device (your computer) and the target device, which is the myRIO connected. Note that the numbers 172.22.11.2 within the bracket is the IP address of the target device. A default main VI is automatically created under the target device.

You have successfully created a myRIO project. Let's next look at some of the special software VIs in the myRIO module which will help you get started with some quick programming.

3.2 LabVIEW Environment for myRIO

LabVIEW is a high level graphical programming environment created by National Instruments. By the end of this document, you would have learnt the basics of a new programming language that can control almost all National Instruments devices.

Double-clicking on the Main.vi under the myRIO target in the Project Explorer window (Figure 3.7) will launch the VI in LabVIEW and bring you to the front panel and also the controls palette.

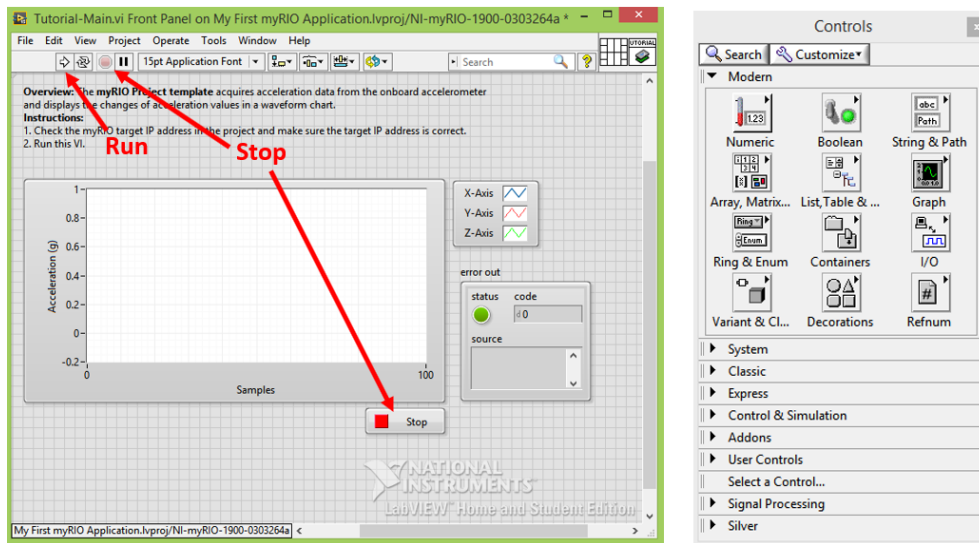


Figure 3.8: Front Panel and Controls Palette

IMPORTANT: If you are not familiar with LabVIEW you can get kickstarted by watching three short videos about the LabVIEW environment here: <http://www.ni.com/academic/students/learn-labview/environment/>

Similar to the embedded systems you are familiar with in EE3002/IM2002, the LabVIEW code has to be compiled and downloaded to the target device (myRIO). This is executed by pressing on the run button highlighted in Figure 3.8. You can stop the program by pressing the red circle near the run button or press on the stop button in the Front Panel window.

Now that you have a better idea on how LabVIEW works, let's see what's new for myRIO. In the LabVIEW front panel, switch to the block diagram view by pressing <Ctrl-E>. A right click would bring a functions palette or press **View >> Functions Palette**. We can see a myRIO function palette to build applications using the myRIO target device.

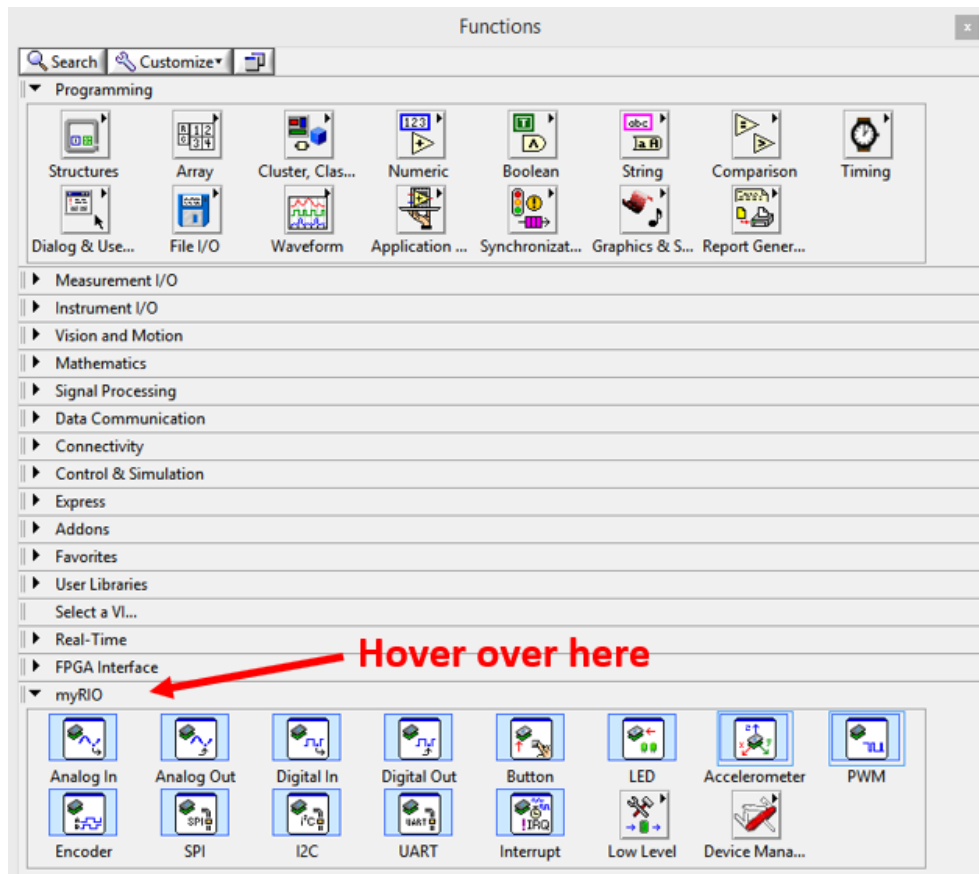


Figure 3.9: Functions palette

Let's see what are the express VIs that are unique to the myRIO. Bring up the context help by pressing on <Ctrl-H>. Hover over the "myRIO" word in the functions palette and click on "Detailed help" as shown in Figure 3.10. You will be brought to the help documentation and be shown a summarized description of the express VIs and more advanced VIs available to the myRIO like in Table 3.2 and Table 3.2, respectively.

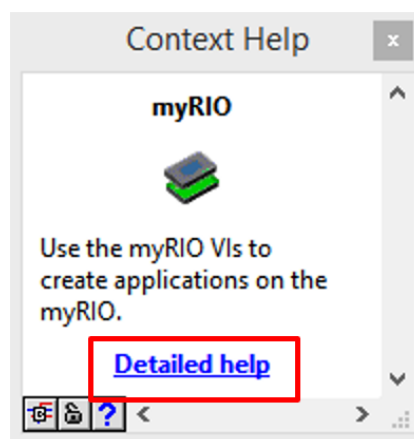


Figure 3.10: Context Help window

TIP: The context help is a very useful tool to see quick explanations on the items in the panels/palettes.

Palette Object	Description
Accelerometer	Reads acceleration values along the X, Y, and Z axes of the accelerometer on the myRIO or the roboRIO.
Analog Input	Reads values from one or more analog input channels on the myRIO or the roboRIO.
Analog Output	Writes values to one or more analog output channels on the myRIO or the roboRIO.
Button	Reads the value from the user button on the myRIO or the roboRIO.
Digital Input	Reads values from one or more digital input channels on the myRIO or the roboRIO.
Digital Output	Writes values to one or more digital output channels on the myRIO or the roboRIO.
Encoder	Reads and decodes signals from an encoder through the encoder channels on the myRIO or the roboRIO. This Express VI reads the number of ticks that the encoder receives since the last counter reset.
I2C	Writes data to or reads data from an Inter-Integrated Circuit (I2C) slave device through the I2C channels on the myRIO or the roboRIO.
Interrupt	Registers analog and digital input interrupts and creates timer interrupts on the myRIO or the roboRIO.
LED	Sets the states of the LEDs on the myRIO or the roboRIO.
PWM	Generates a pulse width modulation (PWM) signal to an external peripheral through the PWM channels on the myRIO or the roboRIO.
SPI	Writes data to or reads data from a serial peripheral interface (SPI) slave device through the SPI channels on the myRIO or the roboRIO.
UART	Writes data to or reads data from a UART device through the UART channels on the myRIO or the roboRIO.

Table 3.1: myRIO Express VIs

Subpalette	Description
Device Management VIs	Use the Device Management VIs to set custom FPGA bitfiles and to reset I/O channels on the myRIO or the roboRIO.
Treatment 2 Personality VIs	Use the High Throughput FPGA Personality VIs to create applications on the myRIO with the high-throughput FPGA personality. The myRIO high-throughput FPGA personality supports high-speed analog or digital data access. You can use the high-throughput personality for audio signals and projects in need of waveform data.
Low Level VIs	Use the Low Level VIs to control the I/O channels on the myRIO or the roboRIO.

Table 3.2: myRIO special VIs

3.3 Analog Input and Analog Output (Express VI)

The LabVIEW myRIO has several express VIs which will help in developing applications where any of the analog input or output port is required.

