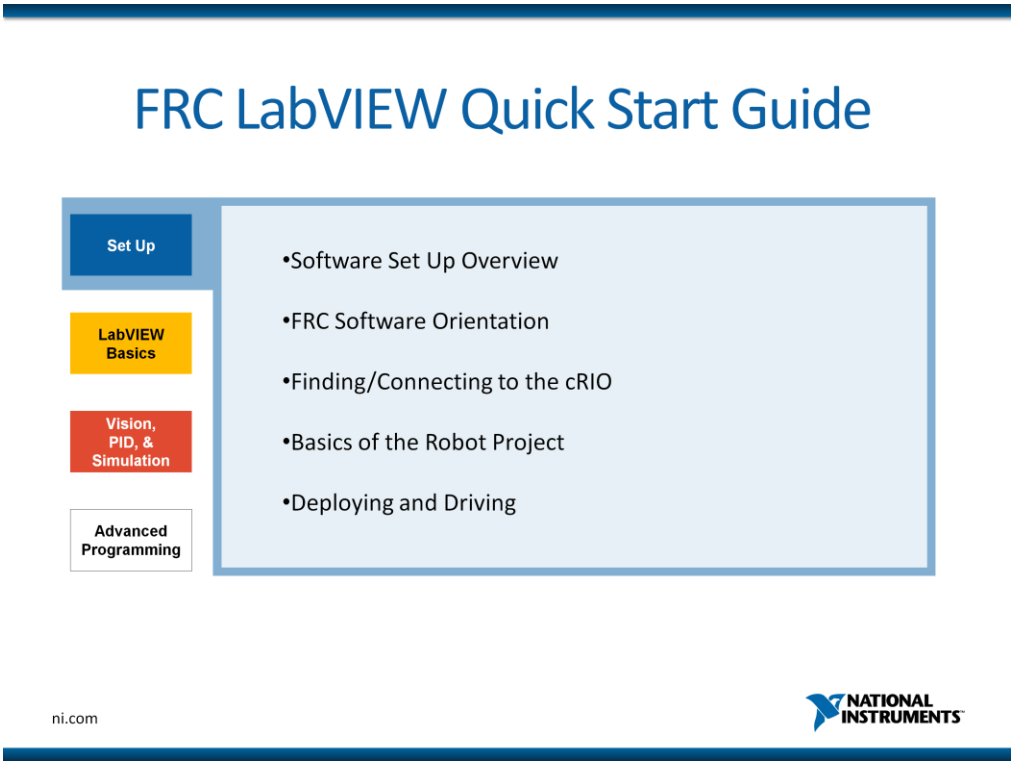
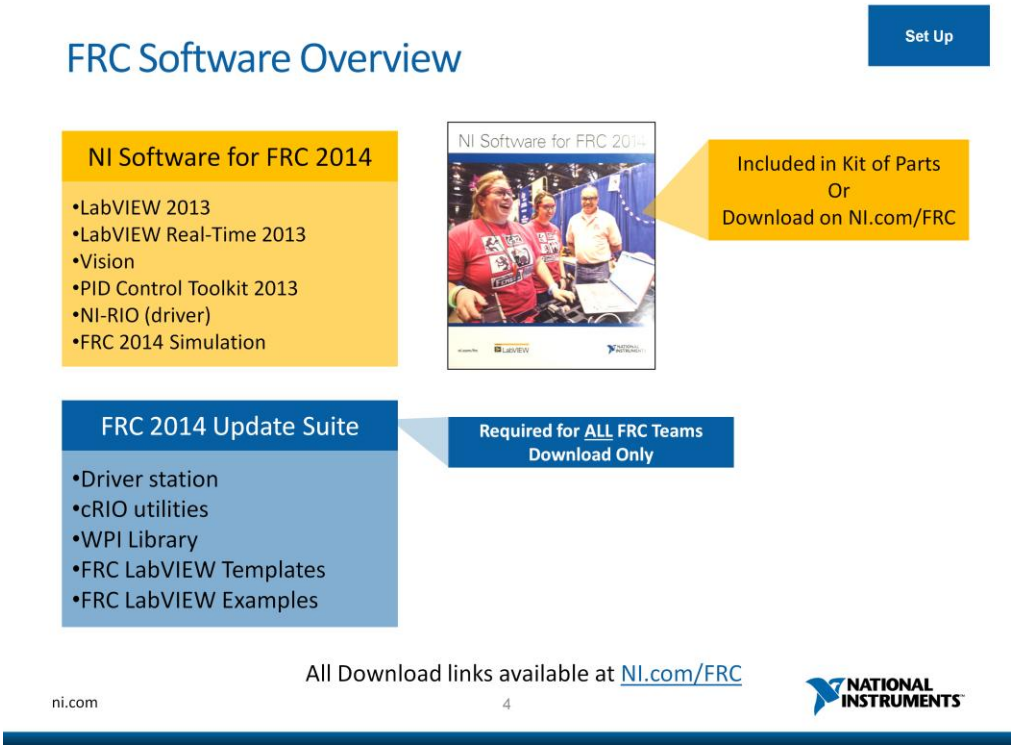
The FRC LabVIEW Quick Start Guide is designed to give teams a quick overview of the software, tools, and utilities they will need to program an FRC robot using LabVIEW.



The Set Up module will help you download and install your software, orient yourself with the software components, connect to your cRIO and deploy the existing template project. If your hardware is all ready set up you should have a drivable robot at the end of this section, no programming required (yet).

This training does not cover hardware build instructions



All teams should get an “NI Software for FRC 2014” cardboard envelope in their kit of parts, with the NI Software DVD. You can use the DVD to install the standard NI software. Be sure to save this envelop incase you need to reinstall or reactivate later.

If you don’t have a DVD drive or you want the NI software before you get your kit of parts you can also download it on NI.com/FRC. It is a 5GB download so it may take a few hours depending on your internet connection. Even if you download you will still need the serial number from the DVD envelope to activate it.

Regardless of what software your team plans to use (C++, Java, LV) you need to download the FRC 2014 Update Suite. It includes the driver station, dashboard, cRIO imaging tool as well as all of the FRC specific content like the FRC Examples and Templates. (~500MB)

Set Up

1. Install NI Software

NI Software for FRC 2014 (DVD or Download)

- If you download the software you must Unzip and run AutoRun.exe

Use the DVD from your kit of parts or download the “NI Software for FRC 2014” from NI.com/FRC.
Note to instructor: Be sure to explain that they need to download, unzip then install the software. (we’ve gotten a lot of questions about this)

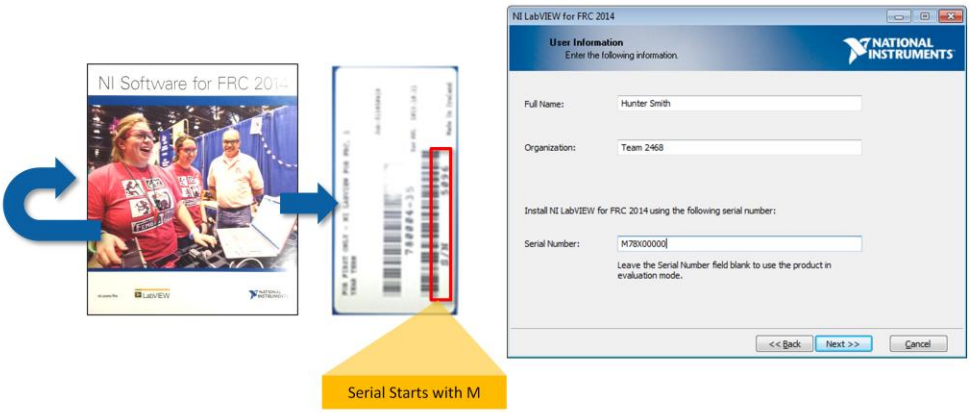
Click next on the first screen then select the products you would like to install.
[Animation]

If you plan to develop your robot in LabVIEW install all of the components.
If you are using C++ or Java but still want to use NI Vision just install vision and deselect all of the other components.

Set Up

2. Activation

- Activate LabVIEW using the serial on the back of the DVD Envelope



During the install process you will be prompted for the serial number, it is located on the back of the DVD envelope. Be sure to enter in your team number for the organization.

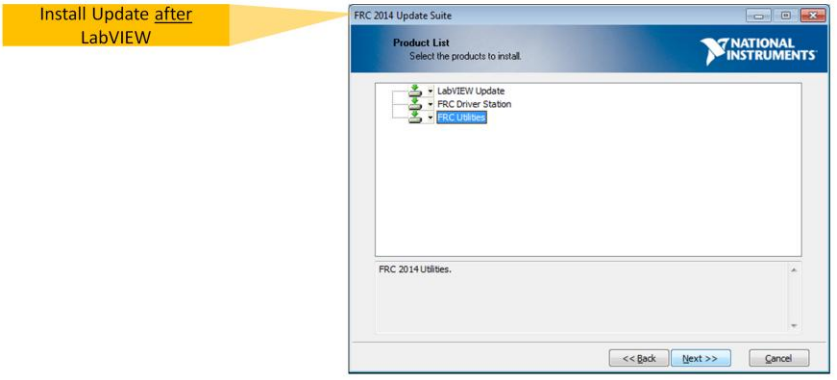
If you don't have your kit of parts yet you can leave the serial number blank to install as a trial until you get your serial number.

Later in the install process you will be prompted to activate Vision, use the same serial number

Set Up

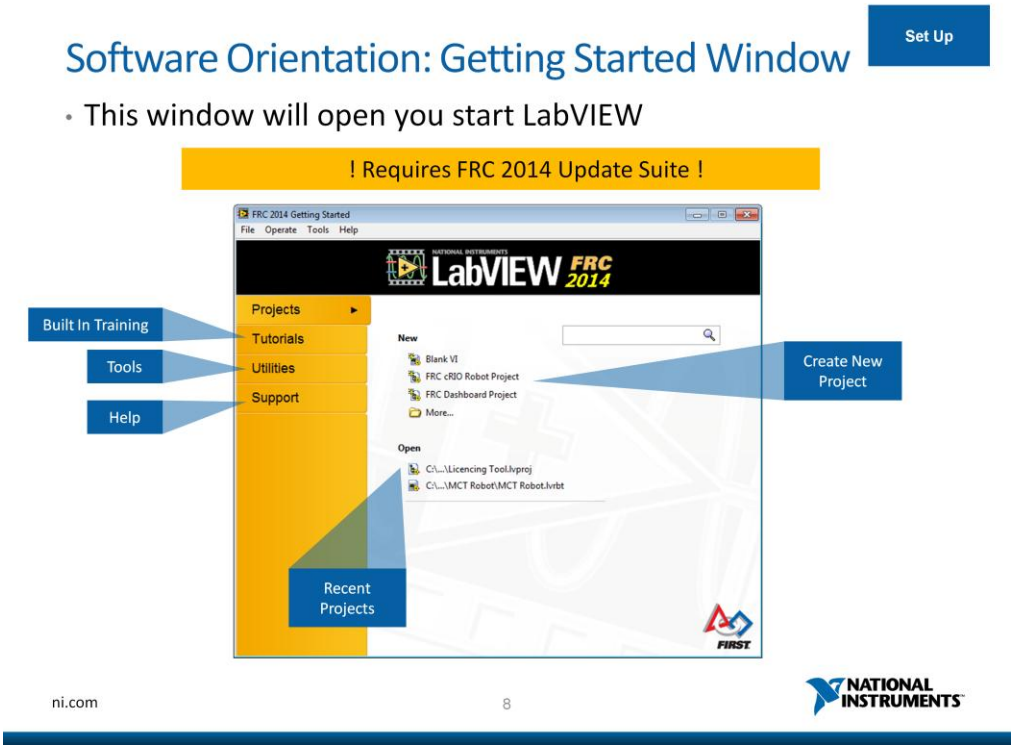
3. FRC 2014 Update Suite

- All Teams must install the Update Suite



The FRC 2014 Update Suite contains all of the FRC specific software made by NI.

[Animation]
All teams will need to install the FRC utilities in order to image the cRIO and the FRC Driver Station in order to control their robot.
If you are using LabVIEW you will be the LabVIEW Updates to install the Project templates.



After you install LabVIEW and the Update Suite when you start LabVIEW you will see this FRC getting started window.

The Getting Started window for LabVIEW FRC is packed with useful information and tools

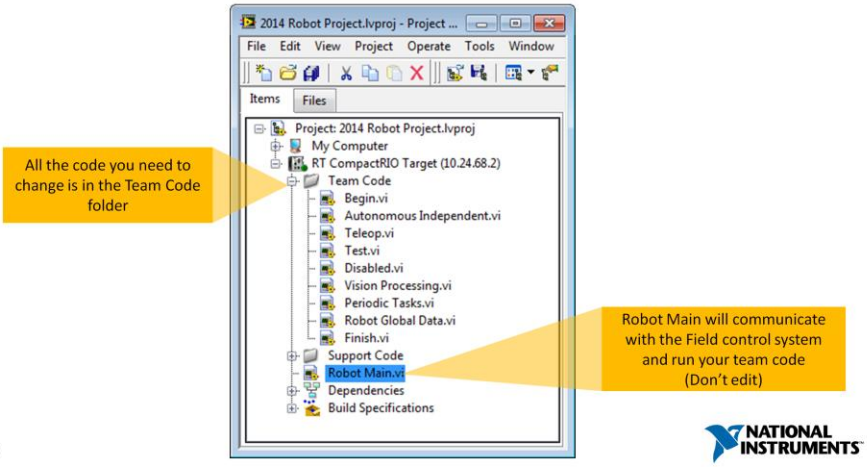
Note: You can open the Getting Started window by closing all Vis and Projects OR you can go **to View>Getting Started Window** from any LV Window.

[Animation]
Project: When you are ready to start programming just create a new FRC cRIO Robot Project to get a project with everything set up and ready to go
[Animation]
Tutorials: Start with the tutorials to go into detail into different aspects of LabVIEW and the cRIO
[Animation]
Utilities: Quick links to all of the tools and utilities you need for FRC
[Animation]
Support: If you ever get stuck there is a link to the forums as well as the phone support number for help from an engineer during the build season.

Set Up

Software Orientation : cRIO Robot Project

- The LabVIEW project window allows you to see all of the code that will run on your robot. The FRC software includes a template project with everything you need to start.



ni.com

NATIONAL INSTRUMENTS

The LabVIEW Project allows us to organize many LabVIEW programs (VIs) (Virtual Instruments) in one place. When you create a new cRIO Robot Project it will come with all of the Vis you need included.

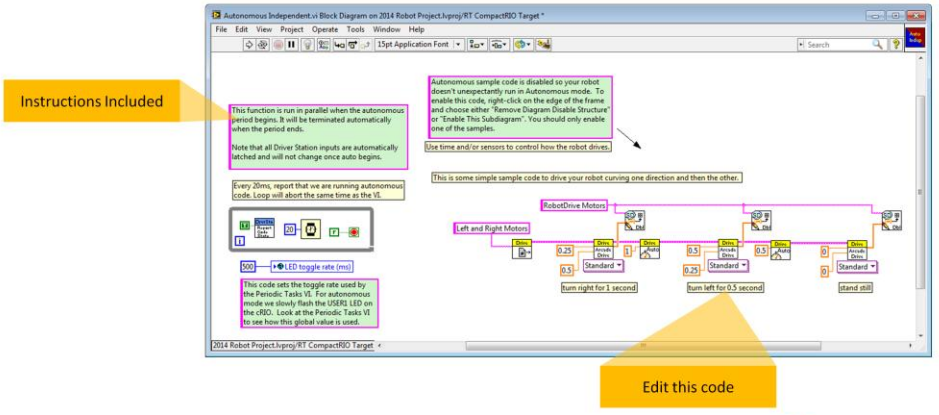
Robot Main is the program that communicates with the Field control system and runs all of your team code, you should not need to edit this.

The Vis your team needs to edit are all included in the Team Code Folder. Each VI includes instructions and example code to help you get started.

Software Orientation: Team Code

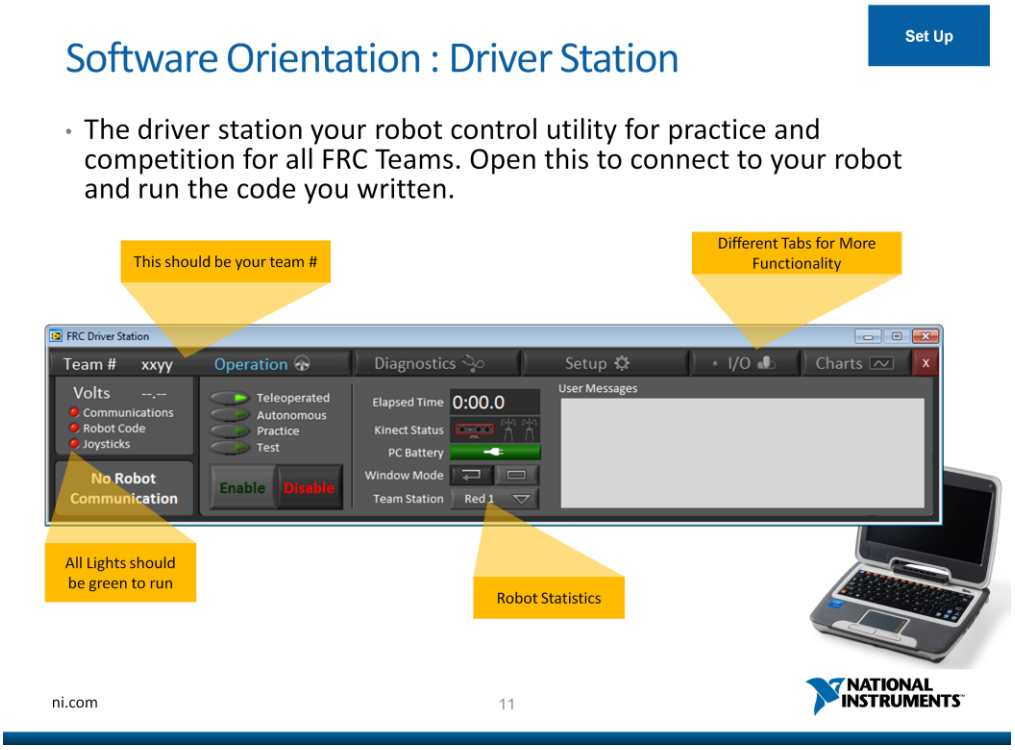
Set Up

- This is the Autonomous Independent VI, create your autonomous program here. This is one of the Vis you will need to edit.



ni.com 10 NATIONAL INSTRUMENTS

This is an example of one of the VIs in the team code folder, “Autonomous Independent.vi” The default code shows you how to access all of the inputs and sensors you need to make a smart autonomous program. Be sure to read through all of the instructions so you know what everything does.



The Driver Station is an executable utility that all FRC Teams are required to use on their driver station PC. Once you have written your code you can connect to the cRIO and deploy and run your code from here/

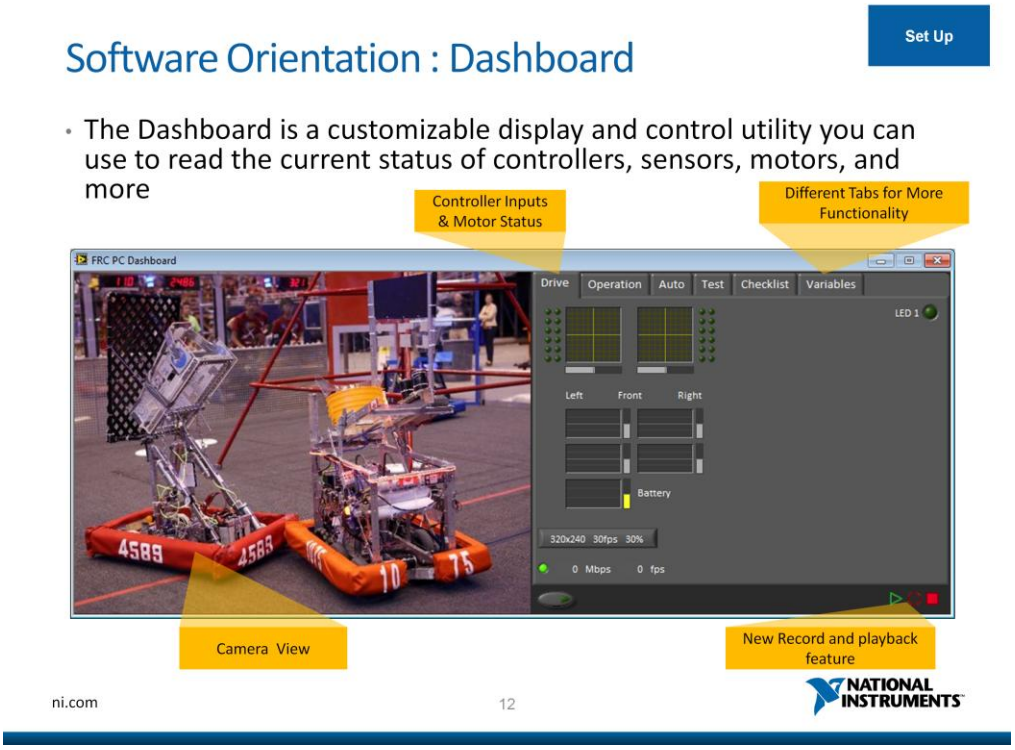
Operation tab: shows indicators of which stage of the game you're in, whether you have communication with the robot, and if the Joysticks/controls are plugged in. From here you can enable your robot, test your code, and set up for competitions.

[Animation]
Diagnostics tab: includes information on what is connected and any error messages generated by the system

[Animation]
Set Up tab: This is where you configure your team info, practice round timing, and choose your dashboard type

[Animation]
I/O: You can control inputs and outputs that can be read by your robot in when autonomous mode starts or during teleop. For instance you could display the current state of limit switches on your.

