**FRC 2020 Software Setup Guide** – January 26, 2020

**Section 1 - Scope/Objectives**

* **Software components -** For C++ or Java development there is only one IDE supported by FIRST.
  + *“Microsoft’s Visual Studio Code is the new supported IDE for C++ and Java development in FRC, replacing the Eclipse IDE used from 2015-2018.”*

<https://docs.wpilib.org/en/latest/docs/software/wpilib-overview/vscode-basics.html>

A few other software components will be needed which will be listed in section 2.

* **Source Control -** With multiple software collaborators reasonable source control is needed to allow developers to work on sub-components and collaborate on the aggregate code base. For many reasons, GitHub is the obvious choice. Basic proficiency with GitHub or similar source control is a desired skill of most software companies. This document will describe the setup and most basic use of GitHub.
* **Navigation** during autonomous (and teleop?) - Nav is a large subject but can be as simple as scoring points during the autonomous period by driving forward for a short period and stopping. The next goals teams typically have involve counting drive wheel rotations with encoders, turning using angular information from a gyro and deploying a game piece into a goal. For the gyro there are two popular choices. The ADXRS453 module plugs directly into the Roborio SPI port. The NavX board is a bit more accurate and plugs directly into the Roborio MXP port. Countless other sensors, including cameras, can provide input to the nav function. This document describes the initial setup, programming and communication using the NavX to provide the gyro input.
* **Vision Processing** requires a relatively large learning curve but can be a valuable and rewarding skill to be applied to future projects. There are two popular approaches for FRC:
  + Some teams swear by *Limelight For FRC.* It seems that the most significant advantage is a smaller learning curve, but the skills learned are a bit specific. I would not invest my time.
  + The Raspberry Pi has become very popular in recent years to perform countless functions. Coupled with OpenCV, this is a very powerful vision processing platform.

*This document describes setting up the Raspberry Pi with OpenCV and communications with the RoboRio (network tables)*

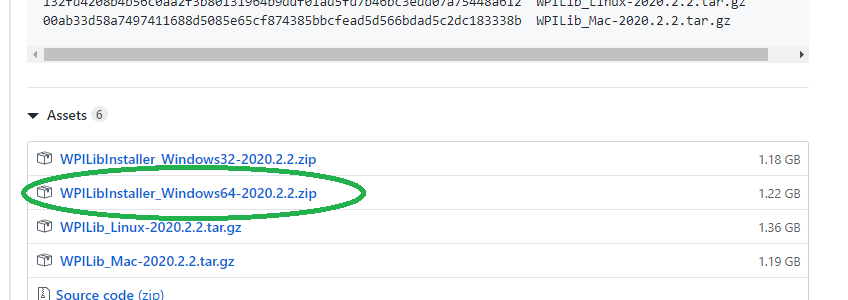
**Section 2 - VS Code, Git and Roborio Firmware - Installation and Setup**

**Step 2A: Install Visual Studio Code with C++, Java and WPILib extensions**

This step will download and install the [WPILib 2020.2.2 Update](https://github.com/wpilibsuite/allwpilib/releases/tag/v2020.2.2). This “Update” is actually the full installation of VS Code, Java and much of the FRC functionality bundled with it.

Here is the link to the download: Scroll down and download the installer zip file. Unzip it and run the installer.

<https://github.com/wpilibsuite/allwpilib/releases/tag/v2020.2.2>



Here are instructions to go with the installer:

<https://docs.wpilib.org/en/latest/docs/getting-started/getting-started-frc-control-system/wpilib-setup.html>

The installer will install these three shortcuts to your desktop:



**Step 2B: Install and Setup Git**

Download from here: Install selecting all default settings

[**https://git-scm.com/download/win**](https://git-scm.com/download/win)

1. From your browser, go to github and **create an account** with your email address.
2. **Specify your github credentials** on your computer using your github email address and password.

* Run CMD (Windows+R » type cmd » Enter)
  + git config --global user.email "you@example.com"
  + git config --global user.name "Your Name"

1. **This step is for the Git hub repository owner only:** 
   1. create an empty remote git project and copy the URL of project to clipboard. The project will be nothing but the default readme file.
   2. Add collaborators to git hub repository

at github .com settings->collaborators->add collaborator -> enter user names or email if user set up with only email

* 1. From VS Code ctrl+shift+P to open command pallet -> git: initialize repository
  2. From VS Code **ctrl**+**shift**+**`** to open terminal. *That is a backtick (below the tilde below the Esc key. Or from the menu click Terminal->new Terminal*
  3. Using your browser, go to the github repository you want to link to the VS Code project. Copy the URL of the git project to the clipboard. There is a button on the githhub site to do so or you can type it into the next step. The URL looks something like this:

*https://github.com/RussC243/WM6423\_roborio\_2020A.git*

* 1. From the terminal window in VS Code type:

*Git push --set-upstream [URL of project] master*

**Step 2C: Install FRC Game tools** (Installs the driver station and RoboRio Imaging tool)

Instructions are here:

<https://docs.wpilib.org/en/latest/docs/getting-started/getting-started-frc-control-system/frc-game-tools.html>