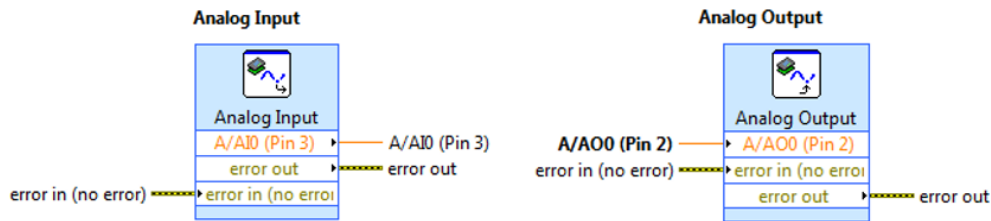


Figure 3.11: Analog Input and Output Express VIs

The Analog I/O express VI can be found on the myRIO function palette. The pins can be chosen to be any of the analog pins of the myRIO module. This can be done by double clicking the object as shown below.

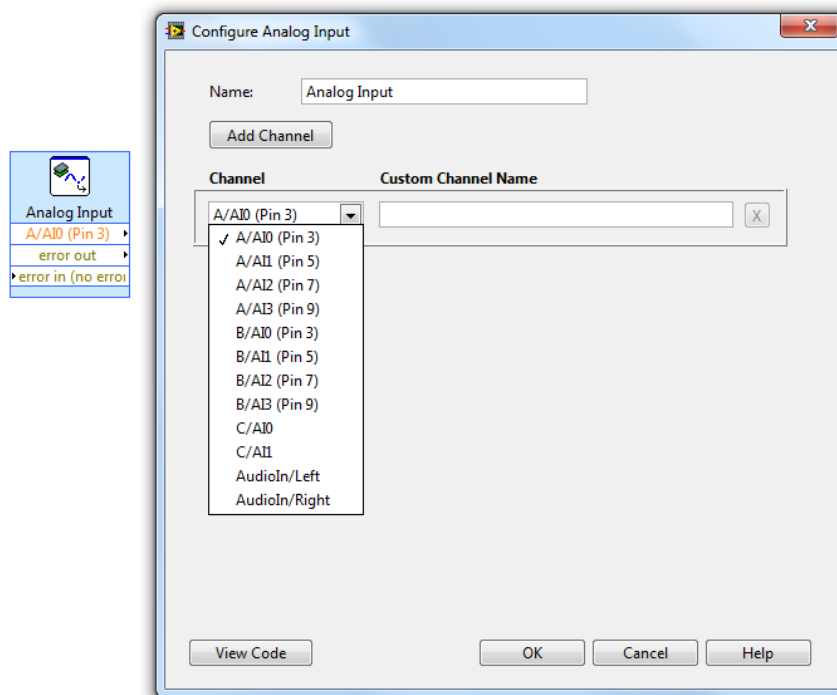


Figure 3.12: Analog input configuration

The channel can be selected to any of the 12 analog I/O pins from the drop down menu in the Configure Analog Input window. Note that the last two analog pins in the menu refers to the Audio IN and Audio OUT jacks of the myRIO device.

An express VI is a VI which is made up off several low level VIs. One can press the View Code button to view the code for this express VI. Sometimes it is better to use these low level VIs to program.

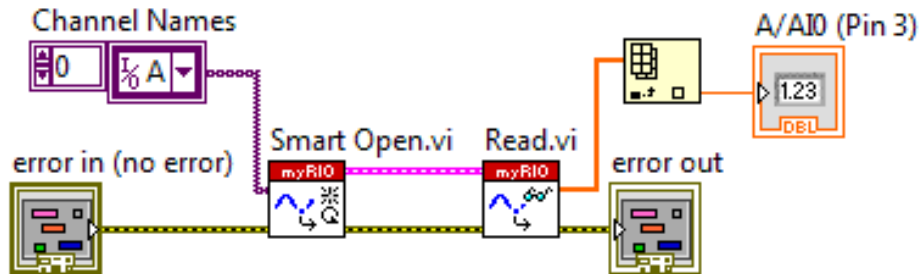


Figure 3.13: Underlying code of Analog Input Express VI

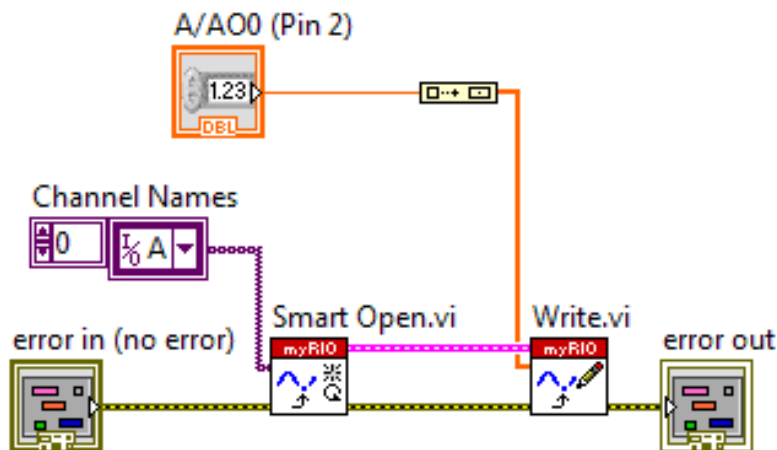


Figure 3.14: Underlying code of Analog Output Express VI

Channel Names: This just indicates the channel name as configured in the configure analog input window. **Error in:** Error in can accept error information wired from VIs previously used. You can right click on the error block on the front panel and click on **Explain Error** to know more about it. **Smart Open:** This opens the analog channel to read the contents from the analog pin. The Smart Open is a special function because this opens the instance of the analog pin only once, so that this can be used in a loop and avoid opening the port at every execution on the loop.

Read: This function just reads the contents of the analog pin. The reference of the pin should be passed on to this function from the open function. **Write:** This function writes values to the analog output pin. The reference of the pin should be passed on to this function from the open function.

The values read are in the form of an n-dimensional array. The values are then indexed to output the array element wise.

Error Out: The error out is required to transmit the error message in case of any error.

In a similar way, other express VI's like the Digital Input, Output, Encoders, PWM, UART modules can be used. The details of all these VI's can be seen by clicking the "Show context help" under the "Help" menu while hovering over the VI. We will next look at accessing the onboard devices of the myRIO module.

3.4 Accessing OnBoard Devices

There are three Onboard Devices in myRIO which can be programmed and tested from the LabVIEW interface, which is shown in the functions palette in Figure 3.9:

1. Accelerometer
2. Button
3. LED

3.4.1 Accelerometer Express VI

The accelerometer reads acceleration values along the X, Y, Z axes of the accelerometer on the NI myRIO device.

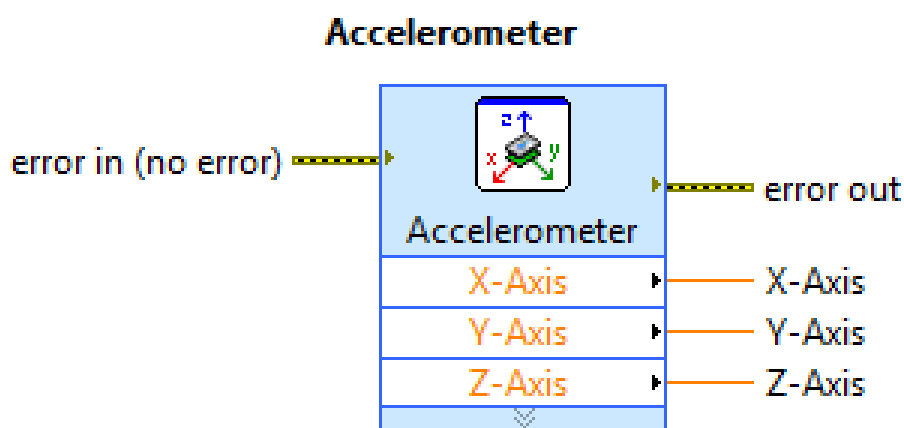


Figure 3.15: Accelerometer Express VI

You can further configure the accelerometer by double clicking the accelerometer.

