# The FRC Bible at SPS

by C.D.Odom January 2022

Modified 12.7.22, 12.11.22, 1.4.23

## Team Quick Look

## 2022-23 Team Info

1. SPS Team Name: 1512
2. Red Trainer
   1. Laptop number: N/A
   2. Radio name: 1512\_RedTrainer
   3. roboRIO data:
      1. Hostname:
      2. MAC Address:
      3. Part number (P/N):
      4. Serial number (S/N):
      5. Current IP address: 10.15.12.93 ???
3. Red Team A
   1. Laptop number: Lenovo ThinkPad Red A
   2. Radio name: 1512\_RedA
   3. roboRIO data:
      1. Hostname:
      2. MAC Address:
      3. Part number (P/N):
      4. Serial number (S/N):
      5. Current IP address: 10.15.12.xxx
4. Red Team B
   1. Laptop number: Lenovo ThinkPad Red B
   2. Radio name: 1512\_RedB
   3. roboRIO data:
      1. Hostname:
      2. MAC Address:
      3. Part number (P/N):
      4. Serial number (S/N):
      5. Current IP address: 10.15.12.2
      6. Ethernet address: 169.254.41.59
5. Red Team C
   1. Laptop number:
   2. Radio name: 1512\_RedC
   3. roboRIO data:
      1. Hostname:
      2. MAC Address:
      3. Part number (P/N):
      4. Serial number (S/N):
      5. Current IP address: 10.15.12.xxx
6. Serial numbers
7. Addresses
   1. 1512 (on the training station)
      1. xxx
   2. 1512\_RedA (on the 2022-23 competition robot)
      1. Radio IP address: 10.15.12.2
      2. MAC Address: xxx
   3. RedB
      1. Radio IP address: 10.15.12.2
      2. MAC Address: 00:80:2F:30:DC:26
8. FTP information:
   1. FTP user name: “lvuser” (that is, lowercase “LVUSER”)
   2. FTP password: “” (i.e., blank)
   3. FTP port: 22
   4. FTP connection for all SPS radios: <ftp://10.15.12.2> (laptop must be on that roboRIO’s network)

## New Laptop Setup

* 1. Checked laptop System:
     1. 64-bit system
     2. 16GB RAM
     3. Disk space before and after installation:
        1. 112GB free before installation.
        2. 106GB free after FRC Game Tools install.
        3. 101GB free after roboRIO Image Tools install.
  2. Logins
     1. Lenovo ThinkPad Red A
        1. Local Admin: RobotAdmin\RobotAdm!n
        2. Local Standard: Robotics\robotics
  3. Admin install:
     1. Wi-Fi (had to use my own credentials, not the RobotAdmin one — shesh!)
     2. Power:
        1. Start >> Settings Gear >> System >> Power & Sleep
           1. While on battery: 30 mins sleep, 60 minutes power down;
           2. While plugged in: never, never
     3. Add Hibernate as a power option:
        + 1. Start >> Settings Gear >> System >> Power & Sleep >> Additional Power Settings
          2. Select **Choose what the power button does** or **Choose what closing the lid does**. They both open the same window.
          3. Click **Change settings that are currently unavailable**.
          4. Additional choices under **Shutdown settings** will become available. Click the box next to **Hibernate**.
          5. Click **Save Changes**.
          6. Now, when you open the **Start**menu and click the **Power** button, you'll see an additional **Hibernate** option along with the others.
     4. Control Panel >> Hardware and Sound >> Power Options >> System Settings
        + 1. When I press the **power button**: **Do nothing; Do nothing**
          2. When I **close the lid**: **Hibernate; Hibernate**
     5. Enable Clipboard History in Windows 10
        + 1. Start >> Settings Gear >> System >> Clipboard >> Clipboard History to On
     6. Pin **Snipping Tool** to the Taskbar.
     7. Pin **Windows Defender** app to the Taskbar.
     8. Edge setup:
        + 1. Dark mode
          2. Vertical tabs
          3. Added Core, Millville Portal, Canvas
     9. O365 login – it recognized the credentials, so nothing was needed. IT already set it up.
     10. OneDrive in File Explorer – this is my SPS OneDrive account!!! Not RobotAdmin!
         + 1. C:\Users\robotadmin\OneDrive – St Paul’s School
           2. Use this folder to share files between admin computers!
     11. Adobe sign in
         + 1. Used the RobotAdmin credentials

## Turn off the Windows Firewall

1. Open Windows Defender Firewall.
   1. This *should* be docked to the Taskbar.
   2. Turn off firewall to all traffic (Domain, Private, and Public) by clicking on “Window Defender Firewall Properties”

## Installation of the Software

1. **Follow this religiously:**
   1. <https://docs.wpilib.org/en/stable/docs/zero-to-robot/introduction.html>
2. **Uninstall Old FRC Game Tools**
   1. If there is an old version of the FRC Game Tools, **first uninstall it**:
      1. <https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-2/frc-game-tools.html#uninstall-old-versions-recommended>
      2. This takes forever!
3. **Install the new** [**FRC Game Tools**](https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-2/frc-game-tools.html)**:**
   1. **Follow this page religiously:**
      1. <https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-2/frc-game-tools.html>
   2. Here is a summary:
   3. Download the [FRC Game Tools](https://www.ni.com/en-us/support/downloads/drivers/download.frc-game-tools.html" \t "_blank) from NI:
      1. <https://www.ni.com/en-us/support/downloads/drivers/download.frc-game-tools.html#440024>
      2. **Important:** The Game Tools 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 FRC Game Tools, restarting the installer if necessary.
      3. Run the downloaded executable file to start the install process. Click Yes if a Windows Security prompt appears.
      4. Agree to the NI Package Manager License Agreement
      5. **Disable Windows Fast Startup!!!**
         1. It is recommended to leave this screen as-is, as Windows Fast Startup can cause issues with the NI drivers required to image the roboRIO. Go ahead and click **Next**.
      6. Click **Next** to all other windows that follow. (Accept any License Agreements.)
      7. Graphical user interface, text, application, email

         Description automatically generatedNear the end, you will be asked to log into your [ni.com](https://ni.com/) account to **activate the FRC software**. THIS PART IS CONFUSING TO ME, AND I STILL DO NOT HAVE A SOLID HANDLE ON THE WORKFLOW HERE!  
           
           
         1. If you don’t have an account, select Create account to create a free account.
         2. We will use Will’s credentials:
            1. User: [wrenauld@sps.edu](mailto:wrenauld@sps.edu)
            2. Pass: robotics
            3. Security question answers: “robotics”
            4. Not what to do next. Here are things that have sort-of worked in the past:

Choose Option #2 in the dropdown menu: “**Enter a serial number**”. Here, enter the Serial Numbers for the roboRIO(s):

Red Trainer. S/N: **31DF9C4**

Red A. S/N: **31BE68A**

Red B. S/N: **305CCF1**

Choose Option #4 in the dropdown menu: “**Connect to a volume license server**”. Here, you will need to enter the *server address or host name*. Enter **B05P07808** (zeros, not oh’s) then click the **Connect** button.

We got this from logging into the FIRST Inspires dashboard >> Payment and Product >> Password and Voucher Codes >> NI LabView Address >> Password/Voucher code: **B05P07808**

Logged into the NI Website and activated the product by entering the roboRIO(s) serial number.

Simply pressing the **Extend Trial** button.

* + - * 1. If this is the first time activating this year’s software on this account, you will see a message about a valid license not being found. **You can ignore this**.

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

* + 1. If your products activate successfully, an Activation Successful message will appear. If everything activated successfully, click **Finish**.

Graphical user interface, text, application, email

Description automatically generated

* + 1. You will be prompted whether to enable the NI update service. You can choose to not enable the update service. It is not recommended to install these updates unless directed by FRC through our usual communication channels (FRC Blog, Team Updates or E-mail Blasts).
    2. Graphical user interface, application

       Description automatically generatedYou must reboot the computer when finished.
    3. After the reboot, you will see the **FRC Driver Station** and **roboRIO Imaging Tool** on the laptop’s desktop, as shown. These are important applications for the next steps.
    4. x
  1. x

1. **Install WPILib:**
   1. **Follow this page religiously:**
      1. <https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-2/wpilib-setup.html>
   2. As explained in the above site, you can download the latest release of the installer from [GitHub](https://github.com/wpilibsuite/allwpilib/releases/latest/). Ensure that you download the correct binary for your OS and architecture:
      1. <https://github.com/wpilibsuite/allwpilib/releases/tag/v2022.4.1>
         1. WPILib is designed to install to different folders for different years, so that it is not necessary to uninstall a previous version before installing this year’s WPILib.
      2. **Important to install roboRIO image 2022\_v4.x for this release!**
      3. ***Upgrading*** *the WPLib is easy: simply download and run the new installer and it will update your current installation. If you already have the 2022 WPILib VScode installed, it will detect it and you can simply click "next" for that installation step. After installation, VScode will prompt you when opening your robot project whether you want to upgrade it to this version. Note running the installer is necessary to install upgraded desktop tools.*
      4. Scroll to “**Assets**” at the bottom of the page:

A screenshot of a computer screen

Description automatically generated with medium confidence

* + 1. We will choose the 64-bit Windows version, which is distributed as a disk image file (\*.iso) for Windows.
    2. It takes a minute to download. When it does, right-click on the file and select “**Mount**”.
    3. Run the installer (**WPILibInstaller.exe**).
    4. In the window that opens, click the **Start** button.
    5. Install **Everything** for **All Users!** (See the WPI Page link above!)

