

---

# **FIRST Robotics Documentation**

**Jul 13, 2019**



<b>1</b>	<b>New for 2019!</b>	<b>1</b>
<b>2</b>	<b>Known Issues</b>	<b>7</b>
<b>3</b>	<b>FRC Software Component Overview</b>	<b>9</b>
<b>4</b>	<b>Offline Installation Preparation</b>	<b>23</b>
<b>5</b>	<b>Windows Offline Install Guide</b>	<b>25</b>
<b>6</b>	<b>MacOS Offline Install Guide</b>	<b>29</b>
<b>7</b>	<b>Linux Offline Install Guide</b>	<b>35</b>
<b>8</b>	<b>Installing the FRC Update Suite</b>	<b>41</b>
<b>9</b>	<b>3rd Party Libraries</b>	<b>57</b>
<b>10</b>	<b>Imaging your roboRIO</b>	<b>61</b>
<b>11</b>	<b>Imaging your Classmate (Veteran Image Download)</b>	<b>67</b>
<b>12</b>	<b>What is WPILib?</b>	<b>73</b>
<b>13</b>	<b>RoboRIO Web Dashboard</b>	<b>75</b>
<b>14</b>	<b>Operating pneumatic cylinders</b>	<b>81</b>
<b>15</b>	<b>Strategies for vision programming</b>	<b>83</b>
<b>16</b>	<b>Read and process video: CameraServer class</b>	<b>87</b>
<b>17</b>	<b>2017 Vision Examples</b>	<b>91</b>
<b>18</b>	<b>Target Info and Retroreflection</b>	<b>93</b>
<b>19</b>	<b>Identifying and Processing the Targets</b>	<b>99</b>
<b>20</b>	<b>Configuring an Axis Camera</b>	<b>105</b>

<b>21 Using the Microsoft Lifecam HD-3000</b>	<b>117</b>
<b>22 Camera Settings</b>	<b>121</b>
<b>23 Calibration</b>	<b>127</b>
<b>24 Axis M1013 Camera Compatibility</b>	<b>135</b>
<b>25 Using the Axis Camera at Single Network Events</b>	<b>137</b>
<b>26 Using the CameraServer on the roboRIO</b>	<b>143</b>
<b>27 Using multiple cameras</b>	<b>147</b>
<b>28 Introduction to GRIP</b>	<b>151</b>
<b>29 Reading array values published by NetworkTables</b>	<b>161</b>
<b>30 Generating Code from GRIP</b>	<b>165</b>
<b>31 Using Generated Code in a Robot Program</b>	<b>169</b>
<b>32 Using GRIP with a Kangaroo Computer</b>	<b>173</b>
<b>33 Using a Coprocessor for vision processing</b>	<b>177</b>
<b>34 Using the Raspberry PI for FRC</b>	<b>179</b>
<b>35 What you need to get the PI image running</b>	<b>181</b>
<b>36 Installing the image to your MicroSD card</b>	<b>183</b>
<b>37 The Raspberry PI</b>	<b>187</b>
<b>38 FRC Control System Hardware Overview</b>	<b>197</b>
<b>39 How to wire an FRC Robot</b>	<b>219</b>
<b>40 Wiring Best Practices</b>	<b>253</b>
<b>41 Wiring Pneumatics</b>	<b>257</b>
<b>42 Status Light Quick Reference</b>	<b>261</b>
<b>43 FRC CAN Device Specifications</b>	<b>273</b>
<b>44 Robot Preemptive Troubleshooting</b>	<b>277</b>
<b>45 IP Networking</b>	<b>287</b>



---

**Note:** This article describes the major changes to the Control System software for FRC 2019

---

## 1.1 Major Changes

- New IDE - VSCode (replacing Eclipse). This also includes a change from the Ant build system to GradleRIO
  - Vendor Libraries
- Pathweaver path planning software
- Raspberry PI image for image processing
- New Shuffleboard API
- New Java version - Java 11
- New roboRIO Webdashboard

## 1.2 Game Specifics

- Bandwidth - Bandwidth limit has been reduced to 4 Mbps. Like airline and hotel booking, the 7 Mbps limit included some assumptions that not all teams would use it. With the Sandstorm driving expected camera usage, we have lowered the limit in attempt to limit the likelihood teams will need to reduce below the limit at an event.
- Sandstorm Coding
  - Mode - Despite the Sandstorm not truly being an “Autonomous” mode, all code (DS, robot code, etc.) will continue to refer to this period as “Autonomous” or “Auto” for continuity.
  - Mode Transitions - The field will transition through modes in the same way as prior seasons. Prior to the start of the match, robot’s will be in Disabled Mode. Once the match starts, they will transition to

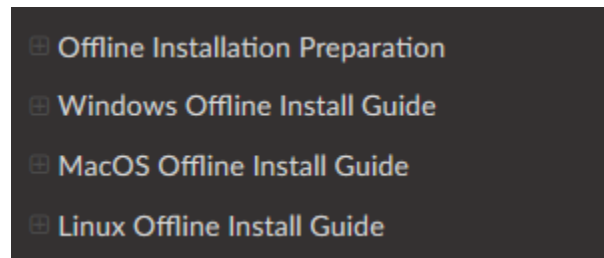
Autonomous Enabled mode. At the end of the Sandstorm, robots will briefly transition through Disabled (~.5s), if team's are driving their robot with joysticks/gamepads at this time, there will be a noticeable hitch. Finally the robot will transition to Teleop Enabled mode for the remainder of the match until the buzzer sounds. When the match expires, the robot will return to Disabled mode.

- Joystick Access - For this season only, joystick data will not be cached on the transition to Autonomous. Teams will be able to access updated joystick data throughout the Autonomous/Sandstorm period.
- Game Data - There is no game data for the 2019 season.

### 1.3 VS Code IDE (C++/Java)

For the 2019 season, FRC is moving from Eclipse to VS Code as the officially supported IDE for C++/Java development. This setup is easier for us to develop the extensions for and we expect it to be more streamlined and functional for the majority of teams. If VS Code doesn't seem to be your thing, the accompanying switch to GradleRIO should make it easier to use alternate IDEs as well (though this is not officially supported).

You can find information about the new IDE in the Offline Installation Guides. They can be found in the sidebar.



### 1.4 Vendor Libraries

- With the change in IDE/Build System there was also a change to how Vendor libraries integrate with the system. For more information, see [here](#):

### 1.5 RoboRIO WebDashboard

- No longer requires Silverlight
- No longer supports CAN device status/configuration
- Recommended to be used in non-IE browser

More information [here](#)

### 1.6 RoboRIO Imaging Tool

- Reworked UI
- Added firmware update capability

More information [here](#)

## 1.7 CAN Device Configuration

- Removed from WebDashboard
- Provided by vendor specific tools.

See this article for updated PDP/PCM configuration instructions

## 1.8 WPILib C++/Java

- A new high level CAN API was added. This enables easier creation of custom CAN devices for users.
- The OSSerialPort class was removed. It was a bugfix for some Serial port bugs, but never actually worked anyway.
- SensorBase was removed. If you were deriving from SensorBase you should now derive from SendableBase (C++) or SendableBase+ErrorBase(Java). If you need methods previously contained in SensorBase, they are now in SensorUtil.
- The Watchdog class was added. This calls a callback function when the specified timeout is exceeded. This class is used by TimedRobot to warn on periodic loop time overruns (nominally 20ms) in user code. **Note: These Timed Robot messages are warnings. They inform you if your code overruns the specified timing. They are not errors.**
- IterativeRobot was deprecated for eventual removal. TimedRobot is a drop-in replacement that is strictly better with regard to execution jitter.
- Linux was updated to require Ubuntu 18.04 as its default OS to run WPILib tools
- Building the libraries:
  - All libraries require C++14 capability to build. On Linux, this means GCC 6, and on Windows MSVC 2015 or newer.
  - Linux was updated to require Ubuntu 18.04 as its default OS to link to the libraries.
  - Windows builds are done with MSVC 2017 update 9, so linking to the libraries requires 2017 update 9 or later.

## 1.9 WPILib C++

- The namespace frc shim was removed. WPILib classes need to be accessed by their full namespaces, or using statements added.
- **Headers were moved to an frc folder.**
- HAL methods were removed from WPILib headers. In order to access HAL methods, their headers must be directly included.
- LlvM renamedspaced to wpi, headers moved to wpi folder
- All classes now properly support move semantics (for those who know what that is and care).
- Doxygen comments were moved from the source files to the header files for easier reference.

## 1.10 WPILib Java

- The CameraServer class was moved to a new package, and the class in the old location was deprecated. Please move to the new package location.
- The JNI classes were moved to a new package.
- Any Java class that had a free method was changed to implement AutoClosable and have a close method instead.
- The main method was moved from being internal in WPILib to explicitly defined by user code. This removes the reflection loading of the main robot class, and makes changing your robot class a compile time error rather than a runtime error. **Teams should not need to edit the Main.java file.**

## 1.11 CameraServer (cscore)

- USB cameras are now supported on Windows desktop builds

## 1.12 All WPILib Tools (Shuffleboard, Robot Builder, etc.)

- Are now installed to `~home/frcYYYY/tools` (where YYYY is the year and `~home` on Windows is `C:/Users/Public`).
- Run ToolsUpdater.bat (Windows) or ToolsUpdater.py (Mac/Linux) or Install Tools from GradleRIO to install tools.
- Folder contains .vbs files for Windows and .sh files for Mac/Linux that should be used to run the program. This sets the program up to run using the FRC specific JDK (which it has been tested with).

## 1.13 Shuffleboard

- New roboRIO API for automatically placing widgets on tabs and setting formatting options. More info here
- Camera viewer widget with adjustable stream parameters

## 1.14 PathWeaver/Pathfinder

- Added PathWeaver as UI to create paths for Pathfinder V1
- Generated paths are automatically downloaded to the RIO as part of the gradle configuration

Find more documentation on Pathweaver here

## 1.15 Raspberry Pi Image for Cameras

A pre-made Raspberry Pi Image for camera streaming/image processing has been developed to lower the barrier to entry to off-board vision processing. This image contains all of the libraries required to implement FRC compatible camera streaming, as well as a helpful web dashboard, read only file system configuration to handle robot power off and more. Learn more in the new manual here.

## 1.16 SmartDashboard

- Support for plugins has been removed. It was prohibitively difficult to maintain this feature when moving to Java 11 for the minimal number of teams believed to still be using it.
- Requires Java 11 to run.

## 1.17 Robot Builder

- Generates projects for the new VS Code/GradleRIO system
- Java/C++: encapsulates hardware in the specific subsystem. RobotMap is removed.
- C++: Updated to idiomatic C++ to match VSCode examples.

## 1.18 Outline Viewer

- Requires Java 11 to run.



This article details known issues (and workarounds) for FRC Control System Software.

### 2.1 VSCode failing to launch offline (no network present)

There are reports that on some builds of Windows 10 (build 1809), VSCode may fail to launch if no network is present (note: this is different than no internet present).

Workaround: Install the Microsoft Loopback Adapter 1. Access the Device Manager (typically by clicking start and typing Device Manager) 2. Click the Network Adapter Category 3. Click the Action menu from the top bar and select Add Legacy Hardware 4. Click Next on the window that pops up 5. Select the second option to install manually 6. Select "Network Adapter" and click Next. You will need to wait a bit for the next screen to populate. 7. Select Microsoft in the Manufacturer pane, then Microsoft Loopback Adapter in the right pane. 8. Click Next twice to install the adapter, then Finish to close the window.

### 2.2 Driver Station Dashboard launching

Issue: When selecting SmartDashboard or Shuffleboard in the FRC Driver Station, the programs fail to launch.

Workaround: Use the Desktop icons to launch the desired dashboard manually.

Solution: [Fixed in 2019.1.0 NI Update Suite](#)

### 2.3 C++ Intellisense - Files Open on launch don't work properly

Issue: In C++, files open when VS Code launches will have issues with Intellisense showing suggestions from all options from a compilation unit and not just the appropriate ones. This is a bug in VS Code

Workaround: Close the files in VS Code, close VS Code, wait ~ 1 min, re-launch VS Code.

## 2.4 Auto SPI Does not work in v12 image (affects Analog Devices IMUs and Gyro)

The Auto SPI functionality used by Analog Devices IMUs (ADIS16448 and ADIS16470) and Gyro (ADXRS450) does not work correctly in the 2019-v12 image that is part of the kickoff release.

Workaround: After imaging, log into the robot console as admin (via serial or SSH) and run the command “updateNIDrivers”, then reboot. A simplified tool to execute this is in progress and more information about a full update with this issue resolved will be coming soon.

Solution: Fixed in 2019v13 image in 2019.1.0 NI Update Suite

## 2.5 SmartDashboard and Simlition fail to launch on Windows N Editions

Issue: WPILib code using CSCore (dashboards and simulated robot code) will fail to launch on Education N editions of Windows.

Solution: Install the Media Feature Pack from <https://www.microsoft.com/en-us/software-download/mediafeaturepack>

## 2.6 Installing and Using GradleRIO Directly May not Deploy JRE

If user installs GradleRIO directly (i.e. not from the offline installers or zips provided by the WPILib release) and then runs a deploy to the roboRIO without internet, the JRE may not be deployed. A single deploy with internet connected (connect to the roboRIO over USB) is needed to dow



---

## FRC Software Component Overview

---

*The FRC control system consists of a wide variety of mandatory and optional software components. These elements are designed to assist you in the design, development, and debugging of your robot code as well as assist with control robot operation and to provide feedback when troubleshooting. For each software component this document will provide a brief overview of its purpose, a link to the package download, if appropriate, and a link to further documentation where available.*

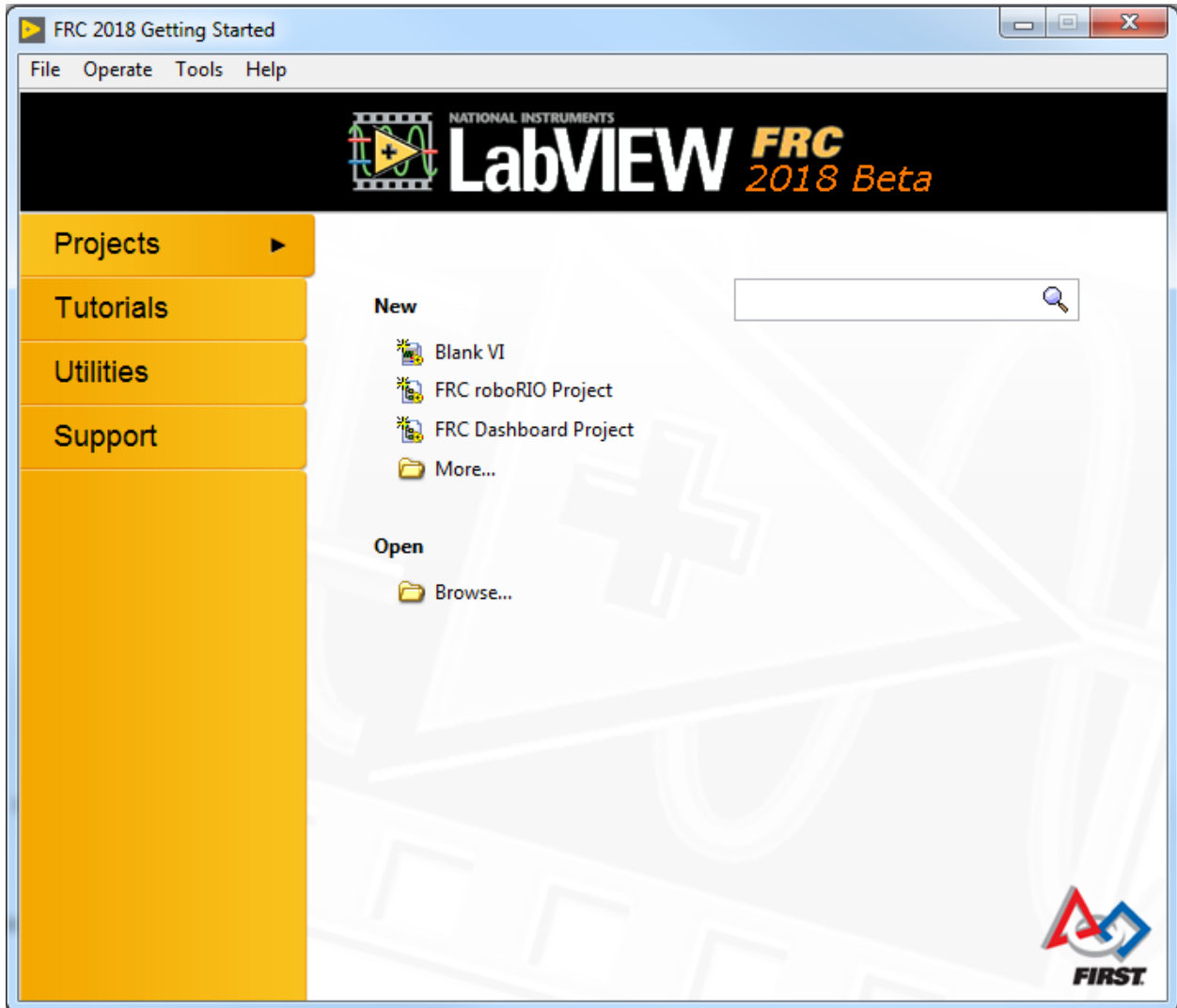
### 3.1 Operating System Compatibility

The primary supported OS for FRC components is Windows. All required FRC software components have been tested on Windows 7, 8, and 10. Windows XP is not supported.

Having said that, many of the tools for C++/Java programming are also supported and tested on Mac and Linux. Teams programming in C++/Java should be able to develop using these systems, using a Windows system for the Windows-only operations such as Driver Station, radio programming, and roboRIO imaging.

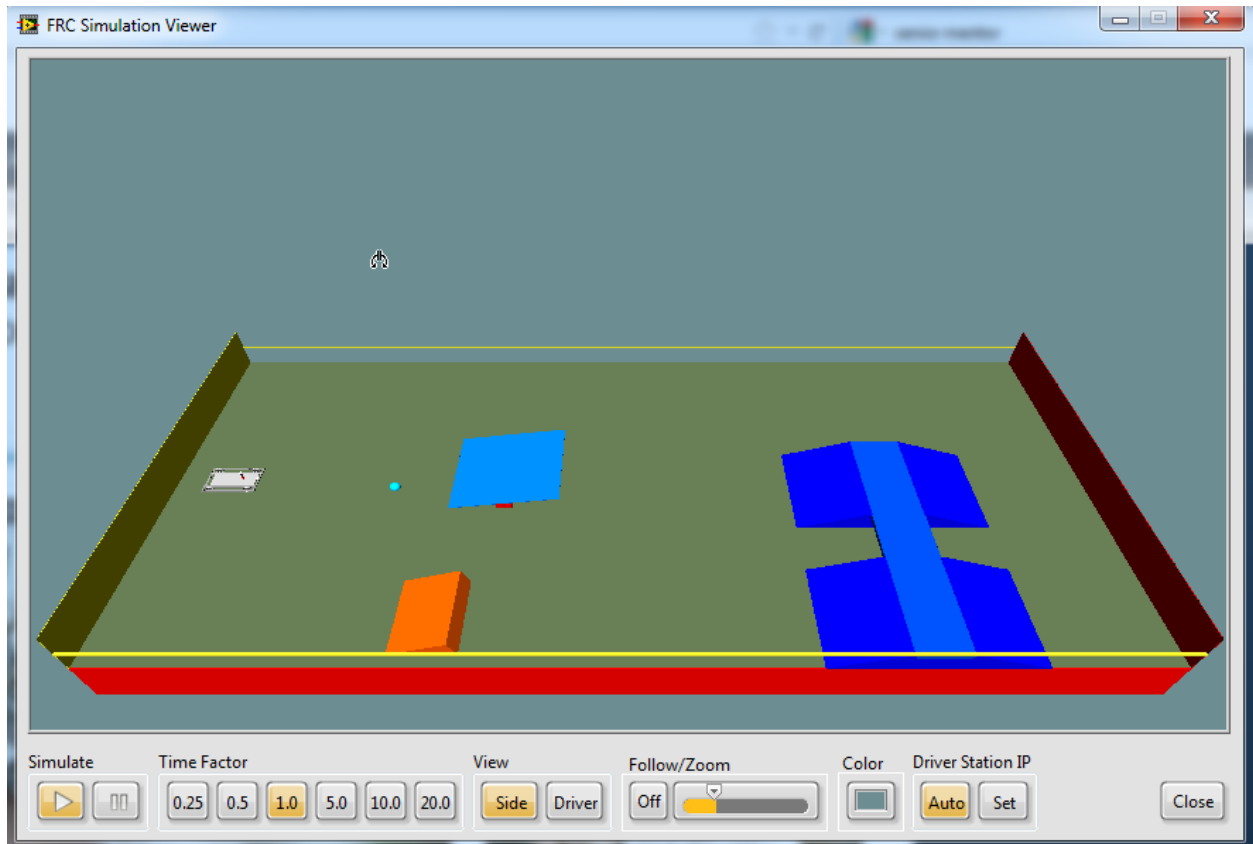
Components supported on all OS's have been marked with an \* below. All other items are Windows only, unless noted.

## 3.2 LabVIEW FRC



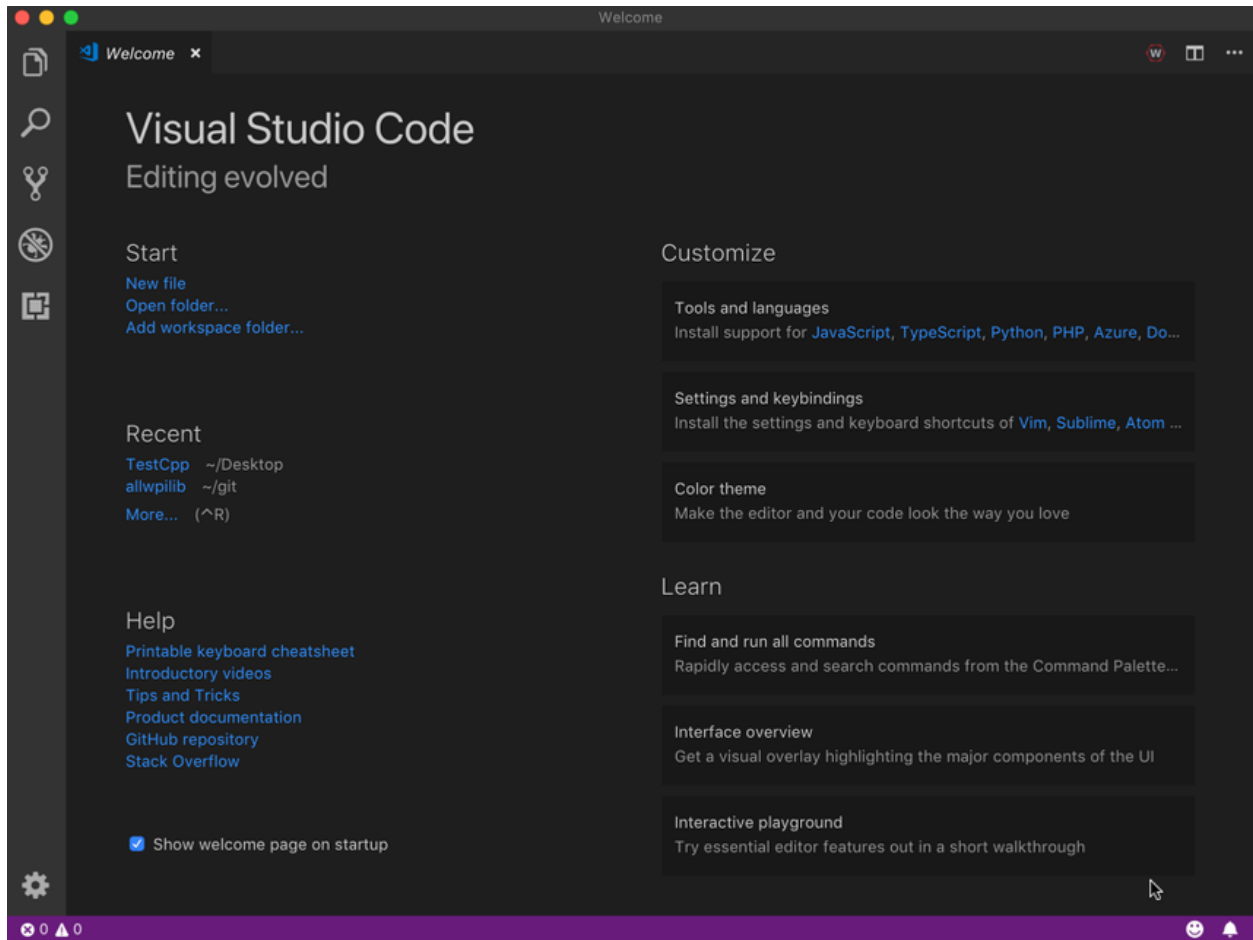
LabVIEW FRC, based on National Instruments' LabVIEW 2018, is the development environment for LabVIEW, one of the three officially supported languages for programming an FRC robot. LabVIEW is a graphical, dataflow-driven language. LabVIEW programs consist of a collection of icons, called VIs, wired together with wires which pass data between the VIs. The LabVIEW FRC installer is distributed on a DVD found in the Kickoff Kit of Parts and is also available for download (see installation instructions page linked below). Instructions for installing the FRC libraries (package also includes Driver Station and Utilities) can be found here. A guide to getting started with the LabVIEW FRC software, including installation instructions can be found here

### 3.2.1 FRC Robot Simulator



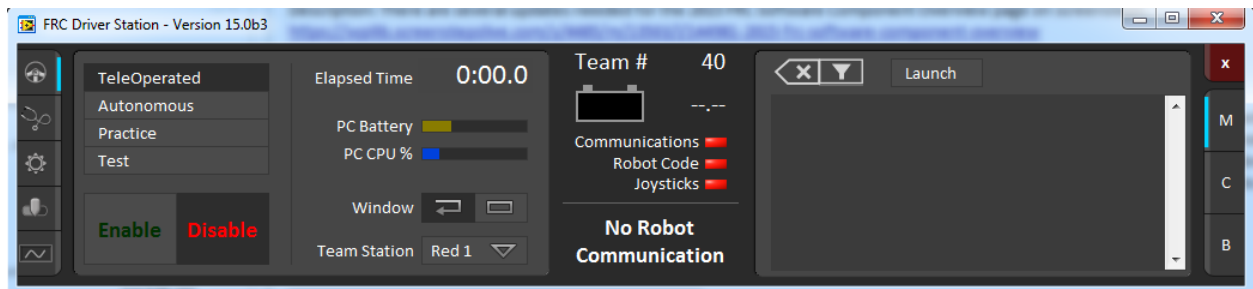
The FRC Robot Simulator is a component of the LabVIEW programming environment that allows you to operate a predefined robot in a simulated environment to test code and/or Driver Station functions. It utilizes a LabVIEW code project as the robot code and communicates with the FRC Driver Station for robot control and the FRC Default Dashboard for robot feedback. The FRC Robot Simulator is installed with the LabVIEW FRC package. Information on using the FRC Robot Simulator can be found by opening the Robot Simulation Readme.html file in the LabVIEW Project Explorer.

### 3.3 Visual Studio Code



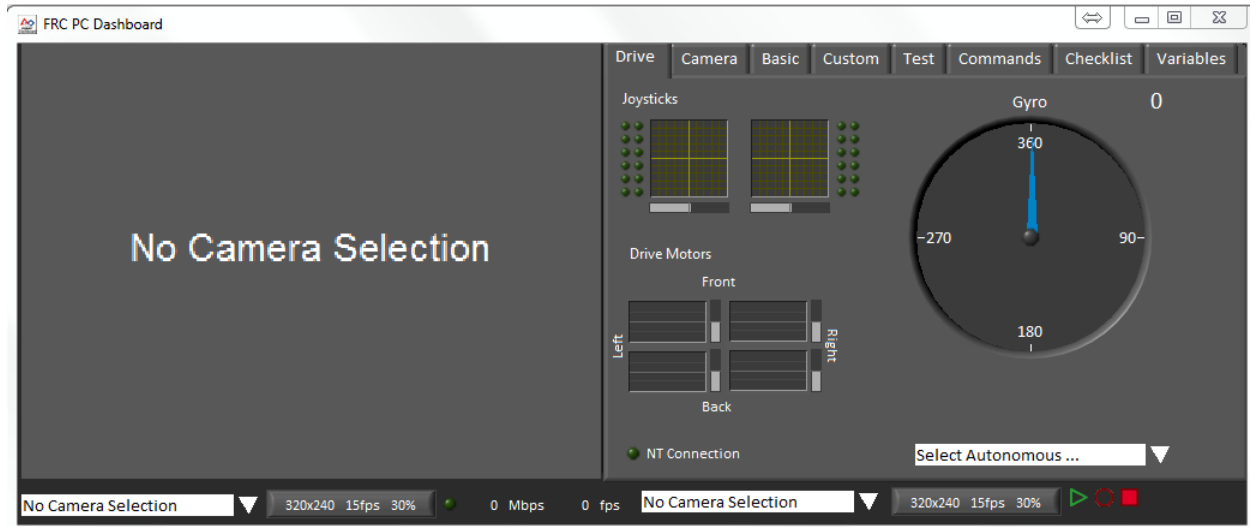
Visual Studio Code is the supported development environment for C++ and Java, two of the three supported languages used for programming an FRC robot. Both are object-oriented text based programming languages. A program in C++ (for FRC) consists of a number of header (.h) and implementation (.cpp) files where as a program in Java consists of .java files contained in one or more packages. A guide to getting started with C++ for FRC, including the installation and configuration of Visual Studio Code can be found [here](#). A guide to getting started with Java for FRC, including the installation and configuration of the Visual Studio Code can be found [here](#).

### 3.4 FRC Driver Station Powered by NI LabVIEW



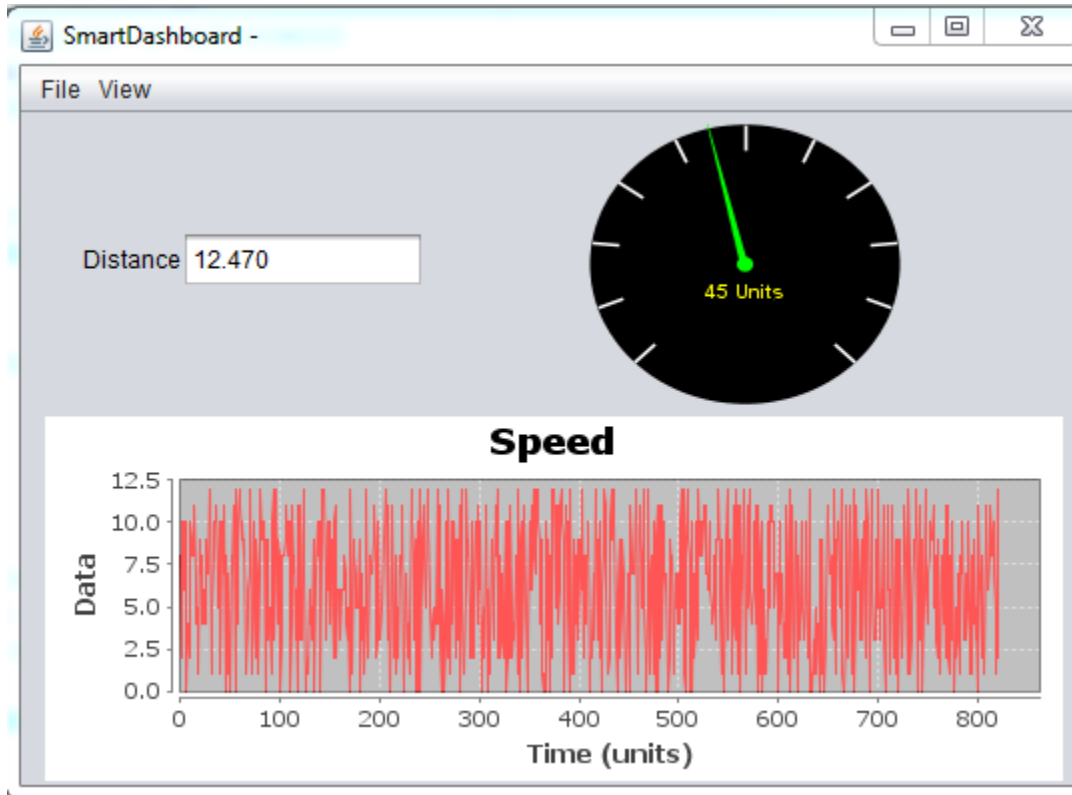
The FRC Driver Station Powered by NI LabVIEW is the only software allowed to be used for the purpose of controlling the state of the robot during competition. This software contains the code necessary to send data to your robot from a variety of input devices such as joysticks, gamepads, and customizable IO boards. It also contains a number of tools used to help troubleshoot robot issues such as status indicators and log file creation. Instructions for installing the FRC Driver Station Powered by NI LabVIEW (included in the FRC Update Suite) can be found [here](#), More information about the FRC Driver Station Powered by NI LabVIEW can be found [here](#).

### 3.5 FRC LabVIEW Dashboard



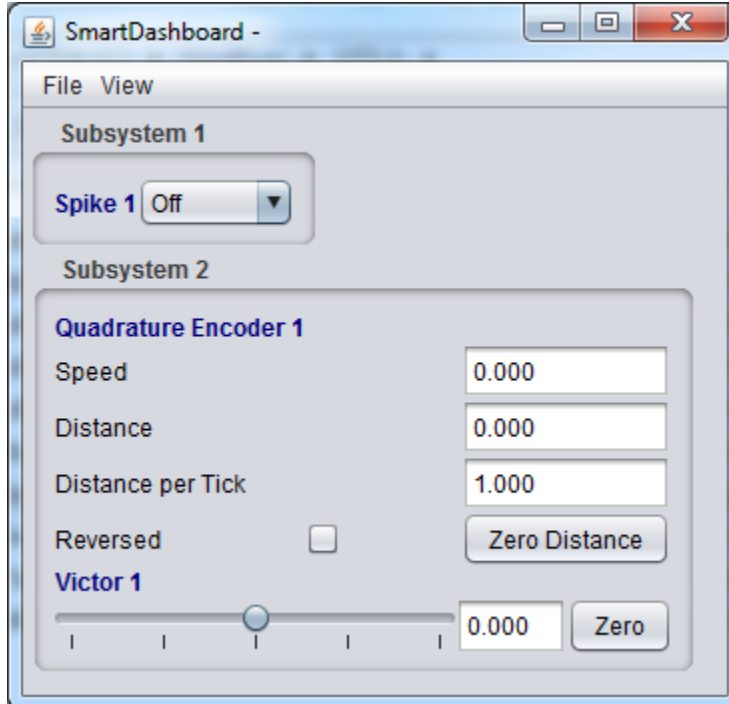
The FRC LabVIEW Dashboard is the default dashboard program installed with, and automatically launched by, the FRC Driver Station. The purpose of the Dashboard is to provide feedback about the operation of the robot. The FRC Default Dashboard serves as an example of the types of feedback teams may want from their robot. It includes a tabbed display that can switch between viewing an image from a camera on the robot or a display of NetworkTables variables, a display of information regarding the joysticks and drive motors, an indicator of the robot IP and battery voltage, and a second tabbed display that can switch between examples of custom indicators and controls, a test tab for use with the Driver Station Test Mode and a Checklist tab that teams can use to enter a custom checklist to complete before each match. The FRC Default Dashboard is included in the FRC Update Suite. Installation instructions can be found [here](#). More information about the FRC Default Dashboard software can be found [here](#).

## 3.6 SmartDashboard



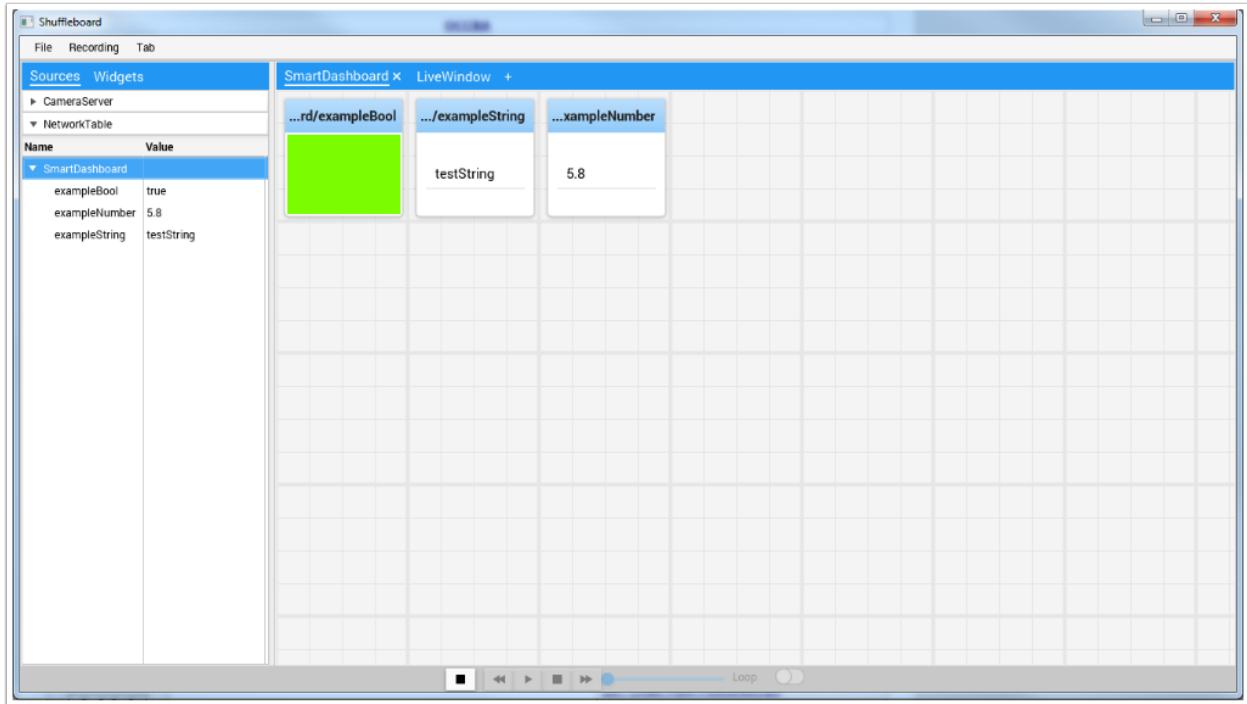
The SmartDashboard is an alternate dashboard application written in Java. The SmartDashboard automatically creates a widget for each variable sent from the Robot sent using the SmartDashboard class or VIs. These widgets can be configured to a number of preset display types, or users can create custom extensions in Java. Vision extensions are available for the SmartDashboard which allow it to display images from the Axis camera on the robot. The SmartDashboard is included in the C++ and Java language updates (enabled by clicking the C++ or Java buttons respectively on the Setup tab of the Driver Station). Additional documentation on the SmartDashboard can be found [here](#).

### 3.6.1 LiveWindow



LiveWindow is a new mode of the SmartDashboard for 2013, designed for use with the new Test Mode of the Driver Station. LiveWindow allows the user to see feedback from sensors on the robot and control actuators independent of the written user code. More information about LiveWindow can be found [here](#).

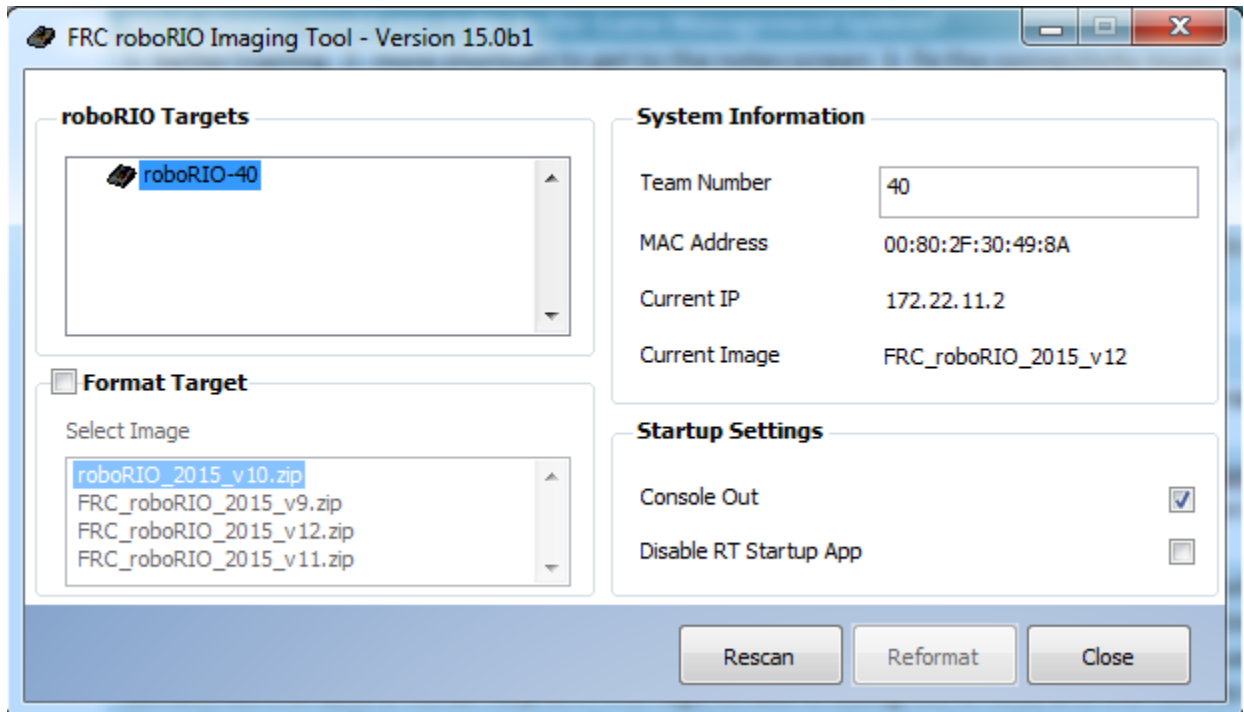
## 3.7 Shuffleboard



Shuffleboard is an alternative dashboard application written in Java. It takes many of the concepts from SmartDashboard such as automatic adding of widgets and adds new features including better layout control and record/playback functionality. Shuffleboard contains all of the basic widget types found in the SmartDashboard as well as a number of new ones intended to make visualizing specific robot components even easier. It has full integration with WPILib's "cscore" for displaying, recording, and playing back camera streams. Shuffleboard is included in the C++ and Java language updates (enabled by clicking the Shuffleboard button on the Setup tab of the Driver Station or by launching it from the WPILib menu in Eclipse). Additional documentation on Shuffleboard can be found [here](#).



## 3.8 FRC roboRIO Imaging Tool



The FRC roboRIO Imaging Tool is a software tool used to format and setup an roboRIO-FRC device for use in FRC. The tool detects any roboRIO device on the network, reports the current MAC, name, IP and Image version. The tool allows the user to configure the team number, set options including Console Out and whether an applications runs on Startup, and install the latest software image on the device. The FRC roboRIO Imaging Tool is installed as part of the FRC Update Suite. Installation instructions can be found [here](#). Additional instructions on imaging your roboRIO using this tool can be found [here](#).

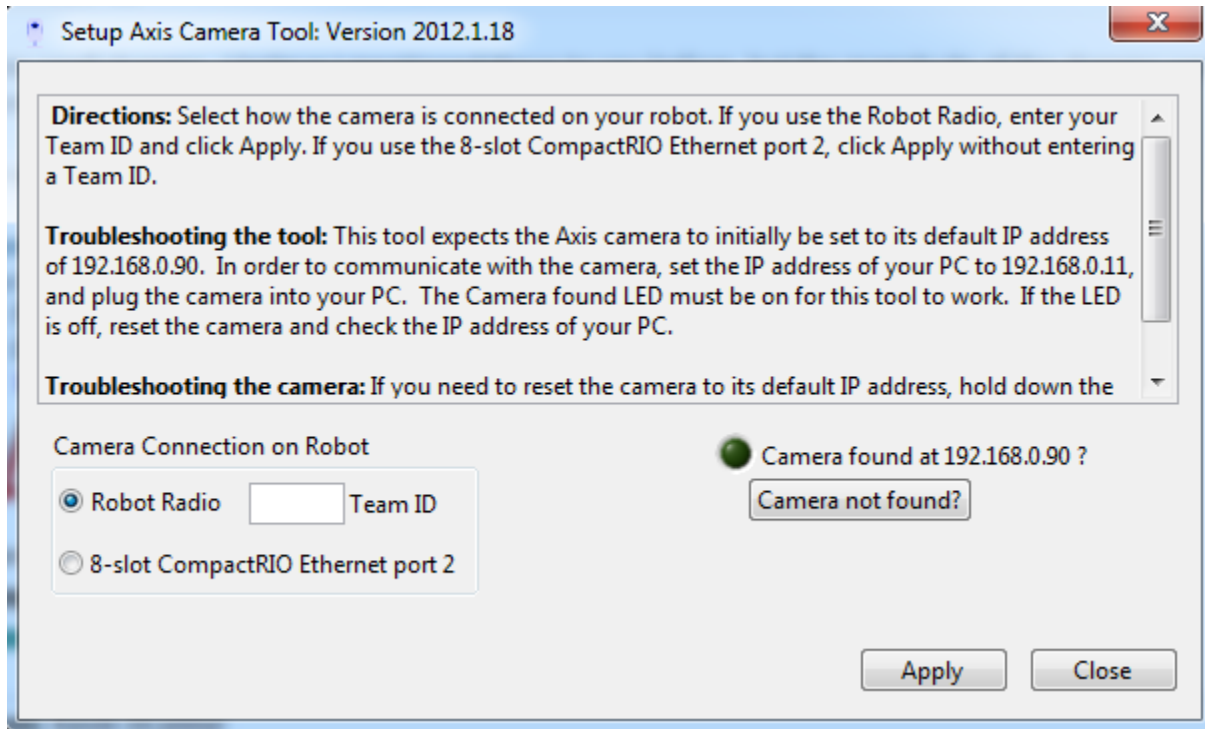
## 3.9 CTRE Toolsuite



The CTRE Toolsuite installs the software libraries for Talon SRX (C++/Java/LabVIEW) as well as the HERO Lifeboat software which can be used to update the roboRIO web based CAN configuration with the latest CTRE-specific features. The installer can be found here: <http://www.ctr-electronics.com/control-system/hro.html>

Note on non-Windows: A separate package (zip) is provided to get the Talon SRX and Pidgeon libraries on non-Windows systems. Users should unzip this file and place the contents into *USERwpilibuser* folder

## 3.10 Setup Axis Camera



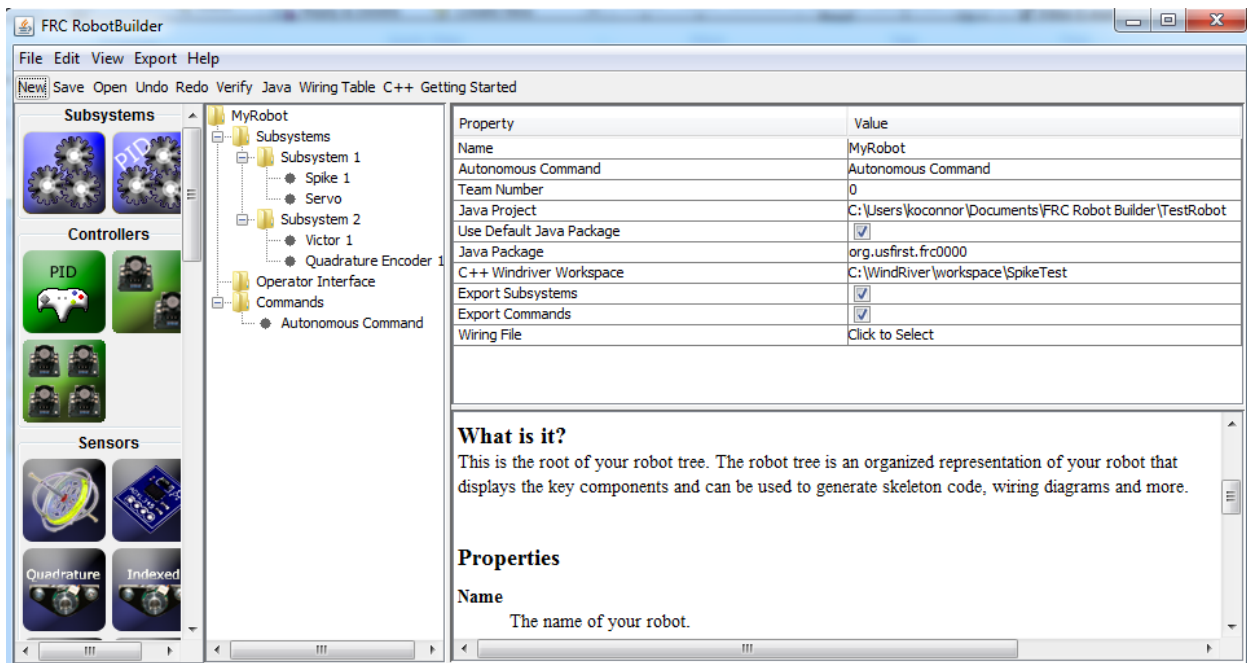
The Setup Axis Camera utility is a LabVIEW program used to configure an Axis 206, M1011 or M1013 camera for use on the robot. The tool takes a factory reset camera connected directly to the computer and configures the IP, username and password, anonymous access, and default framerate and compression (for use with the SmartDashboard or other access methods). The Setup Axis Camera tool is installed as part of the FRC Update Suite. Installation instructions can be found [here](#). Instructions for using the tool to configure the camera are located [here](#).

## 3.11 FRC Driver Station Log Viewer



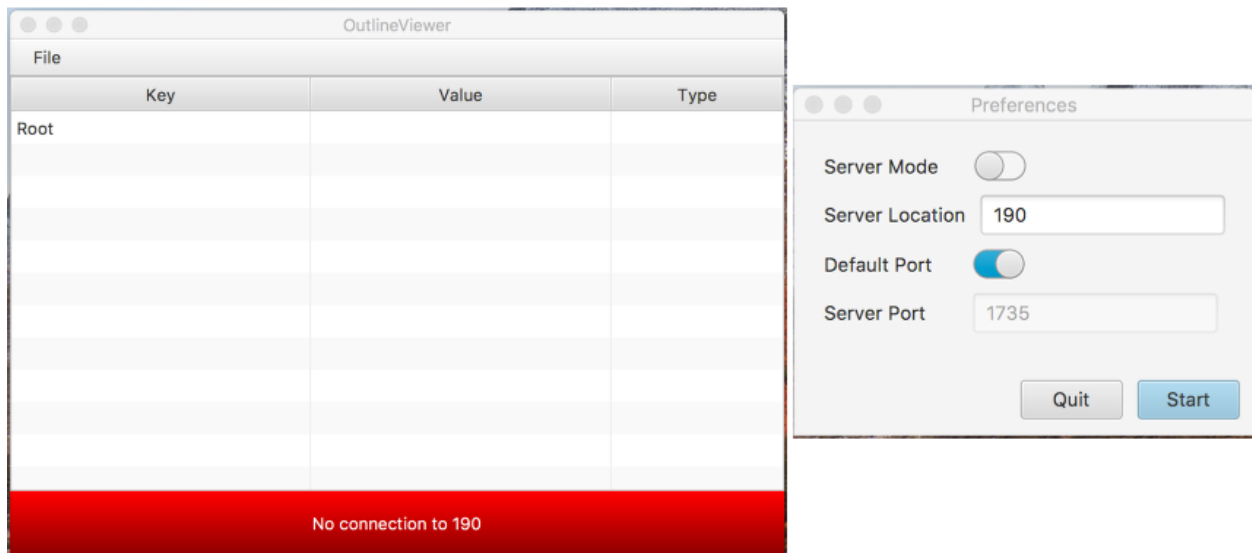
The FRC Driver Station Log Viewer is a LabVIEW program used to view logs created by the FRC Driver Station. These logs contain information such as battery voltage, trip time, CPU% and robot mode, as well as events such as joystick removal. The FRC Driver Station Log Viewer is included in the FRC Update Suite. Installation instructions can be found [here](#). More information about the FRC Driver Station Log Viewer and understanding the logs can be found [here](#).

## 3.12 Robot Builder



RobotBuilder is a tool designed to aid in setup and structuring of a Command Based robot project for C++ or Java. RobotBuilder allows you to enter in the various components of your robot subsystems and operator interface and define what your commands are in a graphical tree structure. RobotBuilder will then verify that you have no port allocation conflicts and can generate a wiring table indicating what is connected to each port as well as C++ or Java code. The code created generates the appropriate files, constructs the appropriate objects and adds LiveWindow code for each sensor and actuator, but does not write any of the actual Subsystem or Command methods. The user must write the appropriate code for these methods for the robot to function. More information about Robot Builder can be found here. More information about the Command Based programming architecture can be found in the C++ and Java sections.

### 3.13 OutlineViewer

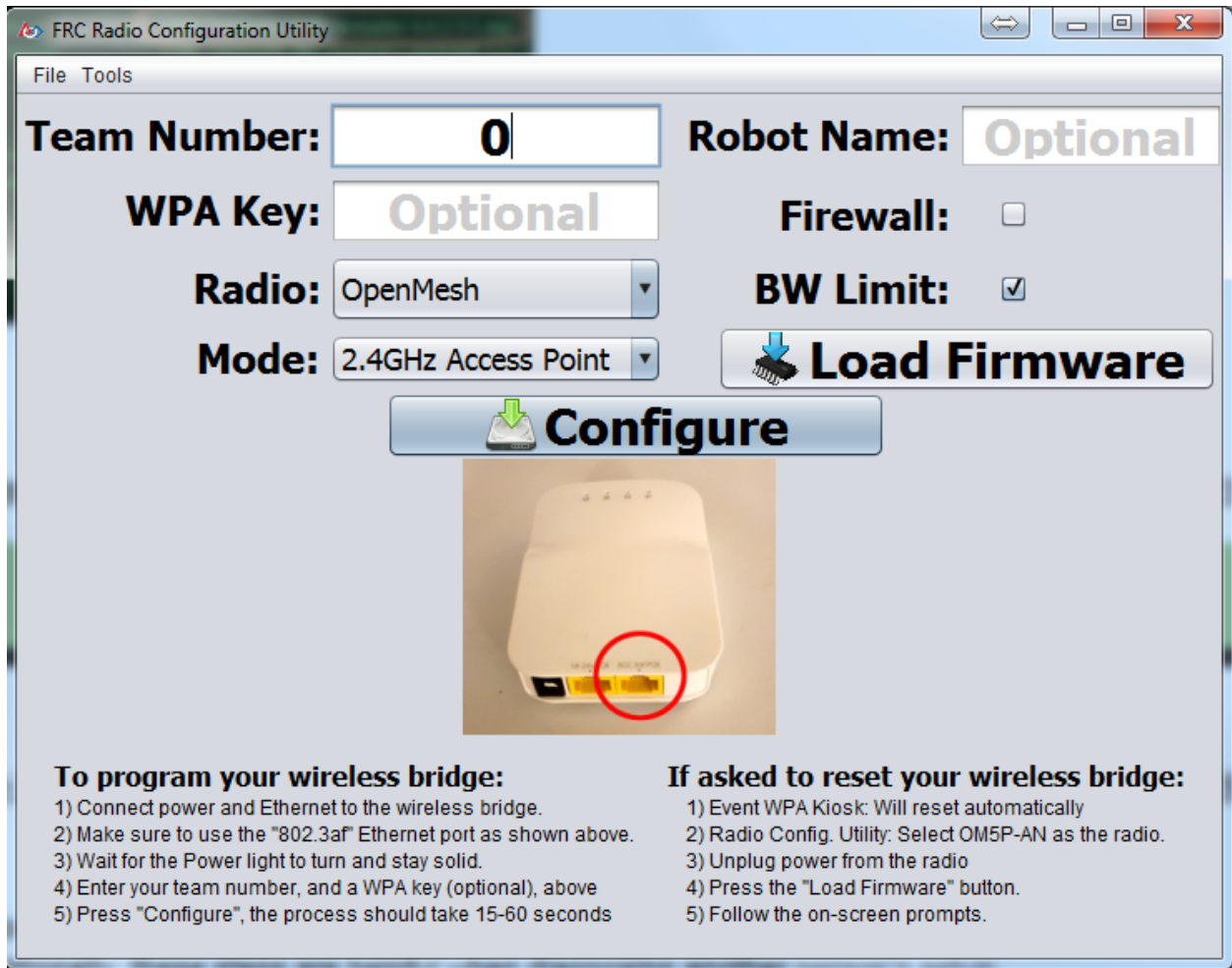


OutlineViewer is a utility used to view, modify and add to the contents of the NetworkTables for debugging purposes. It displays all key value pairs currently in the NetworkTables and can be used to modify the value of existing keys or add new keys to the table. OutlineViewer is included in the C++ and Java language updates (found in *toolswpilib*). Teams may need to install the Java Runtime Environment to use the OutlineViewer on computers not set up for Java programming.

To connect to your robot, open OutlineViewer and set the “Server Location” to be your team number. After you click start, OutlineViewer will connect.

LabVIEW teams can use the Variables tab of the LabVIEW Dashboard to accomplish this functionality.

### 3.14 FRC Radio Configuration Utility



The FRC Bridge Configuration Utility is a tool used to configure the the OpenMesh OM5P-AN or OM5P-AC radio for practice use at home. This tool sets the appropriate IP, and network settings for proper network connection, as well as the QOS settings required to mimic the bandwidth limiting and packet prioritization experience on the FRC playing field. The FRC Bridge Configuration Utility is installed by a standalone installer, instructions on installing and using the FRC Bridge Configuration Utility to configure your radio can be found [here](#).

---

## Offline Installation Preparation

---

This article contains instructions/links to components you will want to gather if you need to do offline installation of the FRC Control System software.

### 4.1 Documentation

The ScreenSteps documentation can be downloaded for offline viewing. At a minimum you will want to get a copy of the Getting Started with the Control System manual, you may also wish to download some or all of the other manuals as well. The link to download the PDF can be found [here](#)

### 4.2 Installers

#### 4.2.1 All Teams:

- [2019 NI Update Suite](#)(Note: Requires decryption key from kickoff broadcast!)
- [2019 FRC Radio Configuration Utility OR 2019 FRC Radio Configuration Utility Israel Version](#)
- (Optional - Veterans Only!) [Classmate/Acer PC Image](#)

#### 4.2.2 LabVIEW Teams:

- [LabVIEW USB](#) (from FIRST Choice) or [Download](#)

#### 4.2.3 C++/Java Teams

- [C++/Java WPILib Installer](#)

- VS Code (if using Windows, run the installer and use it to download the appropriate VS Code file. If using Mac/Linux, download VSCode from here: <https://code.visualstudio.com/download>)

### 4.3 3rd Party Libraries/Software

A number of software components were broken out of WPILib in 2017 and are now maintained by third parties. See [this blog](<http://www.firstinspires.org/robotics/frc/blog/2017-control-system-update> "this blog") for more details.

A directory of available 3rd party software that plugs in to WPILib can be found on this page.



---

## Windows Offline Install Guide

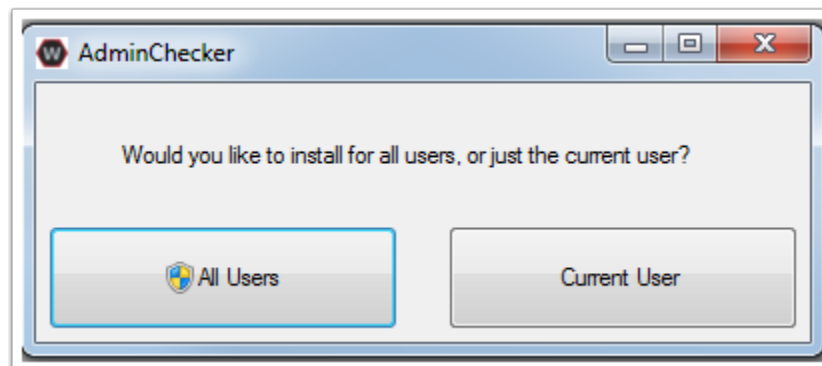
---

### 5.1 Offline Installer

**Note:** Windows 7: You must install the NI Update or .NET Version 4.62 (or later) before proceeding with the install of VSCode for FRC. The NIUpdate installer will automatically install the proper version of .NET. The stand alone .NET installer is here: [https://support.microsoft.com/en-us/help/3151800/the-net-framework-4-6-2-offline-installer-for-windows\\*](https://support.microsoft.com/en-us/help/3151800/the-net-framework-4-6-2-offline-installer-for-windows*)

Download the appropriate installer for your Windows installation (32 bit or 64 bit) from [GitHub](#). If you're not sure, open the Control Panel -> System to check.

Double click on the installer to run it. If you see any Security warnings, click Run (Windows 7) or More Info->Run Anyway (Windows 8+).

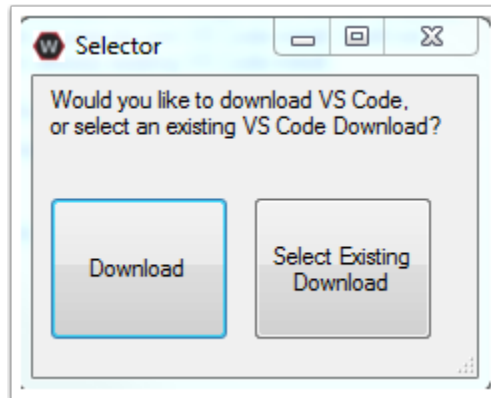


Choose whether to install for All Users on the machine or the Current User. The All Users option requires Admin privileges, but installs in a way that is accessible to all user accounts, the Current User install is only accessible from the account it is installed from.

If you select All Users, you will need to accept the security prompt that appears.

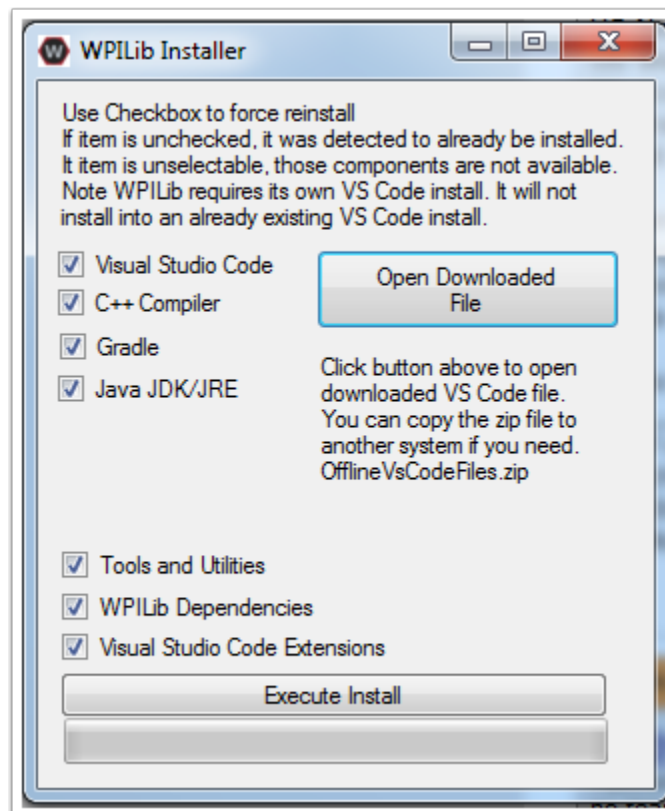
## 5.2 Download VSCode

For licensing reasons, the installer cannot contain the VSCode installer bundled in. Click Select/Download VSCode to either Download the VSCode installer or select a pre-downloaded copy. If you intend to install on other machines without internet connections, after the download completes, you can click Open Downloaded File to be taken to the zip file on the file system to copy along with the Offline Installer.



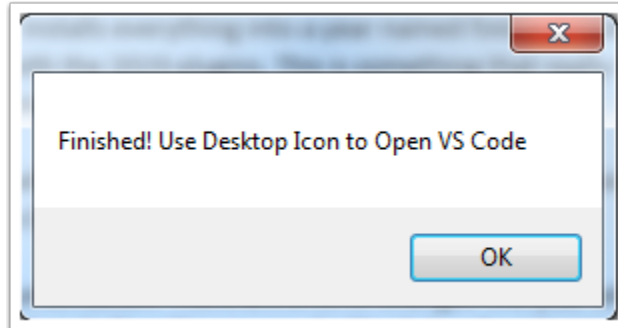
## 5.3 Execute Install

Make sure all checkboxes are checked (unless you have already installed 2019 WPILib software on this machine and the software unchecked them automatically), then click Execute Install.



## 5.4 Finished

When the installer completes, you will now be able to open and use the WPILib version of VSCode. If you are using any 3rd party libraries, you will still need to install those separately before using them in robot code.



## 5.5 What's installed?

The Offline Installer installs the following components:

- Visual Studio Code - The supported IDE for 2019 robot code development. The offline installer sets up a separate copy of VSCode for WPILib development, even if you already have VSCode on your machine. This is done because some of the settings that make the WPILib setup work may break existing workflows if you use VSCode for other projects.
- C++ Compiler - The toolchains for building C++ code for the roboRIO
- Gradle - The specific version of Gradle used for building/deploying C++ or Java robot code
- Java JDK/JRE - A specific version of the Java JDK/JRE that is used to build Java robot code and to run any of the Java based Tools (Dashboards, etc.). This exists side by side with any existing JDK installs and does not overwrite the JAVA\_HOME variable
- WPILib Tools - SmartDashboard, Shuffleboard, Robot Builder, Outline Viewer, Pathweaver
- WPILib Dependencies - OpenCV, etc.
- VSCode Extensions - WPILib extensions for robot code development in VSCode

## 5.6 What's Installed - Continued

The Offline Installer also installs a Desktop Shortcut to the WPILib copy of VSCode and sets up a command shortcut so this copy of VSCode can be opened from the command line using the command "frccode2019"



Both of these reference the specific year as the WPILib C++tools will now support side-by-side installs of multiple environments from different seasons.

---

## MacOS Offline Install Guide

---

The tools (except the Driver Station and the roboRIO Imaging Tool) will run natively on a Mac.

---

**Note:** Note: if you have the alpha release of VSCode for FRC installed, you should uninstall it before proceeding or create a new VSCode install. Failing to do this will have both versions installed at the same time causing things to not operate properly.

---

### 6.1 Getting Visual Studio Code

VSCode is the IDE (Integrated Development Environment) that is used for 2019 and beyond. It needs to be installed on any development computer. It can be downloaded here: <https://code.visualstudio.com>.

The downloaded file will be “VSCode-darwin-stable.zip” (1)

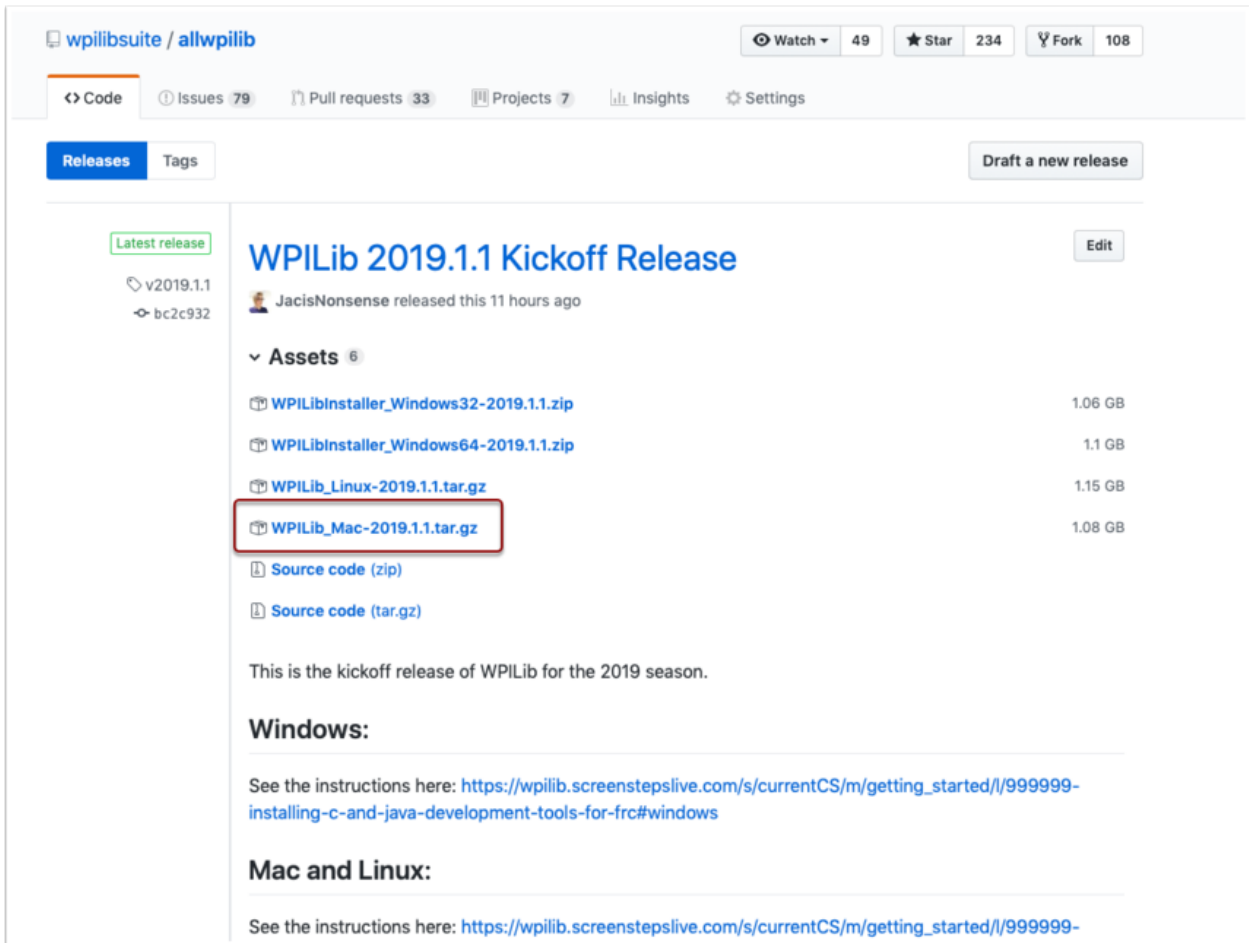
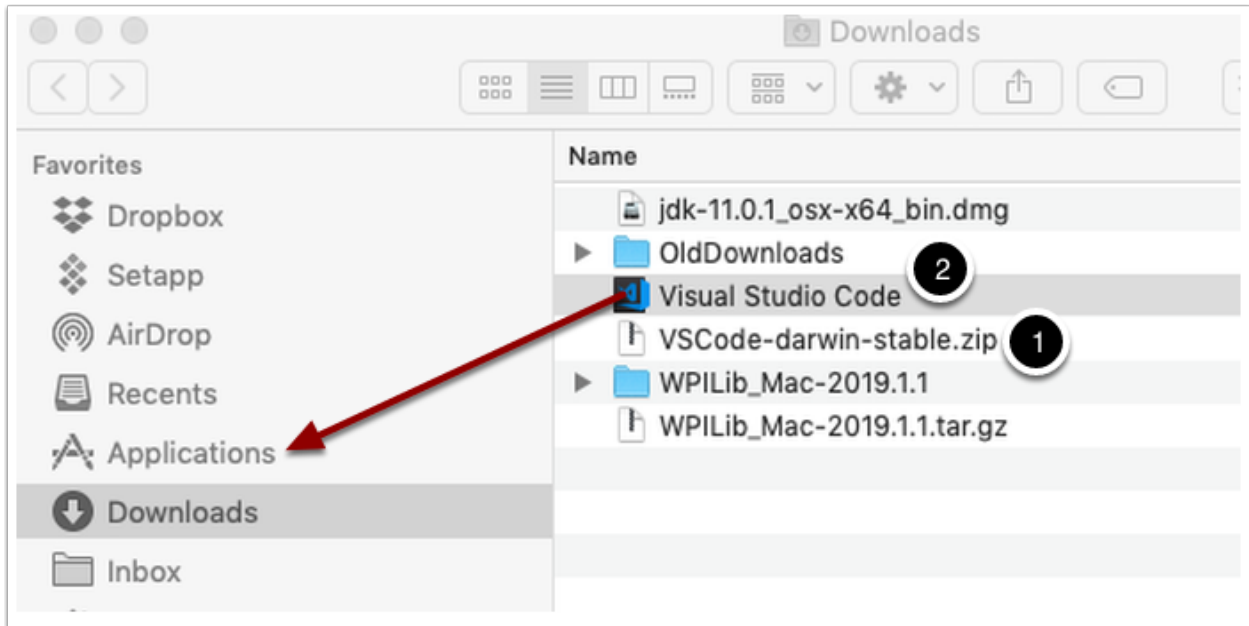
Once downloaded, double-click on the zip file to expand it and copy the new file: “Visual Studio Code” to the Applications folder (2).