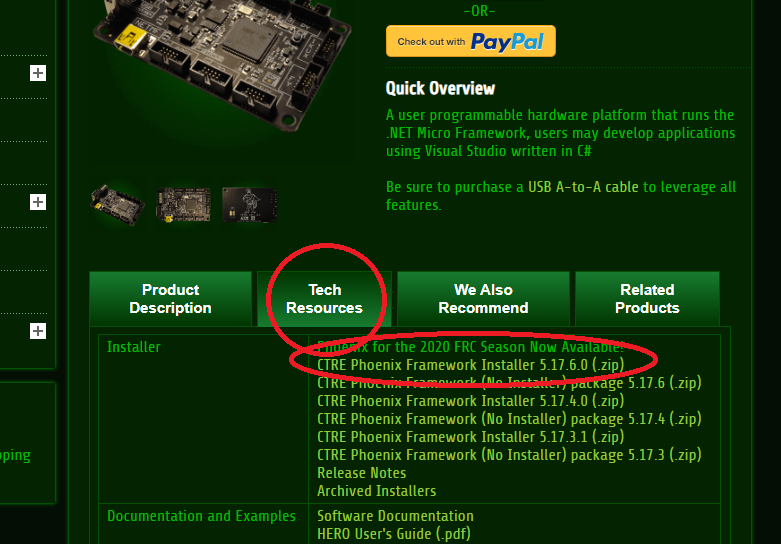
Download from here:

<https://www.ni.com/en-us/support/downloads/drivers/download.frc-game-tools.html#333285>

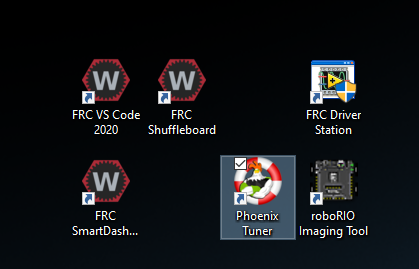
When you open the driver station, you may need to create an account with National Instruments to activate the Vision Processing Module or try clicking “Evaluate” and extend license.

**Step 2D: Install Phoenix Tuner -** (This installs the tools needed to update firmware, set can IDs for the Roborio and speed controllers.)

<https://www.ctr-electronics.com/hro.html#product_tabs_technical_resources>

****

You should now have the following shortcuts on your desktop:

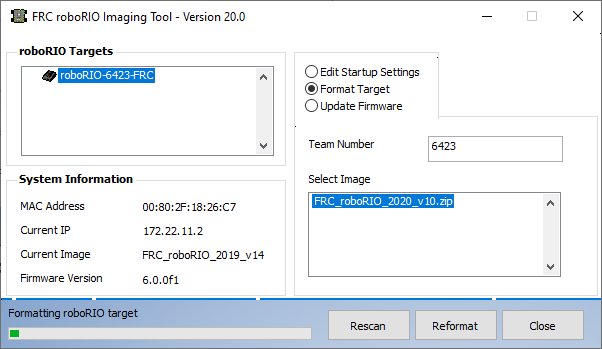
****

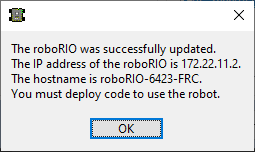
**Step 2E: Update the RoboRIO to 2020 Firmware**

If needed, detailed Instructions can be found here: <https://docs.wpilib.org/en/latest/docs/getting-started/getting-started-frc-control-system/imaging-your-roborio.html>

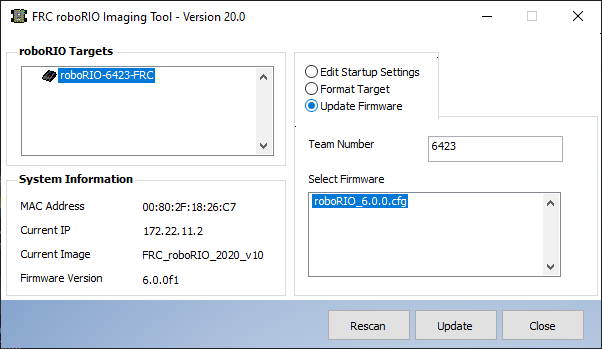
*Caution:* Disconnect the motors from the speed controllers to avoid injury.

* This procedure consists of two steps: **Format Target then Update Firmware**.
* Plug in USB cord(type A 2.0 male to type B 2.0 male) directly between laptop and RoboRIO.
* Start the RoboRIO Imaging tool, Select *Format Target* and click *Reformat*. *This will take about 10 minutes.*





* Then select *Update Firmware* and click *Update. This will take about 5 minutes*



The same message box should pop up as when you formatted the target.

**Step 2F: Set CAN IDs** of devices connected to the CAN bus and **update firmware of any speed controllers** connected to the CAN bus.

What devices are connected to CAN bus?

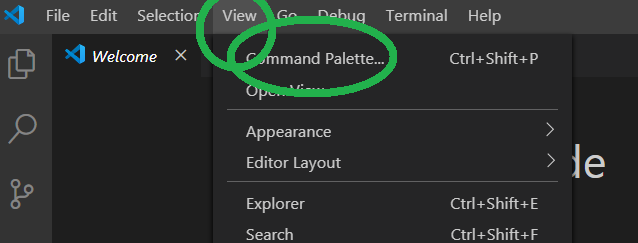
* All systems will have the PDP connected on the CAN bus
* Systems with pneumatics will have the PCM connected on the CAN bus
* Motor controllers are either PWM or CAN bus.