Graphical user interface, text, application

Description automatically generated

* + 1. There is a lot here, so read carefully the WPI Page link above.
  1. Download for **This Computer Only:**

Graphical user interface, text, application, email

Description automatically generated

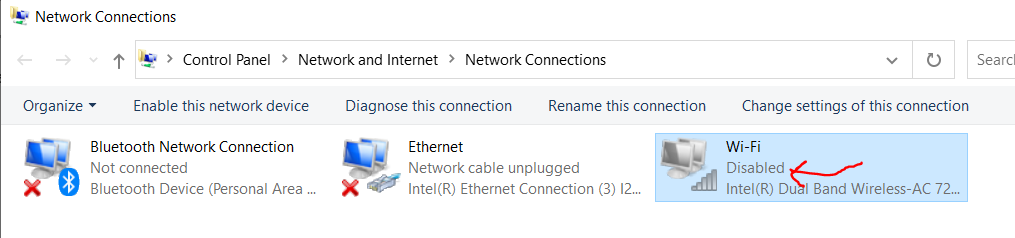
* 1. This will download VS Code (Visual Studio). Press **Next** to begin. It will take a minute.
  2. A picture containing logo

     Description automatically generatedPress **Finish** and find the icon on the desktop:
  3. Click on the “WPLib VS Code” desktop icon to start VS Code. When it opens, right-click on the VS icon in the Taskbar and select “**Pin to Taskbar**”. The icon will look like this:



* 1. Close VS Code and keep going….

## Preparing Your Robot

1. **Image the RoboRio 1 (not 2):**
   * 1. **We use the roboRIO 1, not the 2. (The roboRIO 2 use a microSD card and is faster.)**
     2. **Follow this page religiously:**
     3. See: <https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-3/imaging-your-roborio.html?highlight=imaging-your-roborio>
     4. **Warning:**
        1. Before imaging your roboRIO, you must have completed installation of the [FRC Game Tools](https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-2/frc-game-tools.html) (see <https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-2/frc-game-tools.html>).
        2. 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).
     5. **Not sure if the following step is necessary!?!** It doesn’t seem necessary. However, if you have issues, then do the following step:
        1. ~~Disable the WiFi Network Adapter:~~
           1. ~~Windows >> Settings >> Network & Internet >> Status >> Change Adapter Options >> Disable WiFi~~
     6. Remove all wires from the roboRIO **except** for Power to the roboRIO.
     7. Connect the power wires from the roboRIO to the **Power Distribution Panel (PDP)** (black and red flat box). This should be connected to the port labeled **Vbat CONTROLLER PWR**.
        1. Make sure this side has a **10-amp** fuse!
     8. Power the RIO from a 12V battery.
     9. 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. **Do NOT use an ethernet connection to image the RIO!**
     10. A picture containing text, electronics

         Description automatically generatedLaunch the **Imaging Tool** as an **Administrator**:
     11. This will open the following window:

Graphical user interface, text, application

Description automatically generated

* + - 1. Select **Edit Startup Settings** to configure the startup setting of the roboRIO without imaging the RIO. Not necessary generally.
      2. Select **Format Target** when you need to load a new image or reflash the existing image. This is the most common option.
      3. Select **Update Firmware** if required at the start of the FRC season! No harm in doing it each year.
    1. Let’s **Update the Firmware** now:
       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: **1512**
       4. Select the **latest** firmware file in the bottom right
       5. Click the **Update** button. This takes 3-10 minutes. **Make sure the computer is plugged in!**

Graphical user interface, application

Description automatically generatedGraphical user interface, application

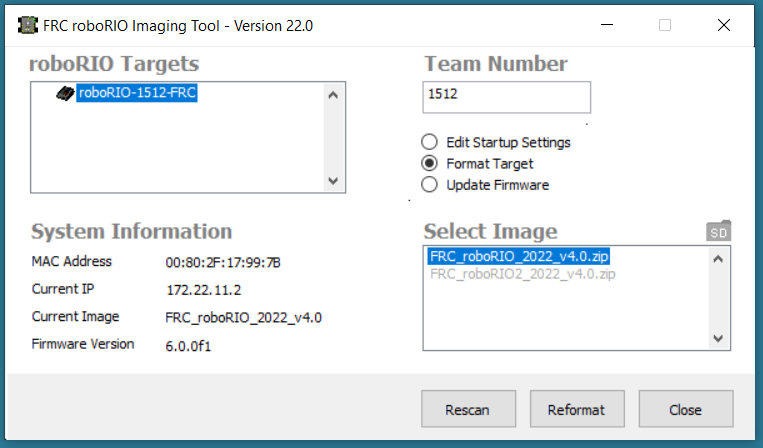
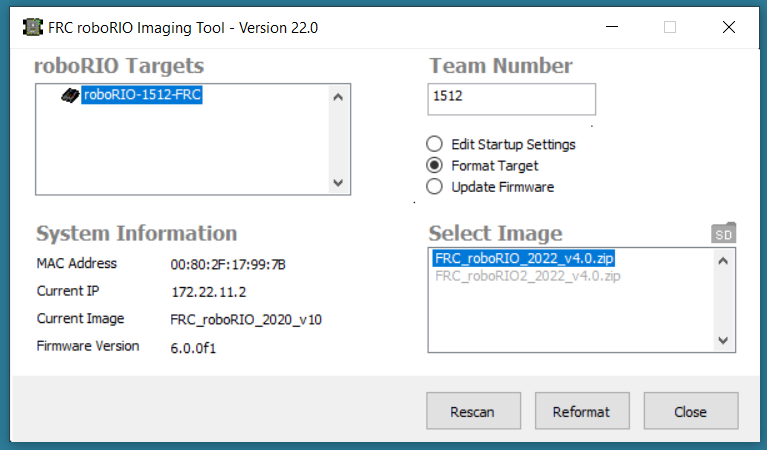
Description automatically generated

* + - 1. Record the roboRIO’s information (Hostname, MAC Address, and Current IP Address) in the section named Team Data at the top of this document.
      2. Close the Imaging tool.
    1. Now, **Image the roboRIO**:
       1. Reopen the Imaging Tool.
       2. Make sure the roboRIO is selected in the top left pane
       3. Select **Format Target** in the right pane
       4. Enter your **team number** in the box: **1512**
       5. Select the **latest** image version in the box.
          1. Again, we are using the robotRIO 1, NOT the robotRIO 2
       6. Click **Reformat** to begin the imaging process. This will take 3-10 minutes. **Make sure the computer is plugged in!**

