LabVIEW Basics

TEAM 4150

FIRST ROBOTICS COMPETITION TUTORIALS

FROBOTICS TEAM 4150

Overview

It’s incredibly hard to completely teach any sort of programming simply through a document or slideshow, so instead the goal of this presentation is to get you inundated with the basics of LabVIEW and at the end to point you in the proper direction for you to continue on and gather a deeper understanding of LabVIEW and programming in general.

This slideshow will cover both the basics for programming a robot in LabVIEW and creating a driver station for your drive team through LabVIEW, along with explaining what things to consider when doing so and the installation of LabVIEW.

If you don’t read the entire presentation, at least read slides 7 and 8 for some LabView tips. Slide 28 is a must-read.

General Information

Installing LabVIEW

- The LabVIEW install media USB doesn't come in the kit of parts, you can however request it as part of your first choice or simply download it from the website.

1. LabVIEW requires a serial number when you install it, which is given to you in your virtual kit of parts (only the head mentors have access to it).
2. Make sure to uninstall all previous FRC software
3. Reboot computer before installing
4. Make sure you install LabVIEW before you install anything else (other languages or other FRC software)
5. Reboot computer after installing

- If you run into any issues or other questions this website has pretty in depth instructions on the installation process.

Using VIs

Virtual Instrumentation or VI is basically the various parts of your program. VIs are basically subroutine equivalents in LabVIEW. When you first open a new project for robot code, it's already preset with 10 VIs for you. Same goes for the dashboard, when you open it, it starts you off with a main dashboard VI, but you could always create more if you find it necessary or you could create sub-VIs for a VI.

A reason you might want to create sub-VIs would be to clean up your code a little more. For example if there's some large complicated equation, you could turn it into a subVI, so it's easier to look at your code as a whole and you can look at that chunk by itself without any other blocks or wires in the way. They can also be reused in other places in the code.

Creating Sub-VIs

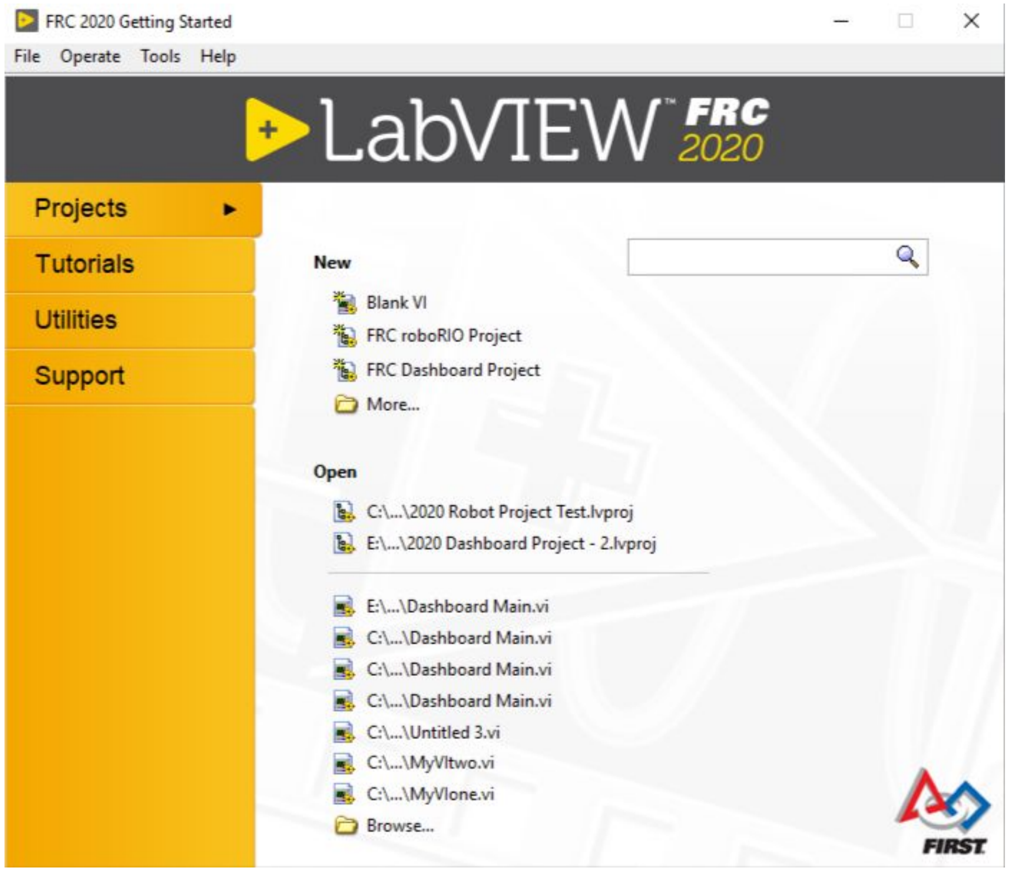
Creating a subVI in one of your programs is pretty simple. Once you have the block code written out that you want to turn into a subVI, highlight it and click on Edit -> Create SubVI and it will replace that section of code with a singular block. There’s a LabVIEW YouTube channel which has a video that shows this process and it contains parts that are helpful both for when creating sub-VIs and VIs. Also here’s some written out step by step instructions.

Creating a New Project

Now that you have LabVIEW downloaded, you can start a new project. When you open LabVIEW it'll look something like the image to the right.

There are three options under "New": VIs which were discussed earlier, and two different kinds of projects, projects are a collection of VIs

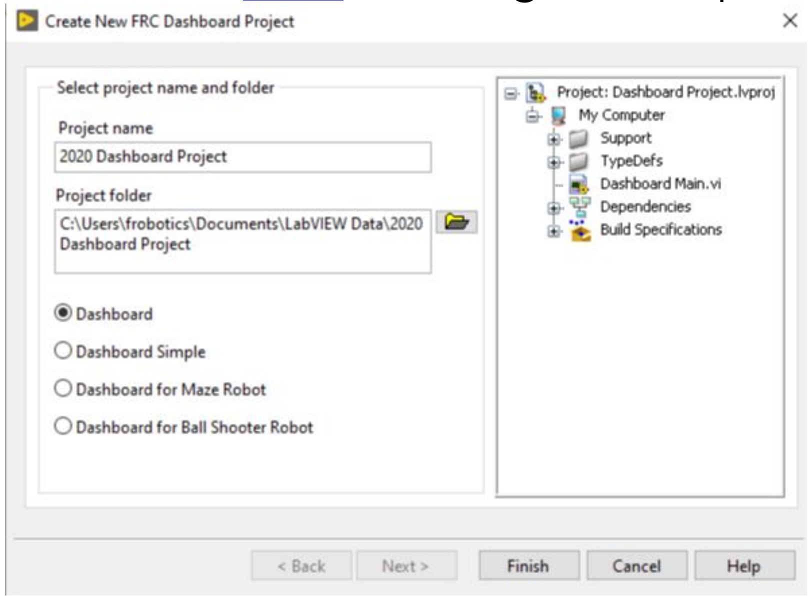
To create a new VI, click "Blank VI", new robot project, click "FRC roboRIO Project", new dashboard, "FRC Dashboard Project".