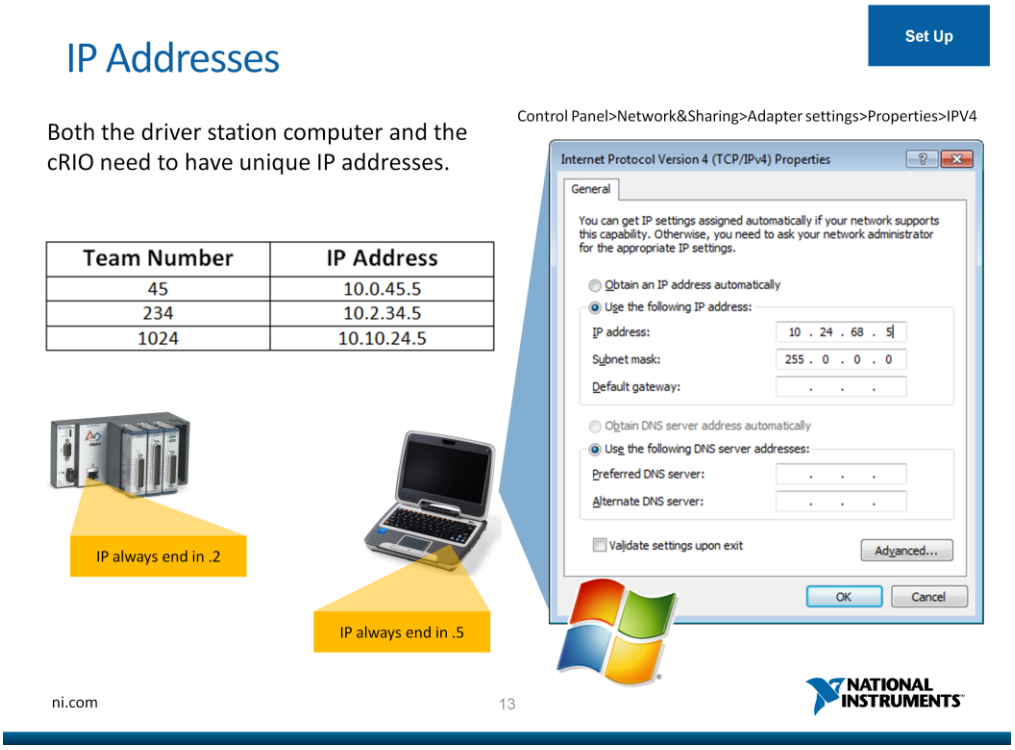
[Animation]
Charts: Displays information about robot battery and CPU usage, as well as network information. If your robot becomes unresponsive check here.



Most students don't even realize the dashboard is different than the driver station since it launches with it by default.

The driver station is used to enable communication and work with the robot, while the dashboard allows real time monitoring while the code is running.
It's a great tool for testing code, troubleshooting, or general robot status information.

It can also be customized for your team's robot and specific needs! To create your own custom dashboard open the FRC Dashboard project from the getting started window and follow the instructions.



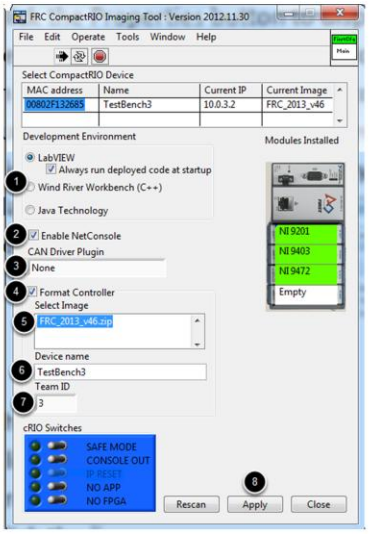
In order to connect the cRIO to your PC you will need to configure unique IP addresses for each device. The IP addresses will be based on your team number using the formula shown in this chart. The cRIO IP address should all ways end in a dot 2 and the PC should always end in a dot 5.

You set the cRIO IP address when you create a new cRIO Robot Project, to set the Windows IP address in the control panel. There is a very detailed tutorial on NI.com/FRC in the training section if you need help finding your settings on different versions of Windows.

Software Orientation : cRIO Imaging Tool

Set Up

- This utility allows you to set up your cRIO controller



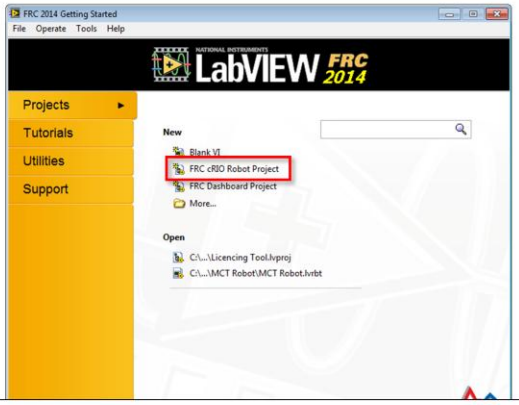
The cRIO imaging tool is used to connect to your cRIO, load the required software libraries and configure the basics settings for the cRIO. You should only need to use this to set up your controller or to wipe it clean if you need a fresh start. If you have your cRIO now, go ahead and image it using this tool.

Troubleshooting Tips:
These are for discovering the cRIO in Measurement and Automation Explorer, but the same principles apply (if you correctly installed software on the PC, you won't need to worry about that section)
http://digital.ni.com/public.nsf/allkb/ABE4BC247E8AC9BC8625734E005CAB42

Drive a Robot: Step 1
Open the Template Project

Set Up

Step1: From the LabVIEW Getting Started Window, select **New >> FRC cRIO Robot Project** to create a new LabVIEW project. (Step 0: Open LabVIEW for FRC!)



If you plan to use real hardware the cRIO and D-Link will need to be properly set up before attempting the rest. You could run the project in simulation

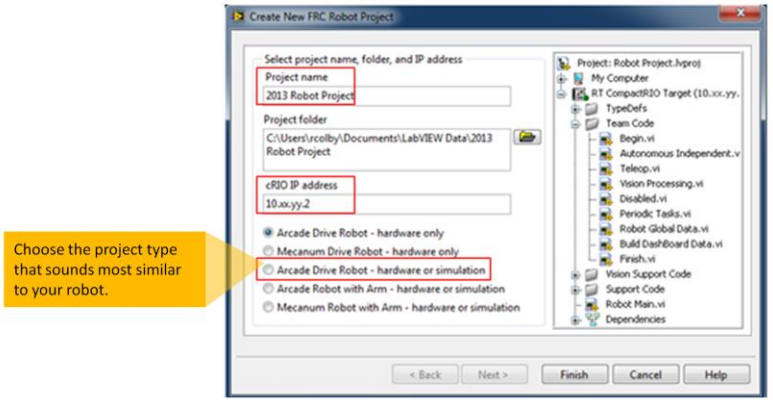
In the next few slides we will walk though how to get driving in 5 steps. If you have LabVIEW and the FRC 2014 Utility Package installed you can follow along. This tutorial can be completed with real hardware or with a simulated robot. You will need a joystick in order to drive so be sure yours is plugged in.

Open LabVIEW and start by creating a new cRIO Robot Project

Drive a Robot: Step 2
Configure your cRIO Robot Project

Set Up

Step2: Fill in the New Project dialog box. Give the project a meaningful name, use your team number to replace xx.yy in the ip address, and pick the type project you want to make.



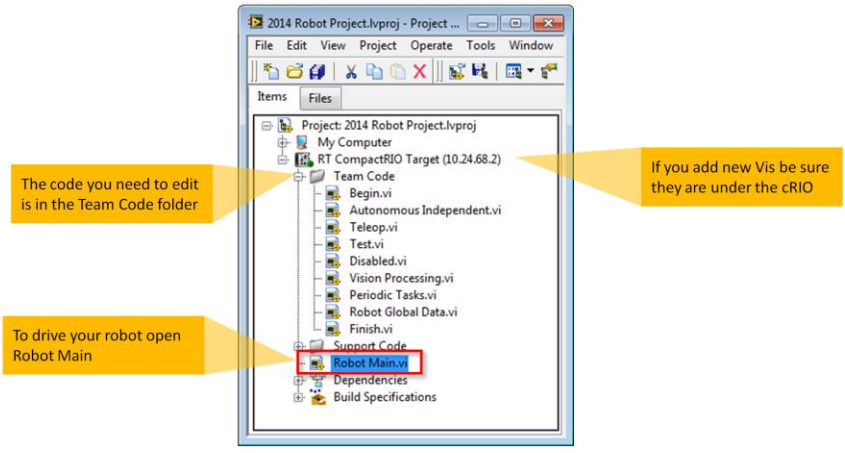
Choose the project type that sounds most similar to your robot.

After you create a new cRIO Robot Project this is the next dialog you will see.

Give your project a good name, specify where you want to save everything, set your IP address and select the drive style that most resembles your robot. Note that some have the option to include simulation while other exclude it, but the project itself would be the same. The non simulation versions will be smaller on disk size. If you don't have real hardware with you be sure to select one of the simulation options. You can always create a new project and paste any modified code in later if you decide to change the drive style.

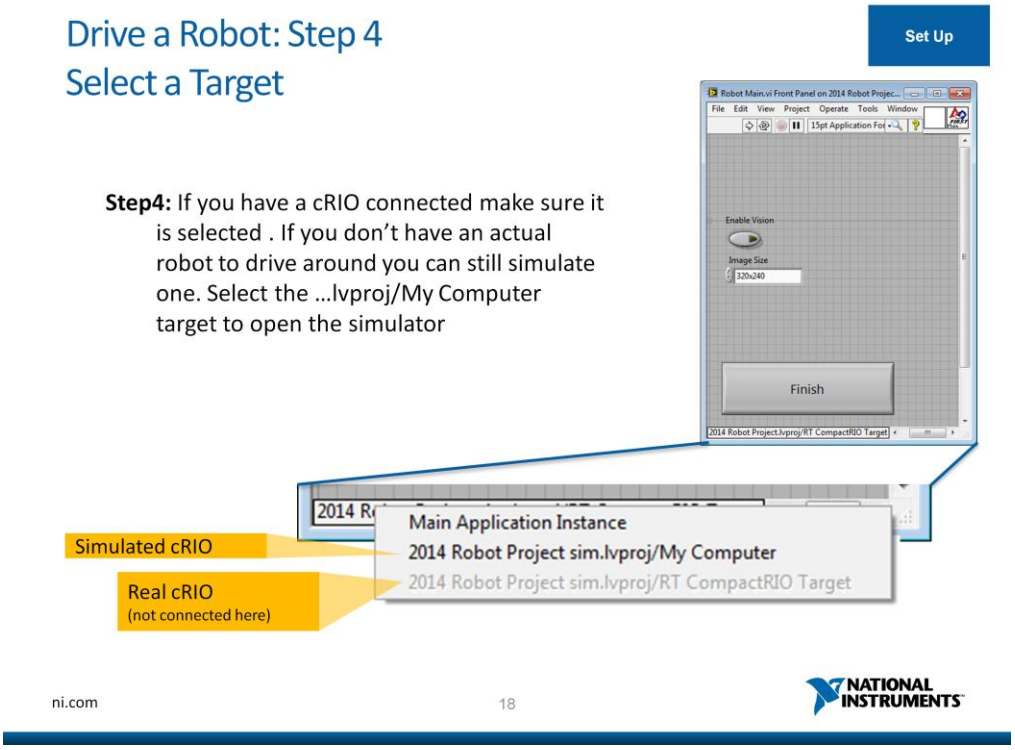
Drive a Robot: Step 3
Open the Template Project

Set Up



Step3: To open **Robot Main.vi** double click on it in the project window. The front panel should launch. Hit **Ctrl+E** to open the block diagram and look at the code.

The default project that is generated is ready drivable as soon as you open it. The default code is simple and just has the drive modes implemented, by default the autonomous code does nothing until you go in and edit it. Robot Main is the master program that will deploy and run all of the team code, for this exercise we just want to run the default code so open Robot main by double clicking on it.



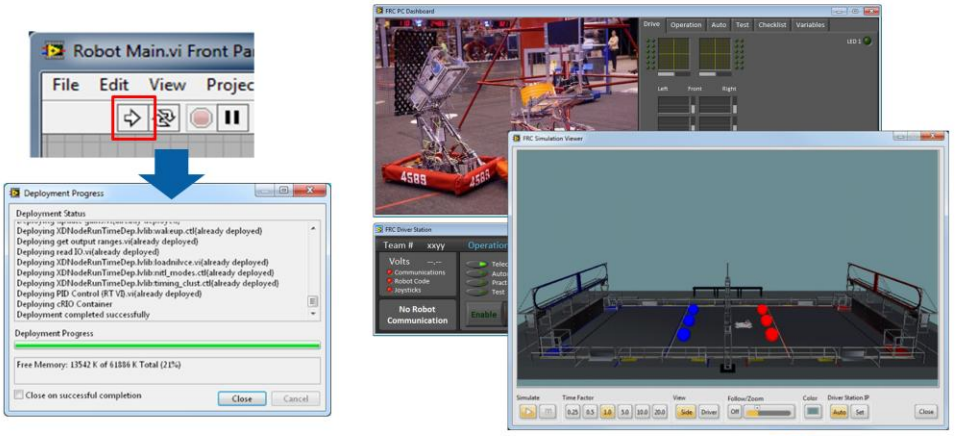
In the bottom left corner of both the front panel (grey) window and the block diagram (white) it lists the target for execution. To change the target right-click on the white box and select your target. When you right click you're changing the deployment environment. It may take a few minutes to load the simulation vi's.

Only a Joystick/Controller is needed to simulate a robot.

Driving a Robot: Step 5
Deploy Code

Set Up

Step5: Click on the **white run arrow** at the top left of Robot Main. The robot project will be deployed to the cRIO/Simulator. Once LabVIEW is finished deploying the code you will need to open up the driver station. The LED's should all be green. Click **TeleOp** button and then **Start**.



NATIONAL INSTRUMENTS

The Run button can be pressed from either the bock diagram or front panel. LabVIEW has to be able to establish a connection to the cRIO in order to deploy to it.

If you are using a real robot; check the three lights in the Driver station to make sure the Joystick, Robot Code, and Communication are all good before enabling TeleOp
Once TeleOp is Enabled the Robot is Live!

Note: Since we ran the code instead of deploying it, the cRIO will only run the code until its reset or power cycled. It will not start up automatically unless the application is built and set to run at start-up

Set Up Resources

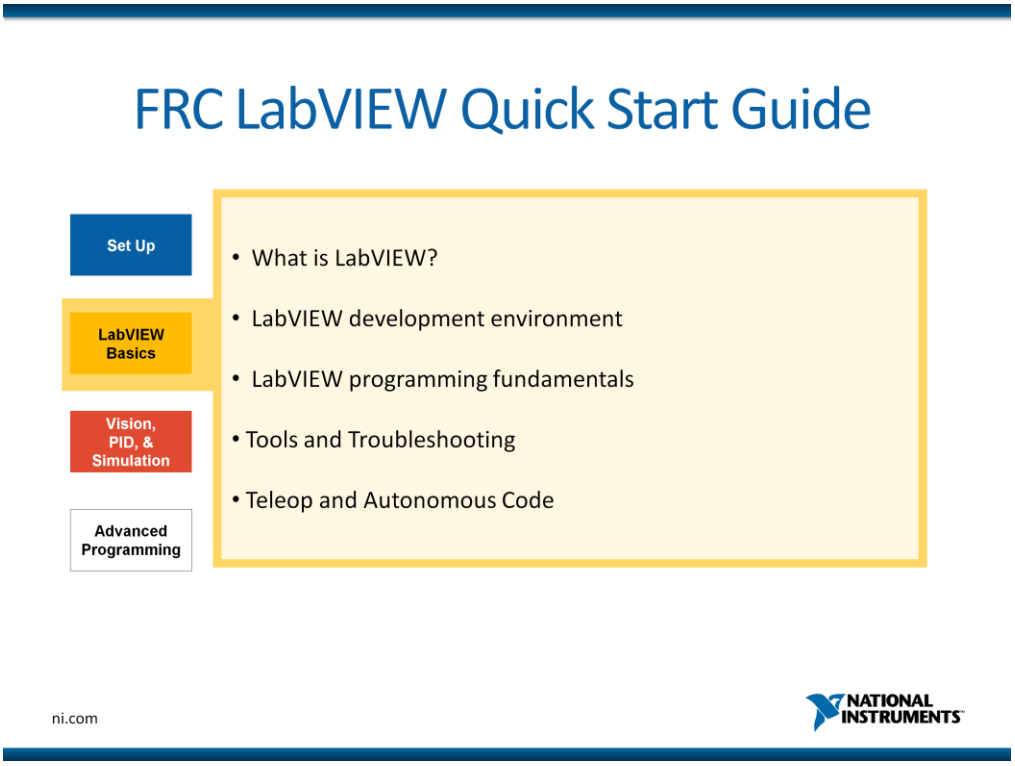
Set Up

- Software Set Up Guide
- LabVIEW Development Suite for FRC 2014
- Imaging cRIO & Windows IP Configuration
- Troubleshooting cRIO connections
- Driver Station Tutorial
- Custom Dashboard Tutorial
- Robotics Framework Tutorial
- Robot Simulation Tutorial

All links available at NI.com/FRC
Click this button

- Forum Support
  - NI.com/FRC – year round
- Phone Support
  - 1(866)511-6285 from 1pm to 7pm (CST) - During build season only

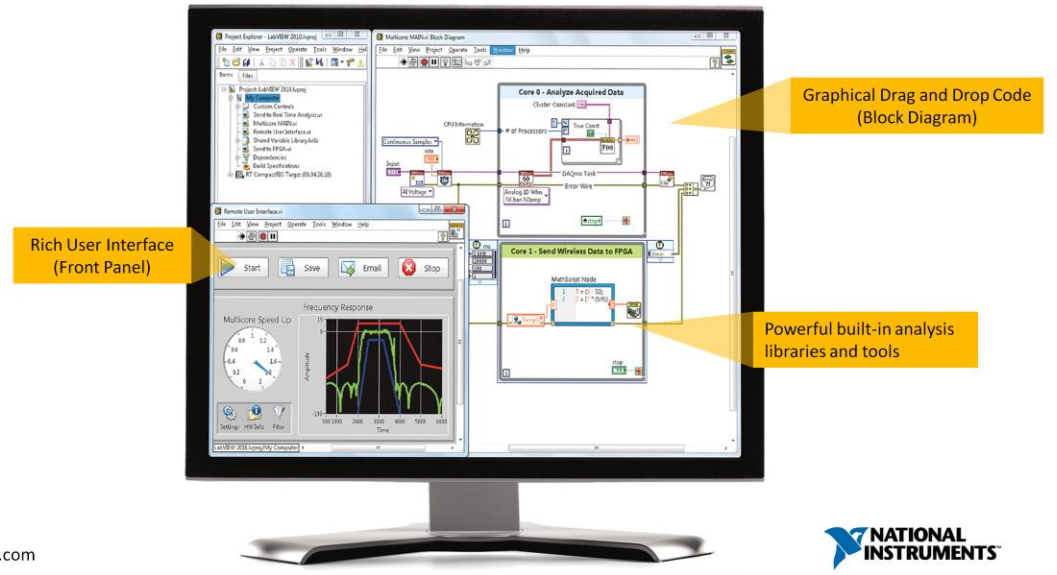
This presentation is just meant to give you a quick overview of all the systems. If you need more help on any of the steps we covered please visit NI.com/FRC and go to the LabVIEW training section



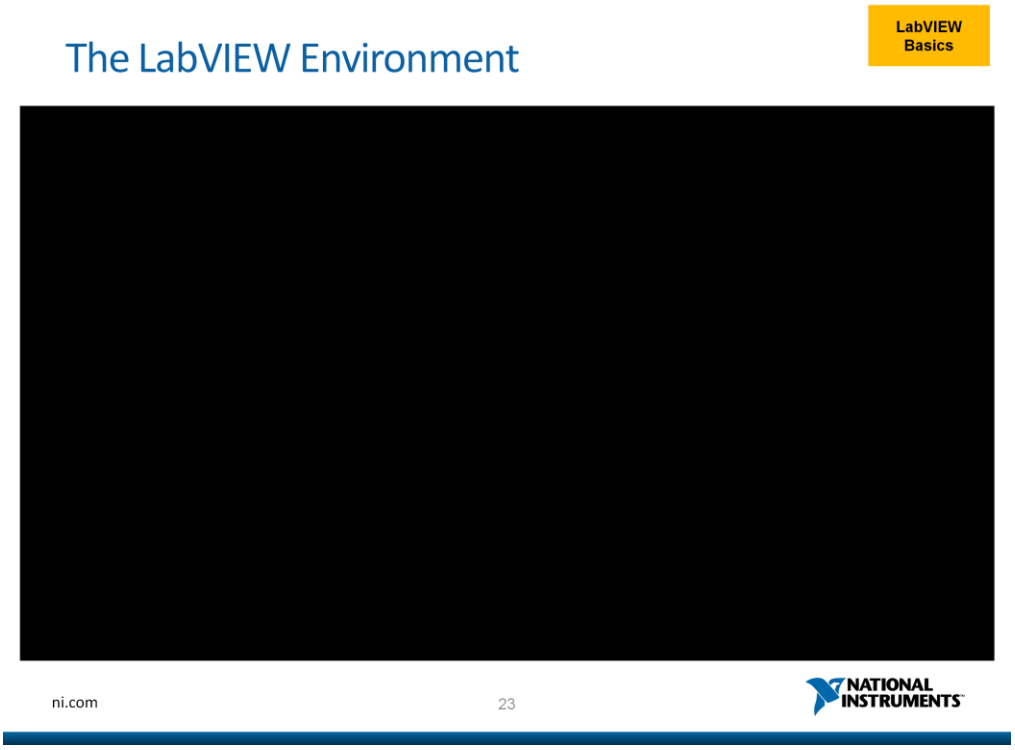
The LabVIEW Basics module will introduce you to the LabVIEW development environment and the programming basics. We will also cover some basic LabVIEW debugging techniques and look through some of the FRC specific code.

