J – Documentation

JDocs (2014) - FRC Team 949

Thanks to team 2853 for graphics and information in The Book of the FRC Electrical

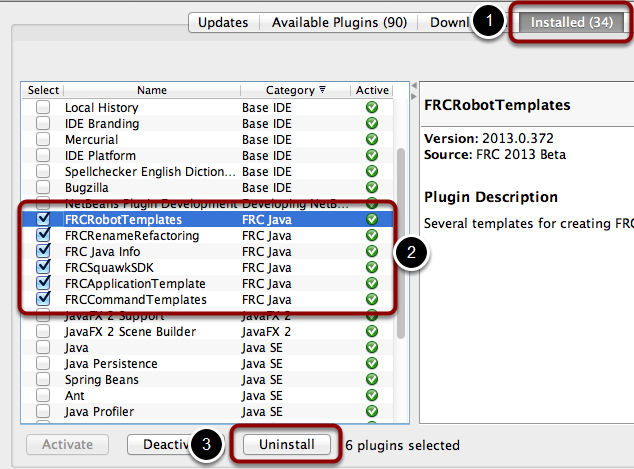
Consider this documentation a quick reference to the most common issues that occurred during the build season. This is by no means an in-depth look at all the topics discussed, but a short refresher to some of the important points on general topics.

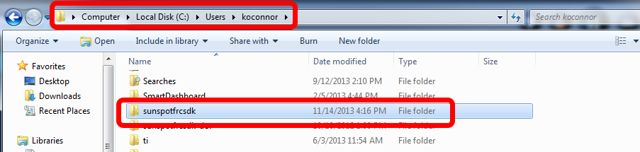
I kept it short and sweet because more detailed documentation is readily available online.

Only the basics are laid out here.

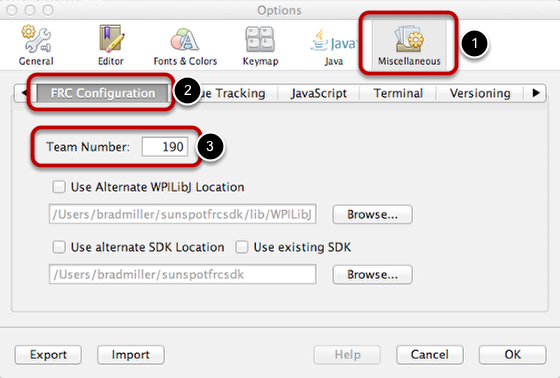
# Updating Software

### Netbeans and Plugins

1. If you need to install software, the easiest way to install Netbeans for FRC is to use the JDK/Netbeans co-bundle provided by Oracle. If you do not already have a JDK installed or do not know if you have JDK installed, it is recommended to use this option.
2. Uninstall previous versions of the FRC plugins in Netbeans (if applicable) 
3. You should also delete the SunspotFRCSDK directory on your machine. For Windows 7 and 8 users this directory is located in the User directory of the user where you installed NetBeans.

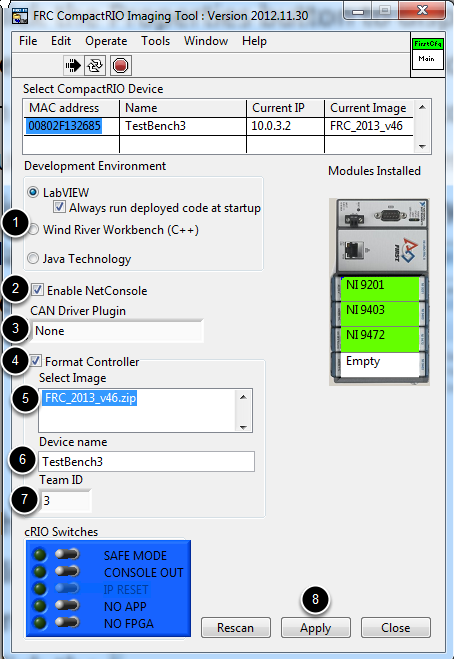


1. To update the plugins automatically, connect to the internet:
   1. Start NetBeans and choose Tools/Plugins from the menu.
   2. Click the "Settings" tab on the Plugins window and select "Add".
   3. Enter a name for the plugins, like "FRC plugins" and enter the URL "<http://first.wpi.edu/FRC/java/netbeans/update/Release/updates.xml>"
   4. Then click "OK" to add WPLILib to the list of available plugins.
2. To install the plugins manually
   1. Go to <http://first.wpi.edu/FRC/java/netbeans/update/Release>
   2. You will see 6 .nbm files listed. Download each of these files to a location on your computer. If you are downloading the plugins on a computer that is not your development system, copy the files onto a flash drive and bring them to the development system.
3. Remember to reconfigure the team number in NetBeans after every update.