Creating A New Project (Continued)

When creating a new robot project or new dashboard, it'll prompt you to name it and give it a place in the directory, it's generally a good idea to let it go into the LabVIEW Data folder, which is what the default location should be when you open it. If you change it, take note of it, as all new VIs you create also need to go there

There are a couple kinds of dashboards, I won't go into detail on them, I do suggest just opening them up yourself and exploring them a little though. There's also a couple kinds of robot projects. Arcade drive is the basic original one and then a few years ago they added command & control, which you can read more about here. The original template is the one that's most often used.


Create New FRC Dashboard Project

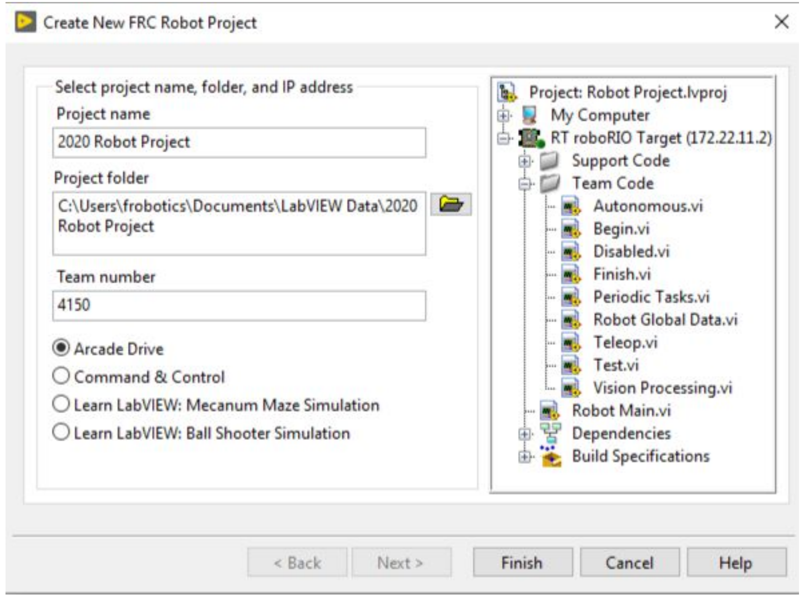
Select project name and folder

Project name: 2020 Dashboard Project

Project folder: C:\Users\frobotics\Documents\LabVIEW Data\2020 Dashboard Project

[✓ Dashboard]
Dashboard Simple
Dashboard for Maze Robot
Dashboard for Ball Shooter Robot

Project: Dashboard Project.lvproj
- My Computer
  - Support
  - TypeDefs
  - Dashboard Main.vi
  - Dependencies
  - Build Specifications


Create New FRC Robot Project

Select project name, folder, and IP address

Project name: 2020 Robot Project

Project folder: C:\Users\frobotics\Documents\LabVIEW Data\2020 Robot Project

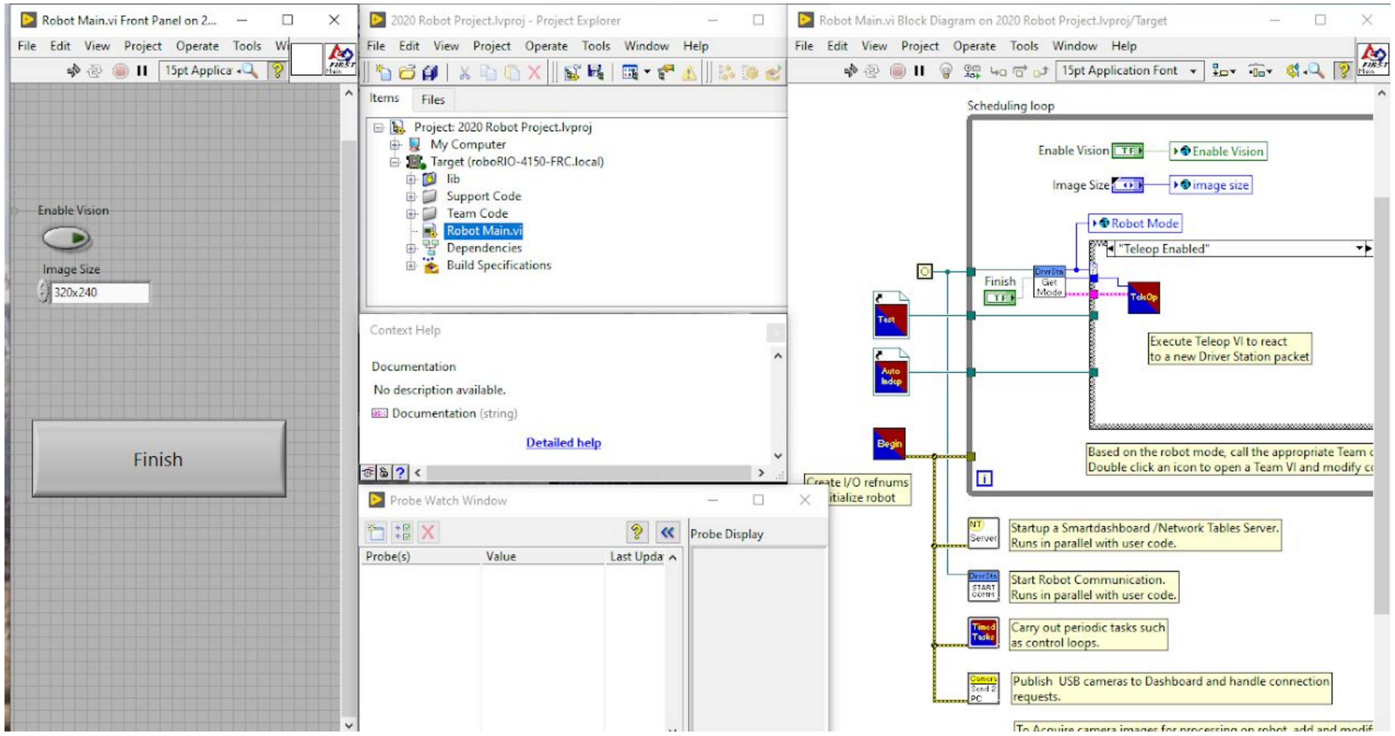
Team number: 4150

[✓ Arcade Drive]
Command & Control
Learn LabVIEW: Mecanum Maze Simulation
Learn LabVIEW: Ball Shooter Simulation

