



Mechatronics and Pneumatics Kit Manual

Georgia Institute of Technology | ME2110 | January 18, 2016

1 Table of Contents

2	The Quick Start Guide	3
3	Introduction.....	4
4	The NI myRIO.....	5
4.1	Inputs-Outputs	5
4.1.1	<i>The Sensor Board</i>	6
4.1.2	<i>The Driver Board</i>	7
5	Programming the myRIO.....	8
5.1	Setting up the myRIO	8
5.2	LabVIEW Basics.....	9
5.2.1	<i>Terminals</i>	10
5.2.2	<i>Wires and Nodes</i>	11
5.2.3	<i>Basic Loops and structures</i>	13
5.2.4	<i>Running the Code</i>	15
5.2.5	<i>Debugging the code</i>	15
5.2.6	<i>Interfacing with the myRIO hardware</i>	16
5.3	Files in the me2110 template project	17
5.4	Understanding the ME2110 template	17
5.4.1	<i>Front Panel</i>	17
5.4.2	<i>Block Diagram</i>	18
5.5	Connecting to the myrio over wifi	19
5.6	Common mistakes.....	20
6	Electromechanical Components	21
6.1	Sensors	21
6.1.1	<i>Switches</i>	21
6.1.2	<i>IR Distance Sensor</i>	22
6.1.3	<i>Potentiometer</i>	24
6.1.4	<i>Encoder</i>	24
6.2	Electromechanical Actuators	25

6.2.1	<i>DC Motors</i>	25
6.2.2	<i>Solenoids</i>	27
7	Pneumatic components	29
8	Example Programs.....	32
9	Helpful links.....	35

2 The Quick Start Guide

This section will give quick references on how to get started with the myRIO. However, to fully understand myRIO and how it is being used in ME2110, you must read through the rest of the Manual.

1. Install LabVIEW using instructions from Section 5 if using a personal computer.
2. Watch the video explaining what the myRIO is and what comes with it.
<https://youtu.be/K9VAOF01gbU>
3. Download the template project from the course website. Explore the examples given. The list of channels that the sensor and driver boards use are in Sections 4.1.1 and 4.1.2 respectively.
4. Watch the video on how to write your first myRIO program.
<https://youtu.be/Ak-N8o6C7C0>
5. Watch the video that shows how to program a more involved myRIO program.
<https://youtu.be/R4tWLmBUqA8>
6. Watch the video that explains the different timing options in LabVIEW.
<https://youtu.be/diF1crkFqFo>

3 Introduction

This document describes the components of the mechatronics and pneumatics kit issued to students of ME2110 which, most importantly, includes the NI myRIO controller shown in Figure 1. You will not be handed items labeled 2,5,7,8. The controller is a portable reconfigurable I/O (RIO) device that students can use to design control, robotics, and mechatronics systems. The controller is programmable with LabVIEW and C languages. There are four electromechanical actuators provided with the kit: two DC motors, a large solenoid, and a small solenoid. In addition, a pneumatic system that contains two pneumatic actuators is included. They are controlled via solenoid valves. There are five sensors included with the kit: two switches, one IR distance sensor, one potentiometer, banana plugs and one encoder. This document includes explanations of the controller, general LabVIEW programming, and operating instructions for each item in the kit.

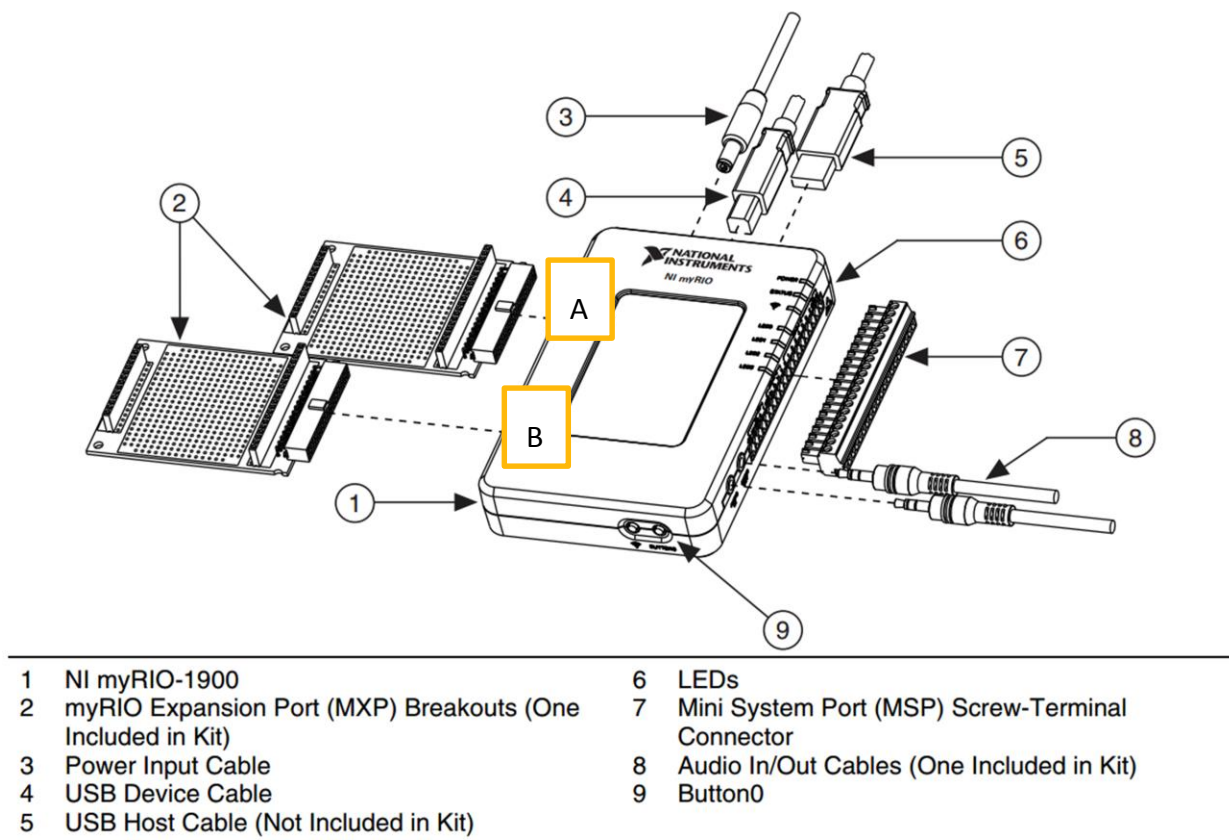


Figure 1 myRIO- 1900

4 The NI myRIO

In this course, you will be using NI myRIO hardware. The NI myRIO is an embedded hardware device designed specifically to help students design real, complex engineering systems quickly and affordably. More information can be found at <http://www.ni.com/myRIO/what-is/>.

Why the myRIO and not other platforms such as the Arduino?

The reason is to introduce ME students to a professional programming environment used extensively in the industry and more importantly in the 3000 and 4000 level lab courses in the ME curriculum. The myRIO is significantly more powerful than typical embedded controllers (Arduinos, Raspberry Pi, TI Launchpad etc) thanks to its dual core ARM processor coupled with an FPGA. In addition, using LabVIEW's toolboxes it is very simple to perform complex data processing actions like FFT's, image processing, Real-Time control, neural networks etc. ME2110 is going to use only the very basic capabilities of the myRIO, but by doing so the students will be more familiar to high level devices to use in their future studies.

4.1 INPUTS-OUTPUTS

The NI myRIO-1900 provides analog input, analog output, digital input and output, audio, and power output. The NI myRIO-1900 connects to a host computer over USB and wireless 802.11b,g,n. The MXP ports (the A and B ports), shown on the left side of Figure 1, both have 16 Digital Input-Output, 2 Analog Output, 4 Analog Input and UART communication channels. The channels can also be configured to accept an Encoder input, 3 PWM outputs and SPI and I2C communication. The MSP (the C port) labeled 7 in Figure 1 has 8 Digital Input-Output, 2 Analog Output, and 2 Analog Input channels. The C port can also accept 2 encoders. See the full myRIO manual at <http://www.ni.com/pdf/manuals/376047a.pdf> for more information about the ports.

To help students connect their hardware, two custom expansion boards have been made. These boards will be discussed in the subsequent sections.

When plugging the sensors and actuators into the boards, make sure that the wires are securely in the sockets and that the stripped parts of the wires are not touching each other (this is known as a short and is very bad in all applications that deal with electricity). Some common mistakes on putting wires into sockets are shown in Figure 2. If the wires are inserted correctly no conductive part of the wire will be exposed even when twisting the wires.

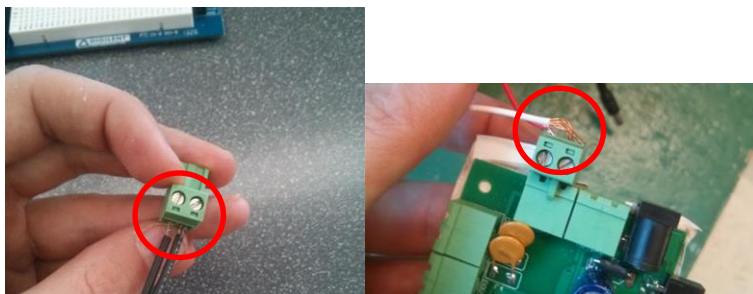


Figure 2 How **NOT** to put wires into plugs

4.1.1 The Sensor Board

The Sensor Board in Figure 3 is plugged into the B port. Table 1 shows how the ports on the Sensor Board correspond with the Channels on the myRIO. All of the sensors are to be connected with the small (2.5 mm spacing) plugs. The operating principles of the sensors are explained in further chapters.