### General Software

1. To update the software resources for the competition, go to the link and download: <http://www.usfirst.org/roboticsprograms/frc/technical-resources>
   1. National Instruments update
   2. Specific software language update (ie. Java update, C++ update)
2. The product install key for the National Instruments update should be in the kit of parts
   1. The update should include the update to the Driver Station
3. Update the cRIO using the cRIO imaging tool and image included in the National Instruments Update
   1. To avoid corruption of the cRIO, especially with our older computer, you MUST connect by Ethernet cable and disable all other network cards.
      1. Go to Control Panel > Network center > manage adapters (or similar)
4. The java documentation is available here: <http://www.wbrobotics.com/javadoc/edu/wpi/first/wpilibj/package-summary.html>



Once the cRIO is detected, it should pop up in the box near the top of the Imaging Tool. Select the options you wish to use, then image the cRIO:

1. Select the programming language to use.
2. Select whether to enable NetConsole (Java teams will have NetConsole enabled automatically)
3. Select a CAN plugin if appropriate. Note that if you select a CAN plugin with the Console Out feature enabled, on a 4-slot cRIO-FRC II the Console Out will be disabled automatically, and on an 8-slot cRIO-FRC you will see a prompt to flip the switch to disable Console Out.
4. Check the box next to **Format Controller**.
5. Verify that the image listed is this year’s version, make sure you have installed the latest update for your programming language.
6. Enter a name for the cRIO device.
7. Enter your FRC team number in the **Team ID** box.
8. Click **Apply.** The cRIO imaging tool will begin imaging your cRIO, after it is complete, you should see a message indicating that the imaging is complete and you need to load code in order to use the cRIO.
   1. **Note: For proper operation on the playing field at competition, the subnet mask should be set back to 255.0.0.0 after imaging is complete if this PC is to be used as a Driver Station.**

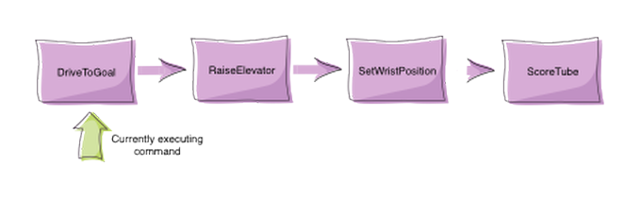
# Programming

### Command Based Programming

The program is organized around two fundamental concepts: **Subsystems** and **Commands**.

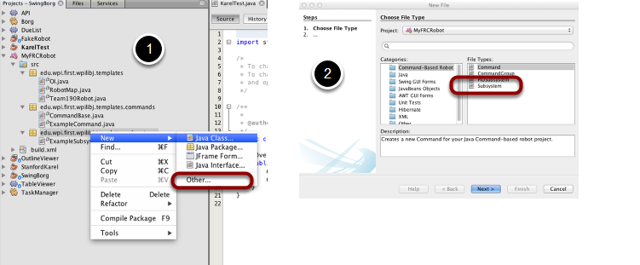
**Subsystems** - define the capabilities of each part of the robot and are subclasses of Subsystem.

**Commands** - define the operation of the robot incorporating the capabilities defined in the subsystems. Commands are subclasses of Command or CommandGroup. Commands run when scheduled or in response to buttons being pressed or virtual buttons from the SmartDashboard.



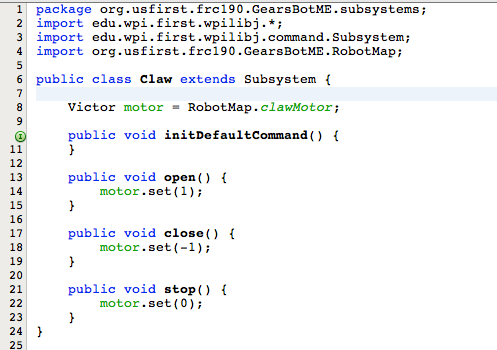
Commands let you break up the tasks of operating the robot into small chunks. Each command has an execute() method that does some work and an isFinished() method that tells if it is done. This happens on every update from the driver station or about every 20ms.