Project: Robot Project.lvproj
- My Computer
- RT roboRIO Target (172.22.11.2)
  - Support Code
  - Team Code
    - Autonomous.vi
    - Begin.vi
    - Disabled.vi
    - Finish.vi
    - Periodic Tasks.vi
    - Robot Global Data.vi
    - Teleop.vi
    - Test.vi
    - Vision Processing.vi
  - Robot Main.vi
  - Dependencies
  - Build Specifications

Inside the VI

These are all the main windows inside a VI, these are things you'll see after creating your project. On the left is the front panel, in the middle column is the project view, the context help, and the probe watch window, and on the right is the block diagram. Probes are actually a useful way of troubleshooting, if you right click on a wire and click "Probe" from the pop-up menu and run the program, then it will tell you whatever data is passing through it.



Documentation

No description available.

Documentation (string)

Execute Teleop VI to react to a new Driver Station packet

Based on the robot mode, call the appropriate Team c
Double click an icon to open a Team VI and modify c

Create I/O refnums
itialize robot

Startup a Smartdashboard /Network Tables Server. Runs in parallel with user code.

Start Robot Communication. Runs in parallel with user code.

Carry out periodic tasks such as control loops.

Publish USB cameras to Dashboard and handle connection requests.

To Acquire camera images for processing on robot, add and modif

Using LabVIEW VI

Once you create a new project, there are a few things that you need to know about using LabVIEW within the VI.

1. There are always two main windows that you'll be using, the front panel and the block diagram.
   a. When doing robot code, you'll probably end up using the block diagram panel more often, as it's where the actual code is.
   b. When doing a dashboard project you'll use a lot of both because the dashboard requires the visuals on the front panel to be nice for the drive team.
   c. You can switch between these panels by either ctrl + e or by going to window -> block diagram or window -> front panel (or you could just leave both panels open ;))
2. To see the the palette of available controls/indicators, right click anywhere on the front panel. To see the function palette right click anywhere on the block diagram
3. There's a search bar on both panels, this can be very important as some blocks can be incredibly hard to get to in the panels.

Using LabVIEW VI (Continued)

4. If you want to see the code connected to a block or the block that's connected to the code, just double click on it and it will highlight whatever you were looking for on the other panel.

   a. For example if I had a button on the front panel of the dashboard and wanted to see the code connected to it, I could double click it and it will take me to the code connected to the button in the block diagram panel.
   b. If you double click on something that isn't a control, but has a VI attached, it'll open the VI, otherwise it'll do nothing

5. Things such as buttons and lights can be annoying to click on to resize/move or double click to look at the code.

   a. If something's giving you a hassle you just have to be patient and try clicking as close to the outside of the object as you can. Computers don't like to respond to angry people.
   b. There's also a tool palette that allows you to choose a specific operation that you want to do tool palette

6. If you ever want to know what a specific block does or what the inputs are press ctrl + H and move your mouse on top of the block and it will give you a diagram of the block and what it does.

Clusters

Clusters combine separate pieces of data into one entity and can be used for a couple different things.

One is when using blocks that require more than one input that goes to the same input spot. For example if you want to make a line on the camera view, the line requires a start point and an end point, which both require an x and y input (example shown in “Dashboard: Camera Lines” slide).

The other one is when you have a group of things in a VI that are related to each other. For example if you had camera settings, you could cluster them into one thing, or diagnostic displays for the robot.

Here's a basic example of clusters in LabVIEW.

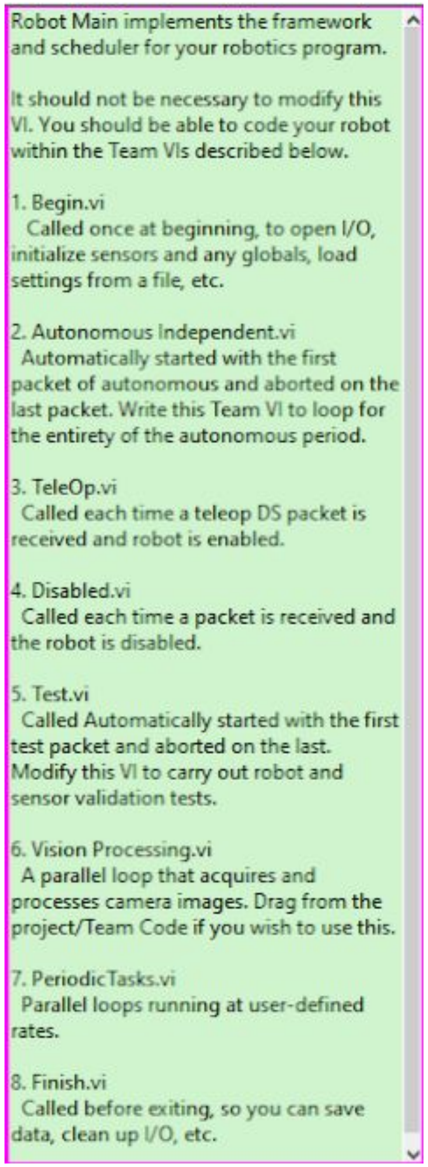
Robot Code

Robot Code: Important Terms

Main, Begin, Auto, and Teleop are all things that you should know what they are. All of which happen to be explained in the Robot Main when you create a new robot program, which I put a screenshot of for you to see on the right (DS means driver station).

Network tables are also important to know about. They are good for debugging and they are the values that are used when creating a dashboard. When you write network table variables they are published to the dashboard. I'll explain the use of them in a little more detail later.

A quick tip on network table variables, if you put them into a separate loop in Periodic Tasks, this will clean up the code and will get rid of some possible network table issues.



Robot Main implements the framework and scheduler for your robotics program.

It should not be necessary to modify this VI. You should be able to code your robot within the Team VIs described below.