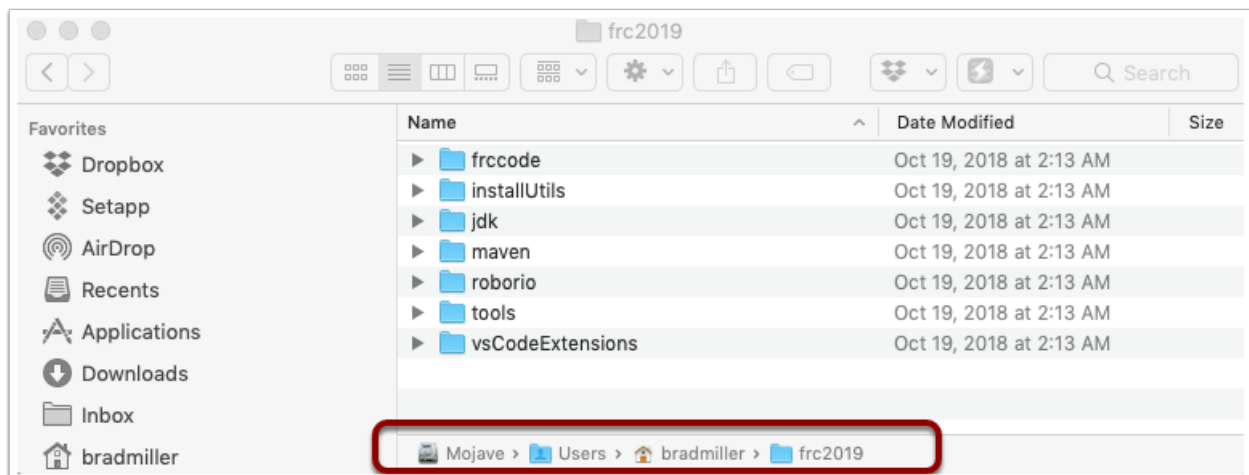
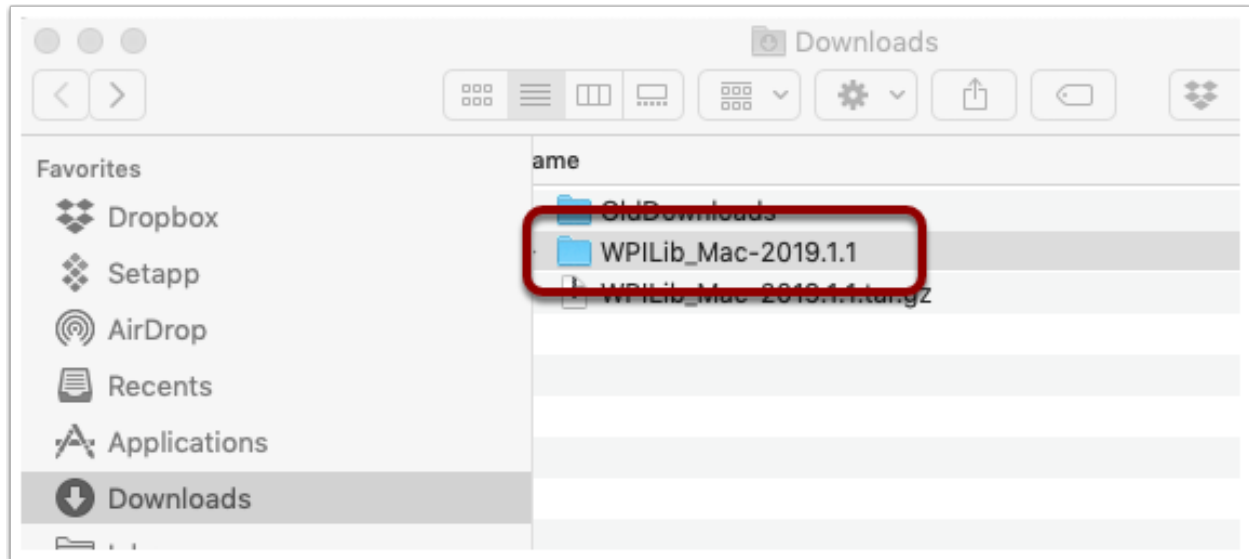
### 6.2 Download the WPILib release and move the directory

Download the software release by navigating to this page: <https://github.com/wpilibsuite/allwpilib/releases> and downloading the Mac release.

Unzip and untar the file by looking at the file in the explorer and double-clicking on it, once or twice to unzip (remove the .gz extension) and again to untar it (remove the .tar extension). When finished it should look like the folder shown below.

Using Finder (or command line) copy the contents of the folder to a new folder in your home directory, ~/frc2019 as shown below.



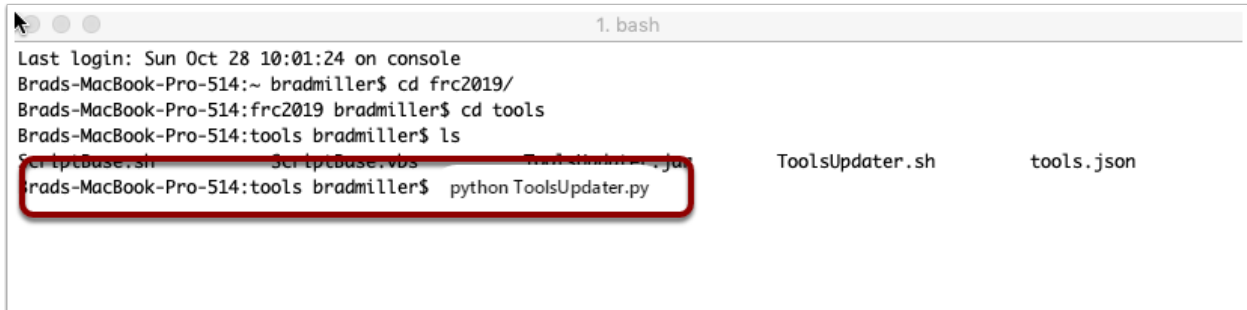


## 6.3 Run the ToolsUpdater.py script

To update all the additional tools WPILib tools, open a terminal window and change directory to `~/frc2019/tools` and run the script `ToolsUpdater.py` with the command:

```
python ToolsUpdater.py
```

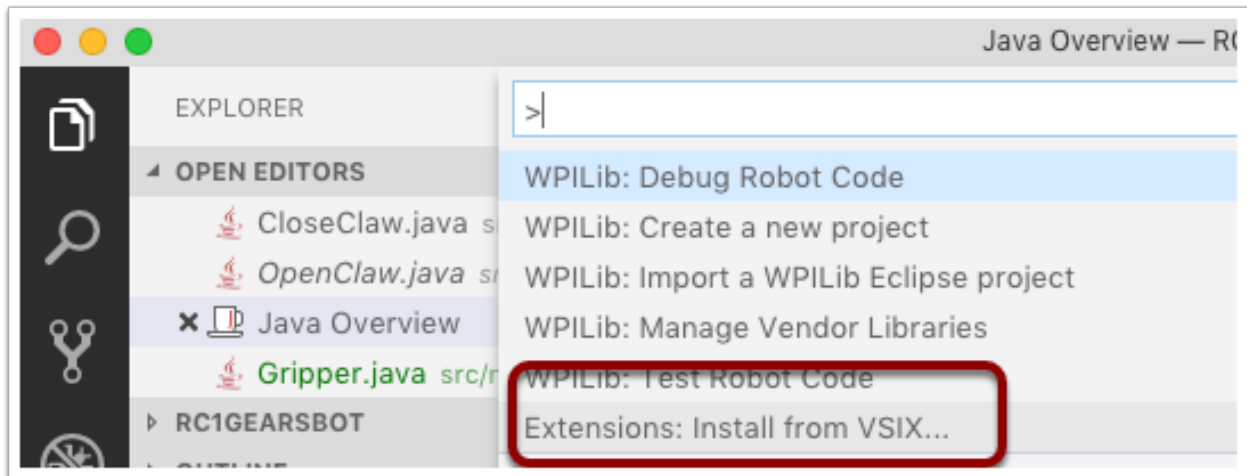
This should populate the tools directory with all of the WPILib tools (Shuffleboard, Robot Builder, PathWeaver, etc.)



```
1. bash
Last login: Sun Oct 28 10:01:24 on console
Brads-MacBook-Pro-514:~ bradmiller$ cd frc2019/
Brads-MacBook-Pro-514:frc2019 bradmiller$ cd tools
Brads-MacBook-Pro-514:tools bradmiller$ ls
ScriptBase.sh      ScriptBase.vbs      ToolsUpdater.py     ToolsUpdater.sh     tools.json
Brads-MacBook-Pro-514:tools bradmiller$ python ToolsUpdater.py
```

## 6.4 Installing the extensions for WPILib development

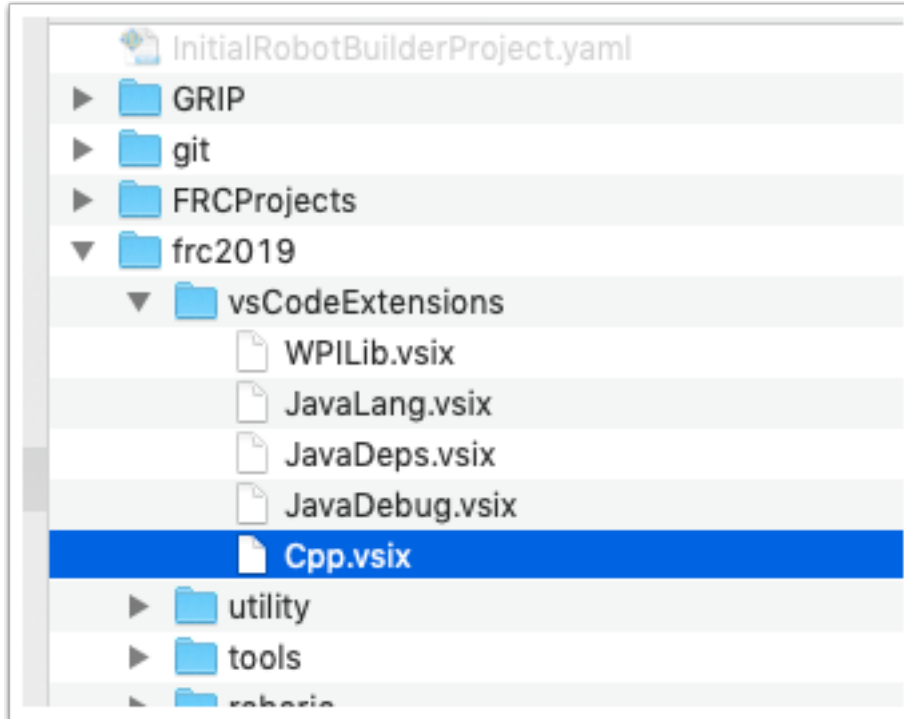
Before using VSCode for WPILib development there are a number of extensions that need to be installed. Start up VSCode and type the shortcut `Cmd-Shift-P` to bring up the list of commands available. Start typing “Install from VSIX” into the search box. Choose that command. In the file selection box select `Cpp.vsix`.



You should see a message confirming the install and asking to reload vscode. Click the reload button then repeat the vsix installation for the rest of the vsix files in this order:

1. Cpp.vsix
2. JavaLang.vsix
3. JavaDeps.vsix
4. JavaDebug.vsix
5. WPILib.vsix

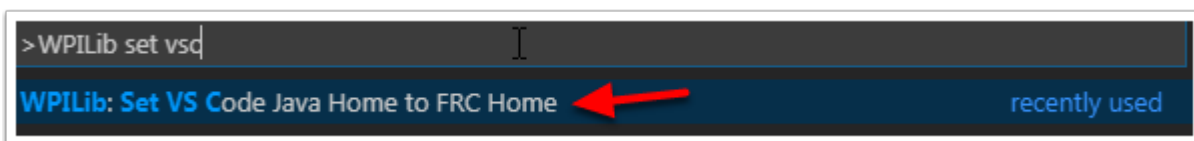




## 6.5 Setting up VSCode to use Java 11

The WPILib installation includes a JDK, however you need to point VS Code at where it is. To do this:

- 1) Open VS Code
- 2) Press Ctrl + Shift + P and type WPILib or click on the WPILib icon in the top right to open the WPILib Command Palette
- 3) Begin typing Set VS Code Java Home to FRC Home and select that item from the dropdown





### 7.1 Installing VS Code

1. Download the Linux .deb file from [code.visualstudio.com](https://code.visualstudio.com)
2. Double-click on the .deb file in the file explorer
3. Click the “Install” button to install VSCode

### 7.2 Download the WPILib release

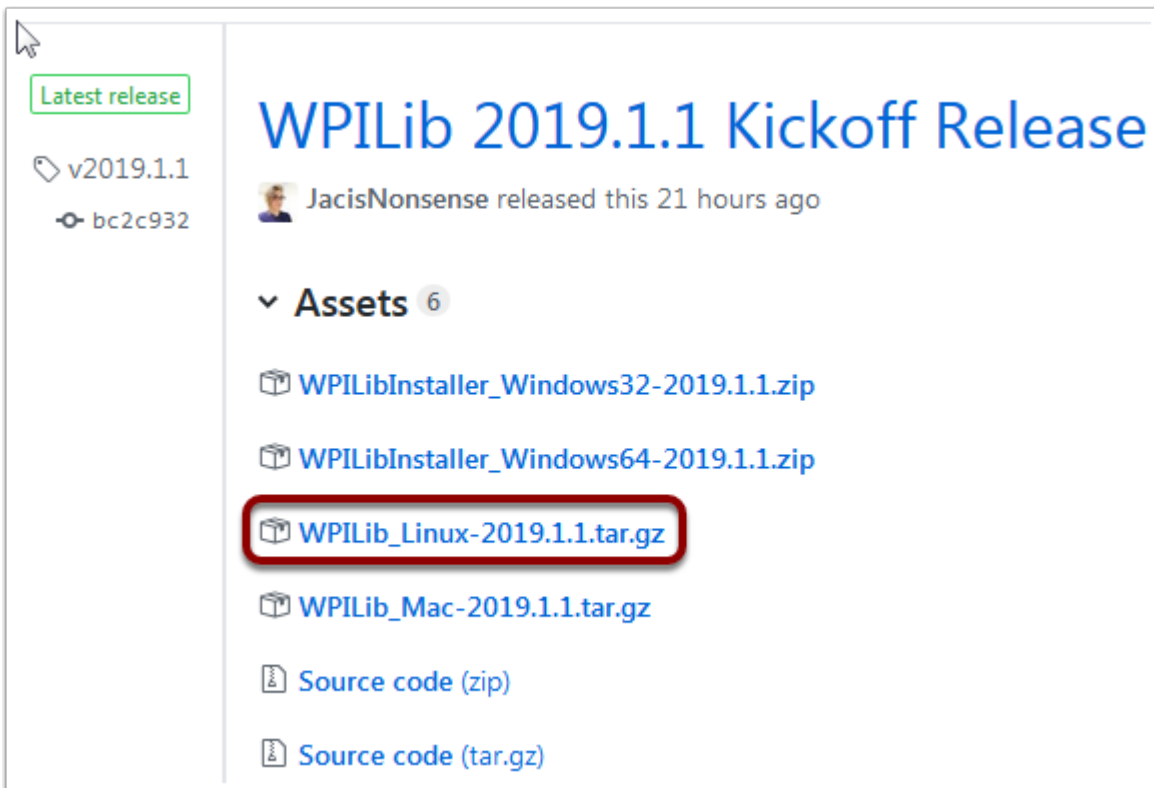
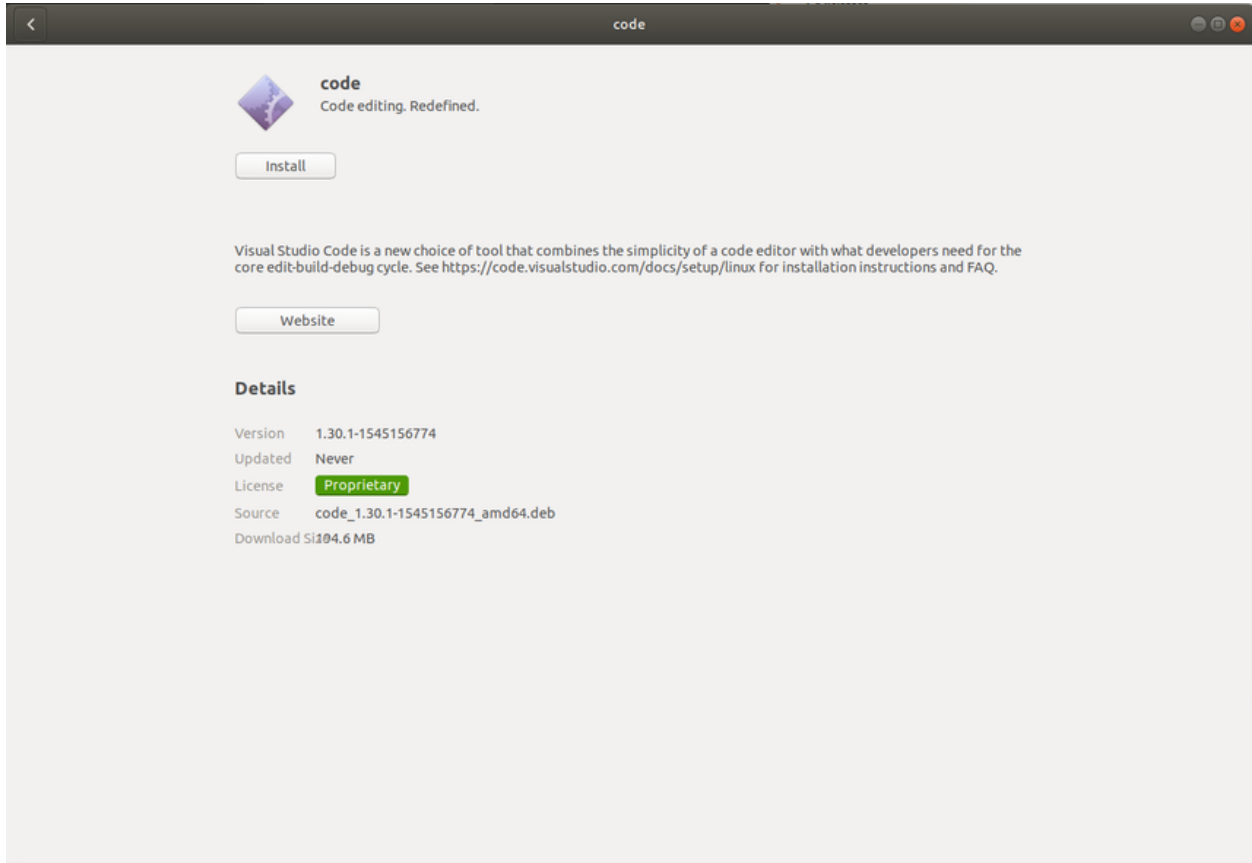
Download the latest Linux release from <https://github.com/wpilibsuite/allwpilib/releases> Right-click on the downloaded archive, click “Extract Here”

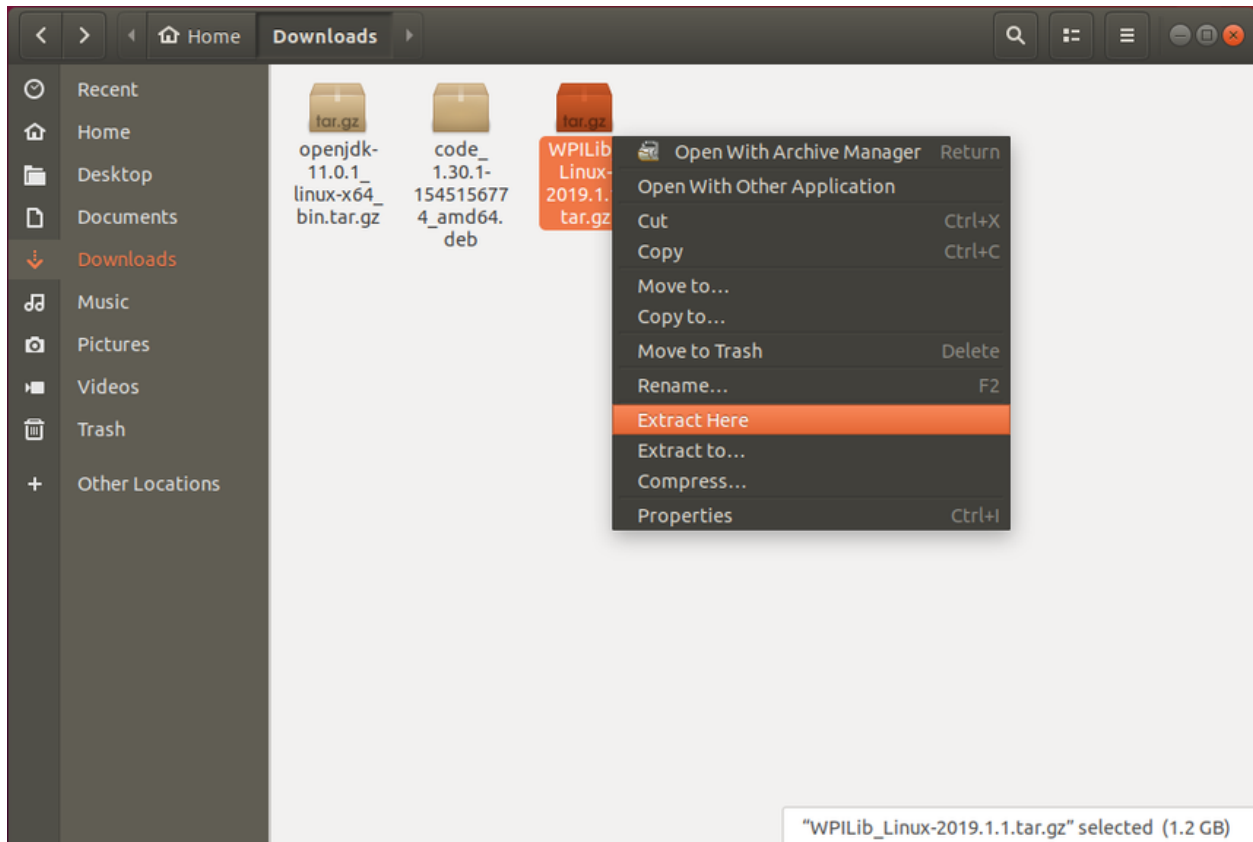
### 7.3 Moving to FRC2019

1. Create a directory in your home directory called frc2019 - either from the file manager or with `$ mkdir ~/frc2019`
2. Drag the contents of WPILIB\_Linux-2019.1.1 directory to ~/frc2019 or run `$ mv -v WPILib_Linux-2019.1.1/* ~/frc2019`

### 7.4 Running Tools Updater

To extract the WPILib tools (Dashboards, Robot Builder, etc.), run: `* $ cd ~/frc2019/tools * $ python3 ToolsUpdater.py`



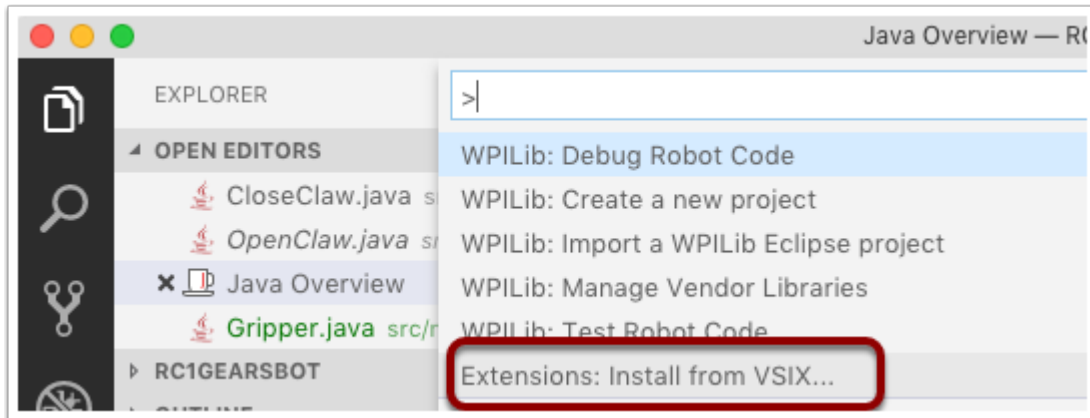


```
sam@sam-vb: ~/frc2019/tools
File Edit View Search Terminal Help
sam@sam-vb:~/Downloads$ cd ~/frc2019/tools
sam@sam-vb:~/frc2019/tools$ python3 ToolsUpdater.py

sam@sam-vb:~/frc2019/tools$
```

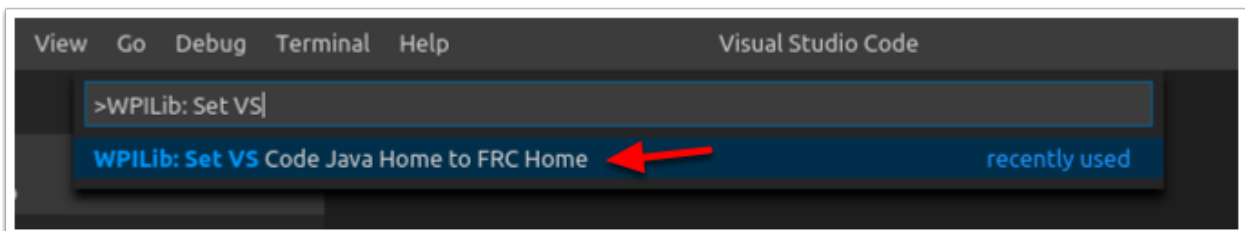
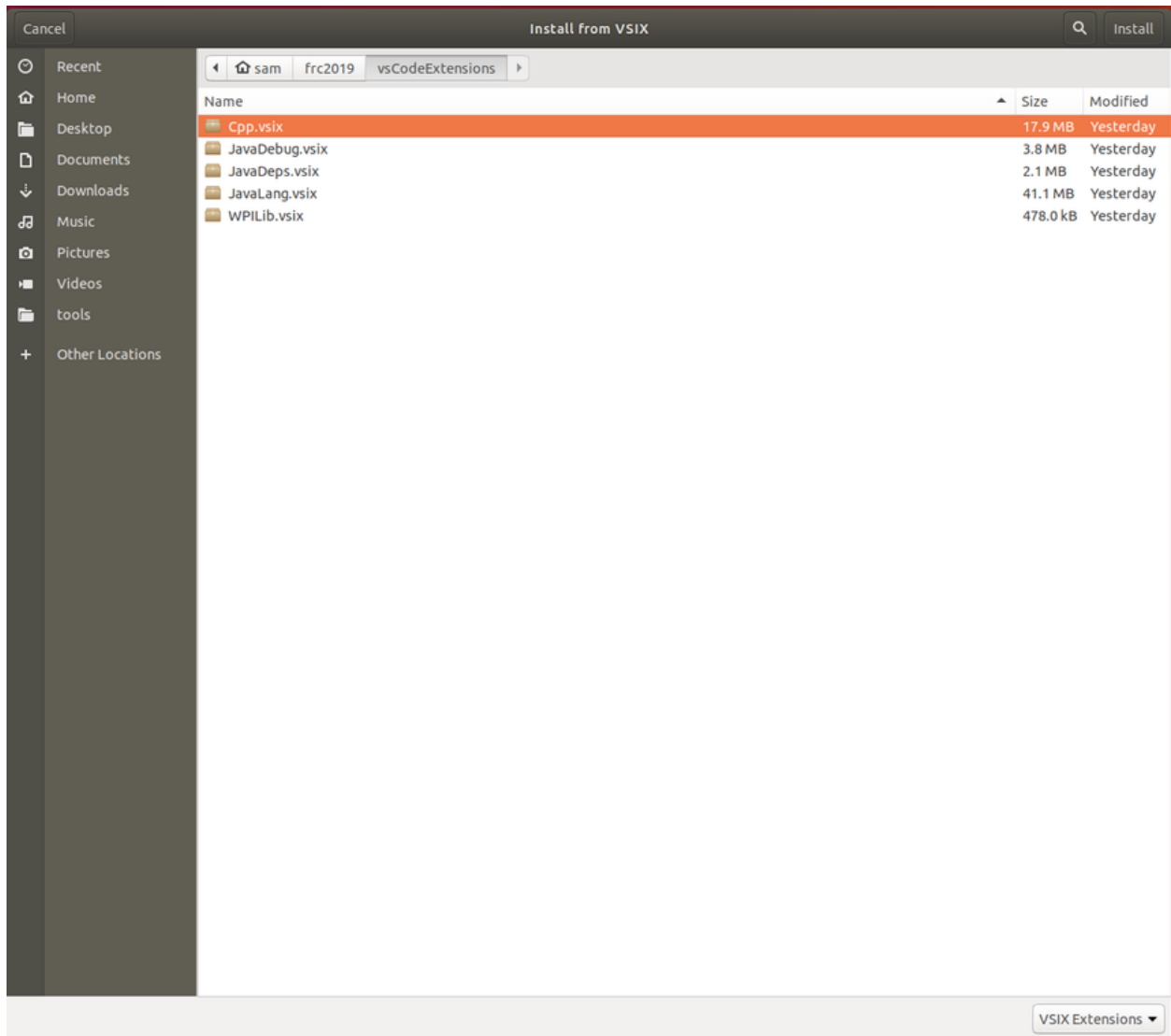
## 7.5 Installing the extensions for WPILib VS Code

1. Start VSCODE (\$ code or search “Visual Studio Code” in your application launcher)
2. **Control-Shift-P** to bring up the command palette, type “Install from VSIX”
3. Select “Extensions: Install from VSIX”
4. Navigate to `~/frc2019/vsCodeExtensions` and select `Cpp.vsix`
5. Repeat for `JavaLang.vsix`, `JavaDeps.vsix`, `JavaDebug.vsix`, and `WPILib.vsix` in that order



## 7.6 Setting up VSCode to use Java 11

The WPILib installation includes a JDK, however you need to point VS Code at where it is. To do this: 1. Open VS Code 2. Press **Ctrl-Shift-P** and type **WPILib** or click on the WPILib icon in the top right to open the WPILib Command Palette 3. Begin typing **Set VS Code Java Home to FRC Home** and select that item from the dropdown.







---

### Installing the FRC Update Suite

---

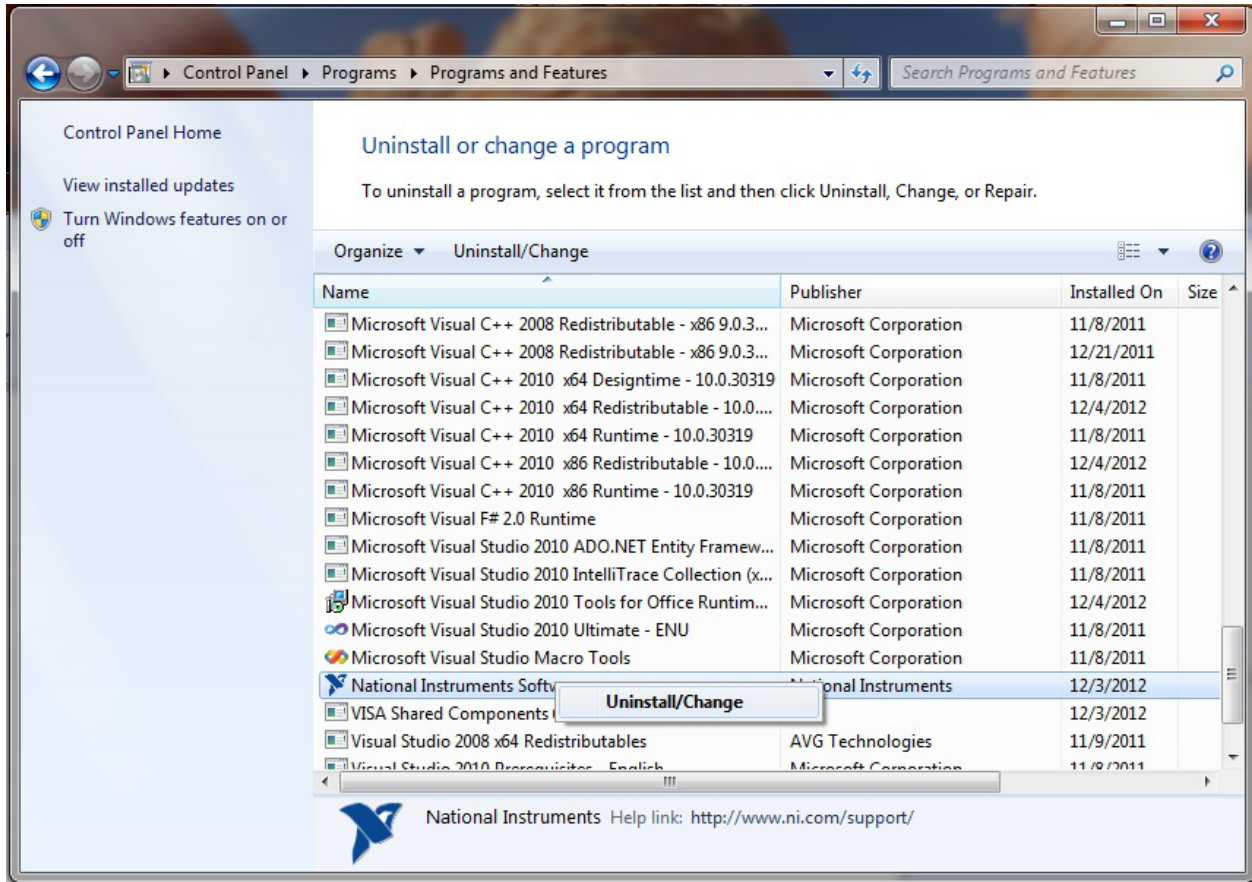
*The FRC Update Suite contains the following software components: LabVIEW Update, FRC Driver Station, and FRC Utilities. If an FRC LabVIEW installation is found, the LabVIEW Update will be installed or updated, otherwise this step will be skipped. The FRC Driver Station and FRC Utilities will always be installed or updated. The LabVIEW runtime components required for the driver station and utilities is included in this package. \*\*No components from the LabVIEW Merged Suite are required for running either the Driver Station or Utilities.\*\**

---

**Note:** Note: The Driver Station will only work on Windows 7, Windows 8, Windows 8.1, and Windows 10. It will not work on Windows XP.

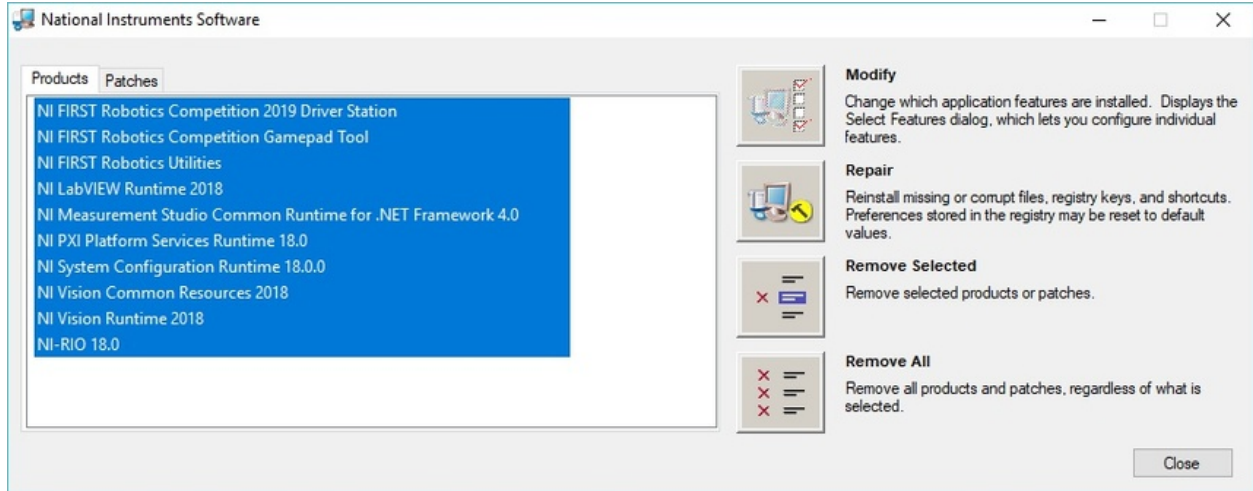
---

## 8.1 Uninstall Old Versions (Recommended)



**LabVIEW teams have already completed this step, do not repeat it.** Before installing the new version of the NI Update it is recommended to remove any old versions. The new version will likely properly overwrite the old version, but all testing has been done with FRC 2019 only. Make sure to back up any team code located in the `User:\LabVIEW\Data` directory before un-installing. Then click **Start >> Control Panel >> Uninstall a Program**. Locate the entry labeled “National Instruments Software”, right-click on it and select **Uninstall/Change**.

## 8.1.1 Select Components to Uninstall



Click **Remove All** and follow any prompts to remove all previous NI products.

## 8.2 Downloading the Update

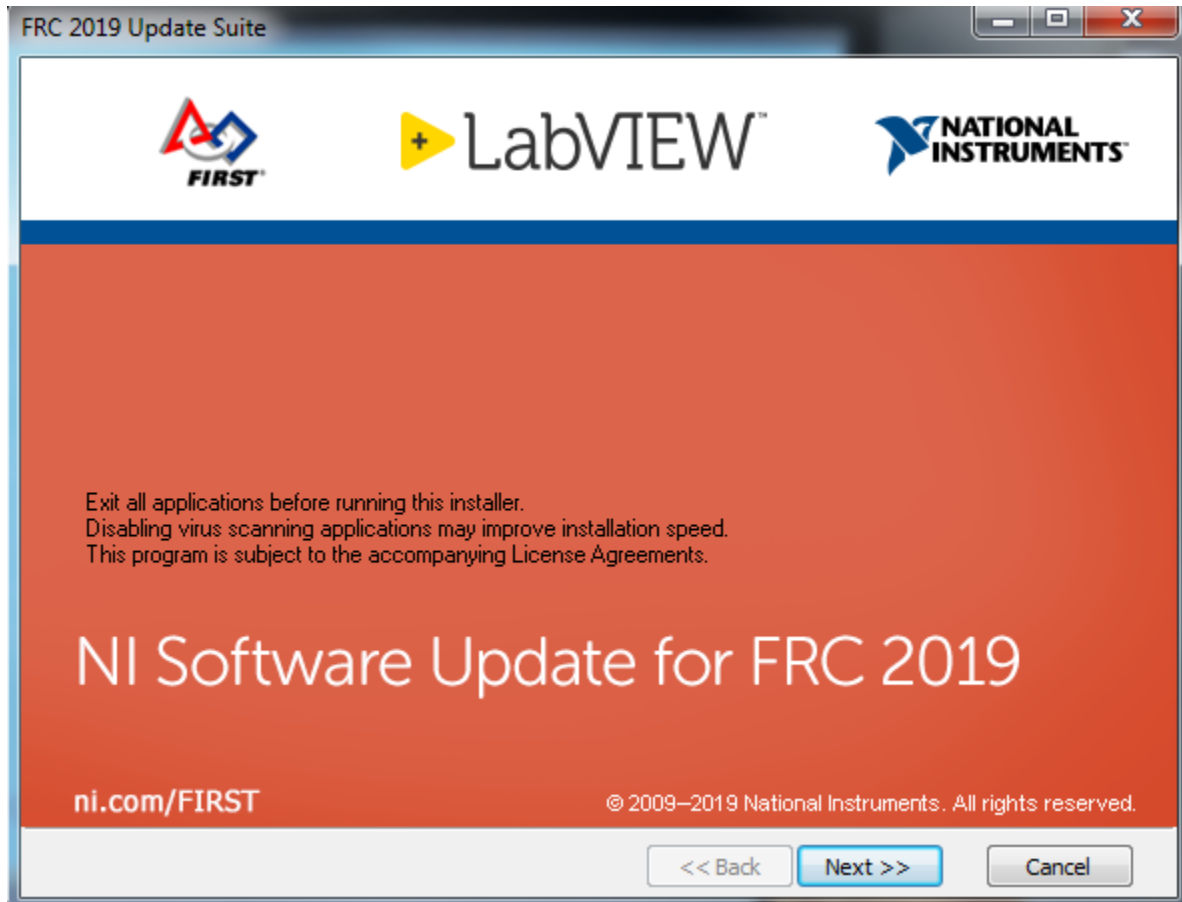
Download the update from <http://www.ni.com/download/first-robotics-software-2017/7904/en/>

**Note:** Note: This download will require the decryption key from the Kickoff broadcast.

## 8.3 .NET Framework 4.6.2

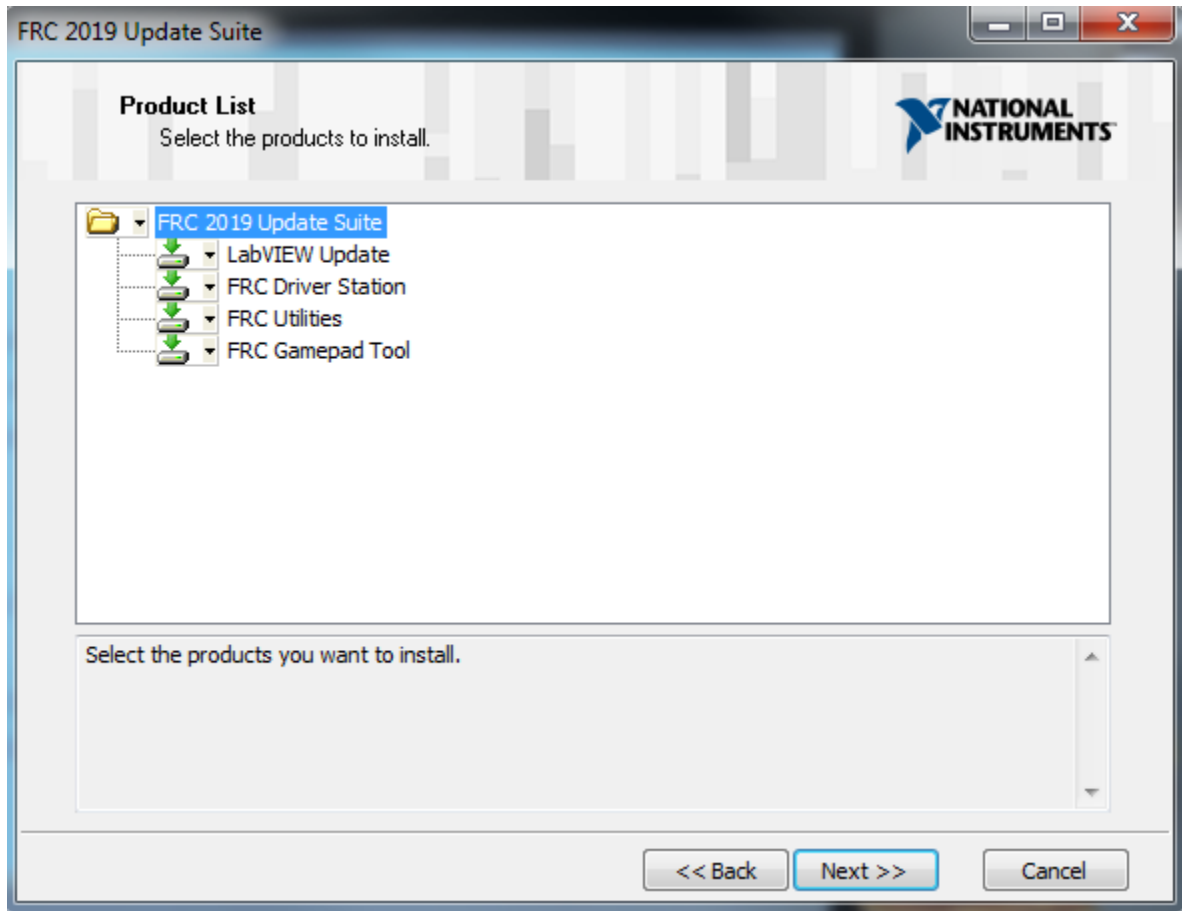
The Update installer may prompt that .NET Framework 4.6.2 needs to be updated or installed. Follow prompts on-screen to complete the installation, including rebooting if requested. Then resume the installation of the NI FRC Update, restarting the installer if necessary.

## 8.4 Welcome



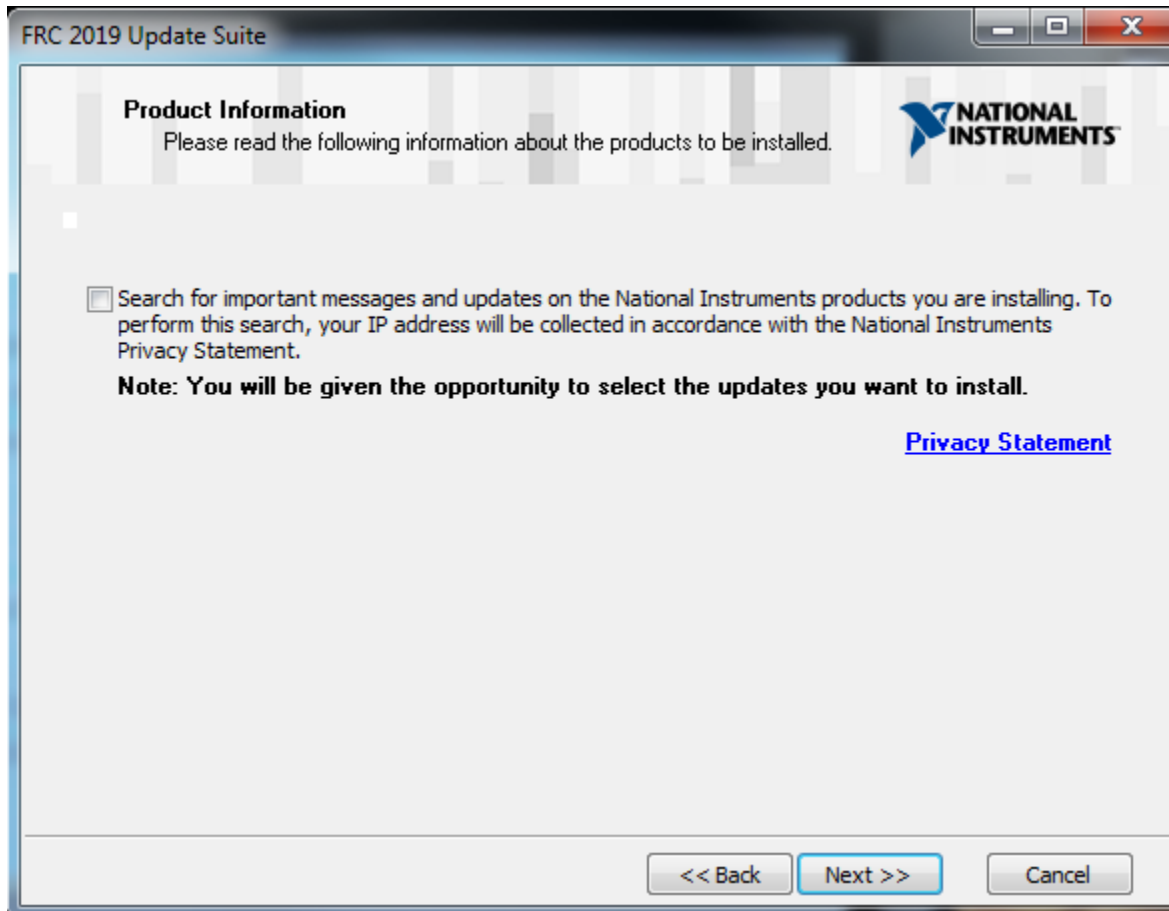
Right click on the downloaded zip file and select Extract All. If you downloaded the encrypted zip file, you will be prompted for the encryption key which will be released at Kickoff. Open the extracted folder and any subfolders until you reach the folder containing “setup” (may say “setup.exe” on some machines). Double click on the setup icon to launch the installer. Click “Yes” if a Windows Security prompt appears. Click “Next” on the splash screen that appears.

## 8.5 Product List



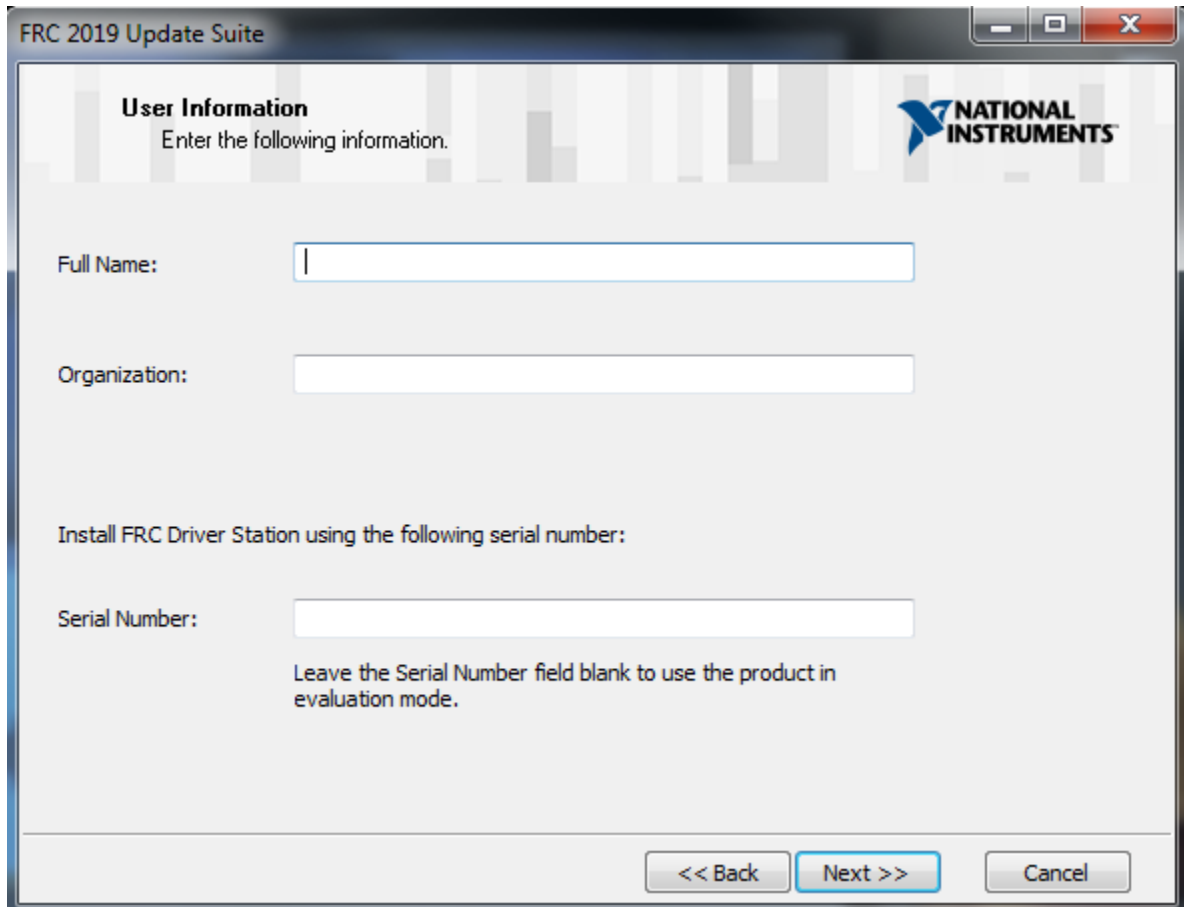
Click “Next”. There is no need to de-select “LabVIEW Update” for C++ or Java teams, if you do not have the base LabVIEW installation (because you are not programming in LabVIEW) this installation will be skipped automatically.

## 8.6 Product Information



Un-check the box, then Click “Next”.

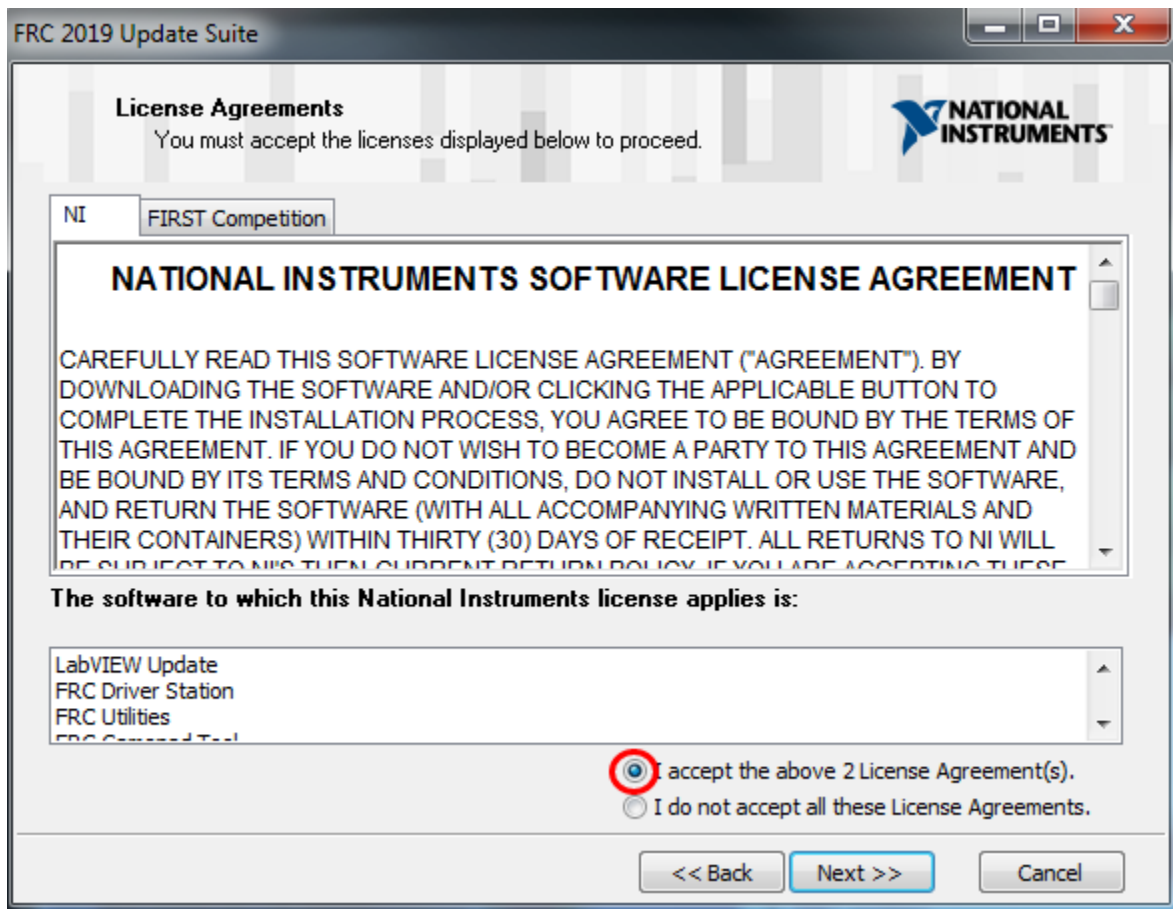
## 8.7 User Information



The screenshot shows a Windows-style dialog box titled "FRC 2019 Update Suite". The dialog has a header area with the title "User Information" and the instruction "Enter the following information." on the left, and the "NATIONAL INSTRUMENTS" logo on the right. Below the header, there are three input fields: "Full Name:" with a text box containing a cursor, "Organization:" with an empty text box, and "Serial Number:" with an empty text box. Below the "Serial Number" field, there is a note: "Leave the Serial Number field blank to use the product in evaluation mode." At the bottom of the dialog, there are three buttons: "<< Back", "Next >>" (which is highlighted with a blue border), and "Cancel".

Enter name and organization and the serial number from the Team Registration System (in the Password/Voucher code section) then click **Next**

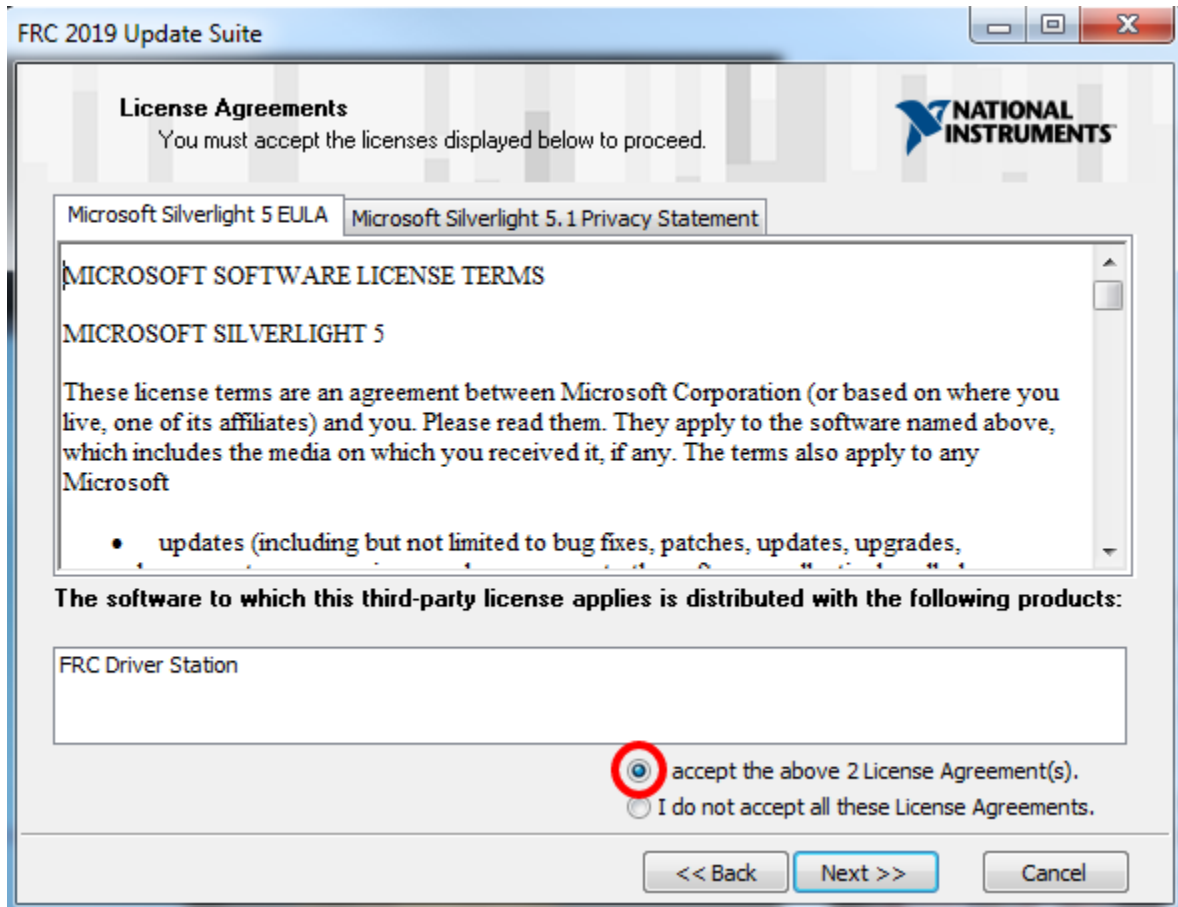
## 8.8 License Agreements



Select "I accept..." then click "Next"



## 8.9 License Agreements Page 2



Select “I accept...” then click “Next” If you see a screen asking to disable Windows Fast Startup, leave it at the recommended option (disable Fast Startup) and click Next. If you see a screen talking about Windows Firewall, click Next.

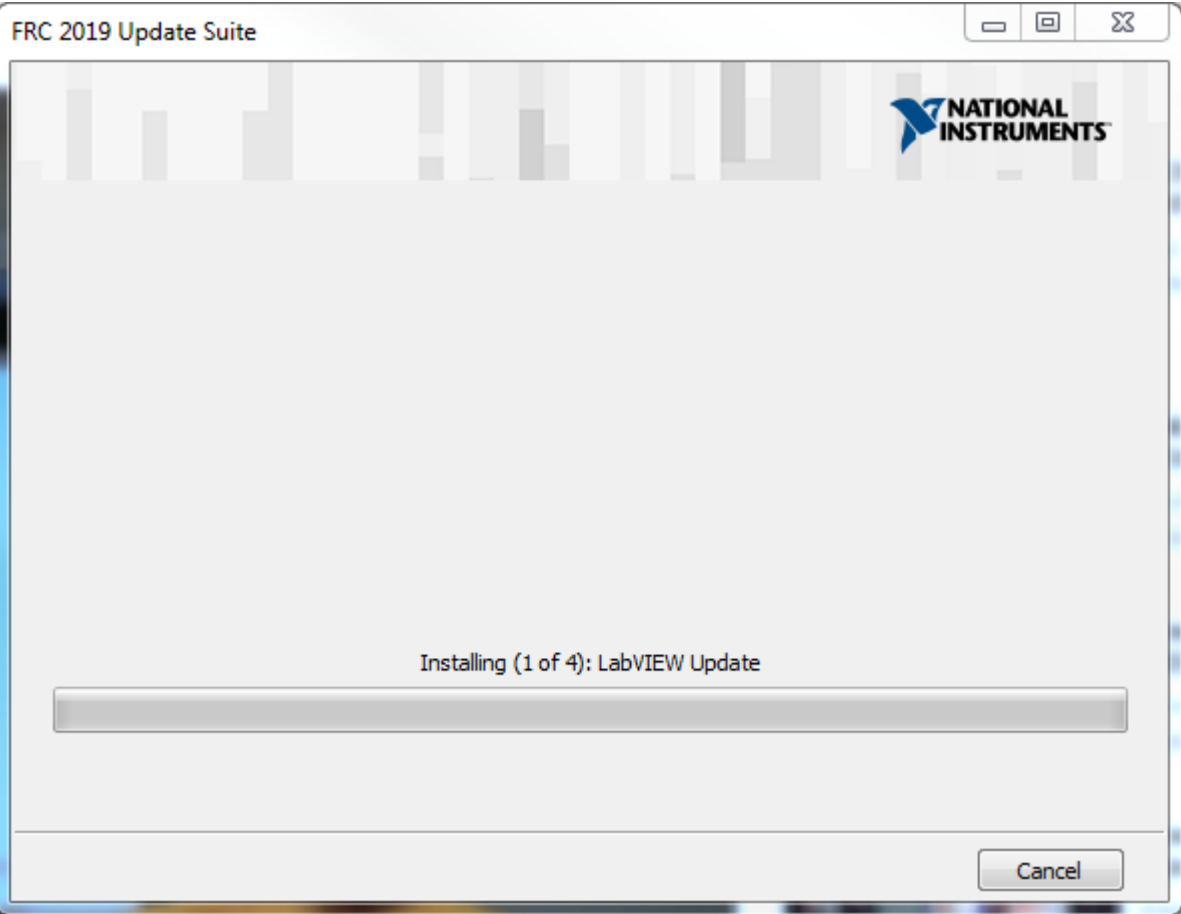


Fig. 1: Summary Progress

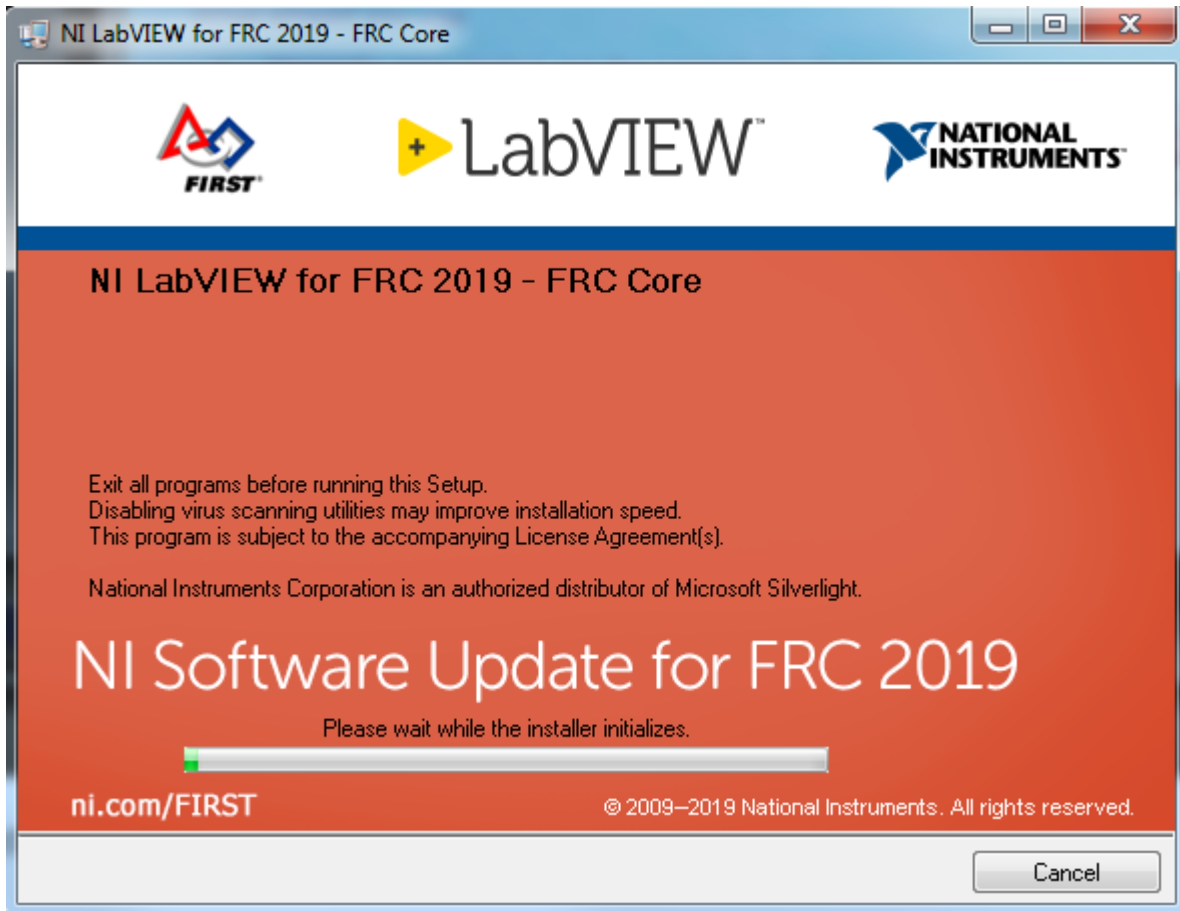
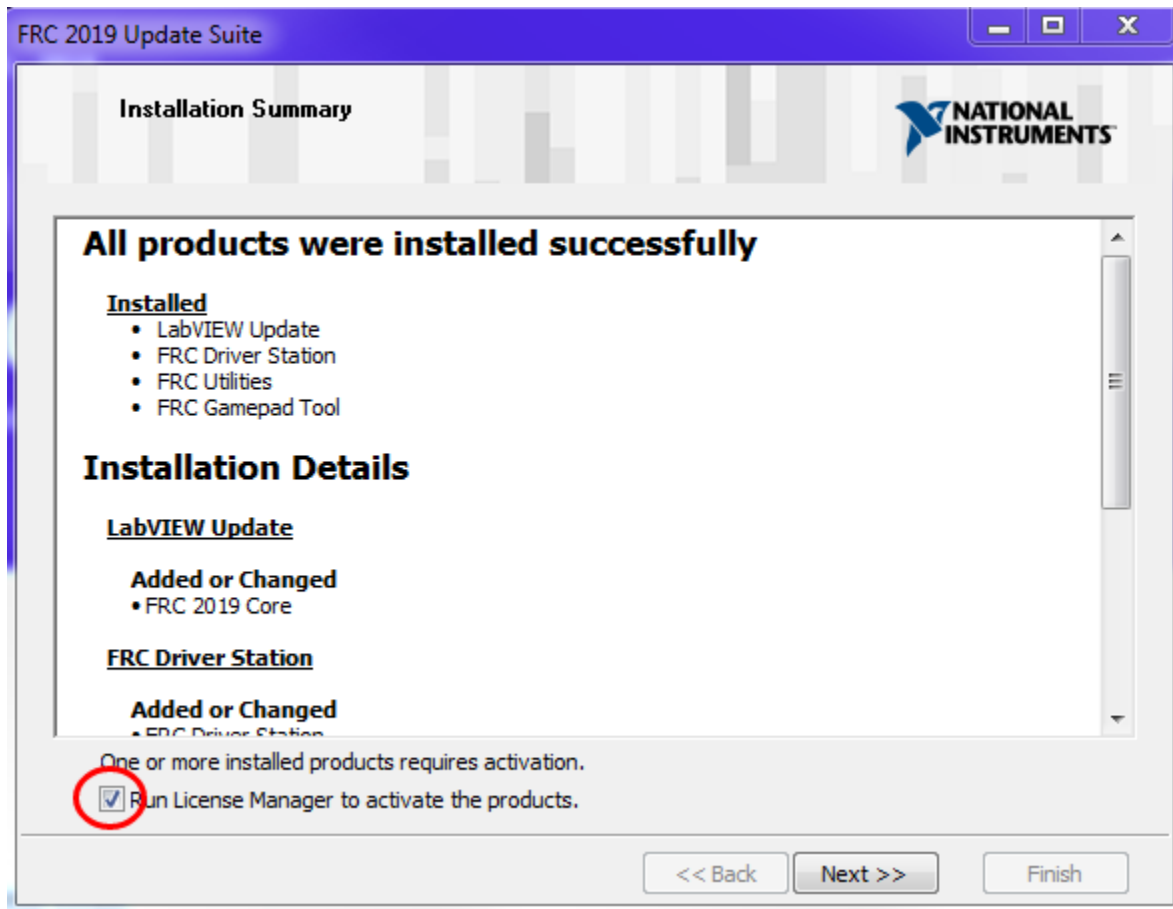


Fig. 2: Detail Progress

## 8.10 Summary Progress

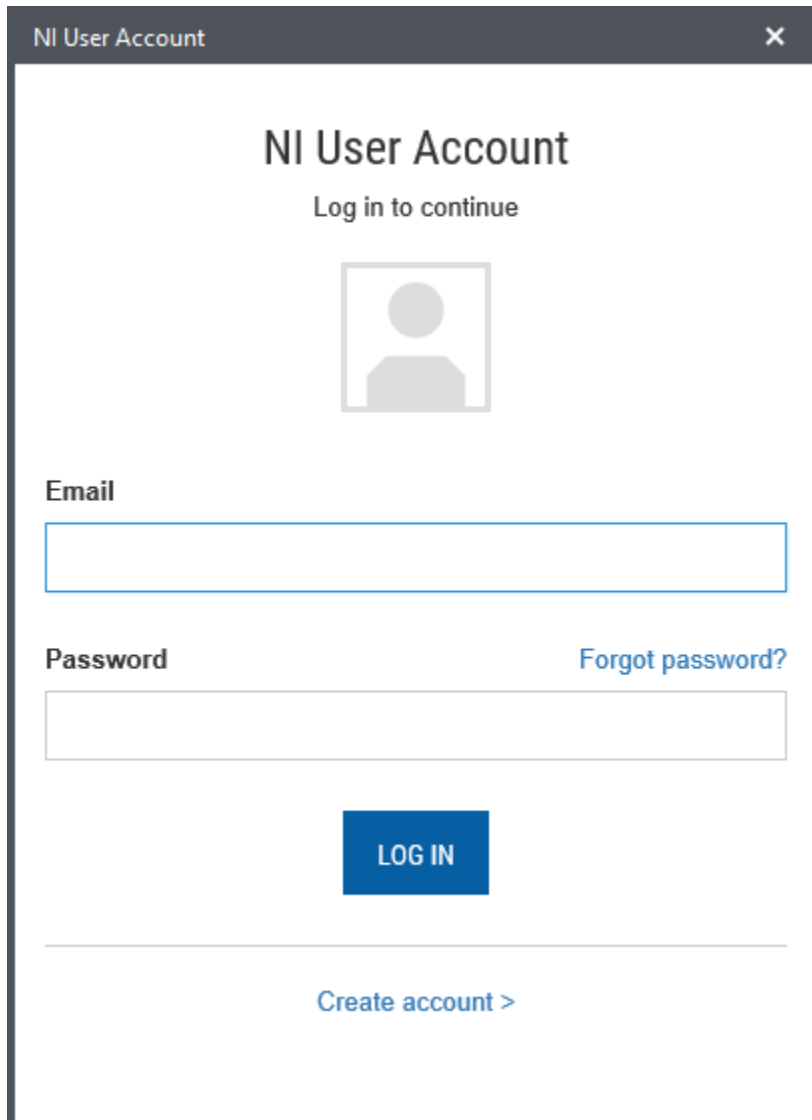
## 8.11 Detail Progress

## 8.12 Installation Summary



Make sure the box is checked to Run License Manager... then click Next or Finish

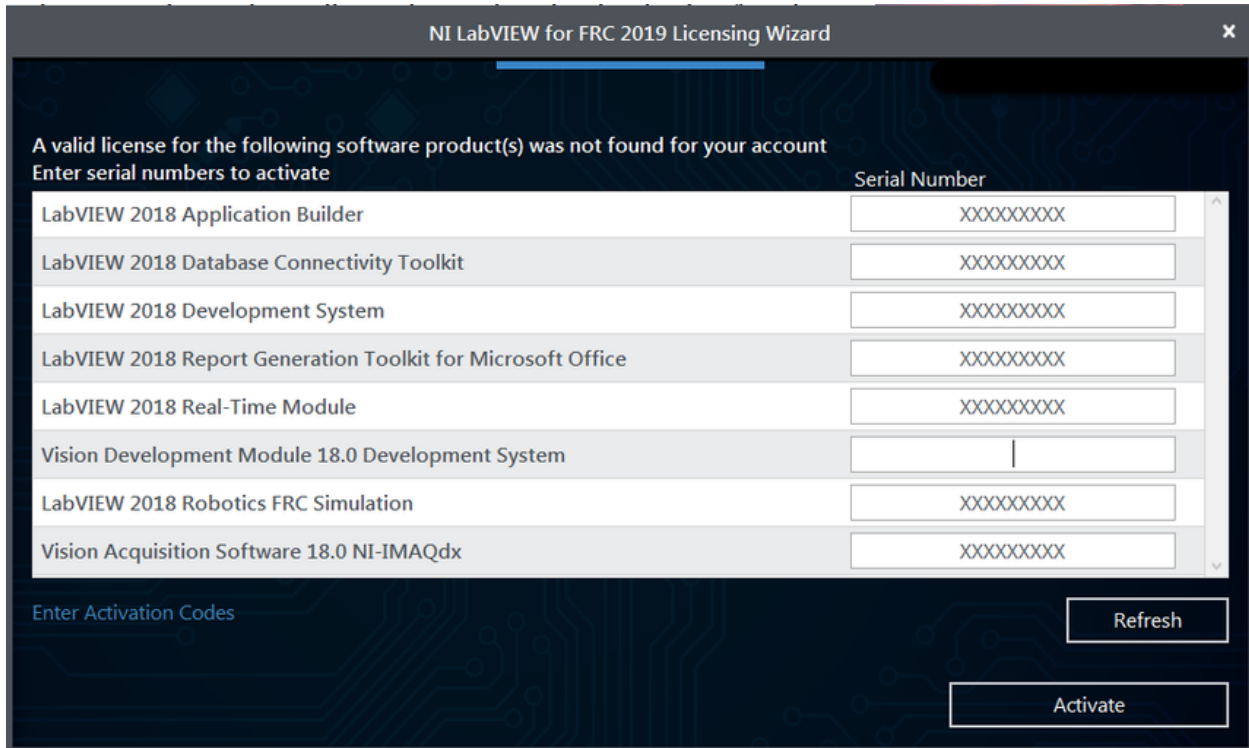
## 8.13 NI Activation Wizard



The screenshot shows a web browser window titled "NI User Account" with a close button in the top right corner. The main content area has the title "NI User Account" and the instruction "Log in to continue". Below this is a placeholder for a profile picture, represented by a square icon with a circle and a person silhouette. There are two input fields: "Email" and "Password". To the right of the password field is a link "Forgot password?". A blue "LOG IN" button is centered below the password field. At the bottom, there is a horizontal line and a link "Create account >".