-Start **Phoenix Tuner** and set each device to a have a different CAN ID. Devices default to ID 0. So, if there are multiple devices that have not been changed, you must select one of the multiples and change to a higher ID or power them one at a time and change their IDs.

-If there are motor controllers connected to the CAN bus, now is a good time to update their firmware.

**Section 3: Create a VS Code project and sync it with Github.**

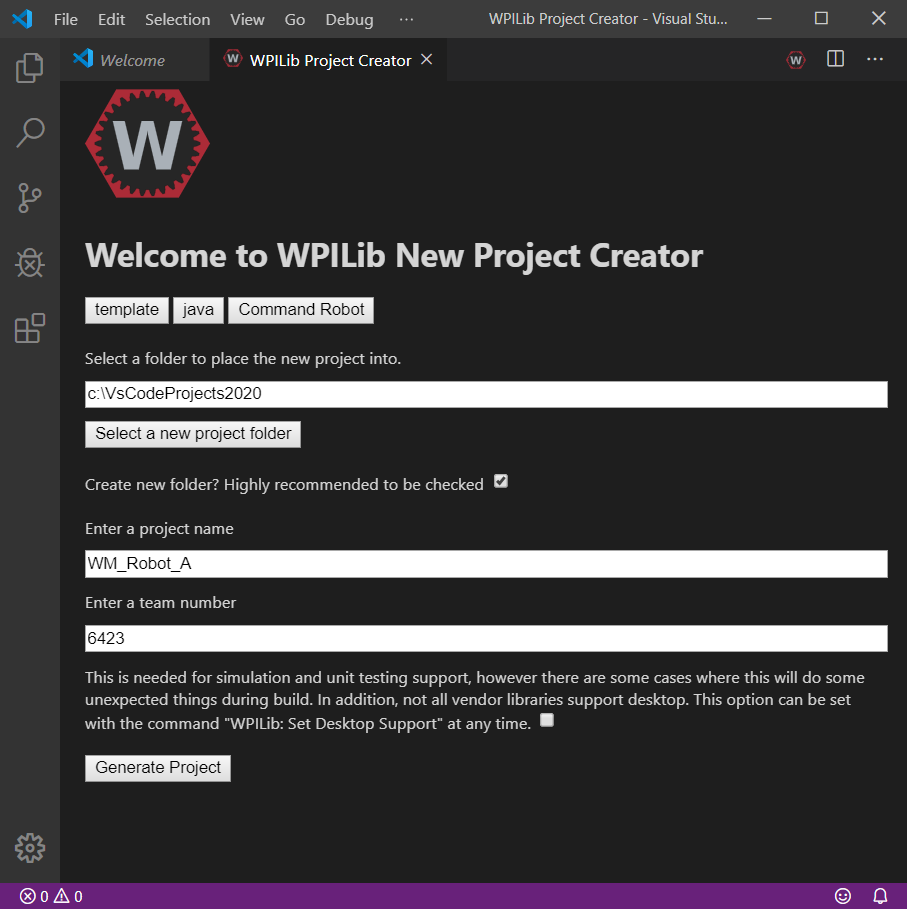
Open VS Code using desktop shortcut**.**

****

**Hint:** When you type into the Command Palette search bar, start your search string with WPI… This way the commands set up specifically by WPI show up.

### 

**Create a new project** (WPILib: Create a new Project) and select options similar to below



Click **“Generate Project”**

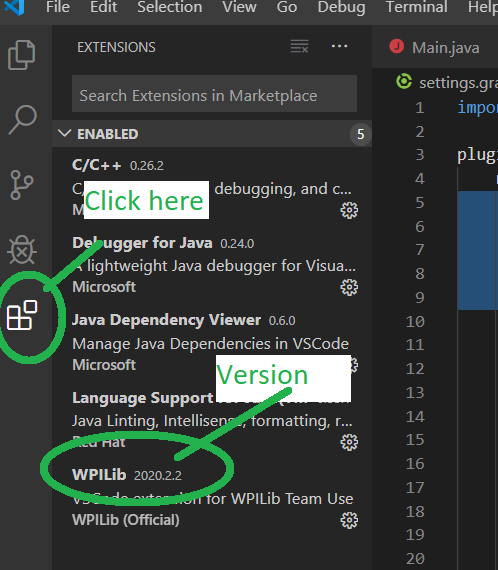
1. **This step is for everyone**

*Clone the code – Procedure is TBD*

*Click the source control icon in VS Code (Icon on left looks like the USB symbol.) What is displayed is “No Source Control Providers Registered”. Then click the + sign to add the github repo.*

* 1. Make changes to code
     1. git remote add origin <Link to GitHub Repo> //maps the remote repo link to local git repo
     2. git remote -v //this is to verify the link to the remote repo
     3. git push -u origin master // pushes the commit-ed changes into the remote repo

1. **How to check or change WPI library version** – You should have WPILib 2019.0.0-alpha-4 until beta or an official 2019 release is available.



1. **Add environment variable** so Java compiler can locate the navX-MXP library

Set name to “navx-mxp” and set the Path to <HomeDirectory>\wpilib\user\java\lib\navx\_frc.jar.