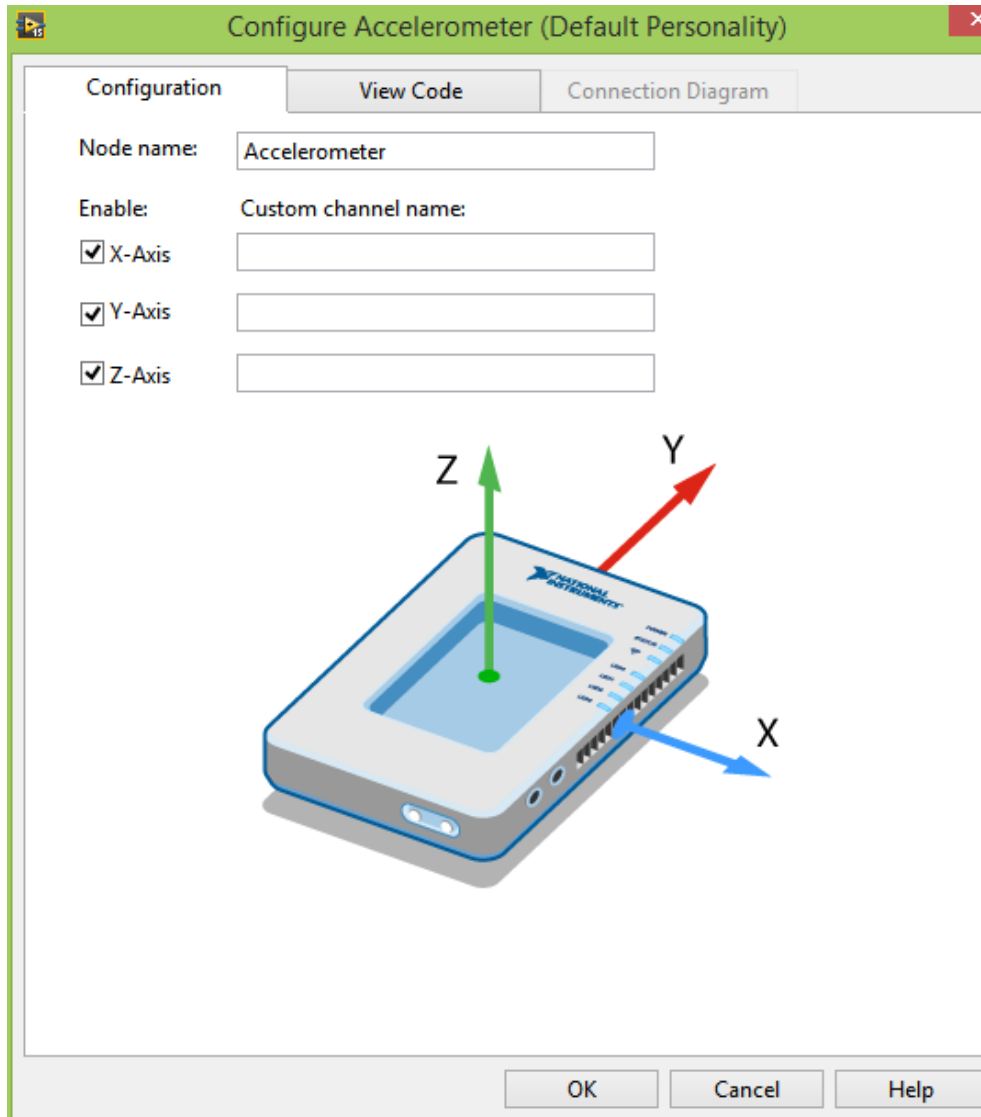


Figure 3.16: Configure Accelerometer

Let us look deeper into the code of the accelerometer express VI. On clicking “View Code”, we can see the low level VIs used to build this express VI’s.

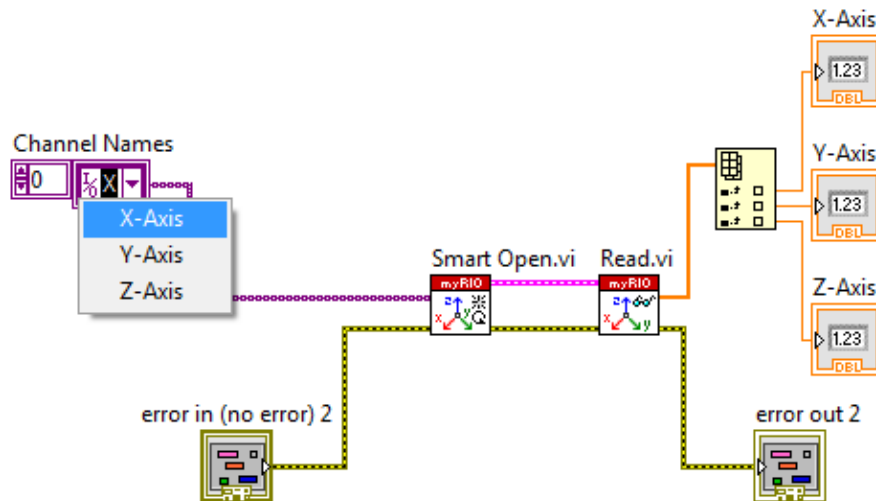


Figure 3.17: Underlying code of Accelerometer Express VI

The X-axis, Y-axis or the Z-axis can be set at the channel names indicator. Once the axes is chosen, the “Smart Open” function opens the reference of the accelerometer axis and then the “Read” function reads the values of the accelerometer values. However, the reference of the accelerometer should be passed by the Smart Open to the Read function. The index array element splits the 3-dimensional array to 3 different arrays and sends it element wise to the indicators X, Y and Z axis. However, in order to read the values continuously; the whole express VI should be within a loop which runs iteratively.

3.4.2 Button Express VI

The NI myRIO device has an external button which can be utilized to give any kind of control to the LabVIEW function. The Button express VI can be used to read the status of the button at any point of time. The NI myRIO button is a non-latching push switch. It returns TRUE when pressed otherwise returns FALSE.

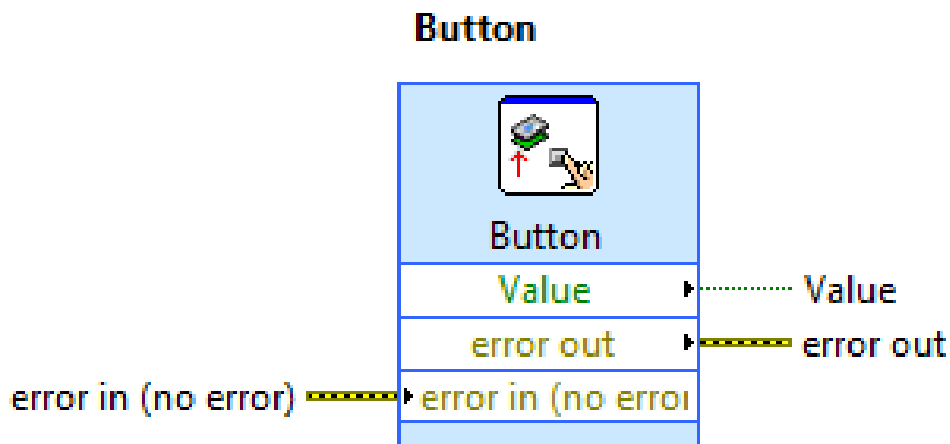


Figure 3.18: Button Express VI

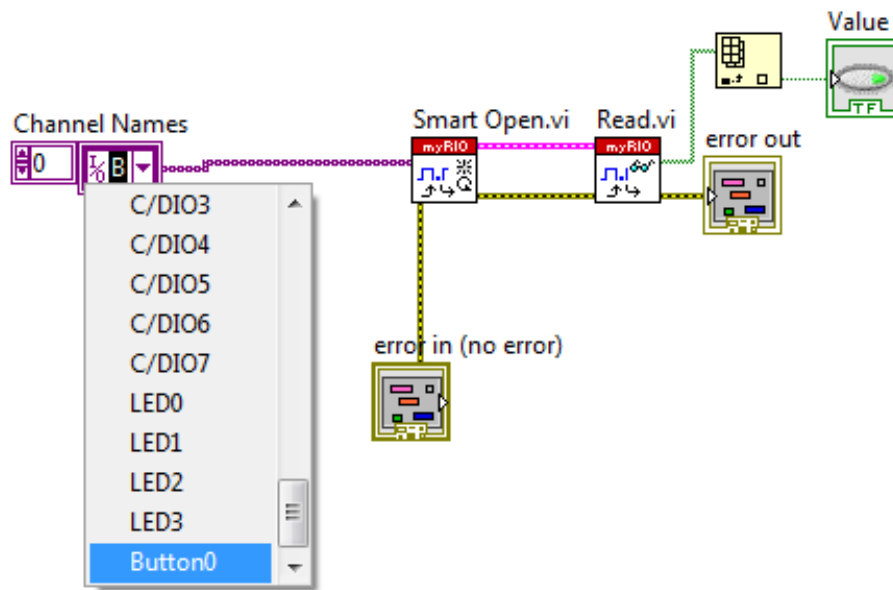


Figure 3.19: Underlying code of Button Express VI

The Button 0 can be chosen in the Channel Names indicator. Like the other accelerometer example, the “Smart Open” function is required for enabling the reading of the button status. The “Read” function will read the status of the button and can be seen as a toggle LED indicator in the front panel.

3.4.3 LED Express VI

The LED function sets the state (either ON or OFF) of the 4 LEDs on the NI myRIO device. The LED's on the device are labeled as LED1, LED2, LED3, LED4.

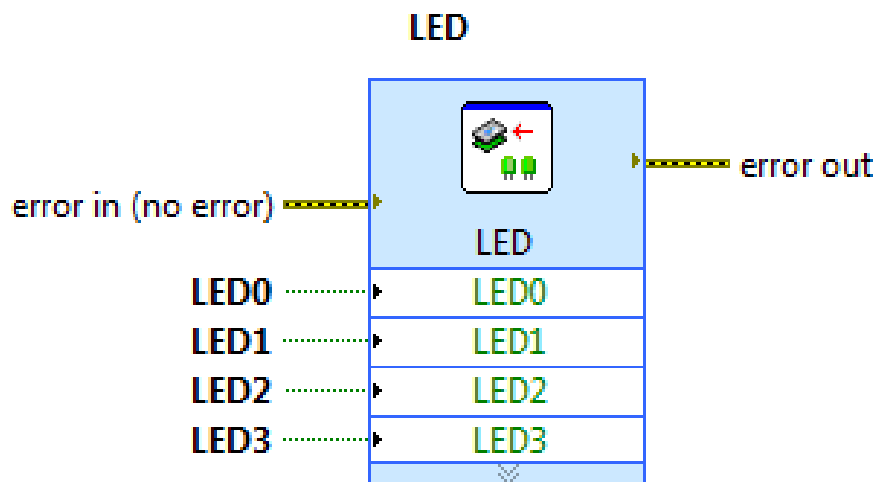


Figure 3.20: LED Express VI

Again, the low level VIs in the LED express VI can be seen by clicking “View Code” in the “Configure LED” window.