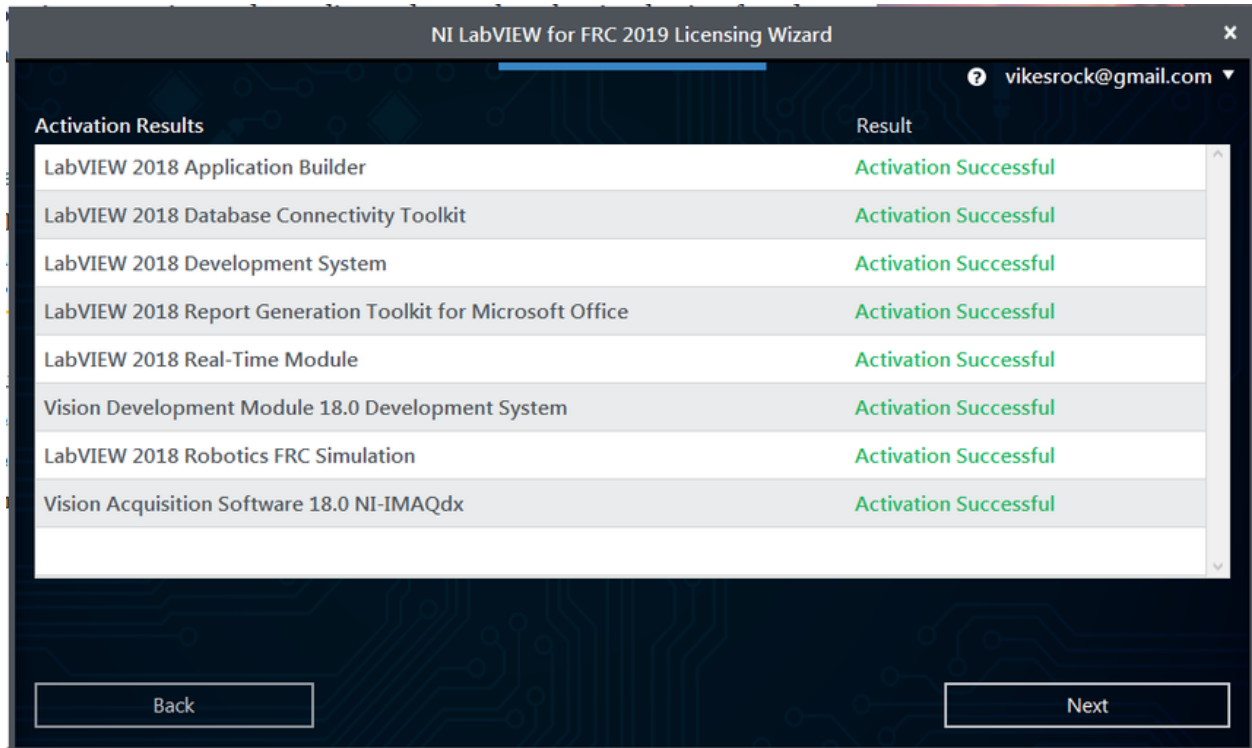
Log into your ni.com account. If you don't have an account, select 'Create account' to create a free account.

## 8.14 NI Activation Wizard (2)



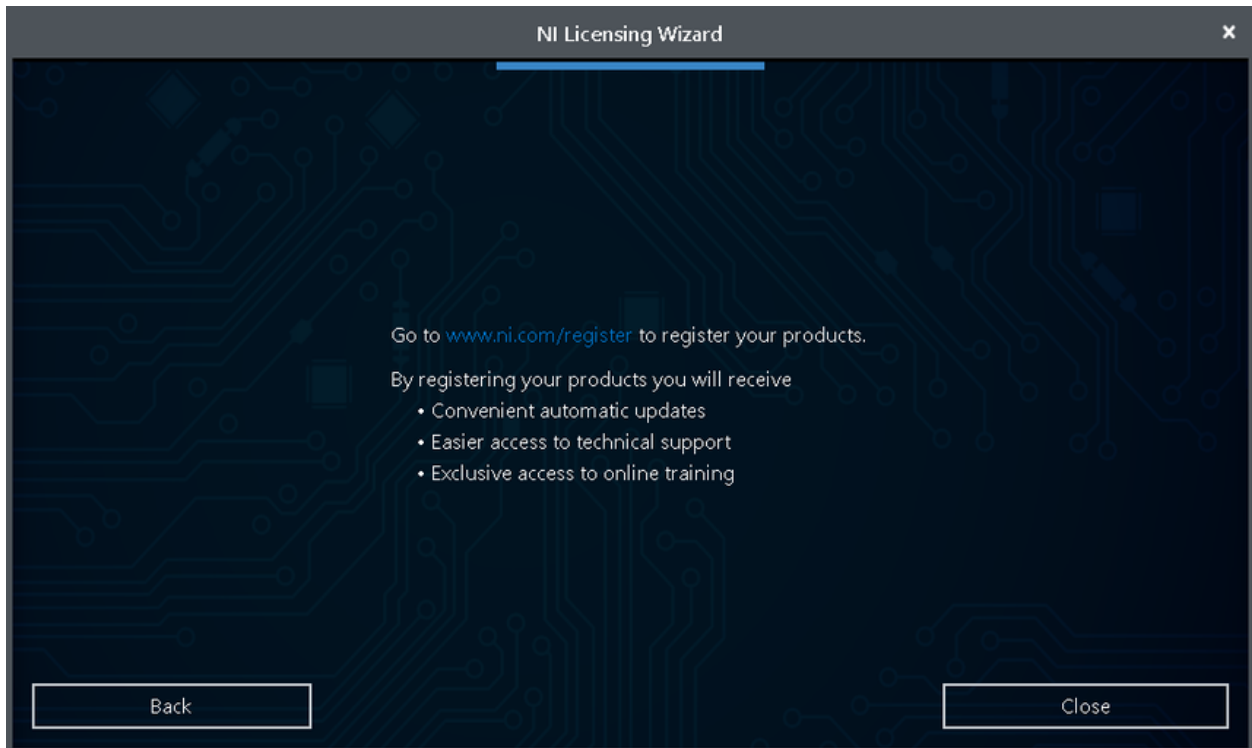
The serial number you entered at the “User Information” screen should appear in all of the text boxes, if it doesn’t, enter it now. Click “Activate”. Note: If this is the first time activating the 2019 software on this account, you will see the message shown above about a valid license not being found. You can ignore this.

## 8.15 NI Activation Wizard (3)



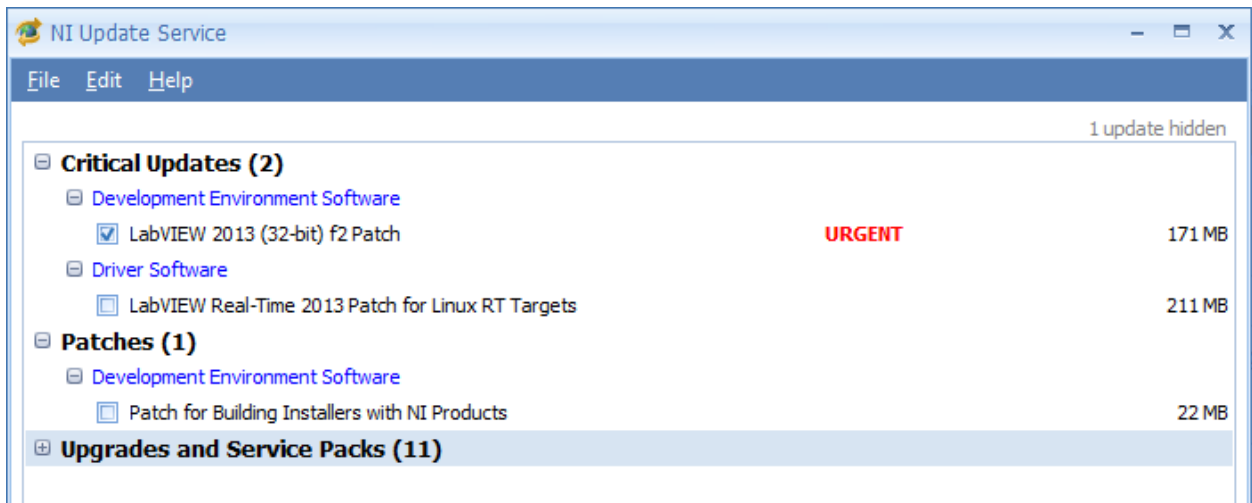
If your products activate successfully, an “Activation Successful” message will appear. If the serial number was incorrect, it will give you a text box and you can re-enter the number and select “Try Again”. If everything activated successfully, click “Next”.

## 8.16 NI Activation Wizard (4)



Click “Close”.

## 8.17 NI Update Service



On occasion you may see alerts from the NI Update Service about patches to LabVIEW. It is not recommended to install these patches. **FRC will communicate any recommended updates through our usual channels** (Frank’s Blog, Team Updates or E-mail Blasts).



---

## 3rd Party Libraries

---

A number of software components were broken out of WPILib starting in 2017 and are now maintained by the third parties who produce the hardware. See this blog for more details.

### 9.1 Libraries

**Warning:** Note: These are not links to directly plug in to the VS Code -> Install New Libraries (online) feature. Click these links to visit the vendor site to see whether they offer online installers, offline installers, or both.

Most vendors have not yet published final releases for 2019, keep an eye on the vendor pages for 2019 software releases.

Analog Devices ADIS16448 IMU - Driver for ADIS16448 IMU. More info on <https://wiki.analog.com/first>

Analog Devices ADIS16470 IMU - Driver for ADIS16470 IMU. More info on <https://wiki.analog.com/first>

CTRE Phoenix Toolsuite - Contains TalonSRX/Victor SPX Libraries and Phoenix Tuner program for configuring CTRE CAN devices

Digilent - DMC-60C library

Kauai Labs - Libraries for NavX-MXP, NavX-Micro, and Sensor Fusion

Mindsensors Libraries - Contains libraries for SD540C and CANLight

Rev Robotics - SPARK MAX Library

Scanse Sweep - C/Java Libraries for Scansense Sweep LIDAR (packaged by Peter Johnson)

## 9.2 The Mechanism

In support of this effort NI (for LabVIEW) and FIRST/WPI (for C++/Java) have developed mechanisms that should make it easy for vendors to plug their code into the WPILib software and for teams to use that code once it has been installed. A brief description of how the system works for each language can be found below.

### 9.2.1 The Mechanism - LabVIEW

For LabVIEW teams, you may notice a few new Third Party items on various palettes (specifically, one in Actuators, one in Actuators->Motor Control labeled “CAN Motor”, and one in “Sensors”). These correspond to folders in `Program Files/National Instruments/LabVIEW 2016/vi.lib/Rock Robotics/WPI/Third Party`. For a library to insert VIs in these palettes, they simply make a subfolder in one of these three Third Party folders containing their VIs and they will be added automatically. To control the appearance of the palette (have some VI's not show up, set the Icon for the folder, etc.) there is a process to create a `dir.mnu` file for your directory. We will be working on documenting that process shortly.

To use installed Third Party libraries, simply locate the VIs in one of these 3 locations and drag them into your project as you would with any other VI.

### 9.2.2 The Mechanism - C++/Java

For C++ and Java a JSON file describing the vendor library is installed on your system to `~home/frcYYYY/vendordeps` (`~home = C:/Users/Public` on Windows). This can either be done by an offline installer or the file can be fetched from an online location using the menu item in VSCode. This file is then used from VS Code to add to the library to each individual project. Vendor library information is managed on a per-project basis to make sure that a project is always pointing to a consistent version of a given vendor library. The libraries themselves are placed in the Maven cache at `C:/Users/Public/frcYYYY/maven`. Vendors can place a local copy here with an offline installer (recommended) or require users to be online for an initial build to fetch the library from a remote Maven location.

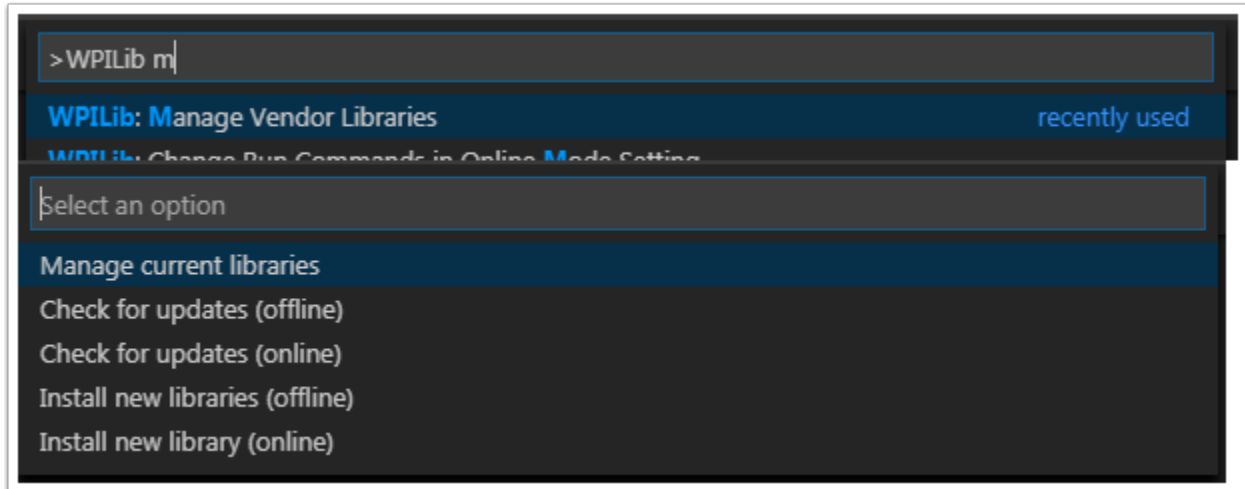
The JSON file allows specification of complex libraries with multiple components (C++, Java, JNI, etc.) and also helps handle some complexities related to simulation. Vendors choosing to provide a remote URL in the JSON also enable users to check for updates from within VS Code.

---

**Note:** Note: The vendor JSON files are actually processed by GradleRIO once they are in your projects `vendordeps` folder. If you are using another IDE, you will need to manually create a “`vendordeps`” folder in your project and copy any desired vendor JSON files from the “`frcYYYY`” folder (where they should be placed by an offline installer) or download them directly from the vendor and place them into the folder in the project.\*\*

---

### 9.2.3 Adding an offline-installed Library



To add a vendor library that has been installed by an offline installer, press **Ctrl+Shift+P** and type WPILib or click on the WPILib icon in the top right to open the WPILib Command Palette and begin typing **Manage Vendor Libraries**, then select it from the menu. Select the option to **Install new libraries (offline)**.



Select the desired libraries to add to the project by checking the box next to each, then click OK. The JSON file will be copied to the **vendordeps** folder in the project, adding the library as a dependency to the project.

### 9.2.4 Checking for Updates (offline)

Remember: Dependencies are now version managed and done on a per-project bases. Even if you have installed an updated library using an offline installer, you will need to Manage Vendor Libraries and select **Check for updates (offline)** for each project you wish to update.

### 9.2.5 Checking for Updates (online)

Part of the JSON file that vendors may optionally populate is an online update location. If a library has an appropriate location specified, running **Check for updates (online)** will check if a newer version of the library is available from the remote location.

### 9.2.6 Removing a library dependency

To remove a library dependency from a project, select **Manage Current Libraries** from the **Manage Vendor Libraries** menu, check the box for any libraries to uninstall and click OK. These libraries will be removed as dependencies from the project.



# CHAPTER 10

---

## Imaging your roboRIO

---

*Before imaging your roboRIO, you must have completed installation of the FRC Update Suite. You also must have the roboRIO power properly wired to the Power Distribution Panel. **\*\*Make sure the power wires to the roboRIO are secure and that the connector is secure firmly to the roboRIO (4 total screws to check).\*\****

### 10.1 Configuring the roboRIO

The roboRIO Imaging Tool will be used to image your roboRIO with the latest software.

#### 10.1.1 USB Connection



Connect a USB cable from the roboRIO USB Device port to the PC. This requires a USB Type A male (standard PC end) to Type B male cable (square with 2 cut corners), most commonly found as a printer USB cable.

---

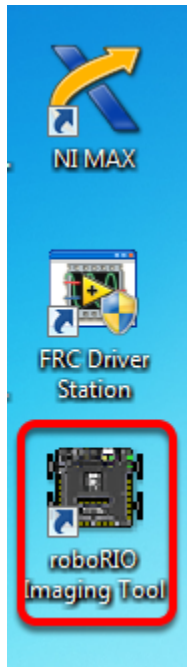
**Note:** Note: The roboRIO should only be imaged via the USB connection. It is not recommended to attempt imaging using the Ethernet connection.

---

### 10.1.2 Driver Installation

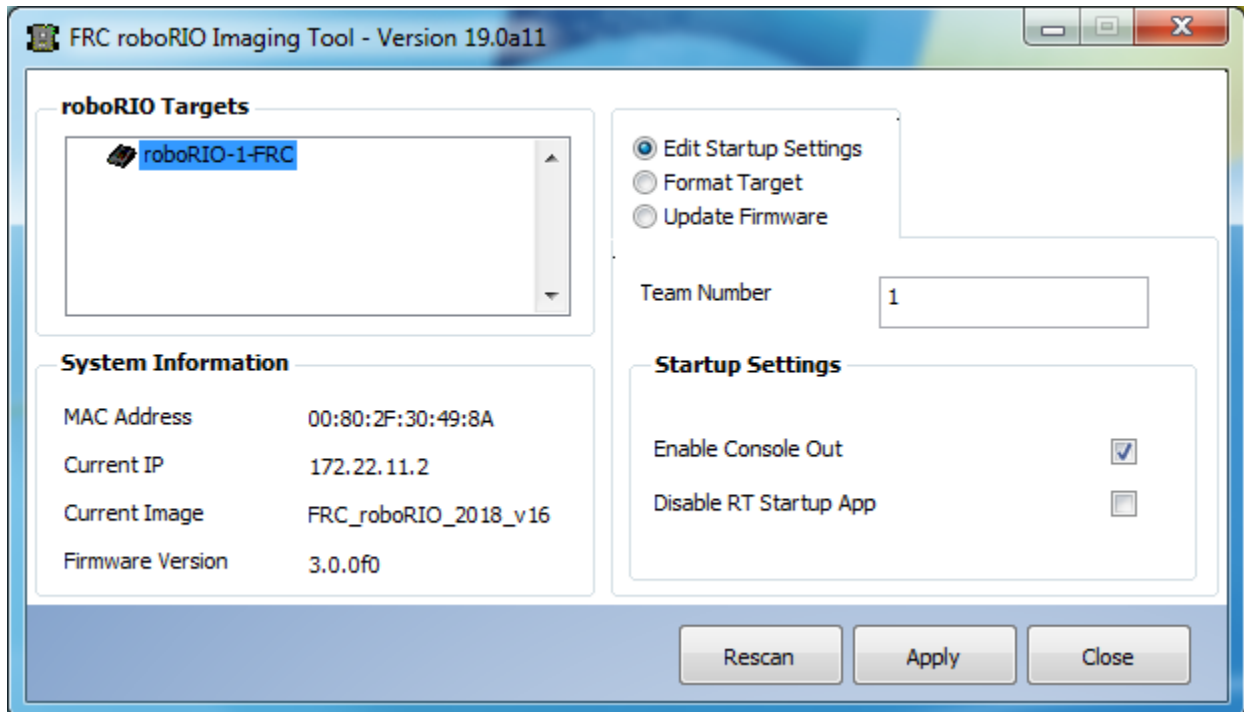
The device driver should install automatically. If you see a “New Device” pop-up in the bottom right of the screen, wait for the driver install to complete before continuing.

## 10.2 Launching the Imaging Tool



The roboRIO imaging tool and latest image are installed with the NI Update Suite. Launch the imaging tool by double clicking on the shortcut on the Desktop. If you have difficulties imaging your roboRIO, you may need to try right-clicking on the icon and selecting Run as Administrator instead.

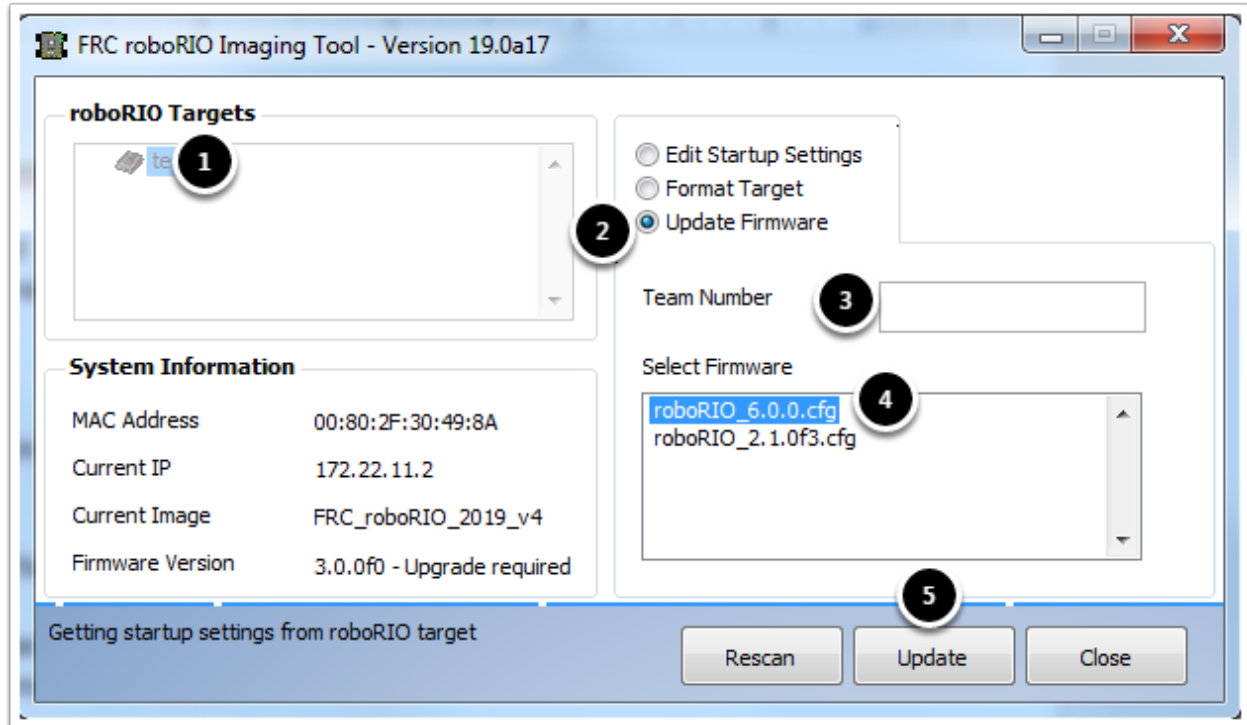
## 10.3 roboRIO Imaging Tool



After launching, the roboRIO Imaging Tool will scan for available roboRIOs and indicate any found in the top left box. The bottom left box will show information and settings for the roboRIO currently selected. The right hand pane contains controls for modifying the roboRIO settings:

- **Edit Startup Settings** - This option is used when you want to configure the startup settings of the roboRIO (the settings in the right pane), without imaging the roboRIO.
- **Format Target** - This option is used when you want to load a new image on the roboRIO (or reflash the existing image). This is the most common option.
- **Update Firmware** - This option is used to update the roboRIO firmware. For this season, the imaging tool will require roboRIO firmware to be version 5.0 or greater.

### 10.3.1 Updating Firmware



RoboRIO firmware must be at least v5.0 to work with the 2019 image. If your roboRIO is at least version 5.0 (as shown in the bottom left of the imaging tool) you do not need to update.

To update roboRIO firmware:

1. Make sure your roboRIO is selected in the top left pane.
2. Select Update Firmware in the top right pane
3. Enter a team number in the Team Number box
4. Select the latest firmware file in the bottom right
5. Click the **Update** button

## 10.4 Imaging the roboRIO

1. Make sure the roboRIO is selected in the top left pane
2. Select Format Target in the right pane
3. Enter your team number in the box
4. Select the latest image version in the box.
5. Click Reformat to begin the imaging process.



## 10.5 Imaging Progress

The imaging process will take approximately 3-10 minutes. A progress bar in the bottom left of the window will indicate progress.

## 10.6 Imaging Complete

When the imaging completes you should see the dialog above. Click Ok, then click the Close button at the bottom right to close the imaging tool. Reboot the roboRIO using the Reset button to have the new team number take effect.

---

**Note:** Note: The default CAN webdash functionality has been removed from the image (CAN devices will still work from robot code). You will need to use the tools provided by individual vendors to service their CAN devices.

---

## 10.7 Troubleshooting

If you are unable to image your roboRIO, troubleshooting steps include:

- Try running the roboRIO Imaging Tool as Administrator by right-clicking on the Desktop icon to launch it.
- Try accessing the roboRIO webpage with a web-browser at <http://172.22.11.2/> and/or verify that the NI network adapter appears in your list of Network Adapters in the Control Panel. If not, try re-installing the NI Update Suite or try a different PC.
- Make sure your firewall is turned off.
- Try a different PC
- Try booting the roboRIO into Safe Mode by pressing and holding the reset button for at least 5 seconds.



---

## Imaging your Classmate (Veteran Image Download)

---

This document describes the procedure for creating a bootable USB drive to restore the 2017 FRC image on a Classmate computer. Note that Veteran teams are not required to re-image their Classmates. If you do not wish to re-image your Classmate then you can start with the appropriate document for C++Java, LabVIEW, or DS only.

### 11.1 Prerequisites

1. E09, E11, E12, or E14 Classmate computer or Acer ES1 computer
2. 16GB or larger USB drive
3. 7-Zip software installed. Download here ([www.7-zip.org](http://www.7-zip.org)) As of the writing of this document, the current released version is 18.06 (2018-12-30)
4. RMprepUSB software installed. Download here (<http://www.rmrepusb.com/documents/release-2-0>) Scroll down the page and select the stable (Full) version's download link. As of the writing of this document, the current stable version is 2.1.741a

### 11.2 Download the Computer Image

Download the image from the FIRST FRC Driver Station System Image Portal(<https://frc-events.firstinspires.org/services/DSImages/2019>). There are several computer images available, one for each model. On the download site, select the option that matches your computer by clicking the button below the image. Due to the limited size of hard drive in the E09, it is supported with a DS/Utilities image only and does not have the IDEs for LabVIEW or C++/Java installed. All other images have the LabVIEW base installation already present.

**NOTE: These images only install the prerequisite core FRC software, it is still necessary to install the FRC specific updates. See the Update Software step for more information.**

## 11.3 Preparation

1. Place the image file downloaded from the site to a folder on your root drive (e.g. C:2016\_Image)
2. Connect 16GB or larger USB Flash drive to the PC to use as the new restoration drive.

## 11.4 RMPrep

Start/Run RMPrepUSB

Select USB Drive

### 11.4.1 Set Partition Size

Set Partition Size to MAX

### 11.4.2 Set Volume Label

Set Volume Label to Generic

### 11.4.3 Set Bootloader Option

Select Bootloader Option “WinPE v2/WinPE v3/Vista/Win7 bootable”

### 11.4.4 Select Filesystem

Select NTFS Filesystem

### 11.4.5 Copy OS Files Option

Ensure the “Copy OS files after Format” box is checked

### 11.4.6 Locate Image

Select the “Choose Folder/File” button

### 11.4.7 Copy Files Dialog

Choose “No” and select your .7z image

### 11.4.8 Prepare Drive

All configuration settings are now complete. Select “Prepare Drive” to begin the process

### 11.4.9 Confirmation Dialog 1

Click “OK” to execute the command on the selected USB Flash drive. A Command Prompt will open showing the progress

### 11.4.10 Confirmation Dialog 2

Click “OK” to format the USB drive

**NOTE: ALL DATA ON THE DRIVE WILL BE ERASED!**

### 11.4.11 Decryption

**Note: If you are using an encrypted version of the image downloaded before kickoff you will be prompted to enter the decryption key found at the end of the Kickoff video.**

### 11.4.12 Copy Complete

Once formatting is complete, the restoration files will be extracted and copied to the USB drive. This process should take ~15 minutes when connected to a USB 2.0 port. When all files have been copied, this message will appear, press OK to continue.

### 11.4.13 Eject Drive

Press the “Eject Drive” button to safely remove the USB drive. The USB drive is now ready to be used to restore the image onto the PC.

## Hardware Setup

1. Make sure the computer is turned off, but plugged in.
2. Insert the USB Thumb Drive into a USB port on the Driver Station computer.

## Boot to USB

### Classmate:

1. Power on the Classmate and tap the F11 key on the keyboard. Tapping the F11 key during boot will bring up the boot menu.
2. Use the up/down keys to select the **USB HDD:** entry on the menu, then press the right arrow to expand the listing
3. Use the up/down arrow keys on the keyboard to select the USB device (it will be called “Generic Flash Disk”). Press the ENTER key when the USB device is highlighted.

### Acer ES1:

1. Power on the computer and tap the F12 key on the keyboard. Tapping the F12 key during boot will bring up the boot menu.
2. Use the up/down keys to select the **USB HDD: Generic** entry on the menu, then press the ENTER key when the USB device is highlighted.

**Acer ES1: If pressing F12 does not pull up the boot menu or if the USB device is not listed in the boot menu, see “Checking BIOS Settings” at the bottom of this article.**

### Image the Classmate

1. To confirm that you want to reimage the Classmate, type “1” and press ENTER.
2. Then, type “Y” and press ENTER. The Classmate will begin re-imaging. The installation will take 15-30 minutes.
3. When the installation is complete, remove the USB drive.
4. Restart the Classmate. The Classmate will boot into Windows.

### Initial Driver Station Boot

The first time the Classmate is turned on, there are some unique steps, listed below, that you’ll need to take. The initial boot may take several minutes; make sure you do not cycle power during the process.

Please note that these steps are only required during original startup.

#### 11.4.14 Enter Setup

1. Log into the Developer account.
2. Click “Ask me later”.
3. Click “OK”. The computer now enters a Set Up that may take a few minutes.

#### 11.4.15 Activate Windows

1. Establish an Internet connection.
2. Once you have an Internet connection, click the Start menu, right click “Computer” and click “Properties”.
3. Scroll to the bottom section, “Windows activation”, and Click “Activate Windows now”
4. Click “Activate Windows online now”. The activation may take a few minutes.
5. When the activation is complete, close all of the windows.

#### 11.4.16 Microsoft Security Essentials

Navigate through the Microsoft Security Essentials Setup Wizard. Once it is complete, close all of the windows.

### Acer ES1: Fix Wireless Driver

#### Acer ES1 PC only!

The default wireless driver in the image may have issues with intermittent communication with the robot radio. The correct driver is in the image, but could not be set to load by default. To load the correct driver, open the Device Manager by clicking start, typing “Device Manager” in the box and clicking Device Manager

### 11.4.17 Open Wireless Device Properties

Click on the arrow next to Network Adapters to expand it and locate the Wireless Network Adapter. Right click the adapter and select Properties.

### 11.4.18 Uninstall-Driver

Click on the Driver tab, then click the Uninstall button. Click Yes at any prompts.

### 11.4.19 Scan for New Hardware

Right click on the top entry of the tree and click “Scan for hardware changes”. The wireless adapter should automatically be re-detected and the correct driver should be installed.

### Update Software

In order for the Classmate images to be prepared on time, they are created before the final versions of the software were ready. To use the software for FRC some additional components will need to be installed. LabVIEW teams should continue with Installing the FRC Update Suite (All Languages). C++ or Java teams should continue Installing C++ and Java Development Tools for FRC.

### Errors during Imaging Process

If an error is detected during the imaging process, the following screen will appear. Note that the screenshot below shows the error screen for the Driver Station-only image for the E09. The specific image filename shown will vary depending on the image being applied.

The typical reason for the appearance of this message is due to an error with the USB device on which the image is stored. Each option is listed below with further details as to the actions you can take in pursuing a solution. Pressing any key once this error message is shown will return the user to the menu screen shown in Image the Classmate.

### 11.4.20 Option 1

*Using same image on the existing USB Flash drive:* To try this option, press any key to return to the main menu and select #1. This will run the imaging process again.

### 11.4.21 Option 2

*Reload the same image onto the USB Flash drive using RMPrepUSB:* It’s possible the error message was displayed due to an error caused during the creation of the USB Flash drive (e.g. file copy error, data corruption, etc.) Press any key to return to the main menu and select #4 to safely shutdown the Classmate then follow the steps starting with RMPrep to create a new USB Restoration Key using the same USB Flash drive.

### 11.4.22 Option 3

*Reload the same image onto a new USB Flash drive using RMPrepUSB:* The error message displayed may also be caused by an error with the USB Flash drive itself. Press any key to return to the main menu and select #4 to safely shutdown the Classmate. Select a new USB Flash drive and follow the steps starting with RMPrep.

### 11.4.23 Option 4

*Download a new image:* An issue with the downloaded image may also cause an error when imaging. Press any key to return to the main menu and select #4 to safely shutdown the Classmate. Starting with Download the Classmate Image create a new copy of the imaging stick.

### Checking BIOS Settings

If you are having difficulty booting to USB, check the BIOS settings to insure they are correct. To do this:

- Repeatedly tap the F2 key while the computer is booting to enter the BIOS settings
- Once the BIOS settings screen has loaded, use the right and left arrow keys to select the Main tab, then check if the line for F12 Boot Menu is set to Enabled. If it is not, use the Up/Down keys to highlight it, press Enter, use Up/Down to select Enabled and press Enter again.
- Next, use the Left/Right keys to select the Boot tab. Make sure that the Boot Mode is set to Legacy. If it is not, highlight it using Up/Down, press Enter, highlight Legacy and press Enter again. Press Enter to move through any pop-up dialogs you may see.
- Press F10 to save any changes and exit.



---

## What is WPILib?

---

The WPI Robotics Library (WPILib) is the standard software library provided for teams to write code for their FRC robots. A [software library](#) is a collection of code that can be imported into and used by other software. WPILib contains a set of useful classes and subroutines for interfacing with various parts of the FRC control system (such as sensors, motor controllers, and the driver station), as well as an assortment of other utility functions.

### 12.1 Supported languages

There are two versions of WPILib, one for each of the two officially-supported text-based languages: WPILibJ for java, and WPILibC for C++. A considerable effort is made to maintain feature-parity between these two languages - library features are not added unless they can be reasonably supported for both Java and C++, and when possible the class and method names are kept identical or highly-similar. While unofficial community-built support is available for some other languages, notably [python](#), this documentation will only cover Java and C++. Java and C++ were chosen for the officially-supported languages due to their appropriate level-of-abstraction and ubiquity in both industry and high-school computer science classes.

In general, C++ offers better high-end performance, at the cost of increased user effort (memory must be handled manually, and the C++ compiler does not do much to ensure user code will not crash at runtime). Java offers lesser performance, but much greater convenience. New/inexperienced users are strongly encouraged to use Java.

### 12.2 Source code and documentation

WPILib is an open-source library - the entirety of its source code is available online on the [WPILib Github Page](#):

- [Official WPILib github](#)

The Java and C++ source code can be found in the [WPILibJ](#) and [WPILibC](#) source directories:

- [Java source code](#)
- [C++ source code](#)

While users are strongly encouraged to read the source code to resolve detailed questions about library functionality, more-concise documentation can be found on the official documentation pages for WPILibJ and WPILibC:

- [Java documentation](#)
- [C++ documentation](#)

---

## RoboRIO Web Dashboard

---

The roboRIO web dashboard is a webpage built into the roboRIO that can be used for checking status and updating settings of the roboRIO.

Unlike the 2015-2018 roboRIO web dashboard, the 2019 web dashboard does not use SilverLight. Users may encounter issues using IE (compatibility) or Edge (mDNS site access). Alternate browsers such as Google Chrome or Mozilla Firefox are recommended for the best experience.

Note: The roboRIO web dashboard was been re-written for 2019. All CAN configuration functionality has been removed. Configuration of CAN devices should be done with software provided by the device vendor. For CTRE devices previously serviced using the webdashboard, the appropriate software is [CTRE Phoenix Tuner](#).

### 13.1 Opening the WebDash



To open the web dashboard, open a web browser and enter the address of the roboRIO into the address bar (172.22.11.2 for USB, or “roboRIO-####-FRC.local where #### is your team number, with no leading zeroes, for either interface). See this document for more details about mDNS and roboRIO networking: [RoboRIO Networking](#)

### 13.2 System Configuration Tab

The home screen of the web dashboard is the System Configuration tab which has 5 main sections:

1. Navigation Bar - This section allows you to navigate to different sections of the web dashboard. The different pages accessible through this navigation bar are discussed below.
2. System Settings - This section contains information about the System Settings. The Hostname field should not be modified manually, instead use the roboRIO Imaging tool to set the Hostname based on your team number. This section contains information such as the device IP, firmware version and image version.
3. Startup Settings - This section contains Startup settings for the roboRIO. These are described in the sub-step below

### 172.22.11.2: System Configuration

#### Settings

Hostname	<input type="text" value="roboRIO-1-FRC"/>
IP Address	0.0.0.0 (Ethernet) 172.22.11.2 (Ethernet)
DNS Name	
Vendor	National Instruments
Model	roboRIO
Serial Number	030498A9
Firmware Version	6.0.0f1
Operating System	NI Linux Real-Time ARMv7-A 4.9.47-rt37-ni-6.0.0f1
Status	Running
System Start Time	Mon Nov 19 2018 14:16:34 GMT-0500 (Eastern Standard Time)
Image Title	roboRIO Image
Image Version	FRC_roboRIO_2019_v12
Comments	<input style="width: 100%; height: 30px;" type="text"/>
Locale	English
VISA Resource Name	system

#### Startup Settings

- Force Safe Mode
- Enable Console Out
- Disable RT Startup App
- Disable FPGA Startup App
- Enable Secure Shell Server (sshd)
- LabVIEW Project Access

## 172.22.11.2: System Configuration

1

Save

**Settings** 2

Hostname	<input type="text" value="roboRIO-1-FRC"/>
IP Address	0.0.0.0 (Ethernet) 172.22.11.2 (Ethernet)
DNS Name	
Vendor	National Instruments
Model	roboRIO
Serial Number	030498A9
Firmware Version	6.0.0f1
Operating System	NI Linux Real-Time ARMv7-A 4.9.47-rt37-ni-6.0.0f1
Status	Running
System Start Time	Mon Nov 19 2018 14:16:34 GMT-0500 (Eastern Standard Time)
Image Title	roboRIO Image
Image Version	FRC_roboRIO_2019_v12
Comments	<input type="text"/>
Locale	English
VISA Resource Name	system

Update Firmware

**Startup Settings** 3

- Force Safe Mode
- Enable Console Out
- Disable RT Startup App
- Disable FPGA Startup App
- Enable Secure Shell Server (sshd)
- LabVIEW Project Access

4. System Resources (not pictured) - This section provides a snapshot of system resources such as memory and CPU load.

### 13.2.1 Startup Settings

**Startup Settings**

- Force Safe Mode
- Enable Console Out
- Disable RT Startup App
- Disable FPGA Startup App
- Enable Secure Shell Server (sshd)
- LabVIEW Project Access

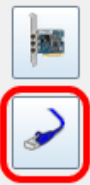
- Force Safe Mode - Forces the controller into Safe Mode. This can be used with troubleshooting imaging issues, but it is recommended to use the Reset button on the roboRIO to put the device into Safe Mode instead (with power already applied, hold the reset button for 5 seconds). **Default is unchecked.**
- Enable Console Out - This enables the on-board RS232 port to be used as a Console output. It is recommended to leave this enabled unless you are using this port to talk to a serial device (note that this port uses RS232 levels and should not be connected to many microcontrollers which use TTL levels). **Default is checked.**
- Disable RT Startup App - Checking this box disables code from running at startup. This may be used for troubleshooting if you find the roboRIO is unresponsive to new program download. Default is unchecked
- Disable FPGA Startup App - **This box should not be checked.**
- Enable Secure Shell Server (sshd) - **It is recommended to leave this box checked.** This setting enables SSH which is a way to remotely access a console on the roboRIO. Unchecking this box will prevent C++ and Java teams from loading code onto the roboRIO.
- LabVIEW Project Access - **\*\* It is recommended to leave this box checked.\*\*** This setting allows LabVIEW projects to access the roboRIO.

## 13.3 Network Configuration

This page shows the configuration of the roboRIO's network adapters. **It is not recommended to change any settings on this page.** For more information on roboRIO networking see this article: [RoboRIO Networking](#)

## 172.22.11.2: Network Configuration

Save



### Ethernet Adapter eth0

#### Settings

MAC Address	00:80:2F:30:49:8A
Configure IPv4 Address	<input type="text" value="DHCP or Link Local"/>
IPv4 Address	0.0.0.0
Subnet Mask	0.0.0.0
Gateway	0.0.0.0
DNS Server	0.0.0.0
Current Link Speed	10Mbit/Half Duplex
Preferred Link Speed	<input type="text" value="Autonegotiate"/>

### Ethernet Adapter usb0

#### Settings

MAC Address	00:80:2F:40:49:8A
Configure IPv4 Address	DHCP Only
IPv4 Address	172.22.11.2
Subnet Mask	255.255.255.248
Gateway	0.0.0.0
DNS Server	0.0.0.0
Current Link Speed	Autonegotiate
Preferred Link Speed	Autonegotiate





---

## Operating pneumatic cylinders

---

### 14.1 Solenoid control

FRC teams use solenoids to perform a variety of tasks, from shifting gearboxes to operating robot mechanisms. A solenoid is a valve used to electronically switch a pressurised air line “on” or “off”. For more information on solenoids, see [this wikipedia article](#). Solenoids are controlled by a robot’s Pneumatics Control Module, or PCM, which is in turn connected to the robot’s RoboRIO via CAN. The easiest way to see a solenoid’s state is via the small red LED (which indicates if the valve is “on” or not), and solenoids can be manually actuated when un-powered with the small button adjacent to the LED.

Single acting solenoids apply or vent pressure from a single output port. They are typically used either when an external force will provide the return action of the cylinder (spring, gravity, separate mechanism) or in pairs to act as a double solenoid. A double solenoid switches air flow between two output ports (many also have a center position where neither output is vented or connected to the input). Double solenoid valves are commonly used when you wish to control both the extend and retract actions of a cylinder using air pressure. Double solenoid valves have two electrical inputs which connect back to two separate channels on the solenoid breakout.

PCM Modules are identified by their CAN Device ID. The default CAN ID for PCMs is 0. If using a single PCM on the bus it is recommended to leave it at the default CAN ID. This ID can be changed with the Phoenix Tuner application, in addition to other debug information. (TODO Link Phoenix tuner article) **TODO FIX THIS LINK:** For more information about setting PCM Node IDs see Updating and Configuring Pneumatics Control Module and Power Distribution Panel.

### 14.2 Single Solenoids in WPILib

#### 14.2.1 Using a single acting solenoid

Single solenoids in WPILib are controlled using the Solenoid class. To construct a Solenoid object, simply pass the desired port number (assumes Node ID 0) or Node ID and port number to the constructor. To set the value of the solenoid call `set(true)` to enable or `set(false)` to disable the solenoid output.

C++

```
frc::Solenoid exampleSolenoid {1};  
  
exampleSolenoid.Set(true);  
exampleSolenoid.Set(false);
```

#### Java

```
Solenoid exampleSolenoid = new Solenoid(1);  
  
exampleSolenoid.set(true);  
exampleSolenoid.set(false);
```

## 14.2.2 Double Solenoids in WPILib

Double solenoids are controlled by the DoubleSolenoid class in WPILib. These are constructed similarly to the single solenoid but there are now two port numbers to pass to the constructor, a forward channel (first) and a reverse channel (second). The state of the valve can then be set to kOff (neither output activated), kForward (forward channel enabled) or kReverse (reverse channel enabled). Additionally, the PCM CAN ID can be passed to the DoubleSolenoid if teams have a non-standard PCM CAN ID

#### C++

```
frc::DoubleSolenoid exampleDouble {1, 2};  
frc::DoubleSolenoid exampleDouble {/* The PCM CAN ID */ 9, 1, 2};  
  
exampleDouble.Set(frc::DoubleSolenoid::Value::kOff);  
exampleDouble.Set(frc::DoubleSolenoid::Value::kForward);  
exampleDouble.Set(frc::DoubleSolenoid::Value::kReverse);
```

#### Java

```
// Using "import static an.enum.or.constants.inner.class.*;" helps reduce verbosity  
// this replaces "DoubleSolenoid.Value.kForward" with just kForward  
// further reading is available at https://www.geeksforgeeks.org/static-import-java/  
import static edu.wpi.first.wpilibj.DoubleSolenoid.Value.*;  
  
DoubleSolenoid exampleDouble = new DoubleSolenoid(1, 2);  
DoubleSolenoid anotherDoubleSolenoid = new DoubleSolenoid(/* The PCM CAN ID */ 9, 4,   
↪5);  
  
exampleDouble.set(kOff);  
exampleDouble.set(kForward);  
exampleDouble.set(kReverse);
```

---

## Strategies for vision programming

---

Using computer vision is a great way of making your robot be responsive to the elements on the field and make it much more autonomous. Often in FRC games there are bonus points for autonomously shooting balls or other game pieces into goals or navigating to locations on the field. Computer vision is a great way of solving many of these problems. And if you have autonomous code that can do the challenge, then it can be used during the teleop period as well to help the human drivers.

There are many options for choosing the components for vision processing and where the vision program should run. WPILib and associated tools support a number of options and give teams a lot of flexibility to decide what to do. This article will attempt to give you some insight into many of the choices and tradeoffs that are available.

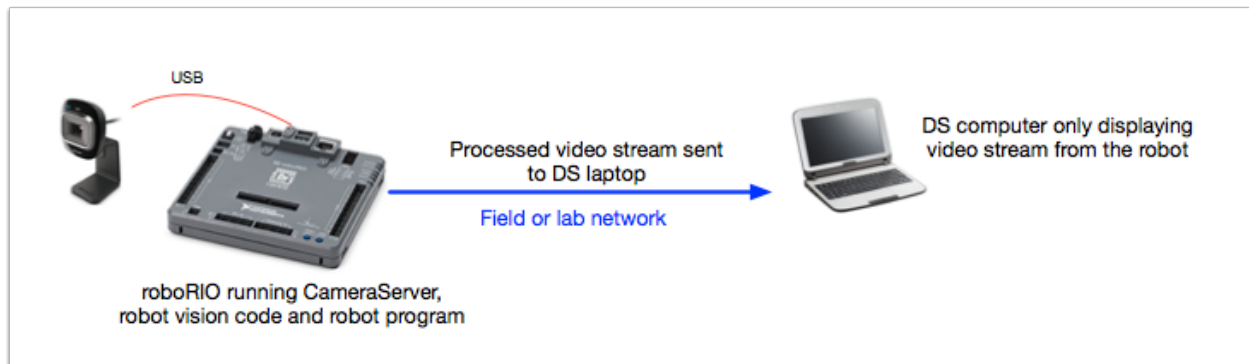
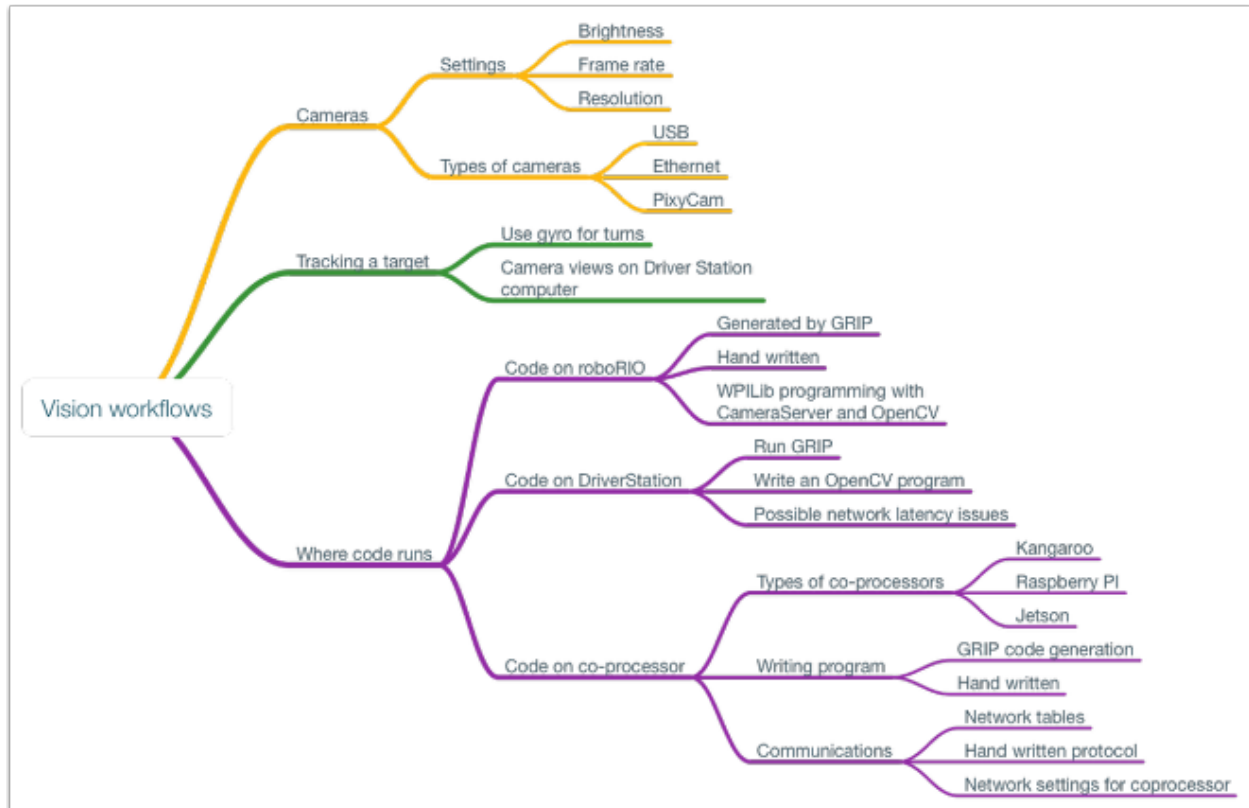
### 15.1 OpenCV computer vision library

**OpenCV** is an open source computer vision library that is widely used throughout academia and industry. It has support from hardware manufactures providing GPU accelerated processing, it has bindings for a number of languages including C++, Java, and Python. It is also well documented with many web sites, books, videos, and training courses so there are lots of resources available to help learn how to use it. For these reasons and more we have adopted OpenCV for use with the C++ and Java versions of WPILib. Starting in 2017 **we have prebuilt OpenCV libraries included with WPILib**, support in the library for capturing, processing and viewing video, and tools to help you create your vision algorithms. For more information about OpenCV see <http://opencv.org>.

### 15.2 Vision code on roboRIO

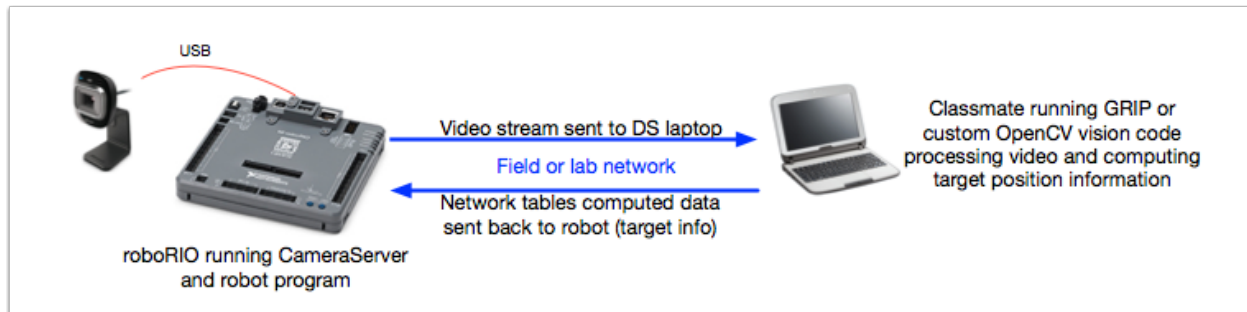
The programming is fairly straightforward since the vision program is just part of the overall robot program. Program can be written by hand or generated by GRIP in either C++ or Java. The disadvantage is that having vision code running on the same processor as the robot program can have performance issues. This is something you will have to evaluate depending on the requirements for your robot and vision program.

The vision code simply produces results that the robot code uses. Be careful about synchronization issues when writing robot code that is getting values from a vision thread. The GRIP generated code and the VisionRunner class in WPILib will make this easier.



The video stream can be sent to the SmartDashboard by using the camera server interface so operators can see what the camera sees. In addition, you can add annotations to the images using OpenCV commands so targets or other interesting objects can be identified in the dashboard view.

## 15.3 Vision code on DS computer

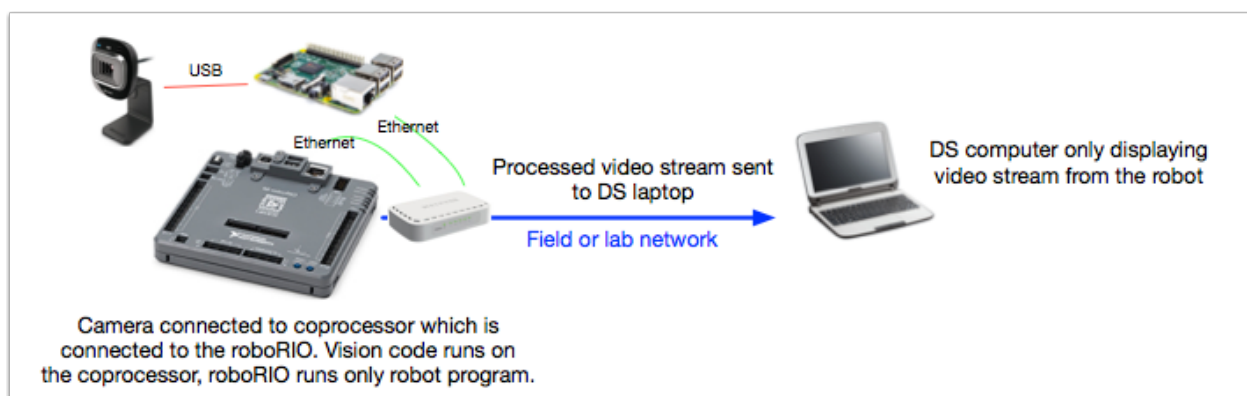


The video is streamed back to the Driver Station laptop for processing. Even the Classmate laptops are substantially faster at vision processing than the roboRIO and don't have real-time programs running on them. GRIP can be run on the Driver Station laptop directly with the results sent back to the robot using NetworkTables. Alternatively you can write your own vision program using a language of your choosing. Python makes a good choice since there is a native NetworkTables implementation and the OpenCV bindings are very good.

After the images are processed, the key values such as the target position, distance or anything else you need can be sent back to the robot with NetworkTables. Latency may be an issue since there might be delays in the images getting to the laptop and the results getting back.

The video stream can be displayed on SmartDashboard or in GRIP.

## 15.4 Vision code on a coprocessor



Coprocessors such as the Raspberry PI or Kangaroo are ideal for supporting vision code. The advantage is that they can run full speed and not interfere with the robot program. In this case, the camera is probably connected to the coprocessor or (in the case of ethernet cameras) an ethernet switch on the robot. The program can be written in any language although Python is a good choice because of its simple bindings to OpenCV and NetworkTables. Some teams have used high performance vision coprocessors such as the Nvidia Jetson for fastest speed and highest resolution although it might be too complex for inexperienced programmers.

Data can be sent from the vision program on the coprocessor to the robot using NetworkTables or a private protocol over a network or serial connection.

### 15.5 Camera options

There are a number of camera options supported by WPILib. Cameras have a number of parameters that can determine how it operates. For example, frame rate and image resolution effect the quality of the received images but when set too high impact processing time and the bandwidth which may have serious effect on your robot program.

New for 2017 there is a CameraServer class available in C++ and Java that interfaces with cameras connected to the robot. It retrieve frames for local processing through a Source object and send stream to your driver station for viewing or processing there. The video stream is processed by the CameraServer.

Details on using cameras with WPILib are detailed in the next section.

---

## Read and process video: CameraServer class

---

### 16.1 Concepts

The cameras typically used in FRC (commodity USB and Ethernet cameras such as the Axis camera) offer relatively limited modes of operation. In general, they provide only a single image output (typically in an RGB compressed format such as JPG) at a single resolution and frame rate. USB cameras are particularly limited as only one application may access the camera at a time.

The 2017 CameraServer supports multiple cameras. It handles details such as automatically reconnecting when a camera is disconnected, and also makes images from the camera available to multiple “clients” (e.g. both your robot code and the dashboard can connect to the camera simultaneously).

#### 16.1.1 Camera names

Each camera in CameraServer must be uniquely named. This is also the name that appears for the camera in the Dashboard. Some variants of the CameraServer `startAutomaticCapture()` and `addAxisCamera()` functions will automatically name the camera (e.g. “USB Camera 0” or “Axis Camera”), or you can give the camera a more descriptive name (e.g. “Intake Cam”). The only requirement is that each camera have a unique name.

### 16.2 USB Camera Notes

#### 16.2.1 CPU Usage

The CameraServer is designed to minimize CPU usage by only performing compression and decompression operations when required and automatically disabling streaming when no clients are connected.

To minimize CPU usage, the dashboard resolution should be set to the same resolution as the camera; this allows the CameraServer to not decompress and recompress the image, instead, it can simply forward the JPEG image received from the camera directly to the dashboard. It’s important to note that changing the resolution on the dashboard does *not* change the camera resolution; changing the camera resolution may be done either through NetworkTables or in your robot code (by calling `setResolution` on the camera object).

## 16.2.2 USB Bandwidth

The roboRio can only transmit and receive so much data at a time over its USB interfaces. Camera images can require a lot of data, and so it is relatively easy to run into this limit. The most common cause of a USB bandwidth error is selecting a non-JPEG video mode or running too high of a resolution, particularly when multiple cameras are connected.

## 16.3 Architecture

The CameraServer for 2017 consists of two layers, the high level WPILib **CameraServer class** and the low level **cscore library**.

## 16.4 CameraServer class

The CameraServer class (part of WPILib) provides a high level interface for adding cameras to your robot code. It also is responsible for publishing information about the cameras and camera servers to NetworkTables so that Driver Station dashboards such as the LabView Dashboard and SmartDashboard can list the cameras and determine where their streams are located. It uses a singleton pattern to maintain a database of all created cameras and servers.

Some key functions in CameraServer are:

- **startAutomaticCapture()**: Add a USB camera (e.g. Microsoft LifeCam) and starts a server for it so it can be viewed from the dashboard.
- **addAxisCamera()**: Add an Axis camera. Even if you aren't processing images from the Axis camera in your robot code, you may want to use this function so that the Axis camera appears in the Dashboard's drop down list of cameras. It also starts a server so the Axis stream can still be viewed when your driver station is connected to the roboRio via USB (useful at competition if you have both the Axis camera and roboRio connected to the two robot radio Ethernet ports).
- **getVideo()**: Get OpenCV access to a camera. This allows you to get images from the camera for image processing on the roboRio (in your robot code).
- **putVideo()**: Start a server that you can feed OpenCV images to. This allows you to pass custom processed and/or annotated images to the dashboard.

## 16.5 cscore Library

The cscore library provides the lower level implementation to:

- Get images from USB and HTTP (e.g. Axis) cameras
- Change camera settings (e.g. contrast and brightness)
- Change camera video modes (pixel format, resolution and frame rate)
- Act as a web server and serve images as a standard M-JPEG stream
- Convert images to/from OpenCV "Mat" objects for image processing



## 16.5.1 Sources and Sinks

The basic architecture of the cscore library is similar to that of MJPGStreamer, with functionality split between sources and sinks. There can be multiple sources and multiple sinks created and operating simultaneously. An object that generates images is a source and an object that accepts/consumes images is a sink. The generate/consume is from the perspective of the library. Thus cameras are sources (they generate images). The MJPEG web server is a sink because it accepts images from within the program (even though it may be forwarding those images on to a web browser or dashboard). Sources may be connected to multiple sinks, but sinks can be connected to one and only one source. When a sink is connected to a source, the cscore library takes care of passing each image from the source to the sink.

- **Sources** obtain individual frames (such as provided by a USB camera) and fire an event when a new frame is available. If no sinks are listening to a particular source, the library may pause or disconnect from a source to save processor and I/O resources. The library autonomously handles camera disconnects/reconnects by simply pausing and resuming firing of events (e.g. a disconnect results in no new frames, not an error).
- **Sinks** listen to a particular source's event, grab the latest image, and forward it to its destination in the appropriate format. Similarly to sources, if a particular sink is inactive (e.g. no client is connected to a configured M-JPEG over HTTP server), the library may disable parts of its processing to save processor resources.

User code (such as that used in a FRC robot program) can act as either a source (providing processed frames as if it were a camera) or as a sink (receiving a frame for processing) via OpenCV source and sink objects. Thus an image processing pipeline that gets images from a camera and serves the processed images out looks like the below graph:

**UsbCamera (VideoSource) -> (cscore internal) -> CvSink (VideoSink) -> (your OpenCV processing code) -> CvSource (VideoSource) -> (cscore internal) -> MjpegServer (VideoSink)**

Because sources can have multiple sinks connected, the pipeline may branch. For example, the original camera image can also be served by connecting the UsbCamera source to a second MjpegServer sink in addition to the CvSink, resulting in the below graph:

**UsbCamera -> CvSink -> (your code) -> CvSource -> MjpegServer [2] -> MjpegServer [1]**

When a new image is captured by the camera, both the CvSink and the MjpegServer [1] receive it.

The above graph is what the following CameraServer snippet creates:

Java

```
// Creates UsbCamera and MjpegServer [1] and connects them
CameraServer.getInstance().startAutomaticCapture()

// Creates the CvSink and connects it to the UsbCamera CvSink cvSink = CameraServer.
↳getInstance().getVideo()
// Creates the CvSource and MjpegServer [2] and connects them CvSource outputStream =
↳CameraServer.getInstance().putVideo("Blur", 640, 480);
```

The CameraServer implementation effectively does the following at the cscore level (for explanation purposes). CameraServer takes care of many of the details such as creating unique names for all cscore objects and automatically selecting port numbers. CameraServer also keeps a singleton registry of created objects so they aren't destroyed if they go out of scope.

Java

```
UsbCamera usbCamera = new UsbCamera("USB Camera 0", 0);

MjpegServer mjpegServer1 = new MjpegServer("serve_USB Camera 0", 1181);
mjpegServer1.setSource(usbCamera); CvSink cvSink = new CvSink("opencv_USB Camera 0");

cvSink.setSource(usbCamera);
CvSource outputStream = new CvSource("Blur", PixelFormat.kMJPEG, 640, 480, 30);
```

(continues on next page)

(continued from previous page)

```
MjpegServer mjpegServer2 = new MjpegServer("serve_Blur", 1182);  
mjpegServer2.setSource(outputStream);
```

## 16.5.2 Reference Counting

All cscore objects are internally reference counted. Connecting a sink to a source increments the source's reference count, so it's only strictly necessary to keep the sink in scope. The CameraServer class keeps a registry of all objects created with CameraServer functions, so sources and sinks created in that way effectively never go out of scope (unless explicitly removed).

### 17.1 LabVIEW

The 2017 LabVIEW Vision Example is included with the other LabVIEW examples. From the Splash screen, click Support->Find FRC Examples or from any other LabVIEW window, click Help->Find Examples and locate the Vision folder to find the 2017 Vision Example. The example images are bundled with the example.

### 17.2 C++/Java

A more complete example is coming soon.

For now, we have provided a GRIP project and the description below, as well as the example images, bundled into a ZIP that [can be found on TeamForge](#).

Details about integrating GRIP generated code in your robot program is in the Using Generated Code in a Robot Program article.

The code generated by the included GRIP project will find OpenCV contours for green particles in images like the ones included in the Vision Images folder of this ZIP. From there you may wish to further process these contours to assess if they are the target. To do this:

1. Use the boundingRect method to draw bounding rectangles around the contours
2. The LabVIEW example code calculates 5 separate ratios for the target. Each of these ratios should nominally equal 1.0. To do this, it sorts the contours by size, then starting with the largest, calculates these values for every possible pair of contours that may be the target, and stops if it finds a target or returns the best pair it found.

In the formulas below, 1 followed by a letter refers to a coordinate of the bounding rect of the first contour, which is the larger of the two (e.g. 1L = 1st contour left edge) and 2 with a letter is the 2nd contour. (H=Height, L = Left, T = Top, B = Bottom, W = Width)

- Group Height =  $1H / ((2B - 1T) * .4)$  Top height should be 40% of total height (4in / 10 in.)
- dTop =  $(2T - 1T) / ((2B - 1T) * .6)$  Top of bottom stripe to top of top stripe should be 60% of total height (6in / 10 in.)

- $L_{Edge} = ((1L - 2L) / 1W) + 1$  The distance between the left edge of contour 1 and the left edge of contour 2 should be small relative to the width of the 1st contour (then we add 1 to make the ratio centered on 1)
- $Width\ ratio = 1W / 2W$  The widths of both contours should be about the same
- $Height\ ratio = 1H / (2H * 2)$  The larger stripe should be twice as tall as the smaller one

Each of these ratios is then turned into a 0-100 score by calculating:  $100 - (100 * \text{abs}(1 - \text{Val}))$

3. To determine distance, measure pixels from top of top bounding box to bottom of bottom bounding box

$$\text{distance} = \text{Target height in ft.} \cdot (10/12) * Y_{Res} / (2 * \text{PixelHeight} * \tan(\text{viewAngle of camera}))$$

The LabVIEW example uses height as the edges of the round target are the most prone to noise in detection (as the angle points further from the camera the color looks less green). The downside of this is that the pixel height of the target in the image is affected by perspective distortion from the angle of the camera. Possible fixes include:

- Try using width instead
- Empirically measure height at various distances and create a lookup table or regression function
- Mount the camera to a servo, center the target vertically in the image and use servo angle for distance calculation (you'll have to work out the proper trig yourself or find a math teacher to help!)
- Correct for the perspective distortion using OpenCV. To do this you will need to calibrate your camera with OpenCV as described here: [http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera\\_calibration/camera\\_calibration.html](http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html) This will result in a distortion matrix and camera matrix. You will take these two matrices and use them with the `undistortPoints` function to map the points you want to measure for the distance calculation to the "correct" coordinates (this is much less CPU intensive than undistorting the whole image)

---

## Target Info and Retroreflection

---

This document describes the Vision Targets from the 2016 FRC game and the visual properties of the material making up the targets. Note that for official dimensions and drawings of all field components, please see the Official Field Drawings.

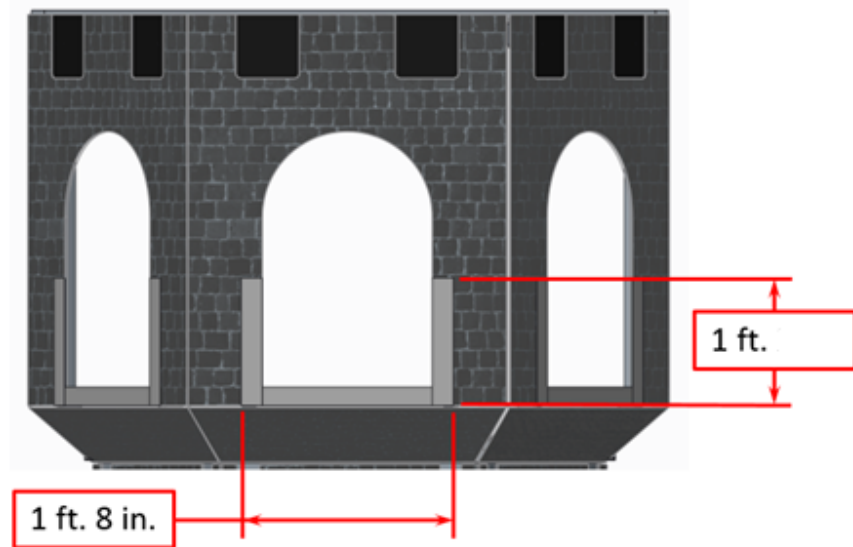
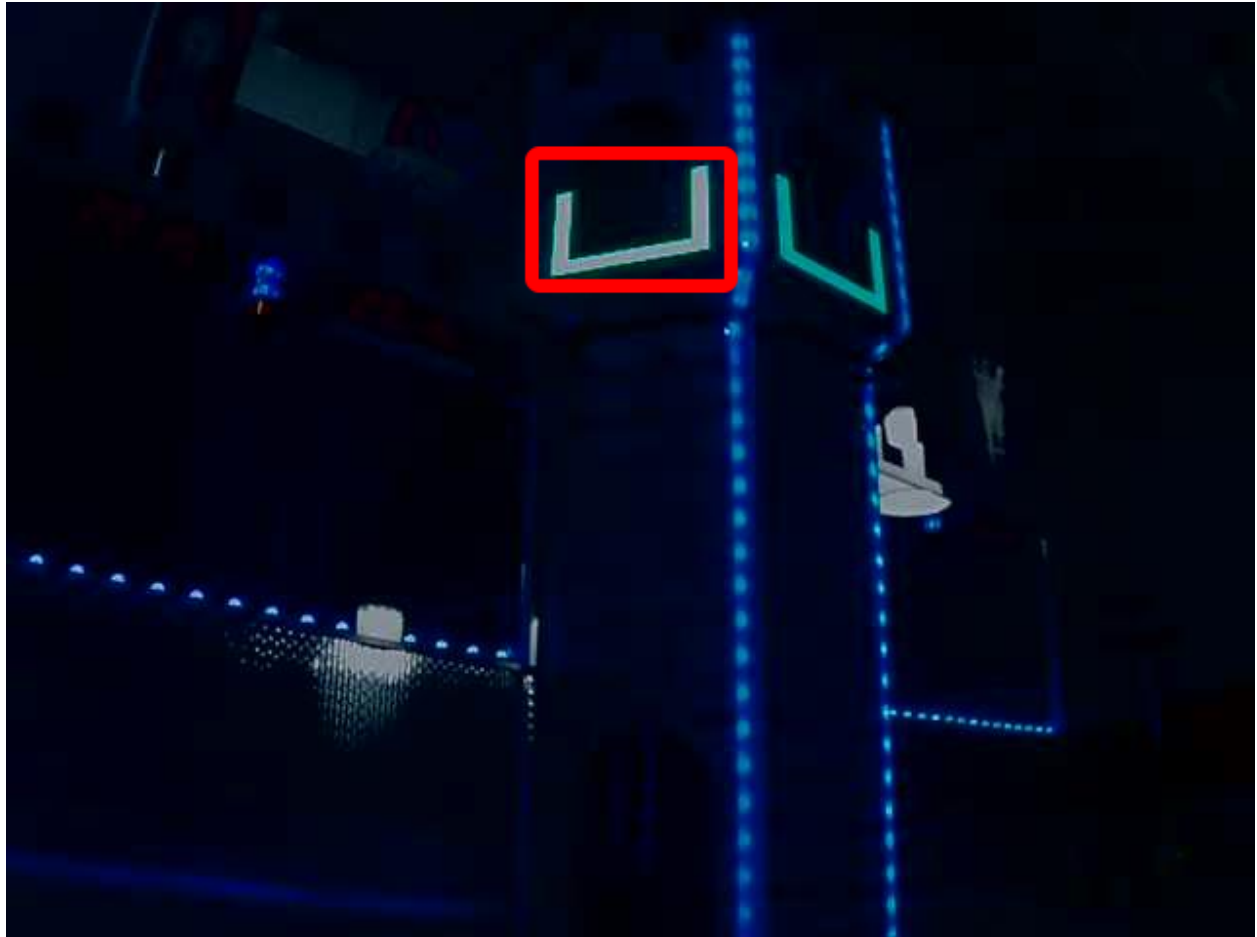
### 18.1 Targets

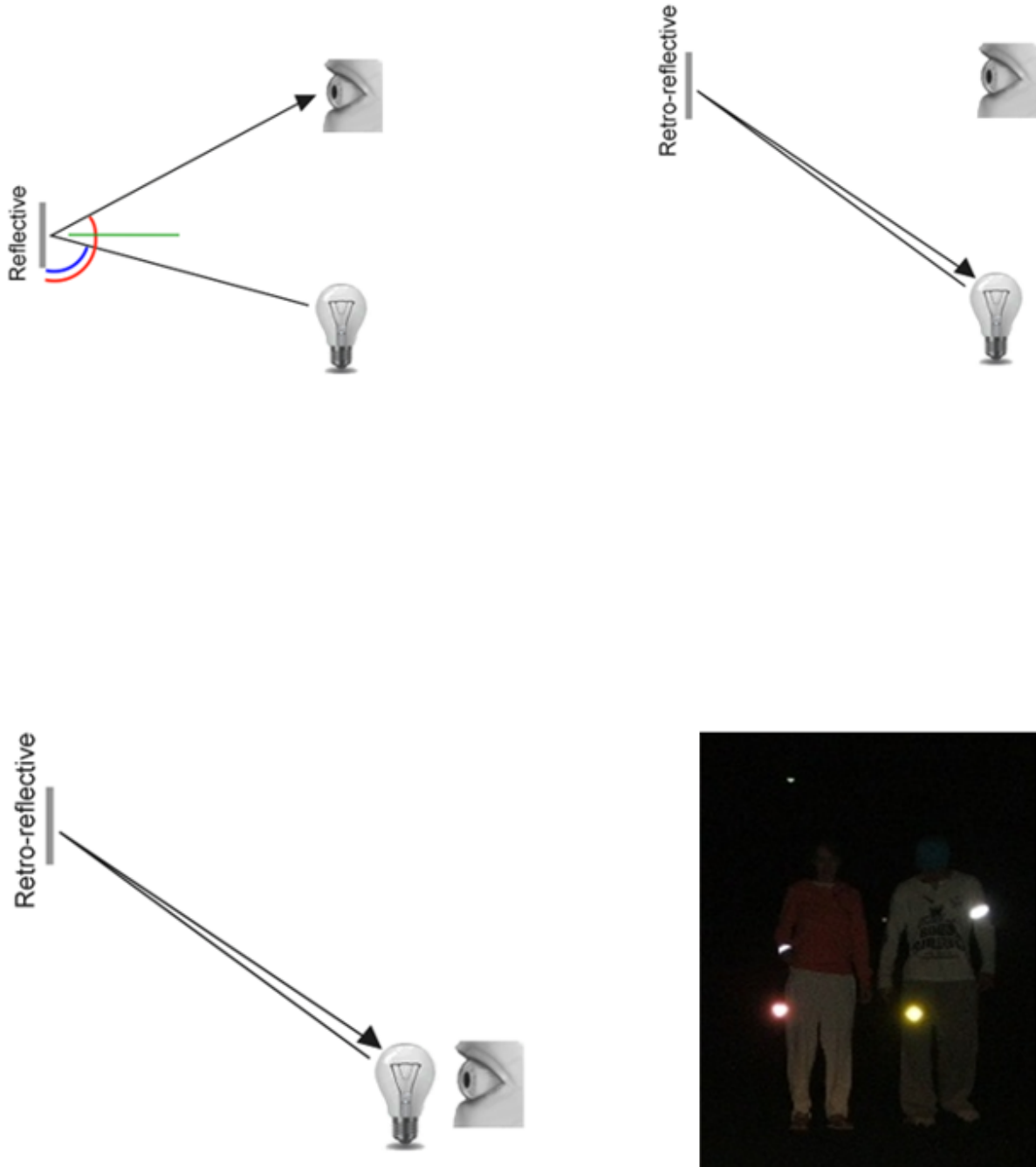
Each Target consists of a 1' 8" wide, 1' tall U-shape made of 2" wide retroreflective material (3M 8830 Silver Marking Film). The targets are located immediately adjacent to the bottom of each high goal. When properly lit, the retroreflective tape produces a bright and/or color-saturated marker.