1. Begin.vi
 Called once at beginning, to open I/O, initialize sensors and any globals, load settings from a file, etc.

2. Autonomous Independent.vi
 Automatically started with the first packet of autonomous and aborted on the last packet. Write this Team VI to loop for the entirety of the autonomous period.

3. TeleOp.vi
 Called each time a teleop DS packet is received and robot is enabled.

4. Disabled.vi
 Called each time a packet is received and the robot is disabled.

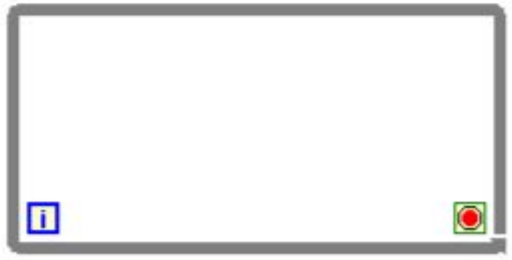
5. Test.vi
 Called Automatically started with the first test packet and aborted on the last. Modify this VI to carry out robot and sensor validation tests.

6. Vision Processing.vi
 A parallel loop that acquires and processes camera images. Drag from the project/Team Code if you wish to use this.

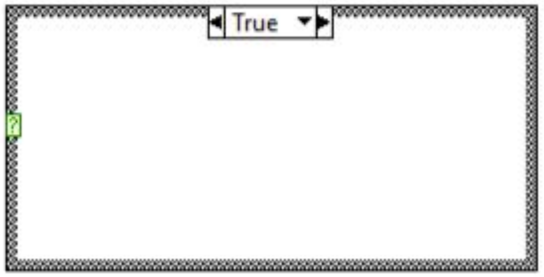
7. PeriodicTasks.vi
 Parallel loops running at user-defined rates.

8. Finish.vi
 Called before exiting, so you can save data, clean up I/O, etc.

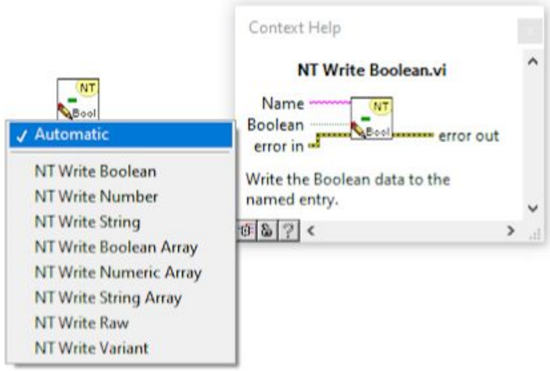
Robot Code: Important Blocks



While loops repeat whatever is inside them until a certain case is met and they always run at least once. Make sure to add a wait function and wire a constant, so that it doesn't continually run



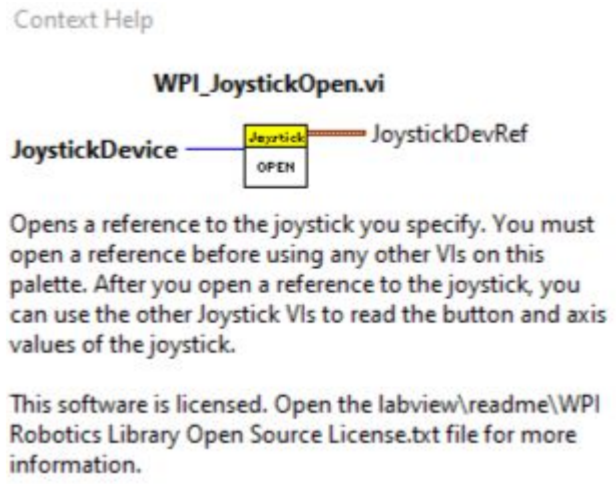
Case structures have boolean inputs, if the input is true, then it runs the code in the true case, if the code is false then it runs the code in the false case. There are also case structures that you can use with numbers or strings, which work similar to the true false, just you can have more variables to them.



Network table writes/reads are important for network tables (I'm sure you figured that much out). This one is NT Write, I'll explain the read later. NT writes require a name and whatever data you're inputting into the NT. The name must be structured as /someName and if you want to create a sub-category you can write /name/myName.

Robot Code: Important Blocks (Cont.)

For these blocks I've put here, I figured that the context help explained it better than I could, so I've just used that. Something to note here too, is that while these examples are for the joysticks, these blocks can also apply to the motors, sensors, and solenoids. A good example of them in use is on FRC Team 358's website.

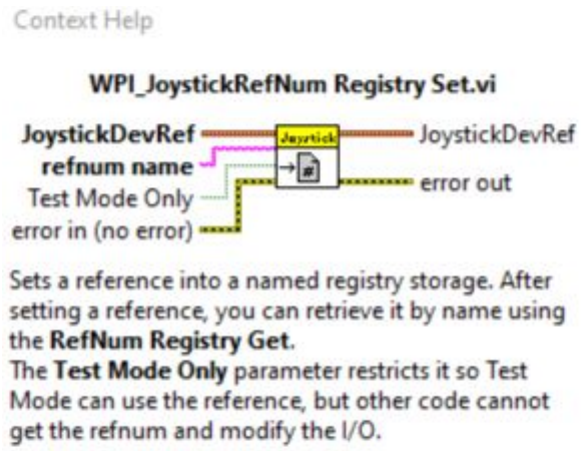


Context Help

WPI_JoystickOpen.vi

Opens a reference to the joystick you specify. You must open a reference before using any other VIs on this palette. After you open a reference to the joystick, you can use the other Joystick VIs to read the button and axis values of the joystick.

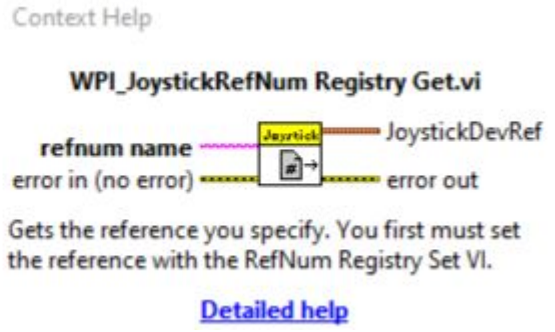
This software is licensed. Open the labview\readme\WPI Robotics Library Open Source License.txt file for more information.