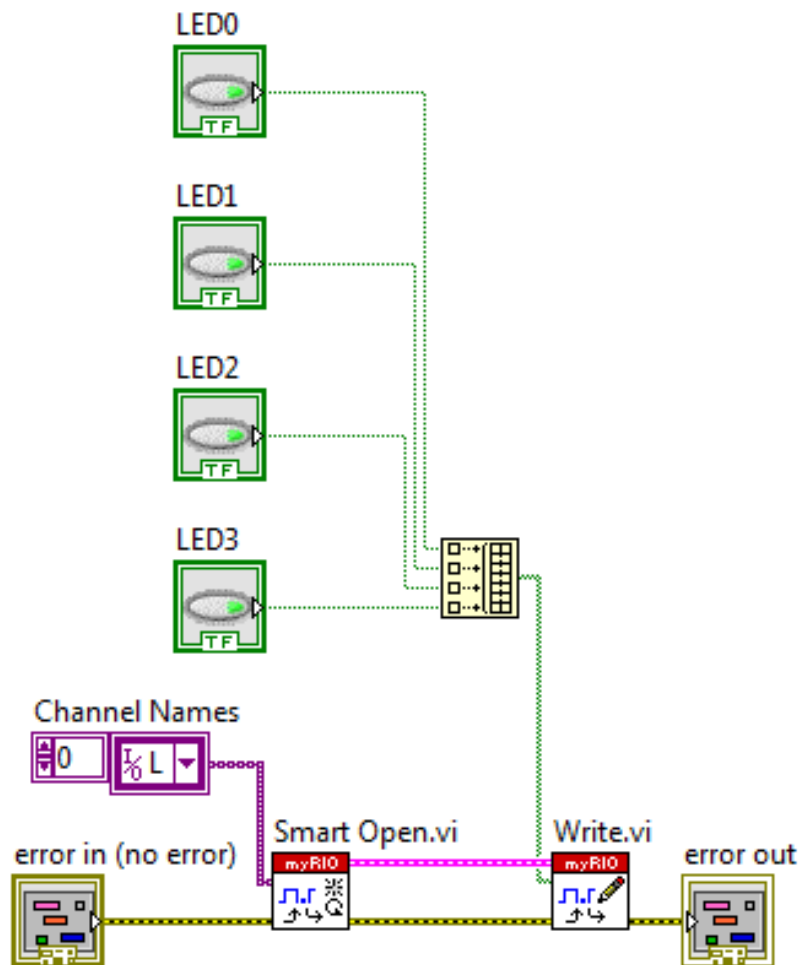


Figure 3.21:

The LED's are given as channel names and the reference of the LEDs are passed on to the Write function. LED 0, 1, 2 and 3 are external toggle buttons on the front panel to switch the LED ON or OFF. The toggle button values are then written on to the Write function and is reflected on the on-board LEDs of the NI myRIO device. This is useful in many applications, where some indicators can be displayed on the LEDs of the myRIO.

3.4.4 Concept of a Shared Variable

Often it is required that a variable is to be shared out of the VI. This is needed especially in myRIO programming for several audio applications. Several of the sound functions which is available in the Sound and Vibration toolbox in LabVIEW is not compatible with the myRIO device. For example, one cannot read a .wav file in a LabVIEW myRIO VI.

To read a .wav file, one has to first create a Shared Variable. Create a VI in the HOST PC (all the sound functions are valid here) where the wav file is first read and the values are all written into the shared variable. This shared variable can then be read inside the myRIO Target VI. It should be noted that both the host VI and the target VI's should be run simultaneously.

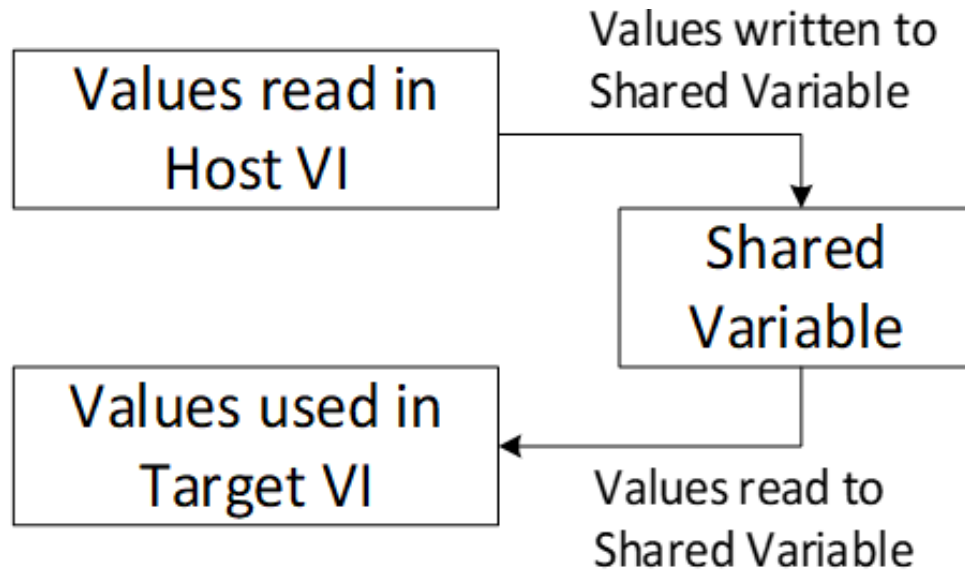


Figure 3.22:

In a case that you need to read the acceleration values in myRIO and share it with the Host VI, you may read the values in the real time target and then transfer it to your Host PC using network communications. Steps to follow:

1. Create a project.
2. Add your myRIO target.
3. Create a shared variable in your project under the myRIO Target. Go to the Project Explorer (Figure 3.7). Right click My RIO target >> New >> Variable to create a shared variable.
4. Create a VI in your RT target (Project Explorer >> Right click My RIO target >> New >> VI). In the block diagram, use the same shared variable in write mode to share the acceleration values.

In the setting of the shared variable, note that the variable type should be “Network Published” and the data type should be the same as the values to be stored in this case being “Double” as the acceleration values are of data type “Double”. These properties can be changed by double clicking the shared variable in the project explorer. All the other settings should not be changed.

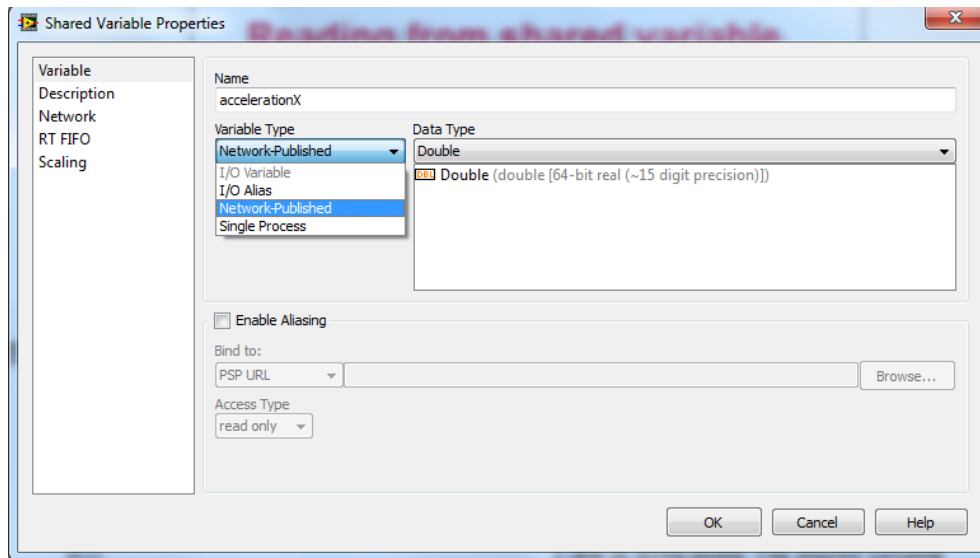


Figure 3.23: Shared Variable Properties

5. Create new Host VI (Right Click My computer»New»VI). Use the shared variable in read mode in this VI. The advantage about the Host VI is that several of the sound functions can be used here which is otherwise unavailable in the LabVIEW Real-Time. Use the VIs in this API (Functions Palette»Programming»Graphics and Sound»Sound) to access the sound functions.

Target VI:

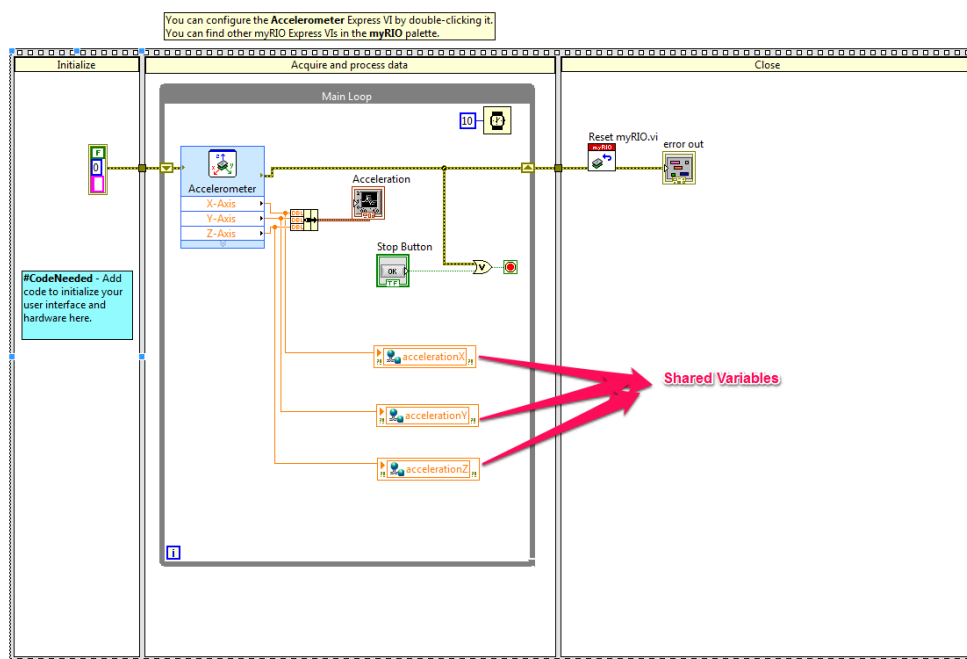
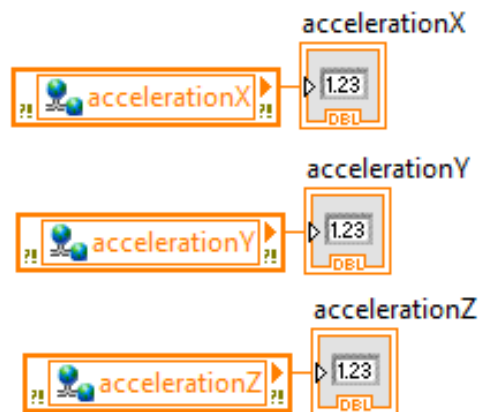


Figure 3.24:

HOST VI



Reading from shared variable

Figure 3.25:

The target and the host VI's are shown in the above figures. In the target VI, the acceleration values are read real time and stored in a shared variable in write mode. The shared variable is read in the Host VI and can be used for developing other applications.