### 18.2 Retroreflectivity vs. Reflectivity

Highly reflective materials are generally mirrored so that light “bounces off” at a supplementary angle. As shown above-left, the blue and red angles sum to 180 degrees. An equivalent explanation is that the light reflects about the surface normal the green line drawn perpendicular to the surface. Notice that a light pointed at the surface will return to the light source only if the blue angle is ~90 degrees.

Retro-reflective materials are not mirrored, but it will typically have either shiny facets across the surface, or it will have a pearl-like appearance. Not all faceted or pearl-like materials are retro-reflective, however. Retro-reflective materials return the majority of light back to the light source, and they do this for a wide range of angles between the surface and the light source, not just the 90 degree case. Retro-reflective materials accomplish this using small prisms, such as found on a bicycle or roadside reflector, or by using small spheres with the appropriate index of refraction that accomplish multiple internal reflections. In nature, the eyes of some animals, including house cats, also exhibit the retro-reflective effect typically referred to as night-shine. The Wikipedia articles on retro-reflection go into more detail on how retro-reflection is accomplished.





## 18.3 Examples of Retroreflection

This material should be relatively familiar as it is often used to enhance nighttime visibility of road signs, bicycles, and pedestrians.

Initially, retro-reflection may not seem like a useful property for nighttime safety, but when the light and eye are near one another, as shown below, the reflected light returns to the eye, and the material shines brightly even at large distances. Due to the small angle between the driver's eyes and vehicle headlights, retro-reflective materials can greatly increase visibility of distant objects during nighttime driving.

## 18.4 Demonstration

To further explore retro-reflective material properties:

1. Place a piece of the material on a wall or vertical surface
2. Stand 10-20 feet away, and shine a small flashlight at the material.
3. Start with the light held at your belly button, and raise it slowly until it is between your eyes. As the light nears your eyes, the intensity of the returned light will increase rapidly.
4. Alter the angle by moving to other locations in the room and repeating. The bright reflection should occur over a wide range of viewing angles, but the angle from light source to eye is key and must be quite small.

Experiment with different light sources. The material is hundreds of times more reflective than white paint; so dim light sources will work fine. For example, a red bicycle safety light will demonstrate that the color of the light source determines the color of the reflected light. If possible, position several team members at different locations, each with their own light source. This will show that the effects are largely independent, and the material can simultaneously appear different colors to various team members. This also demonstrates that the material is largely immune to environmental lighting. The light returning to the viewer is almost entirely determined by a light source they control or one directly behind them. Using the flashlight, identify other retro-reflective articles already in your environment ... on clothing, backpacks, shoes, etc.

## 18.5 Lighting

We have seen that the retro-reflective tape will not shine unless a light source is directed at it, and the light source must pass very near the camera lens or the observer's eyes. While there are a number of ways to accomplish this, a very useful type of light source to investigate is the ring flash, or ring light, shown above. It places the light source directly on or around the camera lens and provides very even lighting. Because of their bright output and small size, LEDs are particularly useful for constructing this type of device.

As shown above, inexpensive circular arrangements of LEDs are available in a variety of colors and sizes and are easy to attach to the Axis cameras. While not designed for diffuse even lighting, they work quite well for causing retro-reflective tape to shine. A small green LED ring is available through FIRST Choice. Other similar LED rings are available from the supplier, [SuperBrightLEDs.com](http://SuperBrightLEDs.com)

## 18.6 Sample Images

Sample Images are located with the code examples for each language (packaged with LabVIEW, in a separate ZIP in the same location as the C++/Java samples).







---

## Identifying and Processing the Targets

---

Once an image is captured, the next step is to identify Vision Target(s) in the image. This document will walk through one approach to identifying the 2016 targets. Note that the images used in this section were taken with the camera intentionally set to underexpose the images, producing very dark images with the exception of the lit targets, see the section on Camera Settings for details.

### 19.1 Additional Options

This document walks through the approach used by the example code provided in LabVIEW (for PC or roboRIO), C++ and Java. In addition to these options teams should be aware of the following alternatives that allow for vision processing on the Driver Station PC or an on-board PC:

1. [RoboRealm](#)
2. SmartDashboard Camera Extension (programmed in Java, works with any robot language)
3. [GRIP](#)

### 19.2 Original Image

The image shown below is the starting image for the example described here. The image was taken using the green ring light available in FIRST Choice. combined with an additional ring light of a different size. Additional sample images are provided with the vision code examples.

### 19.3 What is HSL/HSV?

The Hue or tone of the color is commonly seen on the artist's color wheel and contains the colors of the rainbow Red, Orange, Yellow, Green, Blue, Indigo, and Violet. The hue is specified using a radial angle on the wheel, but in imaging the circle typically contains only 256 units, starting with red at zero, cycling through the rainbow, and wrapping back to red at the upper end. Saturation of a color specifies amount of color, or the ratio of the hue color to a shade of gray.



Higher ratio means more colorful, less gray. Zero saturation has no hue and is completely gray. Luminance or Value indicates the shade of gray that the hue is blended with. Black is 0 and white is 255.

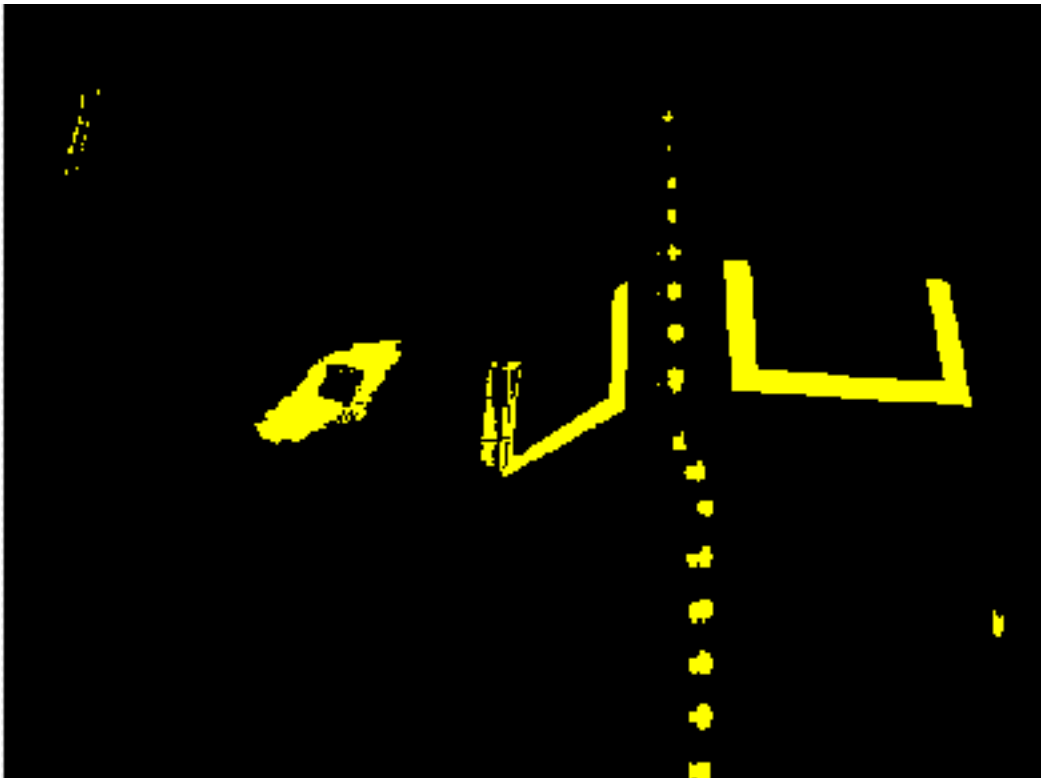
The example code uses the HSV color space to specify the color of the target. The primary reason is that it readily allows for using the brightness of the targets relative to the rest of the image as a filtering criteria by using the Value (HSV) or Luminance (HSL) component. Another reason to use the HSV color system is that the thresholding operation used in the example runs more efficiently on the roboRIO when done in the HSV color space.

## 19.4 Masking

In this initial step, pixel values are compared to constant color or brightness values to create a binary mask shown below in yellow. This single step eliminates most of the pixels that are not part of a target's retro-reflective tape. Color based masking works well provided the color is relatively saturated, bright, and consistent. Color inequalities are generally more accurate when specified using the HSL (Hue, Saturation, and Luminance) or HSV (Hue, Saturation, and Value) color space than the RGB (Red, Green, and Blue) space. This is especially true when the color range is quite large in one or more dimension.

Notice that in addition to the target, other bright parts of the image (overhead light and tower lighting) are also caught by the masking step.

Notice that in addition to the target, other bright parts of the image (overhead light and tower lighting) are also caught by the masking step.



## 19.5 Particle Analysis

After the masking operation, a particle report operation is used to examine the area, bounding rectangle, and equivalent rectangle for the particles. These are used to compute several scored terms to help pick the shapes that are most

rectangular. Each test described below generates a score (0-100) which is then compared to pre-defined score limits to decide if the particle is a target or not.

### 19.5.1 Coverage Area

The Area score is calculated by comparing the area of the particle compared to the area of the bounding box drawn around the particle. The area of the retroreflective strips is 80 square inches. The area of the rectangle that contains the target is 240 square inches. This means that the ideal ratio between area and bounding box area is 1/3. Area ratios close to 1/3 will produce a score near 100, as the ratio diverges from 1/3 the score will approach 0.

### 19.5.2 Aspect Ratio

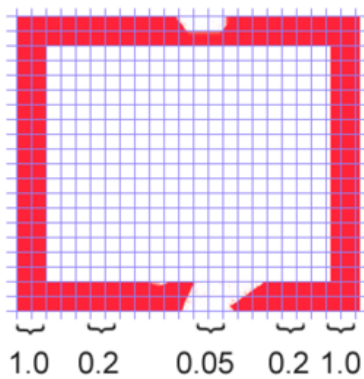
The aspect ratio score is based on (Particle Width / Particle Height). The width and height of the particle are determined using something called the “equivalent rectangle”. The equivalent rectangle is the rectangle with side lengths x and y where  $2x+2y$  equals the particle perimeter and  $x*y$  equals the particle area. The equivalent rectangle is used for the aspect ratio calculation as it is less affected by skewing of the rectangle than using the bounding box. When using the bounding box rectangle for aspect ratio, as the rectangle is skewed the height increases and the width decreases.

The target is 20” wide by 12” tall, for a ratio of 1.6. The detected aspect ratio is compared to this ideal ratio. The aspect ratio score is normalized to return 100 when the ratio matches the target ratio and drops linearly as the ratio varies below or above.

### 19.5.3 Moment

The moment measurement calculates the particles moment of inertia about it’s center of mass. This measurement provides a representation of the pixel distribution in the particle. The ideal score for this test is ~0.28. See: [Moment of Inertia](#)

### 19.5.4 X/Y Profiles



Column averages for a particle rectangle.



White line is the average, red is upper limit, and green is lower limit..

The edge score describes whether the particle matches the appropriate profile in both the X and Y directions. As shown, it is calculated using the row and column averages across the bounding box extracted from the original image and comparing that to a profile mask. The score ranges from 0 to 100 based on the number of values within the row or column averages that are between the upper and lower limit values.

## 19.6 Measurements

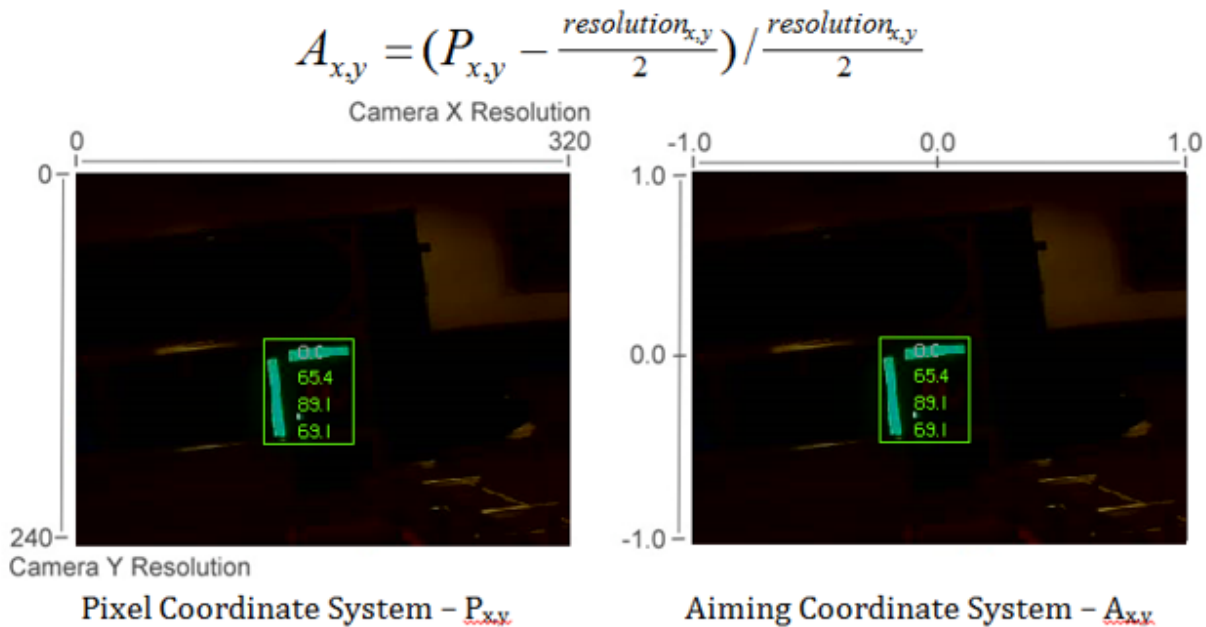
If a particle scores well enough to be considered a target, it makes sense to calculate some real-world measurements such as position and distance. The example code includes these basic measurements, so let's look at the math involved to better understand it.

### 19.6.1 Position

The target position is well described by both the particle and the bounding box, but all coordinates are in pixels with 0,0 being at the top left of the screen and the right and bottom edges determined by the camera resolution. This is a useful system for pixel math, but not nearly as useful for driving a robot; so let's change it to something that may be more useful.

To convert a point from the pixel system to the aiming system, we can use the formula shown below.

The resulting coordinates are close to what you may want, but the Y axis is inverted. This could be corrected by multiplying the point by [1,-1] (Note: this is not done in the sample code). This coordinate system is useful because it has a centered origin and the scale is similar to joystick outputs and RobotDrive inputs.



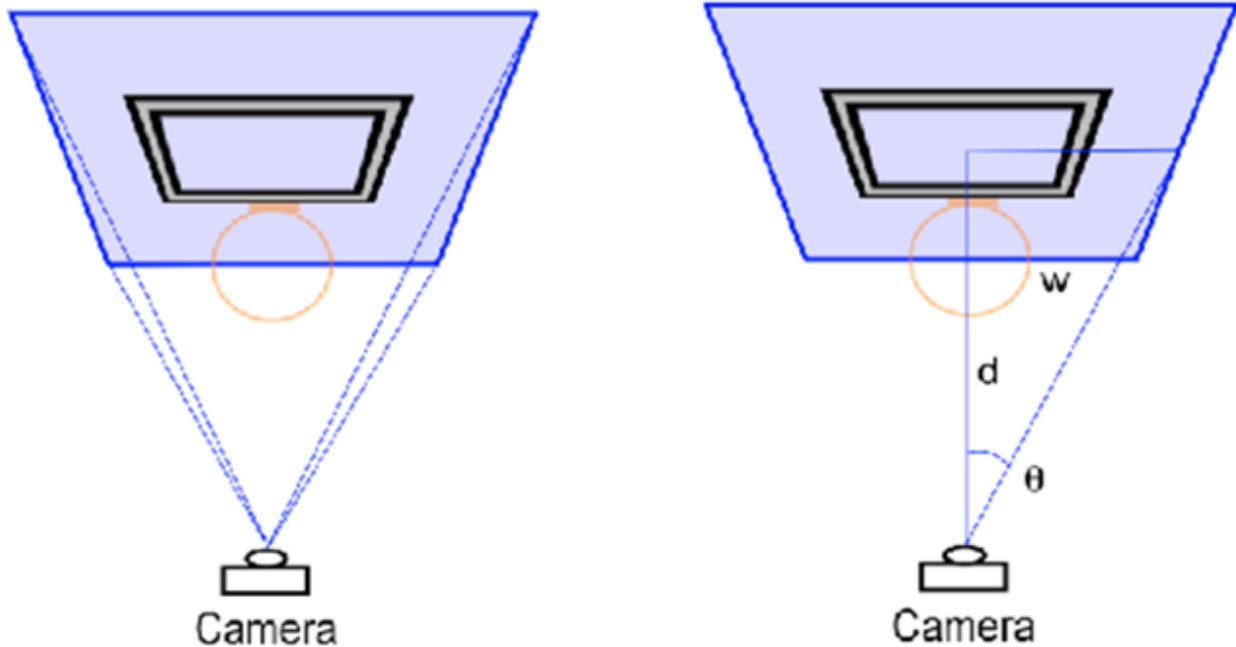
### 19.6.2 Distance

The target distance is computed with knowledge about the target size and the camera optics. The approach uses information about the camera lens view angle and the width of the camera field of view. Shown below-left, a given camera takes in light within the blue pyramid extending from the focal point of the lens. Unless the lens is modified, the view angle is constant and equal to  $2\theta$ . As shown to the right, the values are related through the trigonometric relationship of ...

$$\tan\theta = w/d$$

The datasheets for the cameras can be found at the following URLs: [Axis 206](#), [AxisM1011](#), [Axis M1013](#), [Lifecam HD3000](#). These give rough horizontal view angles for the lenses. Remember that this is for entire field of view, and is therefore  $2\theta$ . This year's code uses the vertical field-of-view and it is therefore highly recommend to perform

calibration (as described in the next article) to determine the appropriate view angle for your camera (empirically determined values for each camera type are included in the code as a reference).



### 19.6.3 Distance Continued

The next step is to use the information we have about the target to find the width of the field of view the blue rectangle shown above. This is possible because we know the target rectangle size in both pixels and feet, and we know the FOV rectangle width in pixels. We can use the relationships of ...

$$T_{ft}/T_{pixel} = FOV_{ft}/FOV_{pixel} \text{ and } FOV_{ft} = 2*w = 2*d*\tan\Theta$$

to create an equation to solve for  $d$ , the distance from the target:

$$d = T_{ft}*FOV_{pixel}/(2*T_{pixel}*\tan\Theta)$$

Notice that the datasheets give approximate view angle information. When testing, it was found that the calculated distance to the target tended to be a bit short. Using a tape measure to measure the distance and treating the angle as the unknown it was found that view angles of 41.7 for the 206, 37.4 for the M1011, and 49 for the M1013 gave better results. Information on performing your own distance calibration is included in the next article.



---

## Configuring an Axis Camera

---

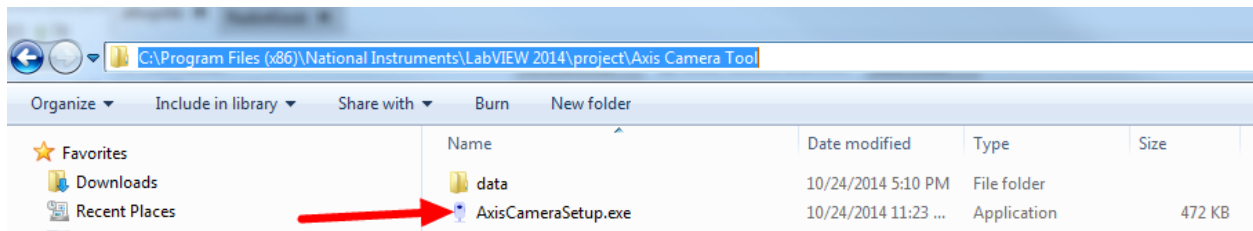
**Note:** Three different Axis camera models are supported by the FRC software, the Axis 206, Axis M1011 and Axis M1013. This document provides instructions on how to configure one of these cameras for FRC use. To follow the instructions in this document, you must have installed the NI FRC Update Suite and configured your radio

---

### 20.1 Connect the camera

Connect the Axis camera to the DAP-1522 radio using an Ethernet cable. Connect your computer to the radio using an ethernet cable or via a wireless connection.

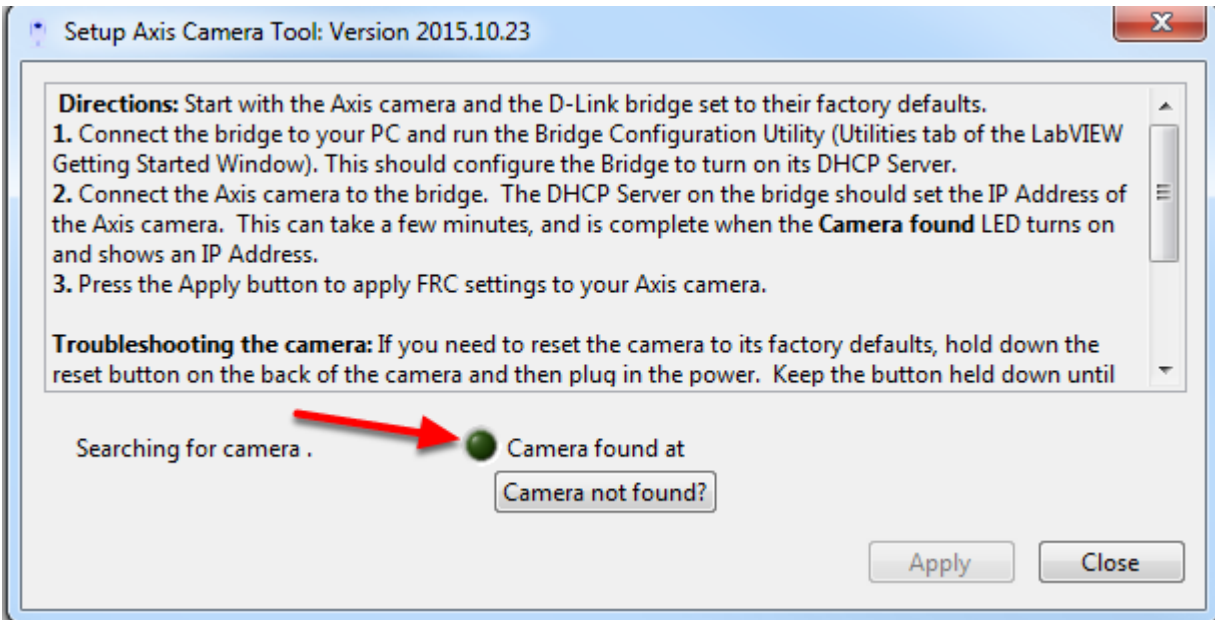
### 20.2 Axis Camera Setup Tool



Browse to **C:\Program Files (x86)\National Instruments\LabVIEW 2014\project\Axis Camera Tool** and double-click on **AxisCameraSetup.exe** to start the Axis Camera Setup Tool.

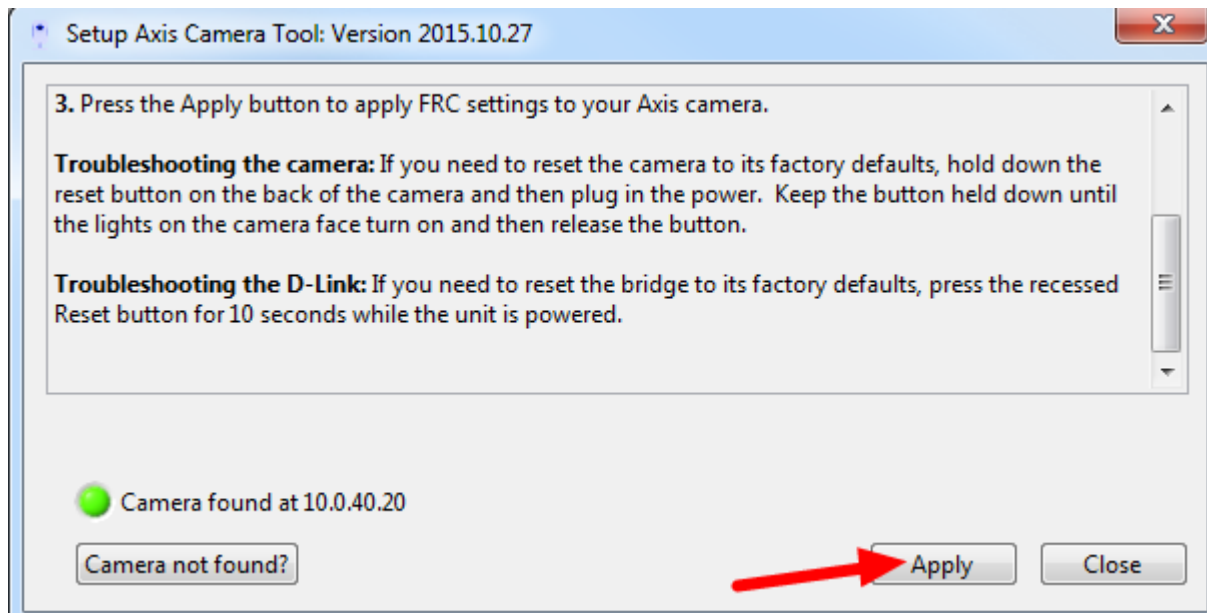
#### 20.2.1 Tool Overview

The camera should be automatically detected and the green indicator light should be lit. If it is not, make sure the camera is powered on (the ring on the camera face should be green) and connected to your computer. If the indicator



remains off follow the instructions in the tool textbox next to Troubleshooting the camera to reset the camera. You can also use the Camera not found? button to check the IP address of your computer, one of the addresses listed should be of the form 10.TE.AM.XX where TEAM is your 4 digit team number.

## 20.2.2 Setup the Camera



To configure the camera, press Apply. This will configure many of the necessary/recommended settings for using the camera for FRC. Currently the tool does not properly configure the DNS name of the camera in many cases.



**AXIS**  
COMMUNICATIONS

Configure Root Password

User name: root

Password:

Confirm password:

OK

The password for the pre-configured administrator root must be changed before the product can be used.

If the password for root is lost, the product must be reset to the factory default settings, by pressing the button located in the product's casing. Please see the user documentation for more information.

### 20.2.3 Camera Webpage

To set the network settings, open a web browser and enter the address shown next to Camera found at in the tool (in the example above this is 10.0.40.20) in the address bar and press enter. You should see a Configure Root Password page, set this password to whatever you would like, but admin is recommended.

### 20.2.4 Setup Page

Click Setup to go to the setup page.

### 20.2.5 Configure Basic Network Settings

To configure the network settings of the camera, click the arrow to expand the System Options pane, then click the arrow to expand Network, then expand TCP/IP and select Basic. Set the camera to obtain an IP address via DHCP by selecting the bubble. Alternately, you may choose to set a static IP in the range 10.TE.AM.3 to 10.TE.AM.19. This is outside the range handed out by the DAP-1522 radio (home use) or FMS system (event use) so you will avoid any IP conflicts.

Click Save.

### 20.2.6 Configure Advanced Network Settings

Next click Advanced under TCP/IP. Set the Host Name Configuration to "Use the host name:" and set the value to "axis-camera" as shown. If you plan to use multiple cameras on your robot, select a unique host name for each. You will need to modify the dashboard and/or robot code to work with the additional cameras and unique host names.

Click Save.



## 20.3 Manual Camera Configuration

It is recommended to use the Setup Axis Camera Tool to configure the Axis Camera. If you need to configure the camera manually, connect the camera directly to the computer, configure your computer to have a static IP of 192.168.0.5, then open a web browser and enter 192.168.0.90 in the address bar and press enter. You should see a Configure Root Password page, set this password to whatever you would like, but admin is recommended.

If you do not see the camera webpage come up, you may need to reset the camera to factory defaults. To do this, remove power from the camera, hold the reset button while applying power to the camera and continue holding it until the lights on the camera face turn on, then release the reset button and wait for the lights to turn green. The camera is now reset to factory settings and should be accessible via the 192.168.0.5 address.

### 20.3.1 Setup Page

Click Setup to go to the setup page.

### 20.3.2 Configure Users

On the left side click Users to open the users page. Click Add then enter the Username FRC Password FRC and click the Administrator bubble, then click OK. If using the SmartDashboard, check the Enable anonymous viewer login box. Then click Save.

The screenshot displays the web interface for an AXIS 206 Network Camera. The top navigation bar includes the AXIS logo, the camera model name "AXIS 206 Network Camera", and links for "Live View", "Setup", and "Help". A left-hand sidebar contains a menu with categories: "Basic Configuration", "Video & Image", "Live View Config", "System Options" (with sub-items: Users, Date & Time, Network, TCP/IP, and "Basic" which is highlighted with a circled '1'), "Advanced", "QoS", "SMTP (email)", "UPnP", "Bonjour", "LED settings", "Maintenance", "Support", and "Advanced". Below the menu are "Language" (with a US flag icon) and "About".

The main content area is titled "Basic TCP/IP Settings" and includes a help icon. It is divided into three sections:

- Network Settings:** A "View" button to see current settings.
- IP Address Configuration:** Two radio button options:
  - Option 1 (circled '2'): "Obtain IP address via DHCP" (selected).
  - Option 2: "Use the following IP address:" with three input fields:
    - IP address: 192.168.0.90
    - Subnet mask: 255.255.255.0
    - Default router: 192.168.0.1A "Test" button is located to the right of the IP address field.
- Services:**
  - A checked checkbox for "Enable ARP/Ping setting of IP Address".
  - A button for "Options for notification of IP address change" labeled "Settings...".
  - A button for "AXIS Internet Dynamic DNS Service" labeled "Settings...".

At the bottom of the settings area, there are "Save" and "Reset" buttons, with a circled '3' next to the "Save" button.

**AXIS** COMMUNICATIONS **AXIS 206 Network Camera** [Live View](#) | [Setup](#) | [Help](#)

## Advanced TCP/IP Settings

- Basic Configuration
- Video & Image
- Live View Config
- System Options
  - Users
  - Date & Time
  - Network
    - TCP/IP
      - Basic
      - Advanced**
      - QoS
      - SMTP (email)
      - UPnP
      - Bonjour
      - LED settings
      - Maintenance
    - Support
    - Advanced
- Language
- About

### DNS Configuration

Obtain DNS server address via DHCP [View](#)

Use the following DNS server address:

Domain name:  (use ; to separate names)

Primary DNS server:

Secondary DNS server:

### NTP Configuration

Obtain NTP server address via DHCP [View](#)

Use the following NTP server address:

Network address:  (host name or IP address)

### Host Name Configuration

Obtain host name via DHCP [View](#)

Use the host name:

Enable dynamic DNS updates

Register DNS name:  (Axisproduct.example.com)

TTL:

### Link-Local Address

Auto-Configure Link-Local Address [View](#)

### HTTP

HTTP port:

### NAT traversal (port mapping)

NAT traversal is disabled. [Enable](#)

Use manually selected NAT router:  (LAN IP address)

Alternative HTTP port:  \*

\* If left blank, a port number will be set automatically upon enable.

### FTP

Enable FTP server

### Network Traffic

Connection type:

[Save](#) [Reset](#)



## Configure Root Password

User name: root

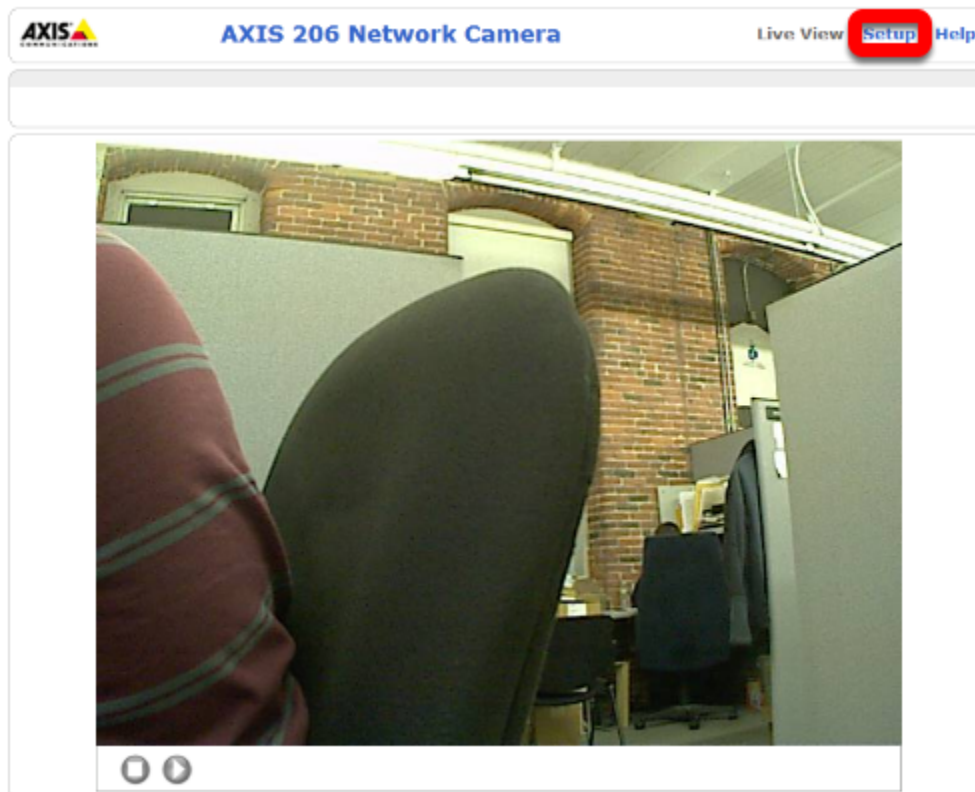
Password:

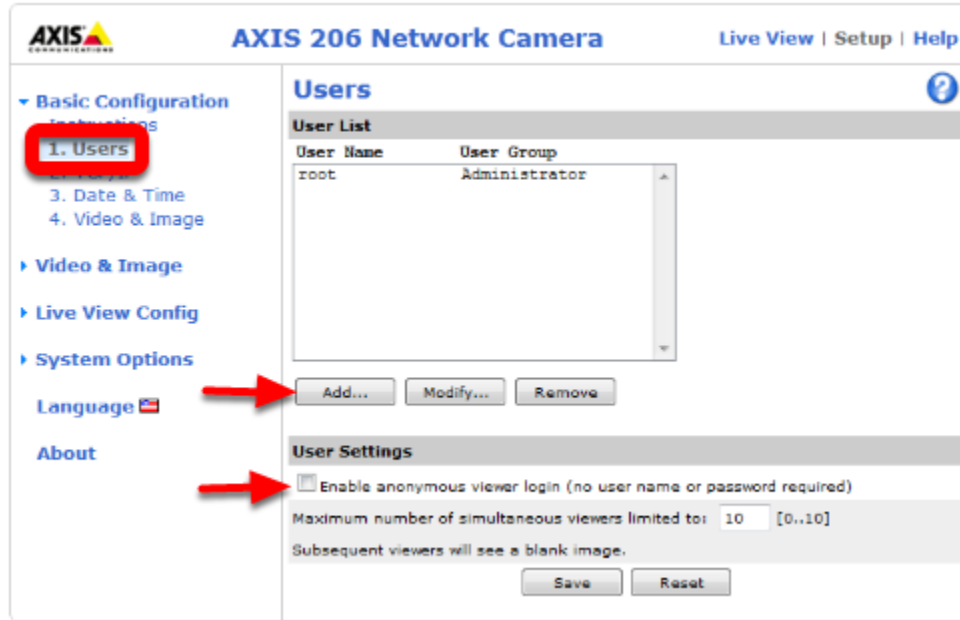
Confirm password:

---

The password for the pre-configured administrator root must be changed before the product can be used.

If the password for root is lost, the product must be reset to the factory default settings, by pressing the button located in the product's casing. Please see the user documentation for more information.





### 20.3.3 Configure Image Settings

Click Video & Image on the left side to open the image settings page. Set the Resolution and Compression to the desired values (recommended 320x240, 30). To limit the framerate to under 30 FPS, select the Limited to bubble under Maximum frame rate and enter the desired rate in the box. Color, Brightness and Sharpness may also be set on this screen if desired. Click Save when finished.

### 20.3.4 Configure Basic Network Settings

To configure the network settings of the camera, click the arrow to expand the System Options pane, then click the arrow to expand Network, then expand TCP/IP and select Basic. Set the camera to obtain an IP address via DHCP by selecting the bubble. Alternately, you may choose to set a static IP in the range 10.TE.AM.3 to 10.TE.AM.19. This is outside the range handed out by the DAP-1522 radio (home use) or FMS system (event use) so you will avoid any IP conflicts.


Click Save.


### 20.3.5 Configure Advanced Network Settings

Next click Advanced under TCP/IP. Set the Host Name Configuration to “Use the host name:” and set the value to “axis-camera” as shown. If you plan to use multiple cameras on your robot, select a unique host name for each. You will need to modify the dashboard and/or robot code to work with the additional cameras and unique host names.

Click Save.




**AXIS 206 Network Camera**
Live View | Setup | Help

- ▶ Basic Configuration
- ▶ Video & Image
  - Video & Image
  - Advanced
- ▶ Live View Config
- ▶ System Options
- Language 
- About

### Image Settings ?

**Image Appearance**

Resolution:  pixels

Compression:  [0..100]

Rotate image:  degrees

Color level:  [0..100] \*

Brightness:  [0..100] (Does not affect Test image)

Sharpness:  (Does not affect Test image)

\* Changes to color level do not affect Test image (exception 0 = B/W)

**Overlay Settings**

Include date     Include time

Include text:  (Does not affect Test image)

Place text/date/time at  of image

**Video Stream**

Maximum video stream time:

Unlimited

Limited to  [1..] seconds per session

Maximum frame rate:

Unlimited

Limited to  [1..30] fps per viewer

**Test**

Test settings before saving.

**AXIS 206 Network Camera** Live View | Setup | Help

**Basic TCP/IP Settings** ?

**Network Settings**

View current network settings:

**IP Address Configuration**

2  Obtain IP address via DHCP

Use the following IP address:

IP address:

Subnet mask:

Default router:

**Services**

Enable ARP/Ping setting of IP Address

Options for notification of IP address change

AXIS Internet Dynamic DNS Service

3

**Navigation Menu:**

- Basic Configuration
- Video & Image
- Live View Config
- System Options
  - Users
  - Date & Time
  - Network
    - 1 **TCP/IP**
      - 1 **Basic**
      - Advanced
      - QoS
      - SMTP (email)
      - UPnP
      - Bonjour
      - LED settings
      - Maintenance
    - Support
    - Advanced
  - Language
  - About

**AXIS** COMMUNICATIONS **AXIS 206 Network Camera** [Live View](#) | [Setup](#) | [Help](#)

## Advanced TCP/IP Settings

- Basic Configuration
- Video & Image
- Live View Config
- System Options
  - Users
  - Date & Time
  - Network
    - TCP/IP
      - Basic
      - Advanced**
      - QoS
      - SMTP (email)
      - UPnP
      - Bonjour
      - LED settings
      - Maintenance
    - Support
    - Advanced
- Language
- About

### DNS Configuration

Obtain DNS server address via DHCP [View](#)

Use the following DNS server address:

Domain name:  (use ; to separate names)

Primary DNS server:

Secondary DNS server:

### NTP Configuration

Obtain NTP server address via DHCP [View](#)

Use the following NTP server address:

Network address:  (host name or IP address)

### Host Name Configuration

Obtain host name via DHCP [View](#)

Use the host name:

Enable dynamic DNS updates

Register DNS name:  (Axisproduct.example.com)

TTL:

### Link-Local Address

Auto-Configure Link-Local Address [View](#)

### HTTP

HTTP port:

### NAT traversal (port mapping)

NAT traversal is disabled. [Enable](#)

Use manually selected NAT router:  (LAN IP address)

Alternative HTTP port:  \*

\* If left blank, a port number will be set automatically upon enable.

### FTP

Enable FTP server

### Network Traffic

Connection type:

[Save](#) [Reset](#)



---

## Using the Microsoft Lifecam HD-3000

---

The Microsoft Lifecam HD-3000 is a USB webcam that was tested with the roboRIO as part of the Beta testing and software development effort. While other USB webcams may work with the roboRIO, this camera has been tested to be compatible with the provided software.

### 21.1 Connecting the camera to the roboRIO

The camera can be connected to either of the roboRIO USB ports.

### 21.2 Using the camera - LabVIEW

To stream the camera back to the Dashboard using LabVIEW, no additional code is necessary. Simply select USB HW (image compression done by the camera, fewer options but lower roboRIO CPU usage) or USB SW (image compressed by roboRIO, more options, but higher roboRIO CPU usage) and the image should begin streaming back.

---

**Note:** The camera should be plugged in before your LabVIEW code starts running to work properly. If you just plugged in the camera rebooting the roboRIO is a quick way to make sure it is recognized properly.

---

The default LabVIEW templates and the image processing examples are already set up for the USB camera if you want to do image processing. On the LabVIEW splash screen, click Tutorials, then click Tutorial 8 for more information about integrating Vision processing in your LabVIEW code.

### 21.3 Using the camera - C++/Java

To stream the camera back to the Dashboard using C++ or Java robot code, you will need to add some code to your robot project. Example have been provided in Eclipse to illustrate the use of the CameraServer class to automatically stream images back to the Dashboard (SimpleVision) and send back modified images (IntermediateVision and the



2015 Vision examples). This class will allow images to be streamed back to either the SmartDashboard “USB Webcam Viewer” or the LabVIEW Dashboard (set to USB HW).

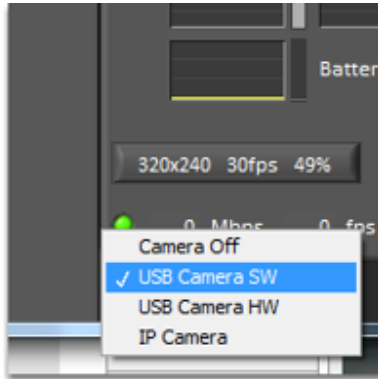
### 21.3.1 Determining the Camera name

Unlike the LabVIEW code which attempts to determine the camera name of the camera you want to use, the C++Java code requires you to specify the camera name. To determine the name of the desired camera, you will need to use the roboRIO webdashboard. For more information about accessing the roboRIO webdashboard see RoboRIO Webdashboard. Open the roboRIO webdashboard and locate the camera in the left pane and note the Name, this is the string you will pass into the camera server or IMAQDx open (in the image above, the camera name is cam0).

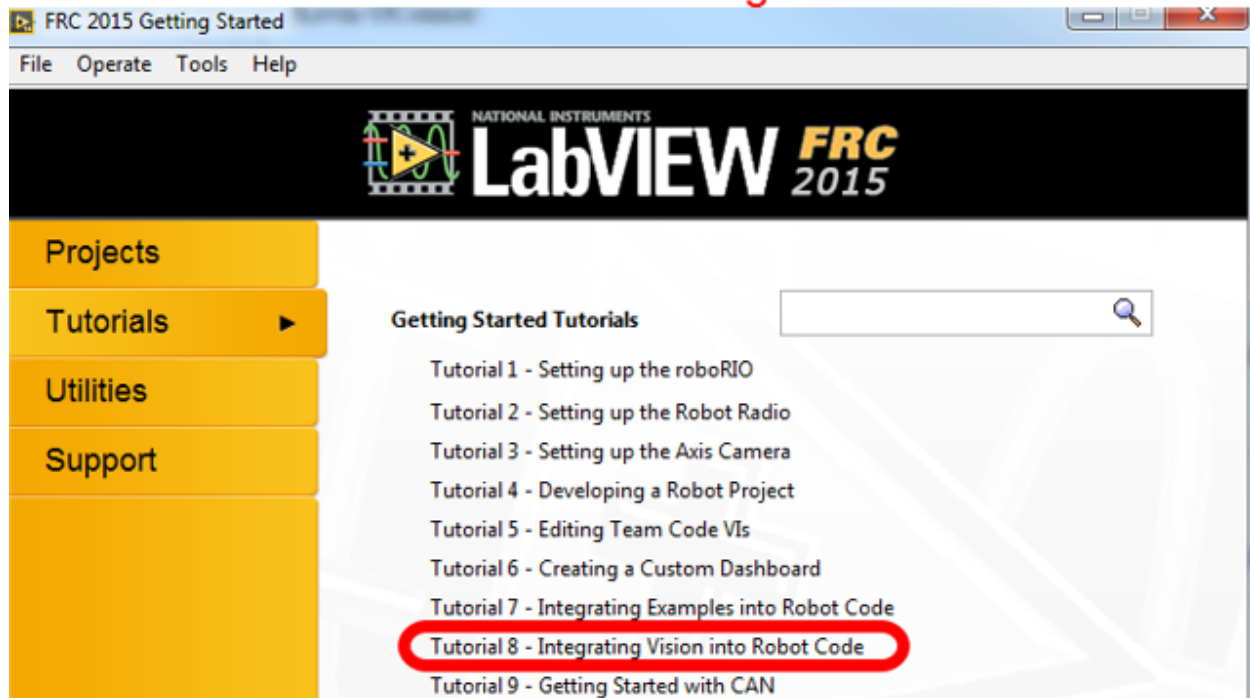
### 21.3.2 Using the SmartDashboard USB Camera Viewer

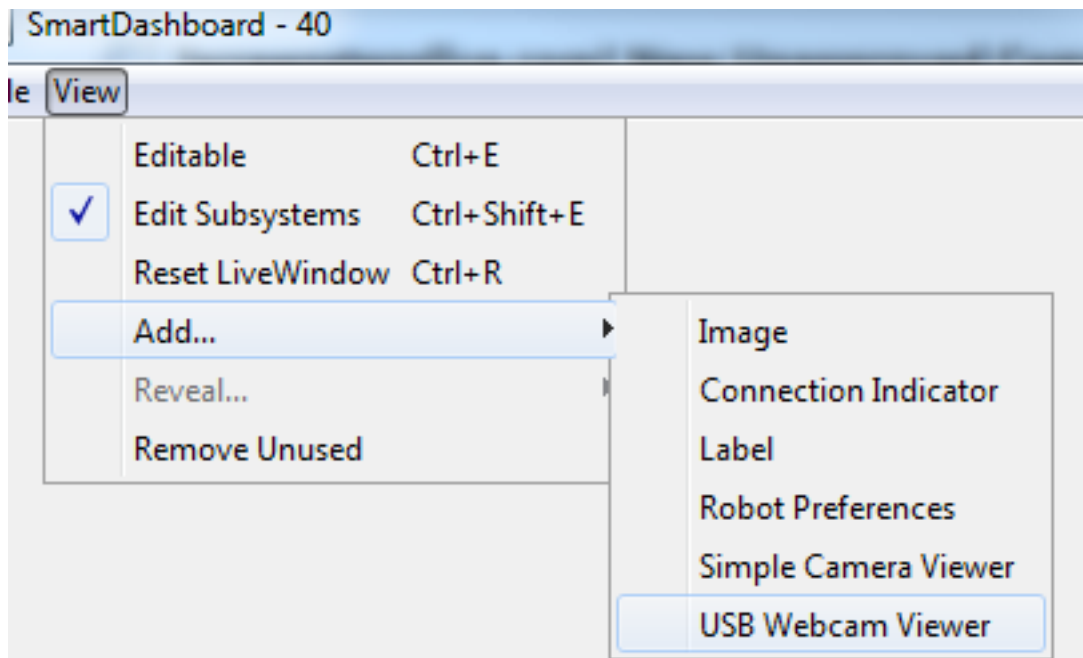
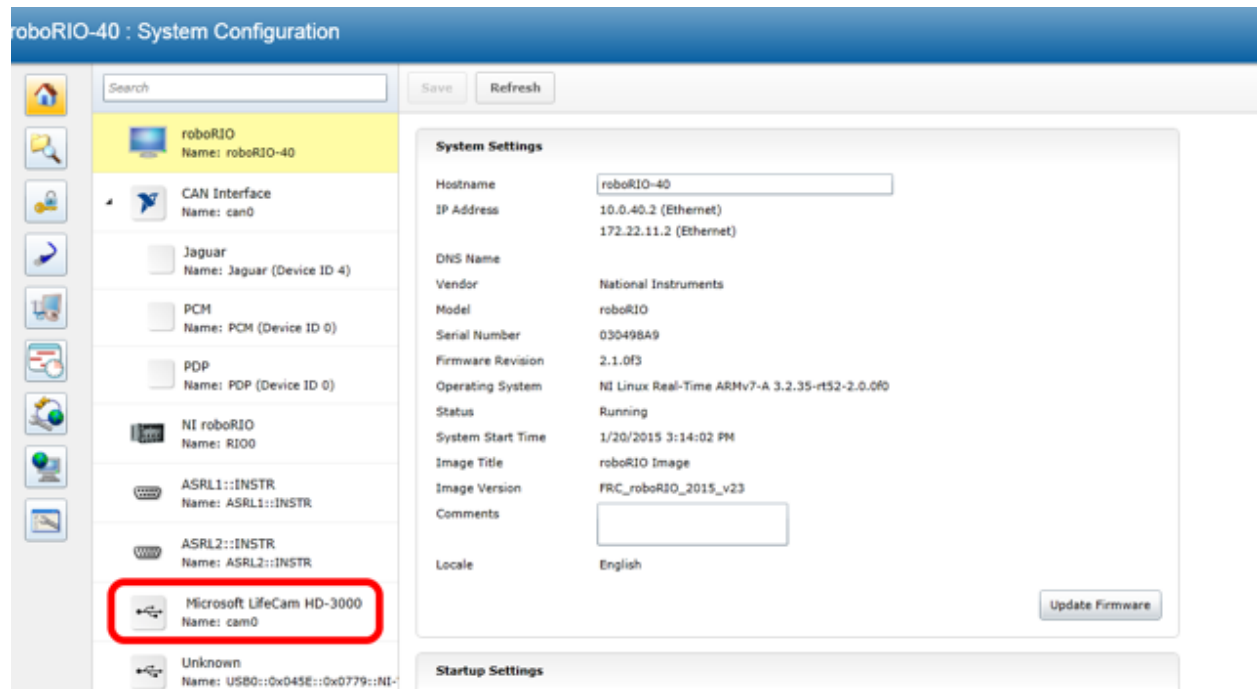
To view the camera stream from the LabVIEW dashboard, set the camera dropdown to USB HW. To view the stream from the SmartDashboard you will need to add a USB Webcam Viewer widget to your layout. For more information on the SmartDashboard and widgets see the SmartDashboard manual. To add the USB Webcam Viewer widget to the SmartDashboard, select View -> Add -> USB Webcam Viewer. To move or resize the viewer widget, make the layout Editable by selecting that option in the View menu (select again to disable).

Dashboard  
Stream



Vision Processing







It is very difficult to achieve good image processing results without good images. With a light mounted near the camera lens, you should be able to use the provided examples, the dashboard or SmartDashboard, NI Vision Assistant or a web browser to view camera images and experiment with camera settings.

### 22.1 Changing Camera Settings

To change the camera settings on any of the supported Axis cameras (206, M1011, M1013), browse to the camera's webpage by entering its address (usually 10.TE.AM.11) in a web browser. Click Setup near the top right corner of the page. On the M1013, the settings listed below are split between the Video Stream page and the Camera Settings page, both listed under the Video section.

### 22.2 Focus

The Axis M1011 has a fixed-focus lens and no adjustment is needed. The Axis 206 camera has a black bezel around the lens that rotates to move the lens in and out and adjust focus. The Axis M103 has a silver and black bezel assembly around the lens to adjust the focus. Ensure that the images you are processing are relatively sharp and focused for the distances needed on your robot.

### 22.3 Compression

The Axis camera returns images in BMP, JPEG, or MJPG format. BMP images are quite large and take more time to transmit to the cRIO and laptop. Therefore the WPILib implementations typically use MJPG motion JPEG. The compression setting ranges from 0 to 100, with 0 being very high quality images with very little compression, and 100 being very low quality images with very high compression. The camera default is 30, and it is a good compromise, with few artifacts that will degrade image processing. Due to implementation details within the VxWorks memory manager, you may notice a performance benefit if you keep the image sizes consistently below 16 kBytes. **Teams are**

- ▶ **Basic Setup** M1013
- ▶ **Video**
  - Video Stream
  - Stream Profiles
  - Camera Settings
  - Overlay Image
  - Privacy Mask
- ▶ **Live View Config**
- ▶ **Detectors**
- ▶ **Events**
- ▶ **Recordings**
- ▶ **System Options**
- About

### Camera Settings ?

**View Area**

 Enable View Area
 

**Image Appearance**

Color level:  50 [0..100]

Brightness:  50 [0..100]

Sharpness:  50 [0..100]

Contrast:  50 [0..100]

**White Balance**

White balance: Fixed Outdoor 1 ▼

**Exposure Settings**

Exposure value:  50 [0..100]

Enable Backlight compensation:

Exposure priority: Default ▼

**View Image Settings**

---

### Image Settings ?

**Image Appearance**

Resolution: 640x480 ▼ pixels

Compression: 30 [0..100]

Rotate image: 0 ▼ degrees

Color level: 50 [0..100] \*

Brightness: 50 [0..100] (Does not affect Test image)

Sharpness: 0 ▼ (Does not affect Test image)

\* Changes to color level do not affect Test image (exception 0 = B/W)

**Overlay Settings**

Include date     Include time

Include text:  (Does not affect Test image)

Place text/date/time at top ▼ of image

**Video Stream**

- ▶ **Basic Configur** 206
- ▶ **Video & Image** M1011
  - Video & Image
  - Advanced
- ▶ **Live View Config**
- ▶ **System Options**
- Language
- About



320x240 Color Image:  
Compression set to 0. Image file size is 20,715 bytes.  
High quality, but large in size. May be slower to compress and decompress - which impacts frame rate.



320x240 Color Image:  
Compression set to 30. Image file size is 6,450 bytes. Good quality, relatively small in size. Some image artifacts are present on edges.



320x240 Color Image:  
Compression set to 100. Image file size is 2,222 bytes. Poor quality for processing. Notice blocky artifacts and rough edges.

Image Appearance	
Resolution:	640x480 (4:3) pixels
Compression:	30 [0..100]

advised to consider how the compression setting on the camera affects bandwidth if performing processing on the Driver Station computer, see the FMS Whitepaper for more details.

## 22.4 Resolution

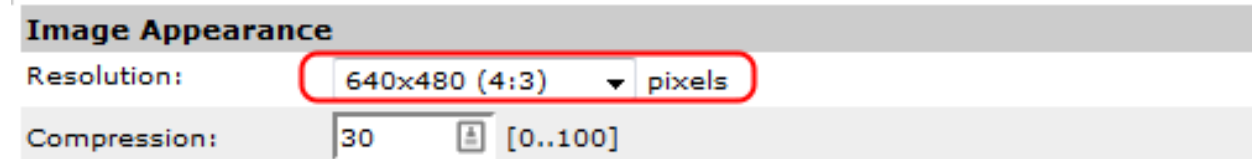


Image sizes shared by the supported cameras are 160x120, 320x240, and 640x480. The M1011 and 1013 have additional sizes, but they aren't built into WPILib. The largest image size has four times as many pixels that are one-fourth the size of the middle size image. The large image has sixteen times as many pixels as the small image.

The tape used on the target is 4 inches wide, and for good processing, you will want that 4 inch feature to be at least two pixels wide. Using the distance equations above, we can see that a medium size image should be fine up to the point where the field of view is around 640 inches, a little over 53 feet, which is nearly double the width of the FRC field. This occurs at around 60 feet away, longer than the length of the field. The small image size should be usable for processing to a distance of about 30 feet or a little over mid-field.

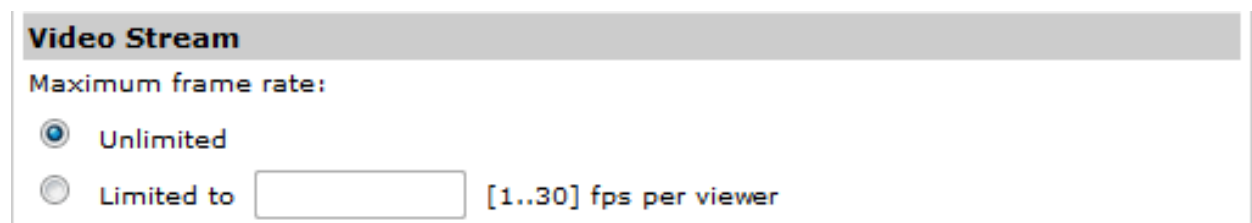
Image size also impacts the time to decode and to process. Smaller images will be roughly four times faster than the next size up. If the robot or target is moving, it is quite important to minimize image processing time since this will add to the delay between the target location and perceived location. If both robot and target are stationary, processing time is typically less important.

---

**Note:** When requesting images using LabVIEW (either the Dashboard or Robot Code), the resolution and Frame Rate settings of the camera will be ignored. The LabVIEW code specifies the framerate and resolution as part of the stream request (this does not change the settings stored in the camera, it overrides that setting for the specific stream). The SmartDashboard and robot code in C++ or Java will use the resolution and framerate stored in the camera.

---

## 22.5 Frame Rate



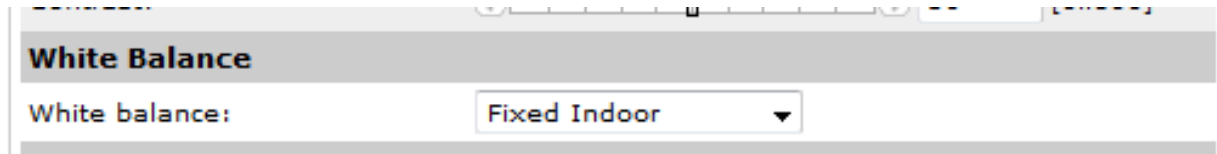
The Axis Cameras have a max framerate of 30 frames per second. If desired, a limit can be set lower to reduce bandwidth consumption.

## 22.6 Color Enable

The Axis cameras typically return color images, but are capable of disabling color and returning a monochrome or grayscale image. The resulting image is a bit smaller in file size, and considerably quicker to decode. If processing is

carried out only on the brightness or luminance of the image, and the color of the ring light is not used, this may be a useful technique for increasing the frame rate or lowering the CPU usage.

## 22.7 White Balance



If the color of the light shine is being used to identify the marker, be sure to control the camera settings that affect the image coloring. The most important setting is white balance. It controls how the camera blends the component colors of the sensor in order to produce an image that matches the color processing of the human brain. The camera has five or six named presets, an auto setting that constantly adapts to the environment, and a hold setting – for custom calibration.

The easiest approach is to use a named preset, one that maintains the saturation of the target and doesn't introduce problems by tinting neutral objects with the color of the light source.

To custom-calibrate the white balance, place a known neutral object in front of the camera. A sheet of white paper is a reasonable object to start with. Set the white balance setting to auto, wait for the camera to update its filters (ten seconds or so), and switch the white balance to hold.

## 22.8 Exposure

The brightness or exposure of the image also has an impact on the colors being reported. The issue is that as overall brightness increases, color saturation will start to drop. Lets look at an example to see how this occurs. A saturated red object placed in front of the camera will return an RGB measurement high in red and low in the other two e.g. (220, 20, 30). As overall white lighting increases, the RGB value increases to (240, 40, 50), then (255, 80, 90), then (255, 120, 130), and then (255, 160, 170). Once the red component is maximized, additional light can only increase the blue and green, and acts to dilute the measured color and lower the saturation. If the point is to identify the red object, it is useful to adjust the exposure to avoid diluting your principle color. The desired image will look somewhat dark except for the colored shine.

There are two approaches to control camera exposure times. One is to allow the camera to compute the exposure settings automatically, based on its sensors, and then adjust the camera's brightness setting to a small number to lower the exposure time. The brightness setting acts similar to the exposure compensation setting on SLR cameras. The other approach is to calibrate the camera to use a custom exposure setting. To do this on a 206 or M1011, change the exposure setting to auto, expose the camera to bright lights so that it computes a short exposure, and then change the exposure setting to hold. Both approaches will result in an overall dark image with bright saturated target colors that stand out from the background and are easier to mask.

The M1013 exposure settings look a little different. The Enable Backlight compensation option is similar to the Auto exposure settings of the M1011 and 206 and you will usually want to un-check this box. Adjust the Brightness and Exposure value sliders until your image looks as desired. The Exposure Priority should generally be set to Motion. This will prioritize framerate over image quality. Note that even with these settings the M1013 camera still performs some auto exposure compensation so it is recommended to check calibration frequently to minimize any impact lighting changes may have on image processing. See the article on Calibration for more details.

<b>Image Appearance</b>	<b>M1013</b>
Color level:	<input type="range" value="50"/> 50 [0..100]
Brightness:	<input type="range" value="0"/> 0 [0..100]
Sharpness:	<input type="range" value="50"/> 50 [0..100]
Contrast:	<input type="range" value="50"/> 50 [0..100]
<b>White Balance</b>	
White balance:	Fixed Indoor ▼
<b>Exposure Settings</b>	
Exposure value:	<input type="range" value="0"/> 0 [0..100]
Enable Backlight compensation:	<input type="checkbox"/>
Exposure priority:	Motion ▼
<b>Lighting Conditions</b>	<b>206/M1011</b>
White balance:	Automatic ▼
Exposure control:	Automatic ▼
<b>Low Light Behavior</b>	
Exposure priority:	None ▼

While many of the numbers for the Vision Processing code can be determined theoretically, there are a few parameters that are typically best to measure empirically then enter back into the code (a process typically known as calibration). This article will show how to perform calibration for the Color (masking), and View Angle (distance) using the NI Vision Assistant. If you are using C++ or Java and have not yet installed the NI Vision Assistant, see the article [Installing NI Vision Assistant](#).

### 23.1 Enable Snapshots

To capture snapshots from the Axis camera, you must first enable the Snapshot button. Open a web-browser and browse to camera's address (10.TE.AM.11), enter the Username/Password combo FRC/FRC if prompted, then click Setup->Live View Config->Layout. Click on the checkbox to Show snapshot button then click Save.

### 23.2 Check Camera Settings

Depending on how you are capturing the image stream in your program, it may be possible to stream a different resolution, framerate and/or compression than what is saved in the camera and used in the Live View. Before performing any calibration it is recommended you verify that the settings in the camera match the settings in your code. To check the settings in the camera, click on the Video and Image header on the left side of the screen, then click Video and Image.

### 23.3 Capture Images

Click the Live View button to return to the Live View page and you should now see a Snapshot button. Clicking this button opens a pop-up window with a static image capture. Right-click on this image, select Save Image as and select your desired location and file name, then save the image.

**AXIS M1013 Network Camera** Live View | **Setup** | Help

**Live View Layout**

**Stream Profile**  
 Stream profile: Motion JPEG

Show stream profile selection

**Default Viewer**  
 Windows Internet Explorer: AMC (ActiveX)  
 Other Browsers: Server push

**Note:** QuickTime is only used with H.264. Motion JPEG will be shown with AMC in Windows Internet Explorer and with server push in other browsers.

**Viewer Settings**

Show viewer toolbar  
 Enable H.264 decoder installation  
 Show crosshair in PTZ joystick mode\*  
 Use PTZ joystick mode as default\*  
 Enable recording button

\* Not applicable to AMC (ActiveX).

**Action Buttons**


Show manual trigger button  
 Show snapshot button

**User Defined Links**

<input type="checkbox"/> Show custom link 1	Use as: <input checked="" type="radio"/> cgi link <input type="radio"/> web link
Name: Custom link 1	URL: http://
<input type="checkbox"/> Show custom link 2	Use as: <input checked="" type="radio"/> cgi link <input type="radio"/> web link
Name: Custom link 2	URL: http://
<input type="checkbox"/> Show custom link 3	Use as: <input checked="" type="radio"/> cgi link <input type="radio"/> web link
Name: Custom link 3	URL: http://
<input type="checkbox"/> Show custom link 4	Use as: <input checked="" type="radio"/> cgi link <input type="radio"/> web link
Name: Custom link 4	URL: http://

Save Reset



**AXIS**  **AXIS 206 Network Camera** Live View | Setup | Help

[Basic Configuration](#)  
[Video & Image](#)  
Video & Image  
[Advanced](#)  
[Live View Config](#)  
[System Options](#)  
[Language !\[\]\(581a37922a09af6d3412377716caf230\_img.jpg\)](#)  
[About](#)

### Image Settings

**Image Appearance**

Resolution:  pixels

Compression:  [0..100]

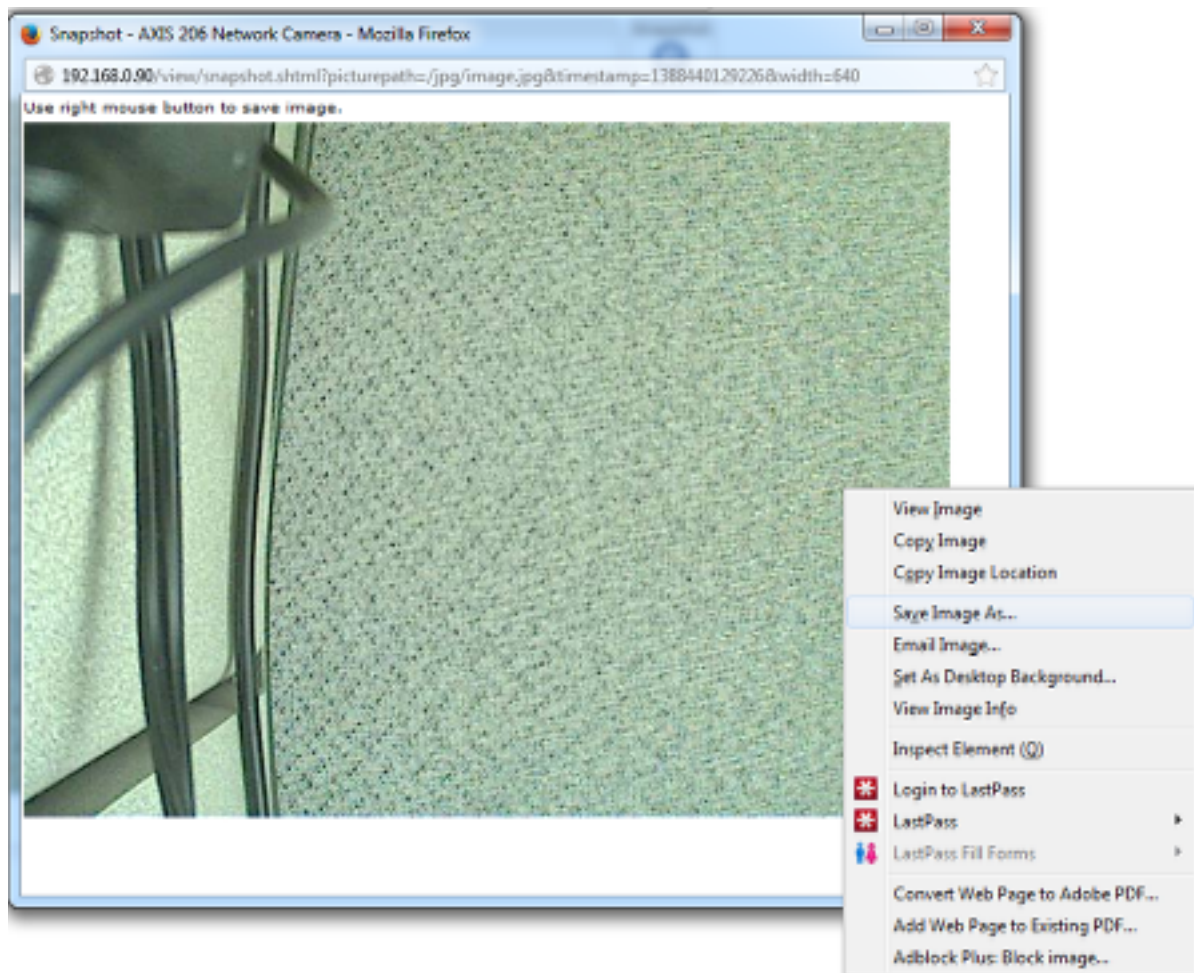
Rotate image:  degrees

Color level:  [0..100] \*

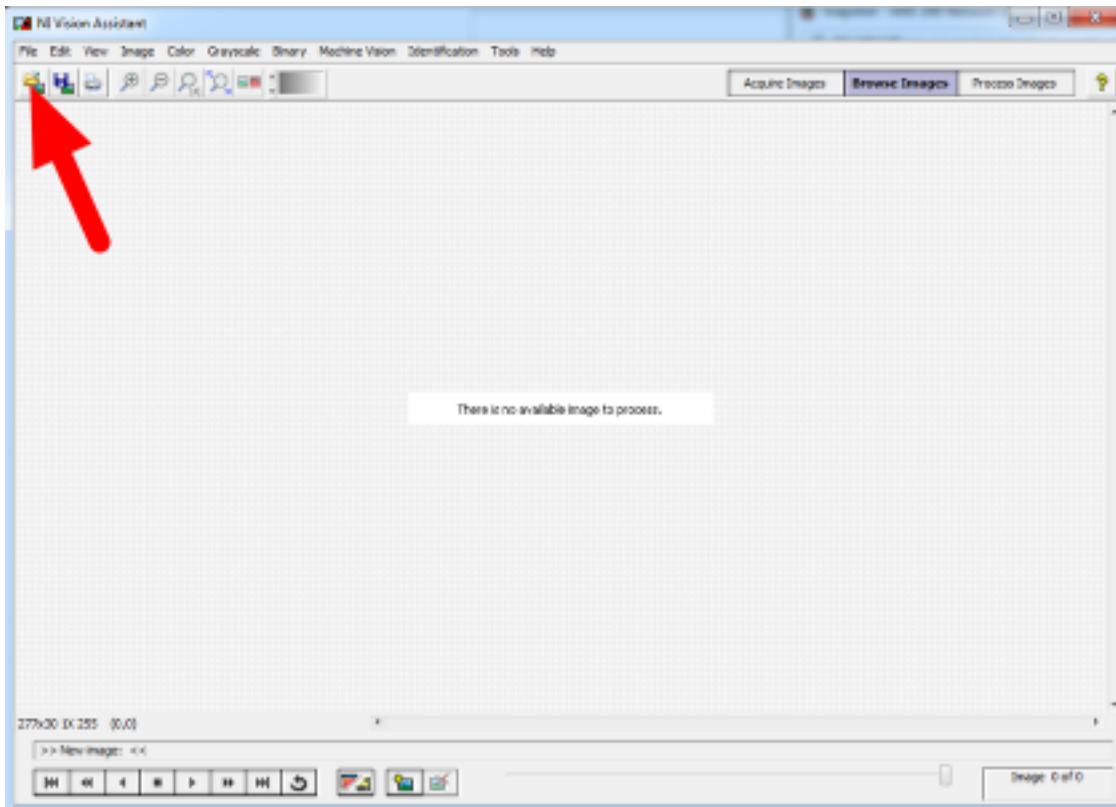
Brightness:  [0..100] (Does not affect Test image)

Sharpness:  (Does not affect Test image)

\* Changes to color level do not affect Test image (exception 0 = B/W)



## 23.4 Load Image(s) in Vision Assistant



Open the NI Vision Assistant and select the Browse Images option. Select the Open Images icon in the top left of the Toolbar, then locate your images. Repeat as necessary to load all desired images.