### Adding Subsystems

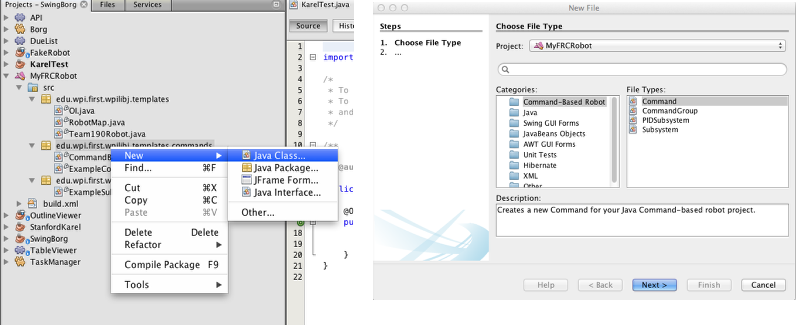


Add a new Subsystem class instance. Right-click on the subsystem package, and select New Subsystem. If it is not there, then select New Other (as shown above), then select subsystem. In the future, subsystem will be a choice on the New menu. Name the subsystem in the next dialog that pops up and click Finish.

A subsystem will contain all the methods that need to be called for that set of motors. This is an example of a simple subsystem for a single claw motor. Notice how the motor is defined and each action has a method to be called.



### Adding commands

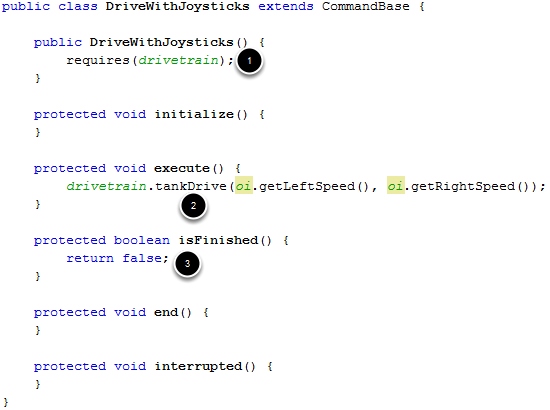


# Basic Command Format

To implement a command, a number of methods are overridden from the WPILib Command class. Most of the methods are boiler plate and can often be ignored, but are there for maximum flexibility when you need it. There a number of parts to this basic command class:

1. Constructor - Might have parameters for this command such as target positions of devices. Should also set the name of the command for debugging purposes. This will be used if the status is viewed in the dashboard. And the command should require (reserve) any devices is might use.
2. initialize() - This method sets up the command and is called immediately before the command is executed for the first time and every subsequent time it is started . Any initialization code should be here.
3. execute() - This method is called periodically (about every 20ms) and does the work of the command. Sometimes, if there is a position a subsystem is moving to, the command might set the target position for the subsystem in initialize() and have an empty execute() method.
4. isFinished() - This method returns true if the command is finished. This would be the case if the command has reached its target position, run for the set time, etc. There are other methods that might be useful to override and these will be discussed in later sections

An example of a simple command:

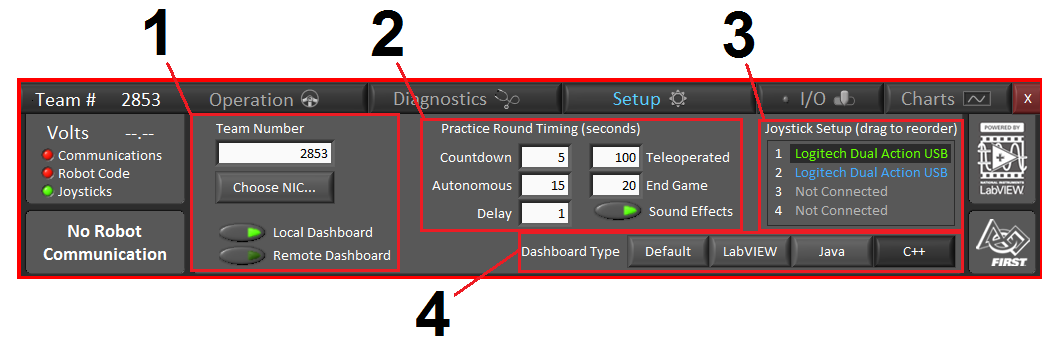


1. This example illustrates a simple command that will drive the robot using tank drive with values provided by the joysticks. The elements we've used in this command:
2. requires(drivetrain) - "drivetrain" is an instance of our Drivetrain subsystem. The instance is instantiated as static in Command Base so it can be referenced here. We need to require the drivetrain system as this command uses it when it executes.
3. execute() - In our execute method we call a tankDrive method we have created in our subsystem. This method takes two speeds as a parameter which we get from methods in the OI class. These methods abstract the joystick objects so that if we want to change how we get the speed later we can do so without modifying our commands (for example, if we want the joysticks to be less sensitive, we can multiply them by .5 in the getLeftSpeed method and leave our command the same).
4. isFinished - Our isFinished method always returns false meaning this command never completes on it's own. The reason we do this is that this command will be set as the default command for the subsystem. This means that whenever the subsystem is not running another command, it will run this command. If any other command is scheduled it will interrupt this command, then return to this command when the other command completes.

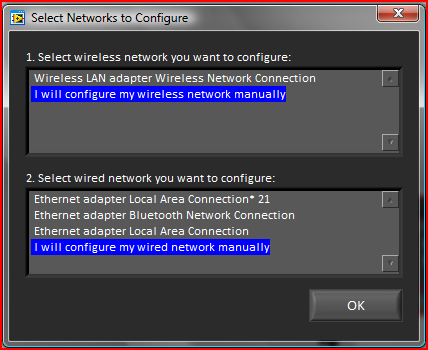
# Driver Station

### https://lh4.googleusercontent.com/3eLnmd_gO1W1G82E7JtG6uq2ROb2r8NA_D8nuMNwYn_nzlhKqYbBngsyCkcvroDdaBk158JFLiU-EjguuCWPOcF7sQNKJc2bPX70gely4sDs2ZUcEf8ViQY

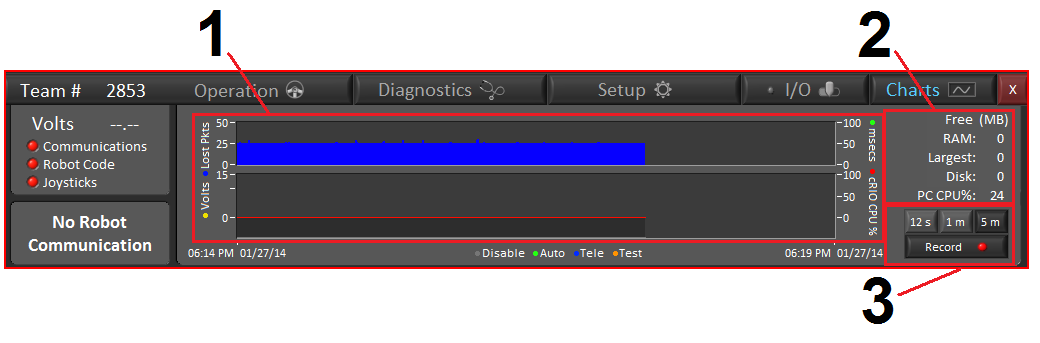
1. USB Devices connected to the computer. Usually you will have a joystick so the first grey light will be green. Unless you are using the Kinect, the red light can be left alone.
2. What the Driver Station has communications with. If you are hooked up to robot directly from computer to cRIO, only the Robot and Bridge lights should be green. If connected wirelessly or direct from computer to router, the DS Radio and Enet Link lights in addition to the former should be green.
3. Prints warning messages based on: it can’t communicate with robot; robot has no code; disconnected joysticks, lights, etc.
4. Clears the messages in **3**
5. Reboots the cRIO module. This must be done when emergency stopped.