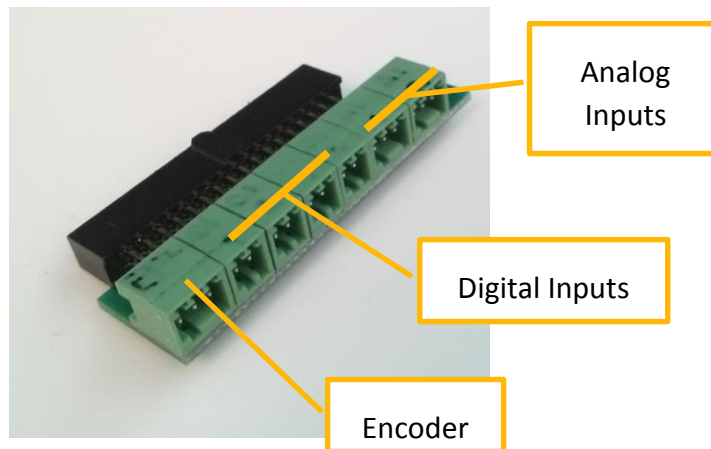



Figure 3 Sensor Board

Table 1 Sensor Board Connections

Label on the Sensor Board	Input Type	Channel	Description
ENC	Encoder	B/ENC	Encoder Input see Section 6.1.4
DI1	Digital Input	B/DIO5	Switch 1 see Section 6.1.1
DI2	Digital Input	B/DIO2	Switch 2
DI3	Digital Input	B/DIO1	Switch 3
DI4	Digital Input	B/DIO0	Switch 4
AI1	Analog Input	B/AI1	IR sensor or Potentiometer (See Sections 6.1.2 and 6.1.3)
AI2	Analog Input	B/AI0	IR sensor or Potentiometer

4.1.2 The Driver Board

The Driver Board in Figure 4 connects the myRIO to the actuators (motors, valves, solenoids). The Driver Board connects to port A on the myRIO. You can either plug the board straight into myRIO or use the ribbon cable, shown in Figure 5, to make sure that the board is secure. There are two holes on the driver board that can be used to secure it to your machine. All of the actuators are connected to with the larger (5.08 mm spacing) plugs. Table 2 shows which channel on the myRIO corresponds to the Driver Board outputs. To use the driver board the power adapter must be plugged in. Please make sure not to confuse the power adapter for the myRIO and the Driver Board as shown in Figure 6. **If some outputs on the board stop working try switching the fuses:  inside the plastic housing, if that does not work, ask the TAs for help. ~800 mA fuses are used for the digital outputs and a 4 A fuse is used for the whole board.**

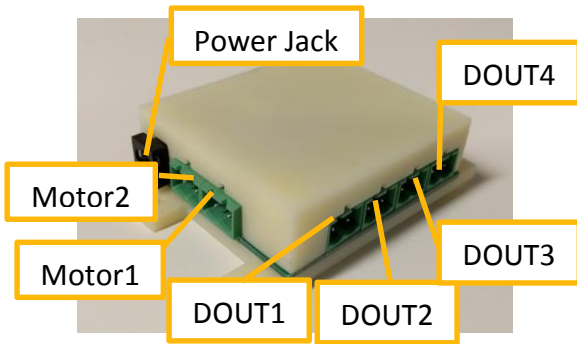


Figure 4 Driver Board



Figure 5 Ribbon Cable Connection

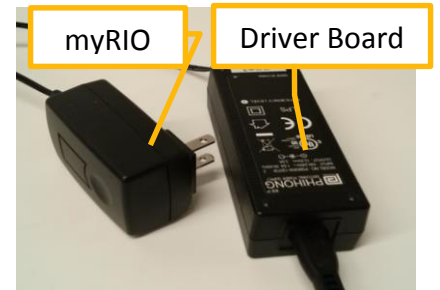


Figure 6 Power Adapters

Table 2 Driver Board Connections

Label on the Driver Board	Channel	Description
DOUT1	A/DIO1	Solenoid/Valve Connection See Section 6.2.2
DOUT2	A/DIO2	Solenoid/Valve Connection
DOUT3	A/DIO4	Solenoid/Valve Connection
DOUT4	A/DIO3	Solenoid/Valve Connection
MOTOR1	A/DIO6 & A/DIO7 for direction A/PWM2 for speed	Motor 1 Connection See Section 6.2.1
MOTOR2	A/DIO14 & A/DIO15 for direction A/PWM0 for speed	Motor 2 Connection

5 Programming the myRIO

5.1 SETTING UP THE MYRIO

1. Download LabVIEW 2014 for myRIO from [here](#), or ask any of the TAs for a DVD (You do not need to install DVD 2) or a USB stick with the installer. If downloading, make sure that the download does not timeout during the download, check the size on disk etc. When using studio computers, choose LabVIEW 2014, and not an earlier version.
2. Install the Software bundle. Get a serial number from software.oit.gatech.edu (find LabVIEW and proceed as you were going to download the program but just copy the serial number when you get to the instructions). **DO NOT update the software to maintain software compatibility with others.**
3. Connect power to the NI myRIO device
4. Connect the USB cable from NI myRIO to your PC.
5. The NI myRIO USB Monitor appears (Figure 7)
6. Launch the Getting Started Wizard and follow the instructions. Update to the recommended drivers if prompted.
7. Go to the course website and download the ME2110 LabVIEW template. Unzip **all** of the contents and place them into a suitable folder. The file **must** be unzipped.
8. Open LabVIEW
9. Click File->Open Project
10. Open the ME2110 template
11. The LabVIEW **Project Explorer** opens
12. Right Click on “myRIO-1900 (172.22.11.2)” and right-click Connect (Figure 8) ¹
13. If the process finishes successfully then everything is set up correctly and you can begin programming the device.

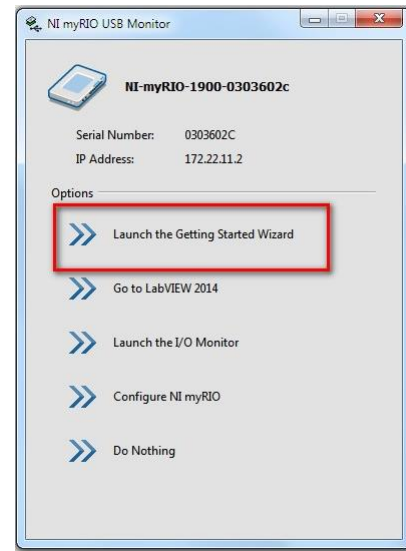


Figure 7 NI myRIO USB monitor

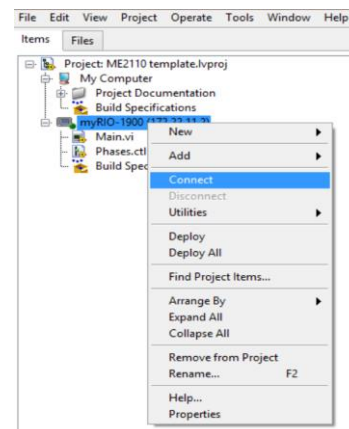


Figure 8 Connect to myRIO in the Project Explorer

¹ If you are using the myRIO over Wi-Fi then click properties and insert the correct IP address and click connect.

5.2 LABVIEW BASICS²

The provided template is written in LabVIEW (the myRIO **CAN** be programmed in C, however **no support will be given by the TAs**). You should go to <http://www.learnni.com> to get initial LabVIEW knowledge. Also, there are links at the end of the Manual that provide more useful references.

LabVIEW is a visual programming language. Execution of the code is determined by a graphical block diagram where the programmer connects different function-nodes by drawing wires. These wires propagate variables and **any** node can execute as soon as all its input data becomes available. Since this might be the case for multiple nodes simultaneously, LabVIEW is inherently capable of parallel execution.

LabVIEW ties the creation of user interfaces, called **front panels**, into the development cycle. LabVIEW programs/subroutines are called **virtual instruments (VIs)**. Each VI has two main components: a block diagram and a front panel. The front panel is built using *controls* and *indicators*. *Controls* are inputs – they allow a user to supply information to the VI. *Indicators* are outputs – they indicate, or display, the results based on the inputs given to the VI.

The block diagram contains the graphical source code. All of the functional objects placed on the front panel will appear on the block diagram as terminals (either input or output). The block

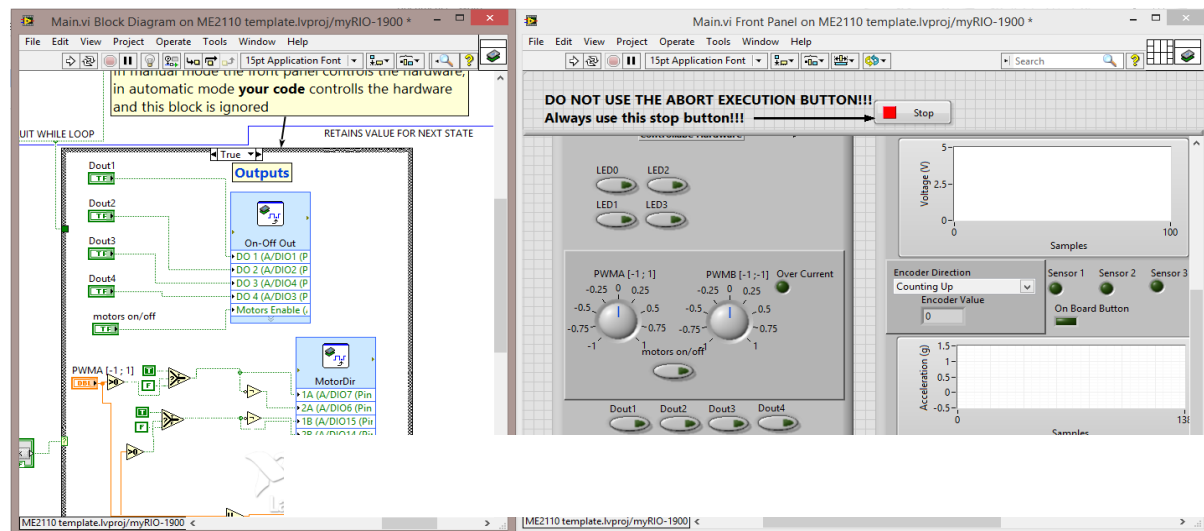


Figure 9 An Example Block Diagram on the Left and a Front Panel on the Right

² It is advised, however not compulsory, that the students complete the “Core 1” LabVIEW training by registering their copy of LabVIEW on ni.com and then choosing the module at <http://sine.ni.com/myni/self-paced-training/app/main.xhtml>.

diagram also contains structures and functions that perform operations on controls and supply data to indicators. Examples of the front panel and the block diagram can be seen in Figure 9.

The structures and functions are found on the Functions palette which will be visible by right-clicking on an empty area in either the front panel or the block diagram. Collectively controls, indicators, structures and functions will be referred to as nodes. Nodes are connected to one another using wires – e.g. two controls and an indicator can be wired to the addition function so that the indicator displays the sum of the two controls.

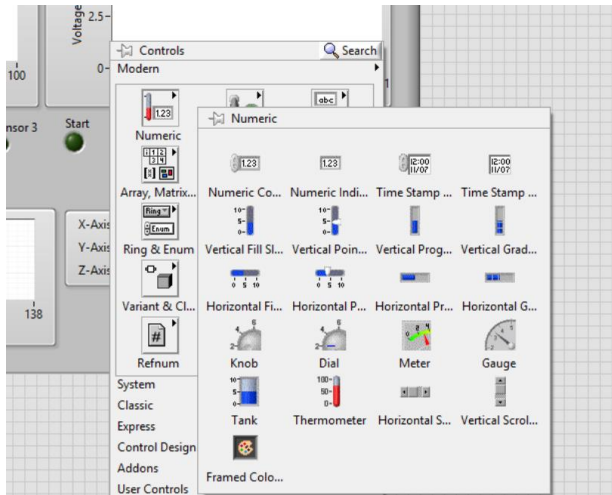
The graphical approach also allows non-programmers to build programs by dragging and dropping virtual representations of lab equipment with which they are already familiar. The LabVIEW programming environment, with the included examples and documentation, makes it simple to create small applications compared to traditional text based programming.