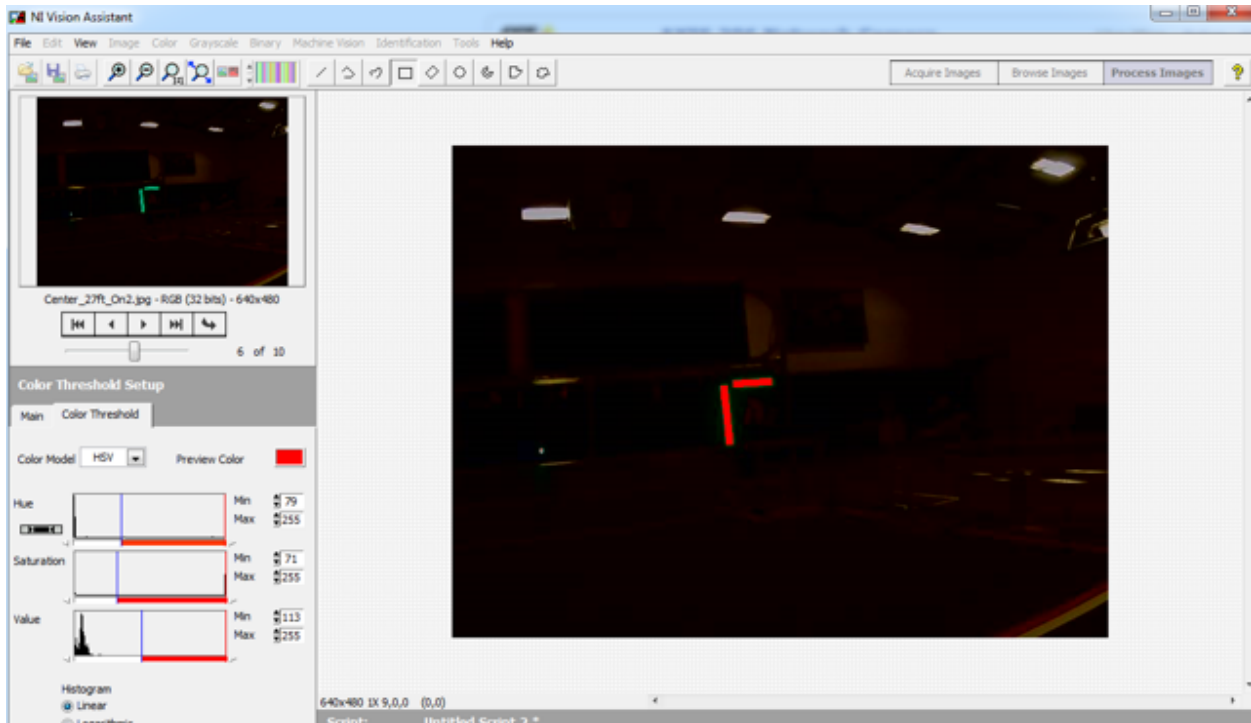
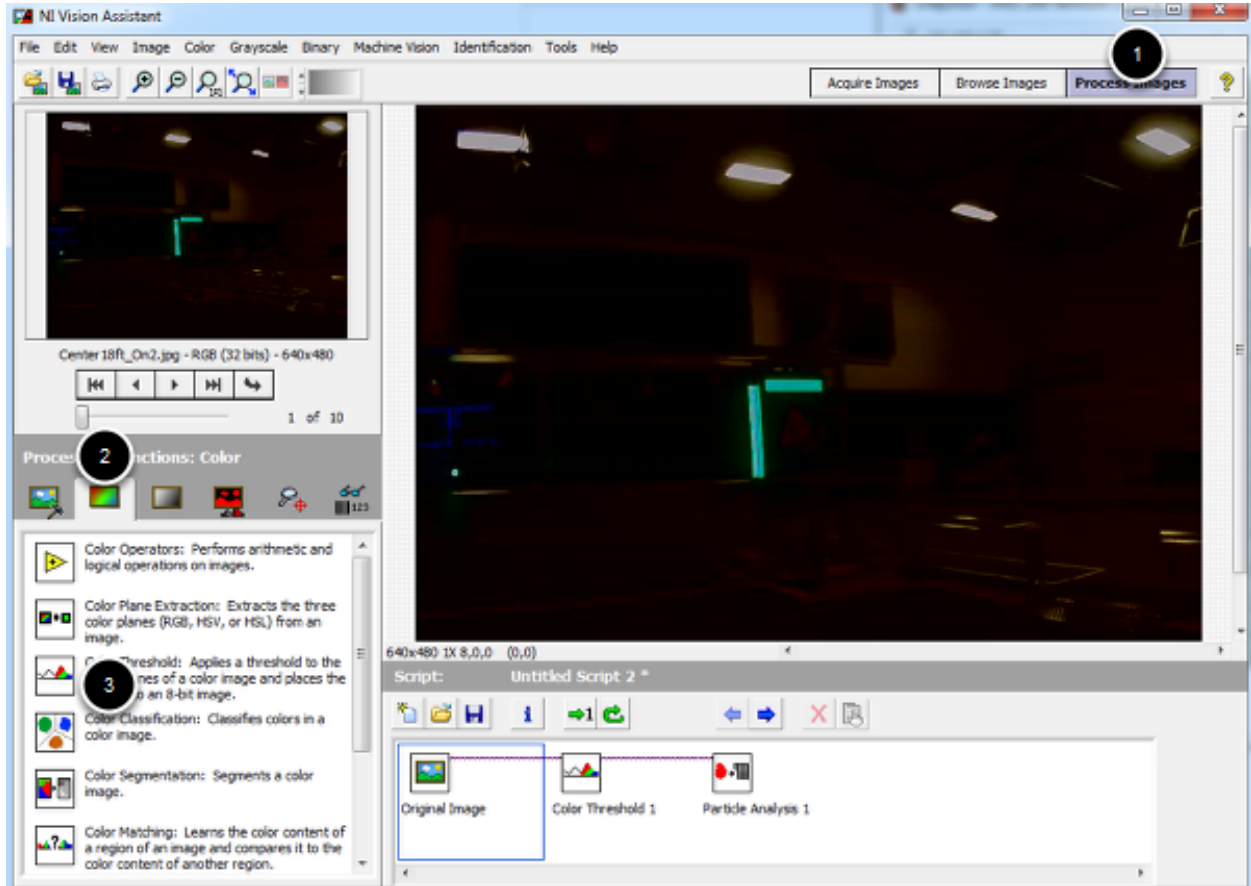
## 23.5 Color Threshold

Click Process Images in the top right, then select the color tab on the bottom right and click the Color Threshold icon.

### 23.5.1 HSV Calibration

Change the Color Model dropdown to HSV. Next tune the window on each of the three values to cover as much of the target as possible while filtering everything else. If using a green light, you may want to use the values in the sample code as a starting point. If you have multiple images you can use the controls in the top left to cycle through them. Use the center two arrow controls or the slider to change the preview image in the top left window, then click the right-most arrow to make it the active image. When you are happy with the values you have selected, note down the ranges for the Hue, Saturation and Value. You will need to enter these into the appropriate place in the vision code. Click OK to finish adding the step to the script.

You may wish to take some new sample images using the time for camera calibration at your event to verify or tweak your ranges slightly based on the venue lighting conditions.



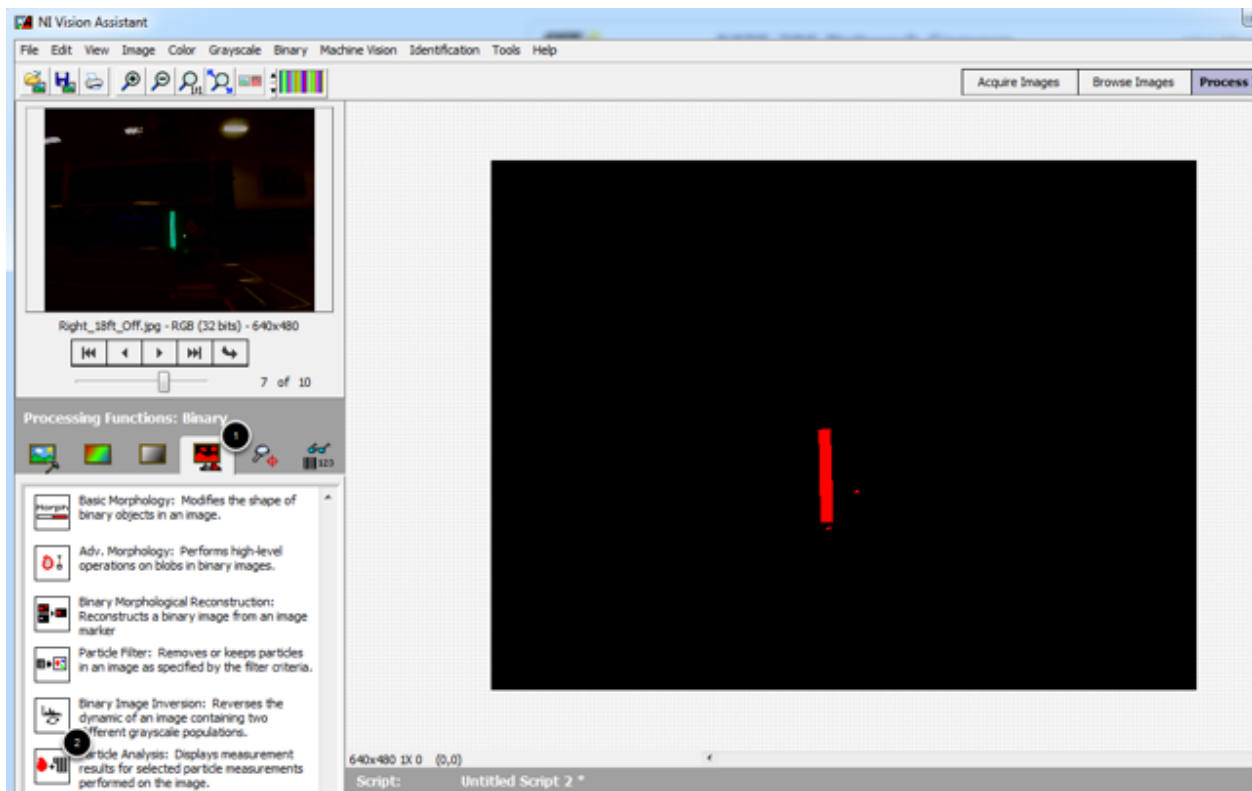
## 23.6 View Angle/Distance Calibration

While a theoretical view angle for each camera model can be found in the datasheet, empirical testing has found that these numbers may be a bit off even for the horizontal view angle. Given that this year's code uses the vertical field-of-view it is best to perform your own calibration for your camera (though empirical values for each camera type are included in the code as a reference). To do this set up an equation where the view angle,  $\Theta$ , is the only unknown. To do this, utilize a target of known size at a known distance, leaving the view angle as the only unknown. Let's take our equation from the previous article,  $d = Tft * FOV_{pixel} / (T_{pixel} * \tan \Theta)$ , and re-arrange it to solve for  $\Theta$ :

$$\tan \Theta = Tft * FOV_{pixel} / (T_{pixel} * d)$$

$$\Theta = \arctan(Tft * FOV_{pixel} / (T_{pixel} * d))$$

### 23.6.1 Taking Measurements

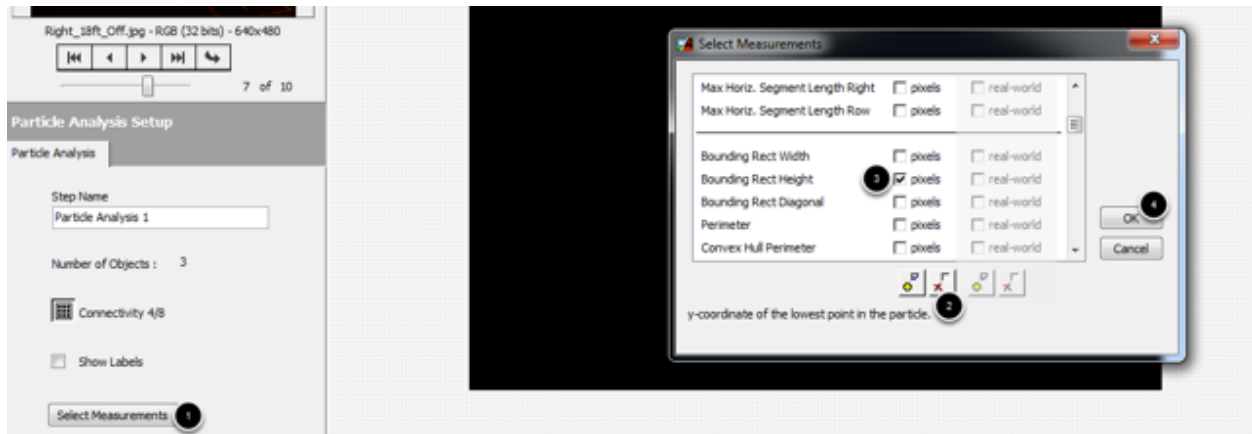


One way to take the required measurements is to use the same images of the retro-reflective tape that were used for the color calibration above. We can use Vision Assistant to provide the height of the detected blob in pixels. By measuring the real-world distance between the camera and the target, we now have all of the variables to solve our equation for the view angle.

To measure the particles in the image, click the Binary tab, then click the Particle Analysis icon.

### 23.6.2 Selecting Measurements

Click on the Select Measurements button. In this case, we are only interested in the bounding box height. Click on the button with the X to deselect all measurements, then locate the Bounding Rect Height measurement and check the box. Click OK to save.



### 23.6.3 Measuring the Particle

The measurements for each particle will now be displayed in the window at the bottom of the screen. If your image has multiple particles, you can click in each box to have Vision Assistant highlight the particle so you can make sure you have the right one. This article will show the calculation using a single image, but you may wish to perform the calculation on multiple images from multiple distances and use a technique such as averaging or least squares fit to determine the appropriate value for the View angle. You can use the same arrow controls described in the color section above to change the active image.

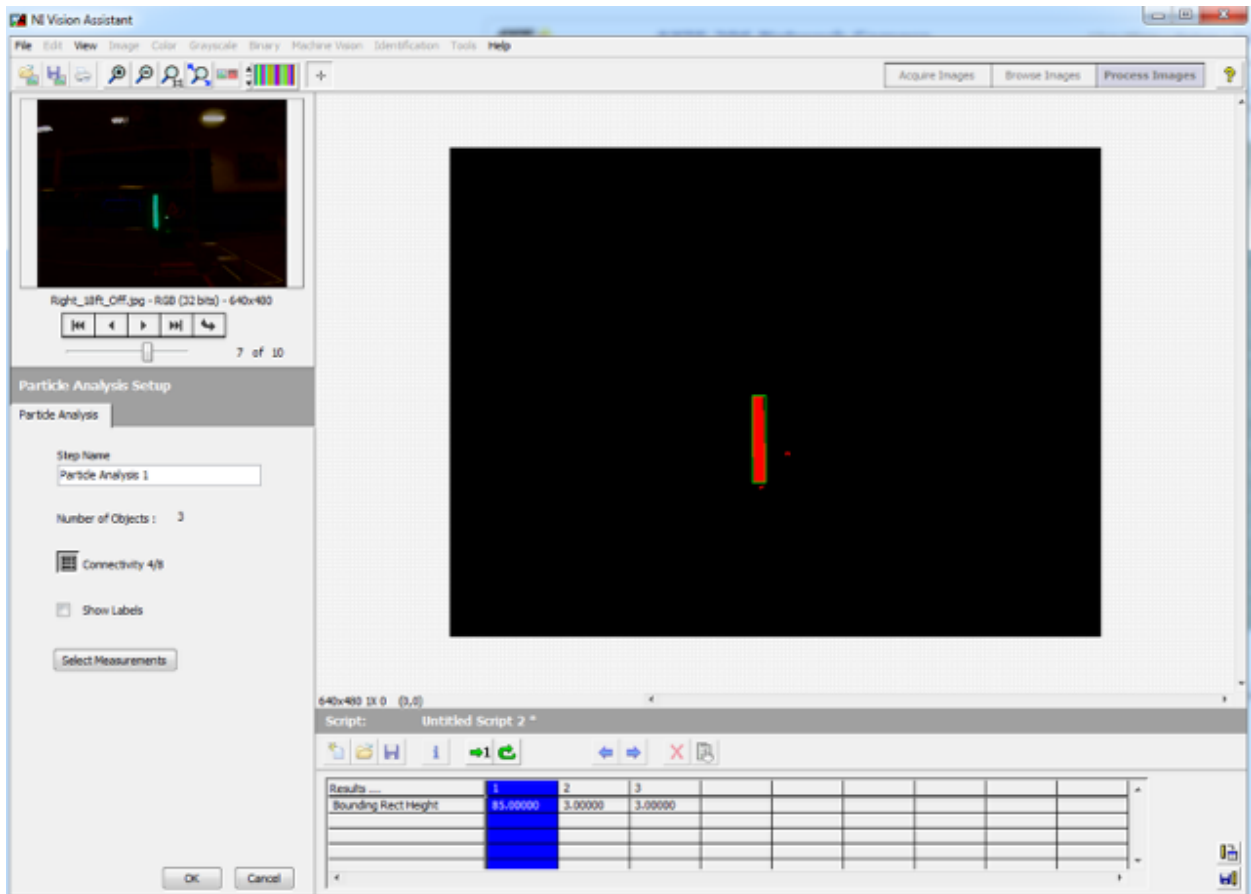
### 23.6.4 Calculation

As seen in the previous step, the particle representing the 32in tall vertical target in this example measured 85 pixels tall in a 640x480 image. The image shown was taken from (very roughly) 18 ft. away. Plugging these numbers into the equation from above. . . .

$$\Theta = \arctan(2.66 * 480 / (2 * 85 * 18)) = 22.65 \text{ degrees}$$

Depending on what you use to calculate the arctangent, your answer may be in radians, make sure to convert back to degrees if entering directly into the sample code as the view angle.

**Note:** The code uses View Angle and we just calculated  $\Theta$ . Make sure to multiply  $\Theta$  by 2 if replacing the constants in the code. Multiplying our result by 2 yields 45.3 degrees. This image is from a M1013 camera, so our value is a bit off from the previously measured 29.1 but given that the 18ft. was a very rough measurement this shows that we are in the ballpark and likely performed the calculation correctly.





---

## Axis M1013 Camera Compatibility

---

It has come to our attention that the Axis M1011 camera has been discontinued and superseded by the Axis M1013 camera. This document details any differences or issues we are aware of between the two cameras when used with WPILib and the provided sample vision programs.

### 24.1 Optical Differences

The Axis M1013 camera has a few major optical differences from the M1011 camera:

1. The M1013 is an adjustable focus camera. Make sure to focus your M1013 camera by turning the grey and black lens housing to make sure you have a clear image at your desired viewing distance.
2. The M1013 has a wider view angle (67 degrees) compared to the M1011 (47 degrees). This means that for a feature of a fixed size, the image of that feature will span a smaller number of pixels

### 24.2 Using the M1013 With WPILib

The M1013 camera has been tested with all of the available WPILib parameters and the following performance exceptions were noted:

1. The M1013 does not support the 160x120 resolution. Requesting a stream of this resolution will result in no images being returned or displayed.
2. The M1013 does not appear to work with the Color Enable parameter exposed by WPILib. Regardless of the setting of this parameter a full color image was returned.

All other WPILib camera parameters worked as expected. If any issues not noted here are discovered, please file a bug report on the WPILib tracker (note that you will need to create an account if you do not have one, but you do not need to be a member of the project).





---

# Using the Axis Camera at Single Network Events

---

The 2015 convention for using the Axis camera uses mDNS with the camera name set to `axis-camera.local`. At home this works fine as there is only one camera on the network. At official events, this works fine as each team is on their own VLAN and therefore doesn't have visibility to another team's camera. At an offseason using a single network, this will cause an issue where all teams will connect to whichever team's camera "wins" and becomes "axis-camera", the other cameras will see that the name is taken and use an alternative name. This article describes how to modify the Dashboard and/or robot code to use a different mDNS name to separate the camera streams.

## 25.1 Changing the camera mDNS name

To change the mDNS name in the camera, follow the instructions in [Configuring an Axis Camera](#) but substitute the new name such as `axis-cameraTEAM` where `TEAM` is your team number.

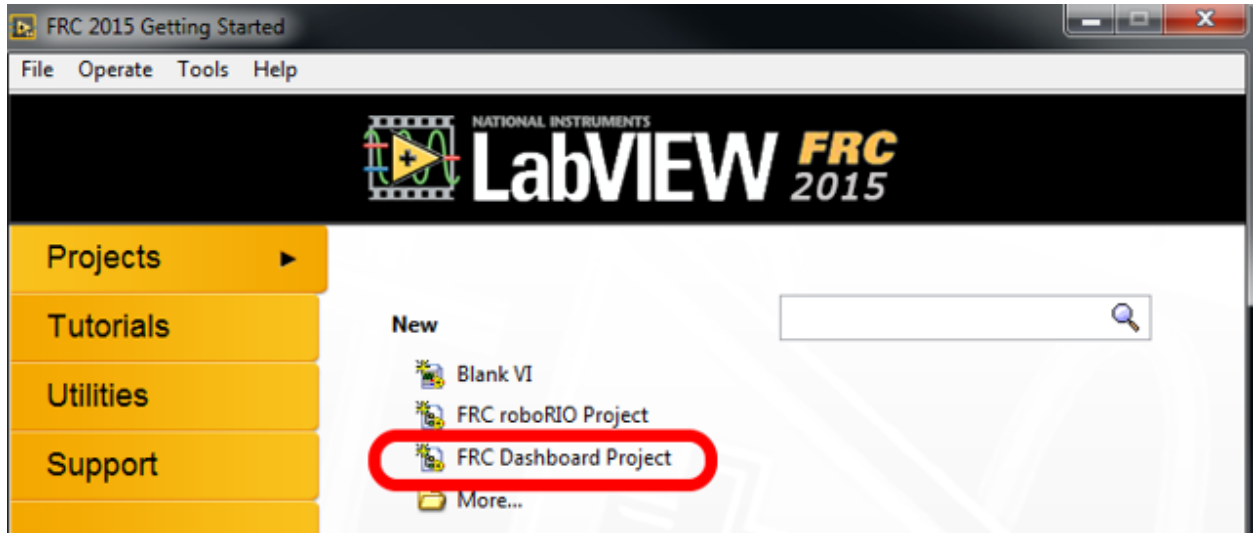
## 25.2 Viewing the camera on the DS PC - Browser or Java SmartDashboard

If you are using a web-browser or the updated Java SmartDashboard (which accepts mDNS names for the Simple Camera Viewer widget), updating to use the new mDNS name is simple. Simply change the URL in the browser or the address in the Simple Camera Viewer widget properties to the new mDNS name and you are all set.

## 25.3 Viewing the camera on the DS PC - LabVIEW Dashboard

To view the camera stream in the LabVIEW Dashboard, you will need to build a customized version of the Dashboard. Note that this customized version will only work for the Axis camera and will no longer work for a USB camera, revert to the default Dashboard to use a USB camera.

### 25.3.1 Creating a Dashboard Project



From the LabVIEW Splash screen, select “FRC Dashboard Project”. Name the project as desired, then click Finish.

### 25.3.2 Locating Loop 2 - Camera IP

Double click on Dashboard Main.vi in the project explorer to open it and press Ctrl+e to see the block diagram. Scroll down to the loop with the comment that says Loop 2 and locate the “Camera IP” input.

### 25.3.3 Editing the camera IP

Delete the Camera IP node, right click on the broken wire and click Create Constant (connect the constant to the wire if necessary). In the box, enter the mDNS name of your camera with a “.local” suffix (e.g. “axis-cameraTEAM.local” where TEAM is replaced with your team number). In this example I have used a sample name for team 9999. Then click File->Save or Ctrl+S to save the VI.

---

**Note:** You may also wish to make a minor modification to the Front Panel to verify that you are running the right dashboard later.

---

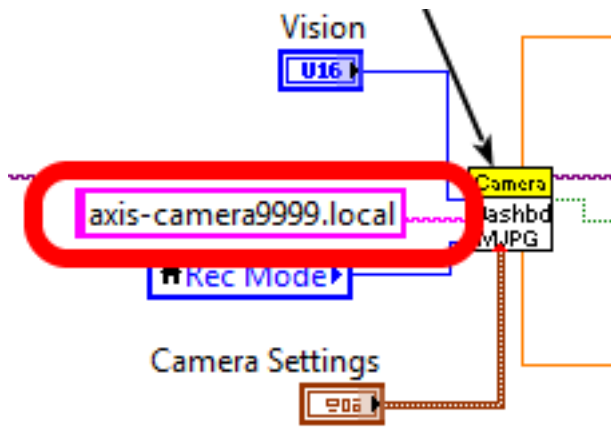
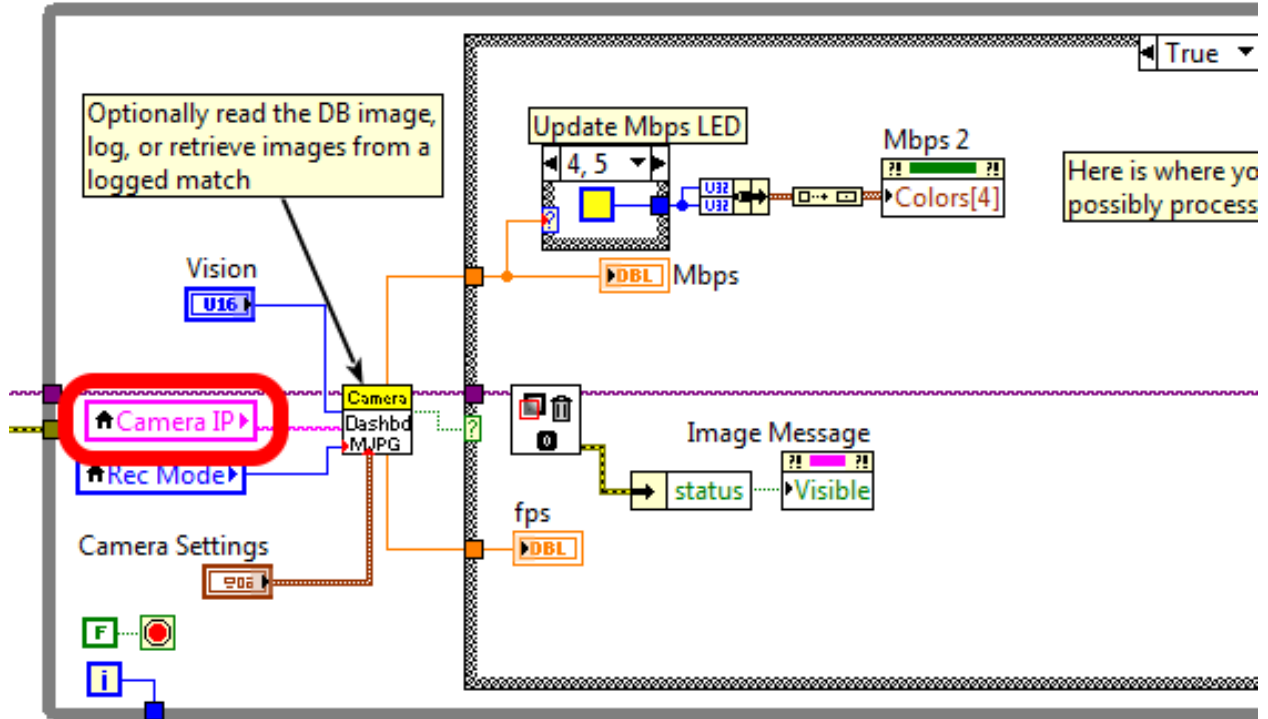
### 25.3.4 Building the Dashboard

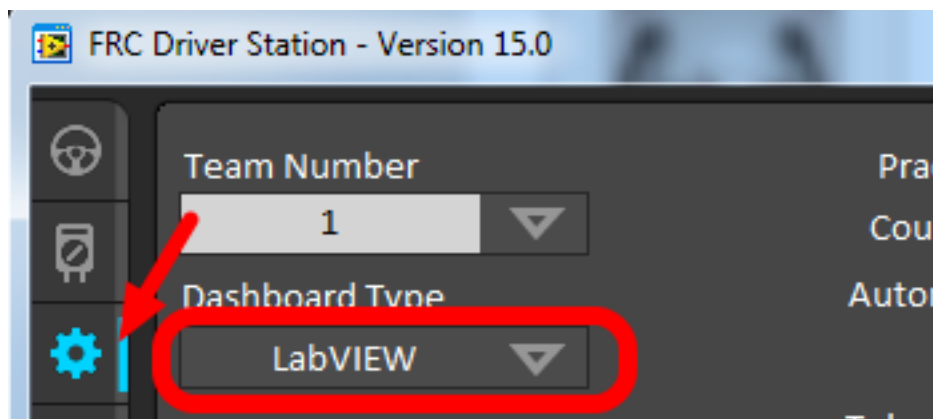
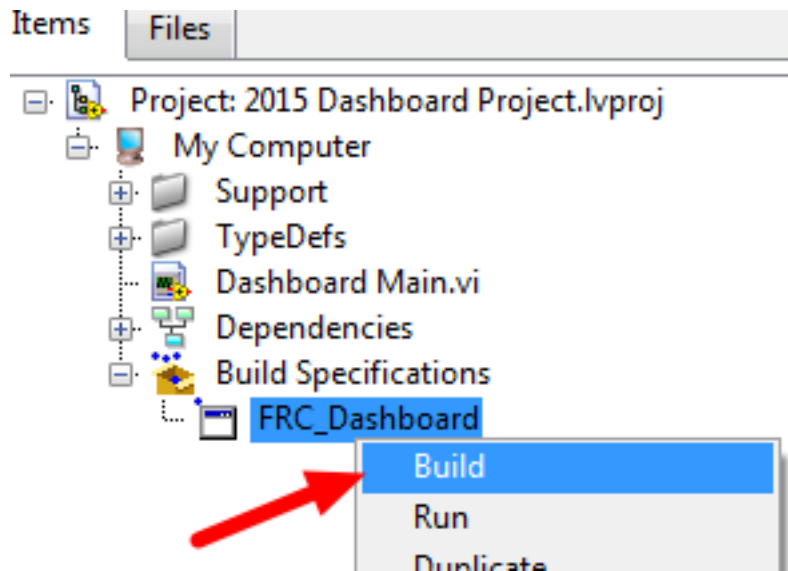
To build the new dashboard, expand Build Specifications in the Project Explorer, right click on FRC\_Dashboard and select Build.

### 25.3.5 Setting the Driver Station to launch the modified Dashboard

On the Setup tab of the Driver Station, change to dropdown box to LabVIEW to launch your new Dashboard.

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





## 25.4 Accessing the camera from Robot Code

If you wish to access the renamed camera from your robot code, you will have to modify it as well. In C++ and Java, just change the String used for the camera host name to match the new name. In LabVIEW follow the step below.

### 25.4.1 Modifying LabVIEW Robot Code



In the Project Explorer, locate Vision Processing.VI and double click to open it. Then press Ctrl+e to open the Block Diagram. Locate the string “axis-camera.local” near the left of the image and replace with “axis-cameraTEAM.local” Also make sure the constant is set to “False” to use the Axis camera instead of USB.



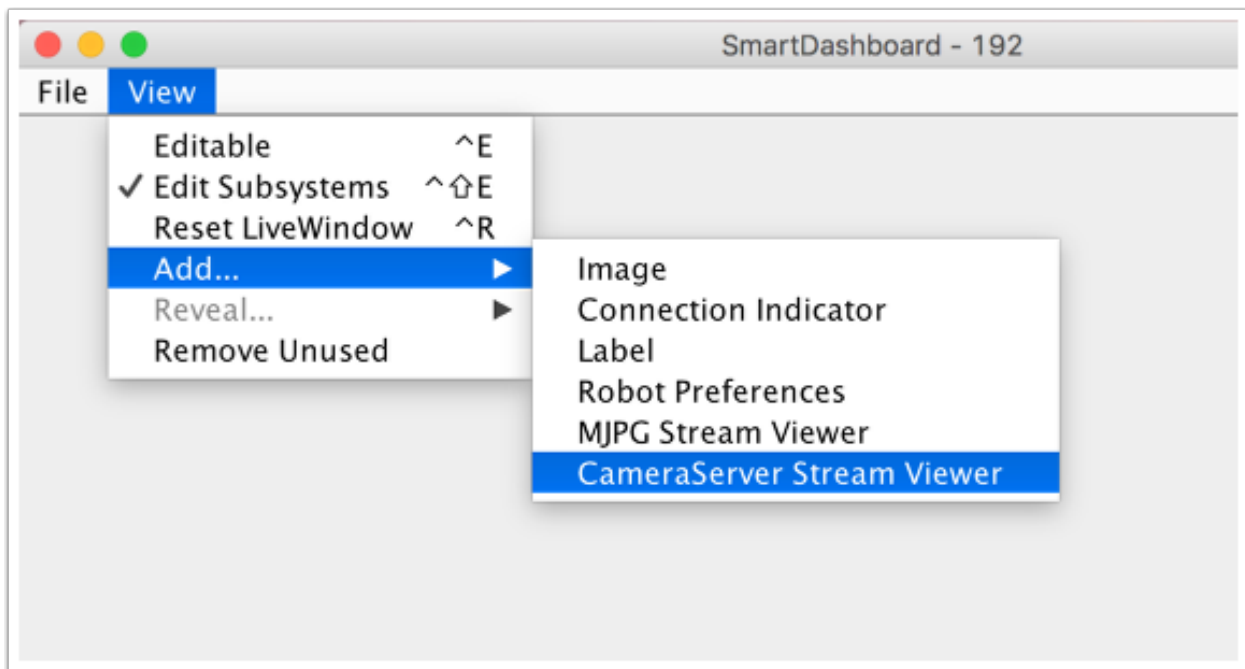
---

## Using the CameraServer on the roboRIO

---

### 26.1 Simple CameraServer program

The following program gets a CameraServer instance and starts automatic capture of a USB camera like the Microsoft LifeCam that is connected to the roboRIO. In this mode, the camera will capture frames and send them to the SmartDashboard. To view the images, create a CameraServer Stream Viewer widget using the “View”, then “Add” menu in the dashboard. The images are unprocessed and just forwarded from the camera to the dashboard.



Java

```

package org.usfirst.frc.team190.robot;

import edu.wpi.first.wpilibj.CameraServer;
import edu.wpi.first.wpilibj.IterativeRobot;

public class Robot extends IterativeRobot {

    public void robotInit() {
        CameraServer.getInstance().startAutomaticCapture();
    }
}

```

## C++

```

#include "WPIlib.h"
class Robot: public IterativeRobot
{
private:
    void RobotInit()
    {
        CameraServer::GetInstance()->StartAutomaticCapture();
    }
};
START_ROBOT_CLASS (Robot)

```

## 26.2 Advanced camera server program

In the following example a thread created in `robotInit()` gets the Camera Server instance. Each frame of the video is individually processed, in this case converting a color image (BGR) to gray scale using the OpenCV `cvtColor()` method. The resultant images are then passed to the output stream and sent to the dashboard. You can replace the `cvtColor` operation with any image processing code that is necessary for your application. You can even annotate the image using OpenCV methods to write targeting information onto the image being sent to the dashboard.

## Java

```

package org.usfirst.frc.team190.robot;

import org.opencv.core.Mat;
import org.opencv.imgproc.Imgproc;

import edu.wpi.cscore.CvSink;
import edu.wpi.cscore.CvSource;
import edu.wpi.cscore.UsbCamera;
import edu.wpi.first.wpilibj.CameraServer;
import edu.wpi.first.wpilibj.IterativeRobot;

public class Robot extends IterativeRobot {

    public void robotInit() {
        new Thread(() -> {
            UsbCamera camera = CameraServer.getInstance().startAutomaticCapture();
            camera.setResolution(640, 480);

            CvSink cvSink = CameraServer.getInstance().getVideo();
            CvSource outputStream = CameraServer.getInstance().putVideo("Blur",
↪ 640, 480);

```

(continues on next page)



(continued from previous page)

```

        Mat source = new Mat();
        Mat output = new Mat();

        while(!Thread.interrupted()) {
            cvSink.grabFrame(source);
            Imgproc.cvtColor(source, output, Imgproc.COLOR_BGR2GRAY);
            outputStream.putFrame(output);
        }
    }.start();
}
}

```

**C++**

```

#include "WPILib.h"
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/core/core.hpp>
class Robot: public IterativeRobot
{
private:
    static void VisionThread()
    {
        cs::UsbCamera camera = CameraServer::GetInstance()->StartAutomaticCapture();
        camera.SetResolution(640, 480);
        cs::CvSink cvSink = CameraServer::GetInstance()->GetVideo();
        cs::CvSource outputStreamStd = CameraServer::GetInstance()->PutVideo("Gray", ↵
↵640, 480);
        cv::Mat source;
        cv::Mat output;
        while(true) {
            cvSink.GrabFrame(source);
            cvtColor(source, output, cv::COLOR_BGR2GRAY);
            outputStreamStd.PutFrame(output);
        }
    }
    void RobotInit()
    {
        std::thread visionThread(VisionThread);
        visionThread.detach();
    }
};
START_ROBOT_CLASS(Robot)

```

Notice that in these examples, the PutVideo method writes the video to a named stream. To view that stream on the SmartDashboard set the properties on the CameraServerStreamViewer to refer to the named stream. In this case that is “Blur” for the Java program and “Gray” for the C++ sample.



---

## Using multiple cameras

---

### 27.1 Switching the driver views

If you're interested in just switching what the driver sees, and are using SmartDashboard, the SmartDashboard CameraServer Stream Viewer has an option ("Selected Camera Path") that reads the given NT key and changes the "Camera Choice" to that value (displaying that camera). The robot code then just needs to set the NT key to the correct camera name. Assuming "Selected Camera Path" is set to "CameraSelection", the following code uses the joystick 1 trigger button state to show camera1 and camera2.

C++

```
cs::UsbCamera camera1;
cs::UsbCamera camera2;
frc::Joystick joy1{0};
bool prevTrigger = false;
void RobotInit() override {
    camera1 = frc::CameraServer::GetInstance()->StartAutomaticCapture(0);
    camera2 = frc::CameraServer::GetInstance()->StartAutomaticCapture(1);
}
void TeleopPeriodic() override {
    if (joy1.GetTrigger() && !prevTrigger) {
        printf("Setting camera 2\n");
        nt::NetworkTableInstance::GetDefault().GetTable("").PutString("CameraSelection",
↪camera2.GetName());
    } else if (!joy1.GetTrigger() && prevTrigger) {
        printf("Setting camera 1\n");
        nt::NetworkTableInstance::GetDefault().GetTable("").PutString("CameraSelection",
↪camera1.GetName());
    }
    prevTrigger = joy1.GetTrigger();
}
```

If you're using some other dashboard, you can change the camera used by the camera server dynamically. If you open a stream viewer nominally to camera1, the robot code will change the stream contents to either camera1 or camera2 based on the joystick trigger.

C++

```

cs::UsbCamera camera1;
cs::UsbCamera camera2;
cs::VideoSink server;
frc::Joystick joy1{0};
bool prevTrigger = false;
void RobotInit() override {
    camera1 = frc::CameraServer::GetInstance()->StartAutomaticCapture(0);
    camera2 = frc::CameraServer::GetInstance()->StartAutomaticCapture(1);
    server = frc::CameraServer::GetInstance()->GetServer();
}
void TeleopPeriodic() override {
    if (joy1.GetTrigger() && !prevTrigger) {
        printf("Setting camera 2\n");
        server.SetSource(camera2);
    } else if (!joy1.GetTrigger() && prevTrigger) {
        printf("Setting camera 1\n");
        server.SetSource(camera1);
    }
    prevTrigger = joy1.GetTrigger();
}

```

## 27.2 Keeping Streams Open

By default, the cscore library is pretty aggressive in turning off cameras not in use. What this means is that when you switch cameras, it may disconnect from the camera not in use, so switching back will have some delay as it reconnects to the camera. To keep both camera connections open, use the `SetConnectionStrategy()` method to tell the library to keep the streams open, even if you aren't using them.

C++

```

cs::UsbCamera camera1;
cs::UsbCamera camera2;
cs::VideoSink server;
frc::Joystick joy1{0};
bool prevTrigger = false;
void RobotInit() override {
    camera1 = frc::CameraServer::GetInstance()->StartAutomaticCapture(0);
    camera2 = frc::CameraServer::GetInstance()->StartAutomaticCapture(1);
    server = frc::CameraServer::GetInstance()->GetServer();
    camera1.
↔SetConnectionStrategy(cs::VideoSource::ConnectionStrategy::kConnectionKeepOpen);
    camera2.
↔SetConnectionStrategy(cs::VideoSource::ConnectionStrategy::kConnectionKeepOpen);
}
void TeleopPeriodic() override {
    if (joy1.GetTrigger() && !prevTrigger) {
        printf("Setting camera 2\n");
        server.SetSource(camera2);
    } else if (!joy1.GetTrigger() && prevTrigger) {
        printf("Setting camera 1\n");
        server.SetSource(camera1);
    }
    prevTrigger = joy1.GetTrigger();
}

```

If both cameras are USB, it's worth noting that you may run into USB bandwidth limitations with higher resolutions, as in all of these cases the roboRio is going to be streaming data from both cameras to the roboRio simultaneously (for a short period in options 1 and 2, and continuously in option 3). It is theoretically possible for the library to avoid this simultaneity in the option 2 case (only), but this is not currently implemented.

Different cameras report bandwidth usage differently. The library will tell you if you're hitting the limit; you'll get this error message: "could not start streaming due to USB bandwidth limitations; try a lower resolution or a different pixel format (VIDIOC\_STREAMON: No space left on device)". If you're using Option 3 it will give you this error during RobotInit(). Thus you should just try your desired resolution and adjusting as necessary until you both don't get that error and don't exceed the radio bandwidth limitations.



---

## Introduction to GRIP

---

GRIP is a tool for developing computer vision algorithms interactively rather than through trial and error coding. After developing your algorithm you may run GRIP in headless mode on your roboRIO, on a Driver Station Laptop, or on a coprocessor connected to your robot network. With Grip you choose vision operations to create a graphical pipeline that represents the sequence of operations that are performed to complete the vision algorithm.

GRIP is based on OpenCV, one of the most popular computer vision software libraries used for research, robotics, and vision algorithm implementations. The operations that are available in GRIP are almost a 1 to 1 match with the operations available if you were hand coding the same algorithm with some text-based programming language.

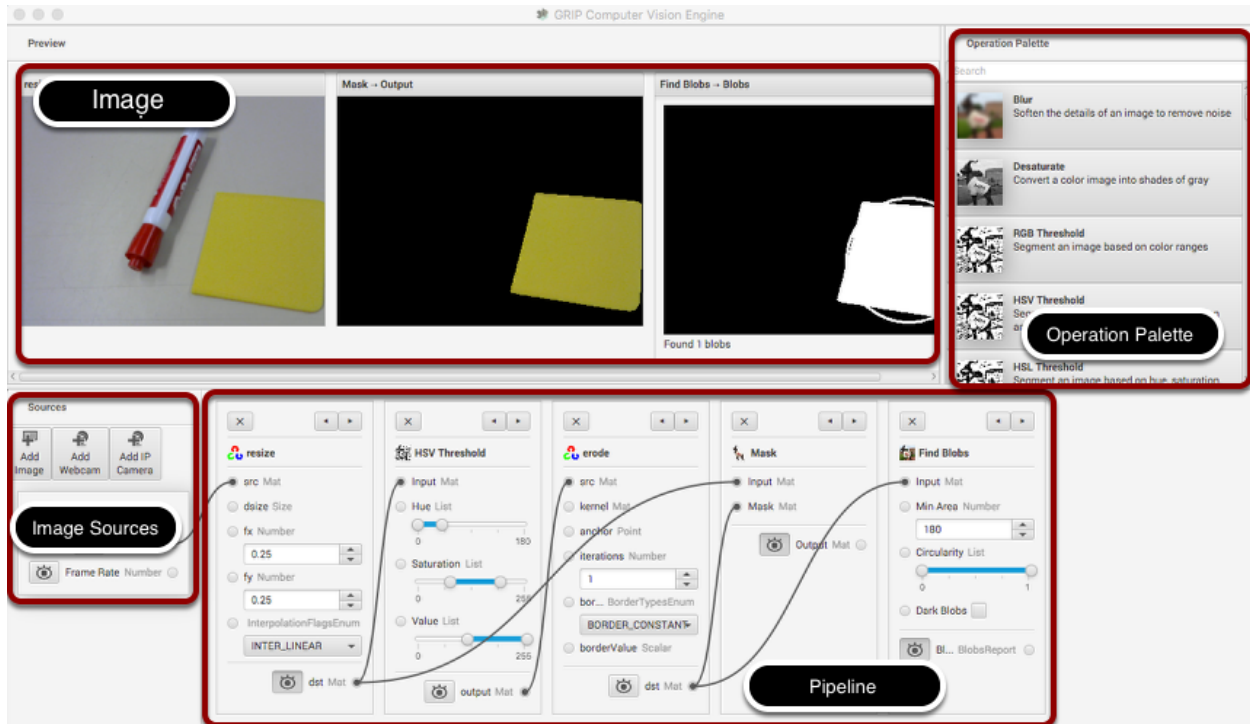
### 28.1 The GRIP user interface

The GRIP user interface consists of 4 parts:

- **Image Sources** are the ways of getting images into the GRIP pipeline. You can provide images through attached cameras or files. Sources are almost always the beginning of the image processing algorithm.
- **Operation Palette** contains the image processing steps from the OpenCV library that you can chain together in the pipeline to form your algorithm. Clicking on an operation in the palette adds it to the end of the pipeline. You can then use the left and right arrows to move the operation within the pipeline.
- **Pipeline** is the sequence of steps that make up the algorithm. Each step (operation) in the pipeline is connected to a previous step from the output of one step to an input to the next step. The data flows from generally from left to right through the connections that you create.
- **Image Preview** are shows previews of the result of each step that has it's preview button pressed. This makes it easy to debug algorithms by being able to preview the outputs of each intermediate step.

### 28.2 Finding the yellow square

In this application we will try to find the yellow square in the image and display it's position. The setup is pretty simple, just a USB web camera connected to the computer looking down at some colorful objects. The yellow plastic



square is the thing that we're interested in locating in the image.

## 28.3 Enable the image source

The first step is to acquire an image. To use the source, click on the “Add Webcam” button and select the camera number. In this case the Logitech USB camera that appeared as Webcam 0 and the computer monitor camera was Webcam 1. The web camera is selected in this case to grab the image behind the computer as shown in the setup. Then select the image preview button and the real-time display of the camera stream will be shown in the preview area.

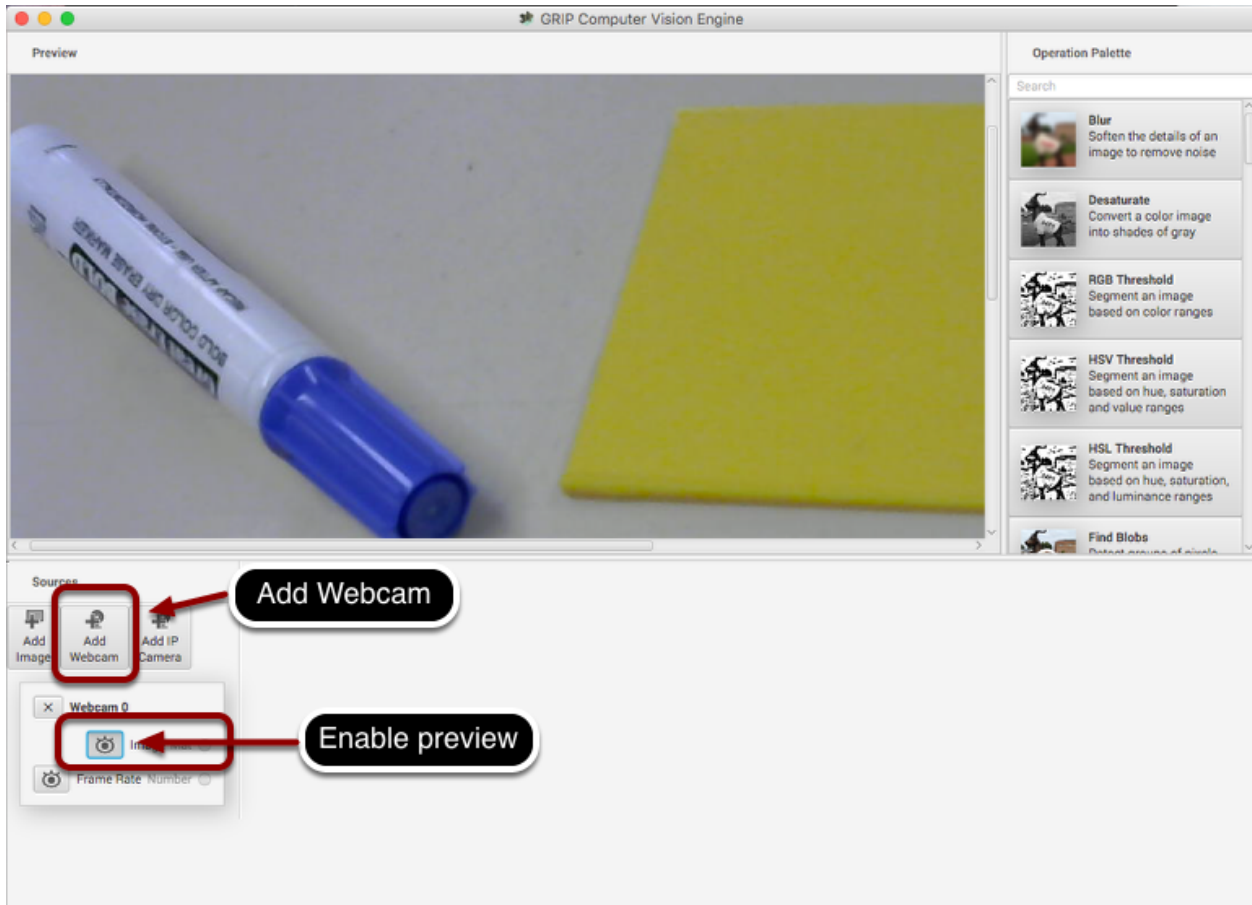
## 28.4 Resize the image

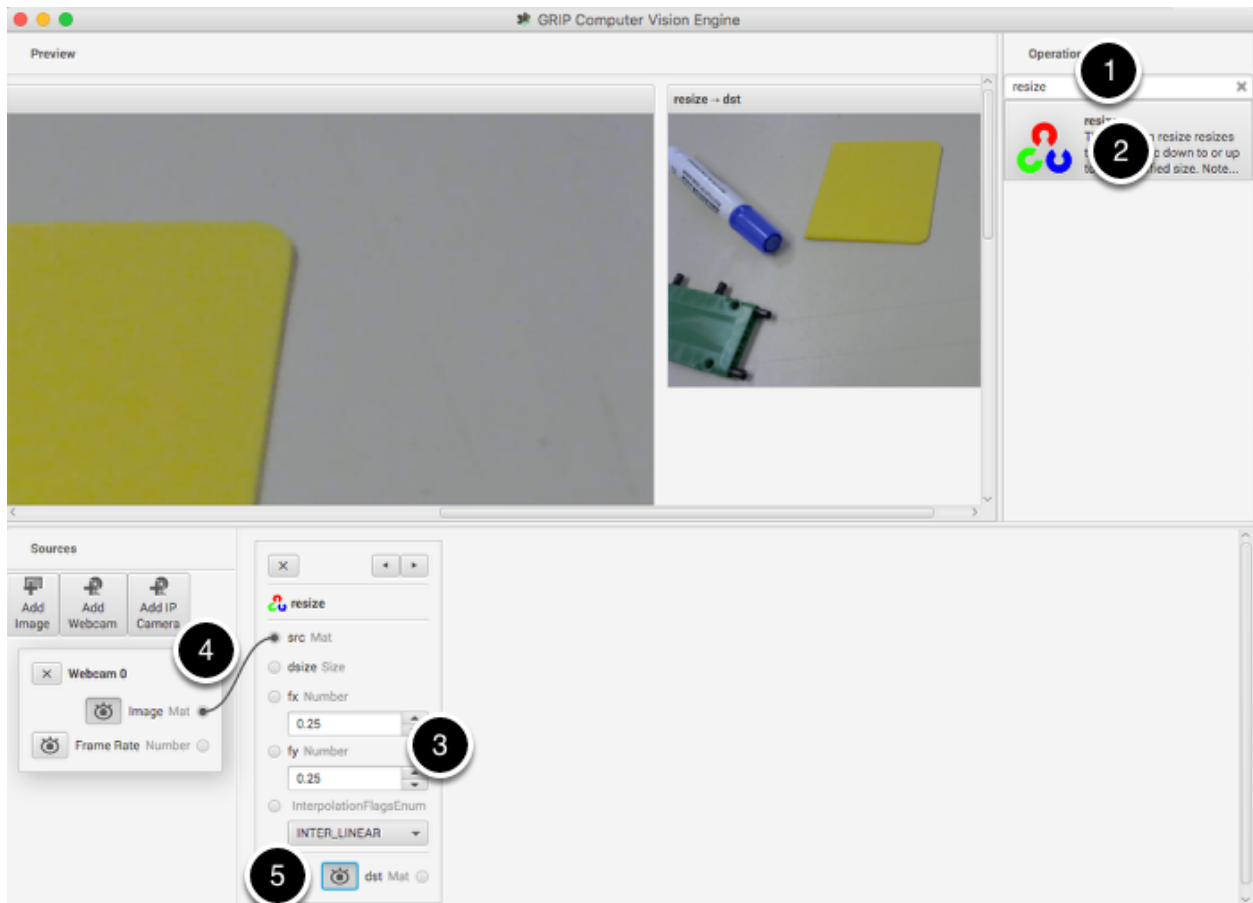
In this case the camera resolution is too high for our purposes, and in fact the entire image cannot even be viewed in the preview window. The “Resize” operation is clicked from the Operation Palette to add it to the end of the pipeline. To help locate the Resize operation, type “Resize” into the search box at the top of the palette. The steps are:

1. Type “Resize” into the search box on the palette
2. Click the Resize operation from the palette. It will appear in the pipeline.
3. Enter the x and y resize scale factor into the resize operation in the pipeline. In this case 0.25 was chosen for both.
4. Drag from the Webcam image output mat socket to the Resize image source mat socket. A connection will be shown indicating that the camera output is being sent to the resize input.
5. Click on the destination preview button on the “Resize” operation in the pipeline. The smaller image will be displayed alongside the larger original image. You might need to scroll horizontally to see both as shown.



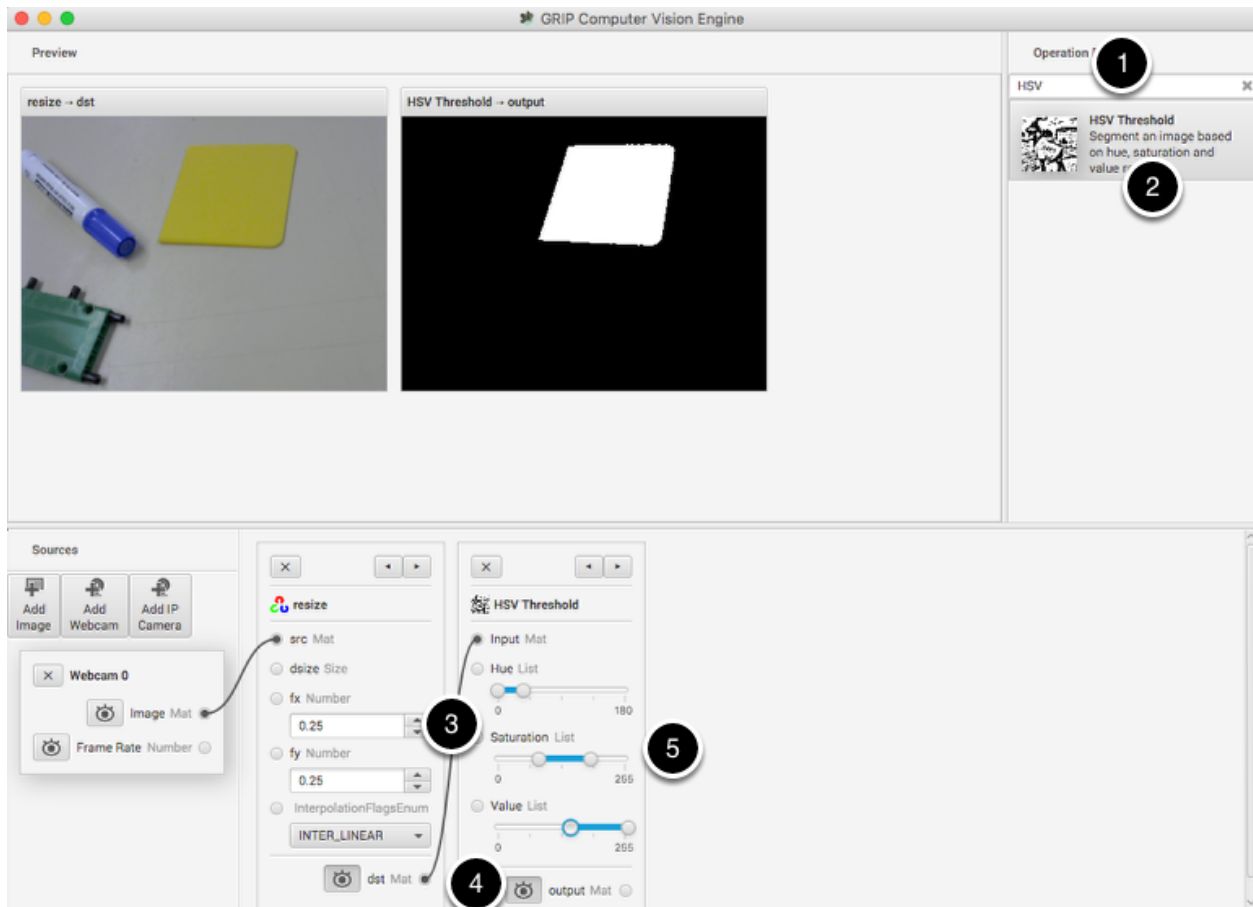






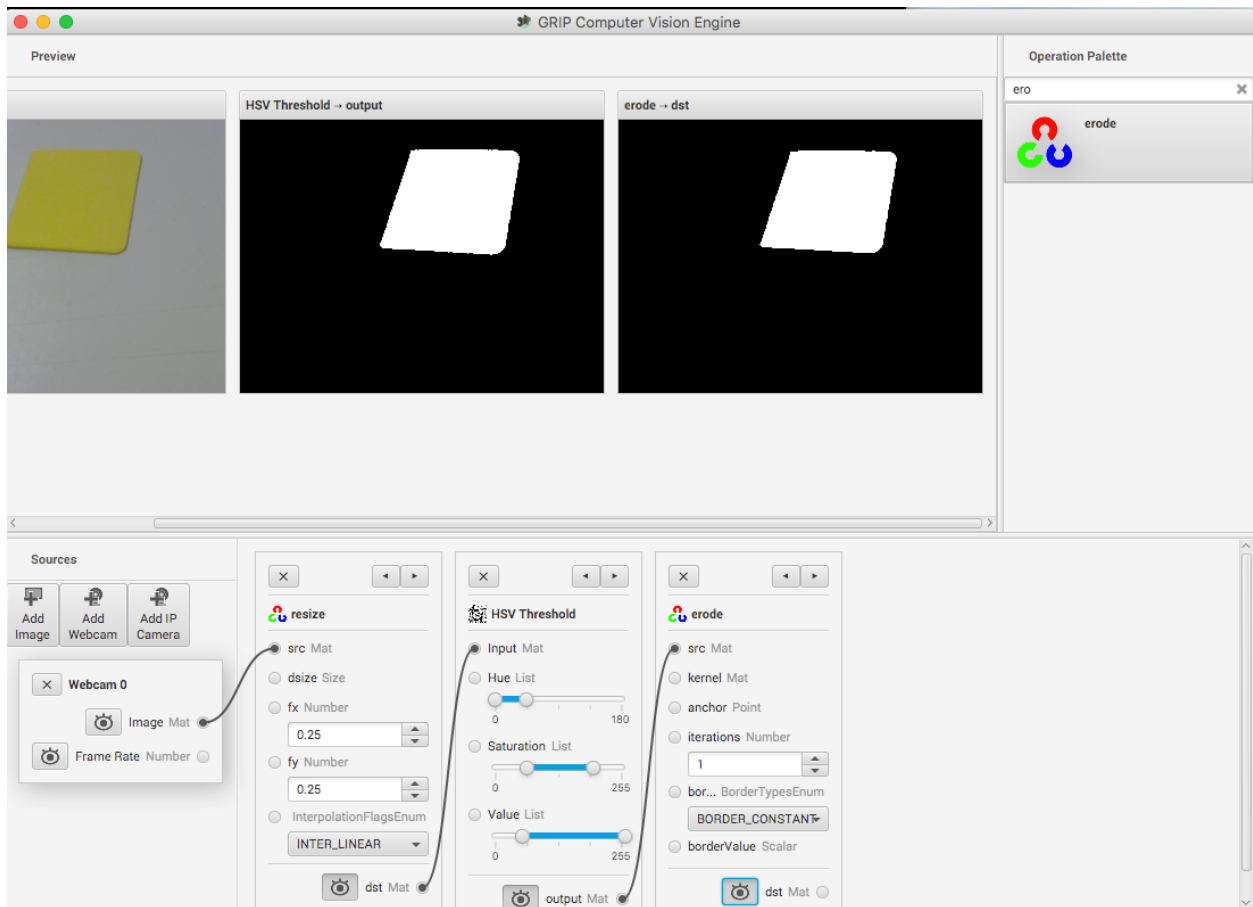
6. Lastly, click the Webcam source preview button since there is no reason to look at both the large image and the smaller image at the same time.

## 28.5 Find only the yellow parts of the image



The next step is to remove everything from the image that doesn't match the yellow color of the piece of plastic that is the object being detected. To do that a HSV Threshold operation is chosen to set upper and lower limits of HSV values to indicate which pixels should be included in the resultant binary image. Notice that the target area is white while everything that wasn't within the threshold values are shown in black. Again, as before:

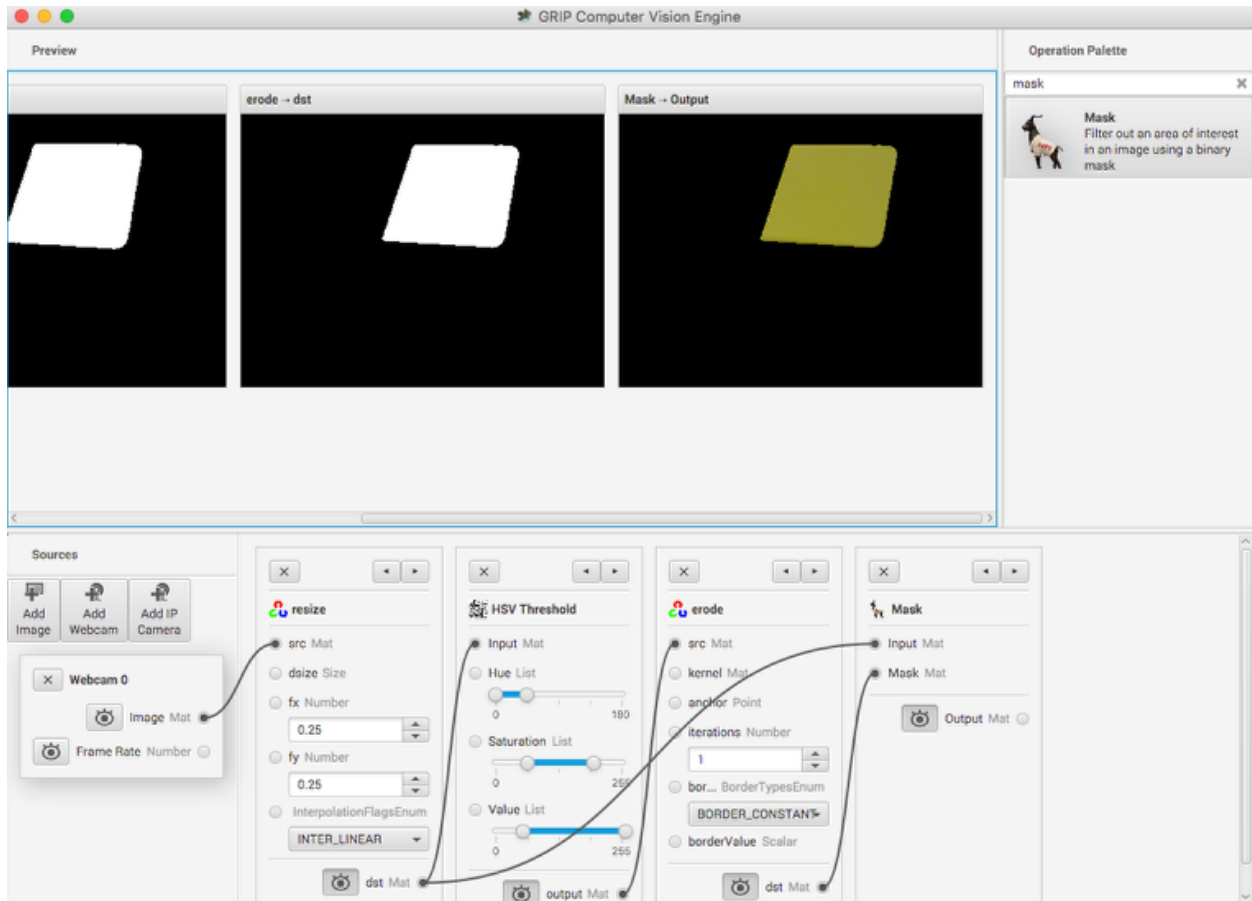
1. Type HSV into the search box to find the HSV Threshold operation.
2. Click on the operation in the palette and it will appear at the end of the pipeline.
3. Connect the dst (output) socket on the resize operation to the input of the HSV Threshold.
4. Enable the preview of the HSV Threshold operation so the result of the operation is displayed in the preview window.
5. Adjust the Hue, Saturation, and Value parameters only the target object is shown in the preview window.



## 28.6 Get rid of the noise and extraneous hits

This looks pretty good so far, but sometimes there is noise from other things that couldn't quite be filtered out. To illustrate one possible technique to reduce those occasional pixels that were detected, an Erosion operation is chosen. Erosion will remove small groups of pixels that are not part of the area of interest.

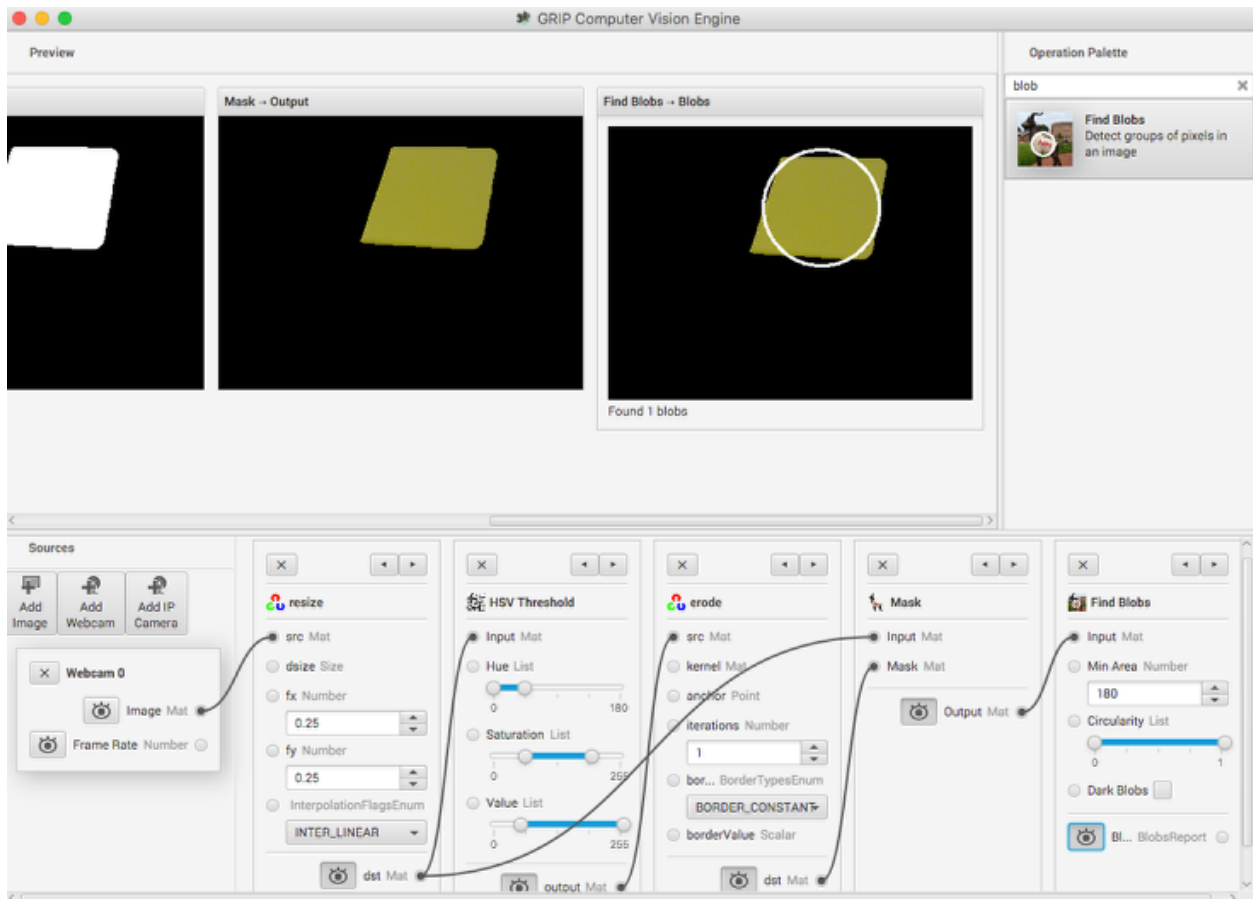
## 28.7 Mask just the yellow area from the original image



Here a new image is generated by taking the original image and masking (and operation) it with the results of the erosion. This leaves just the yellow card as seen in the original image with nothing else shown. And it makes it easy to visualize exactly what was being found through the series of filters.

## 28.8 Find the yellow area (blob)

The last step is actually detecting the yellow card using a Blob Detector. This operation looks for a grouping of pixels that have some minimum area. In this case, the only non-black pixels are from the yellow card after the filtering is done. You can see that a circle is drawn around the detected portion of the image. In the release version of GRIP (watch for more updates between now and kickoff) you will be able to send parameters about the detected blob to your robot program using Network Tables.



## 28.9 Status of GRIP

As you can see from this example, it is very easy and fast to be able to do simple object recognition using GRIP. While this is a very simple example, it illustrates the basic principles of using GRIP and feature extraction in general. Over the coming weeks the project team will be posting updates to GRIP as more features are added. Currently it supports cameras (Axis ethernet camera and web cameras) and image inputs. There is no provision for output yet although Network Tables and ROS (Robot Operating System) are planned.