It is described here for Eclipse but the procedure is the same when using VS Code. <https://pdocs.kauailabs.com/navx-mxp/software/roborio-libraries/java/>

1. **Force Shuffleboard to start as defaul**t – It is described here:

<https://wpilib.screenstepslive.com/s/currentCS/m/24192/l/290135-setting-the-driver-station-to-start-smartdashboard>

My FRC DS Data Storage.ini file look like this:

[Setup]

TeamNumber = "6423"

DashboardType = 3

DashboardCmdLine = ""C:\\Users\\RussS9\\wpilib\\tools\\SmartDashboard.jar""

DashboardRemoteIP = "10.64."

RobotMode = 1

TeamStation = 0

WindowTop = 589

WindowLeft = 250

MoveableWindow = TRUE

[PracticeRound]

Sound Effects = TRUE

Countdown = 5

Autonomous = 150

Delay = 1

TeleOperated = 105

EndGame = 30

GameData = ""

If shuffleboard does not open from driver station or from java -jar shuffleboard.jar make sure jave jre is installed not just sdk.

**Section 3 - Raspberry Pi Development using Netbeans - Installation and Setup**

[**https://github.com/opencv/opencv/issues/10080**](https://github.com/opencv/opencv/issues/10080)

**Scope**: There are many popular hardware and software packages for vision processing on single board computers. The combination of OpenCV, Java, Raspberry Pi and NetBeans was chosen as the best balance of the following:

* Team 6423 prefers Java
* Large knowledge base and community support such that issues can be resolved easily
* Full featured, modern Integrated development environment on par with Eclipse and Visual Studio. Must have standard features such as: step through code, breakpoints, code completion, display errors as they are typed, extensive library support, easy deployment
* High level language such as C# or Java
* Ability to develop on remote machine and target single board computer
* Scalable performance by targeting more capable single board computer

1. **Install Netbeans 9.0**
   1. You should have installed [Java SE 11.0.1(LTS)](https://www.oracle.com/technetwork/java/javase/downloads/jdk11-downloads-5066655.html) from Section 2 above. If not, install it and verify java **JDK** version as **11.0.1**.
      1. Check version by opening this file in notepad or VS Code

jdk version in C:\Program Files\Java\jdk-11.0.1 release

JAVA\_VERSION="11.0.1"

JAVA\_VERSION\_DATE="2018-10-16"

* 1. Verify java **JRE** version as **1.8.0\_191**, from a command prompt, type **java – version** and verify version like this:

java **version "1.8.0\_191"**

Java(TM) SE Runtime Environment (build 1.8.0\_191-b12)

Java HotSpot(TM) Client VM (build 25.191-b12, mixed mode)

You can see the same information if you open the release file from here:

C:\Program Files (x86)\Java\jre1.8.0\_191\release

* 1. Download this binaries zip file **incubating-netbeans-java-9.0-bin.zip** from here:

<https://www.apache.org/dyn/closer.cgi/incubator/netbeans/incubating-netbeans-java/incubating-9.0/incubating-netbeans-java-9.0-bin.zip>

* 1. Unzip and execute C:\Netbeans9.0\netbeans\bin\**netbeans64.exe**. You can right click and pin to start and/or taskbar if you like.
  2. Open C:\Netbeans9.0\netbeans\etc\**netbeans.conf** with Visual Studio Code. File will not look nice and be hard to edit if opened with notepad.
     1. Towards the bottom of the file, uncomment the jdkhome configuration from this

#netbeans\_jdkhome="/path/to/jdk"

To match where your jdk is installed… like this:

netbeans\_jdkhome=" C:\Program Files\Java\jdk-11.0.1"

* 1. Install OpenCV on the PC – Netbeans needs the same version OpenCV on the PC Netbeans is running as on the Pi. Download 3.4.3 Win pack to the PC. Unzip somewhere on you r hard drive like: C:\OpenCV\_3.4.3\opencv

[**https://opencv.org/releases.html**](https://opencv.org/releases.html)

Run the exe to install OpenCV on your PC.

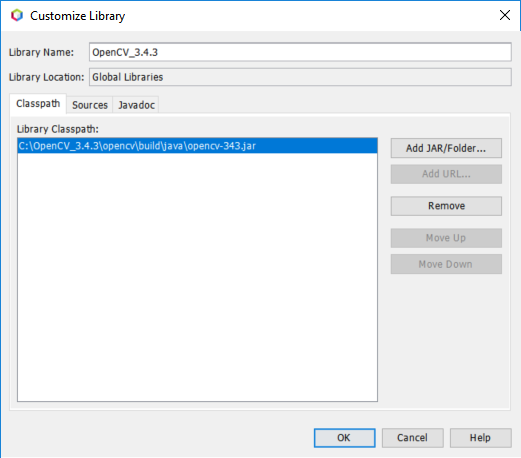
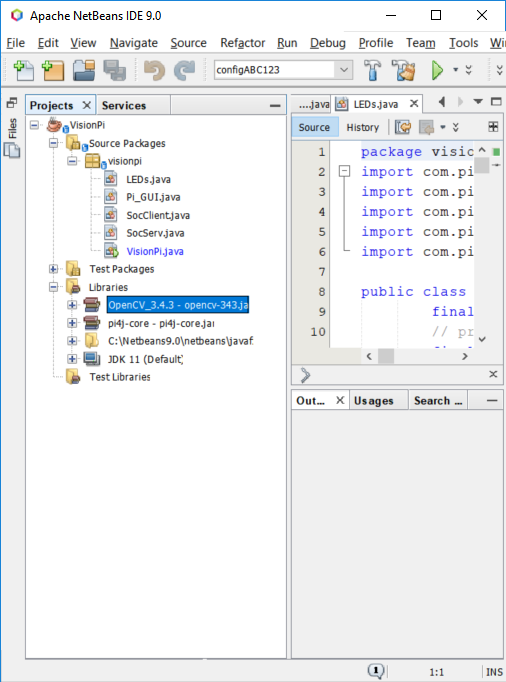
* 1. **Create new project**