VIs are LabVIEW programs, and multiple VIs can be used together to make a LabVIEW application. To group these related VIs together, use a LabVIEW project, that includes references to all the LabVIEW files and non-LabVIEW files in the project, configuration information, build information, and deployment information. When you open a LabVIEW project, the **Project Explorer** opens. From there you can open all the files that are linked to the project and add new ones. *You will need to create a LabVIEW project and have the myRIO included in the project to run your VIs and download them to the myRIO.*

Tip: You can easily switch between the front panel and block diagram by pressing ctrl+e

5.2.1 Terminals

The front panel is created with interactive input and output terminals of the VI, referred to as *controls* and *indicators*, respectively. Controls are knobs, push buttons, dials, and other input devices. Indicators are graphs, LEDs and other displays. Controls simulate input devices and supply data to the block diagram of the VI. Indicators simulate output devices and display data the block diagram acquires or generates. You can get the Controls Palette as seen in Figure 10 by right clicking on an empty area. Objects on the front panel window appear as terminals on the block diagram. Terminals are entry and exit ports that exchange information between the front panel and block diagram.



Tip: You can move between the front panel and block diagram by double-clicking on an indicator or control

Figure 10 Controls Palette

The Controls and Indicators can be of the following types:

- Numeric – Can represent numbers of various types, such as integer or real.
- Boolean – Represents data that only has two possible states, such as TRUE and FALSE or ON and OFF or HIGH and LOW.
- String – Is a sequence of ASCII characters
- Other – All of the data types can be in multiple dimensional arrays and combined with other data types that are defined by “.ctl” files.

5.2.2 Wires and Nodes

Nodes are objects on the block diagram that have inputs and/or outputs and perform operations when a VI runs. They are analogous to statements, operators, functions, and subroutines in text-based programming languages. Nodes can be functions, subVIs, Express VIs, or structures. Structures are process control elements, such as Case structures, For Loops, or While Loops. You can transfer data among block diagram objects through wires. Each wire has a single data source, but you can wire it to many VIs and functions that read the data. Wires have different colors, styles, and thicknesses, depending on their data types. Table 3 shows the most common wire types. Blue lines are integers and orange lines are of the type double.

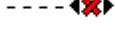
A broken wire appears as a dashed black line with a red X in the middle . To clean up any broken wires press **ctrl+b**³.

Table 3 Common Wire types

Wire Type	
Numeric	
Boolean	
String	

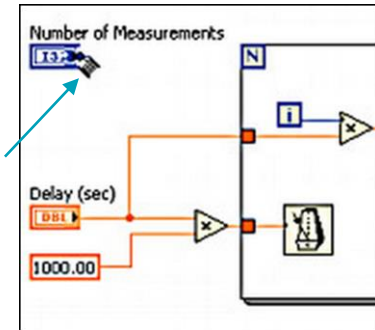


Figure 11 Wiring tool

You must connect the wires to inputs and outputs that are compatible with the data that is transferred with the wire. You must connect the wires to only one input and *at least* one output. To draw a wire hover the mouse over an input or an output until the cursor changes to the “wiring tool”, then left click to start drawing a wire. This is seen in Figure 11.

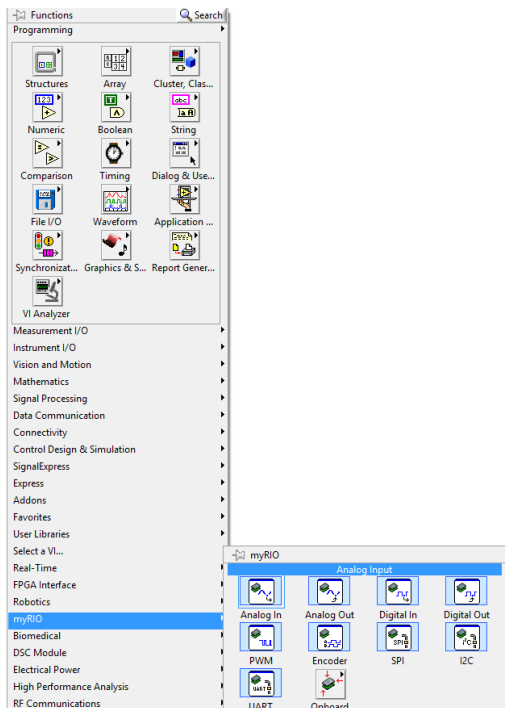


Figure 12 Functions Palette

The **Functions palette** seen in Figure 12 contains the VIs, functions, and constants used to create the block diagram. You can access the Functions Palette by right clicking on an empty area in the block diagram.

*Tip: If you know what you are looking for, you can find it fast with Quick Drop by pressing **ctrl+space***

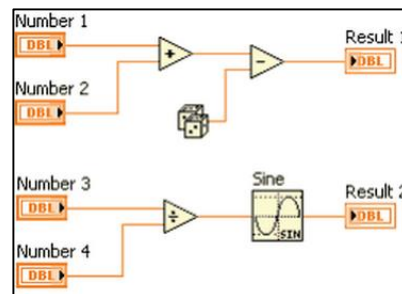


Figure 13 Dataflow Programming Example

³ All keyboard shortcuts can be found at http://zone.ni.com/reference/en-XX/help/371361H-01/lvhowto/keyboard_shortcuts/

The **order of execution** is determined by how your objects on the block diagram are wired together. A node will execute only when data are available at all of its input terminals and supplies data to the output terminals only when the node finishes execution. For example, In Figure 13, it is impossible to know which segment will execute first – the Add, Random Number, or Divide function. This unknown order of execution occurs because inputs to the Add and Divide functions are available at the same time, and the Random Number function has no inputs. In a situation where one code segment must execute before another, and no **data dependency** exists between the nodes, you should use other programming methods, such as sequence structures or error clusters, to force the order of execution. These structures are further explained in the next section.

5.2.3 Basic Loops and structures

The **Case Structure** has one or more subdiagrams, or cases, exactly one of which executes when the structure executes. Whether it executes depends on the value of the Boolean, string, or numeric scalar you wire to the external side of the terminal, labeled selector. A sample Case Structure is shown in Figure 14.

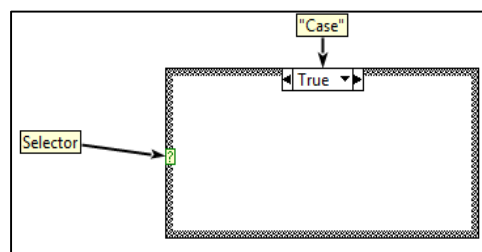


Figure 14 Case Structure

The **Sequence Structure** Consists of one or more subdiagrams, or frames, that execute sequentially. As an option, you can add sequence locals that allow you to pass information from one frame to subsequent frames by wiring a wire to the edge of the structure. You can add more frames by clicking “Add Frame After” from the right-click menu. A Sequence Structure can be seen in Figure 15.

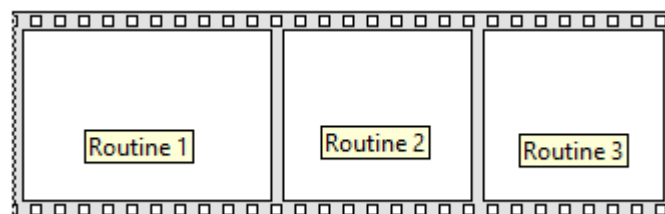


Figure 15 Sequence Structure

A **For Loop** executes its subdiagram N times, where N is the value contained in the count terminal. A For Loop is shown in Figure 16.

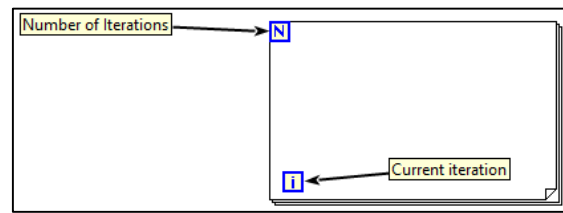


Figure 16 For Loop

A **While Loop** executes its subdiagram until a Boolean value you wire to the *conditional terminal* is TRUE. A While loop can be seen in Figure 17. Note that you can wire any Boolean value to the conditional terminal, it does not have to be a button on the Front Panel. As an option, you can again add shift registers so you can pass information from one iteration to the next. The value you wire to the **shift register** on the right side can be read from the shift register on the left side in the next iteration (these can also be used in For Loops). In the while loop shown in Figure 17 the numeric indicator on the front panel is going to display “1” the first iteration of the while loop and “0” for any consequent because the initial condition is “1” and there is a constant “0” wired to the right side shift register.

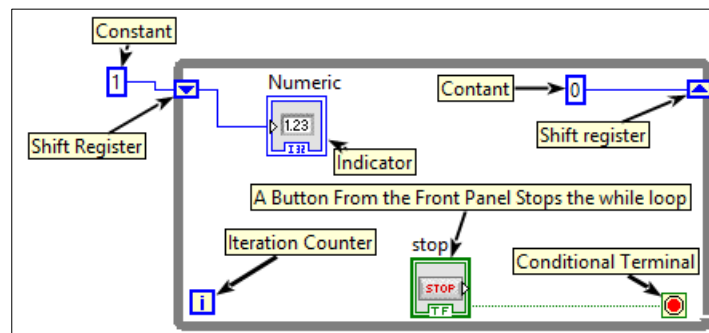





Figure 17 While Loop

A **Local Variable** lets you read or write to one of the controls or indicators on the front panel of your VI. Writing to a local variable has the same result as passing data to a terminal, except that you can write to it even though it is a control, or read from it even though it is an indicator. Local variables are useful when sending data between structures becomes very messy. You can insert a local variable onto the block diagram by searching “Local Variable” in the Quick Drop menu (ctrl+space) or alternatively you can right click on a control or indicator and choose “Local Variable”. You can choose whether to read from the terminal or write to it by right clicking on the variable. Using local variables is not good practice in general, because the program flow will be less clear and race conditions (when a data block is read from and written to at the same time), may occur. However, in some situations, such as transferring data between parallel *while loops*, local variables are acceptable.