Graphical user interface, text, application

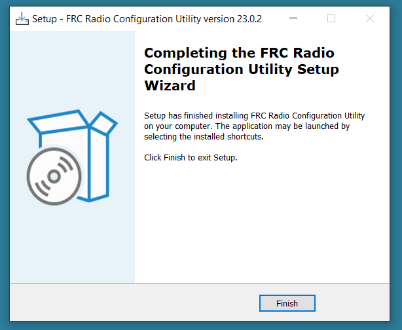
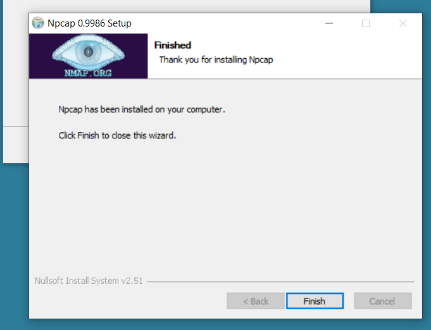
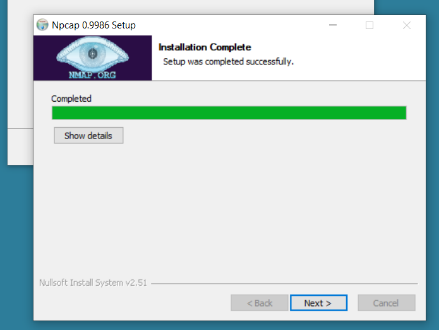
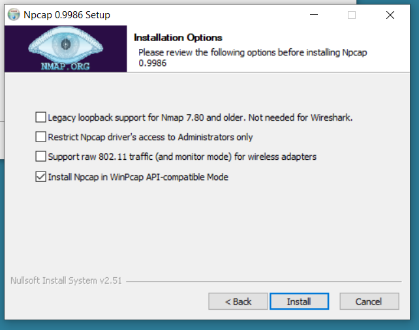
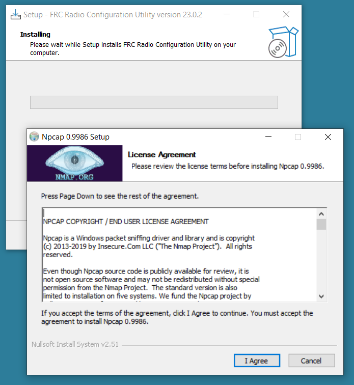
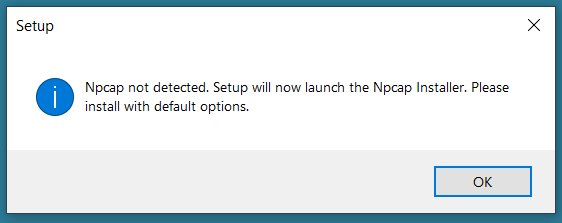
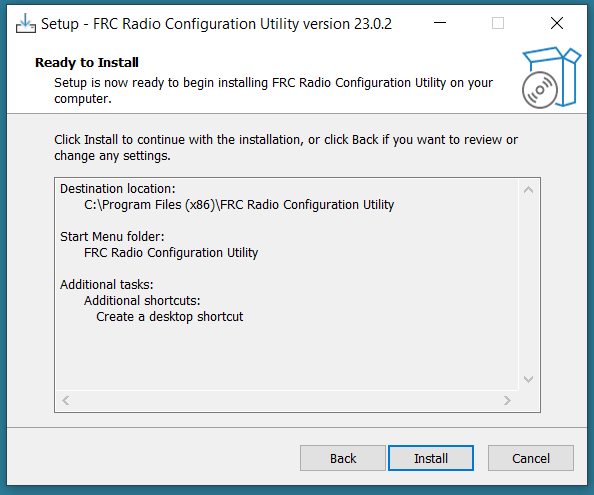
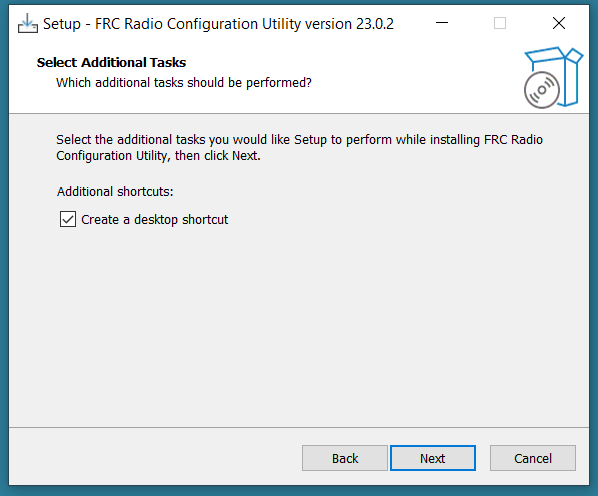
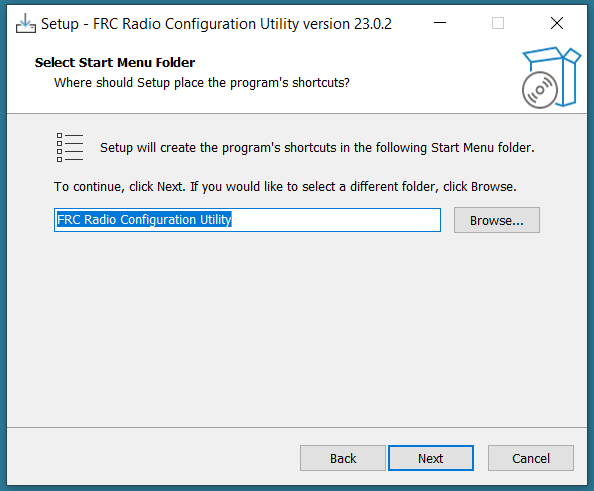
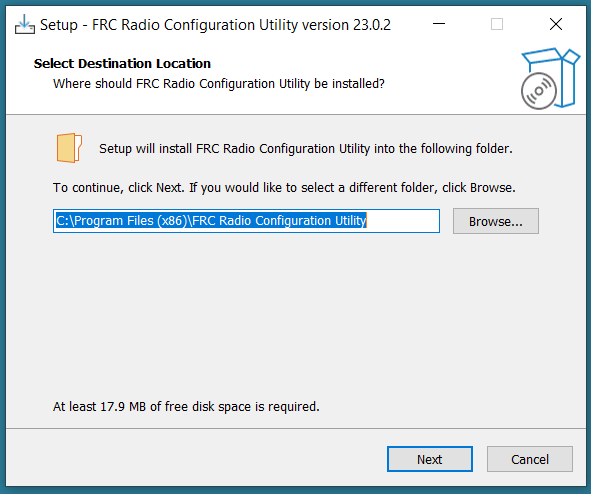
Description automatically generated

* + - 1. When successful, you will get a message box. Click OK. Then press the Close button to close the imaging tool.
      2. Finally, **you will need to reboot the RIO using the Reset Button on the device or cycle the power for the changes to take effect**.
         1. You can reopen the roboRIO Imaging Tool to verify the image has been updated:



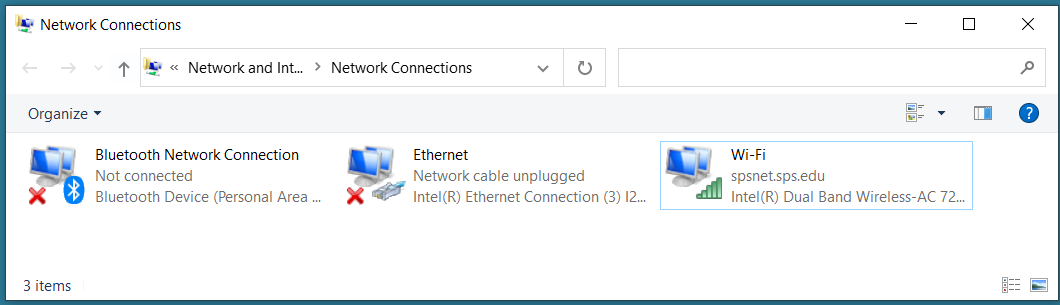
* + 1. See the bottom of the WPI page (<https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-3/imaging-your-roborio.html#troubleshooting>) for **Troubleshooting tips**!

1. **Program the Radio:**
2. **Follow this page religiously:**
   * + 1. See: <https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-3/radio-programming.html>
3. The **FRC Radio Configuration Utility** requires **administrator privileges** to configure the network settings on your machine. The program should request the necessary privileges automatically (may require a password if run from a non-administrator account), but if you are having trouble, try running it from an administrator account.
4. Download the latest FRC Radio Configuration Utility Installer:
   1. <https://firstfrc.blob.core.windows.net/frc2023/Radio/FRC_Radio_Configuration_23_0_2.zip>
5. Install the **FRC Radio Configuration Utility** by running the EXE file you just downloaded.
   * + 1. Part of the installation prompts will include installing **Npcap** if it is not already present. The Npcap installer contains a number of checkboxes to configure the install. You should leave the options as the **defaults**.
       2. Here are the steps in order:

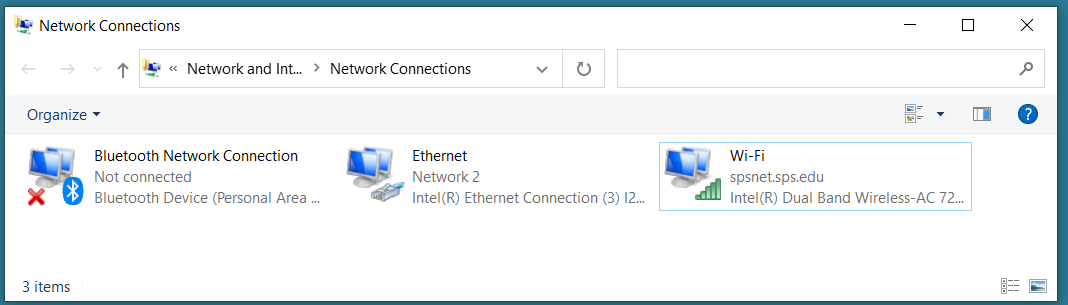




1. When done, you will see this icon (but do NOT open it yet!)
2. **Before using the FRC Radio Configuration Utility** **software:**
   * + 1. First, make sure that all ethernet cables and Bluetooth devices are unplugged from your laptop.
       2. Then, open the network adapters settings: **Settings >> Network & Internet >> Change adapter options >> Change adapter settings**. Notice the devices are enabled but things may not be connected. Note the Wi-Fi may be connected, but other devices should not be connected:



* + - 1. Power up the robot’s radio and give it a couple of minutes to warm up. It is ready to go when the right-most LED on the radio is solid **orange**.
         1. If powering the radio via PoE cable (Power Over Ethernet), plug an Ethernet cable from the PC into the socket side of the PoE adapter (where the roboRIO would plug in). If you experience issues configuring through the PoE adapter, you may try connecting the PC to the alternate port on the radio.
         2. If you are not powering the radio with a PoE cable, power the radio using the 12V/2A port on the Voltage Regulator Module (VRM).
      2. Plug one end of an ethernet cable into the port on the radio **closest to the power jack**. Make sure no other devices are connected to your computer via ethernet.
      3. Plug the other end of the ethernet cable into your laptop and watch the Ethernet device. Notice that it now says a cable is attached:



* + - 1. **Not sure if the following step is necessary!?!** It doesn’t seem necessary. However, if you have issues, then do the following step:
         1. ~~For each of the~~ *~~other~~* ~~adapters (other than the one connected to the radio), right-click on the adapter, and select~~**~~Disable~~**~~from the menu. The devices should now look like this – notice the Ethernet connection is still enabled, while the others are disabled:~~

Graphical user interface, text, application, email

Description automatically generated

1. The static IP address follows this convention: **10.TE.AM.1**, so we would be **10.15.12.1**
2. Application

   Description automatically generated with medium confidenceLaunch the **FRC Radio Configuration Utility** app:
3. Allow the program to make any changes, if prompted.
4. Select “Ethernet” or “Local Network Connection” in the drop-down menu, as shown below. (If this is not an option, click the Refresh Button.) Then click OK.

Graphical user interface, text

Description automatically generated

1. You probably do **not** need to update the radio’s firmware -- unless the radio is an old one (before 2019). If you do, see the steps here: <https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-3/radio-programming.html#open-mesh-firmware-note>
2. Configure the following settings as shown:
   * + 1. Team Number: 1512
       2. Robot Name: RedA, RedB, RedC, RedTrainer, etc. (no spaces!)
       3. Radio: OpenMesh
       4. Mode: 2.4GHz Access Point
       5. BW Limit: checked

Graphical user interface, application, Word

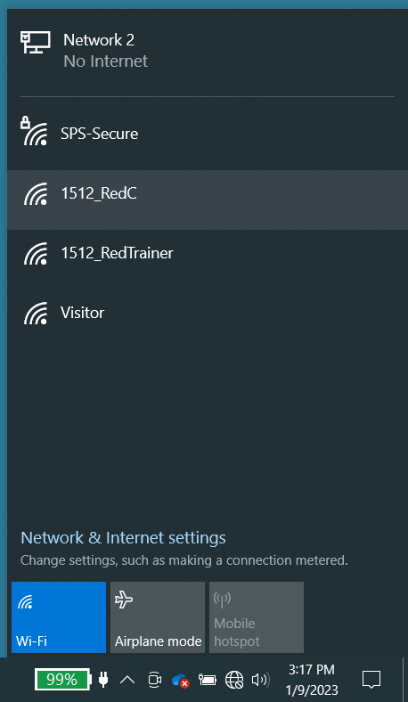
Description automatically generated

1. Then press the **Configure button**.
2. In a few seconds, the following window should appear, indicating the radio has been configured!

Graphical user interface, application

Description automatically generated

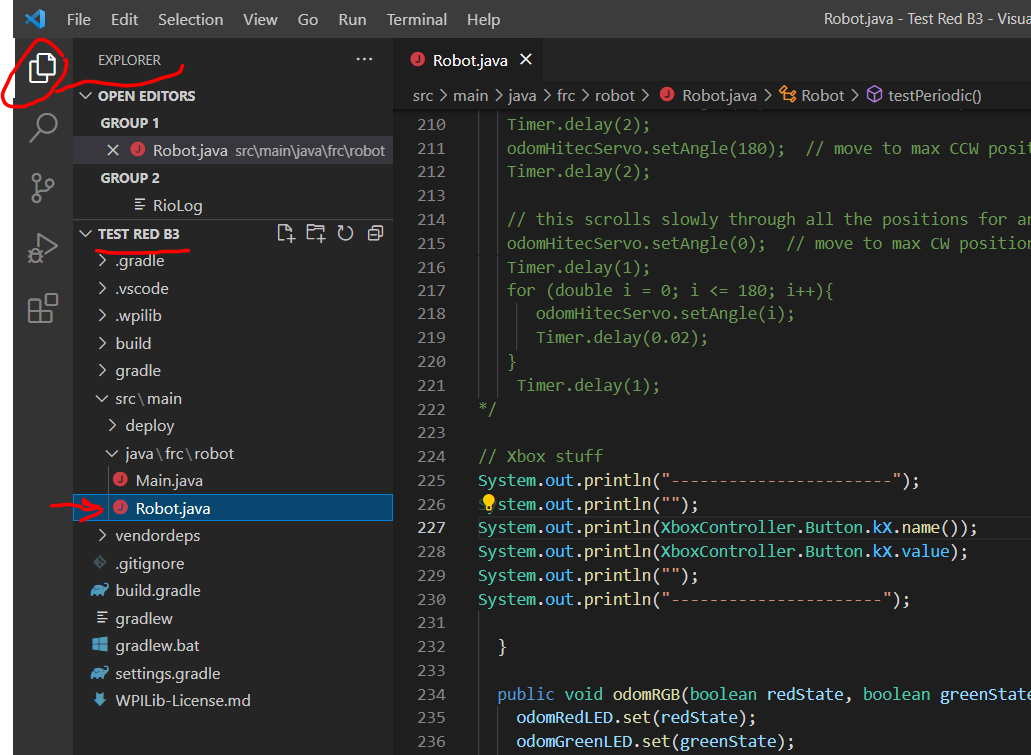
1. Press OK a couple of times and close the **Radio Configuration Utility** program.
2. Unplug the ethernet cable from your laptop and plug it back into the roboRIO.
3. Enable the Bluetooth and Wi-Fi devices once again.
4. Cycle the power to the roboRIO/Radio. Wait a few minutes… (*Thanks Dr. Justinvil!*)
5. You can check if the radio is configured properly by checking the wi-fi network (the World Icon) as shown by the yellow circle in the image below. Notice there happens to be two radios for Team 1512 that you can choose to connect to (RedC and RedTrainer) depending on which robot you want to control!



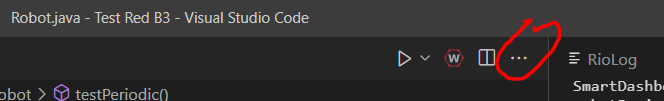
1. You are now ready to program your robot!
2. **Install Phoenix Tuner for CAN devices:**
3. Troubleshooting Sticky Faults
4. To configure CTRE CAN devices such as the PCM and PDP, use [CTRE Phoenix Tuner](https://docs.ctre-phoenix.com/en/stable/ch03_PrimerPhoenixSoft.html" \l "what-is-phoenix-tuner" \t "_blank):
   1. <https://docs.ctre-phoenix.com/en/stable/ch03_PrimerPhoenixSoft.html#what-is-phoenix-tuner>
5. Phoenix-Tuner is the graphical interface that allows for configuration of Phoenix CAN bus devices.
6. Now you can drive your motors and collect data *without writing any software*.
   1. You are now ready to program CAN devices on your robot!

## How to Write a Robot Program in Visual Studio

1. **After you have installed VS Code (Visual Studio IDE) (See Install WPILib section)**
   1. Here are some VS Code tips:
      1. <https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-2/step-2-next-steps.html>
   2. How to code in JAVA/C++:
      1. <https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-4/creating-benchtop-test-program-cpp-java.html>
   3. Creating a new WPILib Project
      1. Open Visual Studio
      2. CTRL+Shift+P
      3. WPILib: Create a New Project
      4. Click on “Select Project Type” button and choose **Example**
      5. Click on “Select Language” button and choose **Java**
      6. Click on “Select Project Base” button and choose **Getting Started**
      7. Base Folder:
         * Click on Select New Folder button
         * DO NOT USE ONEDRIVE to store CODE!!!
         * Ours is C:\Users\robotadmin\FRC Files\Code\20xxx
      8. Project Name:
         * Test Red B 3
      9. Create a New Folder = YES
      10. Team Number = 1512
      11. Click on Generate Project Button
   4. The only way I know how to open an old file that is not in Recents, is to copy the old code and paste it into the new project. The code files are in src > main > java > frc > robot > Robot.java.
   5. See Code Bible below on code samples for most FRC robot functions!
   6. How to upload code to the robot:
      1. Turn on the roboRIO and allow the radio a few minutes to boot up.
      2. Connect your laptop’s Wi-Fi to the robot’s radio network.
      3. Open Visual Studio (VS).
      4. View the java code by opening the left menu: Explorer >> Project Name >> src\main >> java\frc\robot >> Robot.java (see below)



* + 1. (*Optional*) Build the code to compile it by clicking on the dead snowman and selecting **Build Robot Code**.