What Is LabVIEW?

– A graphical programming environment used to develop sophisticated measurement, test, and control systems.

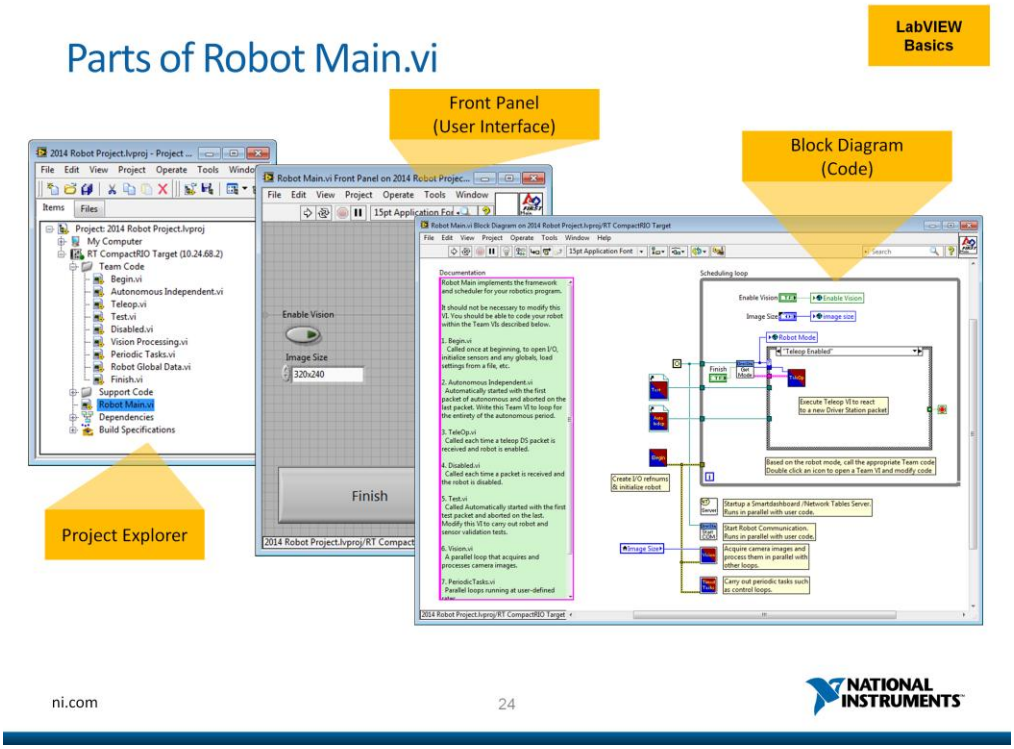


LabVIEW is a graphical programming language quite different from traditional programming languages like C++ and Visual Basic. As we'll learn, one of the greatest strengths of LabVIEW is how intuitive it is to program. We'll see how the Virtual Instrument approach to programming, employed by LabVIEW, makes it possible to create sophisticated and powerful programs with elegant, graphical user interfaces.



This video will give a good general overview of the LabVIEW programming environment

If the video is missing or won't play in powerpoint watch it here
http://www.ni.com/academic/students/learn-labview/environment/



As you will see in the next exercise, when you configure a new project in LabVIEW FRC, a default code template will be available. By using this core template, you should be able to drive the robot in Arcade mode without making any code changes. In the coming slides, we will investigate the block diagram (code) of the Robot Main.vi to understand the key components and their functions.

The **Front Panel** is the user interface for the program where you place **Controls** to input values to the system and **Indicators** to display outputs to the user.

The **Block Diagram** contains the graphical source code.

Icon – Graphical representation of a VI
Connector Pane – Map of the inputs and outputs of a VI

The **Icon/Connector Panes** are necessary to use a VI as a subVI. The connector pane is a map of the inputs and outputs of a VI.

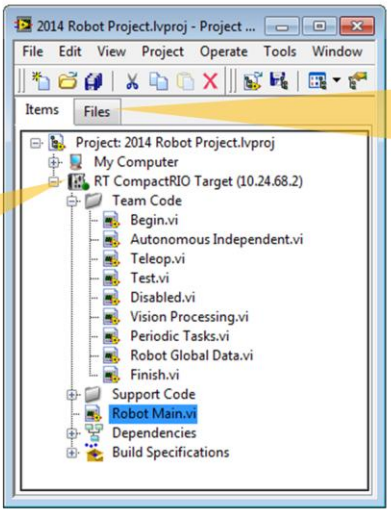
A subVI is similar to a subroutine or function in a text-based programming language.

LabVIEW Basics

Project Explorer

LabVIEW file extensions:

- LabVIEW project —.lvproj
- Virtual instrument (VI) — .vi
- Custom control — .ctl



File Location in Wondows

Right-Click to connect to cRIO

You can use the LabVIEW project for accessing and organizing project files. You view project files in the **Project Explorer** window. Project files can include LabVIEW and non-LabVIEW files.

You can use LabVIEW to build an executable and installer. Various hardware courses will use the project explorer to configure and deploy applications to a target.

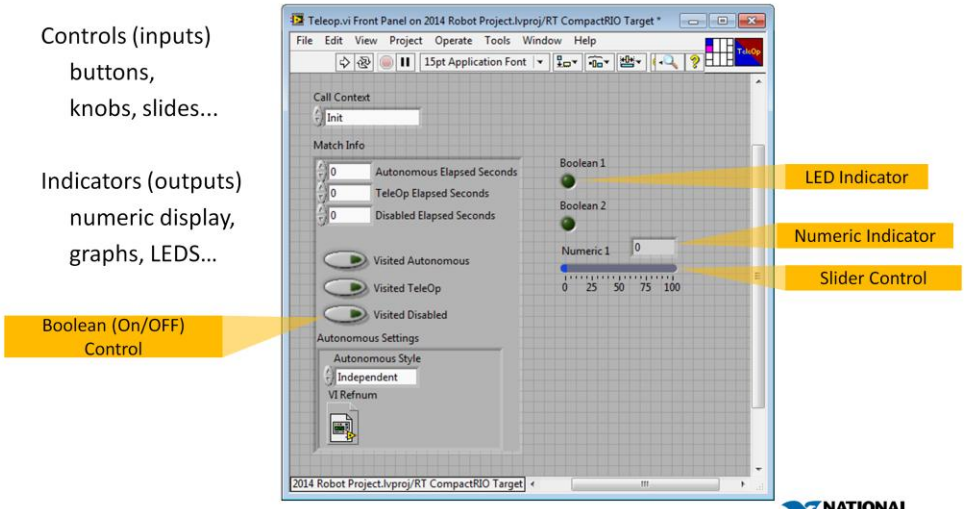
LabVIEW Basics

Parts of a VI – Front Panel

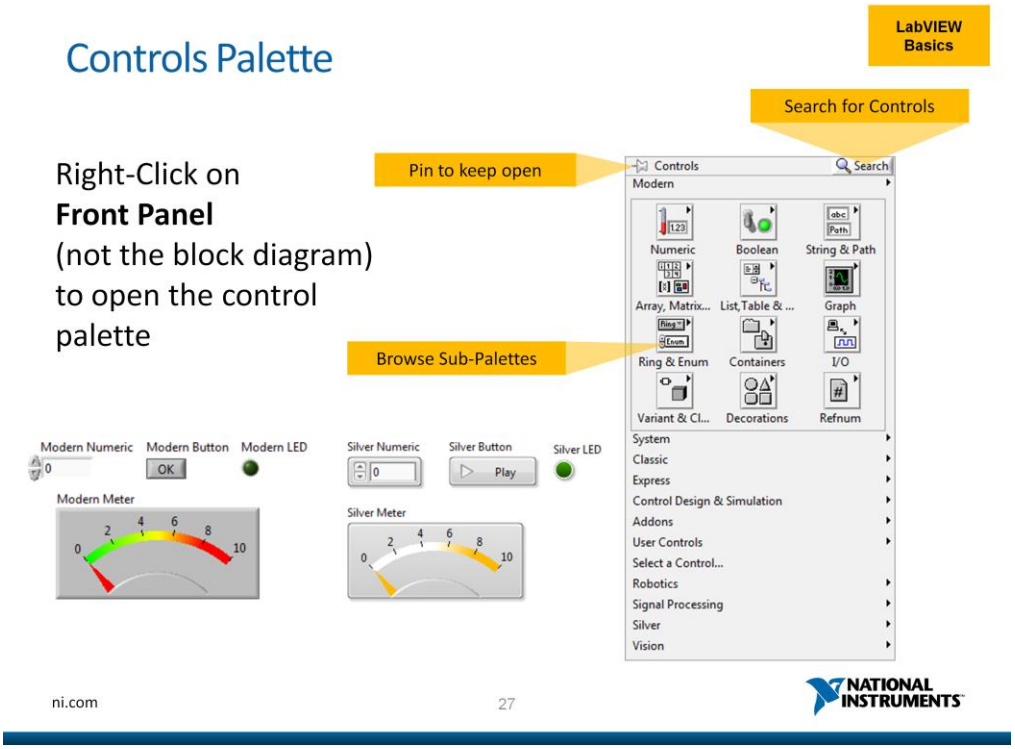
The front panel is the VIs graphical user interface with inputs and outputs.

Controls (inputs)
buttons,
knobs, slides...

Indicators (outputs)
numeric display,
graphs, LEDS...



Discuss how a front panel is the user interface of the VI and contains controls and indicators. A section following this slide concentrates on the front panel in more depth.



The controls palette gives you access to all of the controls and indicators that are placed on the front panel. Right-Click on the FRONT PANEL (not the block diagram) to browse the controls palette.

The thumbtack in the top left corner allows you to lock the front panel so it will stay open

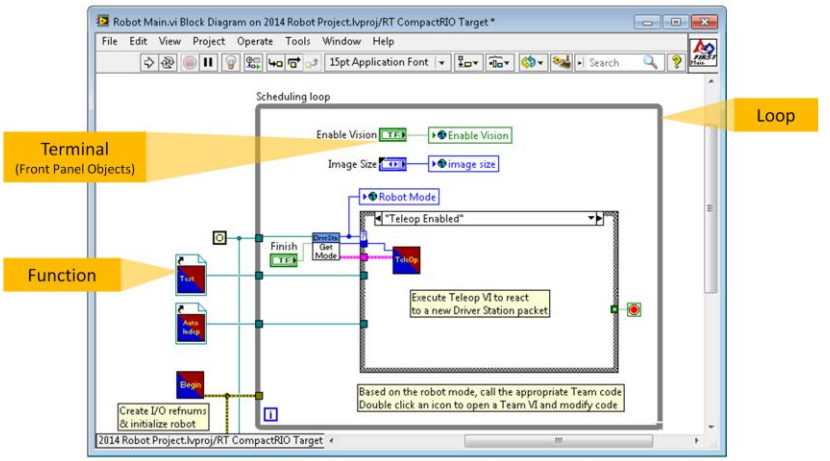
The modern, silver, classic, and system palettes all have a different look and feel, but they all function the same.

You can even create custom controls and indicators using your own graphics, there is a link to a tutorial in the references section.

LabVIEW Basics

Parts of a VI – Block Diagram

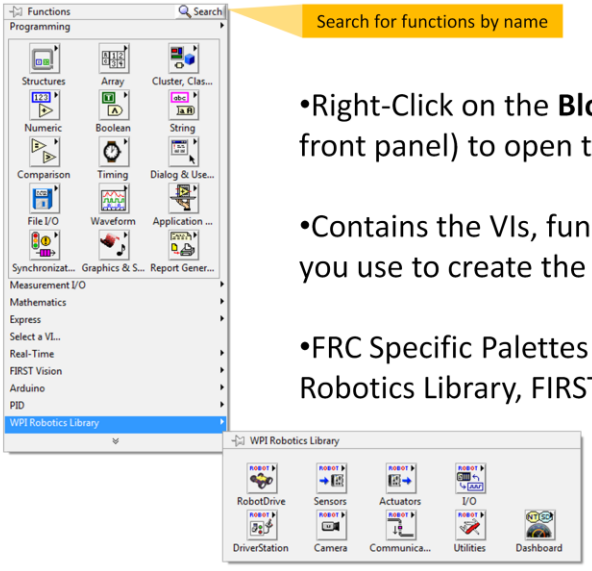
The block diagram is the code for each VI. In LabVIEW graphical blocks are connected with wires to control the execution.



The Block diagram is the graphical code that controls how the program executes. Block diagrams are made up of functions, terminals and constants connected by wires. Like any programming language LabVIEW has loops, case structures and conditional statements, but in LabVIEW its all graphical.

LabVIEW Basics

Functions Palette



Search for functions by name

- Right-Click on the **Block Diagram** (not the front panel) to open the functions palette
- Contains the VIs, functions, and constants you use to create the block diagram.
- FRC Specific Palettes include the WPI Robotics Library, FIRST Vision, and PID

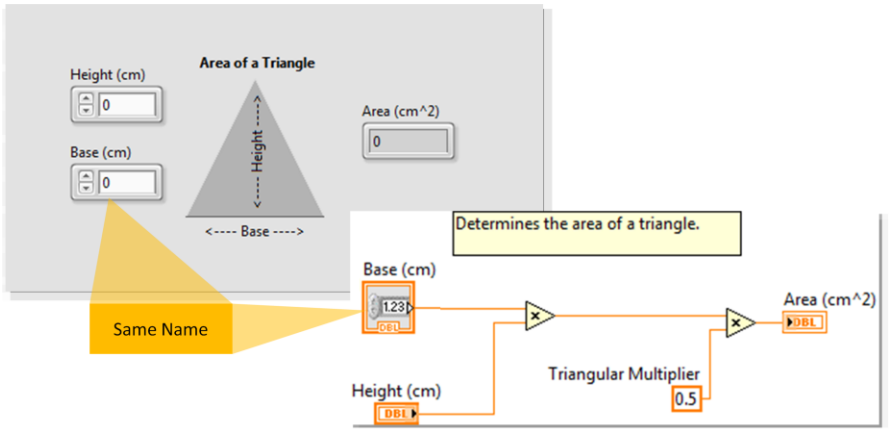
Open the functions palette by right-clicking on the BLOCK DIAGRAM (not the front panel), you can browse through hundreds of functions sorted by type or search directly for a function by name using the search button in the top right corner. When you install the FRC 2014 Update Package you will see a few extra palettes specific to FIRST

The FIRST Vision, PID, WPI Robotics Library all include functions you will need to program your robot.

LabVIEW Basics

Terminals

Terminals allow you to read and write to front panel controls and indicators.



Data values you enter on the front panel controls enter the block diagram through the control terminals on the block diagram.

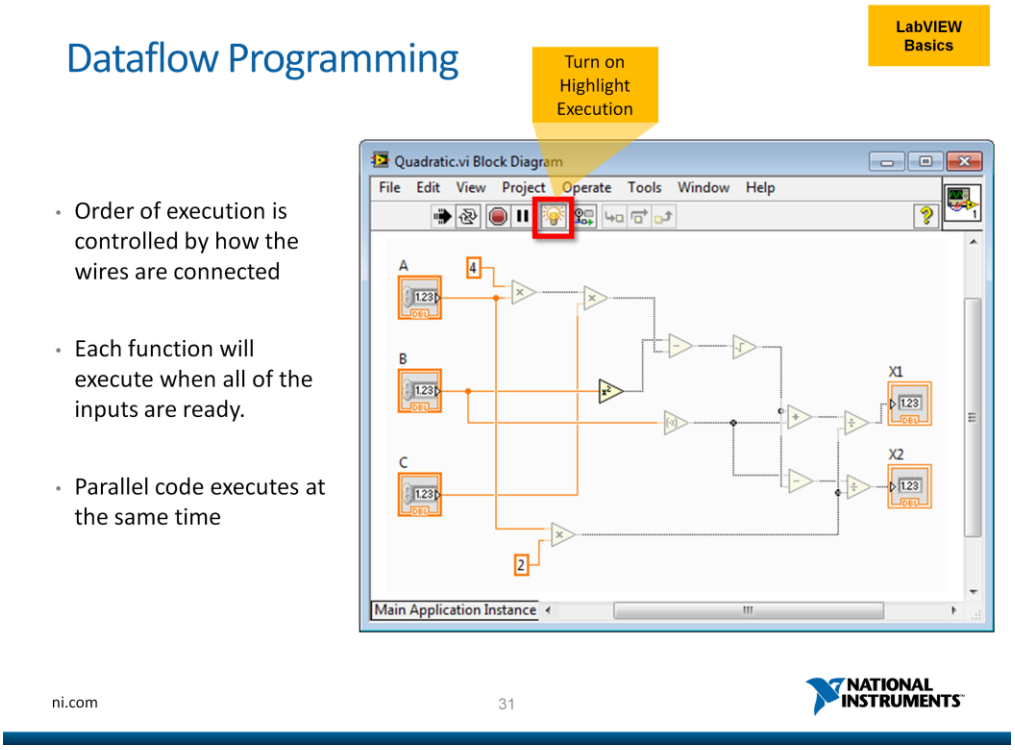
During execution, the output data values from the block diagram pass from indicator terminals to the front panel indicators.

Front panel items share the same label name as the block diagram terminal, when you change one both change.

Notice the visual difference between controls and indicators on the block diagram, the little arrow on the terminal shows were the wire will connect controls output the value on the right side and indicators read the value from the left side.

Note that constants, (the Triangular Multiplier) are available only on the block diagram. You can right-click a terminal to change it between a control, indicator, or constant

Terminals can be big or small, in this image the Base terminal is large and shows the datatype (double percision, DBL) and the control type (numeric). When the terminals are small they only show the datatype.

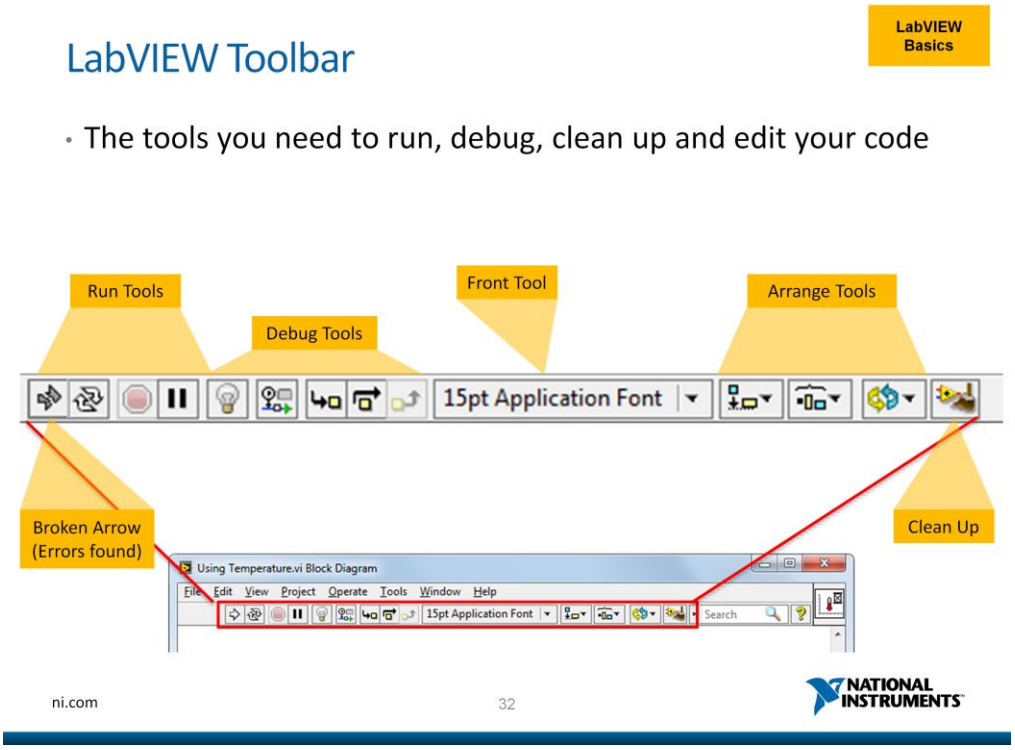


LabVIEW uses dataflow programming, where the order of execution is controlled by how the nodes are wired together. Each function will execute once all of the input parameters are ready. Code does not necessarily execute from left to right, but it is easier to read if you lay it out that way.

You can open up any VI and run it in “highlight execution” mode by clicking the little light bulb on the block diagram toolbar. This will show you how the code is executing as well as the intermediate values

WARNING: When you run code in highlight execution mode it SLOWS DOWN the execution speed dramatically. If you are using this with code running on the cRIO it will causes a timeout on the FIRST safety features (watchdog).

More on Dataflow programming:
http://www.ni.com/gettingstarted/labviewbasics/dataflow.htm



The LabVIEW toolbar can be found on both the front panel and block diagram window. Some of the tools appear on both bars while others are specific to one window.

Click the **Run** button to run the VI. LabVIEW compiles the VI, if necessary. You can run a VI if the **Run** button appears as a solid white arrow.