1. Make sure the correct team number is set here.
   1. Click Choose NIC to make sure that the Driver Station does not automatically set ip addresses during build season
   2. However, during games (or if your router is set to dynamic) let it auto configure



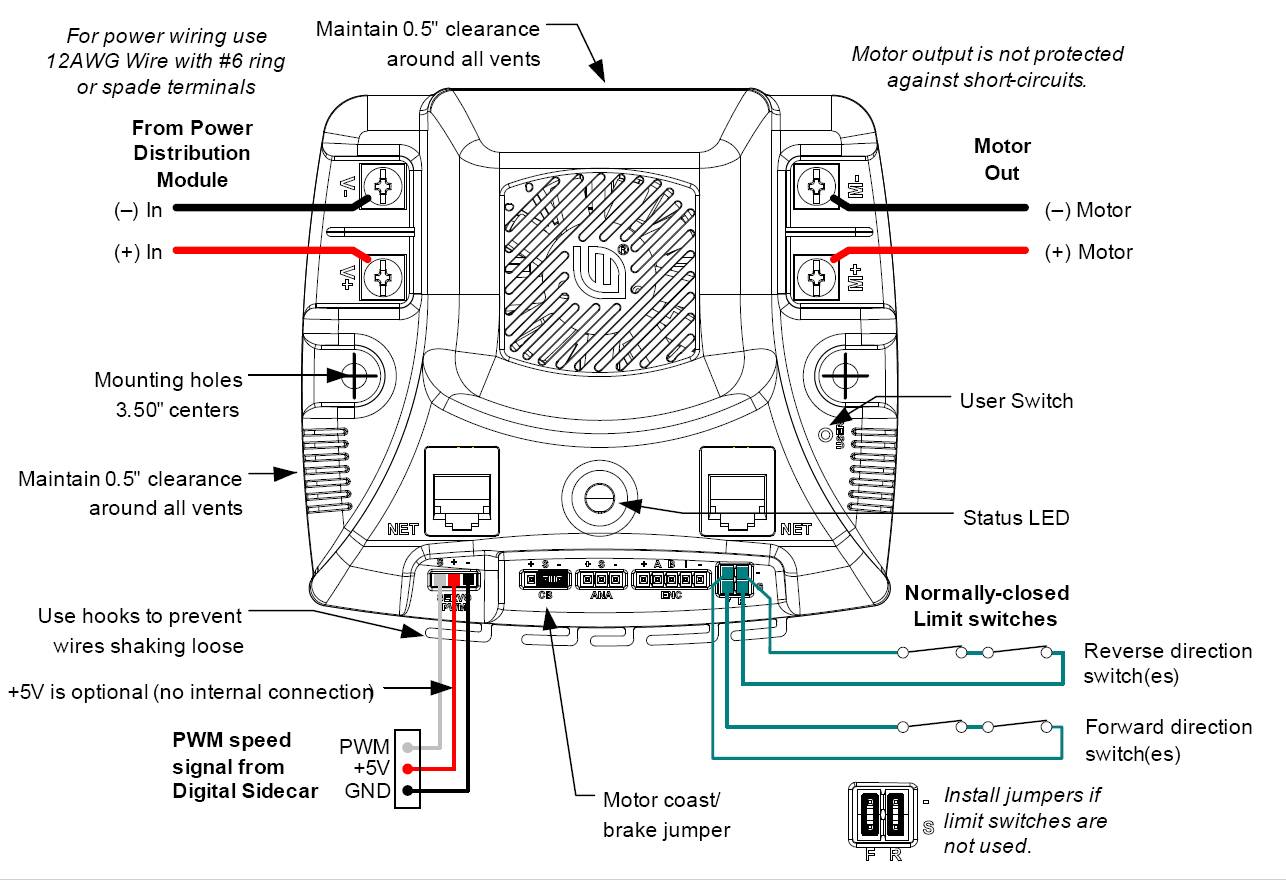
1. This is where the timing for the Practice button in Operations Tab is set. Set times(seconds) for each period; countdown before practice round starts, length of autonomous period, Teleoperated, End Game, and delay between periods. Arrow button with light on means that sounds will play for start and end of each portion of the practice round
2. Where joystick order is setup, green means that it is connected to configured joystick, blue means that a button on a configured joystick is being pressed; click and drag the joystick name to reorder the joysticks
3. When using a dashboard, select the type of programming language you use



1. Top Graph is # of lost packets of data in blue(left axis), trip time from computer to robot(right axis) ; Bottom graph is battery voltage in a yellow line(left axis), and % of cRIO cpu being used as a red line(right axis)
2. Numerical stats of Free memory of cRIO in megabytes; Free Random Access Memory, Largest free RAM block, disk space, and CPU
3. Set time scale for the graphs, and depressed record button means it is recording.