My usual procedure is:

Create new Java project

Inside that project: Create a new package

Inside that package: Create a new Java class

1. Add OpenCV to library folder in netbeans (todo – show steps)
   1. Right click the libraries folder in netbeans
   2. Add library
   3. Create
   4. OK
   5. Add jar file
   6. Navigate to the jar file and select it
   7. 
   8. 
2. **Setup Netbeans to deploy and debug remotely on the Pi** – Follow this article by Richard Bound. <https://blog.idrsolutions.com/2014/08/using-netbeans-remotely-deploy-projects.-raspberry-pi/>

Some tips, potential snags are given below.

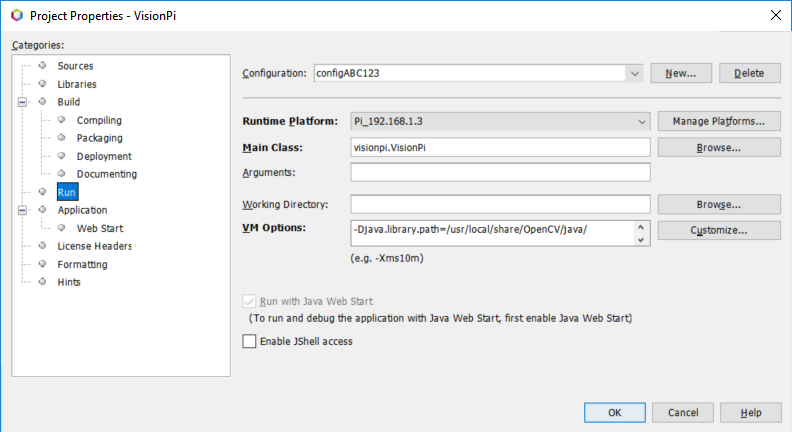
* 1. When setting up the **Remote JRE Path**, the path given in the article is not correct if the pi was set up according to Section 4 of this document. The **arm** part of the path needs to change to **arm32** and **remove the** **bin/java** part at the end. Netbeans 9 apparently appends bin/java to whatever you enter for the path. Note that it’s a path on the PI not your laptop. You can verify by opening a putty session and navigating to verify this folder exists:

/usr/lib/jvm/jdk-8-oracle-arm**32**-vfp-hflt/jre/bin/java

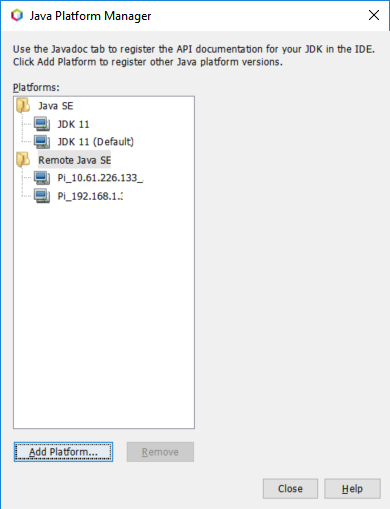
So the path you enter is the following:

**/usr/lib/jvm/jdk-8-oracle-arm32-vfp-hflt/jre**

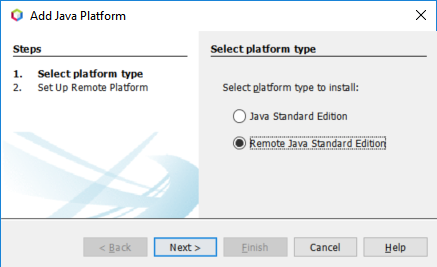
Note: When you try to paste it in, right click does not bring up the typical copy, past menu. So copy the above path with ctrl+c and paste with ctrl+v.