[animation]
If the **Run** button appears broken, the VI contains edit-time errors. Click the broken arrow to display the errors

The Run Tools include

- Run – Execute the VI
- Run Continuously – Execute the VI over and over
- Abort- Quit the VI immediately (note this should not be used as a normal stop button because it does not allow the code to stop nicely.
- Pause – halt the execution

Font tool – Font style, color, size, formatting, bold…

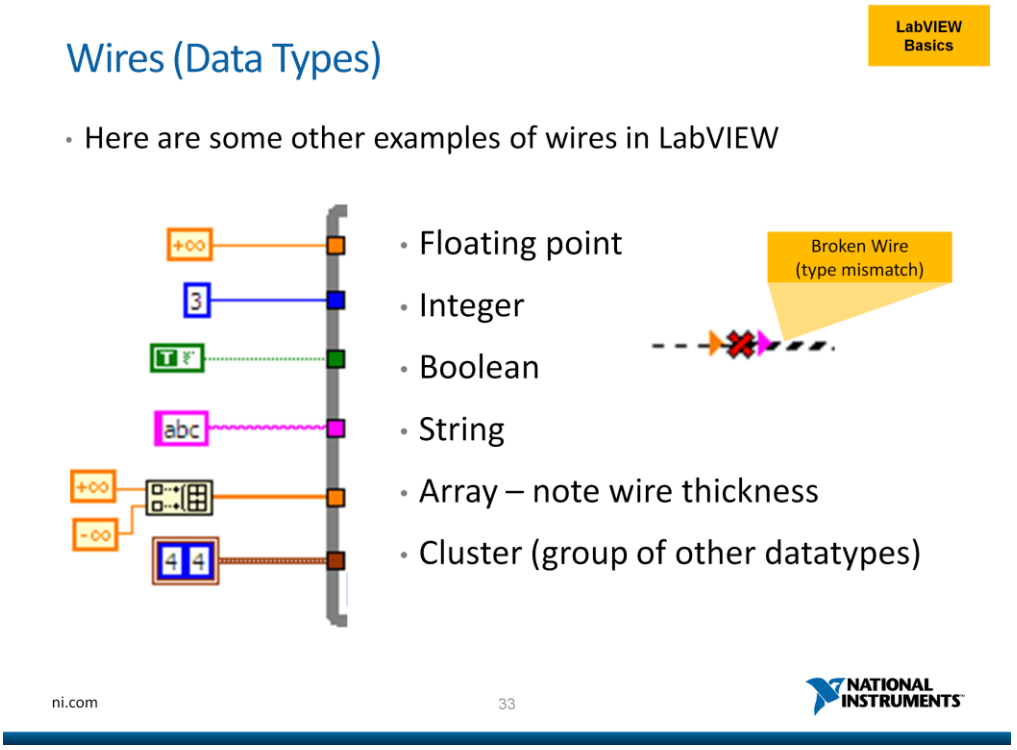
Arrange tools – allow you to align distribute and reorder elements, (just like in powerpoint)

- Align – line up elements along one edge
- Distribute – spread a group of elements out evenly
- Resize – IE make all selected elements as narrow as the narrowest one

Debug Tools

- Highlight Execution – Show dataflow execution
- Retain Wire Values – store executed values in the wire for debugging
- Step into, Step over, step out – execute the VI one block at a time

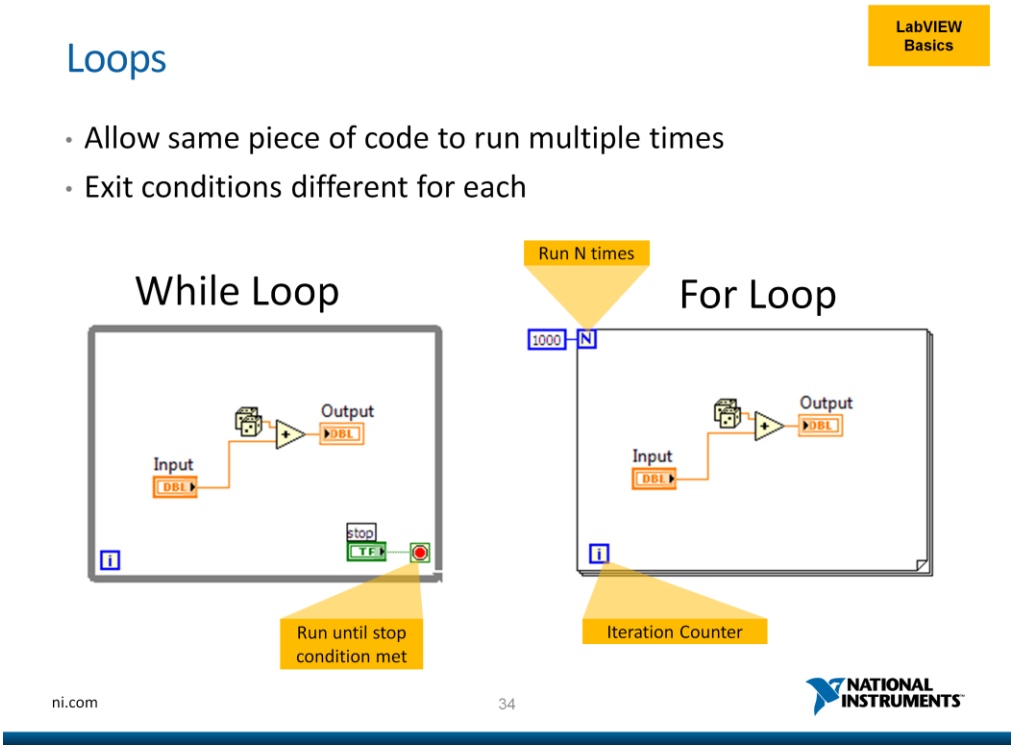
Clean Up – Make the selected Block Diagram elements pretty



In LabVIEW the color and width of the wires indicates the data type. Thicker wires indicate arrays.
These are all examples of block diagram constants of different types

If you connect the two different data types or if you connect to inputs together you will get a broken wire. The image here shows you you have connected a floating point (orange) to a string (pink). (see the little arrows next to the X)

Press <Ctrl-B> to clear all broken wires

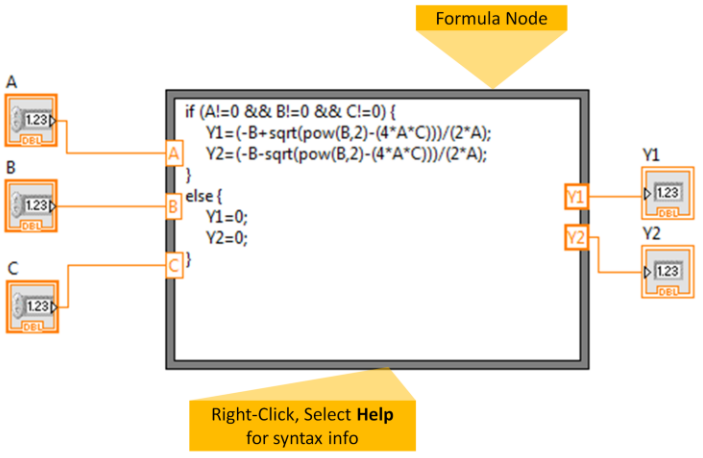


Use While or For Loops to enable sections of your LabVIEW code to run repeatedly. A While Loop will continue to execute until a stop condition is specified. The stop condition can be a simple button press or a series of specific logical conditions. The For Loop will execute a predetermined number of times as specified by the number of iterations you wire to the N input. You may also connect an array wire to the edge of a For Loop and leave the N input unwired. The For Loop's number of iterations will be determined by the array size that is wired at its edge. This is called auto-indexing.

To find the While and For Loops, as well as other control structures, left-click on any empty space on the block diagram and navigate to **Programming»Structures**.

Text Based Math in LabVIEW

LabVIEW Basics



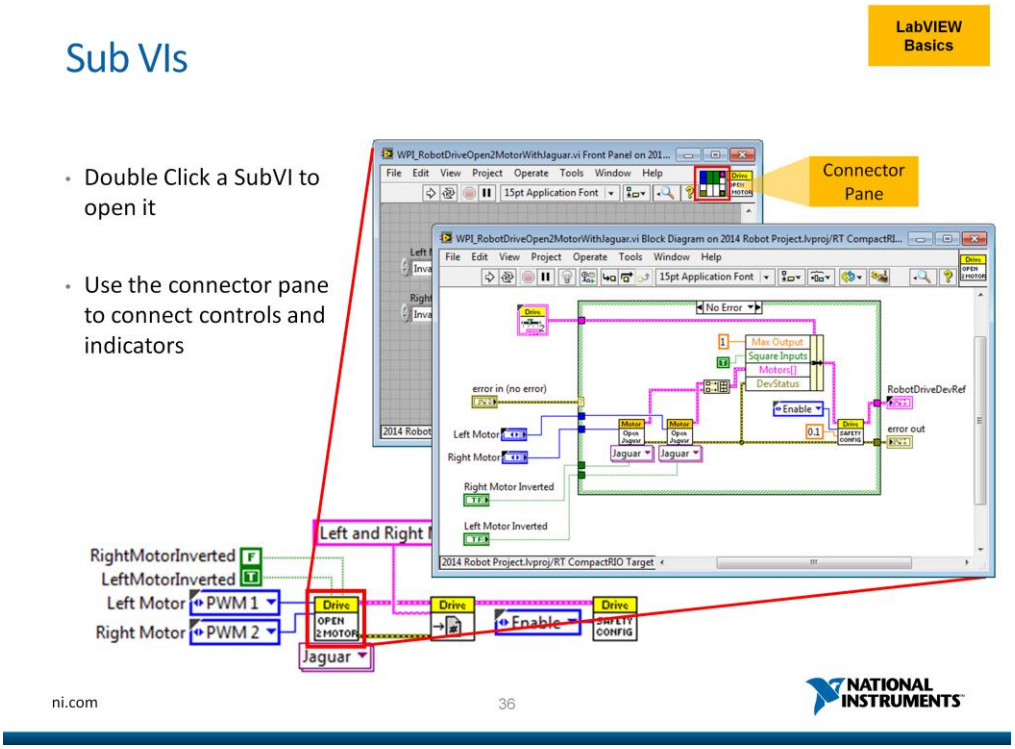
ni.com 35 NATIONAL INSTRUMENTS

Sometimes complex math is easier to understand as a text based equation. LabVIEW allow you to combine Text and Graphical programming using the **Formula Node.** To place a formula node browse to mathematics>Scripts&Formulas and select the formula node from the functions palette. You draw a formula loop just like a while loop in LabVIEW. As you add code the run arrow will indicate any syntax errors. Click the broken run error to see the specific errors. For a guide on syntax right-click the formula node and select Help.

The formula node supports most common math functions as well as basic programming like while loops, for loops, if/else statements and much more.

You can also find the formula node with search, just search “formula”

Formula Node tutorial
http://www.ni.com/white-paper/7572/en/



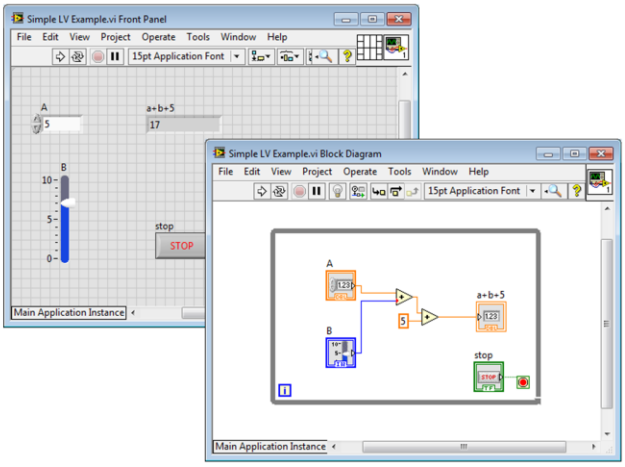
In LabVIEW we use SubVIs to keep code looking clean and making code easy to read. You can double click any subVI to see the underlying code.

In the top right corner of the front panel you can see the connector pane, bind controls and indicators to the connector pane terminals to allow you to wire inputs and outputs to the subVI

LabVIEW Programming Demonstration

Programming Demonstration

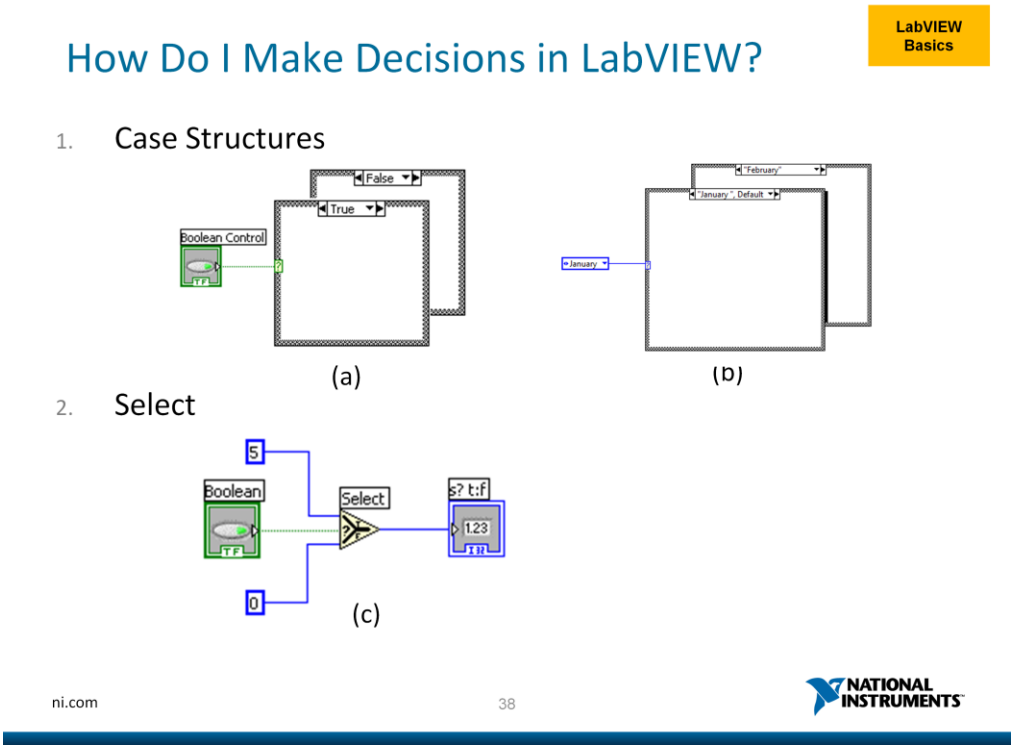
- Front Panel / Block Diagram
- Tools/ Functions Palettes
- Controls / Terminals
- Wiring
- Constants
- Dropping functions
- Run / Run Continuously
- Loops
- Datatypes
- Highlight execution



Take this time to do a basic LabVIEW demonstration by recreating the example above from a blank VI. Below is a scripted order that will cover all of the programming basics we learned in this module. I recommend using this as a demonstration and not a group exercise to give a quick overview and save time. Be sure to narrate your mouse clicks (right vs left) and go slowly so you can show everyone once.

Demo Script

1. Start at the Getting started window and select Blank VI
2. Show the block diagram and the front panel
3. Open up the controls palette and drop a numeric control, name it “A”
4. Drop a numeric slider, name it “B”, show that ther block diagram terminals change as well.
5. Move over to the block diagram and drop two addition blocks
6. Wire A & B into the first addition
7. Wire the result into the second addition
8. Right-Click on the 2nd input of the addition and **select create>constant** (discuss controls vs constants)
9. Right-Click on the output of the addition and **select Create>Indicator** (discuss controls vs indicators)
10. Set the numeric values and **run** the VI
11. Use **Run continuously** to show the difference, then abort the VI (but caution against using abort in robot code)
12. Add a while loop around the code, then right-Click on the stop terminal and select **create>control**
13. Run the code then use the stop button to stop it.
14. Right-Click on the slider and change the **representation to I32** (show that the slider locks to ints now, and the wire changed color)
15. Demonstrate **Highlight Execution**
16. Select the math code and select **Edit>Create SubVI**



There are several ways to make decision in LabVIEW.

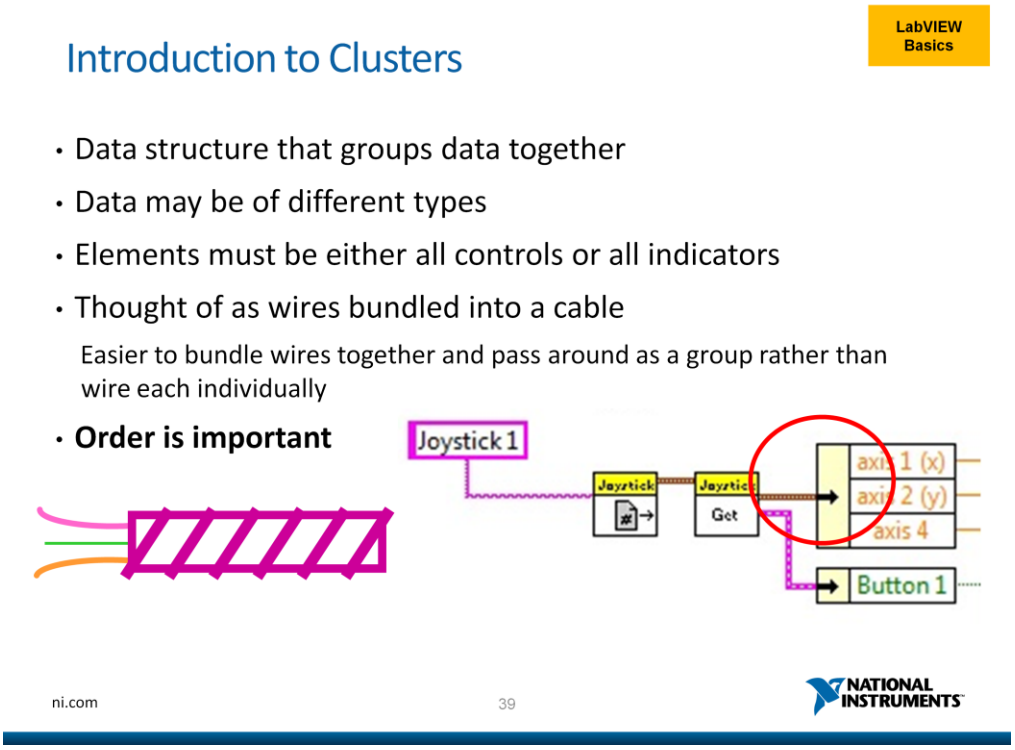
Case Structure

The case structure has one or more subdiagrams, or cases, one of which executes when the structure executes. The value wired to the selector terminal determines which case to execute and can be Boolean, string, integer, or enumerated. Right-click the structure border to add or delete cases. Use the Labeling tool to enter value(s) in the case selector label and configure the value(s) handled by each case. It is found at **Functions»Programming»Structures»Case Structure**. This is equivalent to a If/Else or switch statement

Select

Returns the value wired to the **t** input or **f** input, depending on the value of **s**. If **s** is TRUE, this function returns the value wired to **t**. If **s** is FALSE, this function returns the value wired to **f**. The connector pane displays the default data types for this polymorphic function. It is found at **Functions»Programming»Comparison»Select**.

- **Example A:** Boolean input. Simple if-then case. If the Boolean input is TRUE, the true case executes; otherwise the FALSE case executes.
- **Example B:** Numeric input. The input value determines which box to execute. If out of range of the cases, LabVIEW chooses the default case.
- **Example C:** When the Boolean passes a TRUE value to the Select VI, the value 5 is passed to the indicator. When the Boolean passes a FALSE value to the Select VI,
  0 is passed to the indicator.
- This is equivalent to the Question Mark (?) Operator in C++



Clusters group like or unlike components together. They are equivalent to a *record* in Pascal or a *struct* in ANSI C.
Cluster components may be of different data types.

Examples

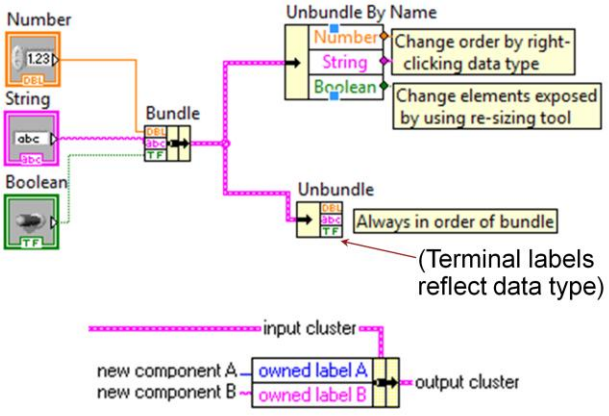
- Error information—Grouping a Boolean error flag, a numeric error code, and an error source string to specify the exact error.
- User information—Grouping a string indicating a user's name and an ID number specifying the user's security code.

All elements of a cluster must be either controls or indicators. You cannot have a string control and a Boolean indicator. Think of clusters as grouping individual wires (data objects) together into a cable (cluster).

Cluster Functions

LabVIEW Basics

- In the Cluster & Variant subpalette of the Programming functions palette
- Can also be accessed by right-clicking the cluster terminal



You can use bundle and unbundle functions to get data in and out of clusters.

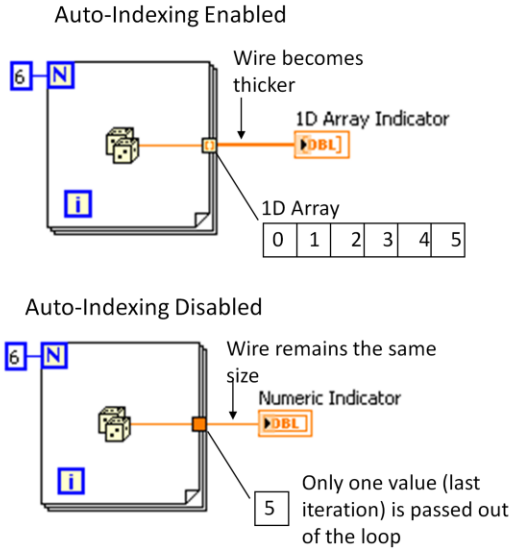
The terms bundle and cluster are closely related in LabVIEW.
Example: You use a bundle function to create a cluster. You use an unbundle function to extract the parts of a cluster.

Bundle—Forms a cluster containing the given objects in the specified order.
Bundle by Name—Updates specific cluster object values (the object must have an owned label).
Unbundle—Splits a cluster into each of its individual elements by data type.
Unbundle by Name—Returns the cluster elements whose names you specify.