The concept of shared variable is important and will be useful for the CA project. Please note that both the Target VI and the Host VI should be run simultaneously. This can be done running the Target VI first in continuous mode and then running the Host VI in continuous mode.

In the next chapter, we would look at more examples on the myRIO platform which will help you to get started hands on.



4. Some Examples on myRIO

4.1 Accelerometer

In this application, we will create a real time accelerometer which constantly monitors the acceleration values along the X-axis, Y-axis and Z-axis.

We will be using the Main.vi program which is created by default when you create a myRIO project in LabVIEW. Double click on the Main.vi under the myRIO target folder to open the front panel of the VI, as you have done in 1.2. Let us first analyze what all functions we would require for this particular application.

1. Accelerometer Express VI: To read the accelerometer X, Y, Z readings at every instant
2. A Timed Loop: To read the values at every “T” ms
3. A waveform: To display continuously the acceleration values of X, Y, Z axis at every instant

Meanwhile on the front panel shown in Figure 3.8, an instance of the waveform, error indicators and the stop buttons are automatically created as the functions are created in the block diagram window accessed through <Ctrl+E>.

Let us look through each function in the block diagram window and understand its working.

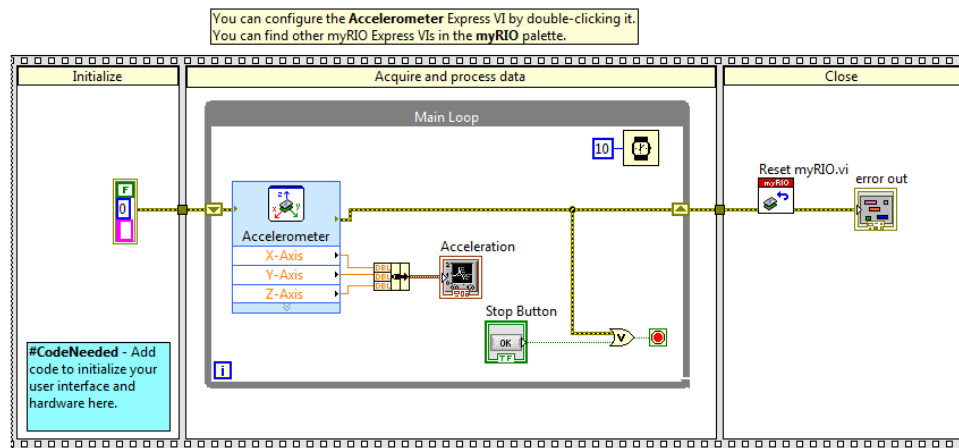


Figure 4.1: Block diagram window

The whole code is placed in a sequential block: Initialize, Acquire and process data, Close being the three parts of the function.

The Initialize block initializes the user interface and the hardware here. This value is passed on to the error in of the accelerometer. This is however optional but would be useful in case of any debugging.

The main part of this program is given in the block: Acquire and process data. The whole code is kept inside a while loop which repeats the code within its subdiagram until a specific condition occurs. This loop waits 10 ms between any two iterations. This means that the acceleration values are read at every 10 ms time interval. This value can be set by the “wait” function placed at the top right corner of the while loop.

The accelerometer express VI gives out the X, Y, Z acceleration values. The X, Y, Z values of the accelerometer are first bundled into a cluster to output as a waveform chart. Note that since this is a multidimensional array, the waveform will have three plots each for X, Y, Z acceleration values.

Any loop must also have a STOP condition. This loop will stop executing either the user presses STOP in the front panel or the accelerometer outputs a valid error out.

Finally in the close block, the myRIO is reset. This function can also be found in the myRIO function palette. The Reset myRIO.vi resets the FPGA target and all the I/O channels on the NI myRIO. This VI resets the FPGA target even when there is code running on the FPGA target and no matter whether there are incoming errors. This VI should only be used with the myRIO Express VIs and should be run only once at the end of an application.

4.2 Motion Sensor

Now, let us build on the previous program we described to develop a motion sensor. This example program makes use of the acceleration values to detect motion and displays it on the onboard LEDs according to the amplitude of the acceleration values of each of the X, Y, Z axis. The motion sensor example project can be launched by creating a project as shown in step 5 of 1.1.

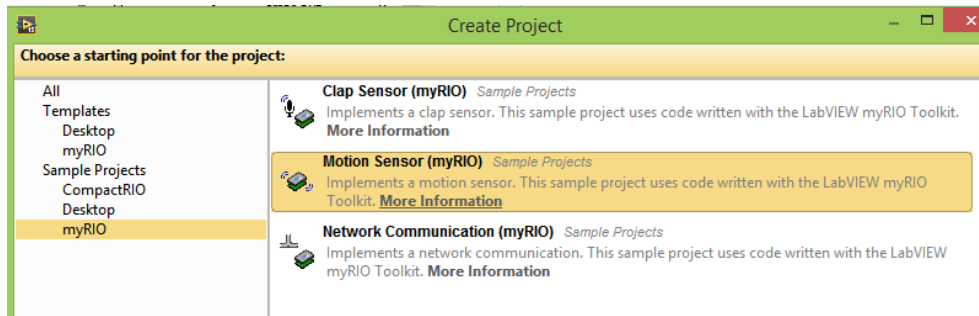


Figure 4.2: Motion Sensor sample project

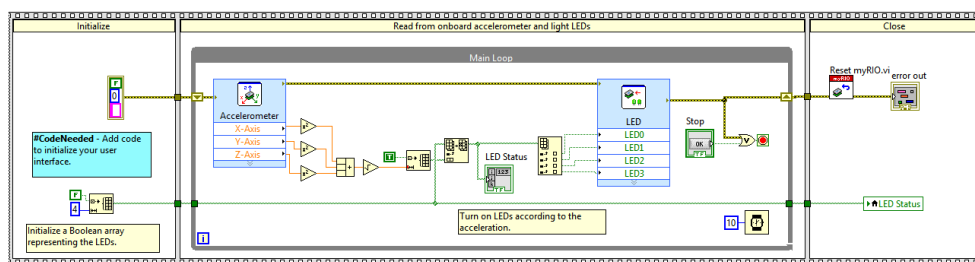


Figure 4.3: Block diagram of Motion Sensor sample

This is quite similar to what we had explained in the previous function that it has the Accelerometer express VI within a timed while loop which runs once every 10 ms. Since the amplitude of the acceleration are to be displayed on the LEDs, we would require the LED express VI explained in the previous chapter.

The initialize block again initializes the accelerometer error in values and the default initial mode of the LED status. The accelerometer X, Y and Z values are first squared and summed. The squared sum is then square rooted to obtain a representative resultant acceleration value.

This value is then stored in an array and is used to turn on the LEDs according to the resultant acceleration value. Run the program, shake the myRIO device to see the LEDs changing according to the acceleration. The decision on how to display the acceleration on the LEDs always depends on the programmer's creativity.

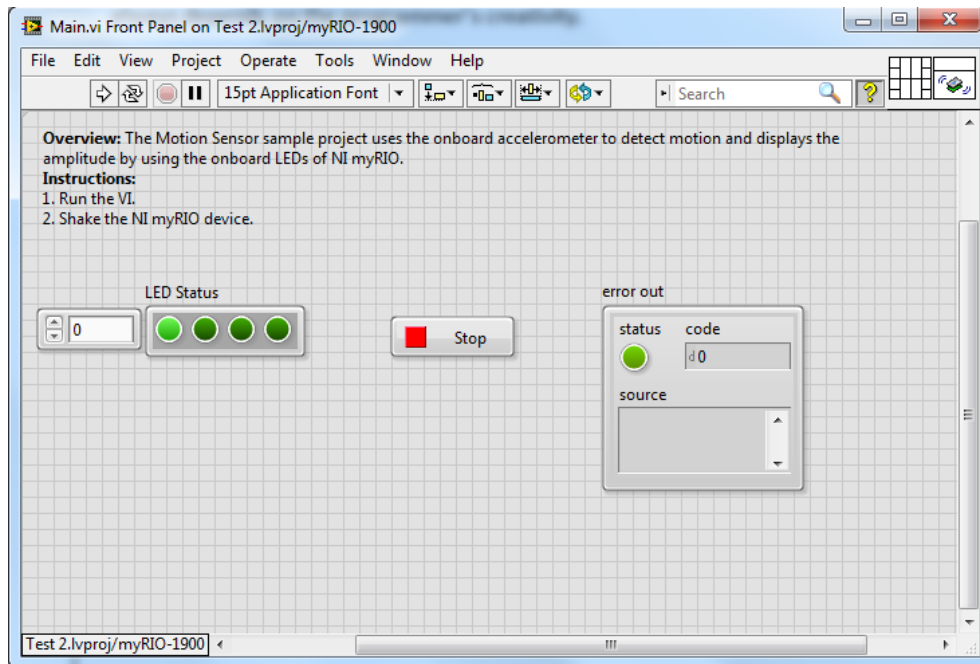


Figure 4.4: Front panel of Motion Sensor sample

In the same way, several other examples can be created. It is utmost important to know how to use the accelerometer and use it as a control to do other things. This will also be extremely useful in the CA which will be handed out by the 6th week of this semester.