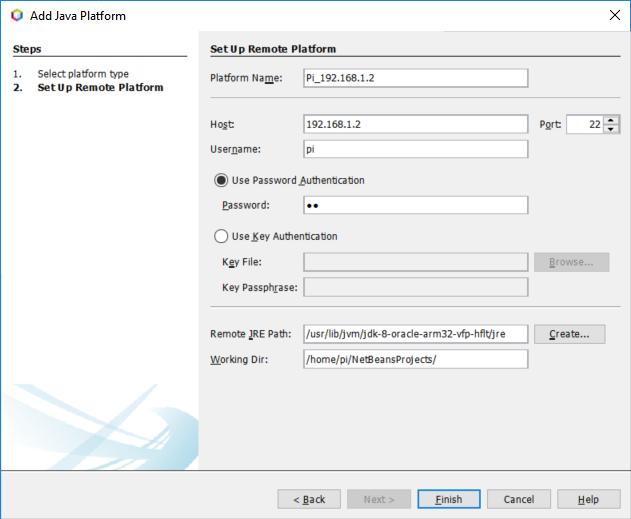




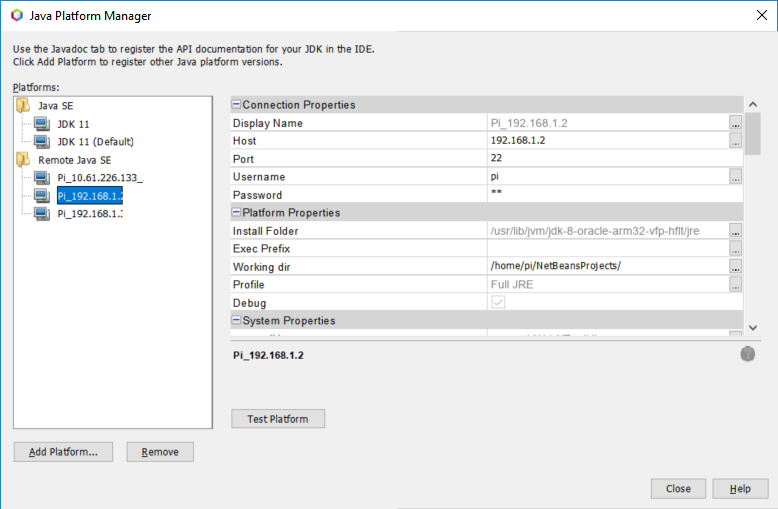




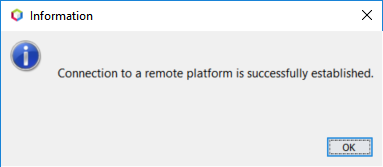


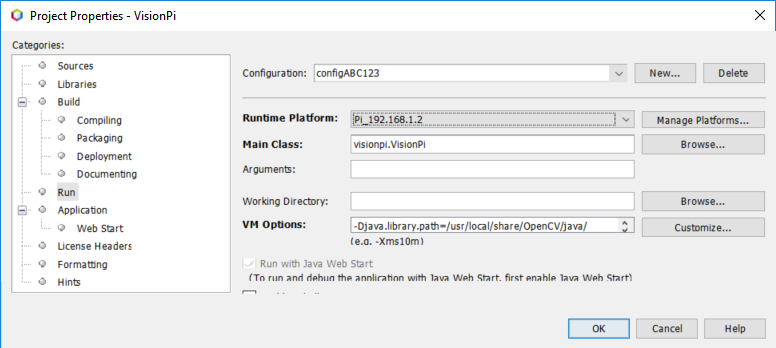










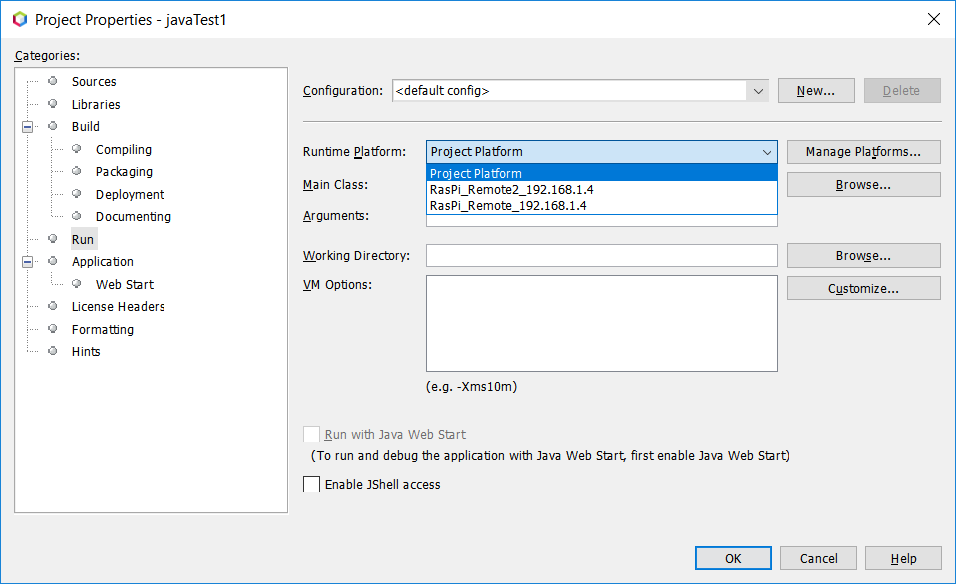




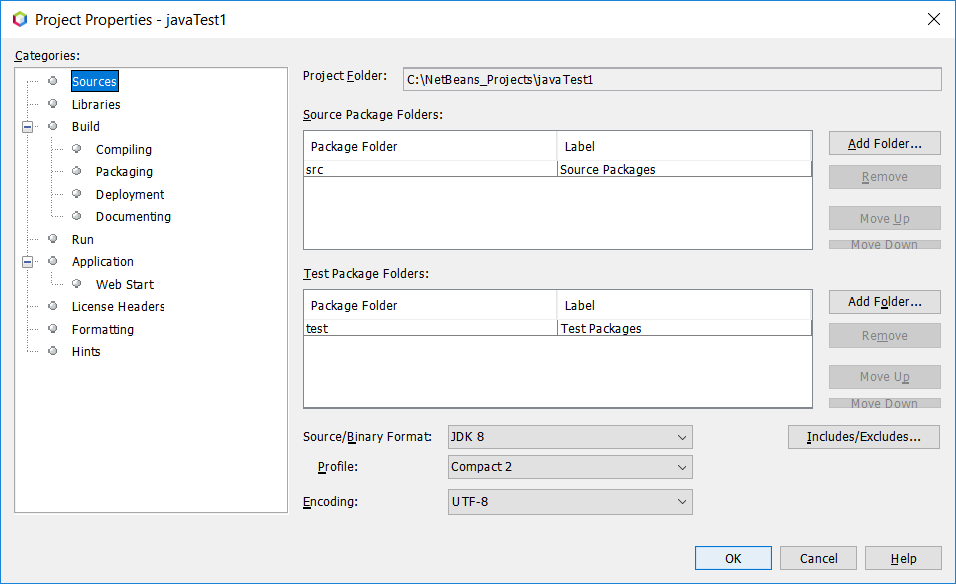
* 1. After you set up the remote platform, right click on the project-> properties->run and try to select the remote platform. If it does not show up like it does below it is probably caused by the runtime platform jre that you set up in the last step being compatible with the project that you set up. It is described here.