Building Arrays with Loops

LabVIEW Basics

- Loops can accumulate arrays at their boundaries with auto-indexing
- For Loops auto-index by default
- While Loops output only the final value by default
- Right-click tunnel and enable/disable auto-indexing



For Loops and While Loops can index and accumulate arrays at their boundaries. This is known as auto-indexing.

- The indexing point on the boundary is called a tunnel
- The For Loop is auto-indexing-enabled by default
- The While Loop is auto-indexing-disabled by default

Examples

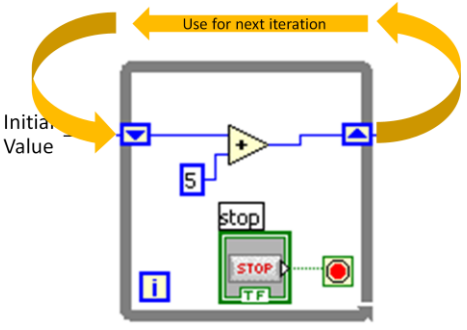
- Enable auto-indexing to collect values within the loop and build the array. All values are placed in the array upon exiting the loop.
- Disable auto-indexing if you are interested only in the final value.

- The index tunnels also work in reverse if you want to input an array and read one value on each iteration.

Shift Register - Access Previous Loop Data

LabVIEW Basics

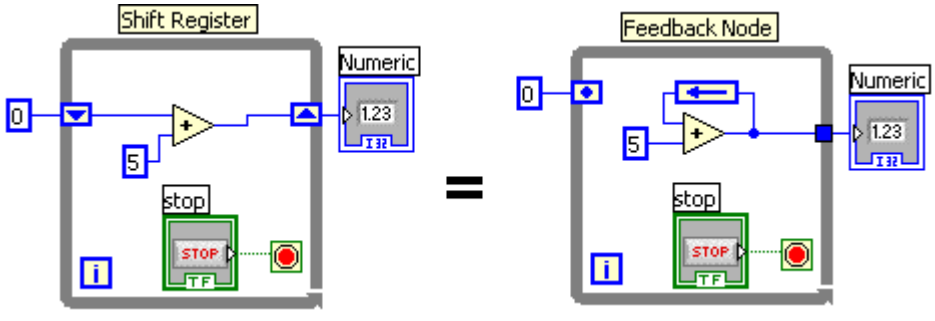
- Available at left or right border of loop structures
- Right-click the border and select Add Shift Register
- Right terminal stores data on completion of iteration
- Left terminal provides stored data at beginning of next iteration



Shift registers transfer data from one iteration to the next:

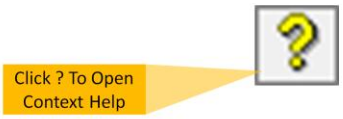
- Right-click on the left or right side of a For Loop or a While Loop and select Add Shift Register.
- The right terminal stores data at the end of an iteration. Data appears at the left terminal at the start of the next iteration.
- A shift register adapts to any data type wired into it.

An input of 0 would result in an output of 5 on the first iteration, 10 on the second iteration, and 15 on the third iteration. Said another way, shift registers are used to retain values from one iteration to the next. They are valuable for many applications that have memory or feedback between states.



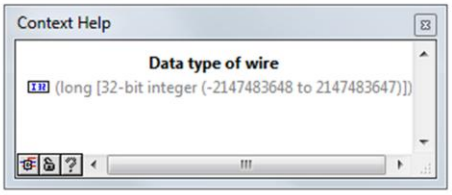
See **Help»**

Context Help

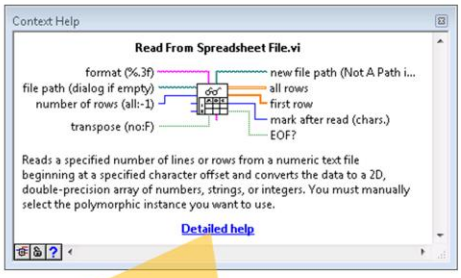


LabVIEW Basics

- Displays basic information about wires and nodes when you hover over with your mouse.



- Turn Context Help On/Off
  - Click the yellow ? on the top right corner of your VI
  - Press <Ctrl-H>
  - Select **Help»Show Context Help** from the LabVIEW menu.



NATIONAL INSTRUMENTS

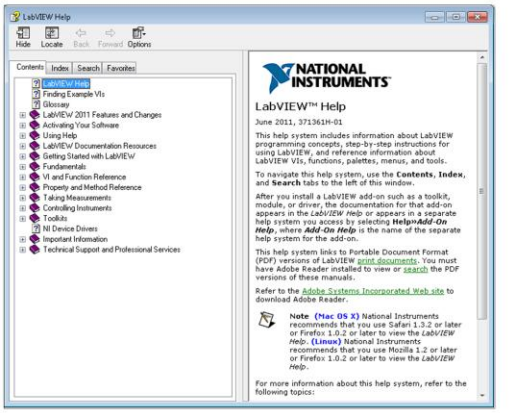
Context help is a very useful tool especially when you are using functions for the first time. Press the yellow question mark in the top right corner of your VI to open or close the window. Use your mouse to hover over a function or a wire to display information about it. You can click “Detailed help” to open up the Full LabVIEW help documentation with more information.

LabVIEW Basics

LabVIEW Help

- Contains detailed descriptions and instructions for most palettes, menus, tools, VIs, and functions.

- Open LabVIEW Help by
  - Selecting **Help»LabVIEW Help** from the menu
  - Clicking the **Detailed help** link in the **Context Help** window
  - Right-clicking an object and selecting **Help** from the shortcut menu.
  - Press **F1**

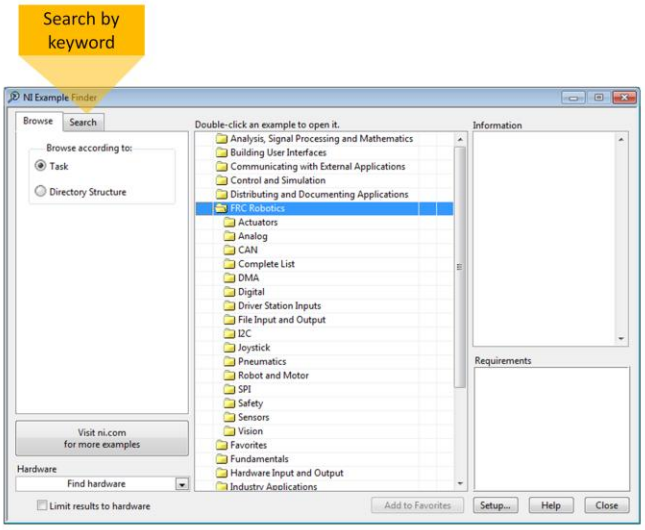


The LabVIEW help is full of useful information on how to use functions, tutorials, and tips. Open the help for a specific element from the context help, or by right-clicking and selecting help, or open the help and use the built in search tools to find the information you are looking for.

Find Examples

LabVIEW Basics

- Find FRC templates in the Support Tab>>Find FRC Examples…
- Or Help>>Find Examples and browse to the FRC Robotics Folder

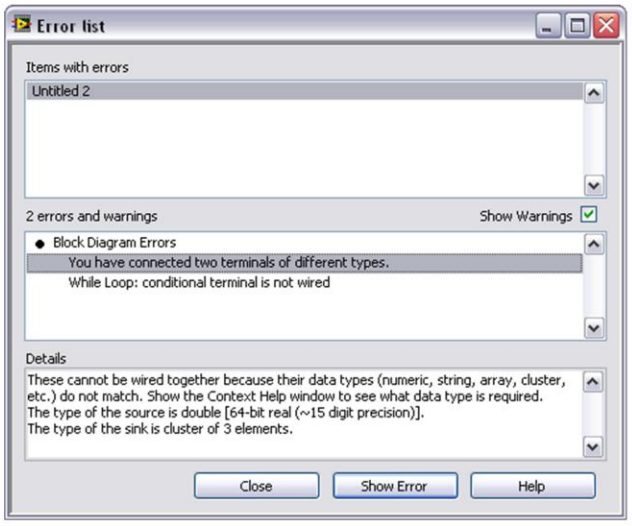


FRC LabVIEW includes hundreds of examples both specific to FRC as well as general LabVIEW examples. To open the example finder from the getting started window select Support and click Find FRC Examples or from any LabVIEW Window select help>>Find Examples and browse to the FRC Robotics folder.

[animation]
All of the examples include instructions and configuration information so you can drop it into your Robot Project.

Debugging: Correcting Broken VIs

Broken Run arrow > VI cannot be compiled > VI cannot be executed



One of the most common debugging features used in LabVIEW is the compiler which is always on. While you are developing a program, the compiler continuously checks for errors and provides semantic and syntactic feedback on the application.

When the **Run** button is not broken, the VI is compiled and can be executed. If an error exists, you cannot run the program. You see a broken **Run** button in the toolbar. The VI is not executable.

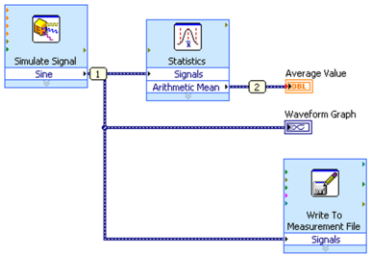
You must resolve any errors noted in the **Errors List** window before you can run the VI.

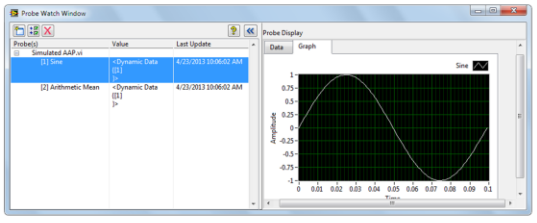
Instructors: This is a good opportunity to remind students that LabVIEW is a compiled language.

LabVIEW Basics

Debugging: Probes

- Use the Probe tool to observe intermediate data values and check the error output of VIs and functions, especially those performing I/O.
- Specify to retain the values in the wires so that you can probe wires for data after execution.

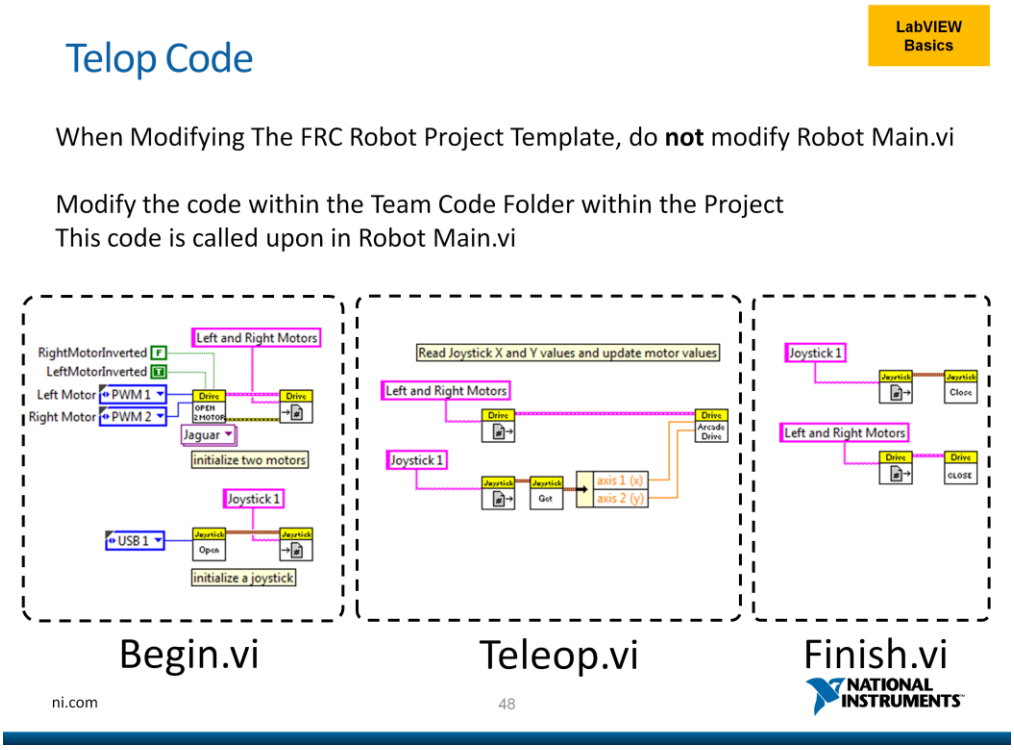




Probes allow you to see the value on a wire as a VI executes, unlike highlight execution this allows your code to operate at normal speed so you can watch it run. Create as many probes as you need, and the corresponding number in the Probe Watch Window will display the data.

To create a probe right-click on a wire and select “probe”

More Details: http://zone.ni.com/reference/en-XX/help/371361H-01/lvhowto/using_the_probe_tool/



Now that you know the basics here are the primary VIs you will be editing.

Begin.vi

This subVI is called once at the start of the Robot Main.vi.
Its purpose is to initialize and define all of the motors, sensors, I/O, and other items connected to your robot.
The default FRC robot code initializes the camera settings, two motors, and joystick.
Turn on the **Context Help** by pressing <Ctrl-H>. Then hover over each function on the block diagram to see its name and description.

Teleop.vi

To see the block diagram of the Teleop SubVI, select Teleop Enable in the case structure and then double-click it in the Robot Main. The front panel of the Teleop SubVI will open.
Switch to its block diagram.
This is the code that reads the joystick and makes the robot move in Arcade Drive.

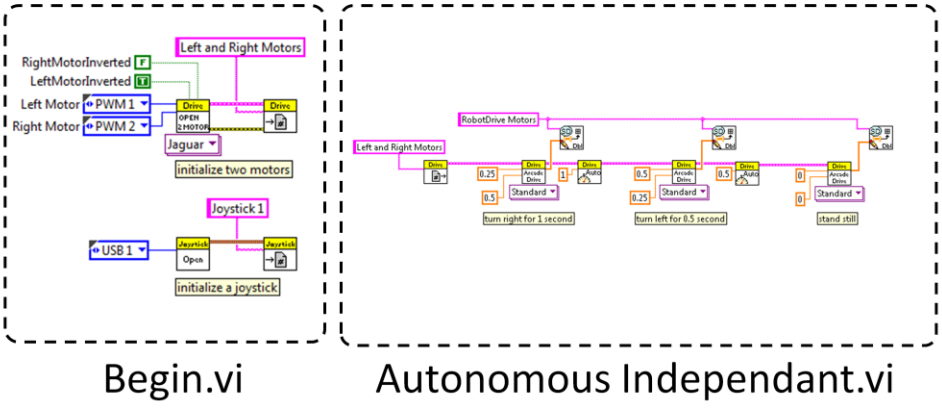
Finish.vi

To see the block diagram of Finish SubVI, select Finish in the case structure and then double-click it in Robot Main.
The default FRC robot code closes the reference of the camera settings, two motors, and joystick.
Whenever a device is added to Begin.vi, you should also close that device in Finish.vi.

LabVIEW Basics

Autonomous Code

- Similar to Teleop, Autonomous executes code by references



Begin.vi Autonomous Independant.vi

In Autonomous mode, the Robot Main will run the same Begin.vi then start your Autonomous Independent VI.

Learn Your Hotkeys

NATIONAL INSTRUMENTS

LabVIEW Quick Reference Guide

Keyboard Shortcuts

File

Ctrl-N	Create new VI
Ctrl-S	Save VI
Ctrl-P	Print

Edit

Ctrl-V	Paste object
Ctrl-U	Clean up diagram
Ctrl-Space	Activate quick drop
Ctrl-B	Remove broken wires
Ctrl-C	Copy an object
Ctrl-X	Cut object
Ctrl-Z	Undo last action
Ctrl-Shift-Z	Redo last action

Operate

Ctrl-R	Run VI
Ctrl-.	Abort VI

Window

Ctrl-E	Display block diagram/ front panel
Right Click	Display controls/ functions palette
Shift-Right Click	Display tools palette
Ctrl-T	Tile block diagram and front panel windows

Help

Ctrl-H	Display context help

Tools Palette

Tool	Icon	Description
Automatic Tool Selection		Automatically choose the appropriate tool
Operating Tool		Change the value of a control or select the text within a control
Positioning Tool		Position, resize, and select objects
Labeling Tool		Edit text and create free labels
Wiring Tool		Wire objects together on a block diagram
Scroll Tool		Scroll the window without using the scroll bars
Breakpoint Tool (used for debugging)		Set breakpoints on VIs, functions, wires, loops, sequences, and cases
Probe Tool (used for debugging)		Create probes on wires and display intermediate values on a wire in a running VI
Color Copy Tool		Copy colors for pasting with the Color Tool
Color Tool		Set the foreground and background colors

Editing Tools

Tool	Icon	Description
Show Context Help		Display the context help window
Text Settings		Change the font setting for the VI, including size, style, and color
15pt Application Font		
Align Objects		Align selected objects
Distribute Objects		Space objects evenly
Resize Objects		Resize multiple front panel objects to the same size
Reorder		Reorder the layers of the objects
Clean Up Diagram		Rearrange wires and objects on the block diagram
Enter		Appears when a new value is available to replace an old value

Debugging Tools

Tool	Icon	Description
Run		Execute the VI
List Errors		List errors that prevent the VI from running
Run Continuously		Execute the VI continuously until abort or pause is pressed
Stop		Stop VI execution immediately
Execution Highlighting		Animate data movement on the block diagram wires
Pause		Temporarily stop execution to debug a portion of the VI
Step Into		Single-step into a subVI or structure to debug it
Step Over		Execute a subVI or structure and resume single-stepping in next main function
Step Out		Execute a subVI or structure and resume single-stepping in calling VI or structure

This is the LabVIEW Quick Reference Guide, with helpful information about the LabVIEW environment and hotkeys. You can download this at NI.com/FRC. Irecomend printing one out and keeping it your programming area for use throughout the season.

LabVIEW Basics

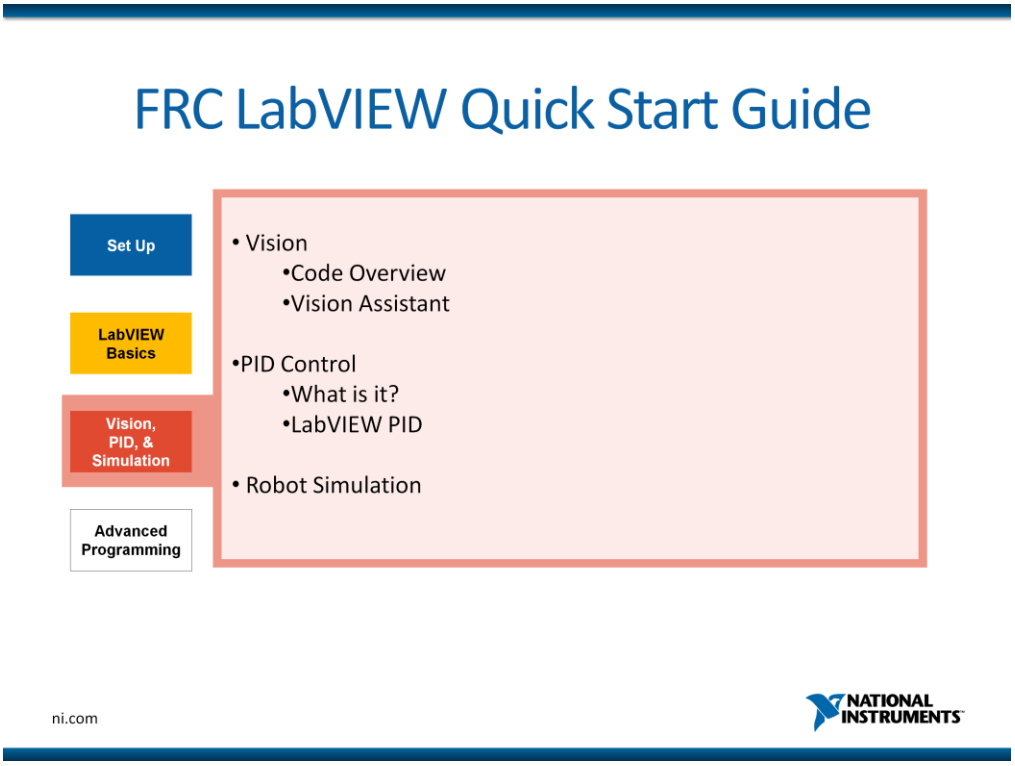
Set Up Resources

- Learn LabVIEW - Video Training
- LabVIEW Environment Overview
- Creating custom controls
- LabVIEW Dataflow
- LabVIEW Tools
- LabVIEW Data Structres
- LabVIEW Debugging
- WPI Library Overview

All links available at NI.com/FRC
Click this button

- Forum Support
  - NI.com/FRC – year round
- Phone Support
  - 1(866)511-6285 from 1pm to 7pm (CST) - During build season only

This presentation is just meant to give you a quick overview of all the systems. If you need more help on any of the steps we covered please visit NI.com/FRC and click on the FRC Quick Start Guide Icon at the top, you can access this powerpoint and all of the links for additional training info.



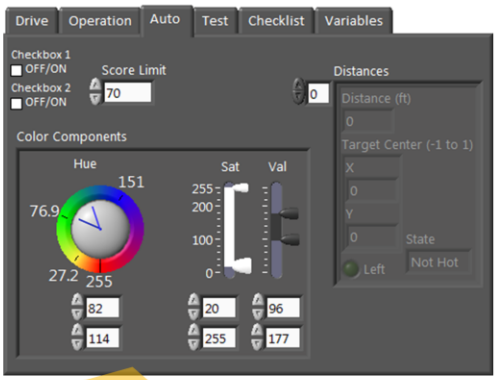
This module covers the Vision, PID and Simulation. While these are not required for competition these tools will allow you to build more advanced, smarter robots that can dynamically react to the game. These techniques are important in developing an advanced autonomous mode.