NOTE: Further information on each of the functionalities can be found at the NI website and LabVIEW help. A detailed description of this sample project can be accessed by pressing on “More Information”, as shown in Figure 4.2.

4.3 Clap Sensor

Navigate to “Create Project” as you have done in 2.2 or go to File»Create Project. Launch the “Clap Sensor” sample project. This sample program requires a mono/stereo microphone in order to serve its intended purpose. It also demonstrates the concept of a timed loop.

Plug in the microphone into the AUDIO IN port and run the program. If you tap on the microphone, the LED lights should react accordingly. If there is no reaction, try adjusting the “Maximum Range” parameter in the block diagram shown in Figure 4.5 **TIP:** Your mobile phone earpiece will not work directly with the AUDIO IN port. Why? Read about it here: [https://en.wikipedia.org/wiki/Phone_connector_\(audio\)](https://en.wikipedia.org/wiki/Phone_connector_(audio))

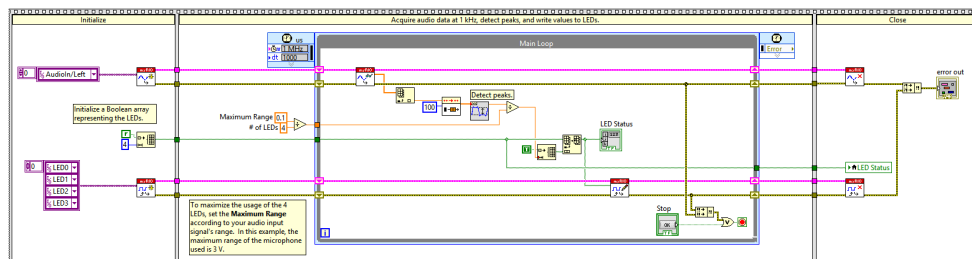


Figure 4.5: Block diagram of the Clap Sensor sample program

Similar to previous examples, the Flat Sequence executes the Initialise, Acquire Audio data at 1kHz, detect peaks, and write values to LEDs, and Close, from left to right.

This sample project uses the Timed Loop to ensure precise timing for data acquisition. The Timed Loop executes the following steps on the myRIO RT target:

1. Samples the audio input from the microphone and constructs a waveform.
2. Detects peaks corresponding to claps.
3. Turns on LEDs to display the intensity.

NOTE: Further info can be found in the same way as 2.2 by clicking on “More Information”.

4.4 Microphone Sensor

Building on the clap sensor sample. We will now modify the program of the clap sensor to display the audio data retrieved from the microphone in a chart.

1. To visualize the data stream from the audio input channel add a ‘Waveform Chart’ to the front panel via the controls panel shown in Figure 4.6.
2. Place the waveform chart down in an empty space.
3. Double click on the Waveform Chart to be brought to the chart’s location in the block diagram.
4. Wire the Waveform Chart to the output of the Read function as shown in Figure 4.7.
5. Run the program and blow into the mic to see a waveform in the chart on the front panel.

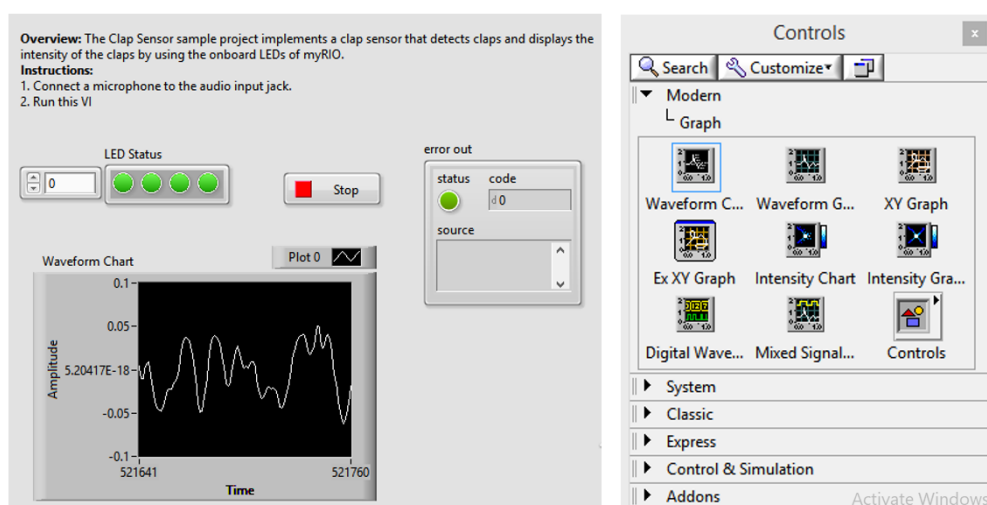


Figure 4.6: Adding Waveform Chart to the front panel

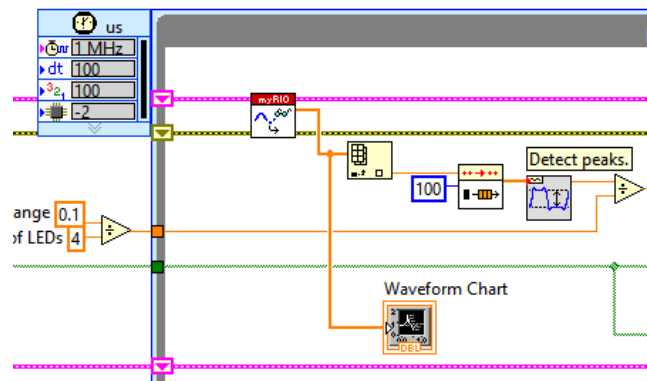


Figure 4.7: Wiring the Waveform Chart

By default, the waveform chart's axes are in autoscale mode. You can refine the display range of the waveform chart by right-clicking on the chart in the front panel and accessing the properties (Figure 4.8). Depending on the microphone type, the range of amplitude axis is different and an offset should be done if needed to remove DC bias.