You can either download a pre-built release of the code from the github page “Releases” section (<https://github.com/WPIRoboticsProjects/GRIP>) or you can clone the source repository and build it yourself. Directions on building GRIP are on the project page. There is also additional documentation on the project wiki.

So, please play with GRIP and give us feedback here on the forum. If you find bugs, you can either post them here or as a Github project issue on the project page.



---

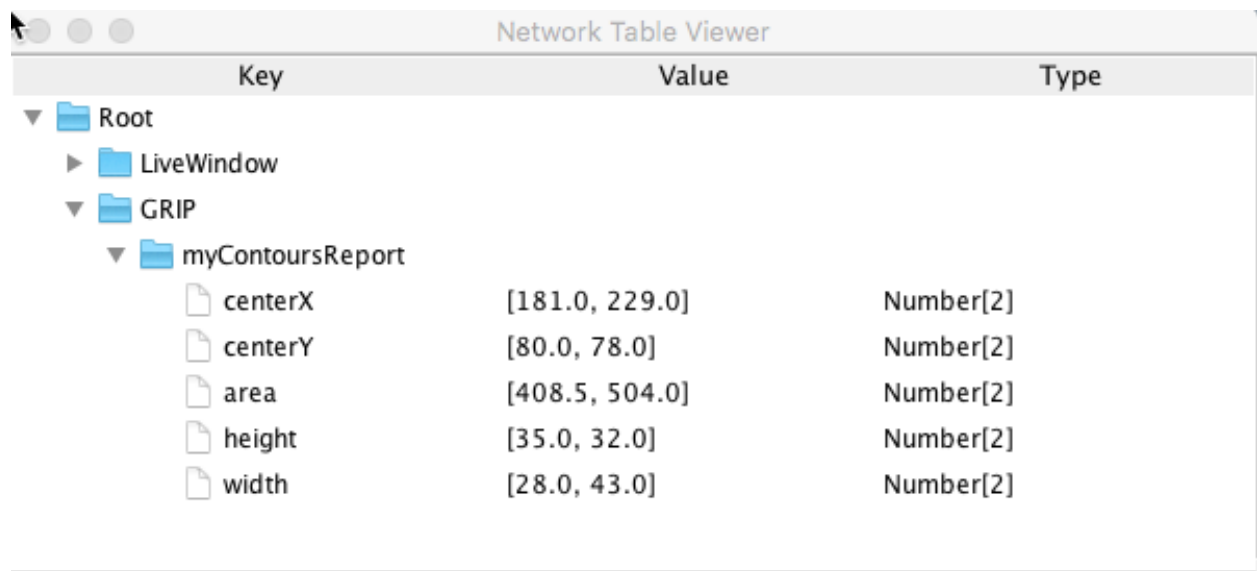
## Reading array values published by NetworkTables

---

This article describes how to read values published by NetworkTables using a program running on the robot. This is useful when using computer vision where the images are processed on your driver station laptop and the results stored into NetworkTables possibly using a separate vision processor like a raspberry pi, or a tool on the robot like GRIP, or a python program to do the image processing.

Very often the values are for one or more areas of interest such as goals or game pieces and multiple instances are returned. In the example below, several x, y, width, height, and areas are returned by the image processor and the robot program can sort out which of the returned values are interesting through further processing.

### 29.1 Verify the network table keys being published



The screenshot shows the Network Table Viewer application. The title bar reads "Network Table Viewer". The interface is divided into two main sections. On the left is a tree view showing the hierarchy of keys: Root, LiveWindow, GRIP, and myContoursReport. On the right is a table with three columns: Key, Value, and Type. The table lists five keys under myContoursReport: centerX, centerY, area, height, and width. Each key has a corresponding value in brackets and a type of Number[2].

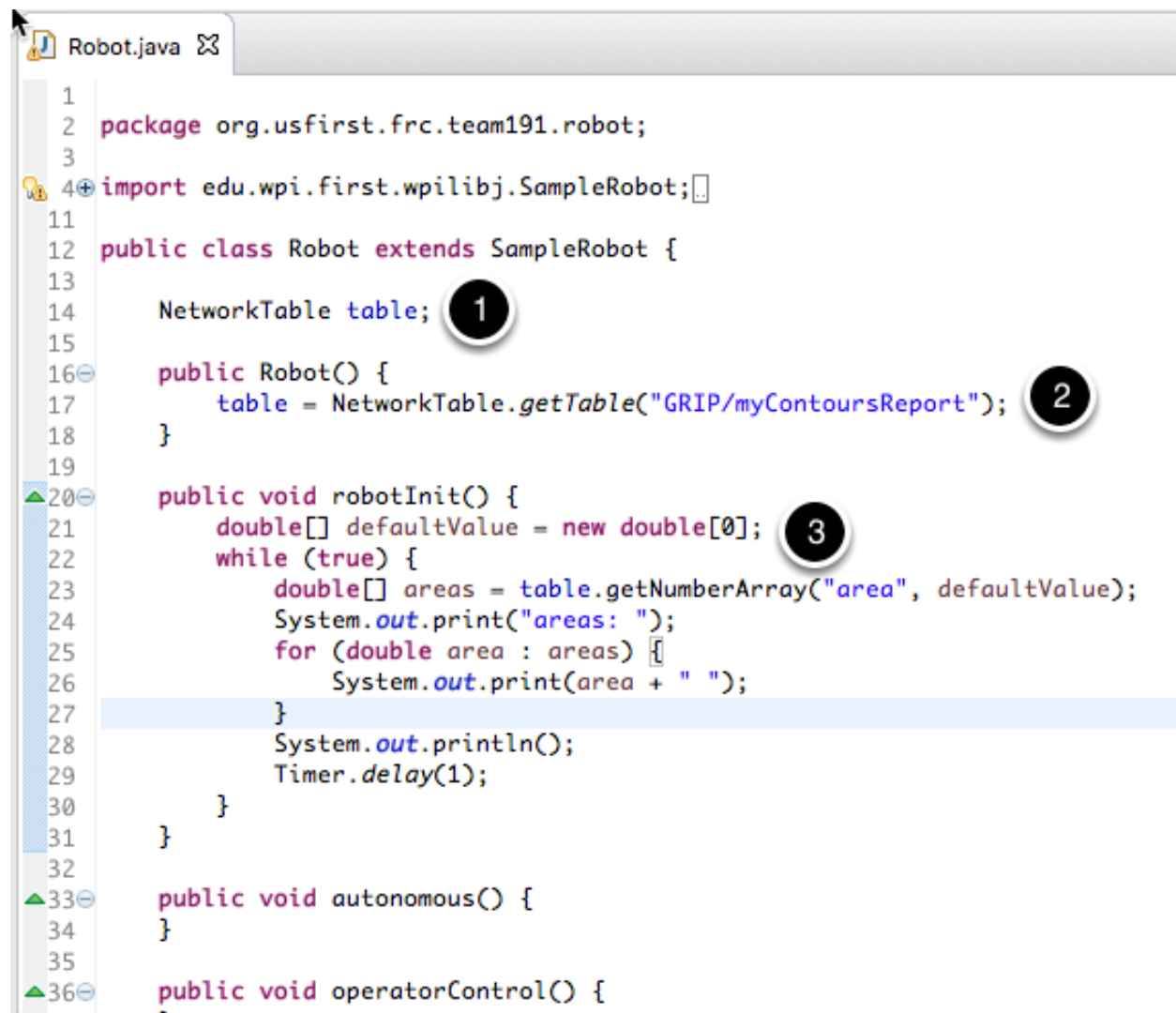
Key	Value	Type
Root		
LiveWindow		
GRIP		
myContoursReport		
centerX	[181.0, 229.0]	Number[2]
centerY	[80.0, 78.0]	Number[2]
area	[408.5, 504.0]	Number[2]
height	[35.0, 32.0]	Number[2]
width	[28.0, 43.0]	Number[2]

You can verify the names of the network table keys used for publishing the values by using the Network Table Viewer application. It is a java program in your user directory in the wpilib/tools folder. The application is started by selecting the “WPILib” menu in eclipse then “OutlineViewer”. In this example, with the image processing program running (GRIP) you can see the values being put into NetworkTables.

In this case the values are stored in a table called GRIP and a sub-table called myContoursReport. You can see that the values are in brackets and there are 2 values in this case for each key. The network table key names are centerX, centerY, area, height and width.

Both of the following examples are extremely simplified programs that just illustrate the use of NetworkTables. All the code is in the robotInit() method so it's only run when the program starts up. In your programs, you would more likely get the values in code that is evaluating which direction to aim the robot in a command or a control loop during the autonomous or teleop periods.

## 29.2 Writing a Java program to access the keys



```
1
2 package org.usfirst.frc.team191.robot;
3
4 import edu.wpi.first.wpilibj.SampleRobot;
5
11
12 public class Robot extends SampleRobot {
13
14     NetworkTable table; 1
15
16     public Robot() {
17         table = NetworkTable.getTable("GRIP/myContoursReport"); 2
18     }
19
20     public void robotInit() {
21         double[] defaultValue = new double[0]; 3
22         while (true) {
23             double[] areas = table.getNumberArray("area", defaultValue);
24             System.out.print("areas: ");
25             for (double area : areas) {
26                 System.out.print(area + " ");
27             }
28             System.out.println();
29             Timer.delay(1);
30         }
31     }
32
33     public void autonomous() {
34     }
35
36     public void operatorControl() {
37     }
38 }
```

The steps to getting the values and, in this program, printing them are:

1. Declare the table variable that will hold the instance of the subtable that have the values.

2. Initialize the subtable instance so that it can be used later for retrieving the values.
3. Read the array of values from NetworkTables. In the case of a communicating programs, it's possible that the program producing the output being read here might not yet be available when the robot program starts up. To avoid issues of the data not being ready, a default array of values is supplied. This default value will be returned if the network table key hasn't yet been published. This code just loops forever and reads values and prints them to the console.

## 29.3 Writing a C++ program to access the keys

```

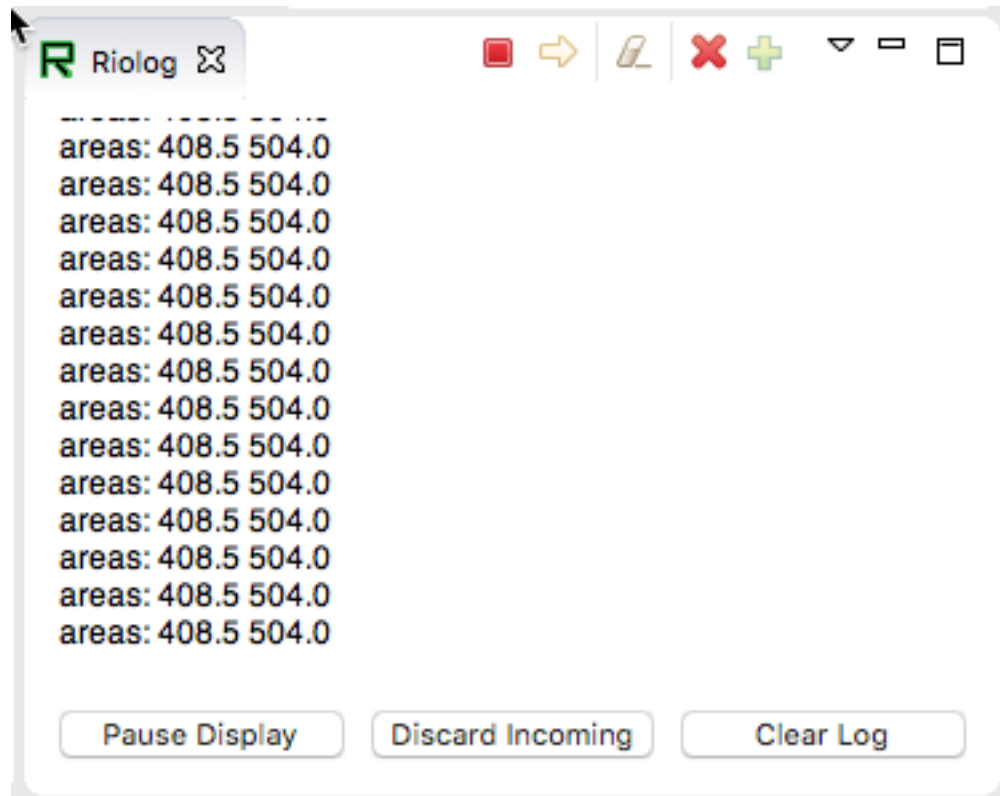
Robot.cpp
1 #include "WPIlib.h"
2 #include <iostream>
3
4 class Robot: public SampleRobot
5 {
6
7 public:
8     std::shared_ptr<NetworkTable> table; 1
9
10 Robot() {
11     table = NetworkTable::GetTable("GRIP/myContoursReport"); 2
12 }
13
14 void RobotInit()
15 {
16     while (true) {
17         std::cout << "Areas: "; 3
18         // get the array with an empty array as the default value
19         std::vector<double> arr = table->GetNumberArray("area", llvm::ArrayRef<double>());
20         for (unsigned int i = 0; i < arr.size(); i++) {
21             std::cout << arr[i] << " ";
22         }
23         std::cout << std::endl;
24         Wait(1);
25     }
26 }
27
28 void Autonomous()
29 {
30 }
31
32 void OperatorControl()
33 {
34 }
35
36 void Test()
37 {

```

The steps to getting the values and, in this program, printing them are:

1. Declare the table variable that will hold the instance of the subtable that have the values. It is a shared pointer where the library takes care of allocation and deallocation automatically.
2. Initialize the subtable instance so that it can be used later for retrieving the values.
3. Read the array of values from NetworkTables. In the case of a communicating programs, it's possible that the program producing the output being read here might not yet be available when the robot program starts up. To avoid issues of the data not being ready, a default array of values is supplied. `llvm::ArrayRef<double>` creates this temporary array reference of zero length that would be returned if the network table key hasn't yet been published. This code just loops forever and reads values and prints them to the console.

## 29.4 Program output



In this case the program is only looking at the array of areas, but in a real example all the values would more likely be used. Using the Riolog in eclipse or the DriverStation log you can see the values as they are retrieved. This program is using a sample static image so they areas don't change, but you can imagine with a camera on your robot, the values would be changing constantly.

---

## Generating Code from GRIP

---

### 30.1 GRIP Code Generation

When running your vision algorithm on a small processor such as a roboRIO or Raspberry PI it is encouraged to run OpenCV directly on the processor without the overhead of GRIP. To facilitate this, GRIP can generate code in C++, Java, and Python for the pipeline that you have created. This generated code can be added to your robot project and called directly from your existing robot code.

Input sources such as cameras or image directories and output steps such as NetworkTables are not generated. Your code must supply images as OpenCV mats. On the roboRIO, the CameraServer class supplies images in that format. For getting results you can just use generated getter methods for retrieving the resultant values such as contour x and y values.

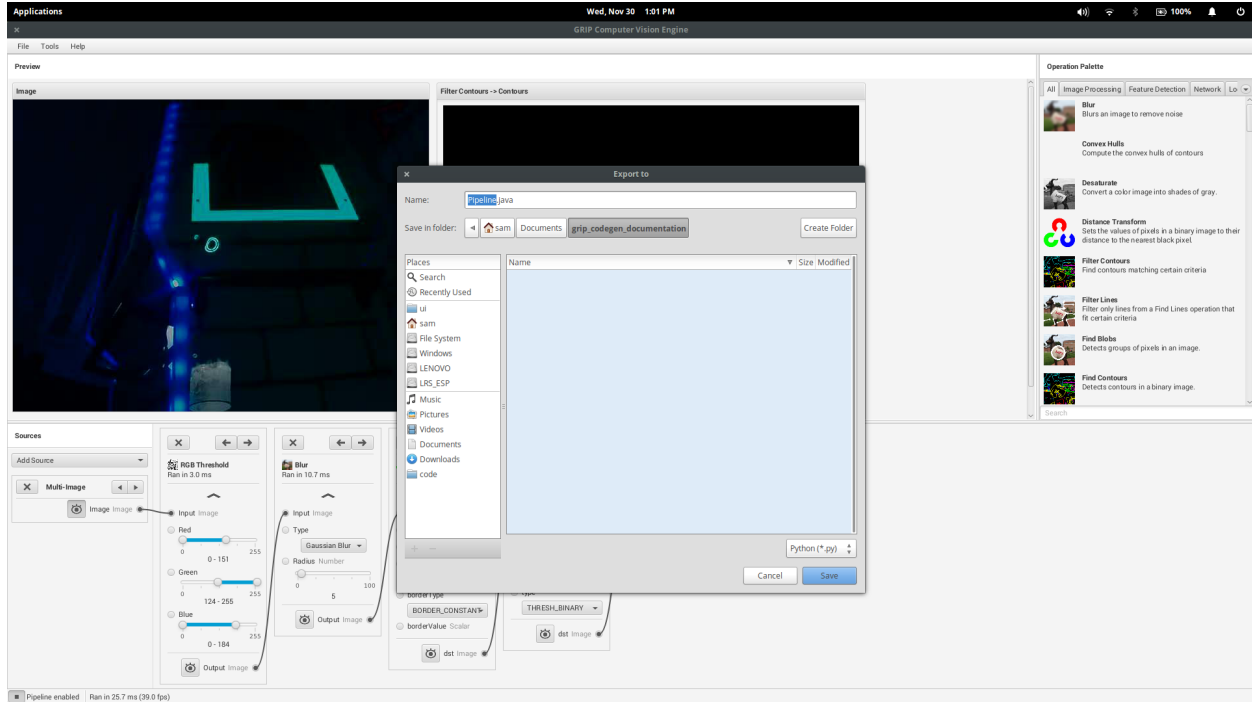
### 30.2 Generating Code

To generate code, go to `Tools > Generate Code`. This will bring up a save dialog that lets you create a C++, Java, or Python class that performs the steps in the GRIP pipeline.

If generating code to be used in a pre-existing project, choose a relevant directory to save the pipeline to.

- **C++ Users:** the pipeline class is split into a header and implementation file
- **Java Users:** the generated class lacks a package declaration, so a declaration should be added to match the directory where the file was saved.
- **Python Users:** the module name will be identical to the class, so the import statement will be something like 

```
from Pipeline import Pipeline
```



### 30.3 Structure of the Generated Code

```

Pipeline:
// Process -- this will run the pipeline
process(Mat source)

// Output accessors
getFooOutput ()
getBar0Output ()
getBar1Output ()
...
    
```

#### 30.3.1 Running the Pipeline

To run the Pipeline, call the process method with the sources (webcams, IP camera, image file, etc) as arguments. This will expose the outputs of every operation in the pipeline with the `getFooOutput` methods.

#### 30.3.2 Getting the Results

Users are able to the outputs of every step in the pipeline. The outputs of these operations would be accessible through their respective accessors. For example:

Operation	Java/C++ getter	Python variable
RGB Threshold	<code>getRgbThresholdOutput</code>	<code>rgb_threshold_output</code>
Blur	<code>getBlurOutput</code>	<code>blur_output</code>
CV Erode	<code>getCvErodeOutput</code>	<code>mcv_erode_output</code>
Find Contours	<code>getFindContoursOutput</code>	<code>find_contours_output</code>
Filter Contours	<code>getFilterContoursOutput</code>	<code>filter_contours_output</code>

If an operation appears multiple times in the pipeline, the accessors for those operations have the number of that operation:

Operation	Which appearance	Accessor
Blur	First	getBlur0Output
Blur	Second	getBlur1Output
Blur	Third	getBlur2Output





---

## Using Generated Code in a Robot Program

---

GRIP generates a class that can be added to an FRC program that runs on a roboRIO and without a lot of additional code, drive the robot based on the output.

Included here is a complete sample program that uses a GRIP pipeline that drives a robot towards a piece of retroreflective material.

This program is designed to illustrate how the vision code works and does not necessarily represent the best technique for writing your robot program. When writing your own program be aware of the following considerations:

1. **Using the camera output for steering the robot could be problematic.** The camera code in this example that captures and processes images runs at a much slower rate that is desirable for a control loop for steering the robot. A better, and only slightly more complex solution, is to get headings from the camera and it's processing rate, then have a much faster control loop steering to those headings using a gyro sensor.
2. **Keep the vision code in the class that wraps the pipeline.** A better way of writing object oriented code is to subclass or instantiate the generated pipeline class and process the OpenCV results there rather than in the robot program. In this example, the robot code extracts the direction to drive by manipulating the resultant OpenCV contours. By having the OpenCV code exposed throughout the robot program it makes it difficult to change the vision algorithm should you have a better one.

### 31.1 Iterative program definitions

Java

```
package org.usfirst.frc.team190.robot;

import org.usfirst.frc.team190.grip.MyVisionPipeline;

import org.opencv.core.Rect;
import org.opencv.imgproc.Imgproc;

import edu.wpi.cscore.UsbCamera;
import edu.wpi.first.wpilibj.CameraServer;
```

(continues on next page)

(continued from previous page)

```

import edu.wpi.first.wpilibj.IterativeRobot;
import edu.wpi.first.wpilibj.RobotDrive;
import edu.wpi.first.wpilibj.vision.VisionRunner;
import edu.wpi.first.wpilibj.vision.VisionThread;

public class Robot extends IterativeRobot {

    private static final int IMG_WIDTH = 320;
    private static final int IMG_HEIGHT = 240;

    private VisionThread visionThread;
    private double centerX = 0.0;
    private RobotDrive drive;

    private final Object imgLock = new Object();

```

In this first part of the program you can see all the import statements for the WPILib classes used for this program.

- The **image width and height** are defined as 320x240 pixels.
- The **VisionThread** is a WPILib class that makes it easy to do your camera processing in a separate thread from the rest of the robot program.
- **centerX** value will be the computed center X value of the detected target.
- **RobotDrive** encapsulates the 4 motors on this robot and allows simplified driving.
- **imgLock** is a variable to synchronize access to the data being simultaneously updated with each image acquisition pass and the code that's processing the coordinates and steering the robot.

Java

```

@Override
public void robotInit() {
    UsbCamera camera = CameraServer.getInstance().startAutomaticCapture();
    camera.setResolution(IMG_WIDTH, IMG_HEIGHT);

    visionThread = new VisionThread(camera, new MyVisionPipeline(), pipeline -> {
        if (!pipeline.filterContoursOutput().isEmpty()) {
            Rect r = Imgproc.boundingRect(pipeline.filterContoursOutput().get(0));
            synchronized (imgLock) {
                centerX = r.x + (r.width / 2);
            }
        }
    });
    visionThread.start();

    drive = new RobotDrive(1, 2);
}

```

The **robotInit()** method is called once when the program starts up. It creates a **CameraServer** instance that begins capturing images at the requested resolution (IMG\_WIDTH by IMG\_HEIGHT).

Next an instance of the class **VisionThread** is created. **VisionThread** begins capturing images from the camera asynchronously in a separate thread. After processing each image, the pipeline computed **bounding box** around the target is retrieved and its **center X** value is computed. This **centerX** value will be the x pixel value of the center of the rectangle in the image.

The **VisionThread** also takes a **VisionPipeline** instance (here, we have a subclass **MyVisionPipeline** generated by GRIP) as well as a callback that we use to handle the output of the pipeline. In this example, the pipeline outputs a list

of contours (outlines of areas in an image) that mark goals or targets of some kind. The callback finds the bounding box of the first contour in order to find its center, then saves that value in the variable `centerX`. Note the synchronized block around the assignment: this makes sure the main robot thread will always have the most up-to-date value of the variable, as long as it also uses **synchronized** blocks to read the variable.

Java

```
@Override
public void autonomousPeriodic() {
    double centerX;
    synchronized (imgLock) {
        centerX = this.centerX;
    }
    double turn = centerX - (IMG_WIDTH / 2);
    drive.arcadeDrive(-0.6, turn * 0.005);
}
```

This, the final part of the program, is called repeatedly during the **autonomous period** of the match. It gets the **centerX** pixel value of the target and **subtracts half the image width** to change it to a value that is **zero when the rectangle is centered** in the image and **positive or negative when the target center is on the left or right side of the frame**. That value is used to steer the robot towards the target.

Note the **synchronized** block at the beginning. This takes a snapshot of the most recent `centerX` value found by the `VisionThread`.



---

### Using GRIP with a Kangaroo Computer

---

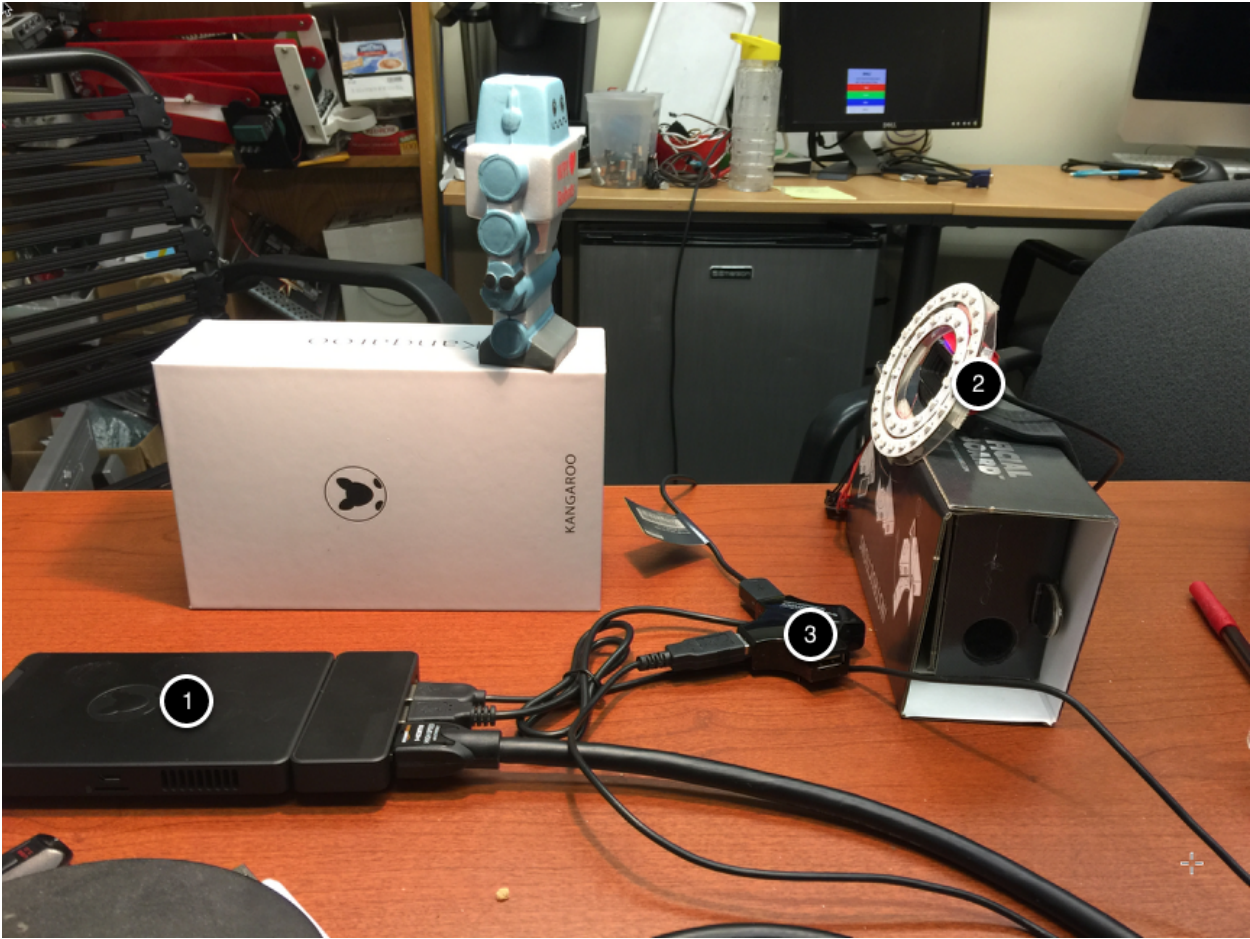
A recently available computer called the Kangaroo looks like a great platform for running GRIP on FRC robots. Some of the specs for this processor include:

- Quad core 1.4Ghz Atom processor
- HDMI port
- 2 USB ports (1 USB2 and 1 USB3)
- 2GB RAM
- 32GB Flash
- Flash card slot
- WiFi
- Battery with 4 hours running time
- Power supply
- Windows 10
- and a fingerprint reader

The advantage of this setup is that it offloads the roboRIO from doing image processing and it is a normal Windows system so all of our software should work without modification. Be sure to read the caveats at the end of this page before jumping in.

**More detailed instructions** for using a Kangaroo for running GRIP can be found in the following PDF document created by Scott Taylor and FRC 1735. His explanation goes beyond what is shown here, detailing how to get the GRIP program to auto-start on boot and many other details.

[Grip Plus Kangaroo](#)

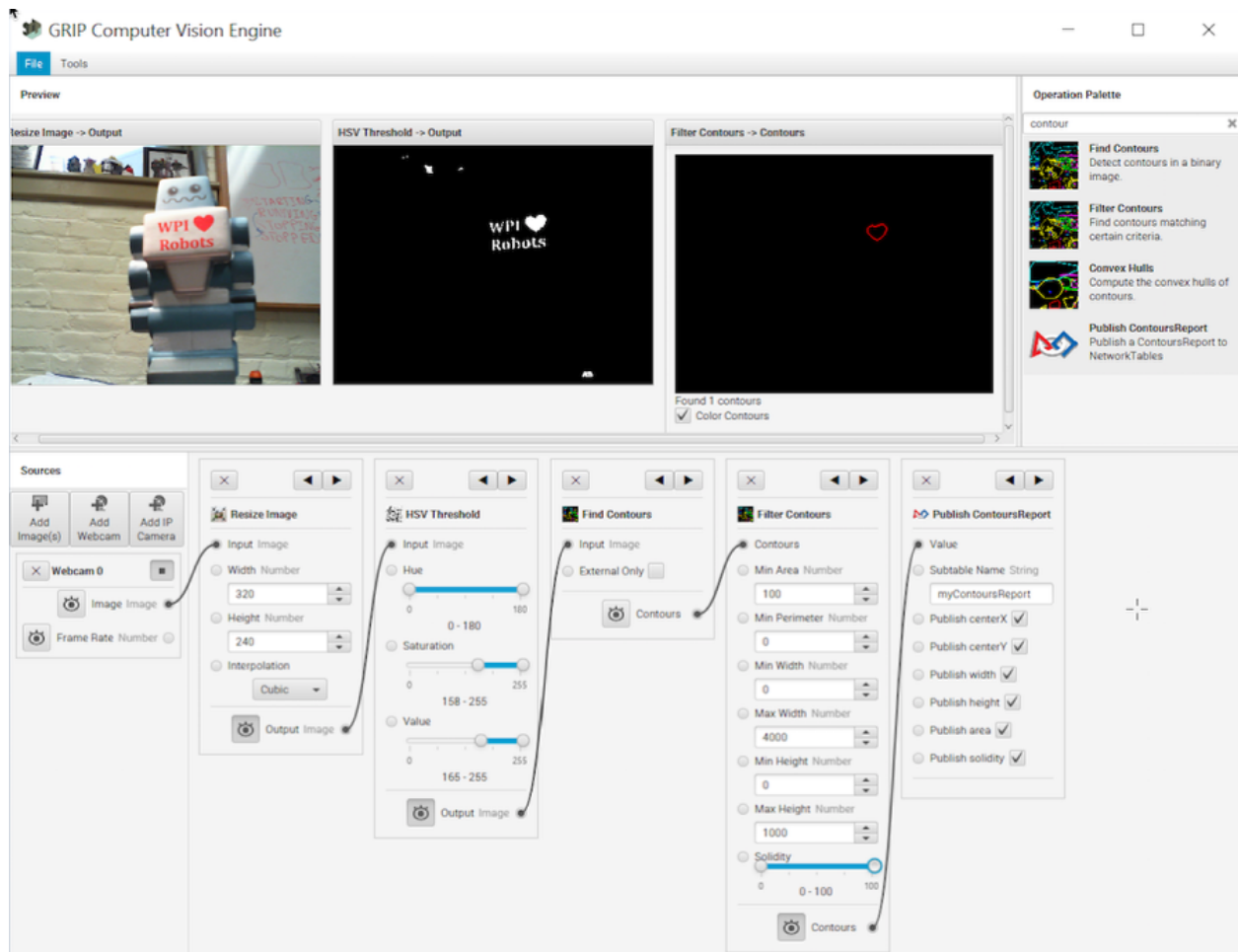


## 32.1 Setup

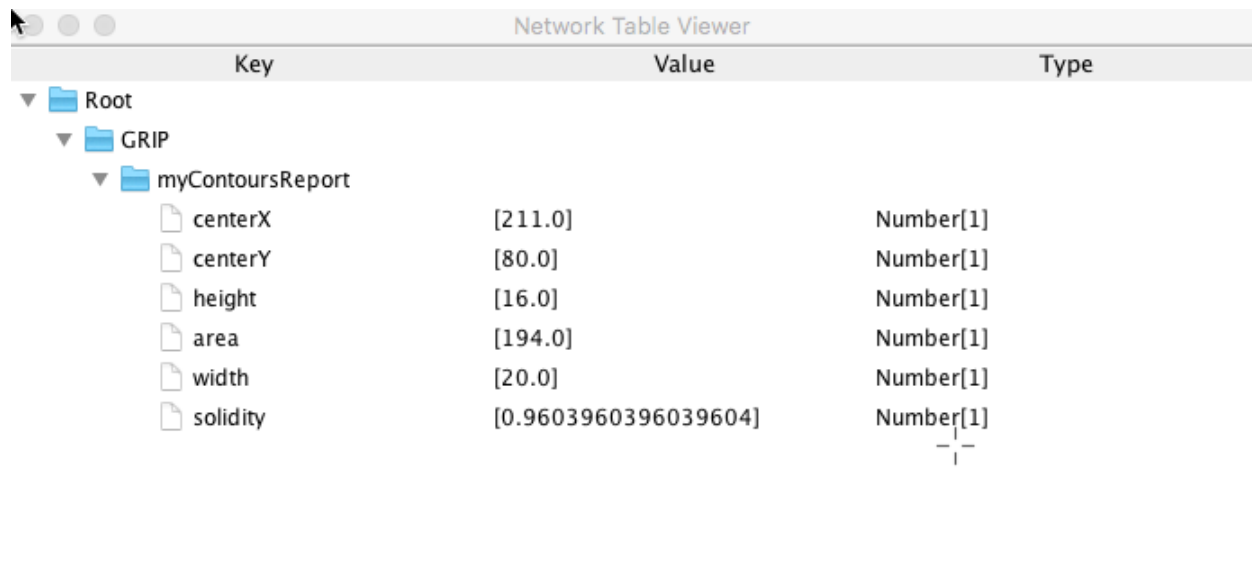
The nice thing about this setup is that you just need to plug in a monitor, keyboard, mouse and (in this case) the Microsoft web camera and you are good to go with programming the GRIP pipeline. When you are finished, disconnect the keyboard, mouse and monitor and put the Kangaroo on your robot. You will need to disable the WiFi on the Kangaroo and connect it to the robot with a USB ethernet dongle to the extra ethernet port on the robot radio.

In this example you can see the Kangaroo computer (1) connected to a USB hub (3), keyboard, and an HDMI monitor for programming. The USB hub is connected to the camera and mouse.

## 32.2 Sample GRIP program



Attached is the sample program running on the Kangaroo detecting the red heart on the little foam robot in the image (left panel). It is doing a HSV threshold to only get that red color then finding contours, and then filtering the contours using the size and solidity. At the end of the pipeline, the values are being published to NetworkTables.



The screenshot shows a window titled "Network Table Viewer" with a tree view on the left and a table on the right. The tree view shows a hierarchy: Root > GRIP > myContoursReport. The table has three columns: Key, Value, and Type. The data rows are as follows:

Key	Value	Type
centerX	[211.0]	Number[1]
centerY	[80.0]	Number[1]
height	[16.0]	Number[1]
area	[194.0]	Number[1]
width	[20.0]	Number[1]
solidity	[0.9603960396039604]	Number[1]

### 32.3 Viewing Contours Report in NetworkTables

This is the output from the OutlineViewer (<username>/WPILib/tools/OutlineViewer.jar), running on a different computer as a server (since there is no roboRIO on the network in this example) and the values being reported back for the single contour that the program detected that met the requirements of the Filter Contours operation.

### 32.4 Considerations

The Kangaroo runs Windows 10, so care must be taken to make sure GRIP will keep running on the robot during a match or testing. For example, it should not try to do a Windows Update, Virus scan refresh, go to sleep, ect. Once configured, it has the advantage of being a normal Intel Architecture and should give predictable performance since it is running only one application.



---

### Using a Coprocessor for vision processing

---

Vision processing using libraries like OpenCV for recognizing field targets or game pieces can often be a CPU intensive process. Often the load isn't too significant and the processing can easily be handled by the roboRIO. In cases where there are more camera streams or the image processing is complex, it is desirable to off-load the roboRIO by putting the code and the camera connection on a different processor. There are a number of choices of processors that are popular in FRC such as the Raspberry PI, the intel-based Kangaroo, the LimeLight for the ultimate in simplicity, or for more complex vision code a graphics accelerator such as one of the nVidia Jetson models.

#### 33.1 Strategy

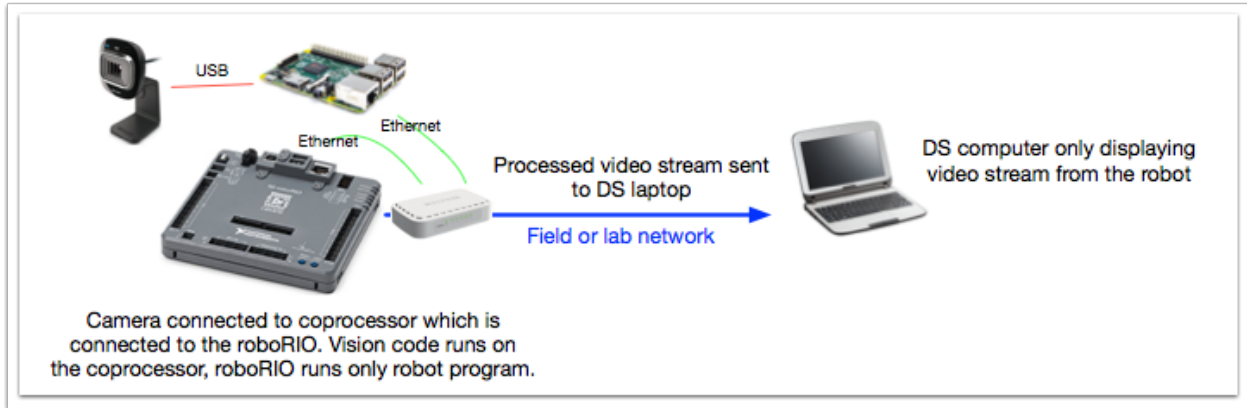
Generally the idea is to set up the coprocessor with the required software that generally includes:

- OpenCV - the open source computer vision library
- Network tables - to commute the results of the image processing to the roboRIO program
- Camera server library - to handle the camera connections and publish streams that can be viewed on a dashboard
- The language library for whatever computer language is used for the vision program
- The actual vision program that does the object detection

The coprocessor is connected to the roboRIO network by plugging it into the extra ethernet port on the network router or, for more connections, adding a small network switch to the robot. The cameras are plugged into the coprocessor, it acquires the images, processes them, and publishes the results, usually target location information, to network tables so it is can be consumed by the robot program for steering and aiming.

#### 33.2 Streaming camera data to the dashboard

It is often desirable to simply stream the camera data to the dashboard over the robot network. In this case one or more camera connections can be sent to the network and viewed on a dashboard such as Shuffleboard or a web browser. Using Shuffleboard has the advantage of having easy controls to set the camera resolution and bit rate as well as integrating the camera streams with other data sent from the robot.



It is also possible to process images and add annotation to the image, such as target lines or boxes showing what the image processing code has detected then send it forward to the dashboard to make it easier for operators to see a clear picture of what's around the robot.

---

## Using the Raspberry PI for FRC

---

One of the most popular coprocessor choices is the Raspberry PI because:

- Low cost - around \$35
- High availability - it's easy to find Raspberry PIs from a number of suppliers, including Amazon
- Very good performance - the current Raspberry PI 3b+ has the following specifications:
- Technical Specifications: - Broadcom BCM2837BO 64 bit ARMv8 QUAD Core A53 64bit Processor powered Single Board Computer run at 1.4GHz - 1GB RAM - BCM43143 WiFi on board - Bluetooth Low Energy (BLE) on board - 40 pin extended GPIO - 4 x USB2 ports - 4 pole Stereo output and Composite video port - Full size HDMI - CSI camera port for connecting the Raspberry - Pi camera - DSI display port for connecting the Raspberry - Pi touch screen display - MicroSD port for loading your operating system and storing data - Upgraded switched Micro USB power source (now supports up to 2.5 Amps).

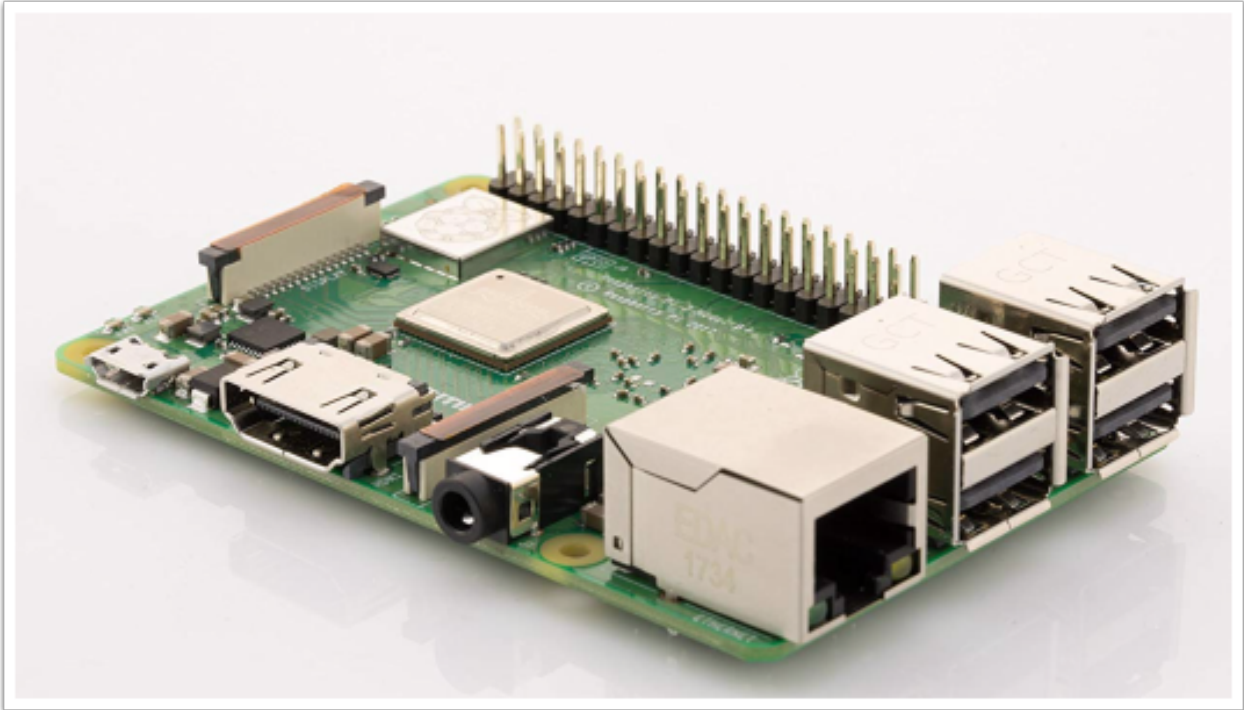
### 34.1 Pre-built Raspberry PI image

To make using the Raspberry PI as easy as possible for teams, there is a provided Raspberry PI image. The image can be copied to a micro SD card, inserted into the PI, and booted. By default it supports:

- A web interface for configuring it for the most common functions
- Supports an arbitrary number camera streams (defaults to one) that are published on the network interface
- OpenCV, Network Tables, Camera Server, and language libraries for C++, Java, and Python custom programs

If the only requirement is to stream one or more cameras to the network (and dashboard) then no programming is required and can be completely set up through the web interface.

The next section discusses how to install the image onto a flash card and boot the PI.



---

### What you need to get the PI image running

---

To start using the Raspberry PI as a video or image coprocessor you need the following:

- A Raspberry PI 3 B or Raspberry PI 3 B+
- A micro SD card that is at least 4Gb to hold all the provided software
- An ethernet cable to connect the PI to your roboRIO network
- A USB micro power cable to connect to the Voltage Regulator Module (VRM) on your robot. It is recommended to use the VRM connection for power rather than powering it from one of the roboRIO USB ports for higher reliability
- A laptop that can write the MicroSD card, either using a USB dongle (preferred) or a SD to MicroSD adapter that ships with most MicroSD cards



Shown is an inexpensive USB dongle that will write the FRC image to the MicroSD card.

---

## Installing the image to your MicroSD card

---

### 36.1 Getting the FRC Raspberry PI image

The image is stored on the GitHub release page for the FRC PI-gen repository: <https://github.com/wpilibsuite/FRCVision-pi-gen/releases>.

In addition to the instructions on this page, see the documentation on the GitHub web page (below).

The image is fairly large so have a fast internet connection when downloading it. Always use the most recent release from the top of the list of releases.

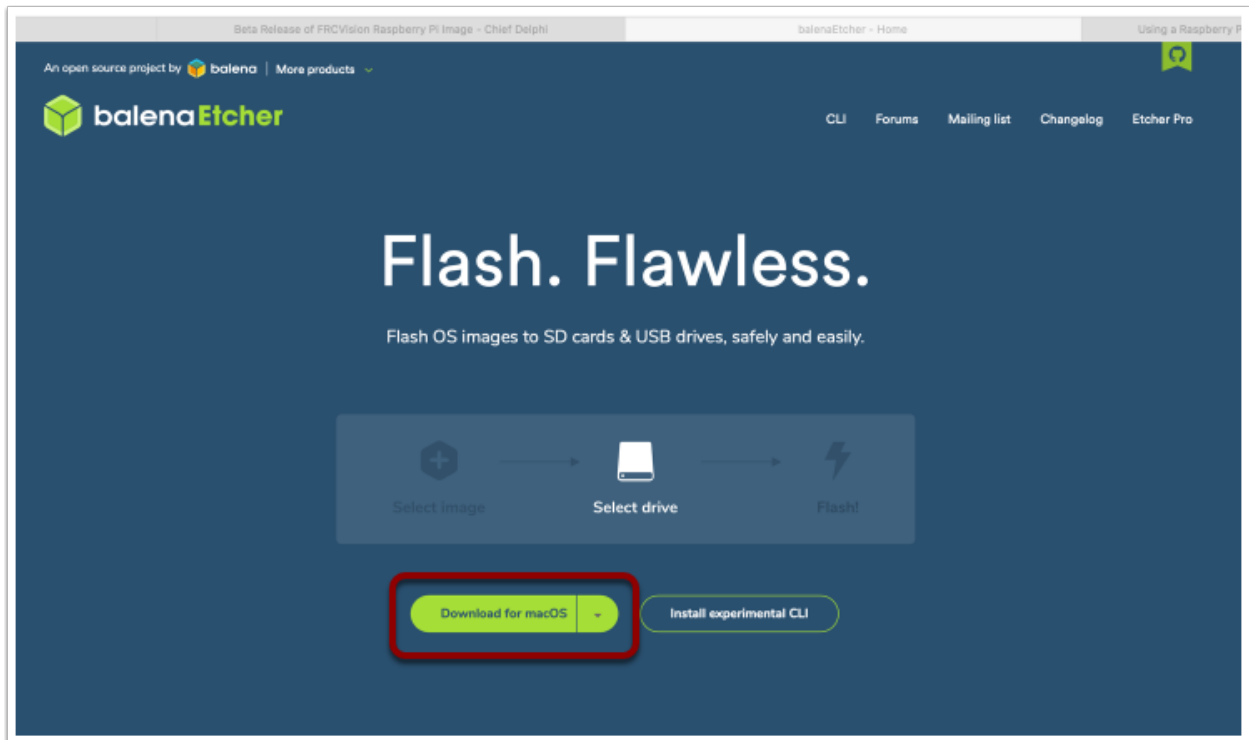
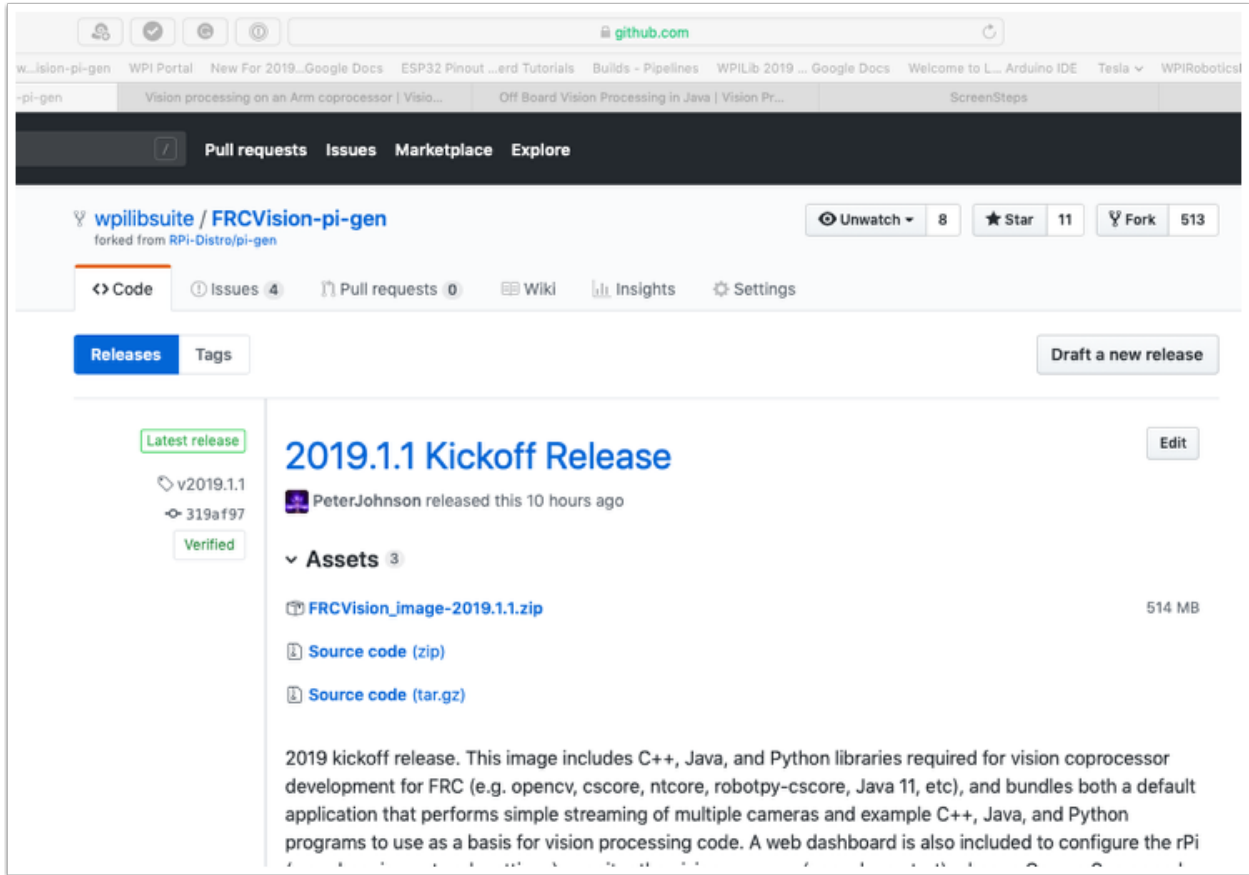
### 36.2 Copy the image to your MicroSD card

Download and install Etcher (<https://www.balena.io/etcher/>) to image the micro SD card. The micro SD card needs to be at least 4 GB. Note: a micro SD to USB dongle such as <https://www.amazon.com/gp/product/B0779V61XB> works well for writing to micro SD cards.

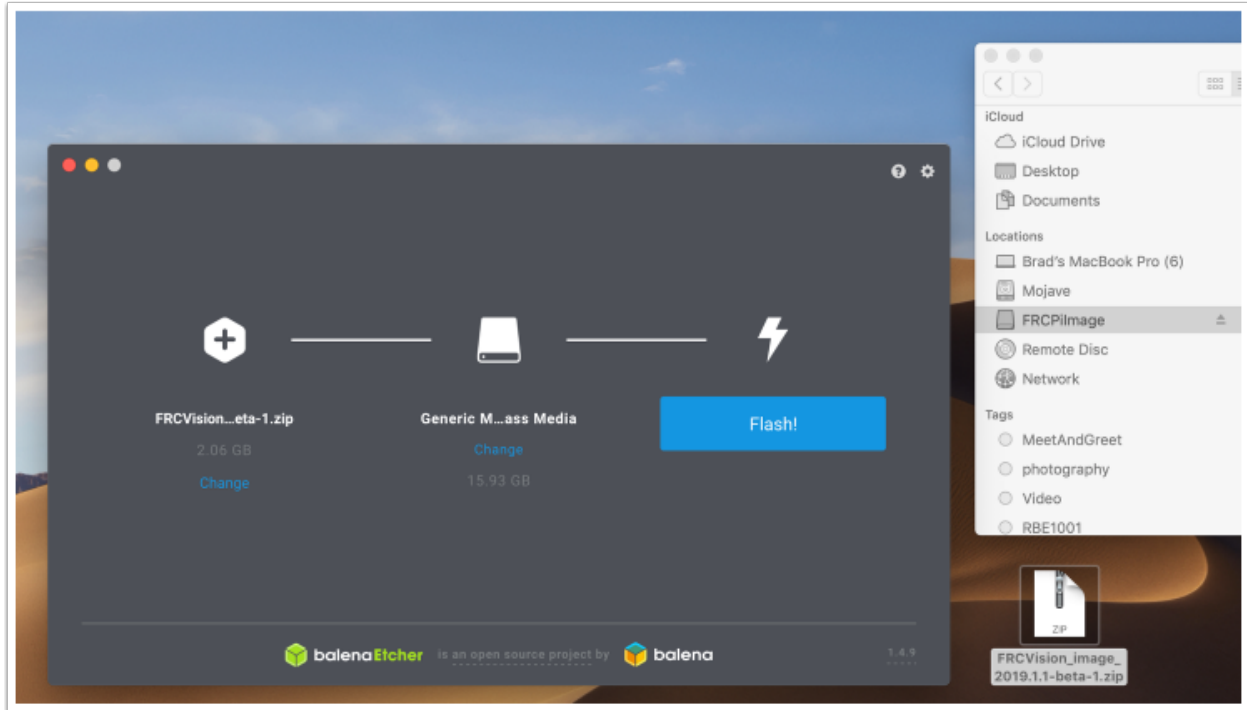
Flash the MicroSD card with the image using Etcher by selecting the zip file as the source, your SD card as the destination and click “Flash”. Expect the process to take about 3 minutes on a fairly fast laptop.

#### 36.2.1 Testing the Raspberry PI

1. Put the micro SD card in a rPi 3 and apply power.
2. Connect the rPi 3 ethernet to a LAN or PC. Open a web browser and connect to <http://frcvision.local/> to open the web dashboard. On the first bootup the filesystem will be writable, but later bootups will default to read only, so it’s necessary to click the “writable” button to make changes.







### 36.2.2 Logging into the Raspberry Pi

Most tasks with the rPi can be done from the web console interface. Sometimes for advanced use such as program development on the rPi it is necessary to log in. To log in, use the default Raspberry PI password:

```
Username: pi
Password: raspberry
```

Known issues with this release (errata). These will be fixed in the next release.

- Downloading the Java example program on the web dashboard does not work. Grab it manually from `/home/pi/zips` instead (using ssh).
- Console output enable button on the web dashboard doesn't persist correctly through a rPi reboot (it needs to be disabled and re-enabled to again get console output).



### 37.1 FRC Console

The FRC image for the Raspberry PI includes a console that can be viewed in any web browser that makes it easy to:

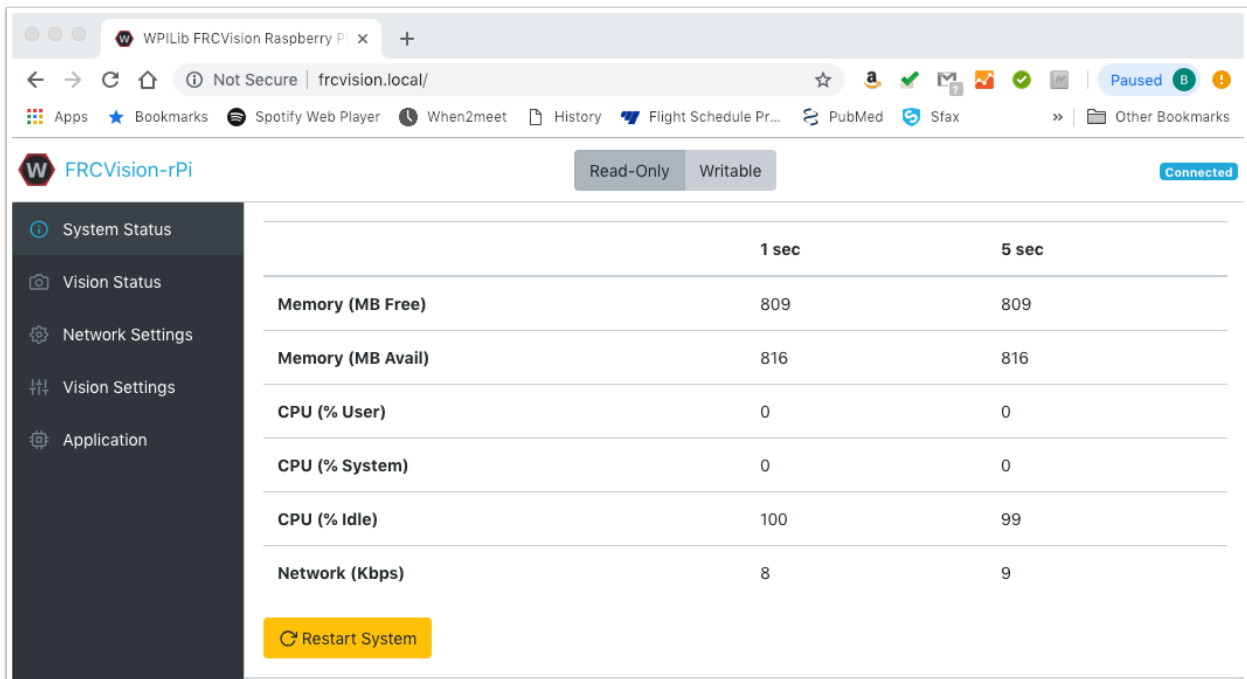
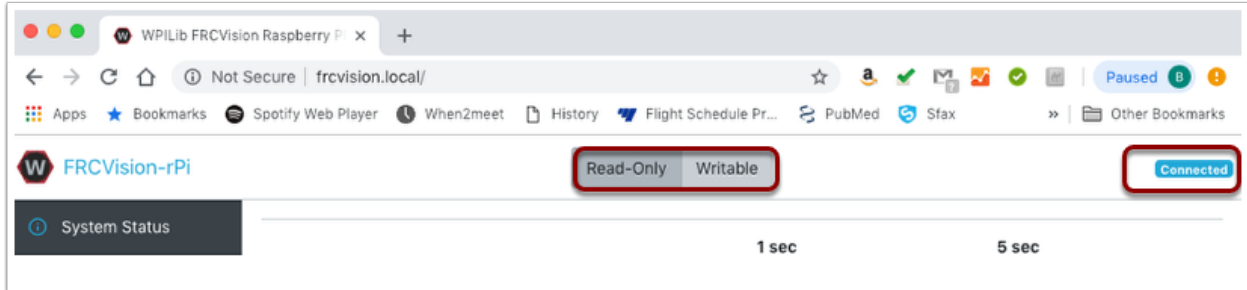
- Look at the Raspberry PI status
- View the status of the background process running the camera
- View or change network settings
- Look at each camera plugged into the rPI and add additional cameras
- Load a new vision program onto the rPI

#### 37.1.1 Setting the rPI to be Read-Only vs. Writable

The rPI is normally set to Read-Only which means that the file system cannot be changed. This ensures that if power is removed without first shutting down the rPi the file system isn't corrupted. When settings are changed (following sections), the new settings cannot be saved while the rPI file system is set as Read-Only. Buttons are provided that allow the file system to be changed from Read-Only to Writable and back whenever changes are made. If the other buttons that change information stored on the rPI cannot be pressed, check the Read-Only status of the system.

#### 37.1.2 Status of the network connection to the rPI

There is a label in the top right corner of the console that indicates if the rPi is currently connected. It will change from Connected to Disconnected if there is no longer a network connection to the rPi.



## 37.2 System status

The system status shows what the CPU on the rPI is doing at any time. There are two columns of status values, one being a 1 second average and the other a 5 second average. Shown is:

- free and available RAM on the PI
- CPU usage for user processes and system processes as well as idle time.
- And network bandwidth - which allows one to determine if the used camera bandwidth is exceeding the maximum bandwidth allowed in the robot rules for any year.

## 37.3 Vision Status

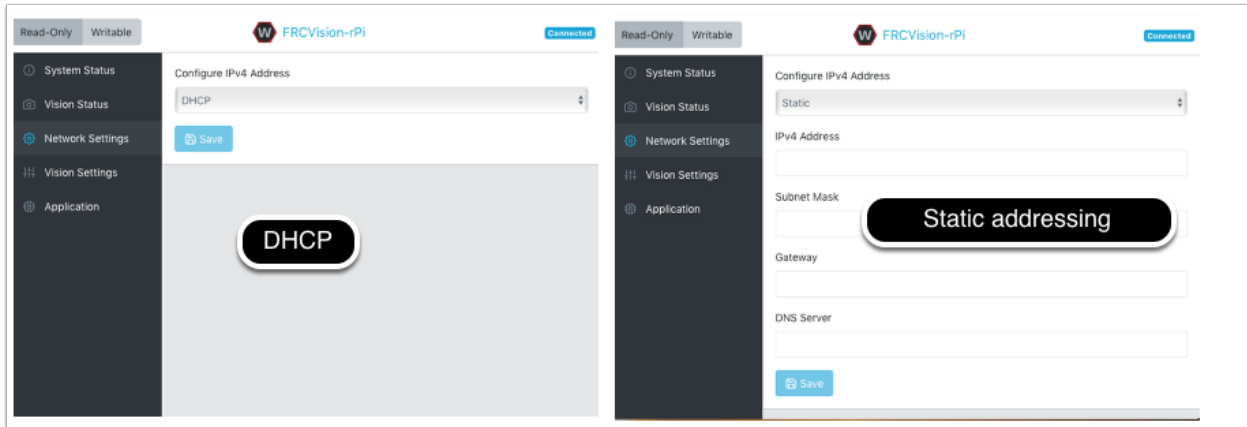
The screenshot shows the FRCVision-rPi interface. At the top, there are tabs for 'Read-Only' and 'Writable', the FRCVision-rPi logo, and a 'Disconnected' status indicator. A sidebar on the left contains navigation options: System Status, Vision Status, Network Settings, Vision Settings, and Application. The main content area is divided into two sections. The first section, 'Background Service (Automatic Restart)', shows an 'Unknown Status' and four control buttons: 'Up', 'Down', 'Terminate', and 'Kill'. The second section, 'Console Output', has an 'Enable' toggle switch turned on. Below the toggle, the console displays several error messages from the Network Tables (NT) service, indicating connection failures to 10.2.94.2 and 172.22.11.2 on port 1735, and errors related to resolving addresses like 'roboRIO-294-FRC.local' and 'roboRIO-294-FRC.frc-field.local'.

Allows monitoring of the task which is running the camera code in the rPI, either one of the default programs or your own program in Java, C++, or Python. You can also enable and view the console output to see messages coming from the background camera service. In this case there are number of messages about being unable to connect to network tables (NT: connect()) because in this example the rPI is simply connected to a laptop with no Network Tables server running (usually the roboRIO.)

## 37.4 Network Settings

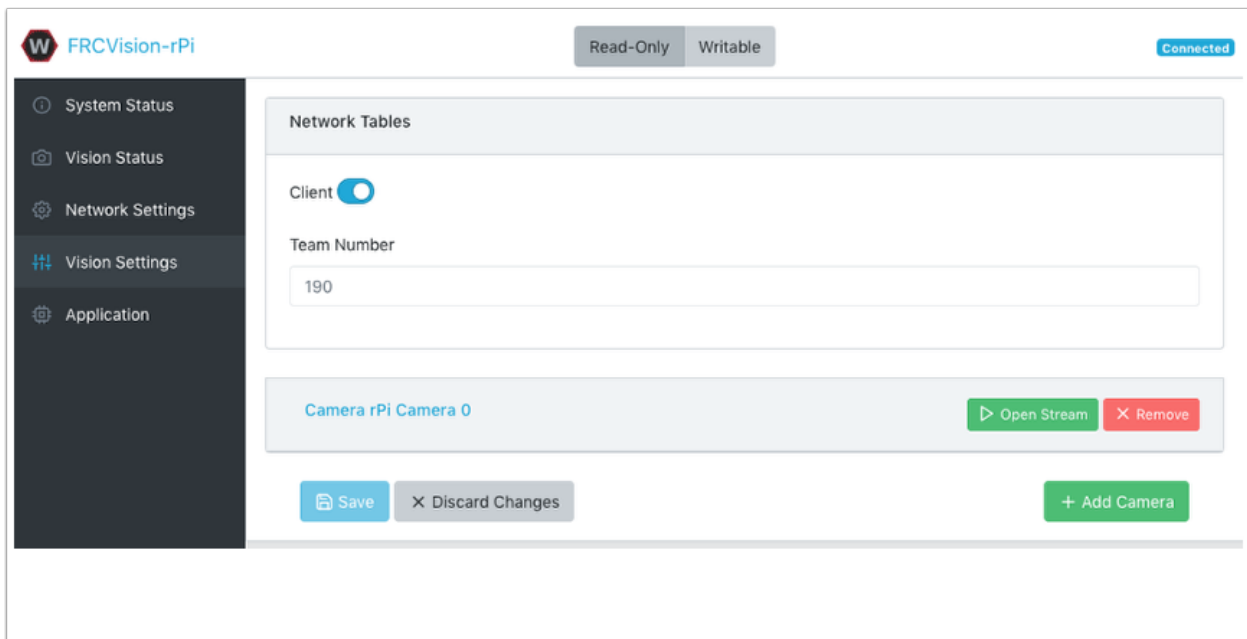
The rPI network settings have options to connect to the PI:

- DHCP - the default name resolution usually used by the roboRIO. The default name is raspberry.local.
- Static - where a fixed IP address, network mask, and router settings are filled in explicitly
- DHCP with Static Fallback - DHCP with Static Fallback - the PI will try to get an IP address via DHCP, but if it can't find a DHCP server, it will use the provided static IP address and parameters



The picture above is showing the settings for both DHCP and Static IP Addressing. The mDNS name for the rPi should always work regardless of the options selected above.

## 37.5 Vision Settings



The Vision Settings are to set the parameters for each camera and whether the rPi should be a NetworkTables client or server. There can only be one server on the network and the roboRIO is always a server. Therefore when connected to a roboRIO, the rPi should always be in client mode with the team number filled in. If testing on a desktop setup with no roboRIO or anything acting as a server then it should be set to Server (Client switch is off).

To view and manipulate all the camera settings click on the camera in question. In this case the camera is called “Camera rPi Camera 0” and clicking on the name reveals the current camera view and the associated settings.

Manipulating the camera settings is reflected in the current camera view. The bottom of the page shows all the possible camera modes (combinations of Width, Height, and frame rates) that are supported by this camera.

### 37.5.1 Getting the current settings to persist over reboots

The rPi will load all the camera settings on startup. Editing the camera configuration in the above screen is temporary. To make the values persist click on the “Load Source Config From Camera” button and the current settings will be filled in on the camera settings fields. Then click “Save” at the bottom of the page. Note: you must set the file system Writeable in order to save the settings. *The Writeable button is at the top of the page.*

There are some commonly used camera settings values shown in the camera settings (above). These values Brightness, White Balance, and Exposure are loaded into the camera before the user JSON file is applied. So if a user JSON file contains those settings they will overwrite the ones from the text field.

## 37.6 Application

The Application tab shows the application that is currently running on the rPi.

### 37.6.1 Vision workflows

There is a sample vision program using OpenCV in each of the supported languages, C++, Java, or Python. Each sample program can capture and stream video from the rPi. In addition, the samples have some minimal OpenCV. They are all set up to be extended to replace the provided OpenCV sample code with the code needed for the robot application. The rPi Application tab supports a number of programming workflows:

- Stream one or more cameras from the rPi for consumption on the driver station computer and displayed using ShuffleBoard
- Edit and build one of the sample programs (one for each language: Java, C++ or Python) on the rPi using the included toolchains
- Download a sample program for the chosen language and edit and build it on your development computer. Then upload that built program back to the rPi
- Do everything yourself using completely custom applications and scripts (probably based on one of the samples)

The running application can be changed by selecting one of the choices in the drop-down menu. The choices are:

- Built-in multi camera streaming which streams whatever cameras are plugged into the rPi. The camera configuration including number of cameras can be set on the “Vision Settings” tab.
- Custom application which doesn’t upload anything to the rPi and assumes that the developer wants to have a custom program and script.
- Java, C++ or Python pre-installed sample programs that can be edited into your own application.
- Java, C++, or Python uploaded program. Java programs require a .jar file with the compiled program and C++ programs require an rPi executable to be uploaded to the rPi.

When selecting one of the Upload options, a file chooser is presented where the jar, executable or Python program can be selected and uploaded to the rPi. In the following picture an Uploaded Java jar is chosen and the “Choose File” button will select a file and clicking on the “Save” button will upload the selected file.

connect\_verbose  1

brightness  50

contrast  50

saturation  50

white\_balance\_temperature\_auto

gain  0

power\_line\_frequency  Disabled  50 Hz  60 Hz

white\_balance\_temperature  4000

sharpness  50

backlight\_compensation  0

exposure\_auto  Manual Mode  Aperture Priority Mode

exposure\_absolute  12

exposure\_auto\_priority

pan\_absolute  0

tilt\_absolute  0

focus\_absolute  0

focus\_auto

zoom\_absolute  100

Supported Video Modes:

Pixel Format	Width	Height	FPS
YUYV	640	480	30
YUYV	640	480	24
YUYV	640	480	15

[Settings JSON](#) | [Source Config JSON](#)



**Network Tables**

Client

Team Number

**Camera rPi Camera 0**

Name:  Path:

Load Source Config From JSON File

Pixel Format	Width	Height	FPS
MJPEG ▾	<input style="width: 80%;" type="text" value="160"/>	<input style="width: 80%;" type="text" value="120"/>	<input style="width: 80%;" type="text" value="30"/>

Brightness:  White Balance:  Exposure:

Custom Properties JSON

```

[{"name": "connect_verbose", "value": 1}, {"name": "contrast", "value": 50}, {"name": "saturation", "value": 50},
{"name": "gain", "value": 0}, {"name": "power_line_frequency", "value": 2}, {"name": "sharpness", "value": 50},
{"name": "backlight_compensation", "value": 0}, {"name": "exposure_auto_priority", "value": true},

```

These settings are filled in when copying the Source Config from the camera

The screenshot shows the FRCVision-rPi interface. At the top, there are tabs for 'Read-Only' and 'Writable', the 'FRCVision-rPi' logo, and a 'Connected' status indicator. A left sidebar contains a menu with options: System Status, Vision Status, Network Settings, Vision Settings, and Application (which is highlighted). The main content area is titled 'Vision Application Configuration'. It features a dropdown menu for 'Application' currently set to 'Built-in multi-camera streaming', and a blue 'Save' button. Below this is an 'Example Programs' section with three green buttons: 'Download Java Example', 'Download C++ Example', and 'Download Python Example'.

This is a close-up of the 'Application' dropdown menu. The title 'Vision Application Configuration' is at the top. The dropdown is open, showing a list of options: 'Built-in multi-camera streaming' (with a checkmark), 'Custom', 'Java example (must be built to work)', 'C++ example (must be built to work)', 'Python example', 'Uploaded Java jar', 'Uploaded C++ executable' (highlighted in blue), and 'Uploaded Python file'. Below the dropdown, the 'Example Programs' section is visible with the same three download buttons: 'Download Java Example', 'Download C++ Example', and 'Download Python Example'.

Note: in order to Save a new file onto the rPi, the file system has to be set writeable using the “Writable” button at the top left of the web page. After saving the new file, set the file system back to “Read-Only” so that it is protected against accidental changes.