5.2.4 Running the Code

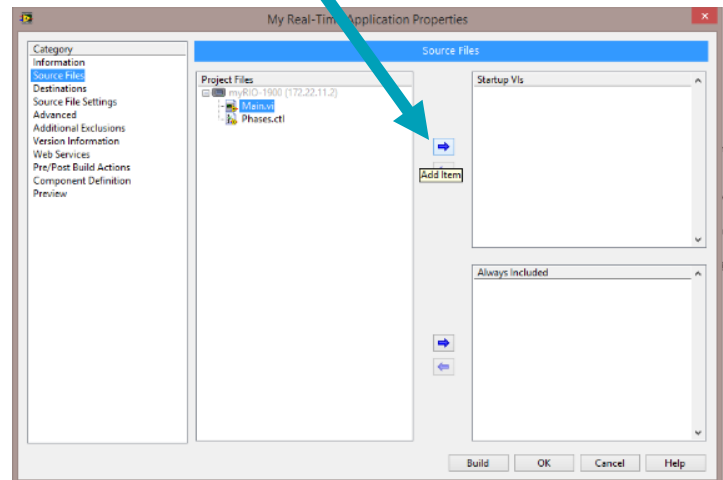
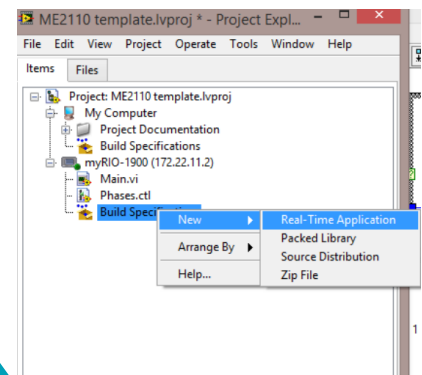
To run the code click on the arrow  on the top of the Block diagram or the Front Panel. If the arrow is broken  it means there are errors in your code and you can see the error list by

Important: Try to avoid using the abort execution button when interfacing with real hardware!!!


pressing on it. The abort execution button  stops the code regardless where execution of the code is. Section 5.4 explains how you should terminate the code when using the ME2110 template with the myRIO.

IMPORTANT: By default the myRIO **will not** run a program when you plug it in. To run the code autonomously, the following steps are necessary :

1. Right click new->Real-Time Application on Build Specifications under the myRIO tree in Project manager.
2. A configuration dialog opens. Go to Source Files and move the VI that you wish to run (e.g., "Main.vi") to the startup VI-s
3. Click OK
4. Right click on the Real-Time application you created under Build Specifications and click Build.
5. Right click on the Real-Time application and click "Set as Startup".
6. Right click on the Real-Time application you created and click "Deploy".
7. If the myRIO reboots your code will run automatically (note that it takes about 20 seconds for the myRIO to start up)
If you change the code you need to build and deploy the code again.



5.2.5 Debugging the code

To debug your code you can press on the highlight execution  button in the block diagram and you will see the code running slowly with values written on wires. **Alternatively** you can right click on a wire and click on **probe** or left click on a wire when running the code. This will

allow you to see the value of that wire when the code is running at full speed. You can also always add indicators that will display data on the front panel.

This was a very brief intro to LabVIEW programming. There is an extensive help section in LabVIEW. In addition, there is a lot of information and examples on ni.com as well as other locations on the Internet. If you register your copy of LabVIEW on ni.com you are eligible for free self-paced trainings offered at <http://sine.ni.com/myni/self-paced-training/app/main.xhtml>

5.2.6 Interfacing with the myRIO hardware

From the Functions Pallet you will find the myRIO section where there are a variety of Express VI which are all you need to utilize the inputs and outputs of the myRIO.

An Express VI is a VI with settings that you can configure interactively through a dialog box. Express VIs appear on the block diagram as expandable nodes with icons surrounded by a blue field.

You can configure an Express VI by setting options in the configuration dialog box that appears when you place the Express VI on the block diagram. You can double-click the Express VI or right-click the Express VI and select **Properties** from the shortcut menu to reopen the configuration dialog box. You can also configure the Express VI by wiring values to terminals of the Express VI on the block diagram.

The following Express VIs are used in the ME2110 template:

- Digital Input – Reads an on/off signal from a sensor such as a switch
- Analog Input – Reads the Voltage on a specified channel
- Encoder – Reads the number of clicks and gives the direction of rotation
- PWM – Outputs a PWM signal on a specified channel
- Digital Output – Writes an on/off value to a channel
- LED – Lets the user control the LEDs on the myRIO⁴
- Button – Reads the on-board Button0 that is on the myRIO⁵
- Accelerometer – Reads the values from the on-board accelerometer

When you select an Express VI, drag it to the block diagram and a configuration dialog box appears where you will choose the channels you want to include. Refer to Table 1 and Table 2 for a list of channels used by the Sensor and Driver expansion Board. More details are given in Section 6.

⁴ Functionality is the same as compared to the Digital Output Express VI

⁵ Functionality is the same as compared to the Digital Input Express VI

5.3 FILES IN THE ME2110 TEMPLATE PROJECT

Once you open the template project file, you will find many files under the myRIO tree. Main.vi gives you a way to test all of your hardware connections as well as show you how to programmatically connect to them. This VI will be further explained in subsequent sections. My First myRIO program.vi gives you a step-by-step guide on how to write the most basic program. (You can find the [youtube tutorial](#) for this problem in the quick start guide at the beginning of this manual.) The other files are to serve as example programs. The problem statements for each of the vi-s are written on the respective Front Panels. All example problems are thoroughly commented (within the VI).

5.4 UNDERSTANDING MAIN.VI

Open Main.vi under the myRIO tree in the Project Manager. The Front panel opens.

Front Panel

*TIP: To change between the block diagram and the front panel fast press **ctrl+E***

On the left side of the Front Panel you can see the controllable hardware section as depicted in Figure 19. The buttons and knobs will control the hardware that is connected to the Driver Board (see Section 4.1.2). The LEDs are on the myRIO itself and do not need to be attached. If you run the motors, make sure that the Enable Motors is pressed. The Fault indicator turns on when the motors are drawing too much current or the driver board is not connected. This can happen when the motors are stalled or try to accelerate too fast. If the LED stays on for more than a couple of seconds check your hardware for errors. Make sure that you refer to Table 2 when interfacing with the hardware.

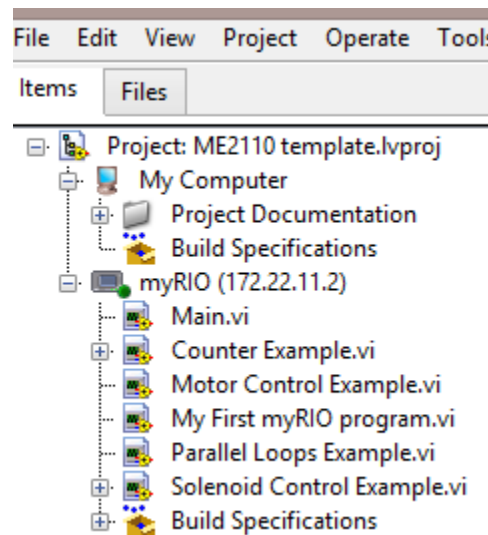


Figure 18 ME2110 Template Project

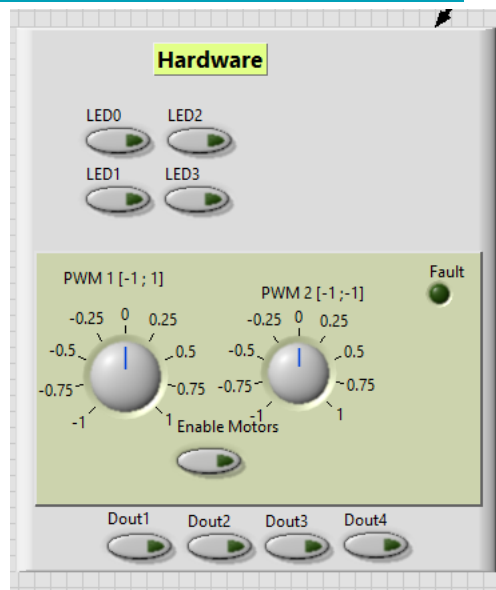


Figure 19 Hardware Controls on the Front Panel

Sensor data is displayed on the right side of the Front Panel as depicted in Figure 20. The accelerometer and the “On Board Button” are on the myRIO and do not need to be attached.

If you wish to change the appearance of any of the objects on the front panel, right-click on them and choose “replace” or “properties”. There is a wide array of indicators and controls available. If you are uncertain on how some of them work, you “help” from the system bar.

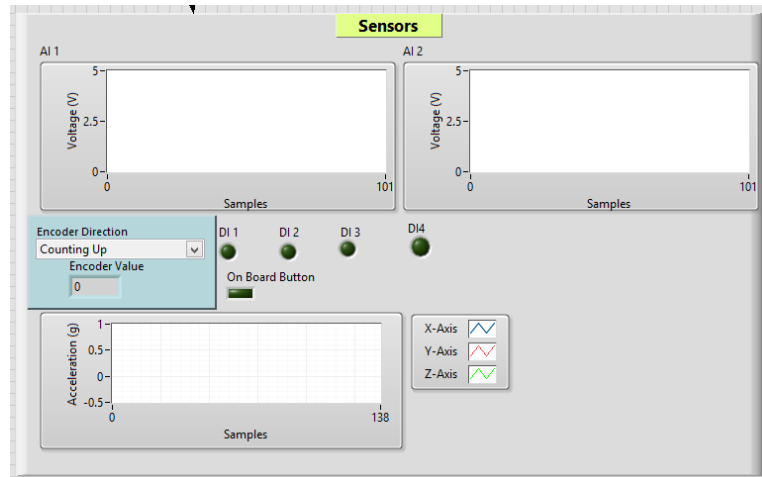
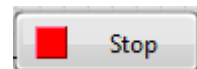


Figure 20 Sensor Data on the Front Panel

Important: When you press the “Stop” button, the myRIO will exit the main While loop and will reset all of the Channels. If you use the Abort Execution button, the myRIO will end mid-code, and will therefore not reset, and will keep the channels at the values that were last written to them. For instance the Solenoids might be On and the motors running. To fix this, you can unplug the power to the Driver Board and press the “Reset” button on the myRIO. When the program gets stuck in an infinite loop that it can’t get out of, you must use the Abort Execution button.

Do not use the Abort Execution button  to stop your code. Use the Stop Button provided!!



Block Diagram

The Block Diagram used in the main.vi and all of the example programs in the ME2110 template project have the structure that is displayed in Figure 21. It is a **Flat Sequence Structure** as seen in the previous section which means that the frames will execute after the previous one has ended. In the “Initialize” frame the initial values are set. The second frame contains a while loop, which is where all of the functional programming is done. It is exited once the code tells it to or the Stop button on the Front Panel is pressed. The while loop may have a delay between

each iteration. After the while loop terminates it the code will go to the last frame where you will need to add the Reset myRIO vi.

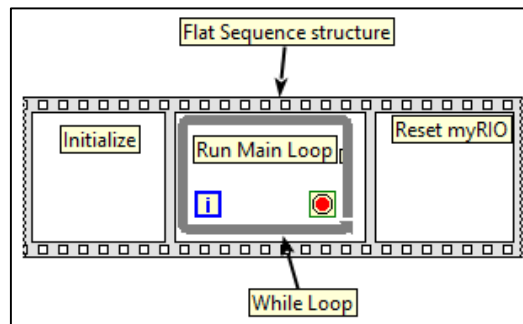


Figure 21 Structure of the Block Diagram

If you open the block diagram for the Main.vi you will see that all of the inputs and the indicators that are on the Front panel are on the left side of the While loop and the controls on the Right. Remember you can find the terminal on the front panel by double clicking on the terminal on the block terminal and vice versa.