# Motors

### Jaguar

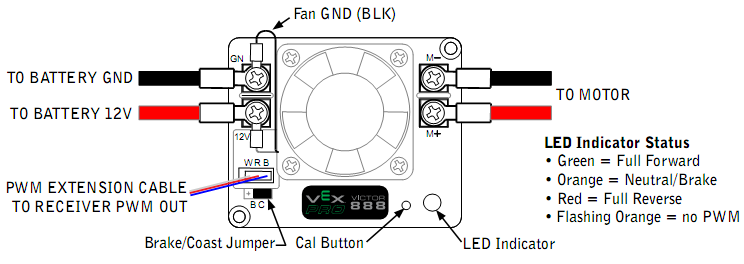


Tl;dr:

* Black wire to the “-“ symbol (or to the right when oriented to the picture).
* Power in (from power distribution board) with the “V+” and “V-“ symbol.
* Power out (to motors) with the “M+” and “M-“ symbol.
* Look for the motor coasting / braking jumper if needed

|  |  |
| --- | --- |
| **LED State** | **Module Status** |
| **Solid Yellow** | Neutral |
| **Fast Blinking Green** | Forward |
| **Fast Blinking Red** | Reverse |
| **Solid Green** | Max Speed Forward |
| **Solid Red** | Max Speed Reverse |
| **Slow Blinking Yellow** | Loss of Servo or Network Link |
| **Fast Blinking Yellow** | Invalid CAN ID |
| **Slow Blinking Red** | Voltage, Temperature, or Limit Switch fault condition |
| **Slow Blinking Red and Yellow** | Current Fault Condition |
| **Fast Blinking Red and Green** | Calibration Mode Active |
| **Fast Blinking Red and Yellow** | Calibration Mode Failure |
| **Slow Blinking Green and Yellow** | Calibration Mode Success |
| **Slow Blinking Red and Green** | Calibration Mode Reset to Factory Default Success |
| **Slow Blinking Green** | Waiting in CAN assignment mode |

### Victor



Tl;dr

* Black wire to the “B“ symbol (or to the right when oriented to the picture).
* White wire to the left when oriented to the picture
* Power in (from power distribution board) with the “GN” and “12V“ symbol.
* Power out (to motors) with the “M+” and “M-“ symbol.
* Flashing orange = not connected with PWM or robot disabled