<https://netbeans.org/bugzilla/show_bug.cgi?id=252299>

The solution is to select Sources and select **JDK 8**, and **compact level 2** as shown below

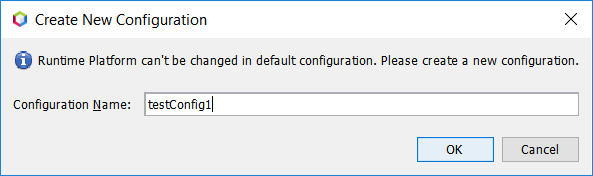








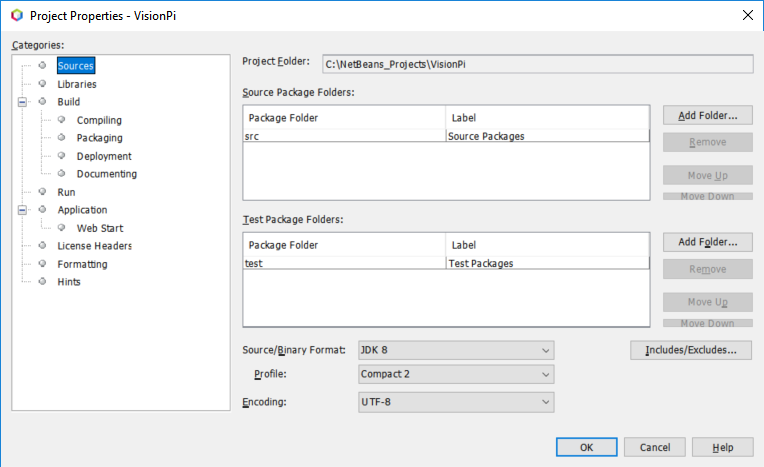
When you try to select the Remote Platform you will get the following message. Make up a configuration name and hit OK.



When you try to run your project with the little green arrow, you will likely see this error:

*java.lang.UnsupportedClassVersionError: javatest1/JavaTest1 has been compiled by a more recent version of the Java Runtime*

*Set the Source binary format to JDK8 and Profile to Full as shown below. Setting Profile to Compact 2 seems to disable code completion so leave it on full*