Context Help

WPI_JoystickRefNum Registry Set.vi

Sets a reference into a named registry storage. After setting a reference, you can retrieve it by name using the **RefNum Registry Get**.
The **Test Mode Only** parameter restricts it so Test Mode can use the reference, but other code cannot get the refnum and modify the I/O.

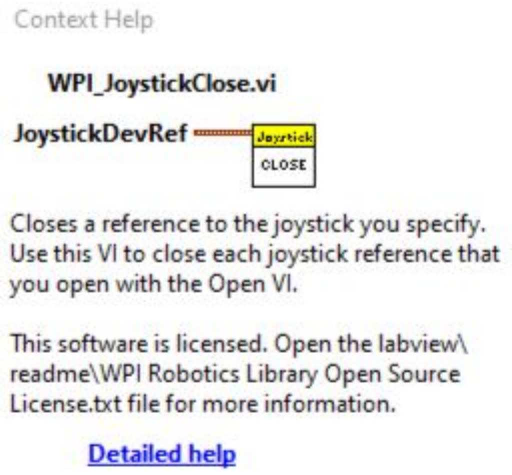


Context Help

WPI_JoystickRefNum Registry Get.vi

Gets the reference you specify. You first must set the reference with the RefNum Registry Set VI.

Detailed help



Context Help

WPI_JoystickClose.vi

Closes a reference to the joystick you specify. Use this VI to close each joystick reference that you open with the Open VI.

This software is licensed. Open the labview\readme\WPI Robotics Library Open Source License.txt file for more information.

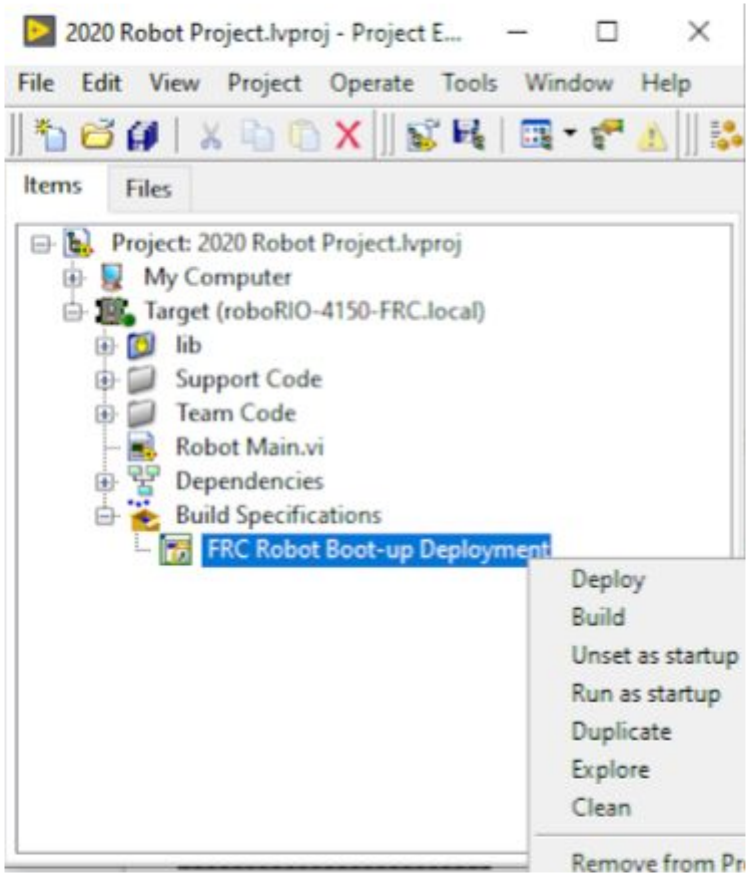
Detailed help

Robot Code: Build & Deploy

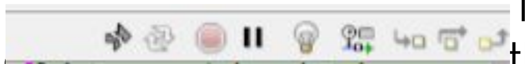
Once you've completed your robot code or feel like you want to test it, you're going to need to build it and deploy it to the roboRIO.

To do this go to the project window, under "Build Specifications", right click "FRC Robot Boot-up Deployment", and then click "Build" to build it and "Run as startup" to put the code onto the robot.

Note: The robot has to be connected/communicating to the robot to put the code onto it.



Robot Code: Troubleshooting

- If you have a general idea as to where the issue is, one way of troubleshooting could be reading through your code and thinking it through in your head as if you were the robot.
- You could also set up some sort of visual key to check that it's working, like displaying text saying that that part of the code is working.
  - Labview actually has a visual thing called "Highlight Execution", it's the little lightbulb in the top bar,  if you click on it and run the VI, while in the block diagram, it'll show you the movement of the data in your program, this way you can follow it and see if something's not working right
  - You can also display code in the network table variables
- Tips for making sure you don't run into major issues in the long-run
  - Keep your code as organized as possible. this may seem difficult, but if you use subVIs and try to make sure things don't get squished together you should be good
  - Test your code as often as possible. One of the worst things you could do is assume that all of your code is good and then test it all at once, it's a lot harder to find the issues then because there could be a lot of them.