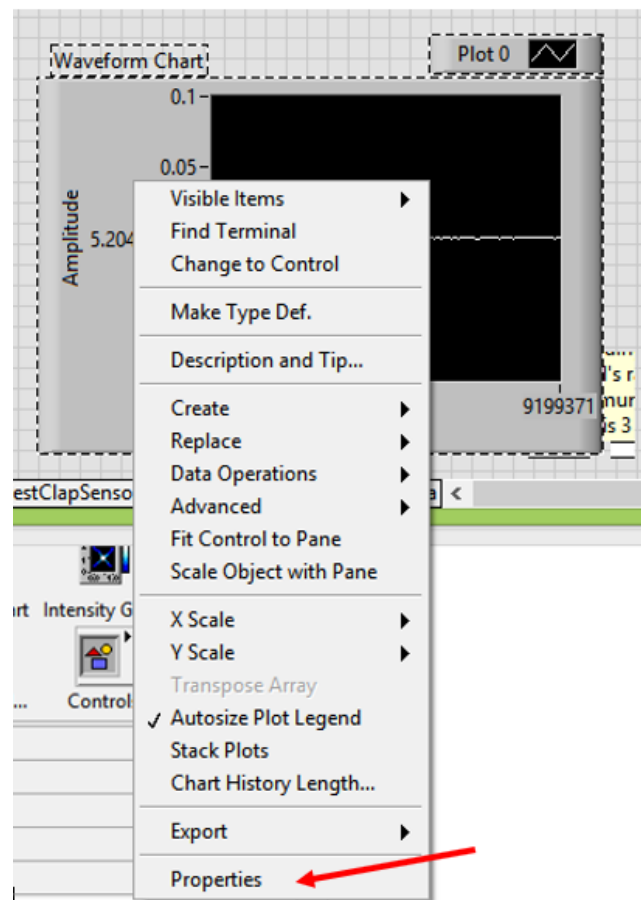


Figure 4.8: Accessing the properties of the Waveform Chart

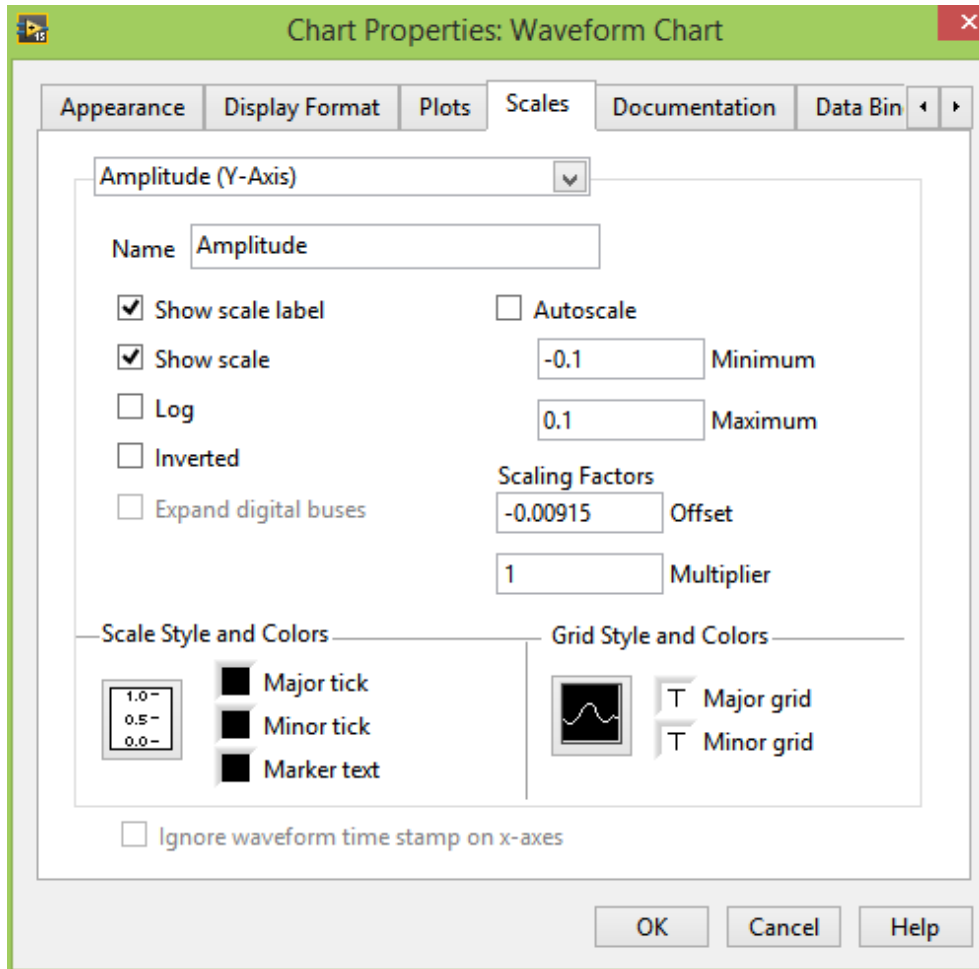


Figure 4.9: Accessing the properties of the Waveform Chart

Congratulations on modifying your first LabVIEW program! To move on to advanced projects and to complete more advanced tasks in your CA, you will need a better understanding on the LabVIEW environment, data types, structures, etc.

IMPT: Head to the NI website to learn more: <http://www.ni.com/academic/students/learn-labview/>

4.5 References

<http://www.youtube.com/user/ntspress>

<http://zone.ni.com/reference/en-XX/help/373925A-01/myrioreference/myrioreference/>

https://decibel.ni.com/content/community/academic/products_and_projects/myrio

<http://www.ni.com/pdf/manuals/376047a.pdf>

<http://www.epi.ro/myRIO.pdf>



Document C: Hands-on CA using myRIO

5	CA1:myRIO AIR MUSIC	49
5.1	Reading the acceleration values	
5.2	Sharing the values of the acceleration with the HOST VI	
5.3	LabVIEW Environment for myRIO	
5.4	Sound Synthesizer	
5.5	More (Optional) Tasks	
6	CA 2: Ambient Sensing	53
6.1	MEMS Microphone	
6.2	USB Flash Drive	
6.3	Logging MEMS microphone audio data on USB Flash drive	
6.4	Analyzing the sound file	
6.5	Optional Tasks	

In this final document C, we list TWO hands-on CA exercises, and students need to attempt only ONE. In each of the two exercises, we also offer ideas to further enhance the students' CA project. This CA is open-ended mini-project and students are free to propose new ideas to extend their CA assignment. Two students share one myRIO platform (which was introduced in Document A) and team up to work on the hands-on CA.

Alternatively, two students can propose their own project using the myRIO, but they have to put up their request through me by email: ewsgan@ntu.edu.sg and get my approval before starting with their CA (beginning of Week 6).

Extra marks will also be awarded to students' creativity and interesting ideas in attempting their CA. At the end of their CA projects, students are required to produce a 10-15 minutes of show-and-tell video of their CA project. Each student in each team presents for 5-7 minutes on their contributions and also demonstrates his/her project, while the other student helps to record his/her friend presentation. They then switch role for the remaining 5-7 minutes. To show the demo, it is better to use loudspeakers instead of earphones. This video clip will be assessed as part of the CA and marks shall be awarded accordingly. No report is required in this CA project.

Submission format: Deposit video in the Edventure website on the above date. More information will be announced in due course.

Best Show-and-Tell video of the CA project: Win an attracting digital gadget and have a chance to showcase your project in the school websites. Winners will be announced at the end of Week 14.



5. CA1:myRIO AIR MUSIC

In this hands-on CA exercise, students are required to use their notebook or PC to program the LabVIEW code, and link up with the NI myRIO via WIFI. The configuration of WIFI for the myRIO module is explained in detail in Document B. In this CA, students have to create a music synthesizer which synthesized different sounds based on the motion of the NI myRIO module. The accelerometer readings are used to control the tone generators to create different sounds. The students' creativity would be tested on how they use the accelerator values to create this music synthesizer.

Note that in this CA, the onboard accelerometer in the myRIO device is used. These accelerometer values are then passed on to the VI in the host PC to synthesize sounds and output through the PC sound card. We are not using the inbuilt sound card of the myRIO as the output of the myRIO sound card comes with inherent noise. This is also because certain important sound functions in LabVIEW are not compatible with the myRIO device. To achieve this, the variable containing the accelerometer values in the myRIO target VI should be shared with the VI which generates sound in the host VI.

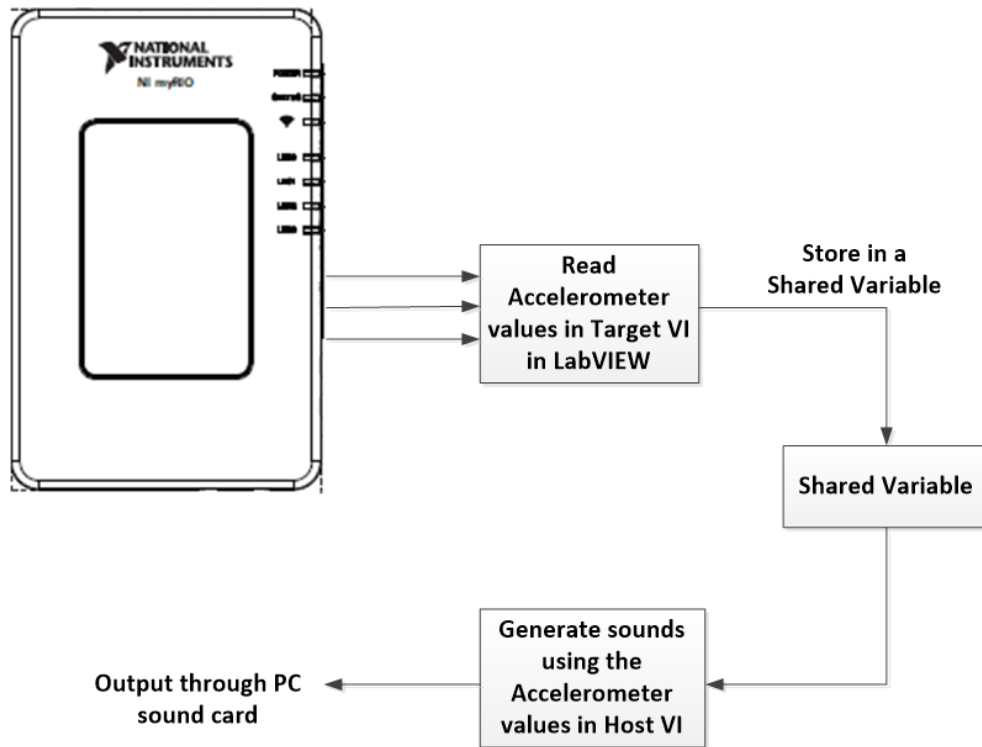


Figure 5.1:

The tasks are listed as follows:

5.1 Reading the acceleration values

In this CA, the first and foremost task is to read the acceleration values of any kind of movement of the myRIO module. This has to be done using the VI's explained in the previous Document B.

The accelerometer values along the X, Y, Z axis of the myRIO can be read using the VI. One can detect the motion is along a particular direction using the acceleration values of each of the axis.

Your ultimate task is to read the acceleration values of any particular movements of the NI myRIO device and create sounds according to these values. Students will be graded based on their creativity skills and think out of the box. All the parameters used should be clearly explained in the video presented.

5.2 Sharing the values of the acceleration with the HOST VI

The acceleration values are written on to a shared variable in the Target VI. The shared variable is a variable used to share a particular variable from the Target VI to the Host VI or vice versa. This is required in our case since the sound synthesis is carried out in the host VI and is played back through the PC soundcard.

The same shared variable is read in the Host VI to get the real time acceleration values. These values can then be analyzed as in the first task explained. The decision parameters can be used to synthesize sound.

The concept of shared variable and how to use them is explained in Document B. Note that when using the shared variables, both the host VI as well as the Target VI should be run simultaneously.

5.3 LabVIEW Environment for myRIO

This is perhaps one of the most important block in this CA exercise. A decision has to be based on what condition of the accelerometer, sounds have to be generated. An example of a decision being mapping the range of acceleration values to a set of frequencies. A case structure can then be used to send a frequency according to the acceleration values along the X, Y or the Z axis. The decision can be made either in the Target VI or in the Host VI.

Alternatively, the resultant acceleration value can be used to form a decision block and synthesize sounds.

5.4 Sound Synthesizer

The final crucial task is to synthesize the sounds based on the decision made in the previous block. Students can use different waveform generators in the sound and graphics palette to generate sounds. Examples of some waveform generators being the sine generator, sawtooth generator, triangular wave generator, square wave generator etc.

The other important thing is the frequencies, sampling rate of the generated sounds. Please note that though the human audible range is from 20 Hz to 20 kHz, we cannot perceive sounds clearly beyond 16 kHz.