* + 1. Download the code to the robot’s roboRIO by pressing **Shift+F5** or by clicking on the dead snowman and selecting **Deploy Robot Code**.
       - The laptop must be connected to the robot’s radio for this step to work!
    2. Open the **FRC Driver Station** and click on the **Enable** button to run the code. Press the **Disable** button to stop its execution.
  1. See above.
  2. Build code.
  3. Deploy Code.
     1. Which ever radio you are attached to is the roboRIO that you will send code to!
  4. Drive Station
     1. Turn on Prints
     2. Enable

## Set up the Venom CAN Motors

1. **Install Phoenix Tuner for CAN devices:**
2. Troubleshooting Sticky Faults
3. To configure CTRE CAN devices such as the PCM and PDP, use [CTRE Phoenix Tuner](https://docs.ctre-phoenix.com/en/stable/ch03_PrimerPhoenixSoft.html" \l "what-is-phoenix-tuner" \t "_blank):

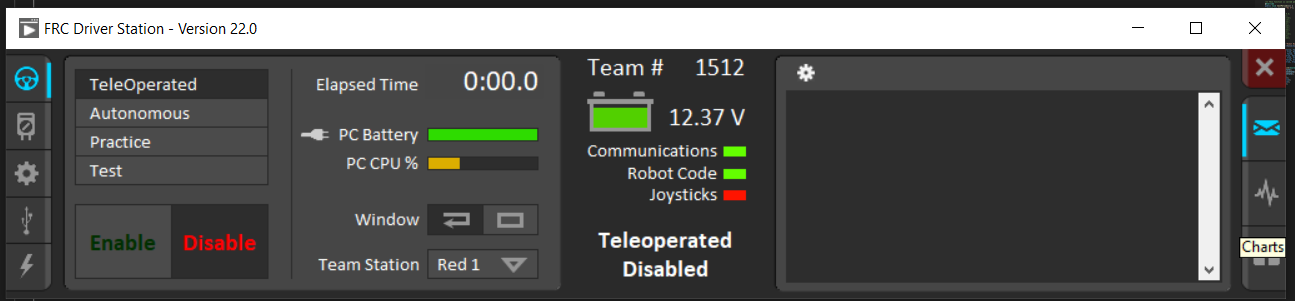
<https://www.playingwithfusion.com/productview.php?pdid=99>

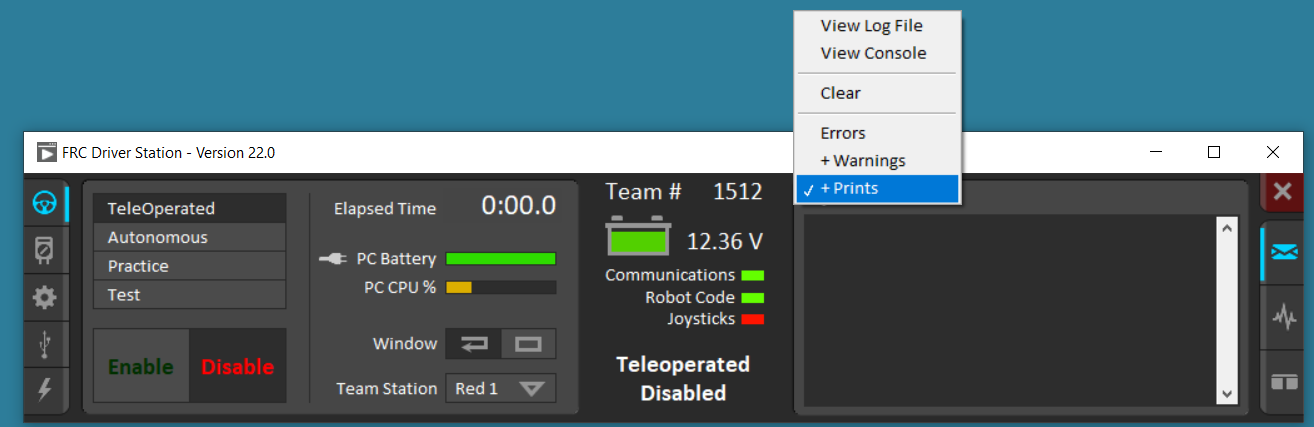
<https://www.playingwithfusion.com/docview.php?docid=1205>

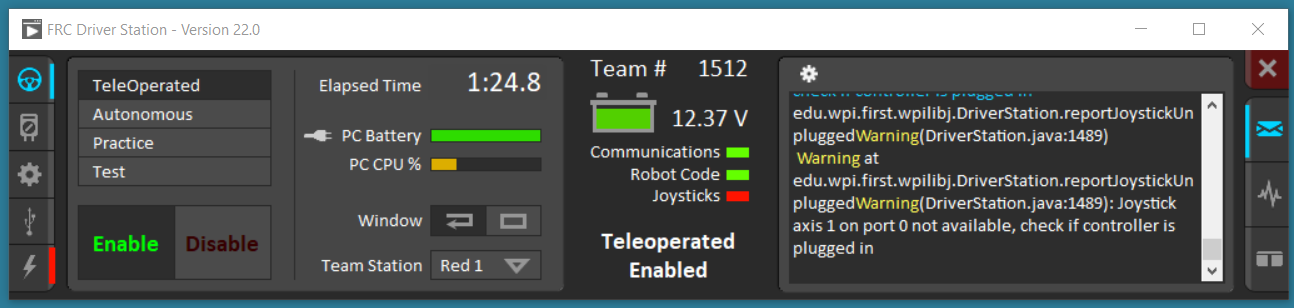
## FRC Code Bible in Visual Studio

### Driver Station Modes

* + 1. TeleOperated
    2. Autonomous
    3. Practice
    4. Test
    5. Initialization
       1. Runs once
    6. Periodic
       1. Runs constantly; once every 20ms or so.
    7. X







### Standard Default Imports:

**import edu.wpi.first.wpilibj.Joystick;**

**import edu.wpi.first.wpilibj.TimedRobot;**

**import edu.wpi.first.wpilibj.Timer;**

**import edu.wpi.first.wpilibj.drive.DifferentialDrive;**

**import edu.wpi.first.wpilibj.motorcontrol.PWMSparkMax;**

### Suggested Additional Imports:

**import edu.wpi.first.wpilibj.DigitalOutput;**

**import edu.wpi.first.wpilibj.DigitalInput;**

**import edu.wpi.first.wpilibj.AnalogInput;**

**import edu.wpi.first.wpilibj.Servo;**

**import edu.wpi.first.wpilibj.XboxController;**

**import com.playingwithfusion.CANVenom; // for Venom motors**

**import com.playingwithfusion.CANVenom.ControlMode; // for Venom motors**

### Standard Default Constructors:

**public class Robot extends TimedRobot {**

**private final PWMSparkMax m\_leftDrive = new PWMSparkMax(0);**

**private final PWMSparkMax m\_rightDrive = new PWMSparkMax(1);**

**private final DifferentialDrive m\_robotDrive = new  
 DifferentialDrive(m\_leftDrive, m\_rightDrive);**

**private final Joystick m\_stick = new Joystick(0);**

**private final Timer m\_timer = new Timer();**

**...**

**}**

### Suggested Additional Constructors:

**public class Robot extends TimedRobot {**

**DigitalInput odomButton = new DigitalInput(0); // DIO pin 0**

**DigitalOutput odomLED = new DigitalOutput(1); // DIO pin 1**

**AnalogInput odomIRRanger = new AnalogInput(0); // analog pin 0**

**DigitalOutput odomServo = new DigitalOutput(9); // servo on DIO 9 (not PWM!!!)**

**Servo odomSpringRCServo = new Servo(9); // SpringRC continuous on PWM pin 9**

**Servo odomHitecServo = new Servo(8);**

**Private final XBoxController odomXbox = new XboxController;**

**CANVenom canMotor1 = new CANVenom(1); // argument is motor ID number. get this  
 by web browser: 10.15.12.2:5812 ...**

**}**

### Printing

* + - System.out.print
    - System.out.println
    - Concatenation
* **System.out.println("testPeriodic called for odom!" + Timer.getMatchTime());**
* **System.out.println("buttonValue = " + buttonValue);**

### Timer methods

* + - Imports

**import edu.wpi.first.wpilibj.Timer;**

* + - Delay:

**Timer.delay(2.5); // delay for this many SECONDS -- you'll get warnings**

**// b/c the code can't update as expected; that’s OK!**

* + - Robot Running Time:

**// the robot running time:**

**double roborRunTime = Timer.getFPGATimestamp();**

**System.out.println("roborRunTime = " + roborRunTime);**

* + - Match Time:

**System.out.println("Match Time = " + Timer.getMatchTime() );**

### Basic Digital Output Devices (LEDs, Buzzers, etc.)

* + - Imports

**import edu.wpi.first.wpilibj.DigitalOutput;**

* + - Constructor:

**public class Robot extends TimedRobot {**

**...**

**DigitalOutput odomLED = new DigitalOutput(1); // DIO pin 1**

**...**

**}**

* + - Code:

**public void testPeriodic() {**

**...**

**odomLED.set(true); // turn LED on**

**Timer.delay(1); // delay for 1 second**

**odomLED.set(false); // turn LED off**

**Timer.delay(0.25); // delay for 0.25 seconds**

**...**

**}**

### Basic Digital Input Devices (Buttons, etc.)

* + - Imports

**import edu.wpi.first.wpilibj.DigitalInput;**

* + - Constructor:

**public class Robot extends TimedRobot {**

**...**

**DigitalInput odomButton = new DigitalInput(0); // DIO pin 0**

**...**

**}**

* + - Code:

**public void testPeriodic() {**

**...**

**// read and print the state of an external button board**

**boolean buttonValue = odomButton.get();**

**System.out.println("buttonValue = " + buttonValue);**

**...**

**}**

### Basic Analog Input Devices (Specifically for Sharp IR Ranger)

* + - Manual
      * See <https://docs.wpilib.org/en/stable/docs/hardware/sensors/analog-inputs-hardware.html>
      * 12-bit resolution (0-4096)
      * Use **\*.setAverageBits()** to average analog readings
        1. Say argument is 4, so 4-bit averaging, which means 16 integer samples will be added together and averaged.
      * Typical values for raw (4-bit averaged) readings:
        1. 17-61 with nothing in front of the sensor range
        2. ~ 2400 with flat object at 5cm
        3. ~ 2000 with flat object at 10cm
        4. ~ 930 with flat object at 25cm
        5. ~ 515 with flat object at 50cm
        6. ~ 350 with flat object at 75cm
        7. ~ 200 with flat object at 80cm
    - Imports

**import edu.wpi.first.wpilibj.AnalogInput;**

* + - Constructor:

**public class Robot extends TimedRobot {**

**...**

**AnalogInput odomIRranger = new AnalogInput(0); // Analog pin 0**

**...**

**}**

* + - Code Example #1:

**public void testPeriodic() {**

**...**

**// read and print the values read by the Sharp IR Ranger Finder**

**int rawIRValue = odomIRranger.getValue();**

**System.out.println("rawIRValue = " + rawIRValue);**

**...**

**}**

* + - Code Example #2:

**public void testPeriodic() {**

**...**

**// set the number of averaging bits to 4 (this is OPTIONAL!):**

**odomIRranger.setAverageBits(4); // adds 16 samples and averages them**

**// read and print the values read by the Sharp IR Ranger Finder**

**int rawIRValue = odomIRranger.getValue();**

**System.out.println("rawIRValue = " + rawIRValue);**

**// beep if object gets too close (2000 = 10cm)!**

**if (rawIRValue > 2000) {**

**odomBuzzer.set(true);**

**}**

**else {**

**odomBuzzer.set(false);**

**}**

**...**

**}**

### Xbox Controller

* + - Imports

**import edu.wpi.first.wpilibj.XboxController;**

* + - Constructor:

**public class Robot extends TimedRobot {**

**...**

**// 0 is the USB Port to be used as indicated on the Driver Station**

**XBoxController odomXbox = new XboxController(0);**

**...**

**}**

* + - Syntax
      * <https://first.wpi.edu/wpilib/allwpilib/docs/release/java/edu/wpi/first/wpilibj/XboxController.html>
    - Xbox Controller
      * Put tiny switch on back to X (**not** D), otherwise the assignments are wacky (X=A, Y=Y, A=B, and B=X)
      * Inputs:
        1. 6 axis switches (double values between -1.0 and 1.0 for 2 joysticks (x and y) and values between 0.0 and 1.0 for 2 triggers)
        2. 9 POV positions (integer values in degrees starting at 12:00 and moving clockwise in 45-degree increments)
        3. 10 buttons (Boolean values).
        4. See names and indices below:



**POV**

135°

225°

45°

315°

90°

270°

0°

180°

-1

Back = 7

Start = 8

Y = 4

X = 3

B = 2

A = 1

L\_Bumper = 5

L\_Trigger [0.0-1.0]

R\_Bumper = 6

R\_Trigger [0.0-1.0]

L\_Joy\_Button = 9

L\_Joy\_X & Y [-1.0 to 1.0]

R\_Joy\_Button = 10

R\_Joy\_X & Y [-1.0 to 1.0]

* + - Driver Station
      * Connect Xbox controller to USB port.
      * In Driver Station, see USB tab (see below) and note joystick, buttons, and POV positions.

A screenshot of a computer

Description automatically generated with medium confidence

* + - Code Example #1 (Brute force printing Xbox buttons, POV, and axis switches):

**public void testPeriodic() {**

**...**

**// brute force methods for the ten Xbox buttons (boolean)**

**System.out.println(odomXbox.getAButton()); // true when GREEN A depressed**

**System.out.println(odomXbox.getBButton()); // true when RED B depressed**

**System.out.println(odomXbox.getXButton()); // true when BLUE X depressed**

**System.out.println(odomXbox.getYButton()); // true when YELLOW Y depressed**

**System.out.println(odomXbox.getLeftBumper()); // true L Bumper depressed**

**System.out.println(odomXbox.getRightBumper()); // true R Bumper depressed**

**System.out.println(odomXbox.getLeftStickButton()); // true L Joystick depressed**

**System.out.println(odomXbox.getRightStickButton()); // true R Joystick depressed**

**System.out.println(odomXbox.getBackButton()); // true when Back depressed**

**System.out.println(odomXbox.getStartButton()); // true when Start depressed**

**// brute force methods for Xbox the nine POV positions**

**// (int as degrees, -1 as default unused)**

**// (int) get POV value (default = -1, then in degrees from 0 to 315**

**// in 45 degree increments running CW from 12 o'clock on the face.)**

**System.out.println(odomXbox.getPOV());**

**// brute force methods for four Xbox axis switches (joysticks and triggers)**

**// (double) data type**

**System.out.println(odomXbox.getLeftX()); // L joystick X [-1 to 1]**

**System.out.println(odomXbox.getLeftY()); // L joystick Y [-1 to 1]**

**System.out.println(odomXbox.getRightX()); // R joystick X [-1 to 1]**

**System.out.println(odomXbox.getRightY()); // R joystick Y [-1 to 1]**

**System.out.println(odomXbox.getLeftTriggerAxis()); // L trigger [0 to 1]**

**System.out.println(odomXbox.getRightTriggerAxis()); // R trigger [0 to 1]**

**...**

**}**

* + - Code Example #2 (Brute force printing Xbox buttons, POV, and axis switches):

**public void testPeriodic() {**

**...**

**// Held vs Pressed vs Released for buttons:**

**// 1. true when RED B button is being depressed:**

**System.out.println(odomXbox.getBButton());**

**// 2. Only true when RED B button is initially pressed:**

**System.out.println(odomXbox.getBButtonPressed() );**

**// 3. Only true when RED B button is released:**

**System.out.println(odomXbox.getBButtonReleased() );**

* + - Code Example #3 (two ways to act on a button press):

**public void testPeriodic() {**

**...**

**// two ways to act on a button press and beep the piezo buzzer**

**// if the B button (index #2 is) is pressed:**

**odomBuzzer.set(odomXbox.getBButton());**

**odomBuzzer.set(odomXbox.getRawButton(2));**

* + - Code Example #4 (Button and Axis index map for easier polling):