You are not obligated to use this template.

*To create your own myRIO project. Open LabVIEW –
File- Create Project*

Find the myRIO template.

5.5 CONNECTING TO THE MYRIO OVER WIFI

It might be desirable to talk to the controller wirelessly. You may want to do so when multiple people wish to communicate with the device, or when you cannot (are not allowed) to be physically in contact with your device.

To set it up please follow the steps.

1. Plug the myRIO to your computer through USB
2. Go to <http://172.22.11.2> in your internet browser
3. Go to “Network Configuration” on the left side menu
4. Choose “Create wireless network” under “Wireless Adapter wlan0”

5. Enter the name of your network under "SSID" and choose the security features you want.
6. In your computers network settings log into the WI-FI network you just created. (If your Ethernet cable is not plugged in you probably will not be able to connect to the internet until you reconnect to your standard network)
7. To program the myRIO and view data over WI-FI right click on the myRIO in the project explorer in LabVIEW and choose "properties". Then under "general" change IP address to 172.16.0.1

*To communicate through USB again, you will need to change the IP address in the project manager back to 172.22.11.2

5.6 COMMON MISTAKES

- The WiFi button on the myRIO does nothing.
- It takes about **20 seconds** for the myRIO to boot up. (Tip: use an LED indicator to show when the myRIO is running.)
- If you have not set a start-up VI the myRIO will do nothing when powered on. See Section 5.2.4.
- If a switch is not connected, the myRIO will read "TRUE" because of this concept: http://en.wikipedia.org/wiki/Pull-up_resistor - you are dealing with low level electronics. This means: switch pushed is "FALSE", released is "TRUE"
- The myRIO can use up to 1 A (the bigger power adapter can supply max 3 A) of current so it is advisable to use both power adapters at the same time when using many actuators to ensure that everything works reliably.
- When you use the "wait" blocks to do timing, the entire structure where it is at will be on pause.
- To make sure that the driver board doesn't come loose, you should use the ribbon cable and the mounting holes provided.
- Any structure or VI used in in your code will only execute if ALL the inputs have been given a value.
- If all elements on your block diagram have executed the program will stop. This is the reason why in most cases you will need a while loop to keep your code running.
- When using studio computers choose LabVIEW 2014, not an earlier version.

6 Electromechanical Components

This chapter will explain the electromechanical components included in the kit. The sensors used in ME2110 include micro switches, an infrared distance sensor, a potentiometer and an encoder. Actuators include two DC motors, pneumatics and solenoids. For each component, its basic principle of operation will be discussed; then, how to program the myRIO to utilize each component will be covered.

6.1 SENSORS

6.1.1 Switches

Switches are shown in Figure 22. They are simply two wires connected by a mechanically-triggered attachment. In the relaxed state, the two wires are said to be open, not connected. When the lever is pressed down, closing the switch, the two wires are shorted and a current flows. The long-arm switch is momentary; its circuit only remains closed while the lever is pressed. The pushbutton switch is a toggle switch. Once pressed, it will remain in the switched position until pressed again. The switches provide an input signal in one of two possible states. A “LOW” corresponds to a closed circuit, the condition when the switch is closed, while a “HIGH” corresponds to an open circuit, which is the condition when the switch is not pressed. “LOW” and “HIGH” can also be interpreted as “FALSE” and “TRUE”.⁶

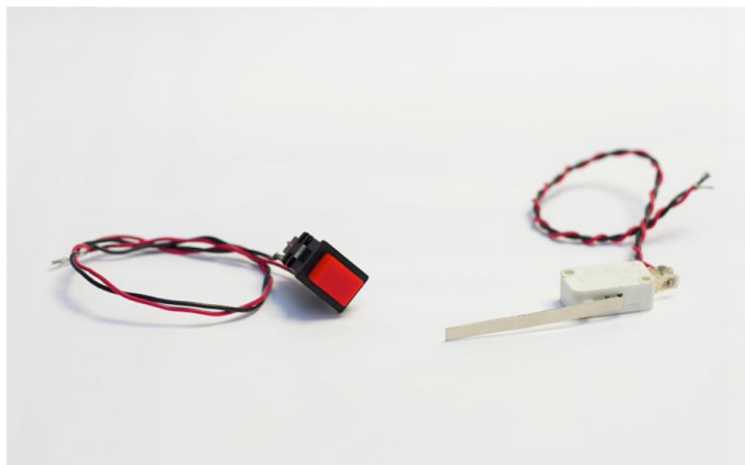


Figure 22 Switches

⁶ It might not seem logical why “LOW” and “HIGH” are interpreted as such but is a common concept in electronics http://en.wikipedia.org/wiki/Pull-up_resistor.

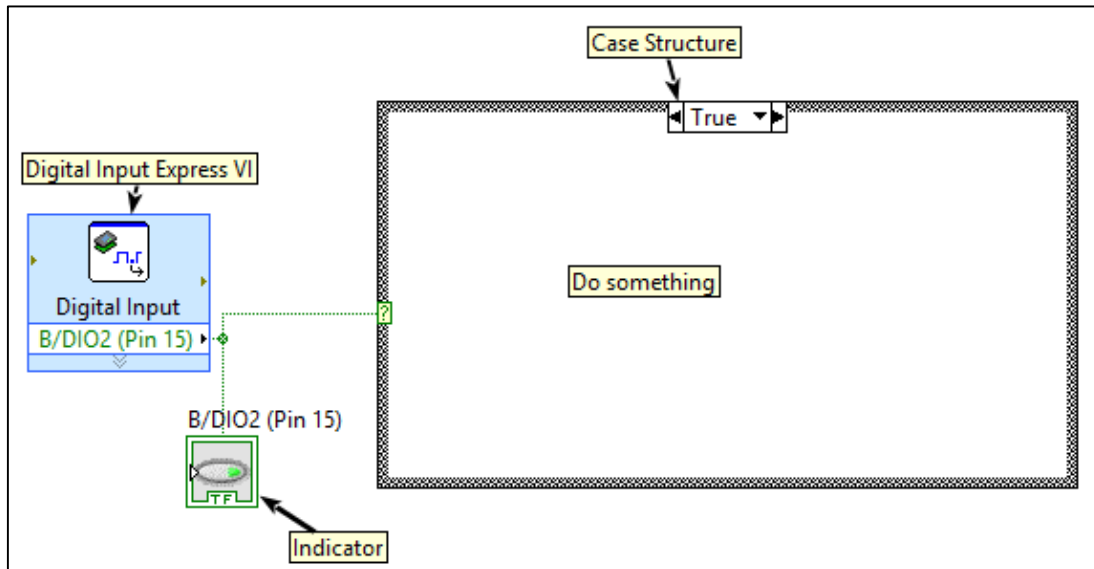


Figure 23 Example code for a Switch

Each switch is connected to a data channel through the Digital Input Express VI. Choose the channels you want to use and name them accordingly. Refer to Table 1 for the list channels. In order for a program to act upon the state of the switch, a case structure (can be used the same way as “If – Then - Else” in text based programming) can be used. An example of this is show in Figure 23. This example program reads the input from the switch using the Digital Input Express VI, displays it on the front panel and executes a case structure where the user can write routines for “TRUE” and “FALSE” cases. You will need to add a loop (usually a while loop) around this code to read the digital input more than one time.

6.1.2 IR Distance Sensor

The Infrared (IR) Distance Sensor, shown in Figure 25, is used to detect objects within a given range (approximately 4 to 31 cm for the sensor in the kit.) To determine the distance of an object, the sensor first sends an IR beam; it then receives this beam, once reflected from an object. The angle at which the beam returns is used to calculate the distance of the object. It is important that the wires from the IR sensor are connected to the correct ports on the controller box. Read more about the measurements [here](#). The proper way to wire the IR sensor is shown in Figure 26. In the figure, the mounting holes of the IR sensor are on the bottom.



Figure 25 The IR Distance sensor



Figure 26 IR Sensor Wiring

The myRIO reads the voltage that is sent by the sensor. Use the Analog Input Express VI. Refer to Table 1 to find the channel you are looking for. It is up to the student to convert the voltage readings to distances. The data you get will be noisy, to help smooth that you can use the *running average* to filter the data. Sample code is shown in Figure 24. This program reads the analog input, computes the running average, displays it to an indicator on the front panel, and does a logical comparison with 2 (an arbitrary number). The code is left blank in the case structure. The while loop is stopped when a stop button is pressed on the front panel. Finally, the myRIO is reset.

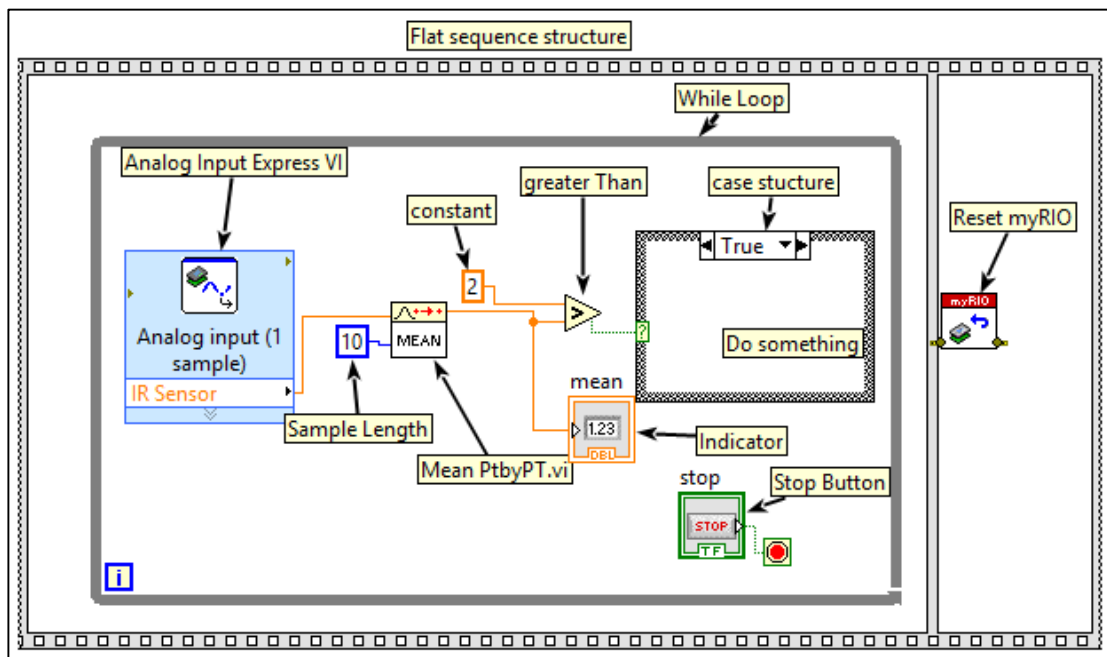


Figure 24 Example Code for the IR sensor

6.1.3 Potentiometer

A potentiometer is a three-terminal resistor with a sliding contact that forms an adjustable voltage divider. The potentiometers that are included in the kits can be used to measure the angle of rotation. The correct wiring is shown in Figure 27. The myRIO reads the voltage the same way as with the IR sensor so an Analog Input Express VI should be used. The potentiometer you receive in your kit might look a little different but the operation principle is the same.