The frequencies generated can also be used as the musical scale. To do this CA, you may need some basic knowledge of musical keyboard. The acceleration values can be mapped to the frequencies of the keyboard. There are 88 keys on a standard piano keyboard. The frequency range is separated into seven and one half octaves. An octave is the frequency interval between the same notes in two successive octaves. For example, the key middle A (A4) is 440 Hz. In the next higher octave, the note (A5) is 880 Hz, while the lower octave (A3) is 220 Hz. There are eight white keys and five black keys in an octave where neighboring keys are separated in frequency by the 12th root of 2 (diatonic scale). For more information, you can refer to http://en.wikipedia.org/wiki/Piano_key_frequencies.

However, it is important to note that the sound of the piano keyboard or any other instrument is not a simple sine tone generator. A trumpet, oboe, flute or an acoustic guitar sounds different even if they play the same frequency. How would you describe these sounds? One might say that the trumpet sounds lively, the oboe sounds harsh, and the guitar sounds mellow. Each note has a fundamental frequency and a set of harmonics. All musical instruments, including your voice, have a unique set of harmonics that make up their sound signatures.

A fundamental sine wave, $\sin(\omega t)$ has a fundamental frequency of ω rad or $\omega/2\pi$ Hz. A composite waveform, however, can have any number of harmonics.

Take, for example, a 'triangular' waveform, which has a fundamental frequency and two harmonics:

Theorem 5.4.1

$$\sin(\omega t) - \frac{1}{9} \sin(2\omega t) + \frac{1}{25} \sin(3\omega t).$$

Hence, your other task is to include some harmonics for the notes so that your sounds synthesized is unique. You can also emulate any other instrument. Information about harmonics of any instrument can be found on the internet.

Please elaborate more on the number of harmonics, amplitude of harmonics, and frequency range of these harmonics used for the sound synthesis in the video presentation.

5.5 More (Optional) Tasks

Students are encouraged to further explore more on myRIO and develop other techniques of using the myRIO board as a musical instrument. Sky is the limit for the creativity.



6. CA 2: Ambient Sensing

Constant monitoring of the ambient environment, such as temperature, humidity, air pollutants, and noise, is an important tool to monitor environmental stressors in urban areas. The information collected enables urban planners to make smarter decisions to reduce city dwellers' exposure to high levels of noise, for instance.

In his CA, students are to program the myRIO through a PC to record audio data using a MEMS microphone and store the recorded data in a flash drive connected to the myRIO.

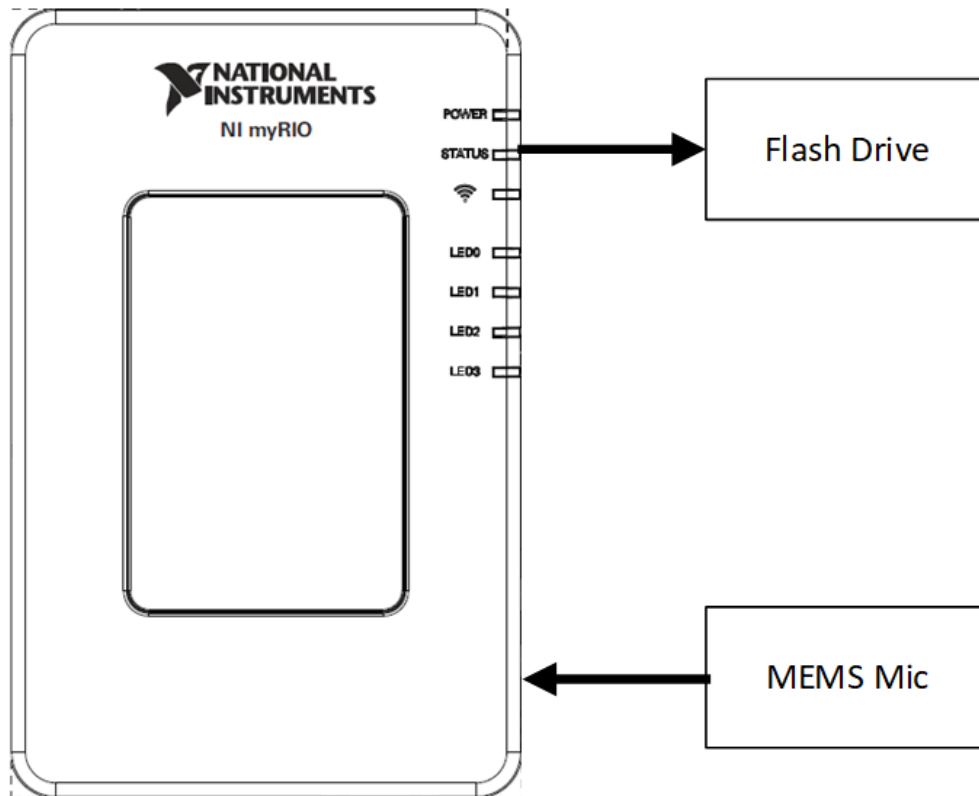


Figure 6.1: MyRIO simple audio logger

2 Tool needed:

- myRIO
- Starter Kit
- Embedded Systems Kit

6.1 MEMS Microphone

Open the NI myRIO Project Essentials Guide (project-guide.pdf). Go to chapter 34 (MEMS Microphone). Implement the MEMS microphone tutorial as detailed in the guide. You are advised to watch the video first: <https://youtu.be/991pj7yUmuY>

IMPT: The wiring of the MEMS microphone is slightly different from the guide. Look at the terminal names on the MEMS microphone in given in the Embedded Systems Kit. There are two methods taught in the guide to connect the microphone, you can use either methods.

The end result is shown in Figure 6.2.

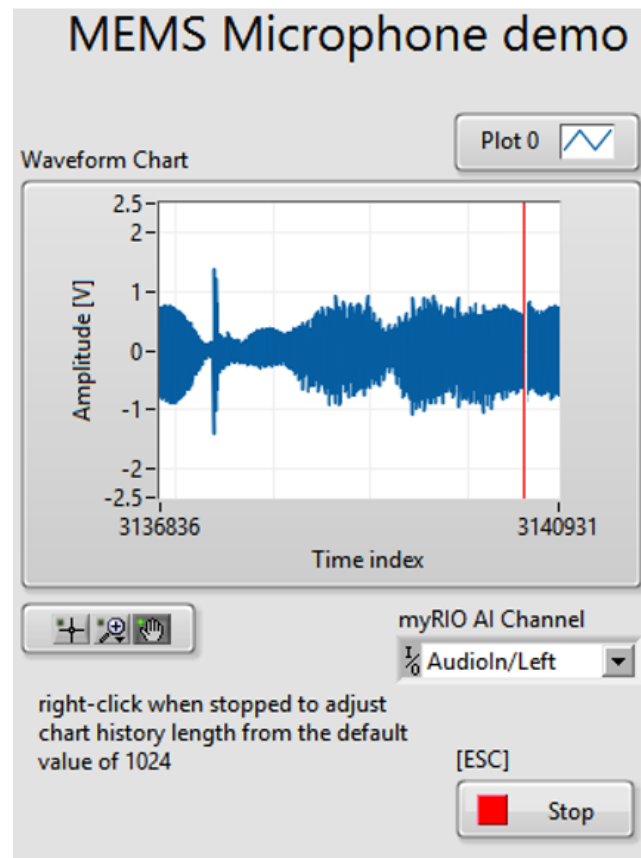


Figure 6.2: A working MEMS microphone demo

Task:

1. Adjust sampling rate (period of one iteration) and check if the time taken for one iteration is longer than specified (by you).

HINT: Look at the top left corner of the Timed Loop, expand the box with the word “Error” by dragging downwards. To get detailed information about this options, bring up the context help by pressing <Ctrl-H>.

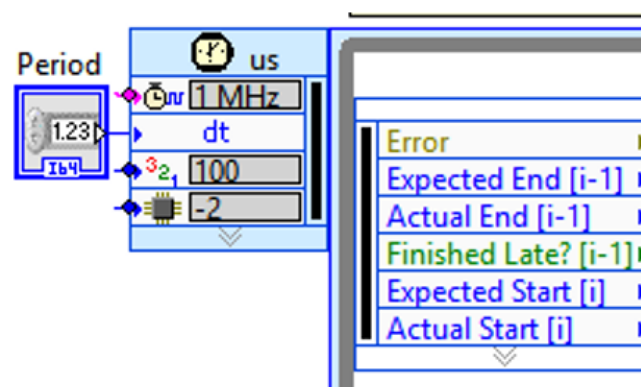


Figure 6.3:

2. Choose the lowest period possible based on your observations.

6.2 USB Flash Drive

Open the NI myRIO Project Essentials Guide (project-guide.pdf). Go to chapter 35 (USB Flash Drive). Implement the MEMS microphone tutorial as detailed in the guide. You are advised to watch the video first: <https://youtu.be/YlQukBt1lWI>

6.3 Logging MEMS microphone audio data on USB Flash drive

Modify the code in 2.1, implement the USB Flash Drive storage to store recorded audio data in the USB Flash drive. Ensure that the designated loop period is not smaller than the actual time take for one loop iteration.

Tasks:

1. Decide on the length of audio to record. This should be limited by the type of sound you want to record (e.g., traffic noise during peak hour, construction noises, etc.) and the capabilities of the myRIO platform.
2. Specify the loop to record only the time duration you want (e.g., 1 min). HINT: The stop button and iteration counts

6.4 Analyzing the sound file

In the host environment (PC), open the data file you have logged in 2.3 and perform a simple frequency analysis (FFT) on the data to see the frequencies in the audio file. Elaborate on the frequency signature of the sound recorded and its relation to the sampling rate used as well.

6.5 Optional Tasks

Besides logging audio data, the myRIO is able to handle many different types of sensors. These are some examples of extensions to CA 2.

1. Implement temperature sensors together with the MEMS microphone and modify the code to capture at a different sample rate.
2. Use the LCD screen to indicate that the myRIO is recording.