**public void testPeriodic() {**

**...**

**// button index map for easier polling:**

**// button indexes begin at 1 not 0 in WPILib for Java and C++!**

**// But regular arrays start at 0, so be careful!**

**String[] buttonName = {"n/a", "A", "B", "X", "Y", "LB", "RB", "Back",   
 "Start", "LJoy", "RJoy"};**

**for (int i=1; i < 11; i++) {**

**System.out.println(i + ". " + buttonName[i]+ " = " +   
 odomXbox.getRawButton(i));**

**if(odomXbox.getRawButton(i)) {**

**System.out.println("The " + buttonName[i] + " has been pressed!");**

**}**

**}**

**System.out.println("----------------------");**

**// axis index map for easier polling:**

**// great diagnostic! NB: at neutral position, joysticks not always (0,0)!**

**String[] axisName = {"Lx", "Ly", "Ltrig", "Rtrig", "Rx", "Ry"};**

**for (int i=0; i < 6; i++) {**

**System.out.println(i + ". " +axisName[i] + "=" + odomXbox.getRawAxis(i));**

**}**

**System.out.println("");**

**System.out.println("----------------------");**

**Timer.delay(2); // use this to delay so you can see the print out**

### Servomotors (Modified (SpringRC, e.g.) and Unmodified (Hitec, e.g.))

* + - Manual
      * **Be careful that you do NOT connect PWM devices to PWM Pins 0 and 1!** The pins for the auto generated PWMSpark motor drivers already use those pins by default!
      * By default [2.4](https://first.wpi.edu/wpilib/allwpilib/docs/release/java/edu/wpi/first/wpilibj/Servo.html#kDefaultMaxServoPWM) ms is used as the maxPWM value
      * By default [0.6](https://first.wpi.edu/wpilib/allwpilib/docs/release/java/edu/wpi/first/wpilibj/Servo.html#kDefaultMinServoPWM) ms is used as the minPWM value
    - Imports

**import edu.wpi.first.wpilibj.Servo;**

* + - Constructor:

**public class Robot extends TimedRobot {**

**...**

**Servo odomSRCServo = new Servo(8); // 360-degree SpringRC on PWM pin 8**

**Servo odomHitecServo = new Servo(9); // 180-degree Hitec on PWM pin 9**

**...**

**}**

* + - Code Example #1:

**public void testPeriodic() {**

**...**

**// Code for continuous rotation SpringRC servo using the .set() method:**

**odomSRCServo.set(0); // max CW speed**

**Timer.delay(1); // give servo time to move**

**odomSRCServo.set(0.5); // stop**

**Timer.delay(0.5); // pause time**

**odomSRCServo.set(1); // max CCW speed**

**Timer.delay(1); // give servo time to move**

**odomSRCServo.set(0.5); // stop**

**Timer.delay(0.5); // pause time**

**...**

**}**

* + - Code Example #2:

**public void testPeriodic() {**

**...**

**// Code for continuous rotation SpringRC servo using the .setAngle() method**

**// spoiler alert -- it does the same thing, just with different arguments**

**odomSRCServo.setAngle(0); // max CW speed**

**Timer.delay(1); // give servo time to move**

**odomSRCServo.setAngle(90); // stop**

**Timer.delay(0.5); // pause time**

**odomSRCServo.setAngle(180); // max CCW speed**

**Timer.delay(1); // give servo time to move**

**odomSRCServo.setAngle(90); // stop**

**Timer.delay(0.5); // pause time**

**...**

**}**

* + - Code Example #3:

**public void testPeriodic() {**

**...**

**// code for unmodified 180-degree servos is practically the same as modified**

**// 360-degree servos, but watch the timing:**

**odomHitecServo.setAngle(0); // move to max CW position and stay there**

**Timer.delay(2);**

**odomHitecServo.setAngle(90); // move to middle position and stay there**

**Timer.delay(2);**

**odomHitecServo.setAngle(180); // move to max CCW position and stay there**

**Timer.delay(2);**

**// this scrolls slowly through all the positions for an unmodified servo**

**odomHitecServo.setAngle(0); // move to max CW position and stay there**

**Timer.delay(1);**

**for (double i = 0; i <= 180; i++){**

**odomHitecServo.setAngle(i);**

**Timer.delay(0.02);**

**}**

**...**

**}**

* + - Code Example #4:

**public void testPeriodic() {**

**...**

**// drive modified 360-degree servo with the values from the Right X Joystick**

**// of the Xbox controller:**

**odomSRCServo.setAngle(odomXbox.getRightX()); // easy!**

**...**

**}**

### PWM Spark Max Motor Control (PWM, not CAN)

* + - Imports

**import edu.wpi.first.wpilibj.motorcontrol.PWMSparkMax;**

* + - Constructor:

**public class Robot extends TimedRobot {**

**...**

**PWMSparkMax odomSpark = new PWMSparkMax(0); // PWM pin 0**

**...**

**}**

* + - Code Example #1:

**public void testPeriodic() {**

**...**

**// brute force motor code to spin CCW, stop, spin CCW, stop.**

**// Argument for .set() command is the motor speed as % output of the**

**// motor, between -1 and 1. + is CCW, - is CW, 0 = stop**

**odomSpark.set(0.25); // 25% of max, CCW spin**

**Timer.delay(1);**

**odomSpark.set(0.0); // 0% = stop**

**Timer.delay(0.2);**

**odomSpark.set(-0.25); // 25% of max, CW spin**

**Timer.delay(1);**

**odomSpark.set(0.0); // 0% = stop**

**Timer.delay(0.2);**

**...**

* + - Code Example #2:

**public void testPeriodic() {**

**...**

**// control the motor with the Xbox controller (Left Joystick Y):**

**odomSpark.set(odomXbox.getLeftY());**

**...**

**}**

### Venom Motor with CAN Communication

* + - Imports

**import xxx**

* + - Constructor:

**public class Robot extends TimedRobot {**

**...**

**xxx**

**...**

**}**

* + - Code Example #1:

**public void testPeriodic() {**

**...**

**xxx**

* + - Code Example #2:

**public void testPeriodic() {**

**...**

**xxx**

**...**

**}**

**Using xbox controller to drive robot:**

|  |
| --- |
| // Copyright (c) FIRST and other WPILib contributors. |
|  | // Open Source Software; you can modify and/or share it under the terms of |
|  | // the WPILib BSD license file in the root directory of this project. |
|  |  |
|  | package edu.wpi.first.wpilibj.examples.arcadedrivexboxcontroller; |
|  |  |
|  | import edu.wpi.first.wpilibj.TimedRobot; |
|  | import edu.wpi.first.wpilibj.XboxController; |
|  | import edu.wpi.first.wpilibj.drive.DifferentialDrive; |
|  | import edu.wpi.first.wpilibj.motorcontrol.PWMSparkMax; |
|  |  |
|  | /\*\* |
|  | \* This is a demo program showing the use of the DifferentialDrive class. Runs the motors with split |
|  | \* arcade steering and an Xbox controller. |
|  | \*/ |
|  | public class Robot extends TimedRobot { |
|  | private final PWMSparkMax m\_leftMotor = new PWMSparkMax(0); |
|  | private final PWMSparkMax m\_rightMotor = new PWMSparkMax(1); |
|  | private final DifferentialDrive m\_robotDrive = new DifferentialDrive(m\_leftMotor, m\_rightMotor); |
|  | private final XboxController m\_driverController = new XboxController(0); |
|  |  |
|  | @Override |
|  | public void robotInit() { |
|  | // We need to invert one side of the drivetrain so that positive voltages |
|  | // result in both sides moving forward. Depending on how your robot's |
|  | // gearbox is constructed, you might have to invert the left side instead. |
|  | m\_rightMotor.setInverted(true); |
|  | } |
|  |  |
|  | @Override |
|  | public void teleopPeriodic() { |
|  | // Drive with split arcade drive. |
|  | // That means that the Y axis of the left stick moves forward |
|  | // and backward, and the X of the right stick turns left and right. |
|  | m\_robotDrive.arcadeDrive(-m\_driverController.getLeftY(), -m\_driverController.getRightX()); |
|  | } |
|  | } |

### Basic Digital Input Devices (Buttons, etc.)

* + - Imports

**import edu.wpi.first.wpilibj.DigitalInput;**

* + - Constructor:

**public class Robot extends TimedRobot {**

**...**

**DigitalInput odomButton = new DigitalInput(0); // DIO pin 0**

**...**

**}**

* + - Code:

**public void testPeriodic() {**

**...**

**// read and print the state of an external button board**

**boolean buttonValue = odomButton.get();**

**System.out.println("buttonValue = " + buttonValue);**

**...**

**}**

### Gryoscopes and Accelerometers

[Accelerometers - Software — FIRST Robotics Competition documentation (wpilib.org)](https://docs.wpilib.org/en/stable/docs/software/hardware-apis/sensors/accelerometers-software.html)

* + - Imports

**import ~~edu.wpi.first.wpilibj.DigitalInput;~~**

* + - Constructor:

**public class Robot extends TimedRobot {**

**...**

**~~DigitalInput odomButton = new DigitalInput(0); // DIO pin 0~~**

**...**

**}**

* + - Code:

**public void testPeriodic() {**

**...**

**~~// read and print the state of an external button board~~**

**~~boolean buttonValue = odomButton.get();~~**

**~~System.out.println("buttonValue = " + buttonValue);~~**

**...**

**}**

### Creating methods/functions

**public class Robot extends TimedRobot {**

**...**

**private void odomBlinkLED(int numBlinks, double onTime, double offTime){**

**for (int i = 0; i < numBlinks; i++) {**

**odomLED.set(true);**

**Timer.delay(onTime);**

**odomLED.set(false);**

**Timer.delay(offTime);**

**}**

**}**

**}**

* 1. CAN
  2. Venom
  3. Robot arcade drive
  4. Camera
  5. PWMSparkMax
  6. Joystick
  7. Pneumatic

### Advanced Digital Output Devices (Servos, etc.)

* + - Constructor:

**public class Robot extends TimedRobot {**

**...**

**DigitalOutput odomServo = new DigitalOutput(9); // DIO 9 (not PWM!!!)**

**...**

**}**

* + - Code:

**public void testPeriodic() {**

**...**

**odomLED.set(true); // turn LED on**

**Timer.delay(1); // delay for 1 second**

**odomLED.set(false); // turn LED off**

**Timer.delay(0.25); // delay for 0.25 seconds**

**...**

**}**

1. **3rd Party CAN Devices**
   1. It’s important that teams using 3rd-party CAN motor controllers look at the [Installing 3rd Party Libraries](https://docs.wpilib.org/en/stable/docs/software/vscode-overview/3rd-party-libraries.html#rd-party-libraries) article as extra steps are required to code for these devices.
      1. <https://docs.wpilib.org/en/stable/docs/software/vscode-overview/3rd-party-libraries.html#rd-party-libraries>

## Wiring Schematics

* 1. Black and red power wiring xxx!
  2. roboRIO user manual image

## Hardware Basics

* 1. <https://docs.wpilib.org/en/stable/docs/controls-overviews/control-system-hardware.html>

## roboRIO Setups

1. Serial numbers
2. Addresses
   1. 1512 (on the training station)
      1. xxx
   2. 1512\_RedA (on the 2022-23 competition robot)
      1. Radio IP address: 10.15.12.2
      2. MAC Address: xxx
   3. RedB
      1. Radio IP address: 10.15.12.2
      2. MAC Address: 00:80:2F:30:DC:26
3. FTP information:
   1. FTP user name: “lvuser” (that is, lowercase “LVUSER”)
   2. FTP password: “” (i.e., blank)
   3. FTP port: 22
   4. FTP connection for all SPS radios: <ftp://10.15.12.2> (laptop must be on that roboRIO’s network)

## ~~Programming the Radio~~

1. ~~Xxx~~
2. ~~Update the firmware first. To do so, follow these steps to the letter:~~
   1. ~~Remove the power cord to the radio.~~
   2. ~~Using an ethernet cable, connect the radio to the laptop. Use the port that is adjacent to the power plug on the radio.~~
   3. ~~Press the “Load Firmware” button.~~
   4. ~~When the window pops open, power up the radio.~~
   5. ~~After a few moments, the radio will have installed the new firmware.~~

## Writing and Uploading Test Code to the RoboRio

1. Xxx

## CAN Venom Motor Info

1. Yellow and green CAN wires:
   1. Yellow is HIGH
   2. Green is LOW
   3. Originate on the roboRIO.
      1. This port includes the necessary
   4. Connection must terminate on the Main Power Distribution bus (“Term/Res On Off”)!!!
2. Motor ID:
   1. To find motor ID, open a web browser, enter the IP roboRIO, then “:[teamName]”. E.g., RedA would be: “10.15.12.2:2812”
   2. Select PWF Device Configuration
   3. Update
   4. Identify
      1. Motors usually have IDs 1, 2, 3 in order of their position on the daisy chain.
   5. See flashing red and green LEDs for motor status

## Xbox Controller Info

1. Xxx

## Using Non-Proprietary Digital and Analog Sensors

1. Xxx

## Understanding the Timers of the FRC Game Tools

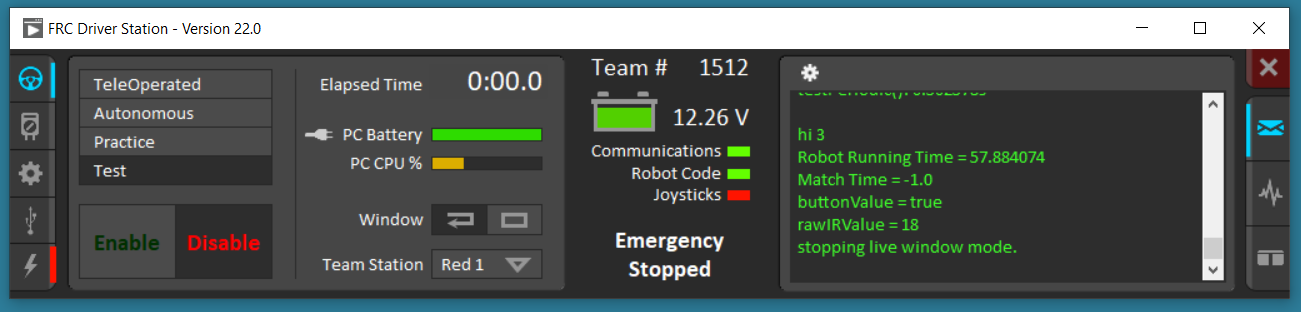
1. Xxx

## GitHub SPS Details

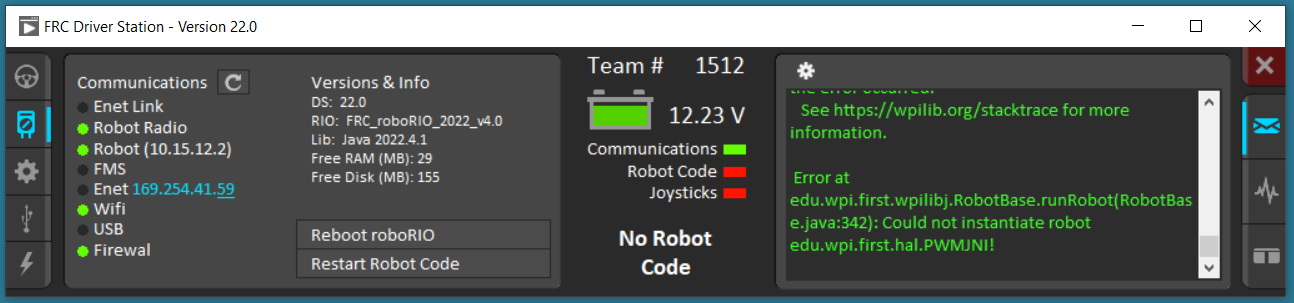
1. Xxx

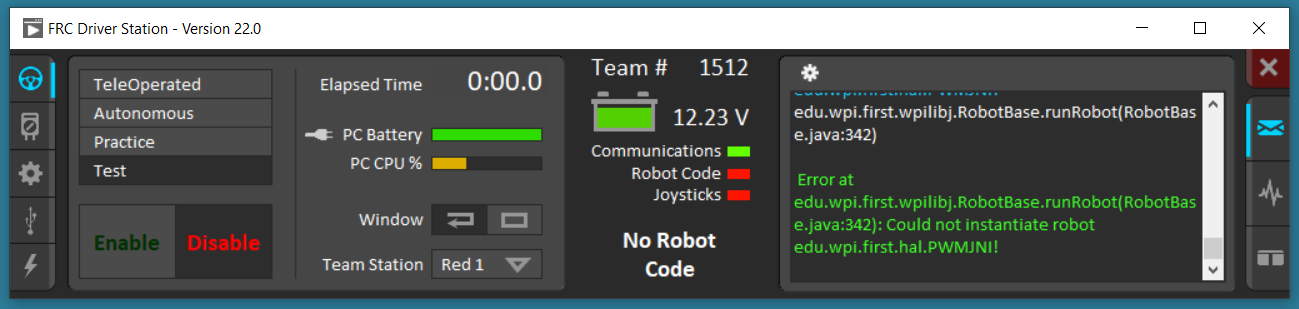
## Emergency Stops, No Robot Code, and Sticky Faults Errors

1. If you unplug something while the roboRIO is powered or your code is bad, you are likely to encounter an **Emergency Stop**, **No Robot Code**, or a **Sticky Fault** error. These are the bane of my existence! If you get them, you cannot Enable the robot from the Driver Station until the fault or error has been cleared.
2. Often you can clear an **Emergency Stop** by closing the Visual Studio and Driver Station windows, and cycling the power to the roboRIO. Of course, you will also then need to reconnect to the radio after startup.



1. The **No Robot Code** error is probably due to an error in your code. For instance, I got the error below on the Driver Station because my servo constructor was bad. The trouble was I had connected my servos to pins 0 and 1, but those were already being used by the auto generated PWMSparkMax constructors code. An easy fix!





1. Clearing a **Sticky Fault** is more involved. To clear the fault, xxx