### Spike

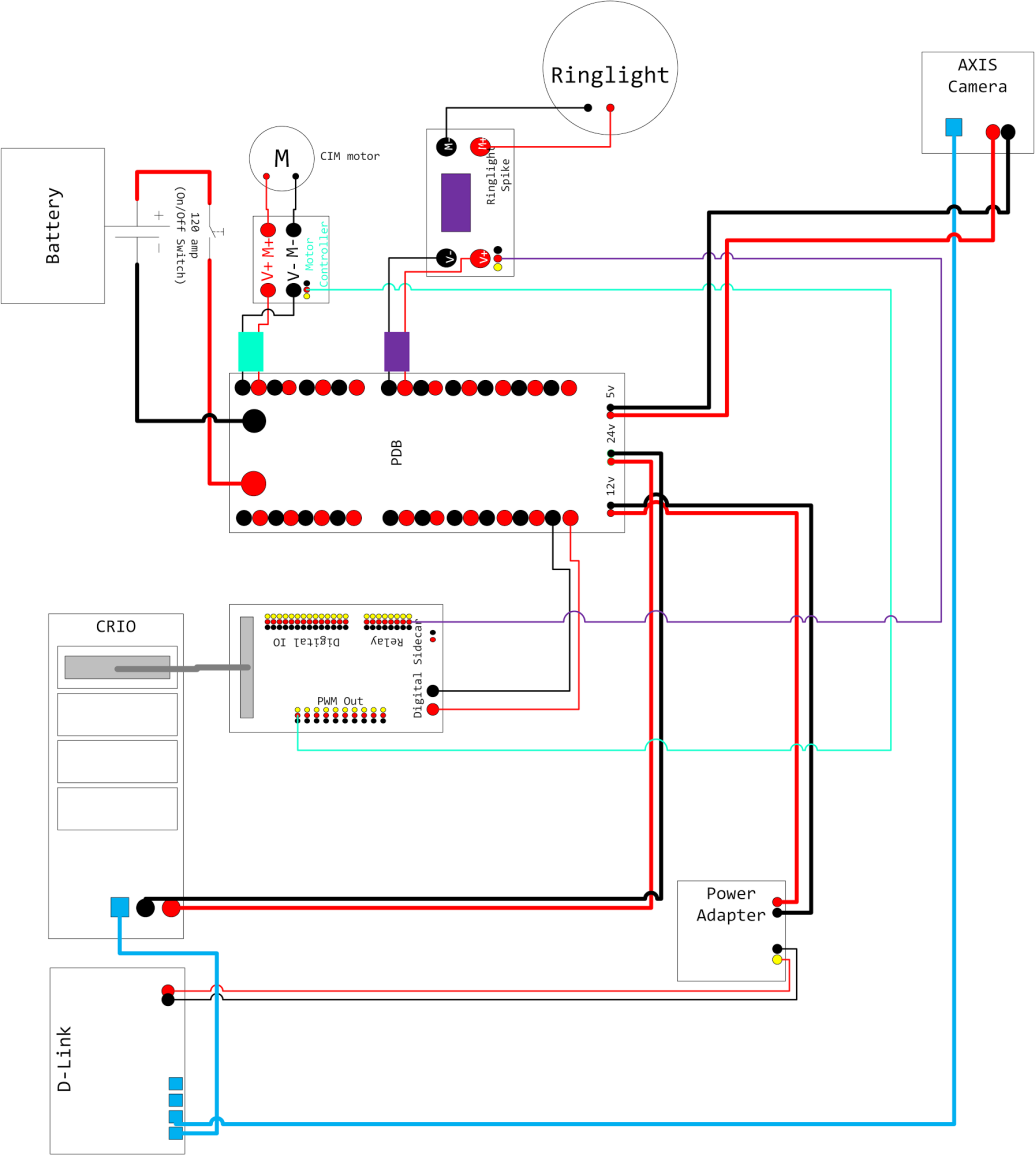
### https://lh5.googleusercontent.com/cs-hVnthplglKVEdIzSiffr1aGGam1NGQgvB9tEf9TO7v7nxOihzWybeJKcZ-QqYLnQHR4DQgFI8gAVhbT0e3bUfFgETwG3marhJ3ajfzcWBRwLJA1lhiFU

* Black wire to the “B“ symbol (or to the right when oriented to the picture).
* White wire to the left when oriented to the picture
* Power in (from power distribution board) with the “GN” and “12V“ symbol.
* Power out (to motors) with the “M+” and “M-“ symbol.
* A clicking noise means the spike is changing states

# Wiring

### Basic Wiring

A very basic electrical layout that can be used for testing purposes. It consists of the core electrical components (battery, cRIO, digital sidecar, D-Link, power adapter, and PDB) and extensions of the control system (motor controller, spike, camera, and ringlight).



### The Power Converter + router

The D-Link requires **5V** to operate--this is why a **12V/24V to 5V power converter** is connected to the PDB. It converts power so the D-Link can operate under its necessary voltage.