Dashboard

Dashboard: Deciding What Should Go On

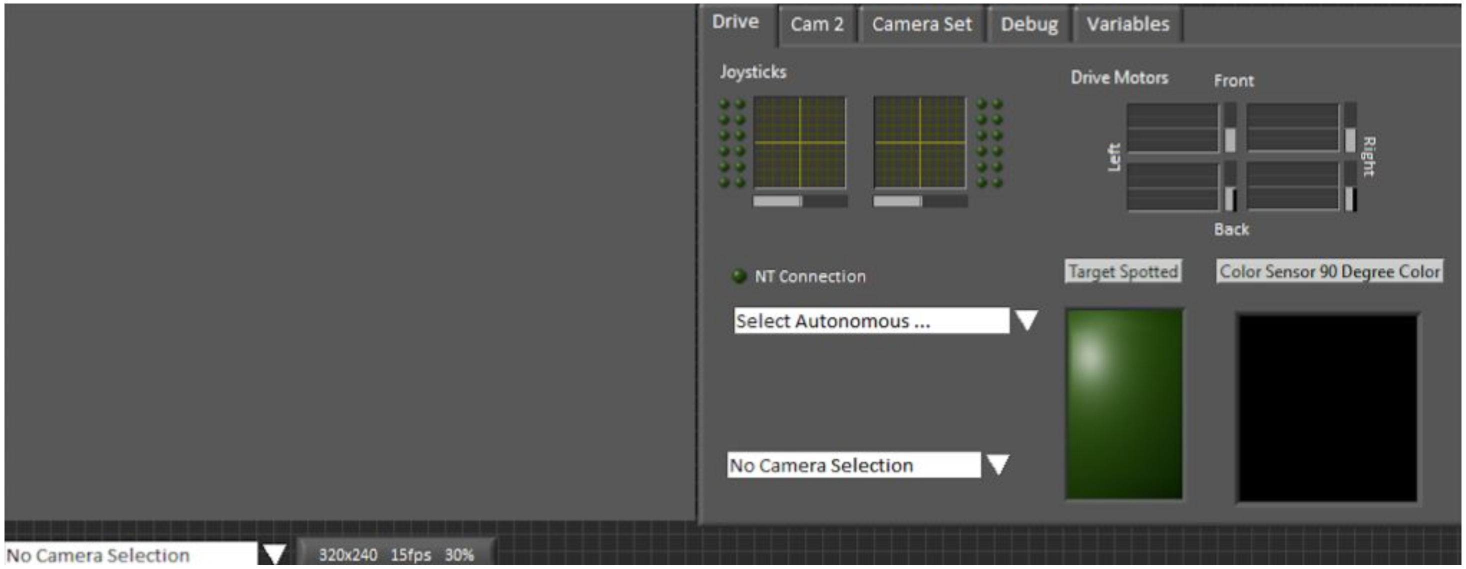
When deciding what to put on the dashboard you have to consider the game, what’s on your robot, and what your drive team wants to be able to see while in a match.

Every year no matter what the game, it’s usually good to have something along the lines of a motor input and motor output, this way if the motor is not working, it might help you to figure out where the issues at.

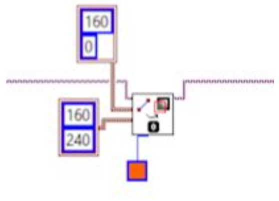
Examples of other things that you might consider putting on the dashboard are things like visuals for sensors, like a light for when a vision target is detected or a color for a color detection sensor, or you could have settings for a camera and things such as that.

Dashboard: Organizing

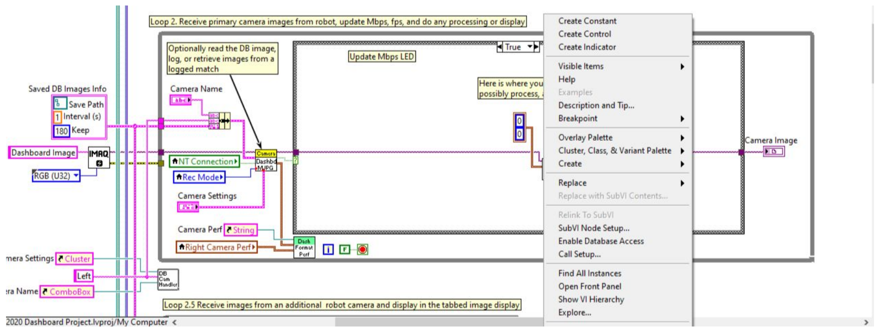
Once you have what you are going to put on the dashboard, it's important for you to organize it to the best of your ability. This usually means making sure the first tab is most important for in game use and then going on from there to things like camera settings, debugging, and all the network table variables (these should usually be the last tab). Other things to note too are that, anything you want the drivers to see, make sure it stands out and one other thing to consider adding is something if there's something wrong with the robot.



Dashboard: Camera Lines



Something that can be very useful is making lines on the camera video, whether it's just a line down the center to help the drive team center the robot or it's a box or circle around where the vision programming thinks the target is. I'll only show the line to you, but the concept is all the same and it shouldn't be hard for you to figure out the rest after that.

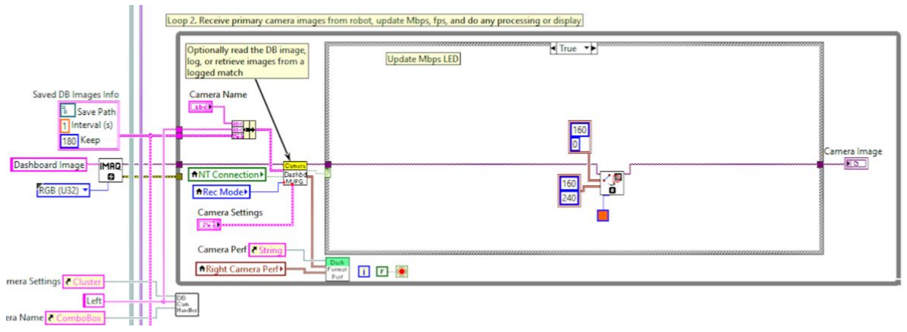

Loop 2. Receive primary camera images from robot, update Mbps, fps, and do any processing or display

Optionally read the DB image, log, or retrieve images from a logged match

Loop 2.5 Receive images from an additional robot camera and display in the tabbed image display

2020 Dashboard Project.lvproj/My Computer

To create the clusters for the X, Y coordinates of the line, right click on the input node and then click, "Create Constant" at the top. X is the top box, Y is the bottom.


Loop 2. Receive primary camera images from robot, update Mbps, fps, and do any processing or display

Optionally read the DB image, log, or retrieve images from a logged match

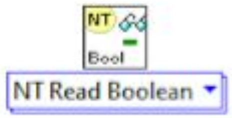
It's important that you give the line a color, or else it won't show up. Also note that the name of the block you want is IMAQ Overlay Line.

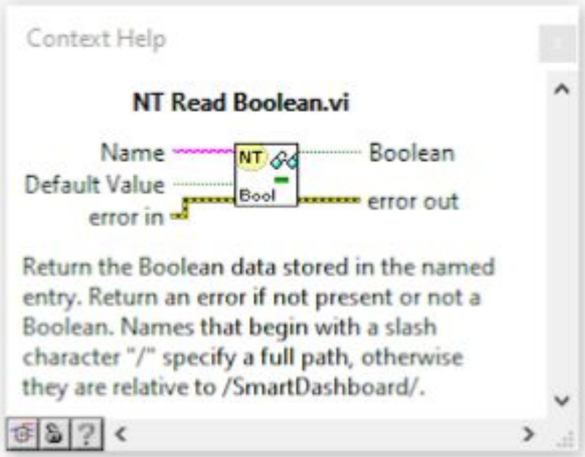
Dashboard: Important Blocks

Names for the network table variables shouldn't have spaces or anything like that.