Vision

NI Vision provides tools to make your robot smarter.

Vision Uses:

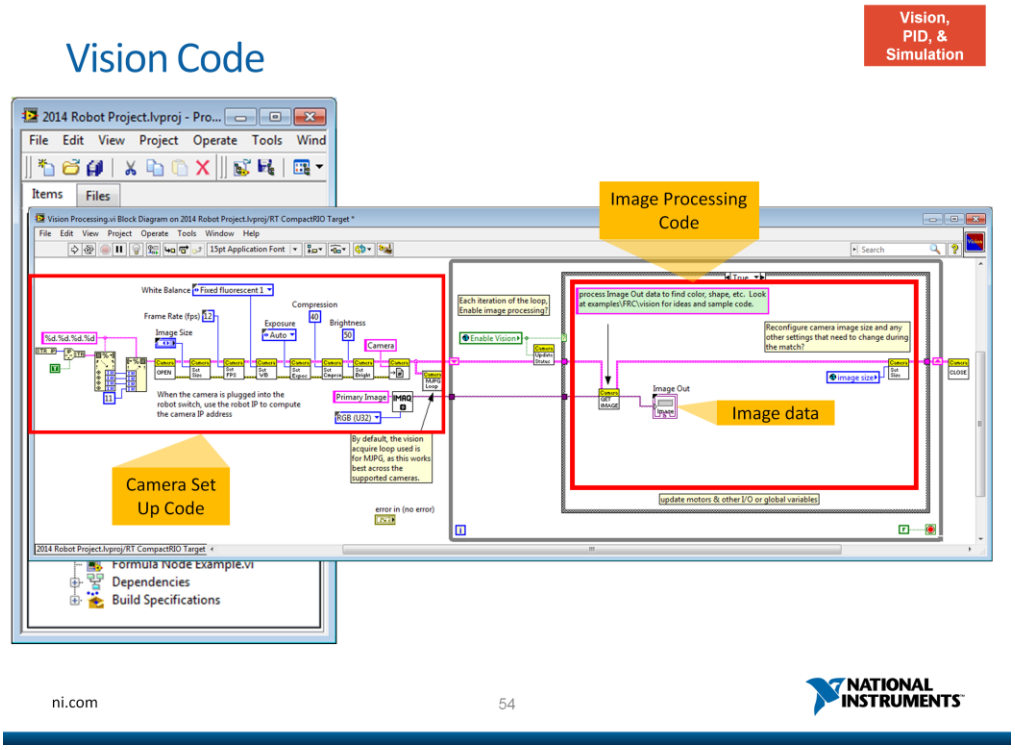
- Locate Colored Object
- Locate Shapes
- Locate game elements
- Calculate distance



Dashboard with Vision Controls

LabVIEW includes a full suite of vision tools that allow you to set up and process images from your camera.

(with two cameras or if object size is known)



To add vision code to our presentation we will need to open up “Vision Processing.vi” from the Team Code Folder

[Animation]

This is the block diagram for the Vision processing

[Animation]

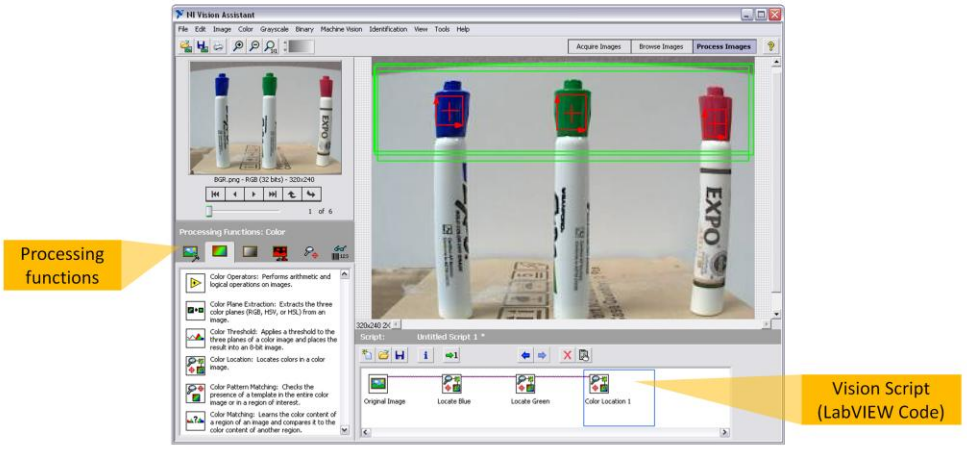
The first part on the left is the camera set up. You can see we set the image size, frame rate, brightness and other settings.

[Animation]

We will need to add some custom vision processing code inside this loop. We can pull the image data from the “Get Image function” then use the data to update motors & other I/O or global variable.

You can create LabVIEW code by browsing through the Vision Palettes and placing functions, but I recommend using the Vision Assitant to create your initial algorithm.

NI Vision Assistant

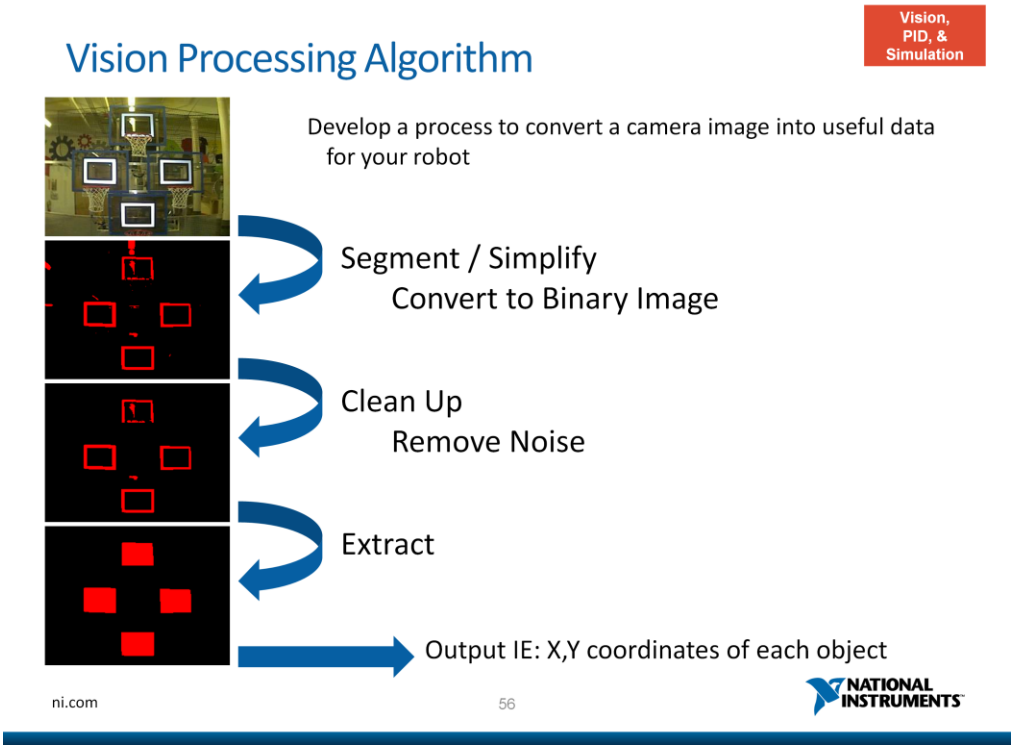


Start>Programs>National Instruments>NI Vision Assistant

Vision Assistant is stand alone application that will allow you to create a vision script which can be converted into LabVIEW code. To open it browse to the start menu path show here or just search for it by name from the start menu.

Vision Assistant allows you to step through each step in your algorithm and tweak the settings along the way. You can use a test image or pull a live image from your camera, then test out your algorithm. Vision Assistant allows you to quickly use “guess and check” to find the right functions, settings, and order to extract the data you need from the picture.

The image shown here shows an vision script that will find the center of the red, green, and blue areas. You can then use that location data to calculate angles, distances, and perspective.



We can simplify most vision processing algorithms into 3 steps. In this example we are using a sample image from the 2012 FRC competition of the basketball goal vision targets.

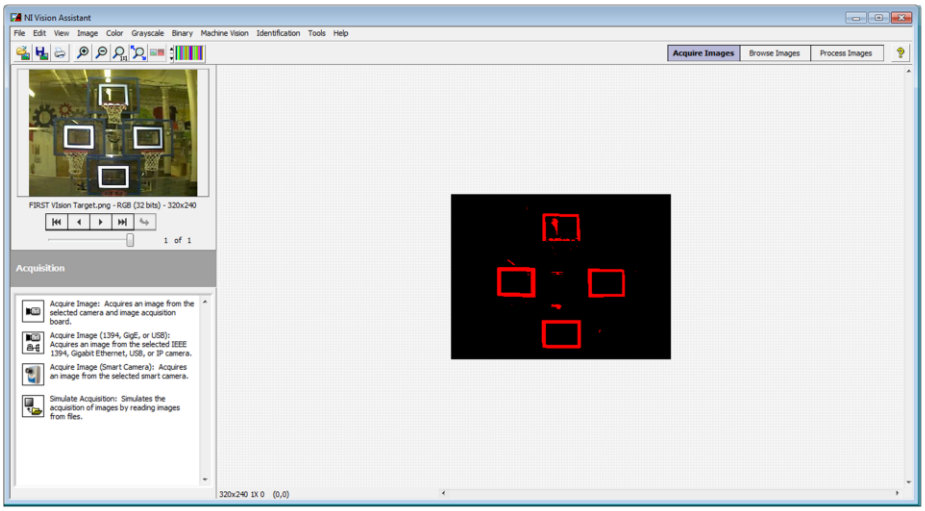
1. First we need to segment or simplify the image so we just have the data we need. Typically you will start by converting your image to grayscale and using a threshold function to convert your image to a binary image (red & black), Most of the processing functions are designed to work with binary images.

2. Second we will need to clean up any stray objects or noise in your image. Once the image is binary we can use functions to remove objects on the edge of the frame, remove small objects, or fill in shapes.

3. Finally we will extract the objects we need from the image so you can output the useful data you need.

Vision, PID, & Simulation

Vision Assistant Demonstration



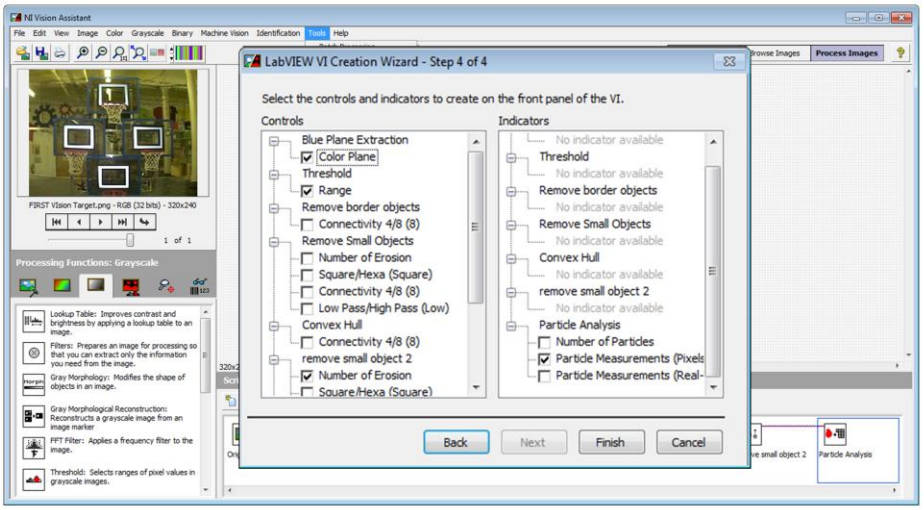
Here is a more detailed look at the example. Our goal is to find the X,Y coordinates for each of the four reflective targets.

[Animation]
First we need to acquire an image. When you first open Vision Assistant it will prompt you to open an image you can either pull an image directly from your AXIS camera or open a saved image from file.
[Animation]
In this case we will just open an image from file and move over to the processing mode, you can always go back to Aquire Image if you want to grab a new image to try.
[Animation]
Once we have the image we will use the Extract Plane Function to pick one of the color planes to use, as a grayscale image. I tried the different colors and found that the Blue Pane had the most contrast compared to other planes.
[Animation]
Next we will use the threshold function to extract the reflective vision targets. You can adjust the slider the find the best value. There is still several extra objects in the image so we will use some other functions to clean it up.
[Animation]
We want to ignore any objects along the edge so just use the remove edge objects mode of the Advanced Morphology function to clean them up
[Animation]
Next we use the remove small objects to get rid of the little splotches and shiny spots.
[Animation]
We now have the rough outline of each box, we will use the convex hull mode of the Advanced Morphology function to
[Animation]
There is still an extra little blob on the screen so we used remove small objects again, now that we have big clear boxes for the main targets. In the left pane you can see the iterations is set to 4 for this step so it will run “remove small objects” four times in this one step. In many cases changing the order of clean up functions or repeating them will change how well they work. So be sure to try different combinations. This type of experimentation is what Vision Assistant is made for.
[Animation]
Finally using the Particle Analysis function we can extract the position data for the four objects on screen. This function can give you the center of mass, edge dimensions, height, width, angle, and much much more for each block. The values are represented in pixels based on the input image size. Once you have the raw numbers you can bring them into LabVIEW and use them to determine various robot parameters. For example you could have the robot turn until a target is in the middle of the camera image.

It helps to have multiple pictures so you can try your algorithm from different positions, different lighting conditions, and other variations to make sure your algorithm is robust and will work reliably.

(You can also open this example “Rebound Rumble Vision Script.vascr” from the Quick Start Module zip file and step through it in Vision Assistant)

Generate Vision Code



Once you have tested your algorithm in Vision Assistant and you are happy with the results, you can export the script into a LabVIEW vi so you can paste it into your team code.

To export a VI select Tools>Create LabVIEW VI…

[Animation]

First select LabVIEW 2013 f2 since that is the version we are using for FRC 2014, and browse to a path to save the generated VI.

[Animation]

You can then select the current script you are working on or select a saved script from file.

[Animation]

Then select the image control option for image source since we are going to paste this code into the existing FRC template rather than acquiring it directly

[Animation]

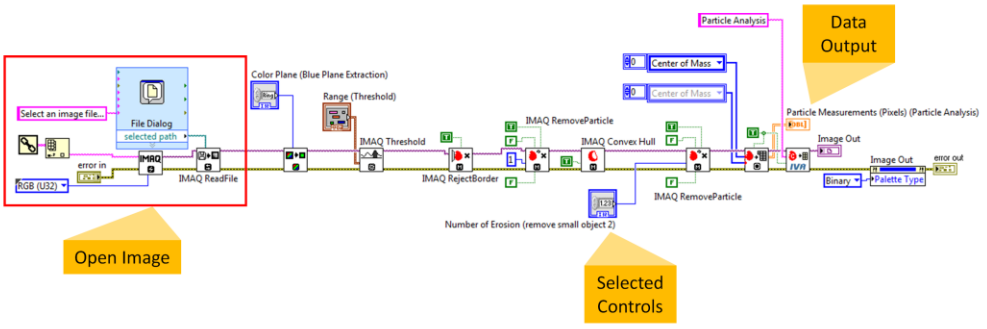
Finally select which values you want to be able to change and monitor with controls and indicators. Click Finish to generate the VI.

Vision, PID, & Simulation

Generated Code

This code was generated from Vision Assistant

- Delete the Open Image code, and paste this into “Vision Processing.vi” in your team code

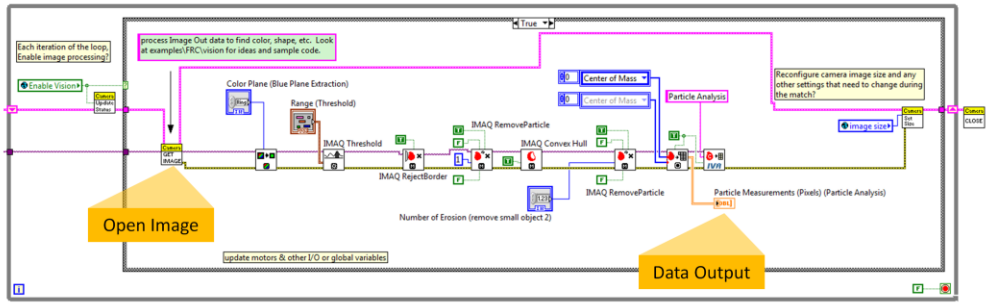


Here is the code we generated from Vision Assistant. The Open Image code on the left opens an image on disc. If you want to keep testing with disc images use it. When you are ready to integrate the code into the FRC project you will not need it, so delete the Open Image code.

Vision, PID, & Simulation

Integrate Vision Code

- Paste generated code in “Vision Processing.vi”
  - Use the Open Image function in Vision Processing.vi
  - Now you can do math and make decision using the Data Output
  - Use Edit>Create SubVI to convert your algorithm to one function to keep your code clean

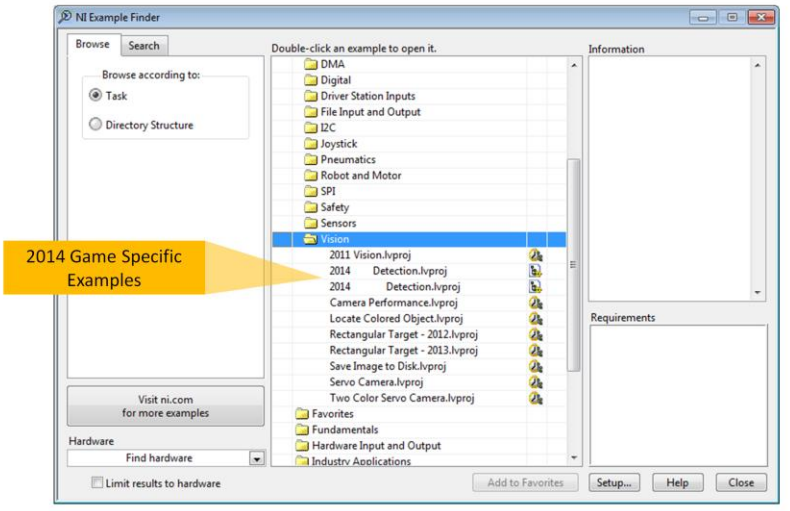


You can now paste your code into “Vision Processing.vi”. You now have the processed Data Output in your robot code, you will still need to add some math or decision making in order to make your robot respond to the images.

[Animation]
I recommend using Edit>Create SubVI to convert this algorithm into a subVI to keep your vision code clean. You can connect the input controls to dashboard variables so you can easily edit the parameters before or during a match.

Vision Examples

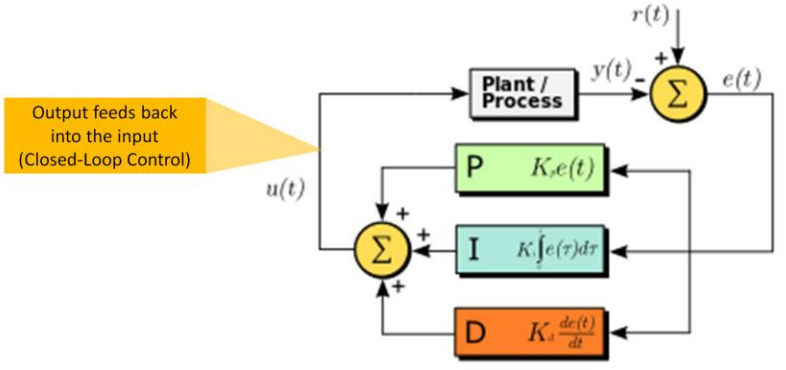
Vision, PID, & Simulation



Be sure to check out the vision examples in example finder there are several specific examples that will give you a great start.
There are two 2014 game specific examples for this season.

PID

- Proportional-Integral-Derivative
- An algorithm used in a control feedback loop to regulate a process such as the motion of a motor or the flow through a valve



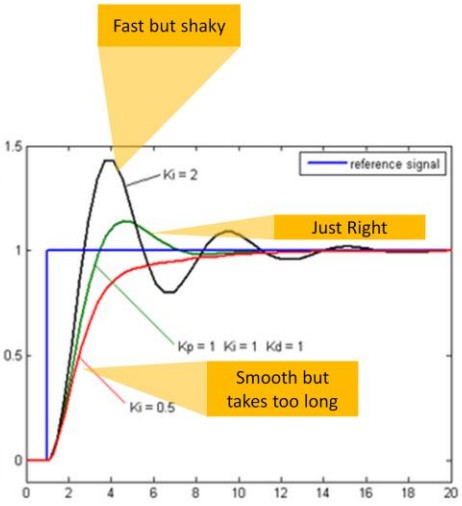
PID control is a fundamental engineering tool that allows you to control dynamic movements smoothly and efficiently.

PID is a common form of “closed-loop feedback” meaning the input is set based on the current output.

In order to use PID you need to be able to monitor the output, for instance a potentiometer or encoder that can detect rotation.

PID in the Real World

Vision, PID, & Simulation



PID is used in many engineering systems in the real world. For instance if you had a giant cannon on a battleship, and you needed it to rotate 90 deg, if you turn the rotation motor on full, then abruptly stop when it reaches the end the cannon would shake violently when you stopped it (red line), if you moved it really smoothly it might take too long (red line). A lot of times it is fastest if you overshoot your target a little and go back (green line). Using PID we can find the best movement profile for your system.

PID allows you to adjust the motor power while you monitor the current position. To change the settings you will have to tweak the P, I and D coefficients to find the best movement for your system.