Figure 27 Potentiometer With the Plug Attached

6.1.4 Encoder

The encoder is shown in Figure 28 and is used to measure rotational motion. The encoder and the potentiometer appear similar but you can tell them apart by how much the shaft turns: the encoder can rotate freely but the potentiometer has a range of motion of about 270° . The rotational motion of the shaft of the encoder is converted to electrical pulses, or counts, that can be interpreted by a microcontroller. The rotational measurement is based upon the number of counts. The encoder included in the kit is a relative encoder, meaning it only counts relative to its starting position for each count command.

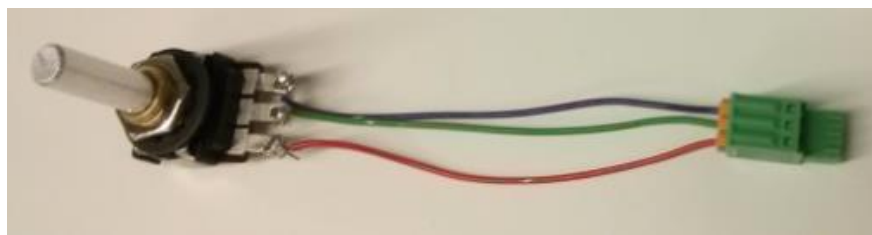


Figure 28 Encoder with Plug Attached

There is an Encoder Express VI for the myRIO to handle the encoder. It outputs the distance rotated in clicks and the direction at which it was last rotated. When configuring the Express VI choose “Quadrature Phase Signal” option. Refer to Table 1 to get the proper channel. Sample code can be seen in Figure 29.

This piece of code displays the counter value and the direction on the front panel. It also compares the value to a constant and checks if the counter is counting up. If both conditions are "TRUE", the indicator will display that on the front panel. You will need to add a loop to read the encoder more than one time.

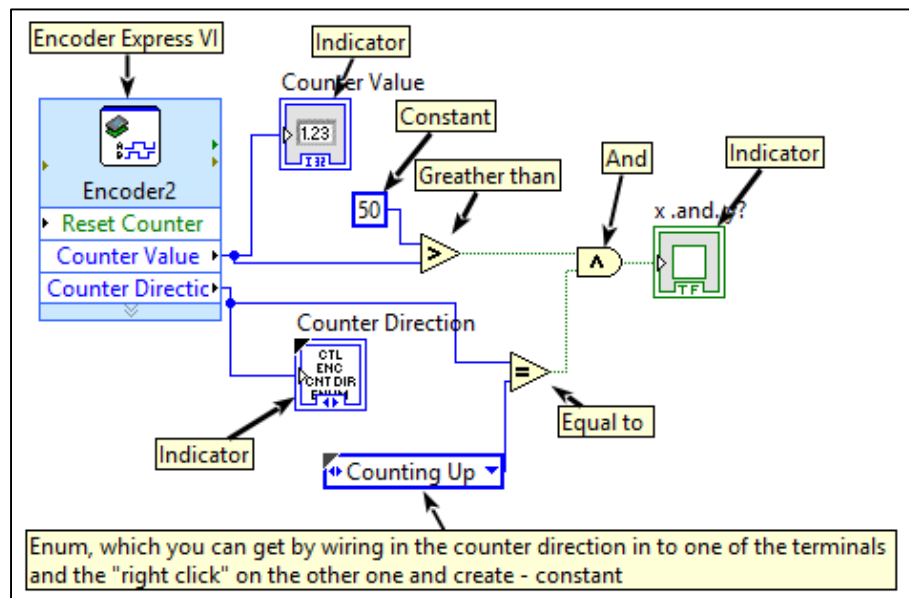


Figure 29 Sample Code for the Encoder

6.2 ELECTROMECHANICAL ACTUATORS

6.2.1 DC Motors

The provided DC motors are brushed motors. The two issued in the kit will be similar to those shown in Figure 30. The actual motors issued may differ, but their functionality and programming interface is the same. The DC motor works by inducing an electromagnetic field using coils of wire around an armature. Around the inside of the casing are permanent



Figure 30 DC Motors

magnets, and as the shaft turns, the brushes reverse the polarity of the armature, causing the shaft to rotate.

The direction of the Motor 1 is determined by channels A/DIO6 and A/DIO7 (A/DIO14, A/DIO15 for Motor 2). When the direction pins differ the motor turns (you can switch the direction by reversing the pins.) If the pins are both “HIGH” the motor will break hard and stall (not recommended!) when both are “LOW” the motor will coast to a stop. The speed of the motors regulated by the Pulse Width Modulation (PWM) signal which is explained Figure 31.

The PWM signal can be given to the motor driver through the PWM Express VI. A/PWM2 is for Motor 1 and A/PWM0 for Motor 2. When configuring the PWM Express VI set Frequency to 20

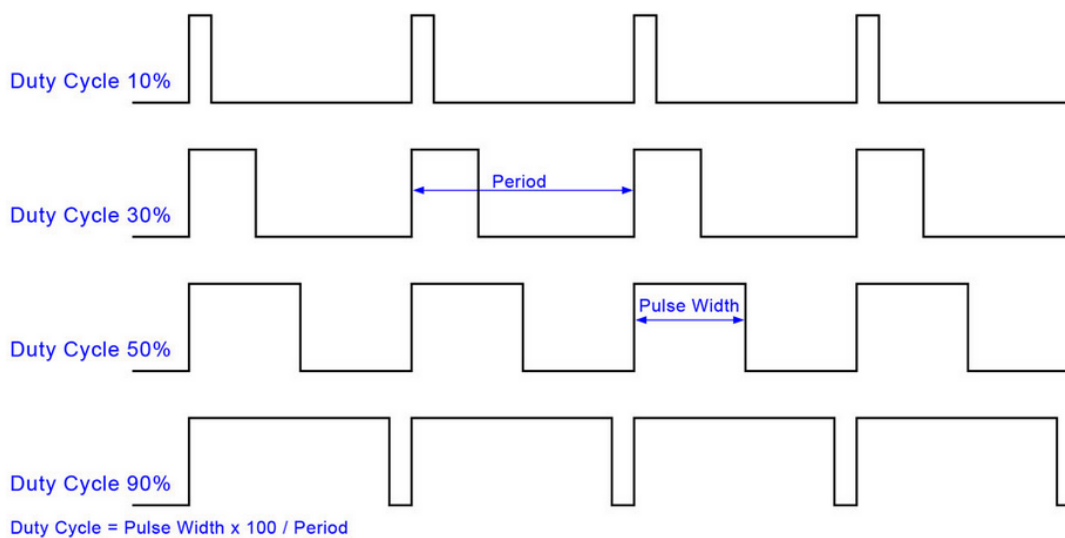


Figure 31 PWM signal

kHz and Duty cycle to “Set using input to Express VI”. The input for the “Express VI” will have to be a “double” datatype from 0 to 1, where 0 will not move the motor, 0.5 will move it at 50% duty cycle and 1 will move the motor at full speed. An example on how to configure the speed and direction can be seen in Figure 32. If the “double” from the Front Panel (from -1 to 1) is greater than zero then the “Switch” node will output “TRUE” which will be sent to the first direction pin and the direction pin will be the inverse of that. This is an easy way to determine the direction and speed with one control input. The absolute value of the float will be sent to the input of the PWM Express VI (Remember the input must be from 0 to 1). To stop the motor the user has to write 0 to the PWM Express VI.

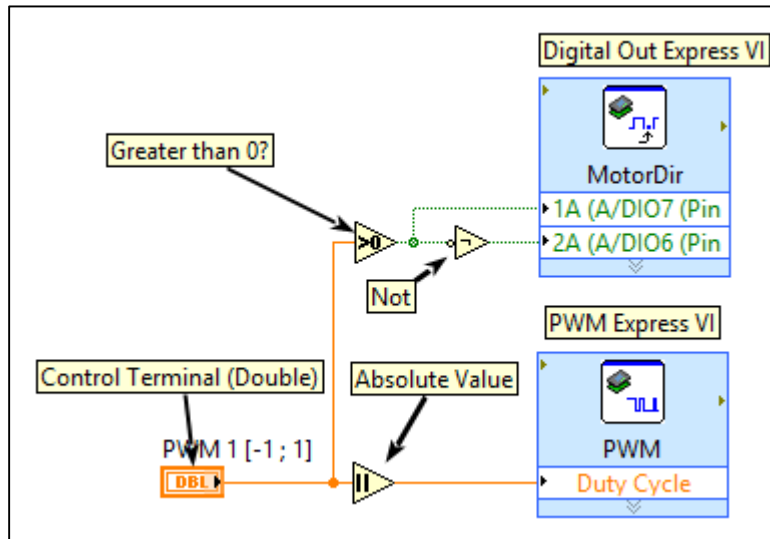


Figure 32 Motor Control sample code

6.2.2 Solenoids

A solenoid is essentially a coil of wire around a metallic plunger. When current is run through the coil of wire, it acts as an electromagnet, pulling in the plunger. Solenoids can be “push”, “pull”, or both. The solenoids in the kit are “pull” solenoids. The two solenoids issued as part of the kit are shown in Figure 33. Solenoids should be connected to the digital outputs on the Driver Board, labeled DOUT1-DOUT4. Refer to Table 2 for the list of channels.



Figure 33 Solenoids

They are controlled using the Boolean (“TRUE”, “FALSE”) inputs. Issuing the “TRUE” command activates the solenoid, pulling in the plunger. “FALSE” deactivates the solenoid, but will not return the plunger to its original position. The solenoids are controlled using the Digital Output Express Vis. Example code for the solenoids is shown in Figure 34.

2 examples are given how to turn two solenoids on and off after 2 seconds: 1) using a sequence structure and a wait and 2) using the Elapsed Time VI.