Ex) /sandwich/peanut butter -> not a good name
/sandwich/peanutButter -> good name

The NT Read requires a name and gives you whatever variable is associated with the NT name that you give it. An important thing to note is that the name has to be written exactly the same as how you wrote it when you created it.

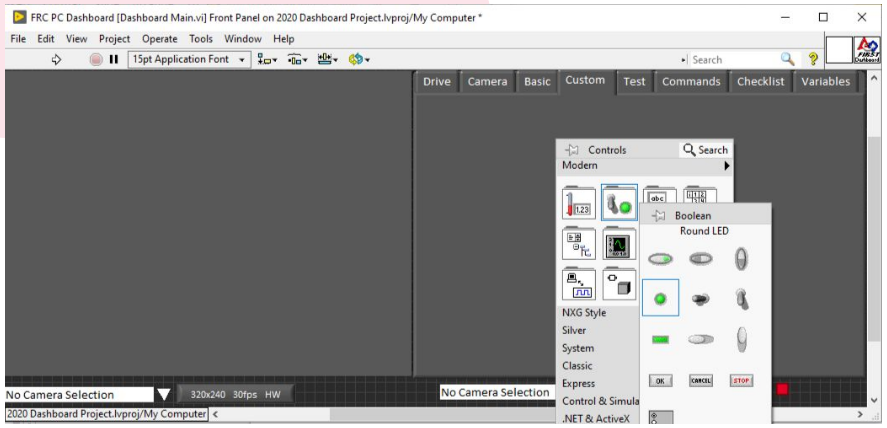




Return the Boolean data stored in the named entry. Return an error if not present or not a Boolean. Names that begin with a slash character "/" specify a full path, otherwise they are relative to /SmartDashboard/.

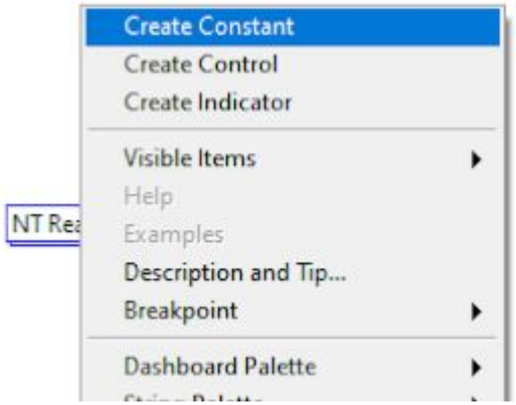
Since the information for the dashboard comes from network table variables, this means that you could program your robot code in whatever language you want and then publish the variables into network tables and create a LabVIEW dashboard with them. You can also not use NT Reads and simply name the control the same name as the network table variable, but it's a lot nicer to use the reads.

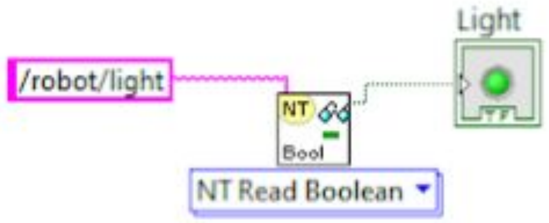
Dashboard: NT Read Example


FRC PC Dashboard [Dashboard Main.vi] Front Panel on 2020 Dashboard Project.lvproj/My Computer *
File Edit View Project Operate Tools Window Help
15pt Application Font
Drive Camera Basic Custom Test Commands Checklist Variables
Controls
Search
Modern
Boolean
Round LED
NXG Style
Silver
System
Classic
Express
Control & Simula
.NET & ActiveX
No Camera Selection
320x240 30fps HW
2020 Dashboard Project.lvproj/My Computer

To start right click on the Front Panel and select what you want to put there, for this example I'm doing a simple light.

After placing your object, double click on it to find it in the Block Diagram panel and find the NT Read in the panel by right clicking and searching for it and then connect it to the light. Once you've done that, you can right click on the pink node on the NT Read and click "Create Constant".





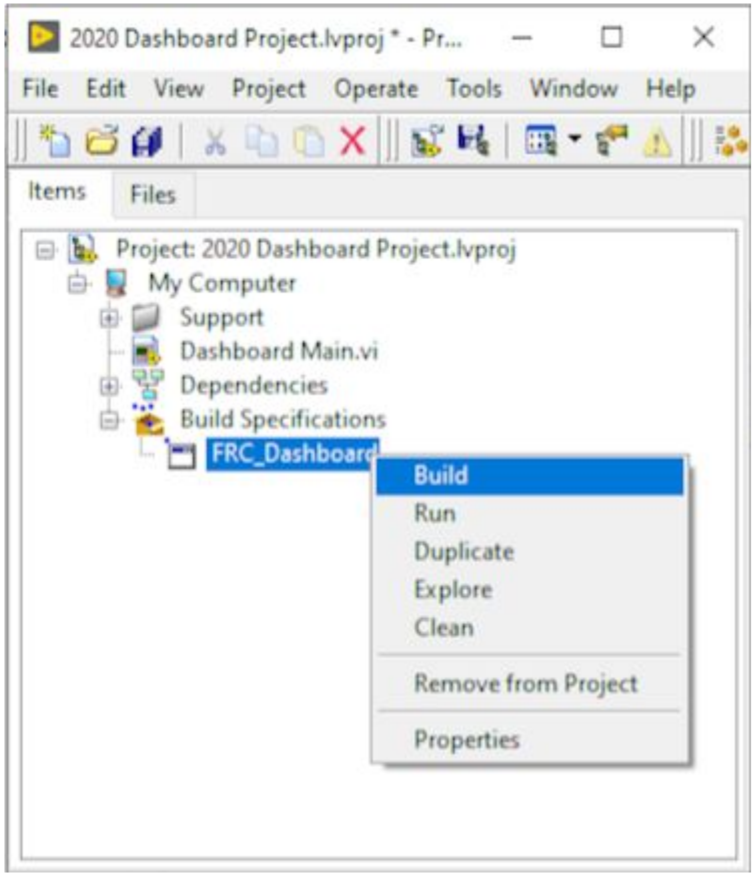
Finally, type in the NT name into the string. Always make sure you're using the right NT Read/Write, in this case I was using a light, which is a boolean, but numbers would need the "Read Number" one.

Dashboard: Building the Dashboard

Once you feel like your dashboard is good to go or you feel like parts of your dashboard are ready to test, you'll need to build it.