PID is important when

the moving parts is large and heavy (Like your robot)

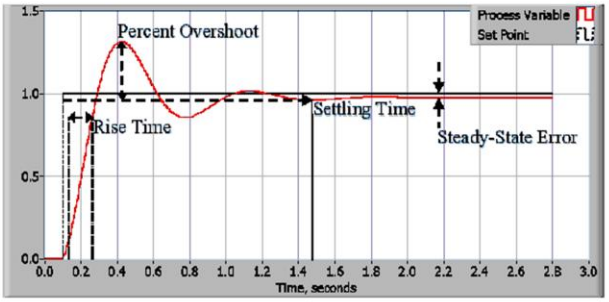
For example if you turn your robot to face a moving target, you don't want it to violently jerk around

You need precision balance

Vision, PID, & Simulation

Tuning a Controller

- Adjust the PID gains to appropriate values for your specific system
- Decrease rise-time and steady-state error

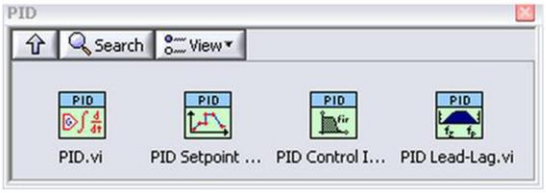


This graph shows movement overtime as the Set Point changes.

Rise Time is the time it takes to reach to set point
Overshoot is how far you go over
Settling Time is how long it takes to get stable
Steady State Error

Using PID in LabVIEW

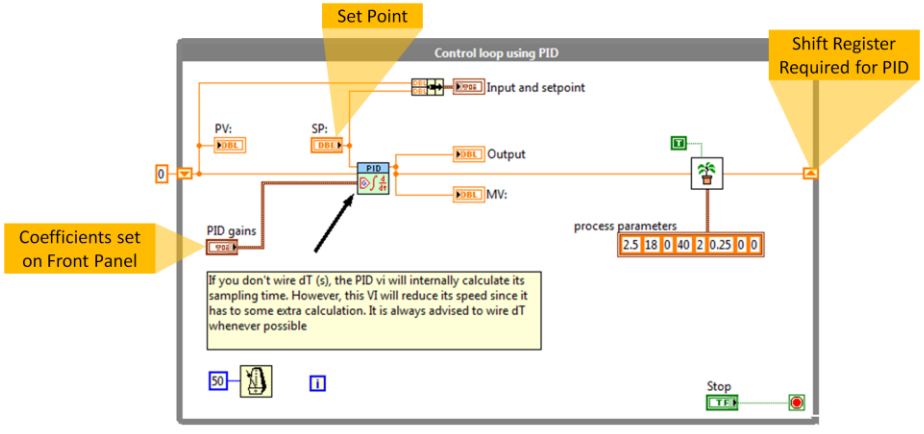
- LabVIEW FRC PID palette consists of four VIs:
  - PID
  - PID Setpoint Profile
  - PID Control Input Filter
  - PID Lead-Lag



The only VI you need to construct a regular PID controller is the PID VI. The other VIs in this palette implement more advanced functions and can be useful for some specific applications.

PID in LabVIEW

Vision, PID, & Simulation



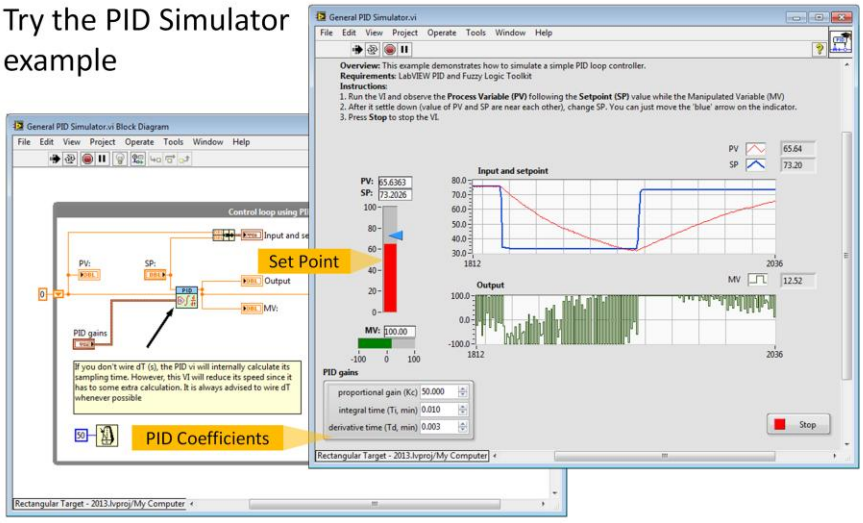
NI Example Finder > General PID Simulator.vi

Here is an example from the NI Example Finder called “General PID Simulator.vi”

Vision, PID, & Simulation

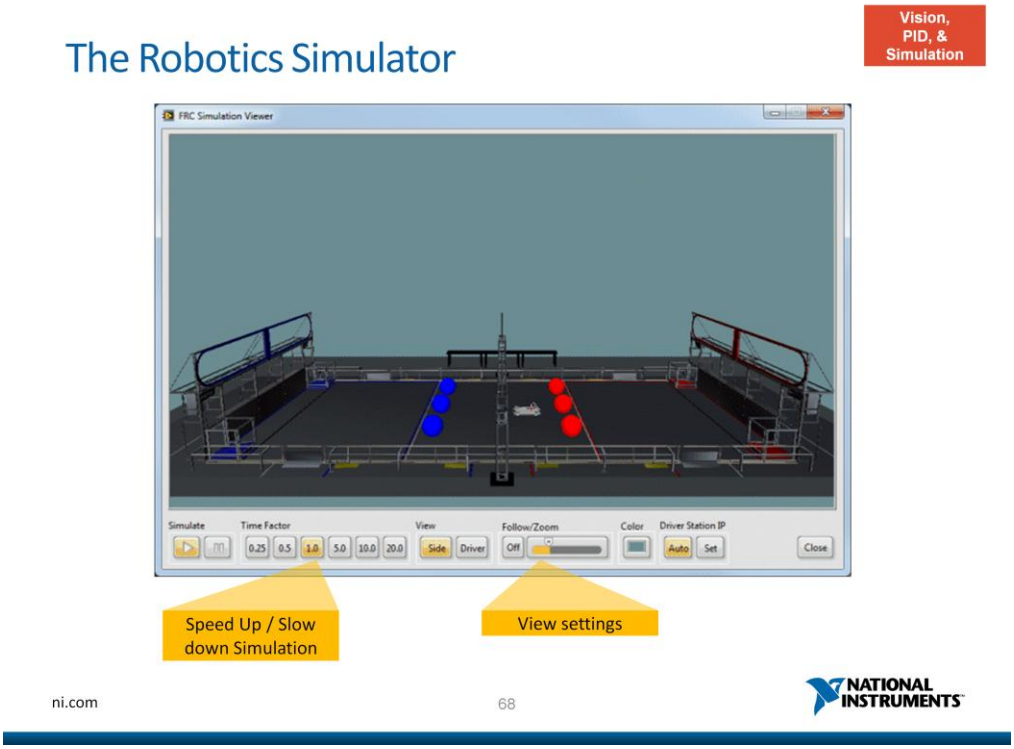
PID Example

- Try the PID Simulator example



NI Example Finder > General PID Simulator.vi

If you are new to the concept of PWM I would recommend playing around with the example PID Simulator. Just search “PID” in the example finder and select “General PID Simulator” test out your ability to tune the P, I, and D coefficients in order to optimize the changes shown on the graph.

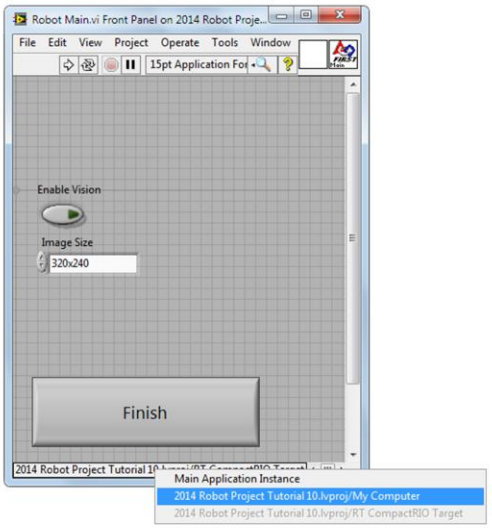


Explain the Options available on the simulator.

1. Simulate – Play/Pause the simulator without stopping code execution
2. Time Factor – For slower computers, increasing the time factor can help in speeding up the responsiveness of the robot drive.
3. View – Driver (view the stage as in competition), Side (view the stage as a spectator)
4. Follow – Sets camera focus on robot (ZOOM)
5. Color – Changes the background color of the simulator
6. Driver Station IP – Use a static IP address for the simulated robot. For most applications you can leave it as Auto

Deploy Code to Simulator

- Start the Driver Station - it is needed for real and simulated robots.
- Right-click in the lower left corner of Robot Main.vi and choose **2014 Robot Project.lvproj/My Computer**
- Run Robot Main.vi and notice that the FRC Simulation Viewer opens
- Enable Teleop mode in order to drive the simulated robot or enable Autonomous for independent control.



1. Start the Driver Station - it is needed for real and simulated robots.
Open Robot Main (under the cRIO context)

2. Right-click in the lower left corner of Robot Main.vi and choose **2014 Robot Project.lvproj/My Computer**

3. Wait a moment while some subVIs reload

4. Run Robot Main.vi and notice that the FRC Simulation Viewer opens
The Driver Station should show you are in Simulated Mode, and you will need to Enable Teleop mode in order to drive the simulated robot.

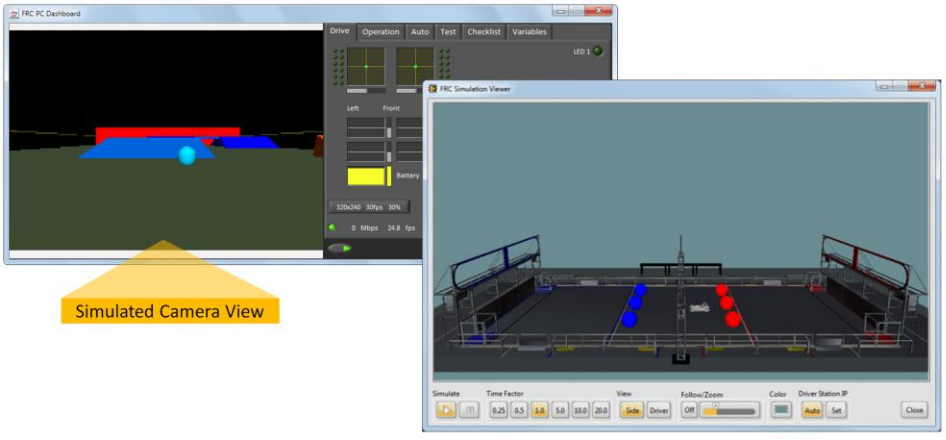
5. To run Robot Main.vi on the cRIO, right-click in the lower left corner of Robot Main.vi and choose **2013 Robot Project.lvproj/RT CompactRIO Target**

To switch back to the cRIO, simply right-click in the lower left corner of Robot Main.vi again and choose 2013 Robot Project.lvproj/RT CompactRIO Target

Vision, PID, & Simulation

Exercise-Deploy FRC cRIO Robot Project

- Deploy the default robot project to the simulator
  - The Driver Station must be running



ni.com 70 NATIONAL INSTRUMENTS

Exercise – 15 Minutes

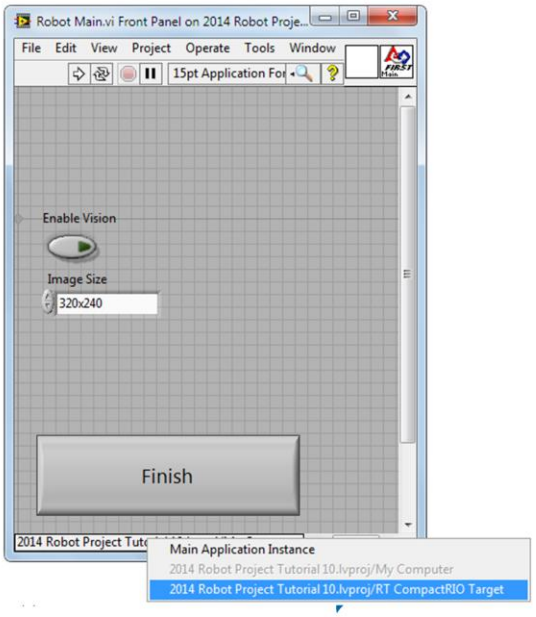
1. Start the Driver Station - it is needed for real and simulated robots.
2. Open your robot Project and Open Robot Main
3. Right-click in the lower left corner of Robot Main.vi and choose **2014 Robot Project.lvproj/My Computer**
4. Wait a moment while some subVIs reload
5. Run Robot Main.vi and notice that the FRC Simulation Viewer opens

The Driver Station should show you are in Simulated Mode, and you will need to **Enable Teleop mode** in order to drive the simulated robot.

Vision, PID, & Simulation

Deploy to Real FRC Robot

- Right-click in the lower left corner of Robot Main.vi and choose **2014 Robot Project.lvproj/RT CompactRIO Target**



ni.com

Now that you have developed your code and tested it with the simulator, deploy to a real FRC robot!

Vision, PID, & Simulation Resources

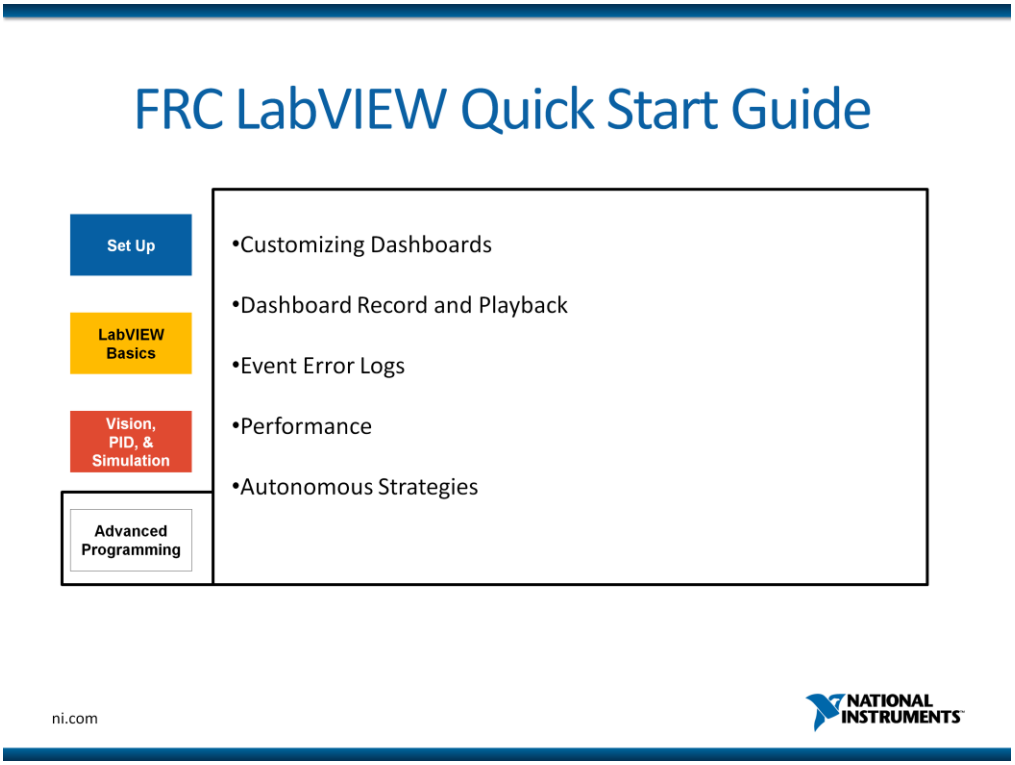
Vision, PID, & Simulation

- Configure an AXIS Camera
- Image Processing Tutorial
- Using Vision Targets
- List of NI Vision Functions
- LabVIEW PID Tutorial
- PID Theory Explained
- Robot Simulator Tutorial

All links available at NI.com/FRC
Click this button

- Forum Support
  - NI.com/FRC – year round
- Phone Support
  - 1(866)511-6285 from 1pm to 7pm (CST) - During build season only

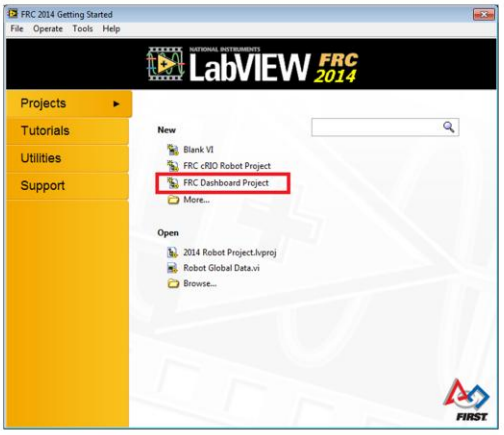
Configure an AXIS Camera
http://wpilib.screenstepslive.com/s/3120/m/8559/l/89729-configuring-an-axis-camera
Image Processing Tutorial
https://decibel.ni.com/content/docs/DOC-26318
Using Vision Targets
https://decibel.ni.com/content/docs/DOC-20173
List of NI Vision Functions
http://zone.ni.com/reference/en-XX/help/370281M-01/nivisionlvbasics/ni_vision_function_palettes/
LabVIEW PID Tutorial
https://decibel.ni.com/content/docs/DOC-26317
PID Theory Explained
http://www.ni.com/white-paper/3782/en/
Robot Simulator Tutorial
https://decibel.ni.com/content/docs/DOC-26300



This module covers some of the advanced LabVIEW Programming concepts.

Customizing Dashboard

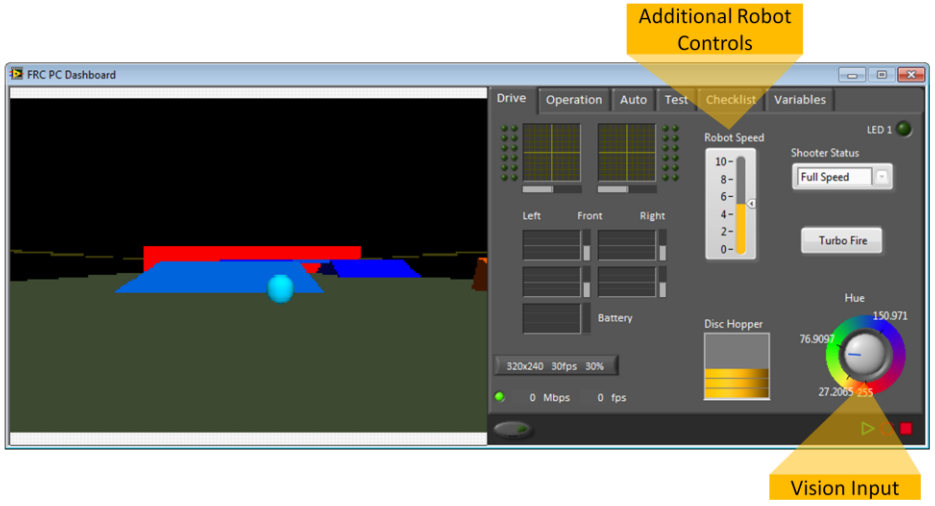
- Start by opening an FRC Dashboard Project



If you want to display some of your custom data or add custom controls you can customize your dashboard so you can see exactly what you need to complete, and nothing you don't need. To start go to the Getting Started Window and create a New FRC Dashboard Project.

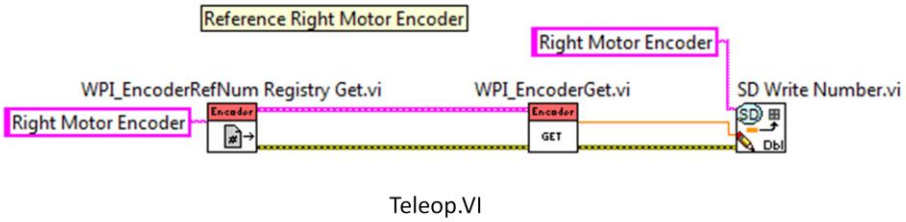
Customizing Dashboards

Advanced Programming



Open the Dashboard main and add any controls or indicators you want. You can place them on any of the tabs to help you stay organized. Be sure to to add too much or your interface will be too cluttered at game time. In this case I added a color selector so I can pick the color my camera is looking for, a slider to adjust Max robot speed, an indicator showing me how many discs I have in the hopper, and a Turbo Fire button. Once you have placed these controls and indicators on the front panel you will need to add some code in your cRIO robot project to make them work.

Connecting Custom Variables to the Dashboard

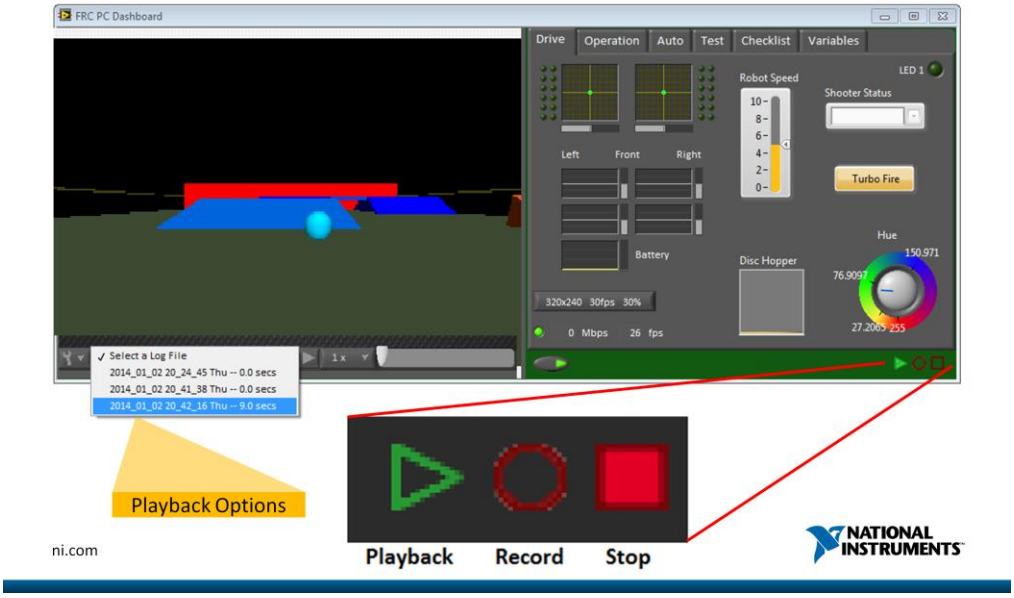


Teleop.VI

Here is an example of the modified code in the cRIO Robot Project. In order to display the encoder values during teleop we just add this code to the teleop VI and use the (Smart Dashboard) SD Write Number.vi to send the data to the dashboard by name. Be sure the names are exactly the same or it might not work.