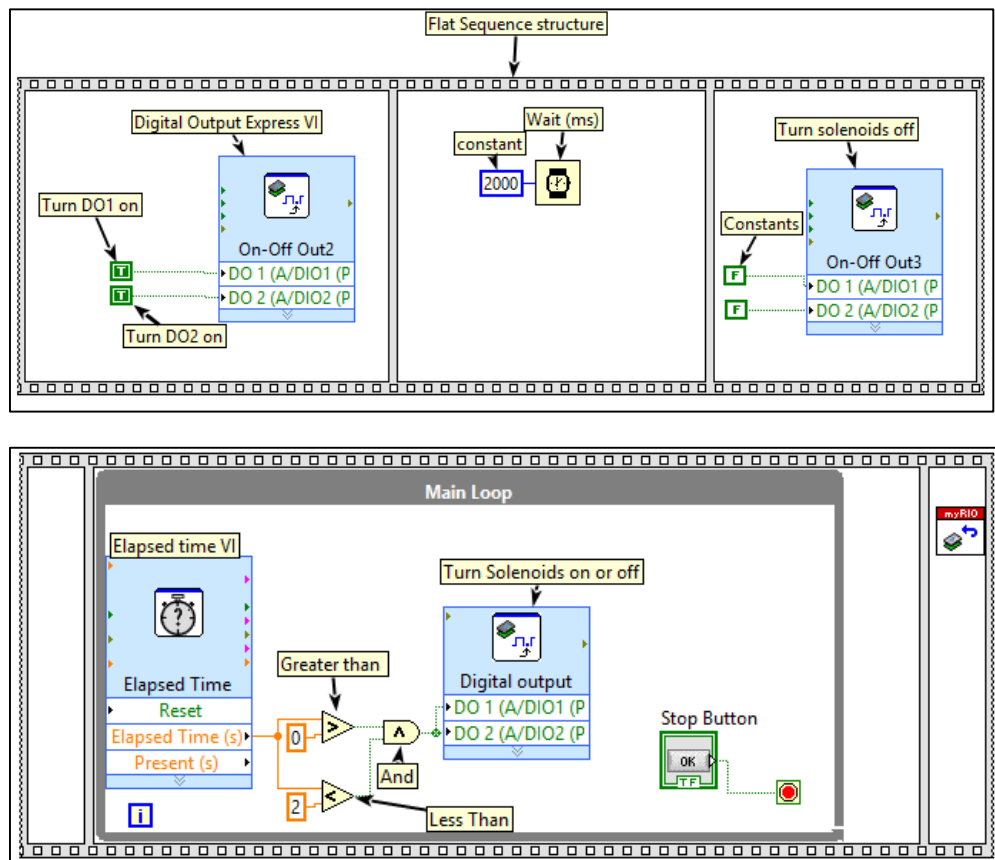


Figure 34 Solenoid Control Examples

7 Pneumatic components

The pneumatics system, shown in Figure 35, centers around pneumatic actuators, which can be repeatedly extended. This section will describe the hardware components of the kit and how to program the myRIO to use the system.

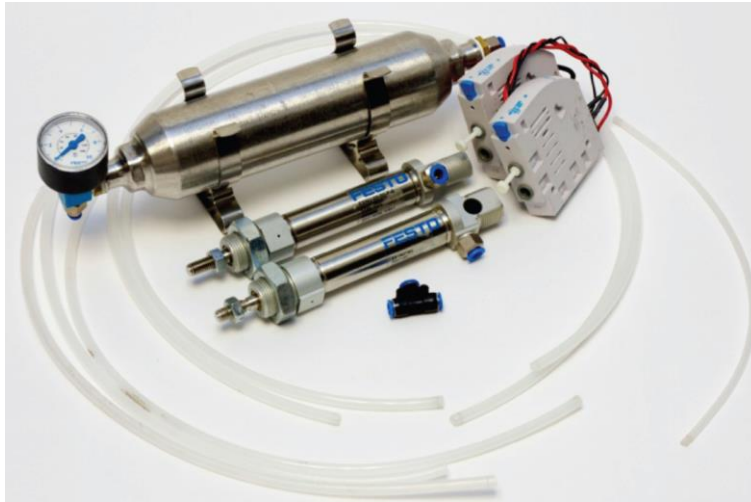


Figure 35 Pneumatic components



Figure 36 The Pneumatic Tank



Figure 37 Pneumatic Actuators

The pneumatics system includes a tank, shown in Figure 36, an actuator/cylinder, shown in Figure 37, a valve, shown in Figure 38 as well as several connecting fittings and tubing. In order to use the pneumatic system, it must be connected as shown in Figure 39. If it is not correctly connected, air will escape, preventing the system from being pressurized. A bicycle pump can be used to provide the pressure needed for the system, up to a maximum of 100 psi. It will not operate properly if the pressure drops below 25 psi. Higher pressure allows more actuations of the cylinder to be achieved before a re-pressurization is needed. However, **DO NOT OVER PRESSURIZE THE SYSTEM**. The pump connects to the tank using a 4mm diameter piece of

tubing, through the 4mm one-way coupler on the back of the pressure gauge. After filling the tank, the 4mm tube should be removed from the tank prior to being removed from the air source. Leaving the tube in the tank keeps the one-way valve open, allowing the air to escape.

The pneumatic valve controls the airflow of the system: it is a solenoid valve with open and



Figure 38 Pneumatic Valves

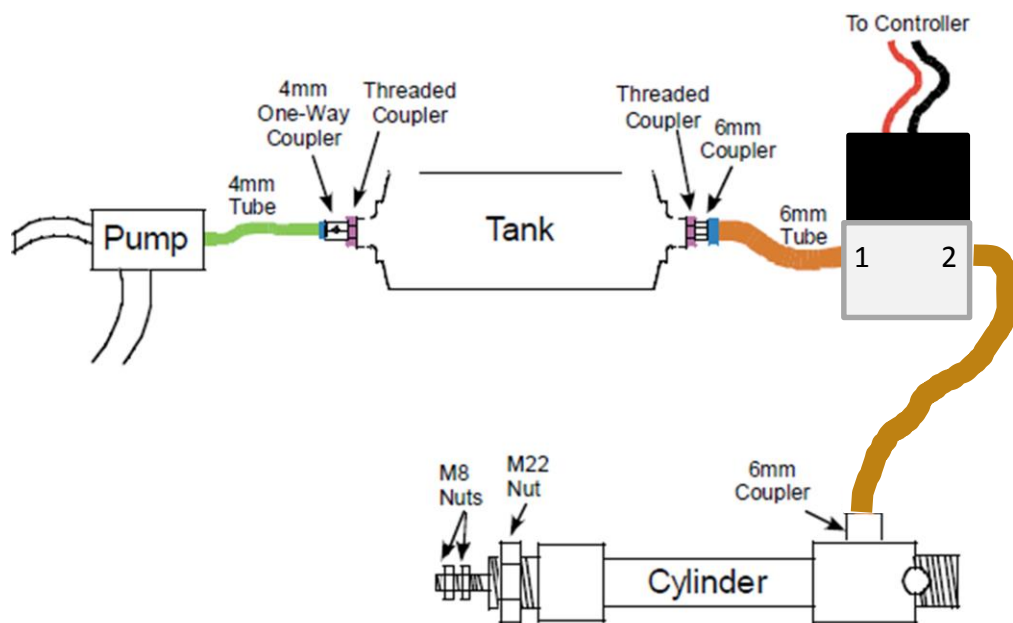


Figure 39 Connection of the Pneumatic System

closed states. In the closed state, ports 1 and 2 of the valve are connected. Once the valve is opened, port 1 is connected to port 2, and air flows from the tank into the cylinder, extending the actuator. When the valve returns to the closed position, port 1 is again plugged. The spring return in the cylinder returns the actuator to un-extended position, pushing air out the open port 3.

The pneumatic valve is a solenoid and is programmed identically to the solenoids issued in the kit. Figure 40 shows some sample code on how to use the solenoids (pneumatic valves). Take note that the “Error Wires” determine the **order** of execution in this case. The same effect could be obtained with a Flat Sequence structure and not using the Error wires as demonstrated in Figure 34.

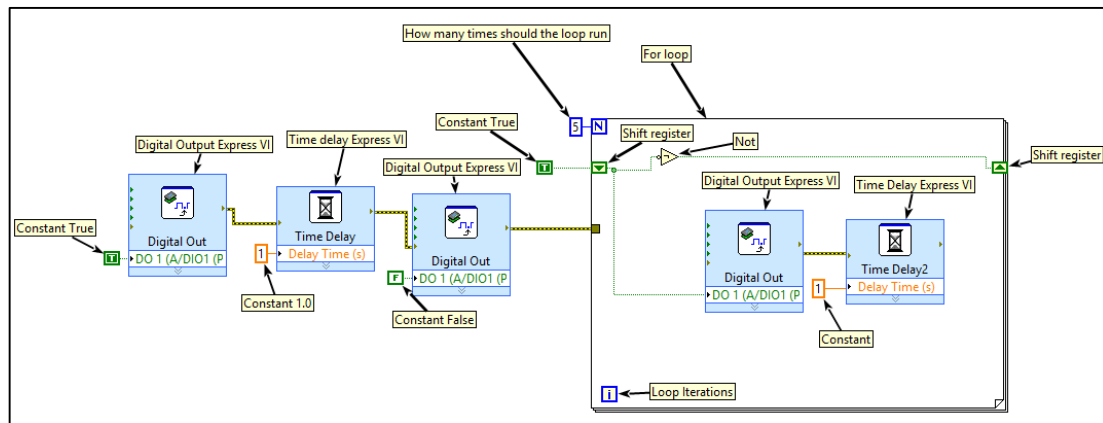


Figure 40 Sample Code

The shift register has the value “TRUE” for the first loop and it will be toggled for the subsequent iterations of the For Loop. The code turns on a solenoid, then waits one second, then turns the solenoid off. After this the “for loop” starts, runs 5 times and the output is toggled.

8 Example Programs

In the Example Program shown in Figure 41, the myRIO will turn on a Solenoid once the Potentiometer value has been greater than 2 at least once (the shift registers will continue to be true after they have been true once because of the OR block) AND 2 switches have been pressed for at least 2 seconds (this will be configured in the Elapsed time settings, which can be opened by double clicking on the icon.) The elapsed time block will be reset when either of the two switches is not pressed. The “Not” block + “Not And” block could simply be replaced with an “AND”. If the output from the “And” block is false, the solenoid will be turned off, this is not shown in the diagram. After the stop button has been pressed the while loop will exit and the myRIO is reset.