**Application**

Uploaded Java jar

**Upload executable file**

Choose File no file selected

Save

**Example Programs**

Download Java Example

Download C++ Example

Download Python Example



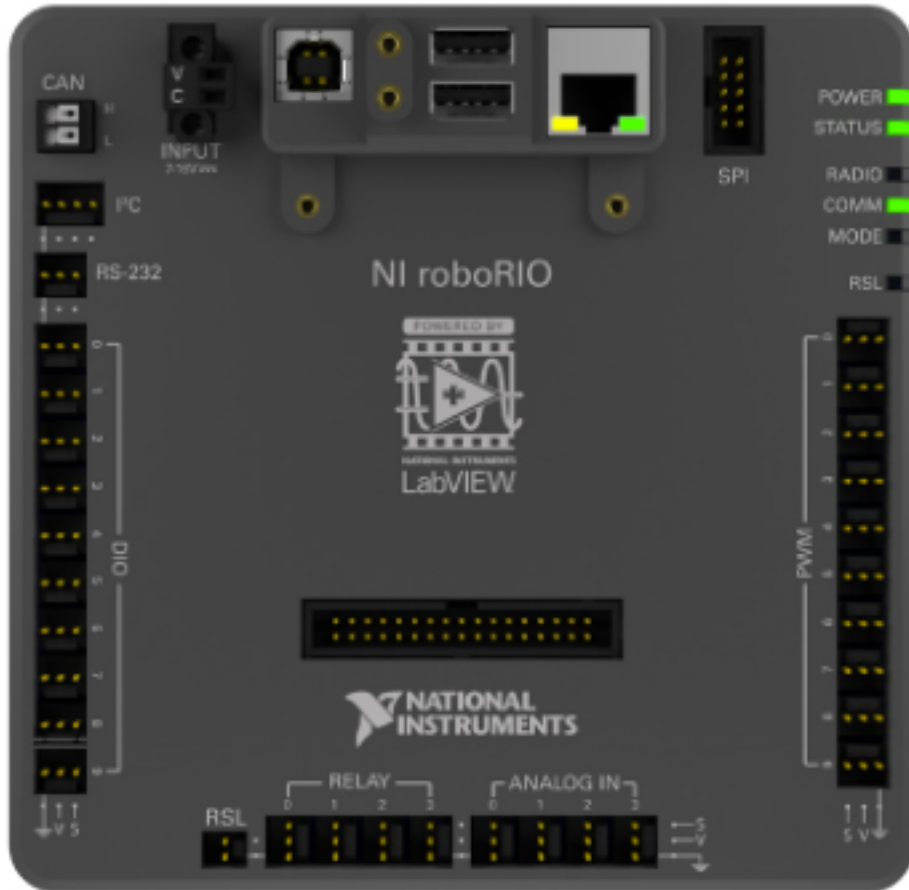
---

### FRC Control System Hardware Overview

---

The goal of this document is to provide a brief overview of the hardware components that make up the FRC Control System. Each component will contain a brief description of the component function, a brief listing of critical connections, and a link to more documentation if available. Note that for complete wiring instructions/diagrams, please see the Wiring the FRC Control System document.

## 38.1 National Instruments roboRIO



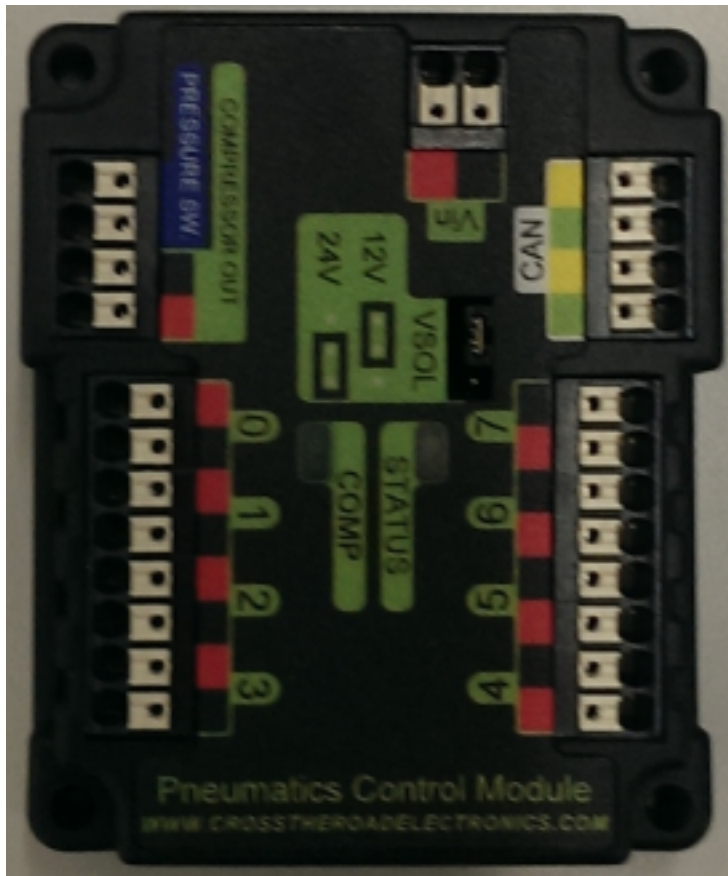
The NI-roboRIO is the main robot controller used for FRC. The roboRIO includes a dual-core ARM Cortex™-A9 processor and FPGA which runs both trusted elements for control and safety as well as team-generated code. Integrated controller I/O includes a variety of communication protocols (Ethernet, USB, CAN, SPI, I2C, and serial) as well as PWM, servo, digital I/O, and analog I/O channels used to connect to robot peripherals for sensing and control. The roboRIO should connect to the dedicated 12V port on the Power Distribution Panel for power. Wired communication is available via USB or Ethernet. Detailed information on the roboRIO can be found in the [roboRIO User Manual](#).

## 38.2 Power Distribution Panel



The Power Distribution Panel (PDP) is designed to distribute power from a 12VDC battery to various robot components through auto-resetting circuit breakers and a small number of special function fused connections. The PDP provides 8 output pairs rated for 40A continuous current and 8 pairs rated for 30A continuous current. The PDP provides dedicated 12V connectors for the roboRIO, as well as connectors for the Voltage Regulator Module and Pneumatics Control Module. It also includes a CAN interface for logging current, temperature, and battery voltage. For more detailed information, see the [PDP User Manual]([http://www.ctr-electronics.com/control-system/pdp.html#product\\_tabs\\_technical\\_resources](http://www.ctr-electronics.com/control-system/pdp.html#product_tabs_technical_resources) "PDP User Manual").

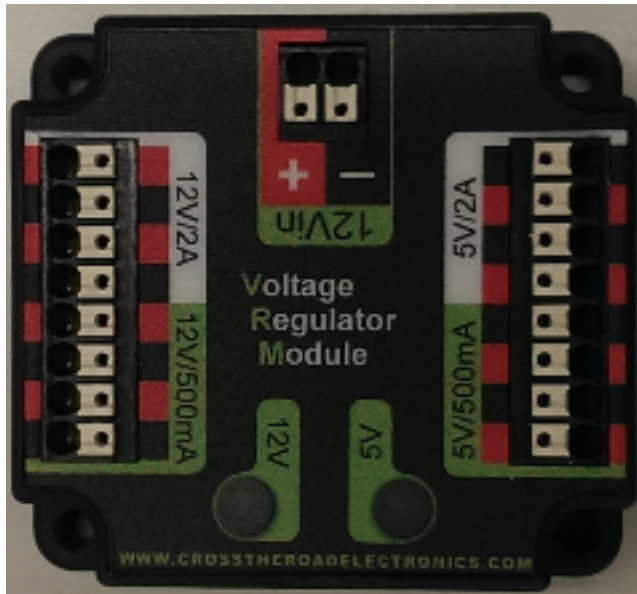
### 38.3 Pneumatics Control Module



The PCM is a device that contains all of the inputs and outputs required to operate 12V or 24V pneumatic solenoids and the on board compressor. The PCM is enabled/disabled by the roboRIO over the CAN interface. The PCM contains an input for the pressure sensor and will control the compressor automatically when the robot is enabled and a solenoid has been created in the code. The device also collects diagnostic information such as solenoid states, pressure switch state, and compressor state. The module includes diagnostic LED's for both CAN and the individual solenoid channels. For more information see the [PCM User Manual](#).



## 38.4 Voltage Regulator Module



The VRM is an independent module that is powered by 12 volts. The device is wired to a dedicated connector on the PDP. The module has multiple regulated 12V and 5V outputs. The purpose of the VRM is to provide regulated power for the robot radio, custom circuits, and IP vision cameras. **Note: The two connector pairs associated with each label have a combined rating of what the label indicates (e.g. 5V/500mA total for both pairs not for each pair). The 12V/2A limit is a peak rating, the supply should not be loaded with more than 1.5A continuous current draw.** For more information, see the [VRM User Manual](#).

## 38.5 Motor Controllers

There are a variety of different motor controllers which work with the FRC Control System and are approved for use. These devices are used to provide variable voltage control of the brushed DC motors used in FRC. They are listed here in alphabetical order.

### 38.5.1 DMC-60 and DMC-60C Motor Controller

---



The DMC-60 is a PWM motor controller from Digilent. The DMC-60 features integrated thermal sensing and protection including current-foldback to prevent overheating and damage, and four multi-color LEDs to indicate speed, direction, and status for easier debugging. For more information, see the DMC-60 reference manual: <https://reference.digilentinc.com/dmc-60/reference-manual>

The DMC-60C adds CAN smart controller capabilities to the DMC-60 controller. This enables closed loop control features and other intelligent control options. For more information see the DMC-60C Product Page: <https://store.digilentinc.com/dmc60c-digital-motor-controller-approved-for-first-robotics/>

### 38.5.2 Jaguar Motor Controller



The Jaguar Motor Controller from VEX Robotics (formerly made by Luminary Micro and Texas Instruments) is a variable speed motor controller for use in FRC. For FRC, the Jaguar may only be controlled using the PWM interface. For more information, see the Jaguar Getting Started Guide, Jaguar Datasheet and Jaguar FAQ on [this page](#).

### 38.5.3 SD540B and SD540C Motor Controllers



The SD540 Motor Controller from Mindsensors is a variable speed motor controller for use in FRC. The SD540B is controlled using the PWM interface. The SD540C is controllable over CAN. Limit switches may be wired directly to the SD540 to limit motor travel in one or both directions. Switches on the device are used to flip the direction of motor travel, configure the wiring polarity of limit switches, set Brake or Coast mode, and put the device in calibration mode. For more information see the Mindsensors FRC page: <http://www.mindsensors.com/68-frc>

### 38.5.4 SPARK Motor Controller



The SPARK Motor Controller from REV Robotics is a variable speed motor controller for use in FRC. The SPARK is controlled using the PWM interface. Limit switches may be wired directly to the SPARK to limit motor travel in one or both directions. The RGB status LED displays the current state of the device including whether the device is currently in Brake mode or Coast mode. For more information, see the REV Robotics SPARK product page: <http://www.revrobotics.com/rev-11-1200/>

### 38.5.5 SPARK MAX Motor Controller



The SPARK MAX Motor Controller from REV Robotics is a variable speed motor controller for use in FRC. The SPARK MAX is capable of controlling either the traditional brushed DC motors commonly used in FRC or the new brushless REV Robotics NEO Brushless Motor. The SPARK MAX can be controlled over PWM, CAN or USB (for configuration/testing only). The controller has a data port for sensor input and is capable of closed loop control modes when controlled over CAN or USB. For more information see the REV Robotics SPARK MAX product page: <http://www.revrobotics.com/rev-11-2158/>

### 38.5.6 Talon Motor Controller



The Talon Motor Controller from Cross the Road Electronics is a variable speed motor controller for use in FRC. The Talon is controlled over the PWM interface. The Talon should be connected to a PWM output of the roboRIO and powered from the Power Distribution Panel. For more information see the [Talon User Manual](#).

### 38.5.7 Talon SRX



The Talon SRX motor controller is a CAN-enabled “smart motor controller” from Cross The Road Electronics/VEX Robotics. The Talon SRX has an electrically isolated metal housing for heat dissipation, making the use of a fan optional. The Talon SRX can be controlled over the CAN bus or PWM interface. When using the CAN bus control, this device can take inputs from limit switches and potentiometers, encoders, or similar sensors in order to perform advanced control such as limiting or PID(F) closed loop control on the device. For more information see the [Talon SRX User Manual](#).

**Note: CAN Talon SRX has been removed from WPILib. See [this blog](#) for more info and find the CTRE Toolsuite installer here: [http://www.ctr-electronics.com/control-system/hro.html#product\\_tabs\\_technical\\_resources](http://www.ctr-electronics.com/control-system/hro.html#product_tabs_technical_resources)**

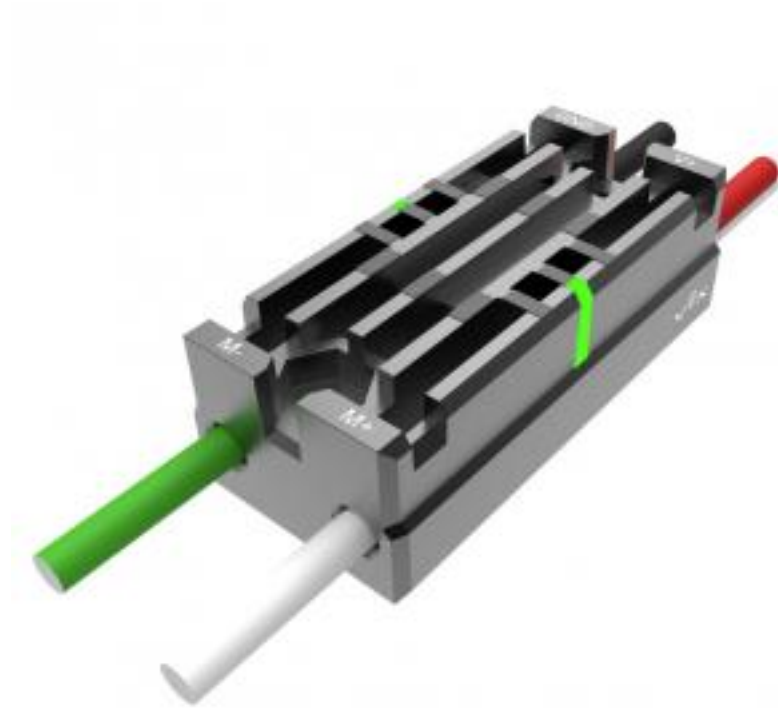


### 38.5.8 Victor 888 Motor Controller / Victor 884 Motor Controller



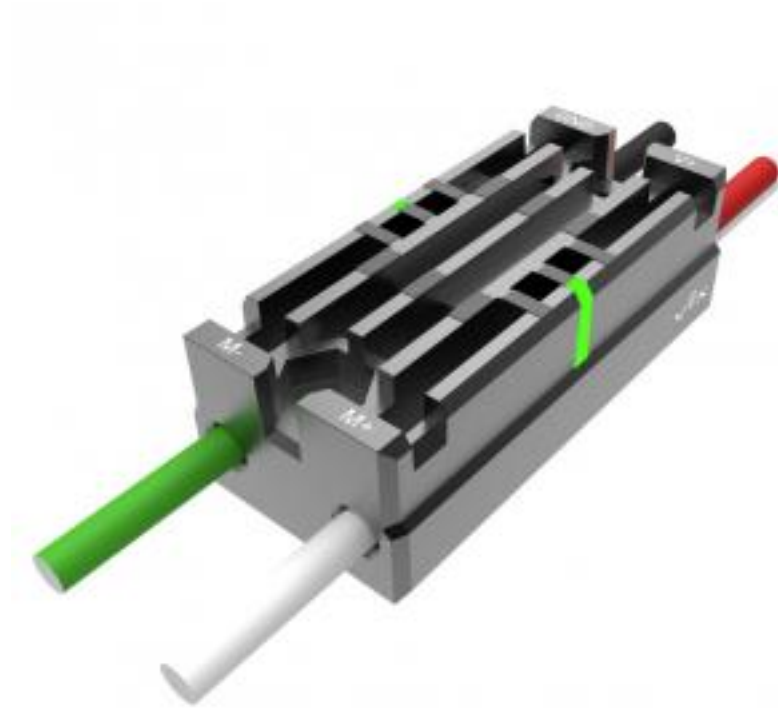
The Victor 888 Motor Controller from VEX Robotics is a variable speed motor controller for use in FRC. The Victor 888 replaces the Victor 884, which is also usable in FRC. The Victor is controlled over the PWM interface. The Victor should be connected to a PWM output of the roboRIO and powered from the Power Distribution Panel. For more information, see the [Victor 884 User Manual](#) and [Victor 888 User Manual](#).

### 38.5.9 Victor SP



The Victor SP motor controller is a PWM motor controller from Cross The Road Electronics/VEX Robotics. The Victor SP has an electrically isolated metal housing for heat dissipation, making the use of the fan optional. The case is sealed to prevent debris from entering the controller. The controller is approximately half the size of previous models. For more information, see the [Victor SP User Manual](#).

### 38.5.10 Victor SPX



The Victor SPX motor controller is a CAN or PWM controlled motor controller from Cross The Road Electronics/VEX Robotics. The device is connectorized to allow easy connection to the roboRIO PWM connectors or a CAN bus chain. When controlled over the CAN bus, the device has a number of the closed loop features also present in the Talon SRX. The case is sealed to prevent debris from entering the controller. For more information, see the [Victor SPX Webpage](#).

**Note: Victor SPX CAN control is not supported from WPILib. See [this blog](#) for more info and find the CTRE Toolsuite installer here: [http://www.ctr-electronics.com/control-system/hro.html#product\\_tabs\\_technical\\_resources](http://www.ctr-electronics.com/control-system/hro.html#product_tabs_technical_resources)**

## 38.6 Spike H-Bridge Relay



The Spike H-Bridge Relay from VEX Robotics is a device used for controlling power to motors or other custom robot electronics. When connected to a motor, the Spike provides On/Off control in both the forward and reverse directions. The Spike outputs are independently controlled so it can also be used to provide power to up to 2 custom electronic circuits. The Spike H-Bridge Relay should be connected to a relay output of the roboRIO and powered from the Power Distribution Panel. For more information, see the [Spike User's Guide](#).

## 38.7 Servo Power Module



The Servo Power Module from Rev Robotics is capable of expanding the power available to servos beyond what the roboRIO integrated power supply is capable of. The Servo Power Module provides up to 90W of 6V power across 6 channels. All control signals are passed through directly from the roboRIO. For more information, see the [Servo Power Module webpage](#).

## 38.8 Axis M1013/M1011/206 Ethernet Camera



The Axis M1013, M1011 and Axis 206 Ethernet cameras are used for capturing images/control-system-hardware for vision processing and/or sending video back to the Driver Station laptop. The camera should be wired to a 5V power output on the Voltage Regulator Module and an open ethernet port on the robot radio. For more information, see [Configuring an Axis Camera](#) and the [Axis 206](#), [Axis M1011](#), [Axis M1013](#) pages.

## 38.9 Microsoft Lifecam HD3000



The Microsoft Lifecam HD3000 is a USB webcam that can be plugged directly into the roboRIO. The camera is capable of capturing up to 1280x720 video at 30 FPS. For more information about the camera, see the [Microsoft product page](#). For more information about using the camera with the roboRIO, see the Vision Processing section of this documentation.

## 38.10 OpenMesh OM5P-AN or OM5P-AC Radio



Either the OpenMesh OM5P-AN or OpenMesh OM5P-AC wireless radio is used as the robot radio to provide wireless

communication functionality to the robot. The device can be configured as an Access Point for direct connection of a laptop for use at home. It can also be configured as a bridge for use on the field. The robot radio should be powered by one of the 12V/2A outputs on the VRM and connected to the roboRIO controller over Ethernet. For more information, see [Programming your radio for home use](#) and the [Open Mesh OM5P-AC product page](#).

The OM5P-AN is [no longer available for purchase](#). The OM5P-AC is slightly heavier, has more cooling grates, and has a rough surface texture compared to the OM5P-AN.

### 38.11 120A Circuit Breaker



The 120A Main Circuit Breaker serves two roles on the robot: the main robot power switch and a protection device for downstream robot wiring and components. The 120A circuit breaker is wired to the positive terminals of the robot battery and Power Distribution boards. For more information, please see the [Cooper Bussmann 18X Series Datasheet](#) (PN: 185120F)

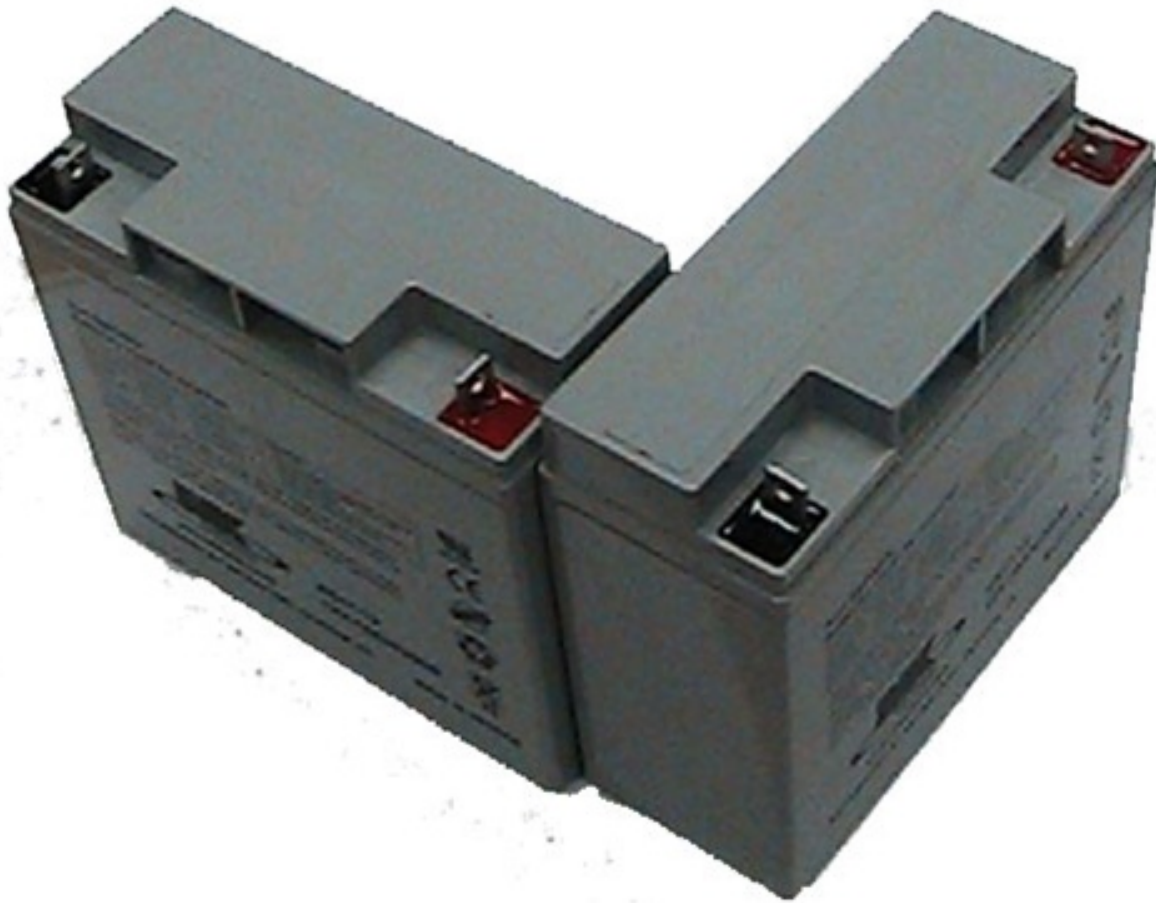


## 38.12 Snap Action Circuit Breakers



The Snap Action circuit breakers, MX5-A40 and VB3 series, are used with the Power Distribution Panel to limit current to branch circuits. The MX5-A40 40A MAXI style circuit breaker is used with the larger channels on the Power Distribution Panel to power loads which draw current up to 40A continuous. The VB3 series are used with the smaller channels on the PDP to power circuits drawing current of 30A or less continuous. For more information, see the Datasheets for the [MX5 series](#) and [VB3 Series](#).

## 38.13 Robot Battery



The power supply for an FRC robot is a single 12V 18Ah battery. The batteries used for FRC are sealed lead acid batteries capable of meeting the high current demands of an FRC robot. For more information, see the Datasheets for the [MK ES17-12](#) and [Energys NP18-12](#). Note that other battery part numbers may be legal, consult the FRC Manual for a complete list.

## 38.14 Image Credits

Image of roboRIO courtesy of National Instruments. Image of DMC-60 courtesy of Digilent. Image of SD540 courtesy of Mindsensors. Images of Jaguar Motor Controller, Talon SRX, Victor 888, Victor SP, Victor SPX, and Spike H-Bridge Relay courtesy of VEX Robotics, Inc. Image of SPARK MAX courtesy of REV Robotics. Lifecam, PDP, PCM, SPARK, and VRM photos courtesy of FIRST. All other photos courtesy of AndyMark Inc.

## CHAPTER 39

---

### How to wire an FRC Robot

---

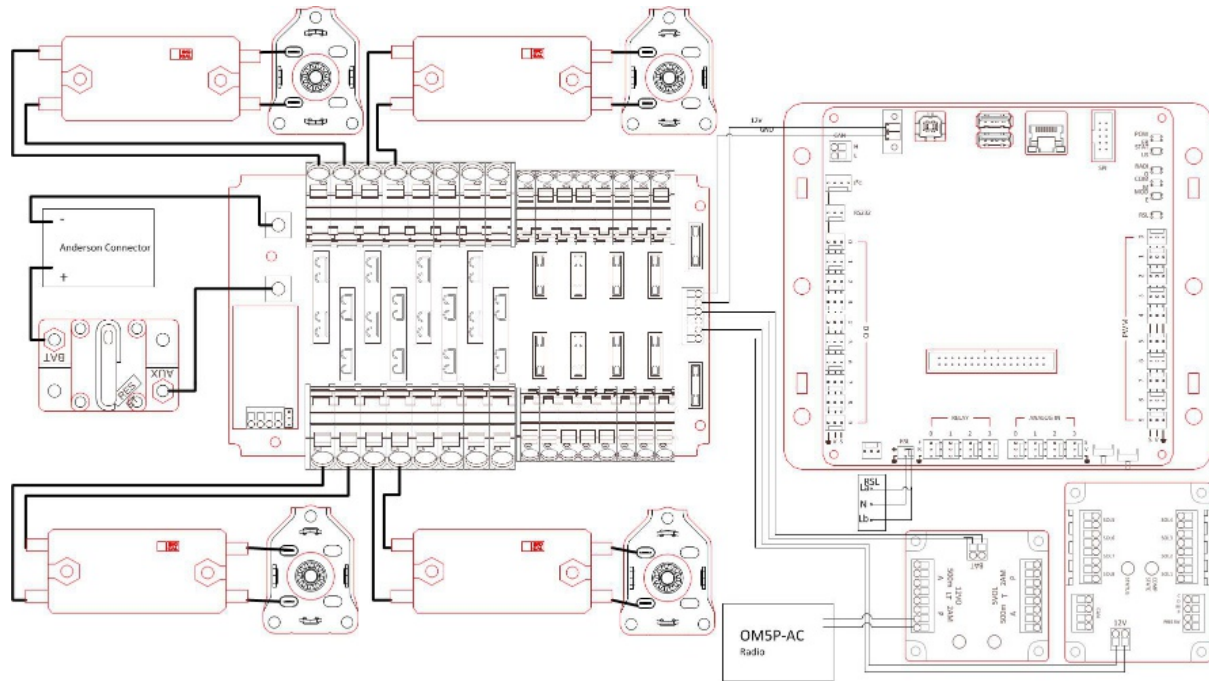
---

**Note:** This document details the wiring of a basic electronics board for bench-top testing.

Some images shown in this section reflect the setup for a Robot Control System using Spark motor controllers. Wiring diagram and layout should be similar for other motor controllers. Where appropriate, a second set of images shows the wiring steps for using PWM controllers with integrated wires.

---

## 39.1 Gather Materials



Locate the following control system components and tools

- **Kit Materials:**

- Power Distribution Panel (PDP)
- RoboRIO
- Pneumatics Control Module (PCM)
- Voltage Regulator Module (VRM)
- OpenMesh radio (with power cable and Ethernet cable)
- Robot Signal Light (RSL)
- 4x Victor SPX or other speed controllers
- 2x PWM y-cables
- 120A Circuit breaker
- 4x 40A Circuit breaker
- 6 AWG Red wire
- 10 AWG Red/Black wire
- 18 AWG Red/Black wire
- 22AWG yellow/green twisted CAN cable
- 16x 10-12 AWG (yellow) ring terminals (8x quick disconnect pairs if using integrated wire controllers)
- 2x Andersen SB50 battery connectors
- 6AWG Terminal lugs

- 12V Battery
- Red/Black Electrical tape
- Dual Lock material or fasteners
- Zip ties
- 1/4" or 1/2" plywood

- **Tools Required:**

- Wago Tool or small flat-head screwdriver
- Very small flat head screwdriver (eyeglass repair size)
- Philips head screw driver
- 5mm Hex key (3/16" may work if metric is unavailable)
- 1/16" Hex key
- Wire cutters, strippers, and crimpers
- 7/16" box end wrench or nut driver

#### Create the Base for the Control System

For a benchtop test board, cut piece of 1/4" or 1/2" material (wood or plastic) approximately 24" x 16". For a Robot Quick Build control board see the supporting documentation for the proper size board for the chosen chassis configuration.

## 39.2 Layout the Core Control System Components



Layout the components on the board. One layout that should work is shown in the images/how-to-wire-a-robot above.

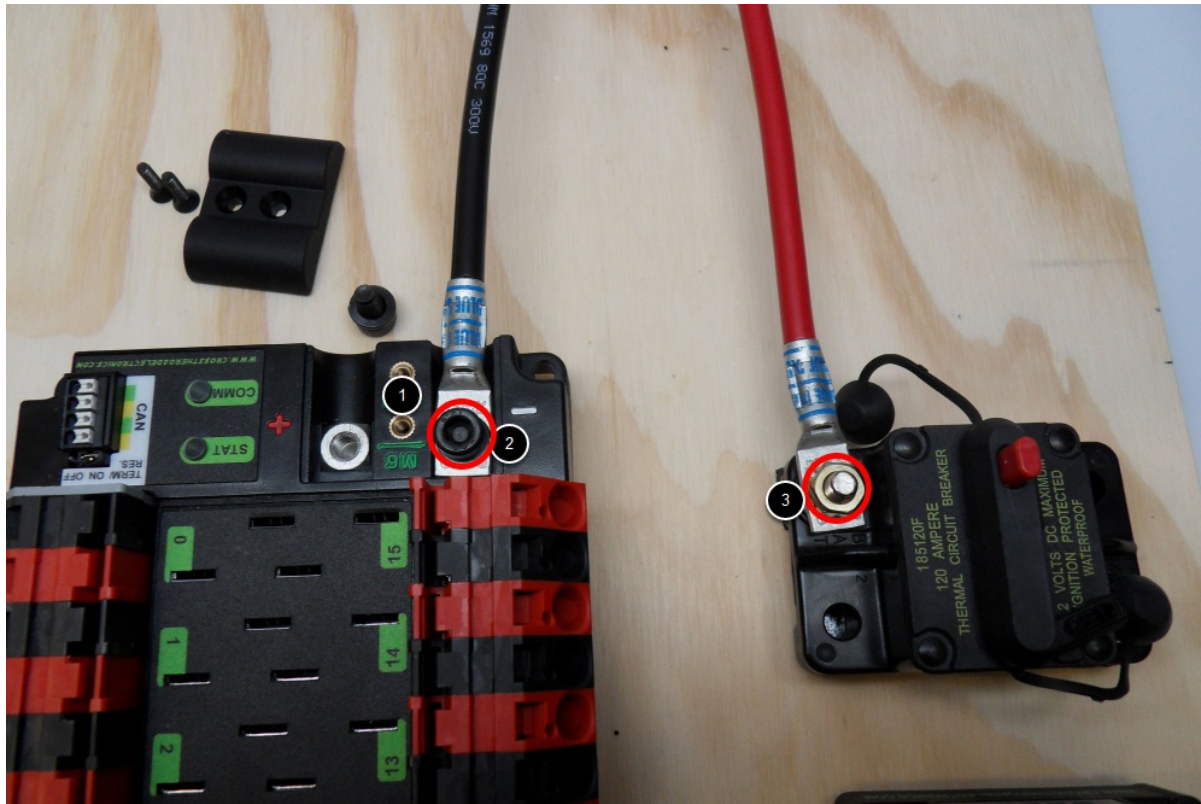


### 39.3 Fasten Components

Using the Dual Lock or hardware, fasten all components to the board. Note that in many FRC games robot-to-robot contact may be substantial and Dual Lock alone is unlikely to stand up as a fastener for many electronic components. Teams may wish to use nut and bolt fasteners or (as shown in the image above) cable ties, with or without Dual Lock to secure devices to the board.



## 39.4 Attach Battery Connector to PDP



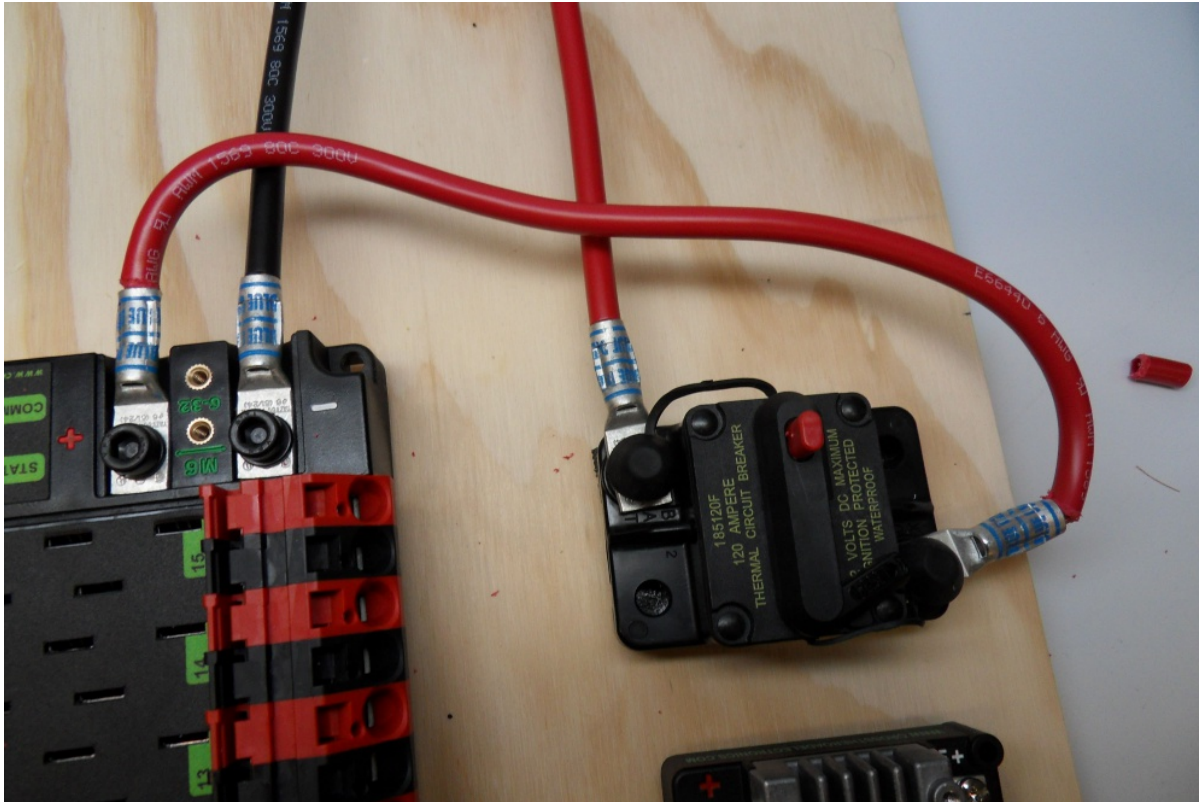
Requires: Battery Connector, 6AWG terminal lugs, 1/16" Allen, 5mm Allen, 7/16" Box end

Attach terminal lugs to battery connector:

1. Using a 1/16" Allen wrench, remove the two screws securing the PDP terminal cover.
2. Using a 5mm Allen wrench (3/16" will work if metric is not available), remove the negative (-) bolt and washer from the PDP and fasten the negative terminal of the battery connector.
3. Using a 7/16" box end wrench, remove the nut on the "Batt" side of the main breaker and secure the positive terminal of the battery connector.



## 39.5 Wire Breaker to PDP

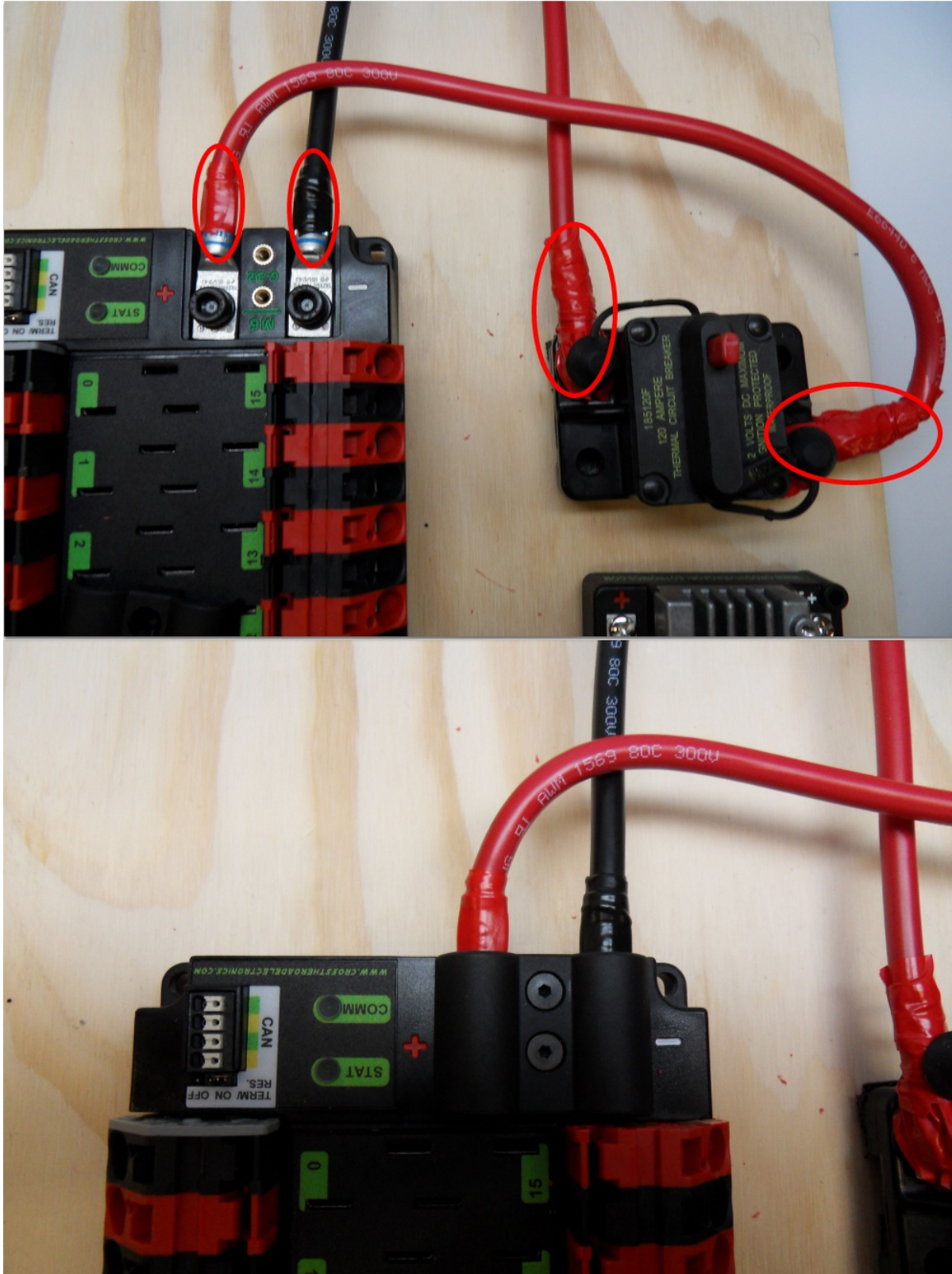


Requires: 6AWG red wire, 2x 6AWG terminal lugs, 5mm Allen, 7/16" box end

Secure one terminal lug to the end of the 6AWG red wire. Using the 7/16" box end, remove the nut from the "AUX" side of the 120A main breaker and place the terminal over the stud. Loosely secure the nut (you may wish to remove it shortly to cut, strip, and crimp the other end of the wire). Measure out the length of wire required to reach the positive terminal of the PDP.

1. Cut, strip, and crimp the terminal to the 2nd end of the red 6AWG wire.
2. Using the 7/16" box end, secure the wire to the "AUX" side of the 120A main breaker.
3. Using the 5mm, secure the other end to the PDP positive terminal.

## 39.6 Insulate PDP connections



Requires: 1/16" Allen, Electrical tape

1. **Using electrical tape, insulate the two connections to the 120A breaker.** Also insulate any part of the PDP terminals which will be exposed when the cover is replaced. One method for insulating the main breaker connections is to wrap the stud and nut first, then use the tape wrapped around the terminal and wire to secure the tape.
2. Using the 1/16" Allen wrench, replace the PDP terminal cover

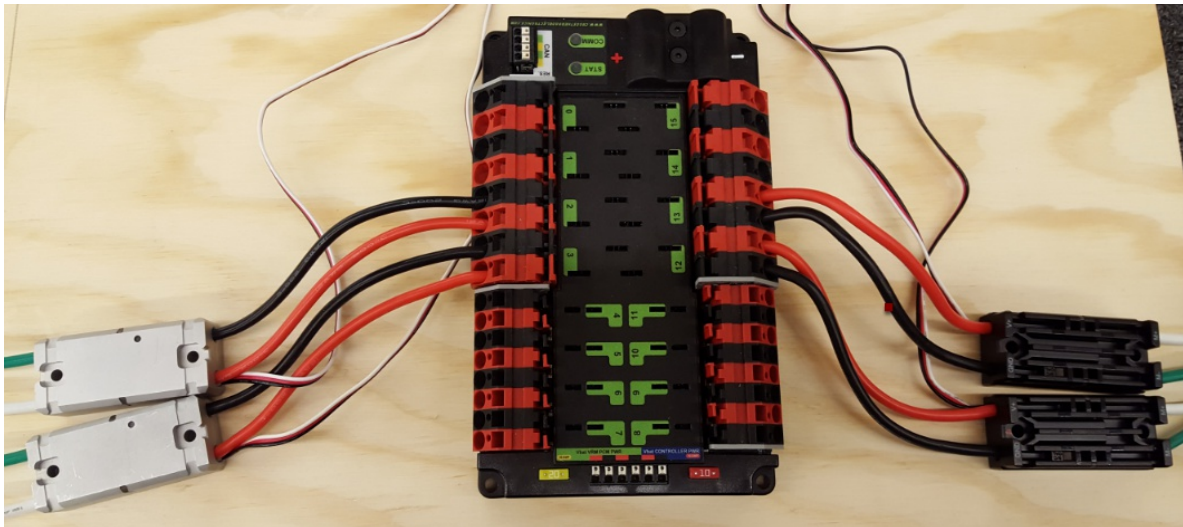
## 39.7 Wago connectors

The next step will involve using the Wago connectors on the PDP. To use the Wago connectors, insert a small flat blade screwdriver into the rectangular hole at a shallow angle then angle the screwdriver upwards as you continue to press in to actuate the lever, opening the terminal. Two sizes of Wago connector are found on the PDP:

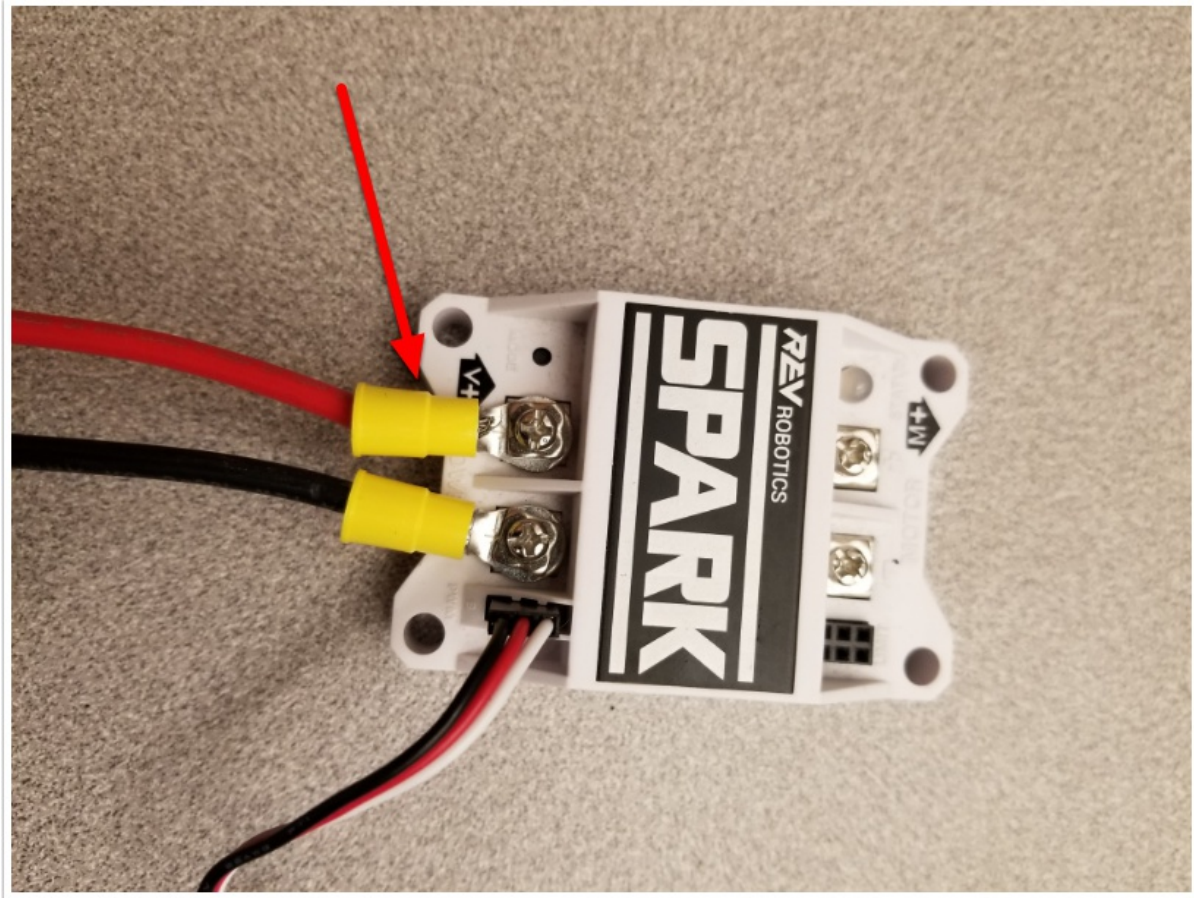
- Small Wago connector: Accepts 10AWG-24AWG, strip 11-12mm (~7/16")
- Large Wago connector: Accepts 6AWG-12AWG, strip 12-13mm(~1/2")

To maximize pullout force and minimize connection resistance wires should not be tinned (and ideally not twisted) before inserting into the Wago connector.

## 39.8 Motor Controller Power







Requires: Wire Stripper, Small Flat Screwdriver, 10 or 12 AWG wire, 10 or 12 AWG fork/ring terminals (terminal controllers only), wire crimper

For Victor SPX or other wire integrated motor controllers (top image):

- **Cut and strip the red and black power input wires**, then insert into one of the 40A (larger) Wago terminal pairs.

For terminal motor controllers (bottom image):

1. Cut red and black wire to appropriate length to reach from one of the 40A (larger) Wago terminal pairs to the input side of the speed controller (with a little extra for the length that will be inserted into the terminals on each end)
2. Strip one end of each of the wires, then insert into the Wago terminals.
3. Strip the other end of each wire, and crimp on a ring or fork terminal
4. Attach the terminal to the speed controller input terminals (red to +, black to -)

## 39.9 Weidmuller Connectors

The correct strip length is  $\sim 5/16$ " (8mm), not the  $5/8$ " mentioned in the video.

A number of the CAN and power connectors in the system use a Weidmuller LSF series wire-to-board connector. There are a few things to keep in mind when using this connector for best results:

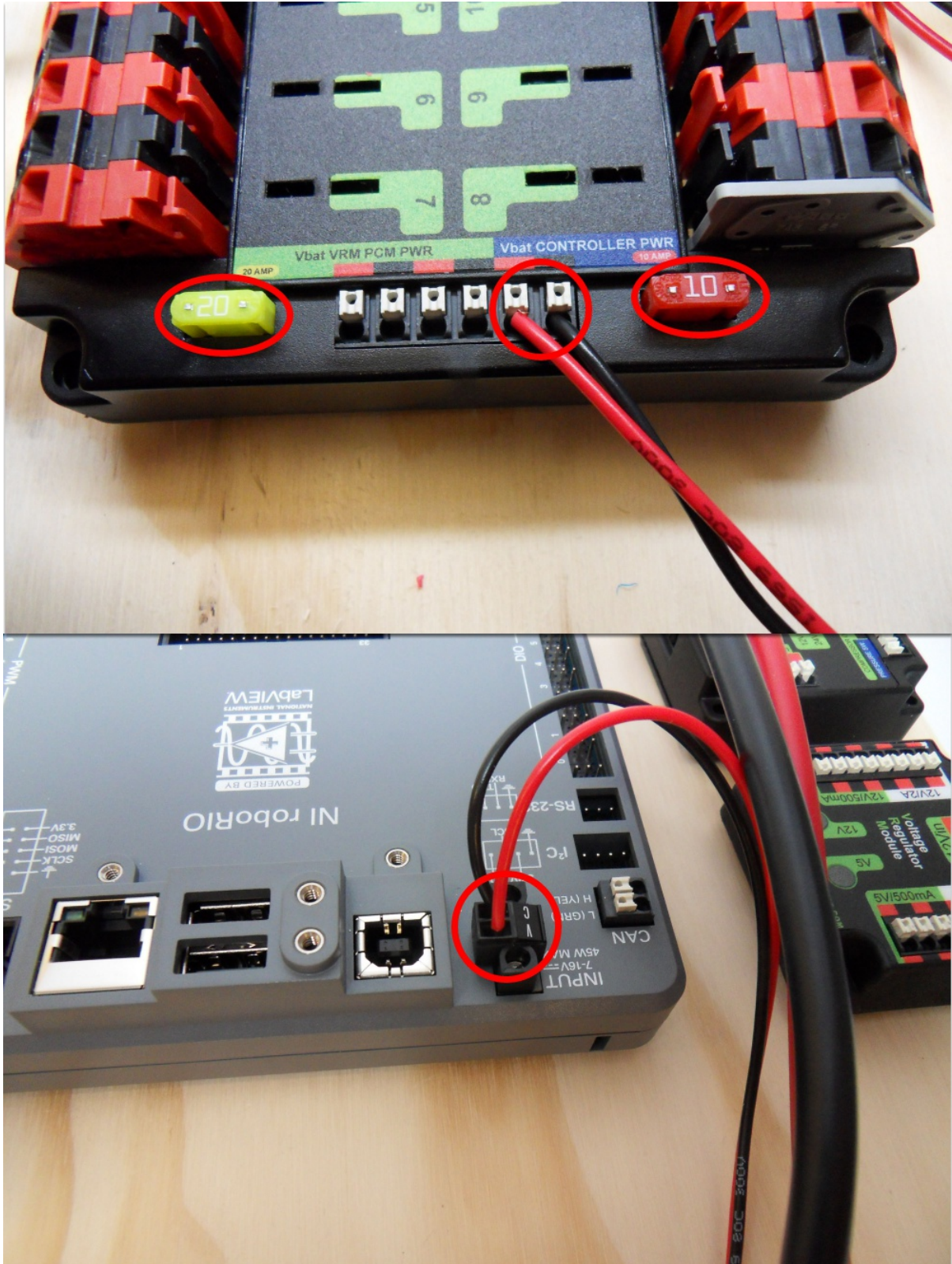
- **Wire should be 16AWG to 24AWG (consult rules to verify required gauge for power wiring)**

- Wire ends should be stripped approximately 5/16"
- **To insert or remove the wire, press down on the** corresponding "button" to open the terminal

After making the connection check to be sure that it is clean and secure:

- **Verify that there are no "whiskers" outside** the connector that may cause a short circuit
- **Tug on the wire to verify that it is seated fully.** If the wire comes out and is the correct gauge it needs to be inserted further and/or stripped back further.

### 39.10 roboRIO Power



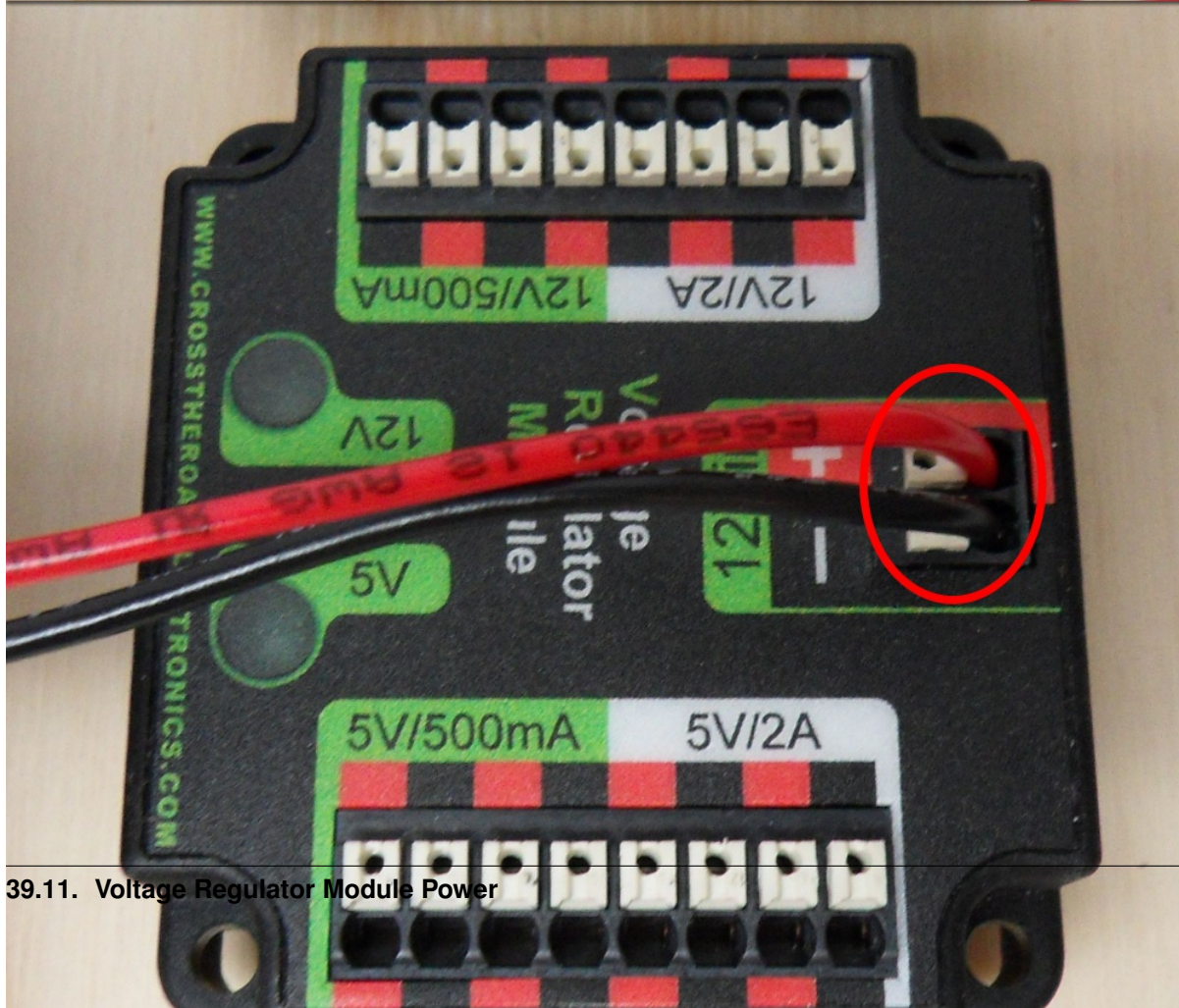
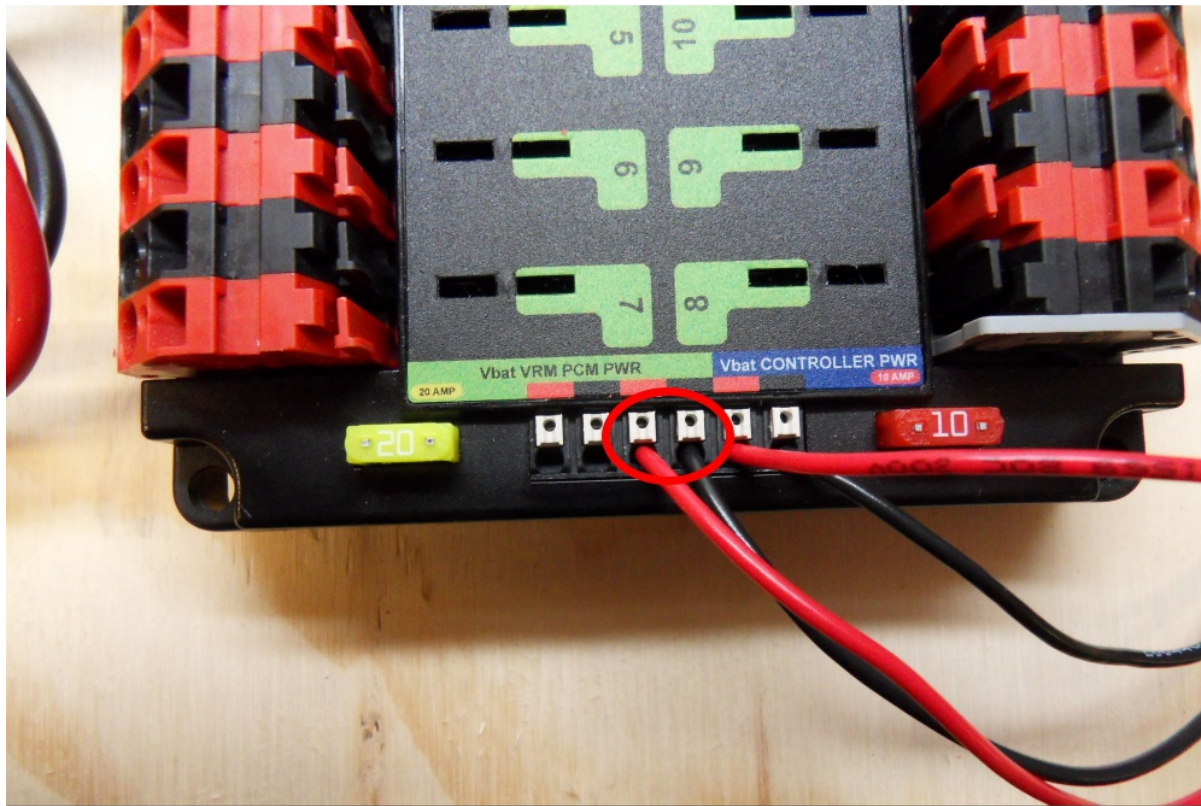
Requires: 10A/20A mini fuses, Wire stripper, very small flat screwdriver, 18AWG Red and Black

1. Insert the 10A and 20A mini fuses in the PDP in the locations shown on the silk screen (and in the image above)
2. Strip ~5/16" on both the red and black 18AWG wire and connect to the "Vbat Controller PWR" terminals on the PDB
3. Measure the required length to reach the power input on the roboRIO. Take care to leave enough length to route the wires around any other components such as the battery and to allow for any strain relief or cable management.
4. Cut and strip the wire.
5. Using a very small flat screwdriver connect the wires to the power input connector of the roboRIO (red to V, black to C). Also make sure that the power connector is screwed down securely to the roboRIO.





### 39.11 Voltage Regulator Module Power

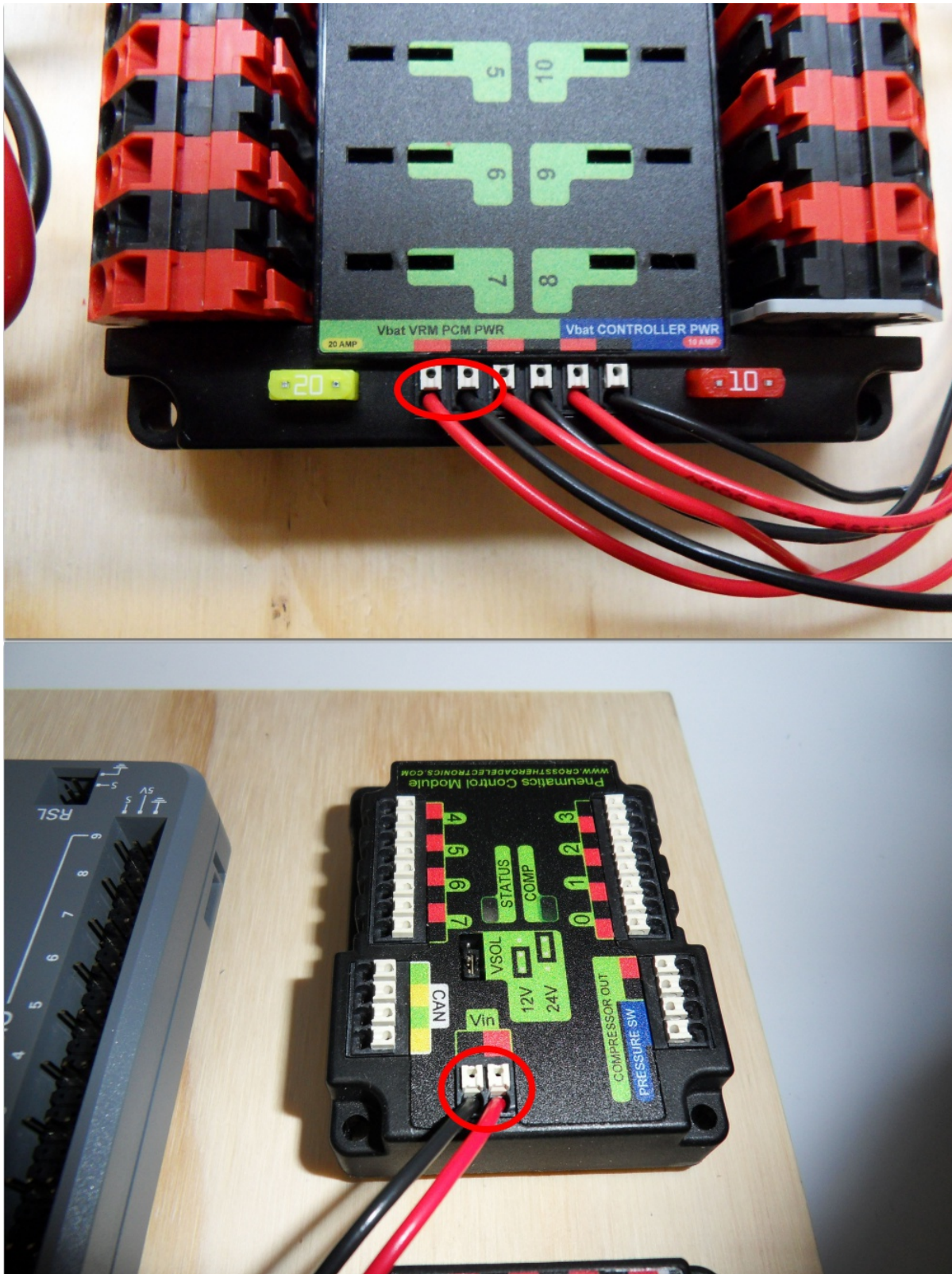


Requires: Wire stripper, small flat screwdriver (optional), 18AWG red and black wire:

1. Strip ~5/16" on the end of the red and black 18AWG wire.
2. Connect the wire to one of the two terminal pairs labeled "Vbat VRM PCM PWR" on the PDP.
3. Measure the length required to reach the "12Vin" terminals on the VRM. Take care to leave enough length to route the wires around any other components such as the battery and to allow for any strain relief or cable management.
4. Cut and strip ~5/16" from the end of the wire.
5. Connect the wire to the VRM 12Vin terminals.



## 39.12 Pneumatics Control Module Power (Optional)



Requires: Wire stripper, small flat screwdriver (optional), 18AWG red and black wire

Note: The PCM is an optional component used for controlling pneumatics on the robot.

1. Strip ~5/16" on the end of the red and black 18AWG wire.
2. Connect the wire to one of the two terminal pairs labeled "Vbat VRM PCM PWR" on the PDP.
3. Measure the length required to reach the "Vin" terminals on the PCM. Take care to leave enough length to route the wires around any other components such as the battery and to allow for any strain relief or cable management.
4. Cut and strip ~5/16" from the end of the wire.
5. Connect the wire to the PCM 12Vin terminals.

### 39.13 Radio Power and Ethernet

<p><b>Warning:</b> DO NOT connect the Rev passive POE injector cable directly to the roboRIO. The roboRIO MUST connect to the female end of the cable using an additional Ethernet cable as shown in the next step.</p>
---



Requires: Small flat screwdriver (optional), Rev radio PoE cable

1. Insert the ferrules of the passive PoE injector cable into the corresponding colored terminals on the 12V/2A

section of the VRM.

2. Connect the male RJ45 (Ethernet) end of the cable into the Ethernet port on the radio closest to the barrel connector (labeled 18-24v POE)



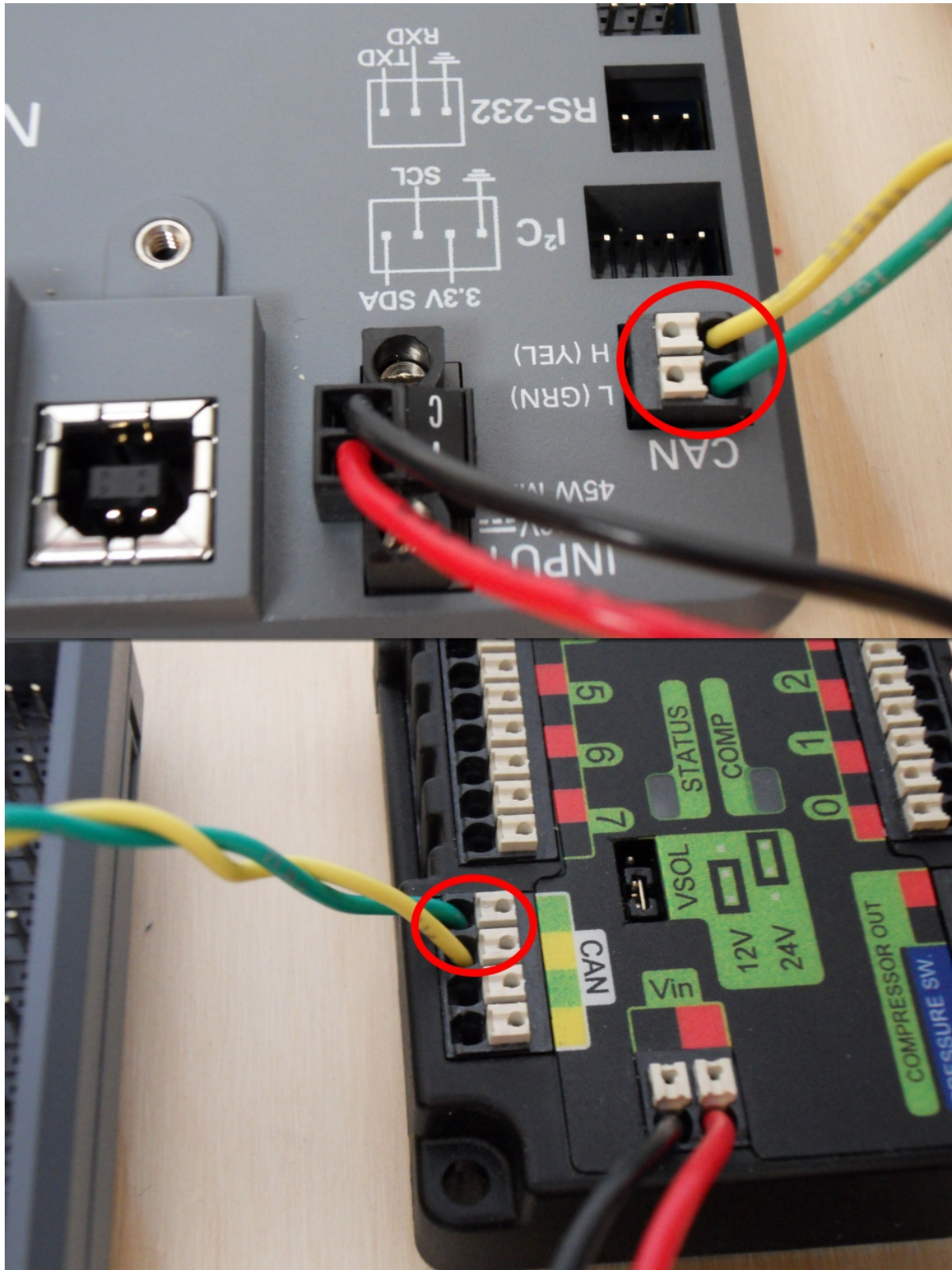
## 39.14 RoboRIO to Radio Ethernet



Requires: Ethernet cable

Connect an Ethernet cable from the female RJ45 (Ethernet) port of the Rev Passive POE cable to the RJ45 (Ethernet) port on the roboRIO. RoboRIO to PCM CAN RoboRIO to PCM CANZoom: RoboRIO to PCM CAN



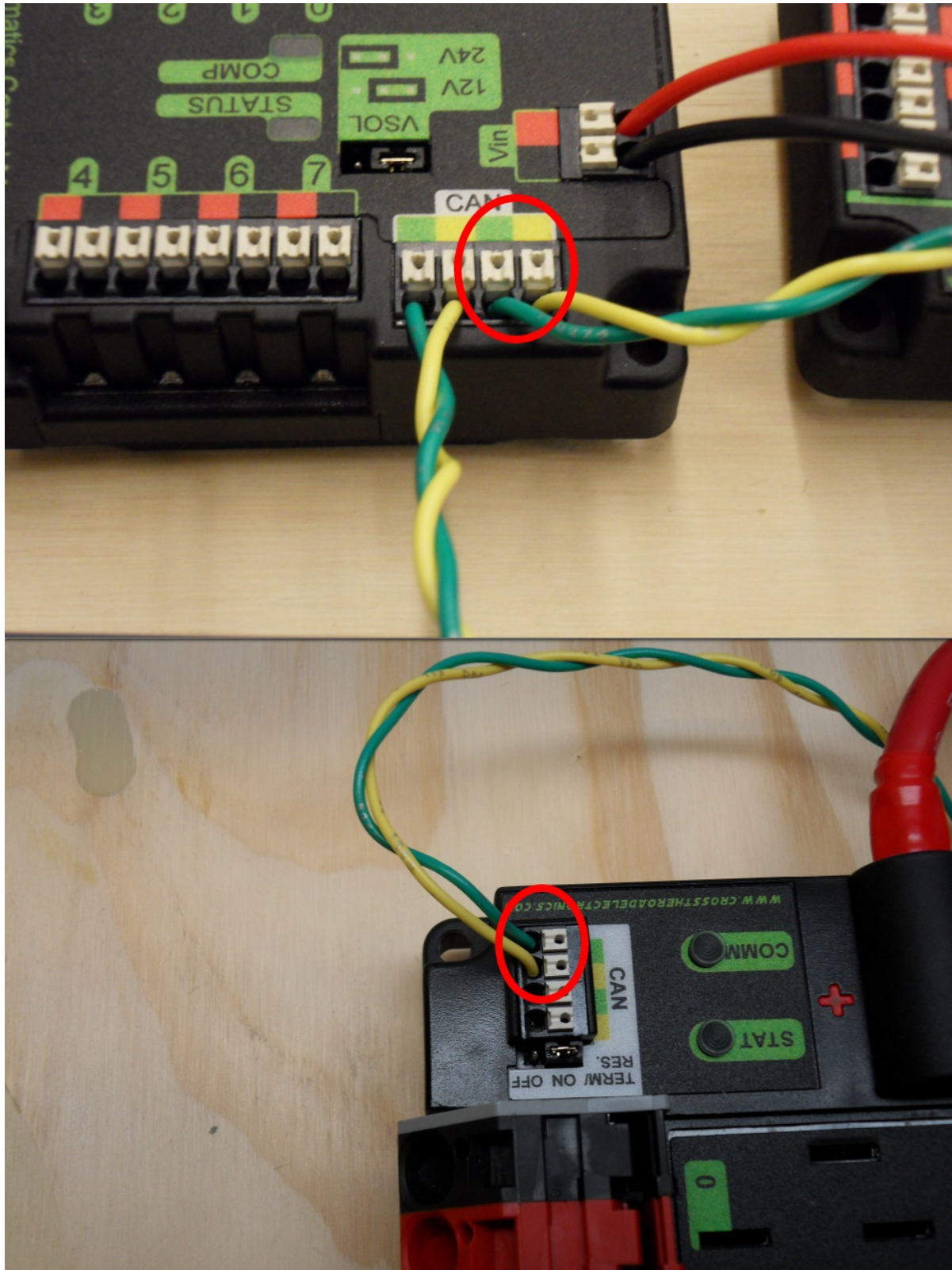


Requires: Wire stripper, small flat screwdriver (optional), yellow/green twisted CAN cable

Note: The PCM is an optional component used for controlling pneumatics on the robot. If you are not using the PCM, wire the CAN connection directly from the roboRIO (shown in this step) to the PDP (shown in the next step).