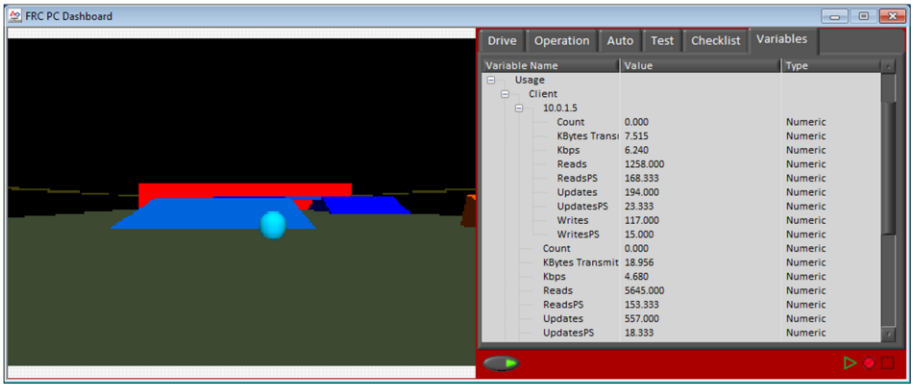
You can also modify Begin.vi, Finish.vi or any other VI in the team code to make the data accessible via the dashboard

Dashboard Record & Playback



New in FRC 2014, the Dashboard can now easily Record drive sessions which can then be played back. This new feature helps the verification and testing phase of the robot. In addition, the record option makes it easier for users to spot any irregularities of input or output data from the robot.

Record

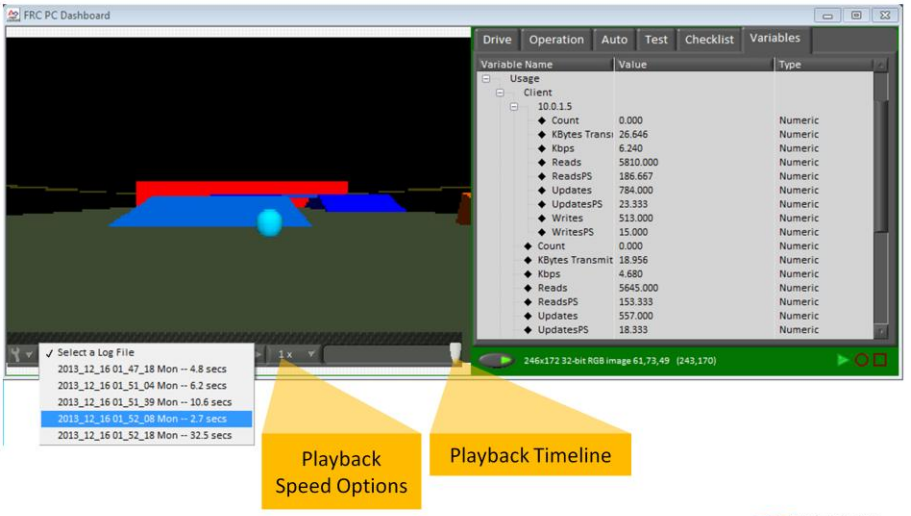


Log Location: C:\Users\Public\Documents\FRC\Log Files\Dashboard

To Record, simply click on the Red Circle to begin recording of your current drive session. The recording will be stored in the following location:

C:\Users\Public\Documents\FRC\Log Files\Dashboard

Playback

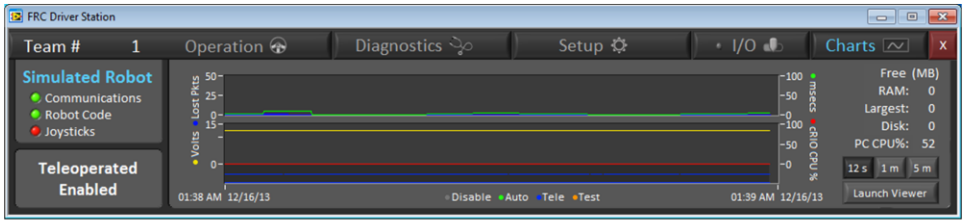


To playback your recordings, simply click on the Green Play Triangle and select the log file you want to play back.

You can change the speed at which the playback happens. You can speed up the playback to skip over unimportant parts and slow the playback down to monitor points of interest.

Troubleshooting: Driver Station Charts

- Network Issues
  - Lost data packets
  - Data packet trip time
- Programming Issue
  - CPU Usage
  - Free RAM on cRIO
- Hardware Issue
  - Robot battery voltage

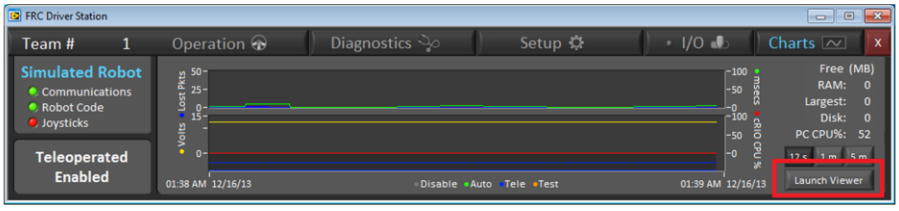


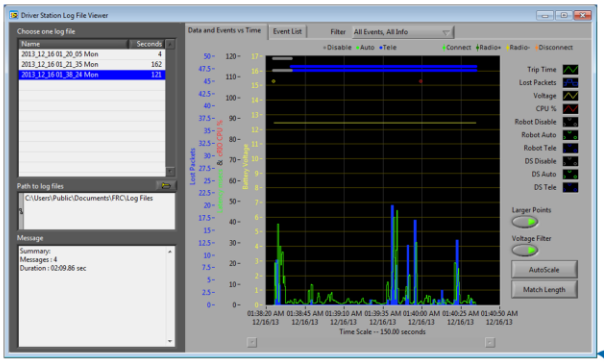
If you keep loosing connection with your robot or if you are having communication and control issues use the charts view on the Driver Station for useful info. If you are having network issues you will see a lot of lost data packets or a long trip time.

You can also use the FRC Driver Station Charts tab to view performance statistics. The cRIO is a Real-Time controller which means if it get overloaded it starts dropping low priority tasks in order to keep the high priority tasks on time. Network communication is a low priority task so it may be the first thing to stop working if the cRIO CPU is overloaded. Ideally CPU usage should be below 80%. Two common things that cause CPU overload include:

1. Loops without delays in them
2. Loops with code that takes longer to run than the defined loop delay

Troubleshooting: Event Error Logs





After driving sessions, the Driver Station logs the low-level data of the Robot, Network, and any if any error occurred. These logs are extremely helpful in identifying errors that you may not be capturing in your Robot code.

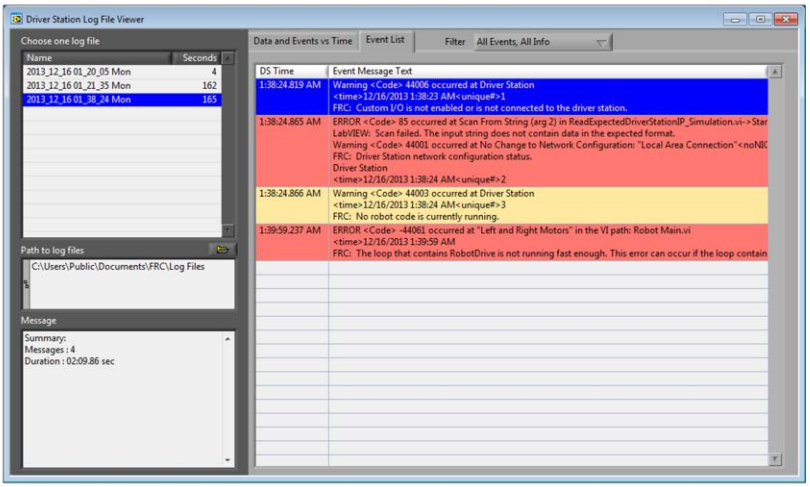
To view these logs, simply click on the “Launch Viewer” button located on the Charts tab of the Driver Station.

Once in the Driver Station Log File Viewer, you can view logs at the left hand side of the screen. Notice there are two tabs on the top.

Data and Events vs Time tab – Graphical representation that provides a “big picture” view of what happened when the robot was driving.

(continued…)

Troubleshooting: Event Error Logs



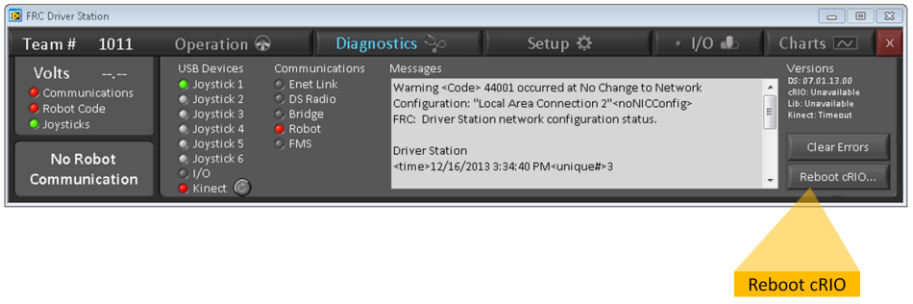
Log Location: C:\Users\Public\Documents\FRC\Log Files

Event List tab – List representation of messages that provide more details on the events.

The Events List tab is particularly helpful in identifying Watchdog or Safety Config timeouts. In addition, it can provide valuable information in-between matches when the robot fails to respond.

Troubleshooting: Driver Station Diagnostics

- The Diagnostics tab in the FRC Driver Station can display important warning and error messages for troubleshooting

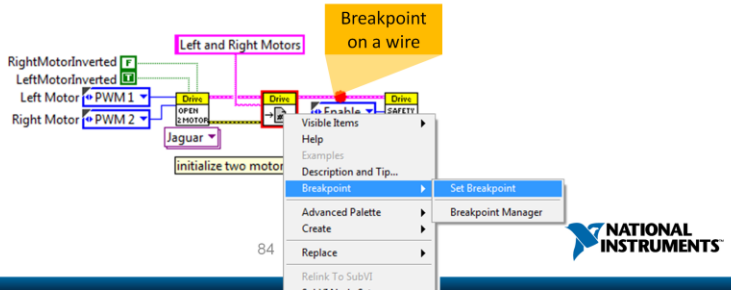


If you are not able to run your program the Driver station diagnostics tab has useful information that will show which components are not working as expected. In this case we can see the joystick is connected but we have lost connection with the robot, and since we can't communicate with the robot we can't check if the Robot Code is present.

You can also reboot your cRIO from here if you are connected.

Debugging: Breakpoints

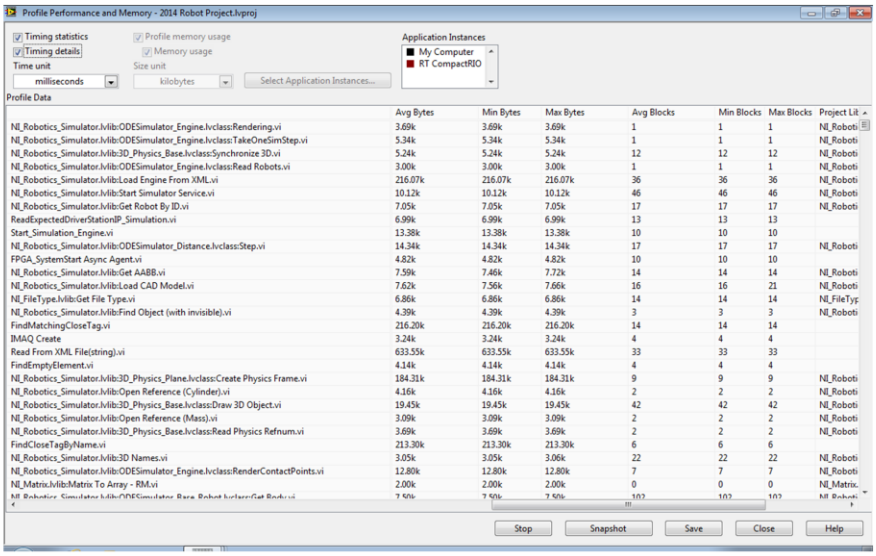
- When you reach a breakpoint during execution, the VI pauses and the **Pause** button appears red.
  - (right-click Breakpoint>Set Breakpoint)
- You can take the following actions at a breakpoint:
  - Single-step through execution using the single-stepping buttons.
  - Probe wires to check intermediate values.
  - Change values of front panel controls.
  - Click the **Pause** button to continue running to the next breakpoint or until the VI finishes running.



Breakpoint can be useful when you are trying to track down a bug in your code.

Note: breakpoints will pause your code and set off watchdogs if you are running code on your cRIO.

Performance



Tools>Profile>Performance and Memory

Tools>>Profile>>Performance and Memory

If we just want the speed of a single VI, this may be an easier solution than editing Elapsed Times.vi. Notice that here the average run time for Build Dashboard Data.vi is basically the same value we found in the previous slide.

Timing statistics—Displays the following statistics for the VI run time:

 # Runs—Number of times that the VI completed a run. For global VIs, this time is the total number of times any of its controls were accessed.

Average—Average amount of time spent by the VI per run. This is the VI time divided by the number of runs.

Shortest—Minimum amount of time the VI spent in a run.

Longest—Maximum amount of time the VI spent in a run.

Memory usage—Displays statistics about the number of bytes and the number of independent memory blocks that a VI uses.**Avg Bytes**—Average number of bytes used by the data space of the VI per run.

Min Bytes—Minimum number of bytes used by the data space of the VI for an individual run.

Max Bytes—Maximum number of bytes used by the data space of the VI for an individual run.

Avg Blocks—Average number of blocks used by the data space of the VI per run.

Min Blocks—Minimum number of blocks used by the data space of the VI for an individual run.

Max Blocks—Maximum number of blocks used by the data space of the VI for an individual run.

Watchdog

Advanced Programming

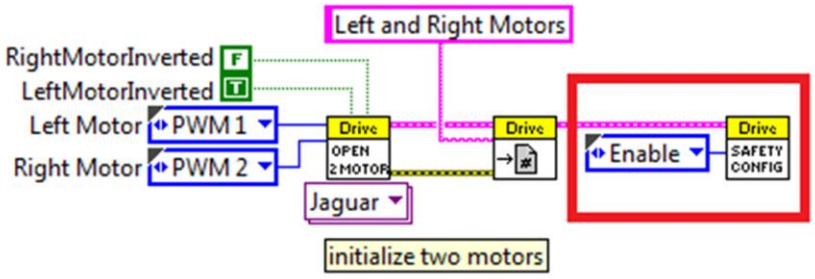
- System Watchdog
  - Monitors communication from Driver Station
  - Turns off outputs if no robot communication in 100ms
  - Causes
    1. No communication – radio issues?
    2. Slow Communication – CPU overload?
    3. Driver Station running too slowly – Dashboard hogging CPU

Some type of watchdog is a needed safety mechanism.

The cause of a System Watchdog is usually a communication issue.

Safety Config Timeouts

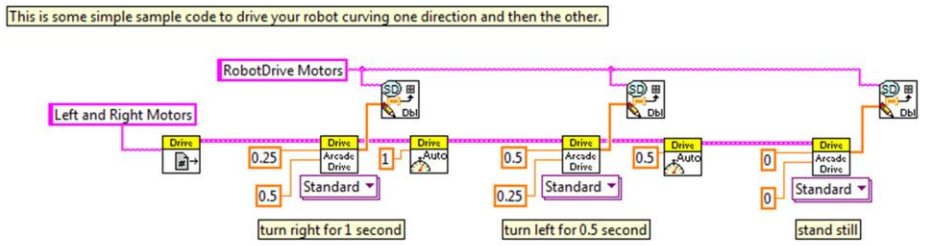
- You can turn on/off safety timeouts
- Available for RobotDrive, MotorControl, PWM, Relay, and Solenoid



The Safety Config VIs were added as a suggested User Watchdog alternative. They are output specific, meaning they can be turned on or off (Enabled or Disabled) for individual outputs and only enabled outputs that don't update in time get affected.

Autonomous Strategies

- Easiest Autonomous – Tell the robot to do the same thing every time!



One of the easiest autonomous strategies for a team is to pre-program the robot's movements.

Using the cRIO Robot Template, you can follow the example above to pre-program your robot to always move in specific directions.

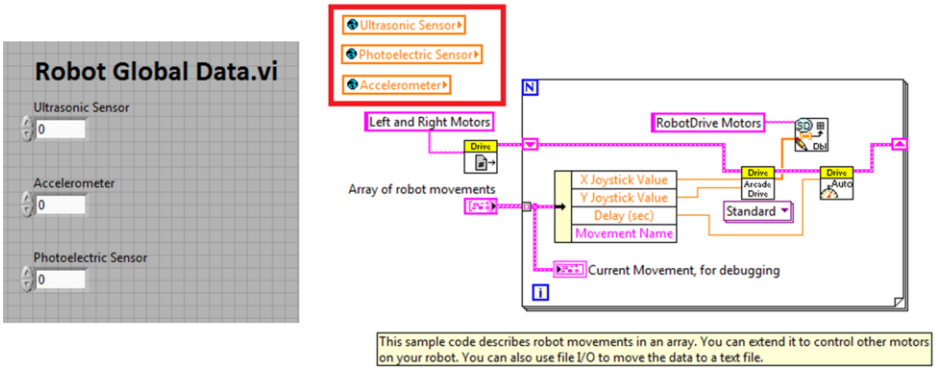
Sure beats not doing nothing!

Although this is easy to implement, pre-programming your robot makes your robot inadaptable to any changes on the field or interference from other robots.

(continued…)

Autonomous Strategies

- Closed-loop Feedback



A better way to implement your autonomous code is to provide your robot with sensor data and allow your robot to make decisions based on the sensor data.

For example, programmers can create global variables that are updated with sensor data so that the autonomous code can make smart decisions.

An ultrasonic sensor can let the robot know that a wall, obstacle, or another robot is nearby.

A Photoelectric Sensor helps the robot detect and follow a line.

Advanced Programming

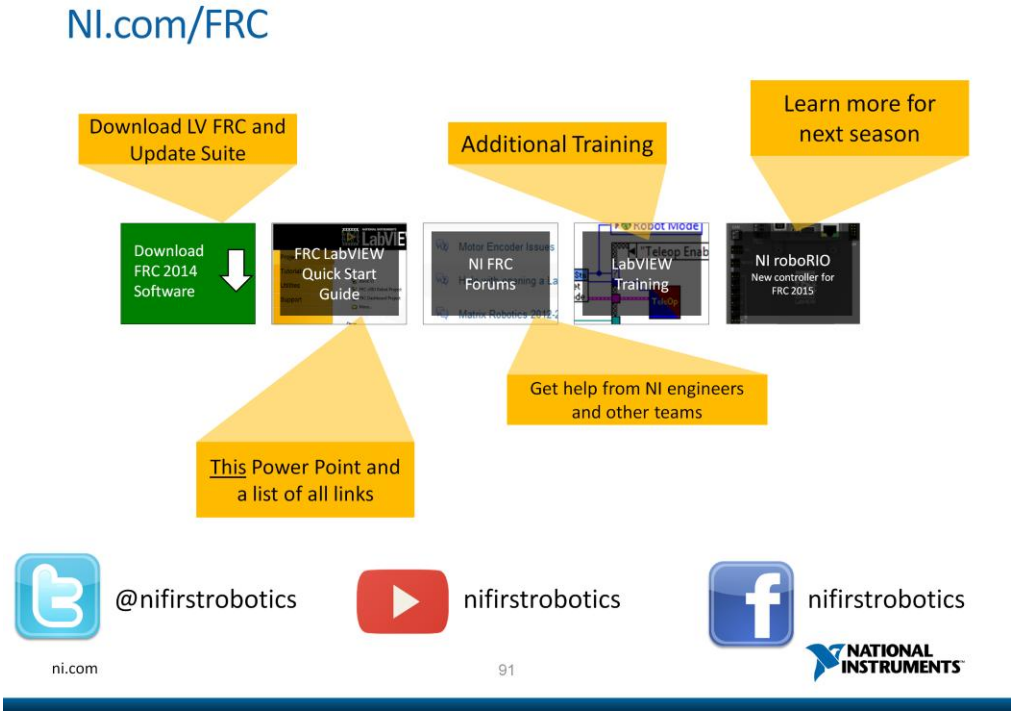
Advanced Programming Resources

- Full FRC LabVIEW Training (Beginner & Advanced) 4.5 hrs each
- Programming for Performance
- Sensor Fusion Tutorial
- Autonomous Timed Movement

All links available at NI.com/FRC
Click this button

- Forum Support
  - NI.com/FRC – year round
- Phone Support
  - 1(866)511-6285 from 1pm to 7pm (CST) - During build season only

Full FRC LabVIEW Training (Beginner & Advanced)
https://decibel.ni.com/content/docs/DOC-33629



Be sure to visit NI.com/FRC for downloads, training, support, and information.