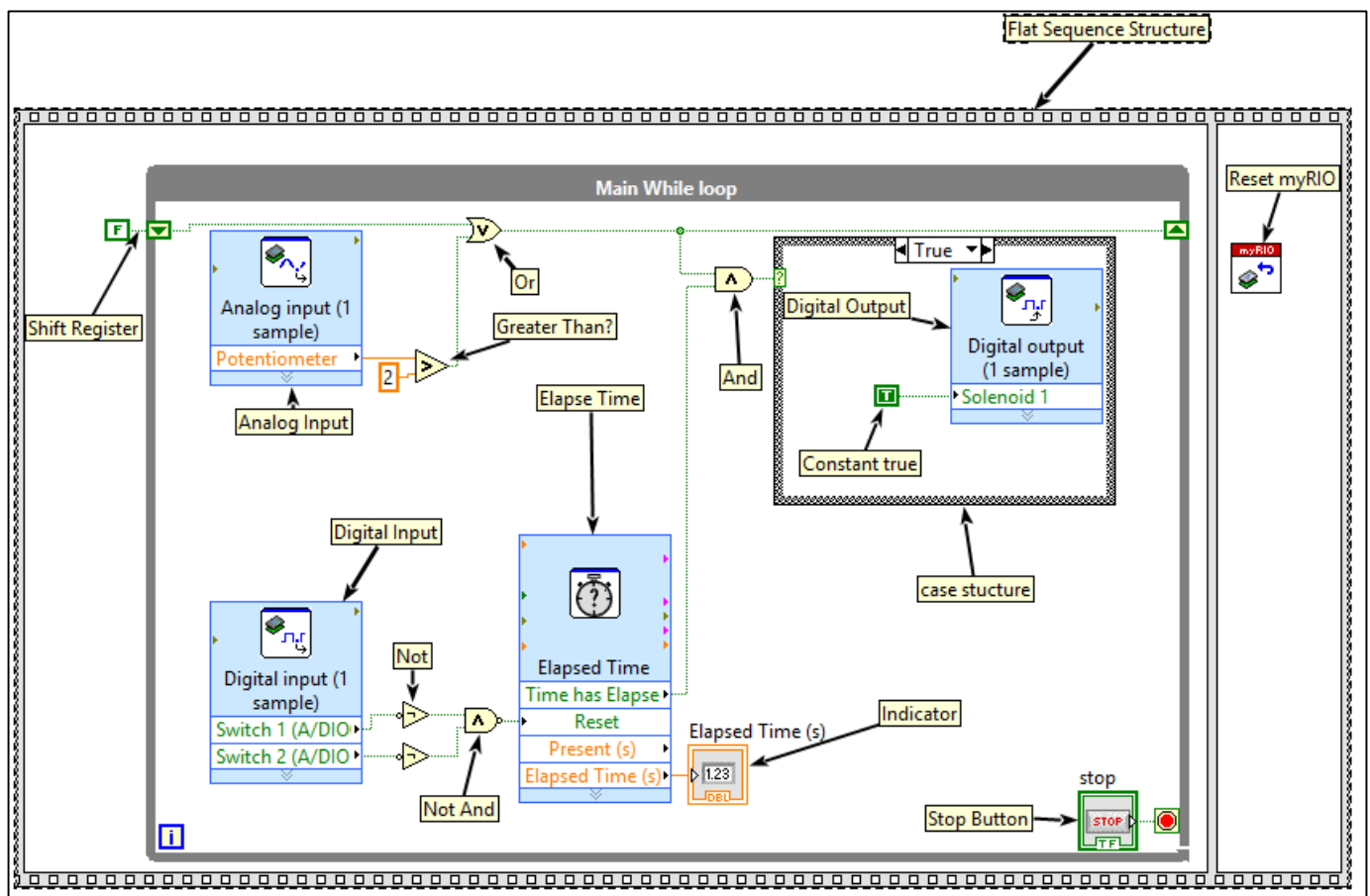


Figure 41 Example Program

In the program shown in Figure 42, initially, one motor runs and one solenoid is turned off. After two switches have been pressed at the same time, the motor should stop and the solenoid is turned on. Then the motor runs again and the solenoid is turned off after 1 second has passed. The cycle will be continued until the “Stop” button is pressed on the front panel. While the program is running, the LED 1 and LED 2 will show the states of the switches (therefore you cannot use any “wait” blocks in your code).

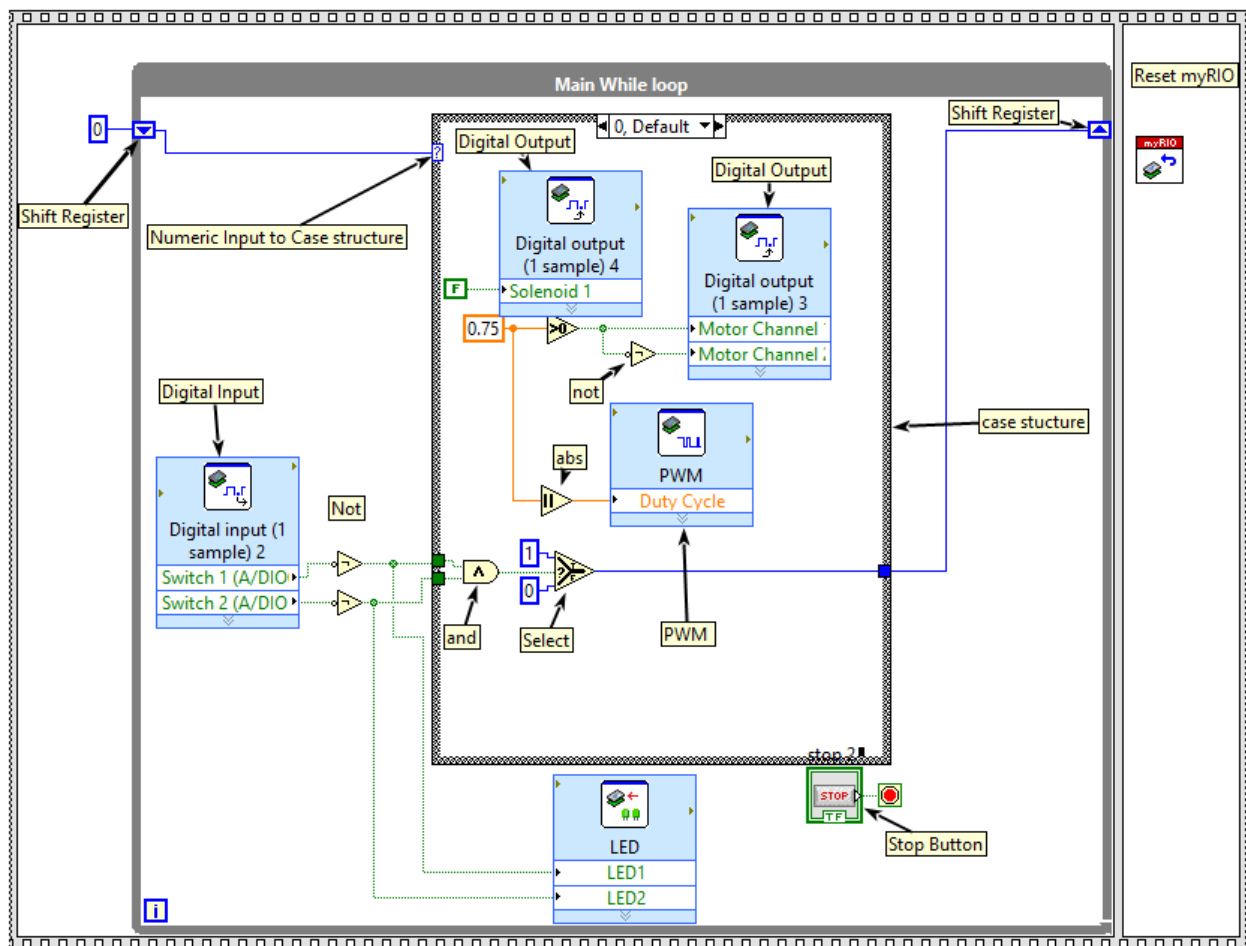


Figure 42 Example Program

Note that this time the case structure takes in numeric inputs (LabVIEW can detect this automatically). In the first instance of the while loop, “0” will be passed through, after that the value of the shift register will determine which, either “0” or “1”, case will be used in the input for the case structure. The selector works by sending the value in the upper value if the Boolean input is true and the bottom one if the Boolean input is false. The values of Switch 1 and “Switch 2” are sent to the case structure and when they are both “true” the selector will output

“1”. This will be sent to the shift register and in the next iteration of the while loop Case “1” will be selected.

Figure 43 shows the case “1”, where the motor is turned off by setting the speed to zero, and the solenoid is turned on. When the “Elapsed Time” Express VI sends a “true” to the **selector** the “0” case will be used the next iteration of the while loop. Note that in this program the selector works the same way as a “Not” element. The program is stopped when a “Stop button” is pressed on the Front panel. This program is very similar to how the ME2110 template works.

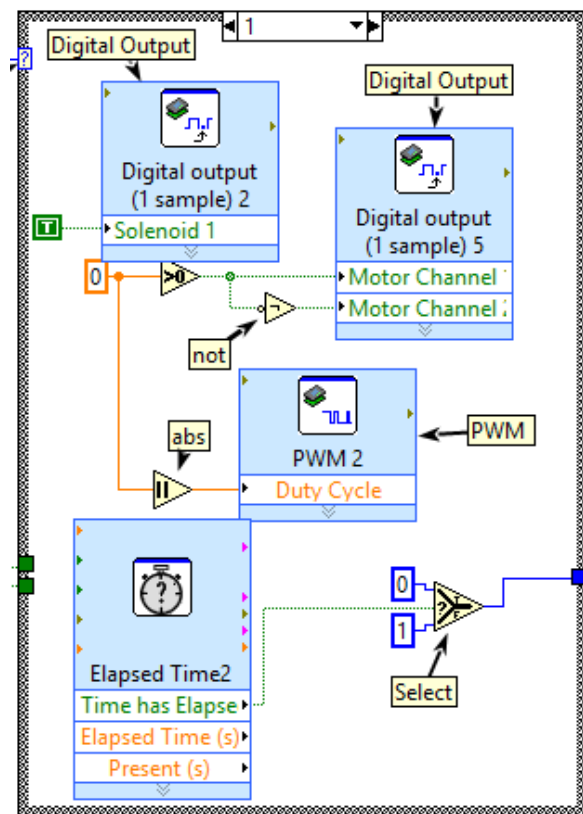


Figure 43 Case “1”

9 Helpful links

<http://www.learnni.com> – Learn LabVIEW

<http://www.learnni.com/getting-started/Home/Index/45> - Debug LabVIEW

<http://www.ni.com/white-paper/14621/en/> - Examples and tutorials on how to use different components with the myRIO

<http://www.ni.com/manuals/> - Manuals for LabVIEW and more

<http://www.ni.com/pdf/manuals/376047a.pdf> - myRIO user guide

<http://ni.com/myRIO> Lots of resources for the myRIO

http://download.ni.com/support/softlib//labview/labview_myrio/2013/C%20Support%20for%20NI%20myRIO%20User%20Guide.pdf –How to program the myRIO in C

<http://www.ni.com/white-paper/14621/en/> - The guide breaks down wiring, I/O requirements, device theory, and programming of over 20 different devices common to NI myRIO projects.

APPENDIX A – Components

Use the table below to get the characteristic values for the mechatronics components

Component	Model Number	Datasheet
Encoder	288T232R12	link
Potentiometer	P120PKF17BR5K	link
IR Sensor	GP2D120XJ00F	link
Pneumatic Actuator	ESNU-20-50-P-A	link
Small Solenoid	69905K4	link
Large Solenoid	69905K6	link
Large DC Motor	DK-37R545123000-41K	link
Small DC Motor	BDSG-37-30-12V-5000-R100	link