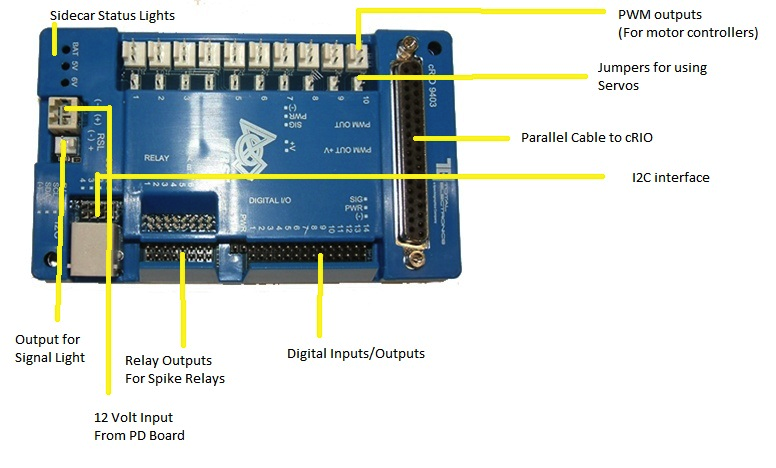




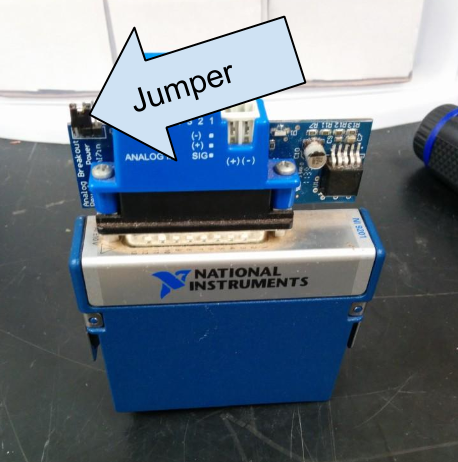
Most important things to remember about the router:

* Make sure the router is configured (refer to the FRC manual or look online)
* Make sure the router has power
* Make sure the ethernet cables are plugged in securely (on both ends)

### Digital Sidecar



### Analog Breakout



The Analog Breakout plugs into the 9201 module and has 8 PWM ports. It samples the analog outputs of sensors and treats them as a digital value (ex. gyros). It takes a 12V input via a WAGO connector and has a 1A 5V switching power supply. The three exposed pins on the left can be configured to monitor battery voltage or enabling port 8 via a jumper. Attaching the jumper to the left will enable battery voltage monitoring and disable port 8. Attaching the jumper to the right will disable battery monitoring and enable port 8.

Tl;dr:

Put a jumper where the picture says. You need it for battery voltage monitoring.

# Troubleshooting

Based off of experiences from the team in the past, these are the MOST probable issues occurring with the robot. These are not guaranteed fixes.

### Communications

* Make sure the computer’s network card is configured correctly.
* Make sure you have configured the router.
* Ensure that all necessary cables (Power and/or Ethernet cables) are plugged in securely.
* Ensure that the switch on the back of the router is set to “AP 2.4 GHz”
* Try resetting the power on the router. Remove its power source, wait 10 seconds, then plug the power cable back in. Sometimes wizards come and magically fix the router when you do this.
* Try resetting the router’s data. Get a toothpick or some other thin object, then use it to hold the reset button down for 10 seconds. Wait about 30 seconds for it to reboot. Then, refer to the initial configuration process above.

### Code

Robot Drive not updated enough:

* You are running a loop that took too long to complete (or never completes at all)
* A sensor (potentiometer) may be giving back values you were not prepared to handle
  + Try to code without complete dependence on sensors. They can, and will fail during competition.
* You may not be calling the drive function often enough, or ever.
* Make sure the drive train subsystem is required.
* Start AND reset any timers that you may have declared in the init function.

Motor not running:

* Make sure the ports were declared correctly in the RobotMap.
* Ensure you are sending the correct values to the motor controller:
  + Spikes are only on or off, forwards or reverse. Check to make sure you set it right.
  + Jaguars, Victors, and Talons need a speed set
* Make sure you set the direction of the motor controller correctly
  + The right and left sides are driving in the opposite direction
* If code is checked through and through, and you are positive it should work, you may have a wiring issue.

If all other possibilities have been checked, try recompiling the code onto the robot. It may still have an older version of the code.

### Wiring

Device not working as intended:

* First things first: Is it plugged in?
* Is it plugged in correctly? Make sure it really is connected, and doesn’t just appear to be connected.
* Are the connections secure? Connectors can come loose, so make sure to check the connectors and any solder points.
* Check PWM wires. Older PWM wires are sketchy and are notorious for being terrible.
* Test the voltage running through the wires.
  + 12 volts should be coming from the power distribution board
  + Unless a motor controller is enabled, there should be no voltage running through
* If wiring and PWM are checked multiple times and proven to be correct, you may have a programming issue.