1. Strip ~5/16" off of each of the CAN wires.
2. Insert the wires into the appropriate CAN terminals on the roboRIO (Yellow->YEL, Green->GRN).
3. Measure the length required to reach the CAN terminals of the PCM (either of the two available pairs). Cut and strip ~5/16" off this end of the wires.
4. Insert the wires into the appropriate color coded CAN terminals on the PCM. You may use either of the Yellow/Green terminal pairs on the PCM, there is no defined in or out.

### 39.15 PCM to PDP CAN





Requires: Wire stripper, small flat screwdriver (optional), yellow/green twisted CAN cable

Note: The PCM is an optional component used for controlling pneumatics on the robot. If you are not using the PCM, wire the CAN connection directly from the roboRIO (shown in the above step) to the PDP (shown in this step).

1. Strip ~5/16" off of each of the CAN wires.
2. Insert the wires into the appropriate CAN terminals on the PCM.
3. Measure the length required to reach the CAN terminals of the PDP (either of the two available pairs). Cut and strip ~5/16" off this end of the wires.
4. Insert the wires into the appropriate color coded CAN terminals on the PDP. You may use either of the Yellow/Green terminal pairs on the PDP, there is no defined in or out.

Note: The PDP ships with the CAN bus terminating resistor jumper in the "ON" position. It is recommended to leave the jumper in this position and place any additional CAN nodes between the roboRIO and the PDP (leaving the PDP as the end of the bus). If you wish to place the PDP in the middle of the bus (utilizing both pairs of PDP CAN terminals) move the jumper to the "OFF" position and place your own 120 ohm terminating resistor at the end of your CAN bus chain.



## 39.16 PWM Cables



Requires: 4x PWM cables (if using non-integrated wire controllers), 2x PWM Y-cable (Optional)

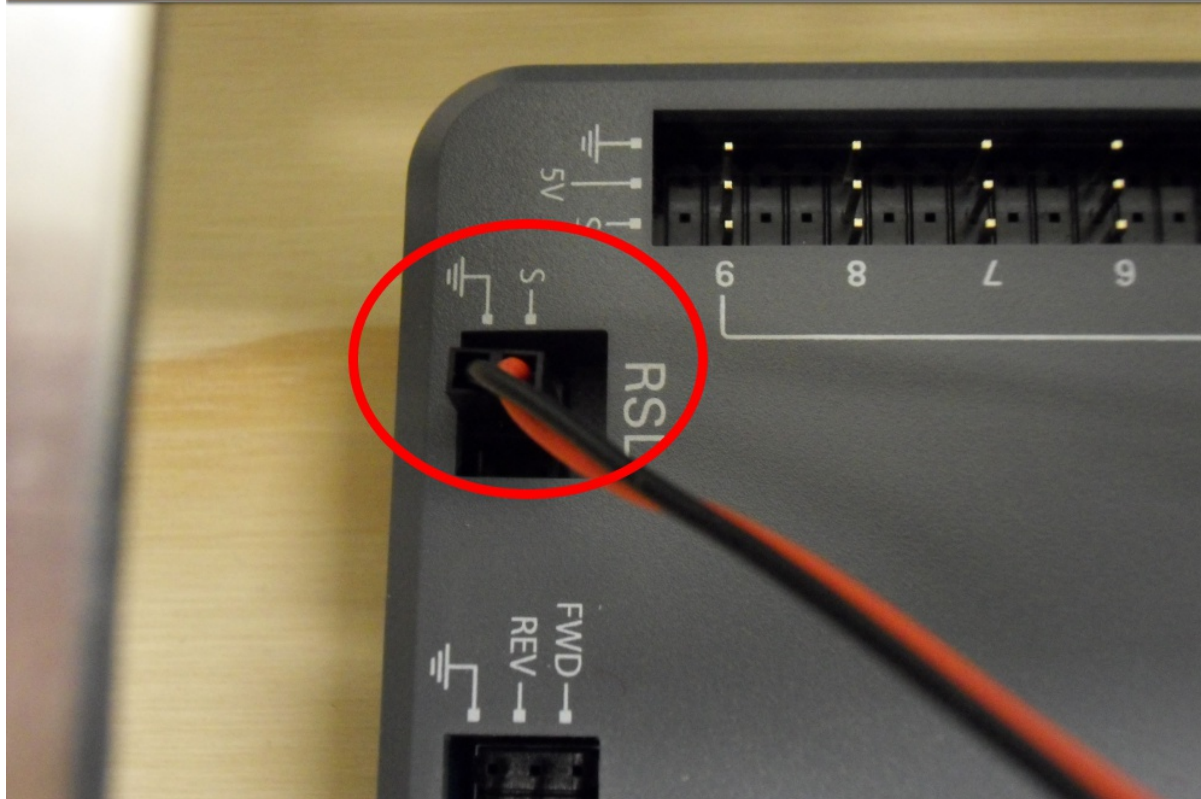
Option 1 (Direct connect):

- Connect the PWM cables from each controller directly to the roboRIO. For Victor SPX's and other PWM/CAN controllers, the green wire (black wire for non-integrated controllers) should be towards the outside of the roboRIO. For controllers without integrated wires, make sure the controller side of the black wire is located according to the markings on the controller. It is recommended to connect the left side to PWM 0 and 1 and the right side to PWM 2 and 3 for the most straightforward programming experience, but any channel will work as long as you note which side goes to which channel and adjust the code accordingly.

Option 2 (Y-cable):

1. Connect 1 PWM Y-cable to the PWM cables for the controllers controlling one side of the robot. The brown wire on the Y-cable should match the green/black wire on the PWM cable.
2. Connect the PWM Y-cables to the PWM ports on the roboRIO. The brown wire should be towards the outside of the roboRIO. It is recommended to connect the left side to PWM 0 and the right side to PWM 1 for the most straightforward programming experience, but any channel will work as long as you note which side goes to which channel and adjust the code accordingly.

### 39.17 Robot Signal Light



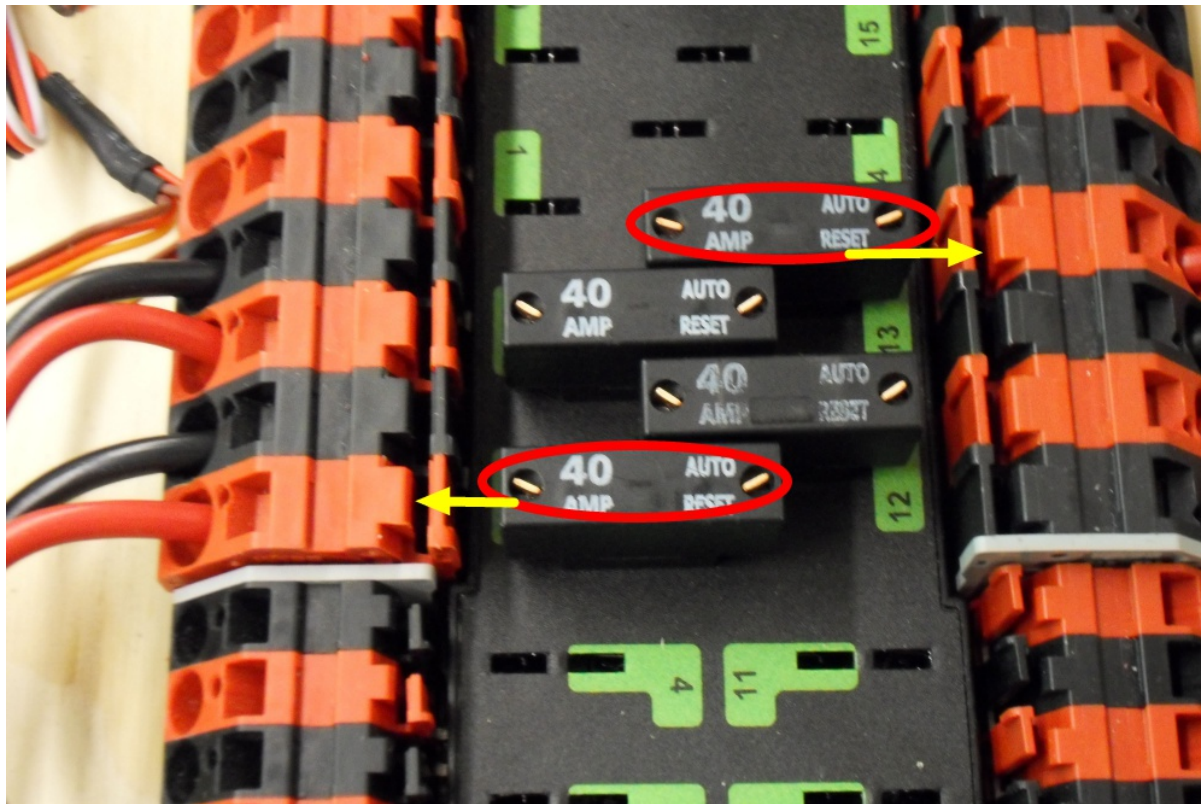


Requires: Wire stripper, 2 pin cable, Robot Signal Light, 18AWG red wire, very small flat screwdriver

1. Cut one end off of the 2 pin cable and strip both wires
2. Insert the black wire into the center, “N” terminal and tighten the terminal.
3. Strip the 18AWG red wire and insert into the “La” terminal and tighten the terminal.
4. Cut and strip the other end of the 18AWG wire to insert into the “Lb” terminal
5. Insert the red wire from the two pin cable into the “Lb” terminal with the 18AWG red wire and tighten the terminal.
6. Connect the two-pin connector to the RSL port on the roboRIO. The black wire should be closest to the outside of the roboRIO.

You may wish to temporarily secure the RSL to the control board using zipties or Dual Lock (it is recommended to move the RSL to a more visible location as the robot is being constructed) Circuit Breakers

### 39.18 Circuit Breakers

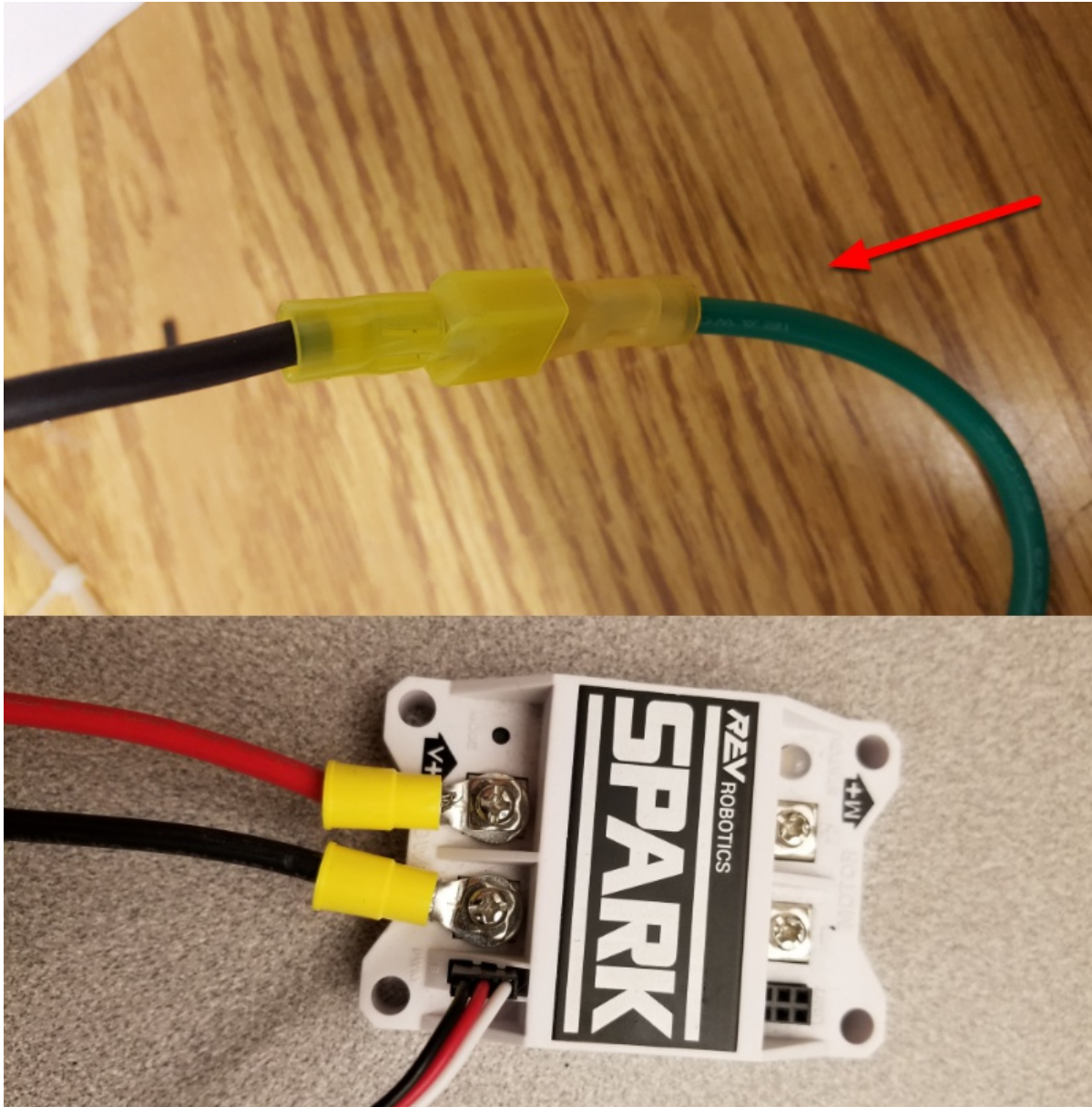


Requires: 4x 40A circuit breakers

Insert 40-amp Circuit Breakers into the positions on the PDP corresponding with the Wago connectors the Talons are connected to. Note that, for all breakers, the breaker corresponds with the nearest positive (red) terminal (see graphic above). All negative terminals on the board are directly connected internally.

If working on a Robot Quick Build, stop here and insert the board into the robot chassis before continuing.

## 39.19 Motor Power



Requires: Wire stripper, wire crimper, phillips head screwdriver, wire connecting hardware

For each CIM motor:

- Strip the ends of the red and black wires from the CIM

For integrated wire controllers (including Victor SPX):

1. Strip the white and green wires from the controller
2. Connect the motor wires to the controller output wires (it is recommended to connect the red wire to the white M+ output). The images/how-to-wire-a-robot above show examples using quick disconnect terminals.

For Sparks or other non-integrated-wire controllers:

1. Crimp a ring/fork terminal on each of the motor wires.

2. Attach the wires to the output side of the motor controller (red to +, black to -)

## 39.20 STOP



**Danger:** STOP!!

**Danger:** Before plugging in the battery, make sure all connections have been made with the proper polarity. Ideally have someone that did not wire the robot check to make sure all connections are correct.

Before plugging in the battery, make sure all connections have been made with the proper polarity. Ideally have

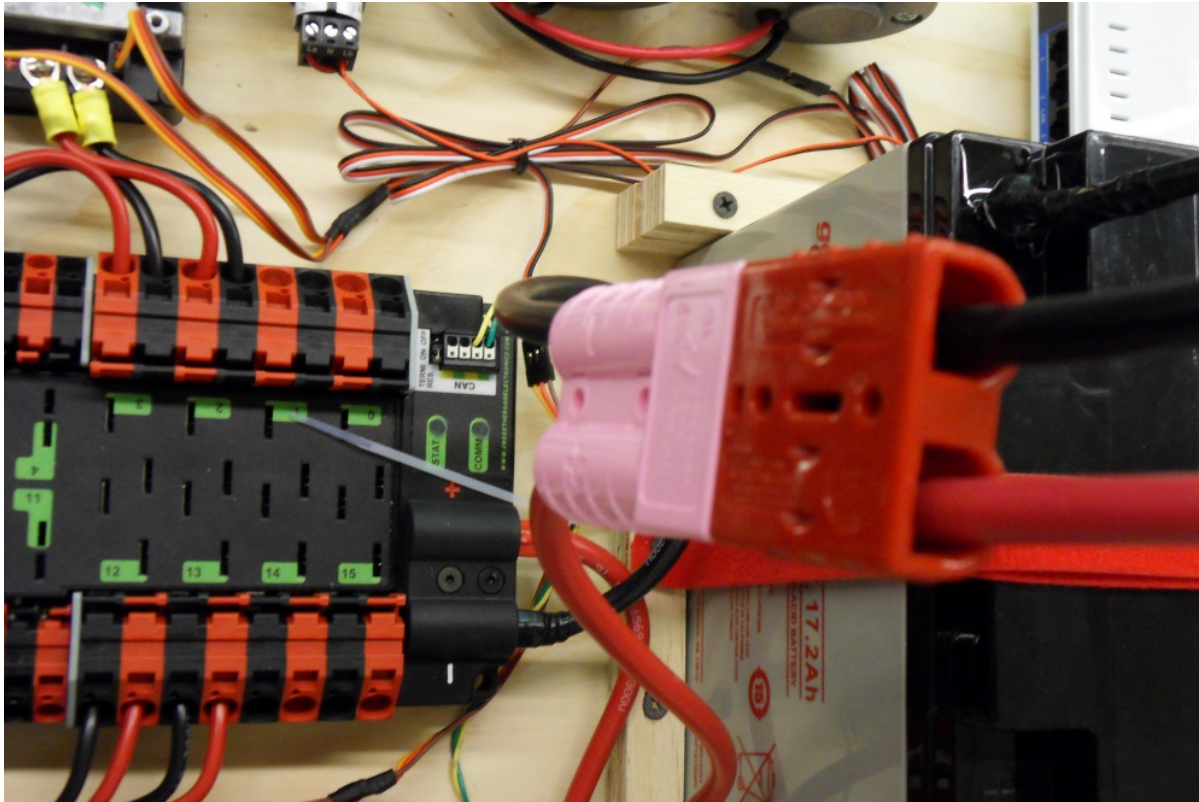


someone that did not wire the robot check to make sure all connections are correct.

- Start with the battery and verify that the red wire is connected to the positive terminal
- Check that the red wire passes through the main breaker and to the + terminal of the PDP and that the black wire travels directly to the - terminal.
- For each motor controller, verify that the red wire goes from the red PDP terminal to the Talon input labeled with the red + (not the white M+!!!!)
- For each device on the end of the PDP, verify that the red wire connects to the red terminal on the PDP and the red terminal on the component.
- Make sure that the orange Passive PoE cable is plugged directly into the radio NOT THE roboRIO! It must be connected to the roboRIO using an additional Ethernet cable.

It is also recommended to put the robot on blocks so the wheels are off the ground before proceeding. This will prevent any unexpected movement from becoming dangerous.

### 39.21 Manage Wires



Requires: Zip ties

Now may be a good time to add a few zip ties to manage some of the wires before proceeding. This will help keep the robot wiring neat. Connect Battery Connect BatteryZoom: Connect Battery

Connect the battery to the robot side of the Andersen connector. Power on the robot by moving the lever on the top of the 120A main breaker into the ridge on the top of the housing. If stuff blinks, you probably did it right. From here, you should connect to the RoboRIO and try uploading your code!

---

## Wiring Best Practices

---

---

**Hint:** The article Wiring the FRC Control System walks through the details of what connects where to wire up the FRC Control System, this article provides some additional “Best Practices” that may increase reliability and make maintenance easier.

---

### 40.1 Vibration/Shock

An FRC Robot is an incredibly rough environment when it comes to vibrations and shock loads. While many of the FRC specific electronics are extensively tested for mechanical robustness in these conditions, a few components, such as the radio, are not specifically designed for use on a mobile platform. Taking steps to reduce the shock and vibration these components are exposed to may help reduce failures. Some suggestions that may reduce mechanical failures:

- **Vibration Isolation** - Make sure to isolate any components which create excessive vibration, such as compressors, using “vibration isolators”. This will help reduce vibration on the robot which can loosen fasteners and cause premature fatigue failure on some electronic components.
- **Bumpers** - Use Bumpers to cover as much of the robot as possible for your design. While the rules require specific bumper coverage around the corners of your robot, maximizing the use of bumpers increases the likelihood that all collisions will be damped by your bumpers. Bumpers significantly reduce the g-forces experienced in a collision compared to hitting directly on a hard robot surface, reducing the shock experienced by the electronics and decreasing the chance of a shock related failure.
- **Shock Mounting** - You may choose to shock mount some or all of your electronic components to further reduce the forces they see in robot collisions. This is especially helpful for the robot radio and other electronics such as co-processors, which may not be designed for use on mobile platforms. Vibration isolators, springs, foams, or mounting to flexible materials all may reduce the shock forces seen by these components.

## 40.2 Redundancy

Unfortunately there are few places in the FRC Control System where redundancy is feasible. Taking advantage of opportunities for redundancy can increase reliability. The primary example of this is wiring the barrel connector to the radio in addition to the provided PoE connection. This ensures that if one of the cables becomes damaged or dislodged, the other will maintain power to the radio. Keep an eye out for other potential areas to provide redundancy when wiring and programming your robot.

## 40.3 Port Savers

For any connections on the Robot or Driver station that may be frequently plugged and unplugged (such as DS joysticks, DS Ethernet, roboRIO USB tether, and Ethernet tether) using a “Port Saver” or “pigtail” can substantially reduce the potential for damaging the port. This type of device can serve double duty, both reducing the number of cycles that the port on the electronic device sees, as well as relocating the connection to a more convenient location. Make sure to secure the port saver (see the next item) to avoid port damage.

## 40.4 Wire Management and Strain Relief

One of the most critical components to robot reliability and maintenance is good wire management and strain relief. Good wire management is comprised of a few components:

- Make sure cables are the correct length. Any excess wire length is just more to manage. If you must have extra wire due to additional length on COTS cabling, secure the extra into a small bundle using separate cable ties before securing the rest of the wire.
- Ensure that cables are secured close to connection points, with enough slack to avoid putting strain on connectors. This is called strain relief, and is critical to minimizing the likelihood that a cable comes unplugged or a wire breaks off at a connection point (these are generally stress concentrators).
- Secure cables near any moving components. Make sure that all wire runs are secure and protected from moving components, even if the moving components were to bend or over-travel.
- Secure cables at additional points as necessary to keep wiring neat and clean. Take care to not over secure wires; if wires are secured in too many locations, it may actually make troubleshooting and maintenance more difficult.

## 40.5 Documentation

A great way to make maintenance easier is to create documentation describing what is connected where on the robot. There are a number of ways of creating this type of documentation which range from complete wiring diagrams to excel charts to a quick list of what functions are attached to which channels. Many teams also integrate these lists with labeling (see the next bullet).

When a wire is accidentally cut, or a mechanism is malfunctioning, or a component burns out, it will be much easier to repair if you have some documentation to tell you what is connected where without having to trace the wiring all the way through (even if your wiring is neat!)

## 40.6 Labeling

Labeling is a great way to supplement the wiring documentation described above. There are many different strategies to labeling wiring and electronics, all with their own pros and cons. Labels for electronics and flags for wires can be made by hand, or using a label maker (some can also do heatshrink labels), or you can use different colors of electrical tape or labeling flags to indicate different things. Whatever system you choose, make sure you understand how it complements your documentation and make sure everyone on your team is familiar with it.

## 40.7 Check all wiring and connections

After all wiring on the robot is complete, make sure to check each connection, pulling on each, to ensure that everything is secure. Additionally, ensure that no stray wire “whiskers” are sticking out of any connection point and that no uninsulated connections are exposed. If any connections come loose while testing, or any “whiskers” are discovered, re-make the connection and make sure to have a second person check it when complete.

A common source of poor connections is screw-type or nut-and-bolt fasteners. For any connections of this type on the robot (e.g. battery connections, main breaker, PDP, roboRIO), make sure the fasteners are tight. For nut-and-bolt style connections, ensure that the wire/terminal cannot be rotated by hand; if you can rotate your battery wire or main breaker connection by grasping the terminal and twisting, the connection is not tight enough.

Another common source of failures is the fuses at the end of the PDP. Ensure these fuses are completely seated; you may need to apply more force than you expect to seat them completely. If the fuses are seated properly they will likely be difficult or impossible to remove by hand.

Snap-in connections such as the SB-50 connector should be secured using clips or cable ties to ensure they do not pop loose during impacts.

## 40.8 Re-Check Early and Often

Re-check the entire electrical system as thoroughly as possible after playing the first match or two (or doing very vigorous testing). The first few impacts the robot sees may loosen fasteners or expose issues.

Create a checklist for re-checking electrical connections on a regular basis. As a very rough starting point, rotational fasteners such as battery and PDP connections should be checked every 1-3 matches. Spring type connections such as the Wago and Weidmuller connectors likely only need to be checked once per event. Ensure that the team knows who is responsible for completing the checklist and how they will document that it has been done.

## 40.9 Battery Maintenance

Take good care of your batteries! A bad battery can easily cause a robot to function poorly, or not at all, during a match. Label all of your batteries to help keep track of usage during the event. Many teams also include information such as the age of the battery on this label.

- **Never lift or carry batteries by the wires! Carrying batteries by the wires** has the potential to damage the internal connection between the terminals and the plates, dramatically increasing internal resistance and degrading performance.
- **Mark any dropped battery bad until a complete test can be conducted.** In addition to the mentioned terminal connections, dropping a battery also has the potential to damage individual cells. This damage may not register on a simple voltage test, instead hiding until the battery is placed under load.

- **Rotate batteries evenly. This helps ensure that batteries have the most time to** charge and rest and that they wear evenly (equal number of charge/discharge cycles)
- **Load test batteries if possible to monitor health. There are a number of** commercially available products teams use to load test batteries, including at least one designed specifically for FRC. A load test can provide an indicator of battery health by measuring internal resistance. This measurement is much more meaningful when it comes to match performance than a simple no-load voltage number provided by a multimeter.

### 40.10 Check DS Logs

After each match, review the DS logs to see what the battery voltage and current usage looks like. Once you have established what the normal range of these items is for your robot, you may be able to spot potential issues (bad batteries, failing motors, mechanical binding) before they become critical failures.



---

**Hint:** Wiring pneumatics has been made very simple in the 2019 Control System. A single Pneumatics Control Module is all that will be needed for many pneumatics applications, with additional PCMs supporting more complex designs including more than 8 solenoid channels or a mix of 12V and 24V solenoids.

---

### 41.1 Wiring Overview

A single PCM will support many pneumatics applications, providing an output for the compressor, input for the pressure switch and outputs for up to 8 solenoid channels (12V or 24V selectable). The module is connected to the roboRIO over the CAN bus and powered via 12V from the PDP.

### 41.2 PCM Power and Control Wiring

The first PCM on your robot can be wired from the PDP VRM/PCM connectors on the end of the PDP. The PCM is connected to the roboRIO via CAN and can be placed anywhere in the middle of the CAN chain (or on the end with a custom terminator). For more details on wiring a single PCM see *Wiring the 2015 FRC Control System*. Additional PCMs can be wired to a standard Wago connector on the side of the PDP and protected with a 20A or smaller circuit breaker. Additional PCMs should also be placed anywhere in the middle of the CAN chain.

### 41.3 The Compressor

The compressor can be wired directly to the Compressor Out connectors on the PCM. If additional length is required, make sure to use 18 AWG wire or larger for the extension.

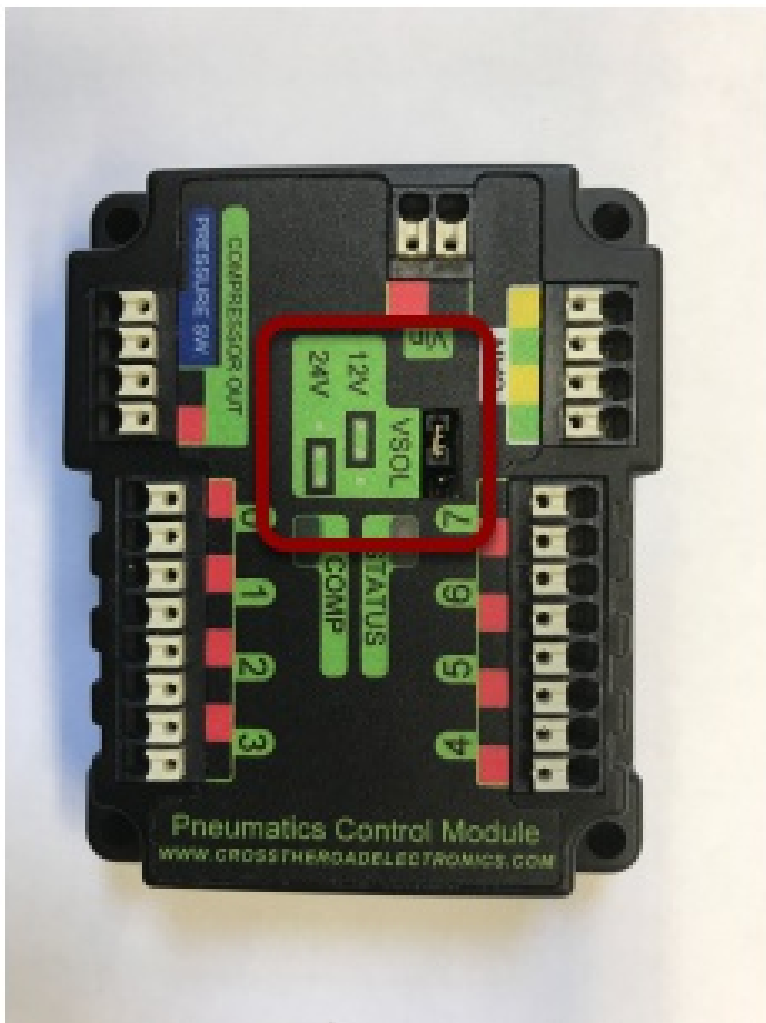
## 41.4 The Pressure Switch

The pressure switch should be connected directly to the pressure switch input terminals on the PCM. There is no polarity on the input terminals or on the pressure switch itself, either terminal on the PCM can be connected to either terminal on the switch. Ring or spade terminals are recommended for the connection to the switch screws (note that the screws are slightly larger than #6, but can be threaded through a ring terminal with a hole for a #6 screw such as the terminals shown in the image).

## 41.5 Solenoids

Each solenoid channel should be wired directly to a numbered pair of terminals on the PCM. A single acting solenoid will use one numbered terminal pair. A double acting solenoid will use two pairs (as shown in the image above). If your solenoid does not come with color coded wiring, check the datasheet to make sure to wire with the proper polarity.

## 41.6 Solenoid Voltage Jumper



The PCM is capable of powering either 12V or 24V solenoids, but all solenoids connected to a single PCM must be the same voltage. The PCM ships with the jumper in the 12V position as shown in the image. To use 24V solenoids move the jumper from the left two pins (as shown in the image) to the right two pins. The overlay on the PCM also indicates which position corresponds to which voltage. You may need to use a tool such as a small screwdriver, small pair of pliers, or a pair of tweezers to remove the jumper.



## CHAPTER 42

---

### Status Light Quick Reference

---

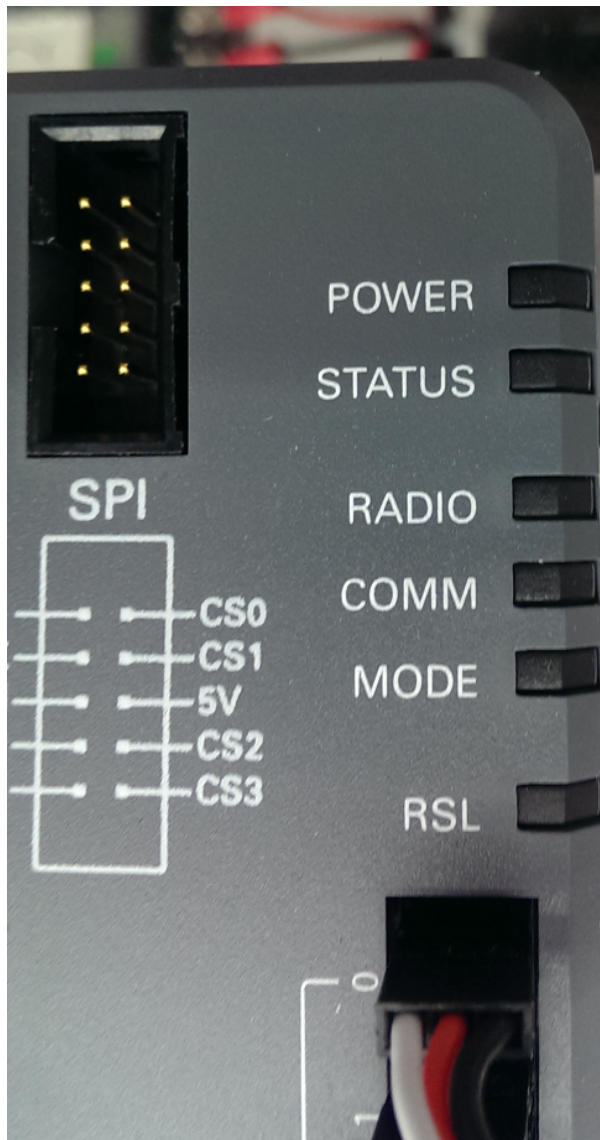
Many of the components of the FRC Control System have indicator lights that can be used to quickly diagnose problems with your robot. This guide shows each of the hardware components and describes the meaning of the indicators. Photos and information from Innovation FIRST and Cross the Road Electronics.

## 42.1 Robot Signal Light (RSL)



- Solid ON - Robot On and Disabled
- Blinking - Robot On and Enabled
- Off - Robot Off, roboRIO not powered or RSL not wired properly.

## 42.2 RoboRIO



### 42.2.1 Power

- Green - Power is good
- Amber - Brownout protection tripped, outputs disabled
- Red - Power fault, check user rails for short circuit

### 42.2.2 Status

- On while the controller is booting, then should turn off
- 2 blinks - Software error, reimage roboRIO

- 3 blinks - Safe Mode, restart roboRIO, reimage if not resolved
- 4 blinks - Software crashed twice without rebooting, reboot roboRIO, reimage if not resolved
- Constant flash or stays solid on - Unrecoverable error

### 42.2.3 Radio

- Not currently implemented

### 42.2.4 Comm

- Off - No Communication
- Red Solid - Communication with DS, but no user code
- Red Blinking - E-stop
- Green Solid - Good communication with DS

### 42.2.5 Mode

- Off - Outputs disabled (robot in Disabled, brown-out, etc.)
- Amber/Orange - Autonomous Enabled
- Green - Teleop Enabled
- Red - Test Enabled

### 42.2.6 RSL

- See above

## 42.3 OpenMesh Radio

<b>Power</b>	
Blue	On or Powering Up
Blue Blinking	Powering Up
<b>Eth Link</b>	
Blue	Link Up
Blue Blinking	Traffic Present
<b>WiFi</b>	
	Bridge Mode, Unlinked or non-FRC firmware
Off	
Red	AP, Unlinked
Yellow/Orange	AP, Linked
Green	Bridge Mode, Linked

WiFi light only works after radio has been power cycled.



### 42.3.1 Power

- Blue - On or Powering Up
- Blue Blinking - Powering Up



### 42.3.2 Eth Link

- Blue - Link Up
- Blue Blinking - Link Up + Traffic Present

### 42.3.3 WiFi

- Off - Bridge Mode Unlinked or Non-FRC Firmware
- Red - AP Mode Unlinked
- Yellow/Orange - AP Mode Linked
- Green - Bridge Mode Linked

## 42.4 Power Distribution Panel



LED Fault Table

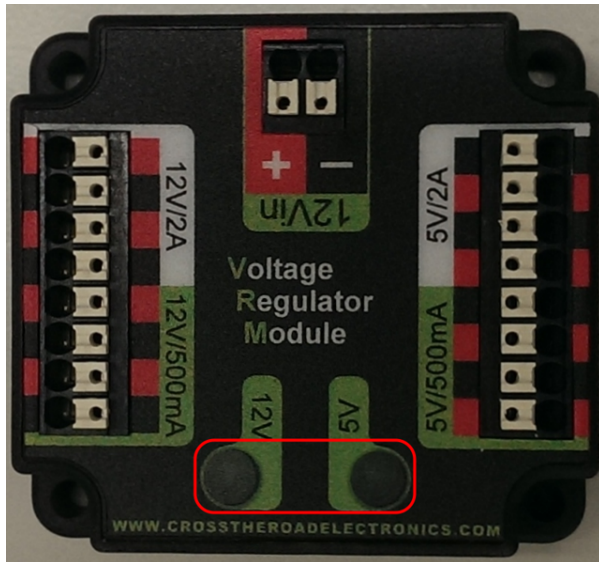
LED	Strobe	Slow	Long
Green	No Fault - Robot Enabled	No Fault - Robot Disabled	NA
Orange	NA	Sticky Fault	NA
Red	NA	No CAN Comm	NA

\*If PCM LED contains more than one color, see LED Special States Table

LED Special States Table

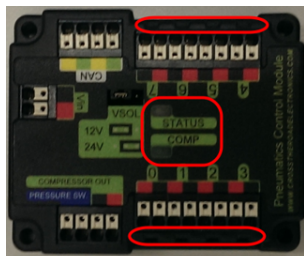
LED Colors	Problem
Red/ Orange	Damaged Hardware
Green/ Orange	In Bootloader
No LED	No Power / Incorrect Polarity

## 42.5 Voltage Regulator Module



The status LEDs on the VRM indicate the state of the two power supplies. If the supply is functioning properly the LED should be lit bright green. If the LED is not lit or is dim, the output may be shorted or drawing too much current.

## 42.6 Pneumatics Control Module



LED Fault Table

LED	Strobe	Slow	Long
Green	No Fault - Robot Enabled	No Fault - Robot Disabled	NA
Orange	NA	Sticky Fault	NA
Red	NA	No CAN Comm OR Solenoid Fault (Blinks Solenoid Index)	Compressor Fault

\*If PCM LED contains more than one color, see LED Special States Table

LED Special States Table

LED Colors	Problem
Red/ Orange	Damaged Hardware
Green/ Orange	In Bootloader
No LED	No Power / Incorrect Polarity

Solenoid Channel LEDs - These LEDs are lit red if the Solenoid channel is enabled and not lit if it is disabled.

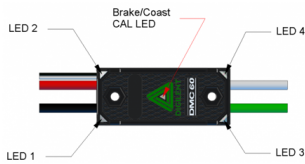
### 42.6.1 Comp

This is the Compressor LED. This LED is green when the compressor output is active (compressor is currently on) and off when the compressor output is not active.

### 42.6.2 Status

The status LED indicates device status as indicated by the two tables above. For more information on resolving PCM faults see the PCM User Manual. Note that the No CAN Comm fault will not occur only if the device cannot see communicate with any other device, if the PCM and PDP can communicate with each other, but not the roboRIO you will NOT see a No Can Comm fault.

## 42.7 Digilent DMC-60



At power-on the RGB LEDs will display a progressive blue color, which continually gets brighter. This lasts for approximately five seconds. During this time the motor controller will not respond to an input signal, nor will the output drivers be enabled. After the initial power-on has completed the device will begin normal operation and what gets displayed on the RGB LEDs will be a function of the input signal being applied, as well as the current fault state. Assuming that no faults have occurred the RGB LEDs will function as follows:

Servo Input Signal Applied	LED State
No input signal or Invalid input pulse width	Alternate between top (LED1 and LED2) and bottom (LED3 and LED4) LEDs being on and off. When on, the LEDs display color is orange.
Neutral input pulse width	All 4 LEDs on solid orange
Positive input pulse width	LEDs blink green in a clockwise circular pattern (LED1—>LED2—>LED3—>LED4—>LED1). The rate at which the LEDs update is proportional to the duty cycle of the output and increases with increased duty cycle. At 100% duty cycle, all four LEDs turn on solid green.
Negative input pulse width	LEDs blink red in a counter-clockwise circular pattern (LED1—>LED4—>LED3—>LED2—>LED1). The rate at which the LEDs update is proportional to the duty cycle of the output and increases with increased duty cycle. At 100% duty cycle, all four LEDs turn on solid red.

#### 9 Fault Indicators

When a fault condition is detected the output duty cycle is reduced to 0% and a fault is signaled. The output will remain disabled for 3 seconds. During this time the onboard LEDs (LED1, LED2, LED3, and LED4) are used to indicate the fault condition. The fault condition is indicated by toggling between the top (LED1 and LED2) and bottom (LED3 and LED4) LEDs being on and off. The top LEDs will be Red during them on state. The color of the bottom LEDs depends on which faults are presently active. The table below describes how the color of the bottom LEDs maps to the presently active faults.

Color	Over Temperature	Under Voltage
Green	✓	X
Blue	X	✓
Cyan/Aqua	✓	✓

When the center LED is off the device is operating in coast mode. When the center LED is illuminated the device is operating in brake mode. The Brake/Coast mode can be toggled by pressing down on the center of the triangle and then releasing the button.

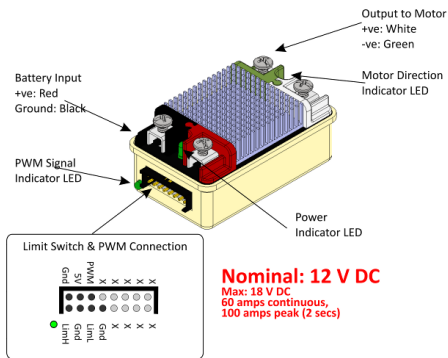
## 42.8 Jaguar speed controllers



LED State	Module Status	LED State	Module Status
<b>Normal Operating Conditions</b>			
Solid Yellow	Neutral (speed set to 0)	<b>Calibration Conditions</b>	
Fast Flashing Green	Forward	Fast Flashing Red and Green	Calibration mode active
Fast Flashing Red	Reverse	Fast Flashing Red and Yellow	Calibration mode failure
Solid Green	Full-speed forward	Slow Flashing Green and Yellow	Calibration mode success
Solid Red	Full-speed reverse	Slow Flashing Red and Green	Calibration mode reset to factory default settings success
<b>Fault Conditions</b>			
Slow Flashing Yellow	Loss of servo or Network link	<b>Other Conditions</b>	
Fast Flashing Yellow	Invalid CAN ID	Slow Flashing Green	Waiting in CAN Assignment mode
Slow Flashing Red	Voltage, Temperature, or Limit Switch fault condition		
Slow Flashing Red and Yellow	Current fault condition		

image here

## 42.9 Mindsensors SD 540



**Power LED**

This LED will turn Red when Power is supplied.

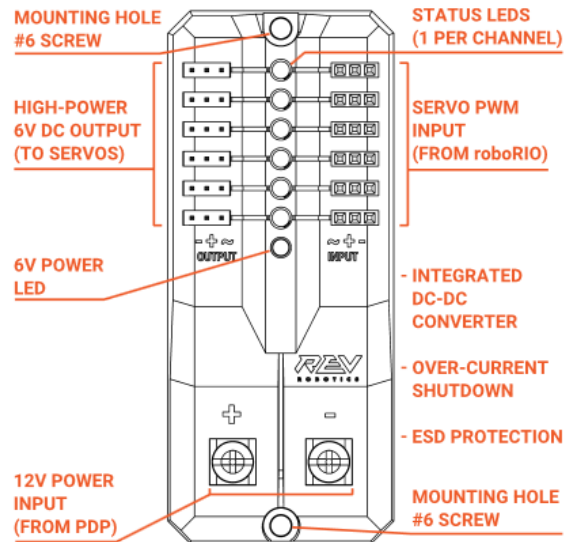
**Motor LED**

This LED turns Red in Forward direction and Green in Reverse direction.

**PWM Signal LED**

This LED turns Red when no valid PWM signal is detected, and turns Green when valid PWM signal is detected.

## 42.10 REV Robotics Servo Power Module



### STATUS LEDs

Each channel has a corresponding status LED that will indicate the sensed state of the connected PWM signal. The table below describes each state's corresponding LED pattern.

State	Pattern
No Signal	Blinking Amber
Left/Reverse Signal	Solid Red
Center/Neutral Signal	Solid Amber
Right/Forward Signal	Solid Green

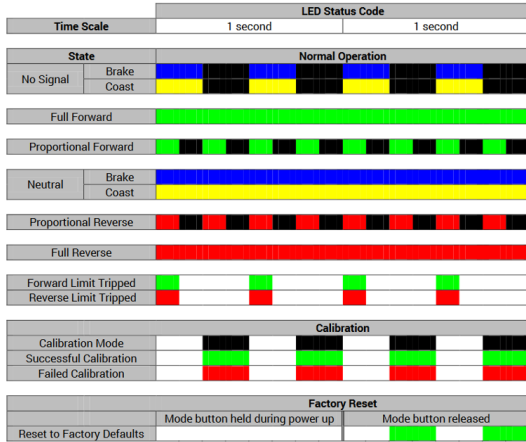
- 6V Power LED off, dim or flickering with power applied = Over-current shutdown

## 42.11 REV Robotics SPARK

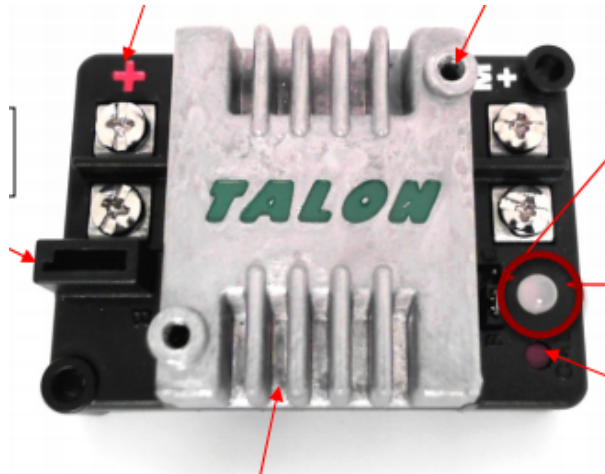
### 2.6 STATUS LED

The SPARK can display information about its current mode of operation via its tri-colored STATUS LED. The STATUS LED is located next to the motor output terminals and is labeled as STATUS with raised lettering on the SPARK housing.

Figure 2-6 shows the status codes associated with each operating state of the SPARK.



## 42.12 Talon speed controllers



The LED is used to indicate the direction and percentage of throttle and state of calibration. The LED may be one of three colors; red, orange or green. A solid green LED indicates positive output voltage equal to the input voltage of the Talon. A solid Red LED indicates an output voltage that is equal to the input voltage multiplied by -1(input voltage = 12 volts, output equals -12 volts). The LED will blink it's corresponding color for any throttle less than 100% (red indicates negative polarity, green indicates positive). The rate at which the led blinks is proportional to the percent throttle. The faster the LED blinks the closer the output is to 100% in either polarity.

The LED will blink orange any time the Talon is in the disabled state. This will happen if the PWM input signal is lost, or in FRC, when the robot is disabled. If the Talon is in the enabled state and the throttle is within the 4% dead band, the LED will remain solid orange.

Flashing Red/Green indicate ready for calibration. Several green flashes indicates successful calibration, and red several times indicates unsuccessful calibration.

## 42.13 Victor speed controllers

### 42.13.1 LED Indicator Status:

- Green - full forward
- Orange - neutral / brake
- Red - full reverse
- Flashing orange - no PWM signal
- Flashing red/green - calibration mode
- Flashing green - successful calibration
- Flashing red - unsuccessful calibration

## 42.14 Victor-SP speed controllers



Brake/Coast/Cal Button/LED - Red if the controller is in brake mode, off if the controller is in coast mode

### 42.14.1 Status

The Status LEDs are used to indicate the direction and percentage of throttle and state of calibration. The LEDs may be one of three colors; red, orange or green. Solid green LEDs indicate positive output voltage equal to the input voltage of the Victor-SP. Solid Red LEDs indicate an output voltage that is equal to the input voltage multiplied by -1 (input voltage = 12 volts, output equals -12 volts). The LEDs will blink in the corresponding color for any throttle less than 100% (red indicates negative polarity, green indicates positive). The rate at which the LEDs blink is proportional to the percent throttle. The faster the LEDs blink the closer the output is to 100% in either polarity.

The LEDs will blink orange any time the Victor-SP is in the disabled state. This will happen if the PWM input signal is lost, or in FRC, when the robot is disabled. If the Victor-SP is in the enabled state and the throttle is within the 4% dead band, the LED will remain solid orange.

Flashing Red/Green indicate ready for calibration. Several green flashes indicates successful calibration, and red several times indicates unsuccessful calibration.

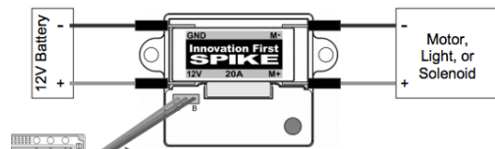
## 42.15 Talon-SRX speed controllers

Blink Codes During Calibration	
Status LEDs Blink Code	Talon SRX State
Flashing Red/Green	Calibration Mode
Blinking Green	Successful Calibration
Blinking Red	Failed Calibration

Blink Codes During Normal Operation		
LEDs	Colors	Talon SRX State
Both	Blinking Green	Forward throttle is applied. Blink rate is proportional to Duty Cycle
Both	Blinking Red	Reverse throttle is applied. Blink rate is proportional to Duty Cycle
None	None	No Power is being applied to Talon SRX
LEDs Alternate <sup>1</sup>	Off/Orange	CAN bus detected, robot disabled
LEDs Alternate <sup>1</sup>	Off/Slow Red	CAN bus/PWM is not detected
LEDs Alternate <sup>1</sup>	Off/Fast Red	Fault Detected
LEDs Alternate <sup>1</sup>	Red/Orange	Damaged Hardware
LEDs Strobe "towards" (M+) <sup>2</sup>	Off/Red	Forward Limit Switch or Forward Soft Limit
LEDs Strobe "towards" (M-) <sup>2</sup>	Off/Red	Reverse Limit Switch or Reverse Soft Limit
LED1 Only "closest" to M+/V+	Green/Orange	In Boot-loader

B/C CAL Blink Codes	
B/C CAL Button Color	Talon SRX State
Solid Red	Brake Mode
Off	Coast Mode

## 42.16 Spike relay configured as a motor, light, or solenoid switch

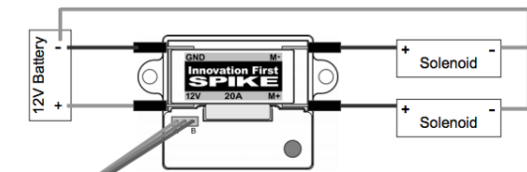


INPUTS		OUTPUTS		Indicator	Motor Function
Fwd(Wht)	Rev(Red)	M+	M-		
0	0	GND	GND	Orange	OFF / Brake Condition (default)
1	0	+12v	GND	Green	Motor rotates in one direction
0	1	GND	+12v	Red	Motor rotates in opposite direction
1	1	+12v	+12v	Off	OFF / Brake Condition

Notes:

- 'Brake' refers to the dynamic stopping of the motor due to the shorting of the motor inputs. This condition is not optional when going to an off state.
- The INPUT Fwd and Rev are defined as follows: 0 (Off) and 1 (On).

## 42.17 Spike relay configured as for one or two solenoids



INPUT		OUTPUTS		Indicator	Solenoid Function
Fwd(Wht)	Rev(Red)	M+	M-		
0	0	GND	GND	Orange	Both Solenoids OFF (default)
1	0	+12v	GND	Green	Solenoid connected to M+ is ON
0	1	GND	+12v	Red	Solenoid connected to M- is ON
1	1	+12v	+12v	Off	Both Solenoids ON

Note:

- The INPUT Fwd and Rev are defined as follows: 0 (Off) and 1 (On).



---

## FRC CAN Device Specifications

---

This document seeks to describe the basic functions of the current FRC CAN system and the requirements for any new CAN devices seeking to work with the system.

### 43.1 Addressing

FRC CAN nodes assign arbitration IDs based on a pre-defined scheme that breaks the ID into 5 components:

#### 43.1.1 Device Type

This is a 5-bit value describing the type of device being addressed. A table of currently assigned device types can be found below. If you wish to have a new device type assigned from the `Reserved` pool, please submit a request to `FIRST`.

Device Types	
Broadcast Messages	0
Robot Controller	1
Motor Controller	2
Relay Controller	3
Gyro Sensor	4
Accelerometer	5
Ultrasonic Sensor	6
Gear Tooth Sensor	7
Power Distribution Module	8
Pneumatics Controller	9
Miscellaneous	10
IO Breakout	11
Reserved	12-30
Firmware Update	31

### 43.1.2 Manufacturer

This is an 8-bit value indicating the manufacturer of the CAN device. Currently assigned values can be found in the table below. If you wish to have a manufacturer ID assigned from the `Reserved` pool, please submit a request to FIRST.

Manufacturer	
Broadcast	0
NI	1
Luminary Micro	2
DEKA	3
CTR Electronics	4
REV Robotics	5
Grapple	6
MindSensors	7
Team Use	8
Reserved	9-255

### 43.1.3 API/Message Identifier

The API or Message Identifier is a 10-bit value that identifies a particular command or message type. These identifiers are unique for each Manufacturer + Device Type combination (so an API identifier that may be a “Voltage Set” for a Luminary Micro Motor Controller may be a “Status Get” for a CTR Electronics Motor Controller or `Current Get` for a CTR Power Distribution Module).

The Message identifier is further broken down into 2 sub-fields: the 6-bit API Class and the 4-bit API Index.

### 43.1.4 API Class

The API Class is a 6-bit identifier for an API grouping. Similar messages are grouped into a single API Class. An example of the API Classes for the Jaguar Motor Controller is shown in the table below.

API Class	
Voltage Control Mode	0
Speed Control Mode	1
Voltage Compensation Mode	2
Position Control Mode	3
Current Control Mode	4
Status	5
Periodic Status	6
Configuration	7
Ack	8

### 43.1.5 API Index

The API Index is a 4-bit identifier for a particular message within an API Class. An example of the API Index values for the Jaguar Motor Controller Speed Control API Class is shown in the table below.

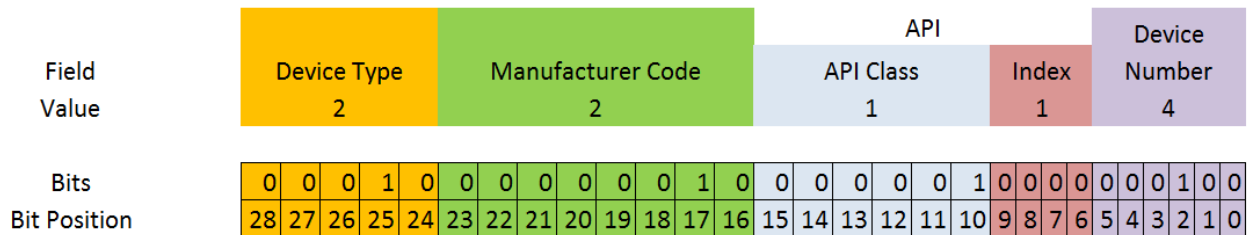
API Index	
Enable Control	0
Disable Control	1
Set Setpoint	2
P Constant	3
I Constant	4
D Constant	5
Set Reference	6
Trusted Enable	7
Trusted Set No Ack	8
Trusted Set Setpoint No Ack	10
Set Setpoint No Ack	11

### 43.1.6 Device Number

Device Number is a 6-bit quantity indicating the number of the device of a particular type. Devices should default to device ID 0 to match other components of the FRC Control System. Device 0x3F may be reserved for device specific broadcast messages.

Example

Speed Control Mode Disable from Luminary Micro Jaguar Speed Controller (dev # 4)



## 43.2 Protected Frames

FRC CAN Nodes which implement actuator control capability (motor controllers, relays, pneumatics controllers, etc.) must implement a way to verify that the robot is enabled and that commands originate with the main robot controller (i.e. the roboRIO).

## 43.3 Broadcast Messages

Broadcast messages are messages sent to all nodes by setting the device type and manufacturer fields to 0. The API Class for broadcast messages is 0. The currently defined broadcast messages are shown in the table below:

API Index	
Enable Control	0
Disable Control	1
Set Setpoint	2
P Constant	3
I Constant	4
D Constant	5
Set Reference	6
Trusted Enable	7
Trusted Set No Ack	8
Trusted Set Setpoint No Ack	10
Set Setpoint No Ack	11

Devices should disable immediately when receiving the Disable message (arbID 0), implementation of other broadcast messages is optional.

## 43.4 Requirements for FRC CAN Nodes

For CAN Nodes to be accepted for use in the FRC System, they must:

- Communicate using Arbitration IDs which match the prescribed FRC format:
  - A valid, issued CAN Device Type (per Table 1 - CAN Device Types)
  - A valid, issued Manufacturer ID (per Table 2 - CAN Manufacturer Codes)
  - API Class(es) and Index(s) assigned and documented by the device manufacturer
  - A user selectable device number if multiple units of the device type are intended to co-exist on the same network.
- Support the minimum Broadcast message requirements as detailed in the Broadcast Messages section.
- If controlling actuators, utilize a scheme to assure that the robot is issuing commands, is enabled, and is still present
- Provide software library support for LabVIEW, C++, and Java or arrange with FIRST or FIRSTs Control System Partners to provide such interfaces.

---

Robot Preemptive Troubleshooting

---

---

**Note:** In FIRST Robotics Competition, robots take a lot of stress while driving around the field. It is important to make sure that connections are tight, parts are bolted securely in place and that everything is mounted so that a robot bouncing around the field does not break.

---

## 44.1 Check battery connections

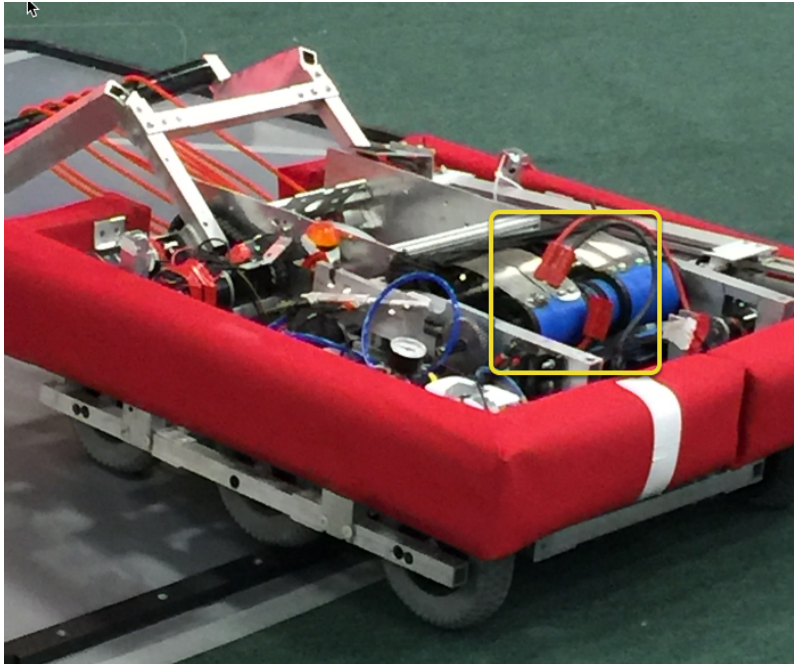


The tape that should be covering the battery connection in these examples has been removed to illustrate what is going on. On your robots, the connections should be covered.

Wiggle battery harness connector. Often these are loose because the screws loosen, or sometimes the crimp is not completely closed. You will only catch the really bad ones though because often the electrical tape stiffens the connection to a point where it feels stiff. Using a voltmeter or Battery Beak will help with this.

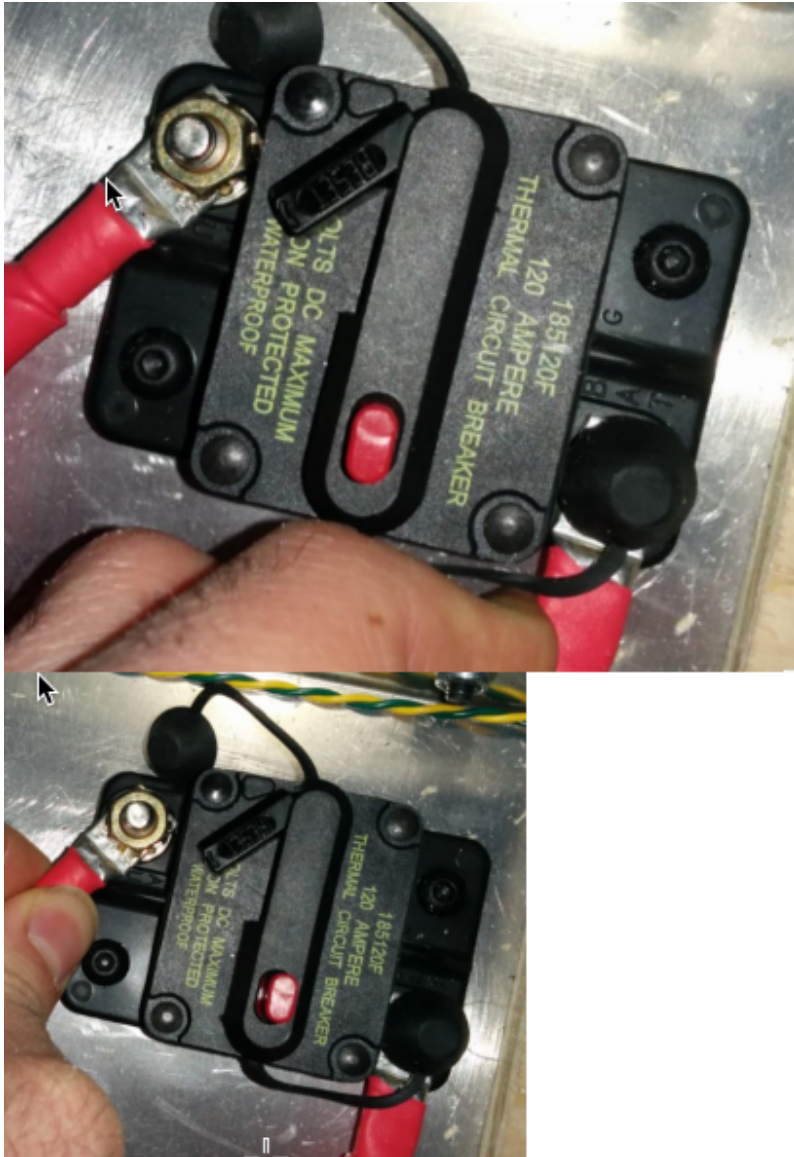
Apply considerable force onto the battery cable at 90 degrees to try to move the direction of the cable leaving the battery, if successful the connection was not tight enough to begin with and it should be redone.

## 44.2 Secure the battery to robot connection



In almost every event we see at least one robot where a not properly secured battery connector (the large Anderson) comes apart and disconnects power from the robot. This has happened in championship matches on the Einstein and everywhere else. Its an easy to ensure that this doesn't happen to you by securing the two connectors by wrapping a tie wrap around the connection. 10 or 12 tie wraps for the piece of mind during an event is not a high price to pay to guarantee that you will not have the problem of this robot from an actual event after a bumpy ride over a defense.

## 44.3 120 Amp circuit breaker



Apply a twisting force onto the cable to rotate the harness. If you are successful then the screw is not tight enough. Split washers might help here, but in the mean time, these require checking every few matches.

Because the metal is just molded into the case, every once in awhile you will break off the bolt, ask any veteran team and they'll tell you they go through a number of these every few seasons. After tightening the nut, retest by once again trying to twist the cable.



## 44.4 Power Distribution Panel (PDP)



**Just by removing the battery cover, often you can confirm the washer.**

Make sure that split washers were placed under the PDP screws, but it is not easy to visually confirm, and sometimes you can't. You can check by removing the case. Also if you squeeze the red and black wires together, sometimes you can catch the really loose connections.

## 44.5 Tug test everything



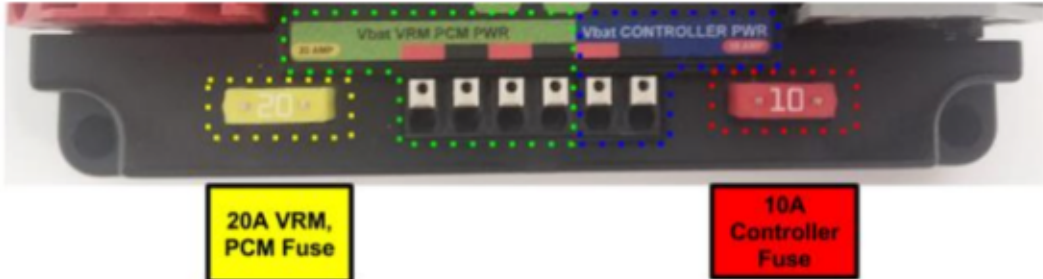
The Weidmuller contacts for power, compressor output, roboRIO power connector, and radio power are important to verify by tugging on the connections as shown. Make sure that none of the connections pull out.

Look for possible or impending shorts with Weidmuller connections that are close to each other, and have too-long wire-lead lengths (wires that are stripped extra long).

Spade connectors can also fail due to improper crimps, so tug-test those as well.

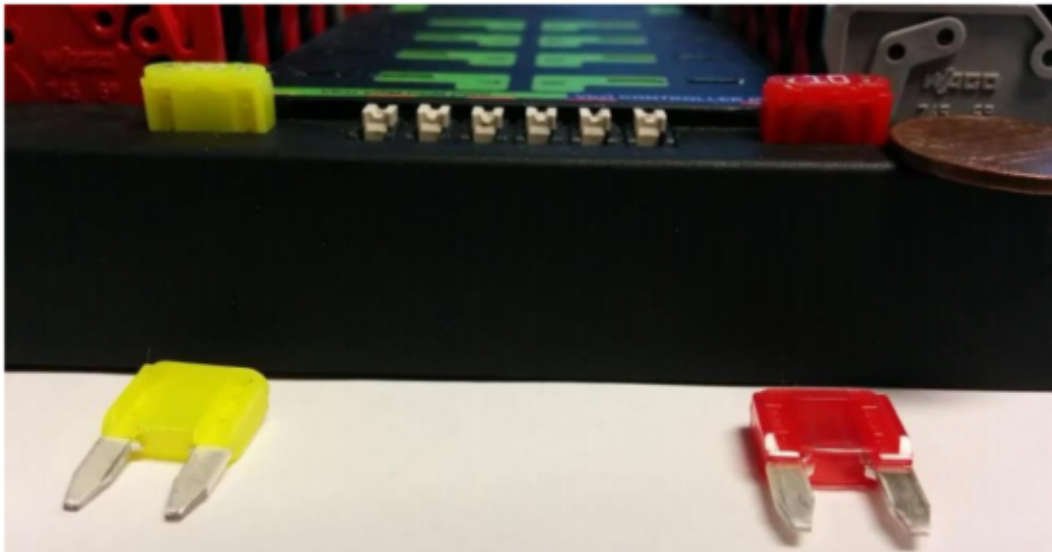
## 44.6 Blade fuses

### 2.4. Blade Fuse



Be sure to place the 20A fuse (yellow) on the left and the 10A fuse (red) on the right.

**Warning:** Also take care to ensure **fuses are fully seated** into the fuse holders. The fuses should descend **at least** as far as the figure below (different brand fuses have different lead lengths). It should be **nearly impossible to remove the fuse** with bare hands (without the use of pliers). If this is not properly done, the robot/radio may exhibit **intermittent connectivity issues**.



If you can remove the blade fuses by hand then they are not in completely. Make sure that they are completely seated in the PDP so that they don't pop out during robot operation.

## 44.7 RoboRIO swarf

Swarf is: fine chips or filings of stone, metal, or other material produced by a machining operation. Often modifications must be made to a robot while the control system parts are in place. The circuit board for the roboRIO is conformally coated, but that doesn't absolutely guarantee that metal chips won't short out traces or components inside the case. In this case, you must exercise care in making sure that none of the chips end up in the roboRIO or any of the other components. In particular, the exposed 3 pin headers are a place where chips can enter the case. A quick sweep through each of the four sides with a flashlight is usually sufficient to find the really bad areas of infiltration.

## 44.8 Radio barrel jack

Make sure the correct barrel jack is used, not one that is too small and falls out for no reason. This isn't common, but ask an FTA and every once in awhile a team will use some random barrel jack that is not sized correctly, and it falls out in a match on first contact.

## 44.9 Ethernet cable

If the RIO to radio ethernet cable is missing the clip that locks the connector in, get another cable. This is a common problem that will happen several times in every competition. Make sure that your cables are secure. The clip often breaks off, especially when pulling it through a tight path, it snags on something then breaks.

## 44.10 Cable slack

Cables must be tightened down, particularly the radio power and ethernet cable. The radio power cables don't have a lot of friction force and will fall out (even if it is the correct barrel) if the weight of the cable-slack is allowed to swing freely.

Ethernet cable is also pretty heavy, if it's allowed to swing freely, the plastic clip may not be enough to hold the ethernet pin connectors in circuit.

## 44.11 Reproducing problems in the pit

Beyond the normal shaking and rattling of all cables while the robot is power and tethered, you might try picking up one side of the robot off the ground and drop it, and see if you lose connection. The driving on the field, especially when trying to breach defenses will often be very violent. It's better to see it fail in the pit rather than in a critical match.

When doing this test it's important to be ethernet tethered and not USB tethered, otherwise you are not testing all of the critical paths. Check firmware and versions

Robot inspectors do this, but you should do it as well, it helps robot inspectors out and they appreciate it. And it guarantees that you are running with the most recent, bug fixed code. You wouldn't want to lose a match because of an out of date piece of control system software on your robot.

## 44.12 Driver station checks

We often see problems with the Drivers Station. You should:

- ALWAYS bring the laptop power cable to the field, it doesn't matter how good the battery is, you are allowed to plug in at the field.
- Check the power and sleep settings, turn off sleep and hibernate, screen savers, etc.
- Turn off power management for USB devices (dev manager)
- Turn off power management for ethernet ports (dev manager)
- Turn off windows defender
- Turn off firewall



- Close all apps except for DS/Dashboard when out on the field.
- Verify that there is nothing unnecessary running in the application tray in the start menu (bottom right side)

## 44.13 Handy tools



There never seems to be enough light inside robots, at least not enough to scrutinize the critical connection points, so consider using a handheld LED flashlight to inspect the connections on your robot. They're available from home depot or any hardware/automotive store.

Wago tool is nice to for redoing weidmuller connections with stranded wires. Often I'll do one to show the team, and then have them do the rest using the WAGO tool to press down the white-plunger while they insert the stranded wire. The angle of the WAGO tool makes this particularly helpful.



---

**Note:** This document describes the IP configuration used at events, both on the fields and in the pits, potential issues and workaround configurations.

---

### 45.1 TE.AM IP Notation

The notation TE.AM is used as part of IPs in numerous places in this document. This notation refers to splitting your four digit team number into two digit pairs for the IP address octets.

Example: 10.TE.AM.2

Team 12 - 10.0.12.2

Team 122 - 10.1.22.2

Team 1212 - 10.12.12.2

Team 3456 - 10.34.56.2

### 45.2 On the Field

This section describes networking when connected to the Field Network for match play

#### 45.2.1 DHCP (typical configuration)

The Field Network runs a DHCP server with pools for each team that will hand our addresses in the range of 10.TE.AM.20 and up with subnet masks of 255.0.0.0

- OpenMesh OM5P-AN or OM5P-AC radio - Static 10.TE.AM.1 programmed by Kiosk

- roboRIO - DHCP 10.TE.AM.2 assigned by the Robot Radio
- Driver Station - DHCP (“Obtain an IP address automatically”) 10.TE.AM.X assigned by field
- IP camera (if used) - DHCP 10.TE.AM.Y assigned by Robot Radio
- Other devices (if used) - DHCP 10.TE.AM.Z assigned by Robot Radio

### 45.2.2 Static (workaround configuration)

It is also possible to configure static IPs on your devices to accommodate devices or software which do not support mDNS. When doing so you want to make sure to avoid addresses that will be in use when the robot is on the field network. These addresses are 10.TE.AM.1 and 10.TE.AM.4 for the OpenMesh radio and the field access point and anything 10.TE.AM.20 and up which may be assigned to a device still configured for DHCP. The roboRIO network configuration can be set from the webdashboard.

- OpenMesh radio - Static 10.TE.AM.1 programmed by Kiosk
- roboRIO - Static 10.TE.AM.2 would be a reasonable choice, subnet mask of 255.255.255.0 (default)
- Driver Station - Static 10.TE.AM.5 would be a reasonable choice, subnet mask must be 255.0.0.0
- IP Camera (if used) - Static 10.TE.AM.11 would be a reasonable choice, subnet 255.255.255.0 should be fine
- Other devices - Static 10.TE.AM.6-.10 or .12-.19 (.11 if camera not present) subnet 255.255.255.0

## 45.3 In the Pits

**New for 2018: There is now a DHCP server running on the wired side of the Robot Radio in the event configuration.**

### 45.3.1 DHCP (typical configuration)

- OpenMesh radio - Static 10.TE.AM.1 programmed by Kiosk.
- roboRIO - 10.TE.AM.2, assigned by Robot Radio
- Driver Station - DHCP (“Obtain an IP address automatically”), 10.TE.AM.X, assigned by Robot Radio
- IP camera (if used) - DHCP, 10.TE.AM.Y, assigned by Robot Radio
- Other devices (if used) - DHCP, 10.TE.AM.Z, assigned by Robot Radio

### 45.3.2 Static (workaround configuration)

It is also possible to configure static IPs on your devices to accommodate devices or software which do not support mDNS. When doing so you want to make sure to avoid addresses that will be in use when the robot is on the field network. These addresses are 10.TE.AM.1 and 10.TE.AM.4 for the OpenMesh radio