The way that you do this is by going back to the project view, under build specifications and right click on "FRC_Dashboard", then click on "Build".

Note that if you didn't let the project go under the LabVIEW data folder, then you may have some issues, check the Troubleshooting slides for more detail



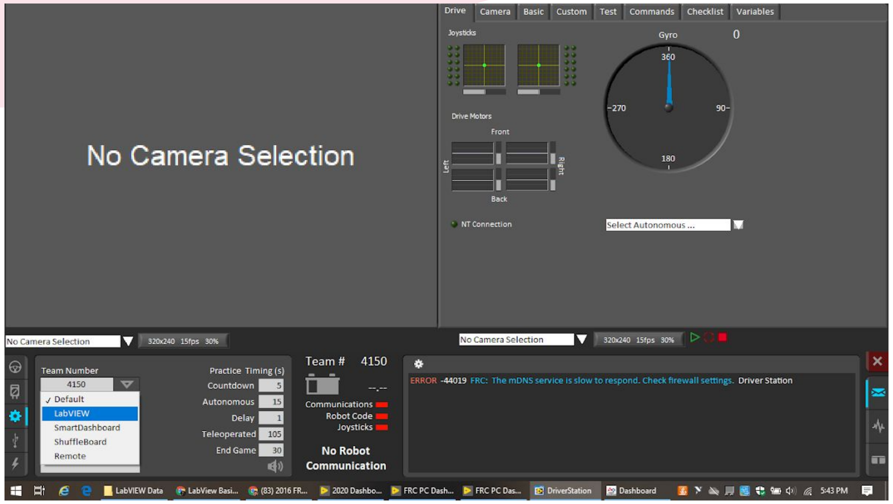
Dashboard: Troubleshooting

If your dashboard isn't working, there's a good chance that you just forgot to connect something or didn't connect something properly. The easiest way to check this is with the arrow at the top of the dashboard. White means that all of your code is connected properly, grey and broken means there's something wrong, you can see why if you click on the arrow and then if you click on the issue it'll take you to where it is in the code.

If your dashboard is having trouble building, it could be because the builds folder is not in the same file as the your project file/the file before your actual project.

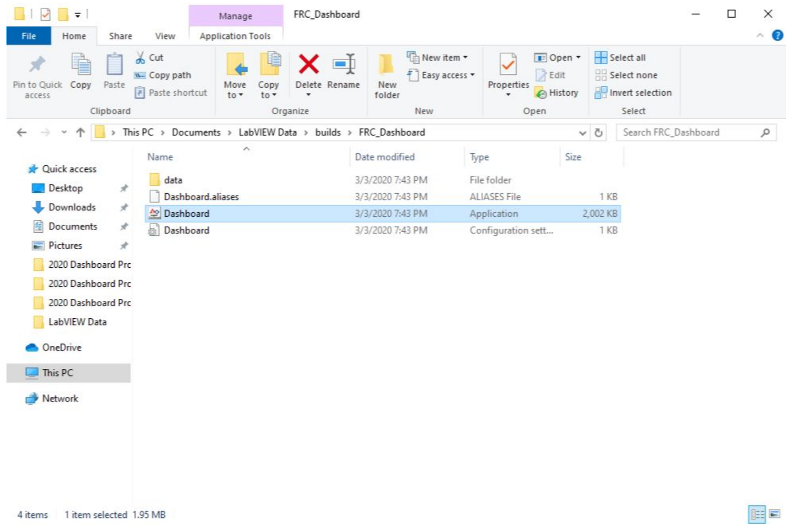
If you open up your dashboard and it looks all wonky, this usually means that when you were on the Front Panel you changed the screen size or scrolled to the side on accident. You have to go back to the project and make sure that it's perfectly (and I really mean perfectly) lined up and save it and rebuild it or else it will continue to look wonky.

Dashboard: Troubleshooting (Cont.)



ERROR -44019 FRC: The mDNS service is slow to respond. Check firewall settings. Driver Station

If you go to open up the dashboard and the default shows up instead of your dashboard, don't panic, it usually just means that the setting on the dashboard is wrong. Click on the gear on the bottom left of the dashboard, click the dropdown arrow that says "Default" and then click "LabVIEW".



This PC > Documents > LabVIEW Data > builds > FRC_Dashboard

Name	Date modified	Type	Size
data	3/3/2020 7:43 PM	File folder	
Dashboard.aliases	3/3/2020 7:43 PM	ALIASES File	1 KB
Dashboard	3/3/2020 7:43 PM	Application	2,002 KB
Dashboard	3/3/2020 7:43 PM	Configuration sett...	1 KB

4 items 1 item selected 1.95 MB

If only the bottom half of the dashboard is showing up, go to the "FRC_Dashboard" folder that's inside the "builds" folder and create a shortcut on your desktop of the "Dashboard" thing that I have highlighted in the photo, so that if only the bottom half opens up again, you can click on the shortcut and it'll open the top half as well.

Where to go next

Understanding State Machines -> Video 1, Video 2

There are tutorials called the FUNdamentals of LabVIEW for FRC, the first one teaches the basics, a lot of what was covered in this slideshow, it'd be good to skim through, the second one talks about vision and control, and the third one talks about the command and control robot project type. The videos were made in 2017, but a lot of it is still relevant.

There's a video on robot motor control.

The Secret Book of FRC LabVIEW can also be very helpful as it covers a wide range of LabVIEW programming, including a couple things discussed here.

Some other general things you should learn that I don't have specific resources for: variables, graphs, timing, position control, PID control (a more advanced topic), also it's the job of a good programmer to understand the various parts of the robot, such as the motors, enconders, and solenoids.

Final Thoughts

If there’s one thing I want to say before you exit this presentation, it’s that you shouldn’t look at this and think it’s too much to do or that you can’t do it, you won’t learn anything by doing that. One of the best ways for you to learn is by doing, so I suggest that you go, open up LabVIEW, and just mess around with it for a bit and you’ll start to understand and see that it’s not so bad.

Plus this is just the tip of the iceberg for you beginning to understand LabVIEW programming. There are so many great resources out there created by other FIRST teams and even resources from National Instruments.

So go out and create the most epic program you’ve ever created and never be discouraged because anything’s possible if you work hard enough.

Thank You

Credits

- This lesson was written by FRC 4150 in partnership with FRC 8027 for FRCTutorials.com
- You can contact the author at froboticsteam4150@gmail.com.

- More lessons for FIRST Robotics Competition are available at www.FRCtutorials.com