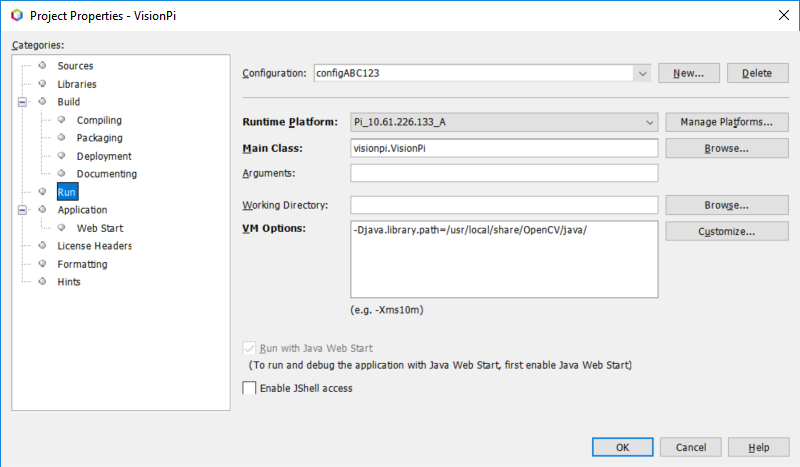




1. Select the Run category and add the following in the **VM Option**: box.

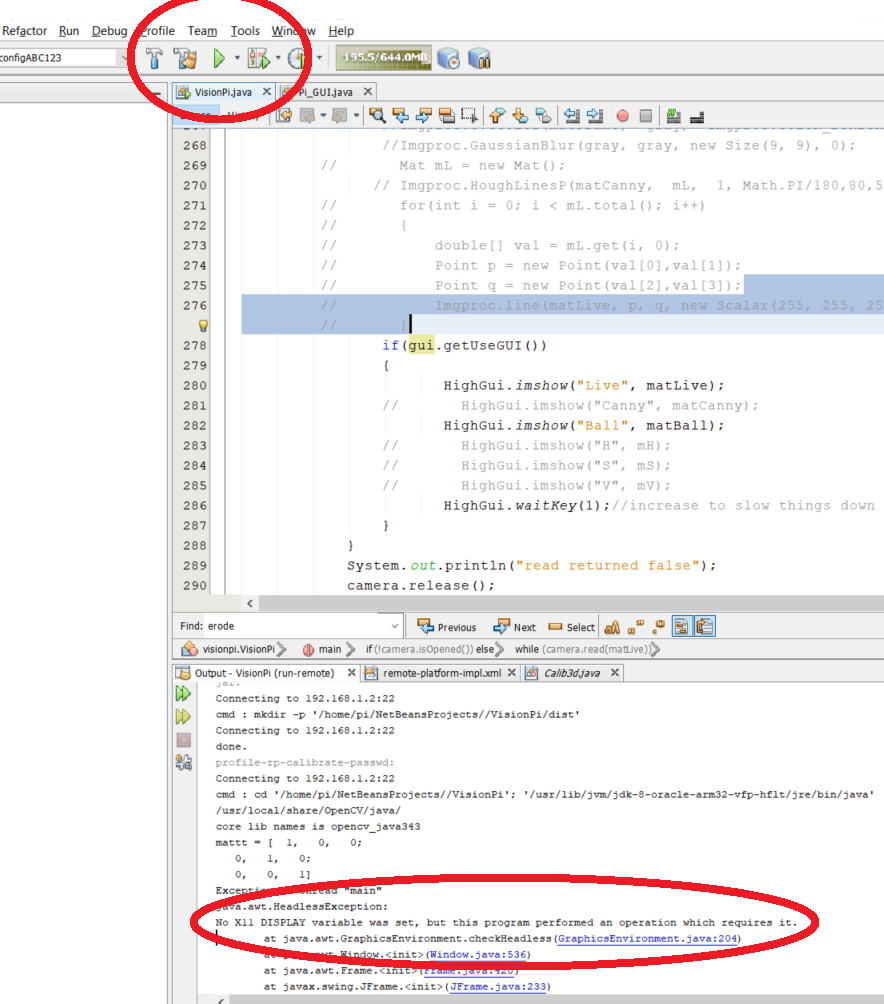
**-Djava.library.path=/usr/local/share/OpenCV/java/**

This file should exist in that directory on Pi: **opencv\_java343.jar**





1. Click the green run triangle and the program will be loaded to the Pi and run. You will get an X11 error.



1. Start xMing on the PC. You should see the little red Xming symbol in the “show hidden icons” menu on the bottom right side of the windows system tray / task bar.
2. Using Putty
   1. connect/log in to the Pi
   2. start camera driver with **sudo modprobe bcm2835-v4l2**
   3. navigate to the folder where the Netbean project is loaded and start the program with this command:

**java -jar -Djava.library.path=/usr/local/share/OpenCV/java/ VisionPi.jar**

or you can start the program from the pi/home directory by specifying the path with this command:

**java -jar -Djava.library.path=/usr/local/share/OpenCV/java/ /home/pi/NetBeansProjects/VisionPi\_DeepSpaceTargetTrack/dist/VisionPi.jar**

*Note - To start the program on boot:*

* disable the GUI in your program with ENABLE\_GUI = false; The program will exit during the boot if the GUI is enabled.
* Open the nano editor with:

sudo nano /etc/rc.local

* add these lines near the bottom (just before exit 0) of the rc.local file to load the camera driver and start the program on boot:

**modprobe bcm2835-v4l2**

**java -jar -Djava.library.path=/usr/local/share/OpenCV/java/ /home/pi/NetBeansProjects/VisionPi/dist/VisionPi.jar**

*Note - To stop the program that was loaded on boot:*

* One way is to comment out the lines you added to rc.local and reboot.
* Another way is to kill the process from the putty session. The ps command will list the running processes

**ps -x -a**

* Scroll down to the process for your program. If will look similar to this:

424 ? Sl 17:38 java -jar -Djava.library.path=/usr/local/share/OpenCV/java/ /home/pi/NetBeansProjects/VisionPi/dist/VisionPi.jar

* Note the process number that is shown. In this case 424. Kill the process

**sudo kill -9 424**

* 1. **Any windows created by your program should show on your PC.**



1. **Add Pi4j library to project –** Pi4j allows us to turn on/off GPIO on the Pi’s 40 pin header so we can light LEDs.

Download the release build, pi4j-1.2-SNAPSHOT, zip file from here. Note the 1.2 snapshot version works with the pi version 3b+. Pi4j ver 1.1 gives an error at runtime that the processor is incorrect so make sure you use 1.2.

<http://pi4j.com/download.html>

Unzip the file and you should see various pi4j jar files in the lib folder.

 -In the project's tree, right-click the "Libraries" folder and select "Add Library"  
-"Create" a new library using Library Name: "Pi4J-Core"  
-Make sure the "Classpath" option is selected and then "Add JAR/Folder" - select the "pi4j-core.jar" file from your local file system (assuming you already have downloaded Pi4J on your computer)  
-Select the "Sources" option and add the "pi4j-core-sources.jar".  
-Select the "JavaDoc" option and add the "pi4j-core-javadoc.jar".  
-Select the "OK" button to finish adding the Pi4J Library to NetBeans.

You should now be able to resolve these imports in the Netbeans IDE

import com.pi4j.io.gpio.GpioController;

import com.pi4j.io.gpio.GpioFactory;

import com.pi4j.io.gpio.GpioPinDigitalOutput;

import com.pi4j.io.gpio.PinState;

import com.pi4j.io.gpio.RaspiPin;

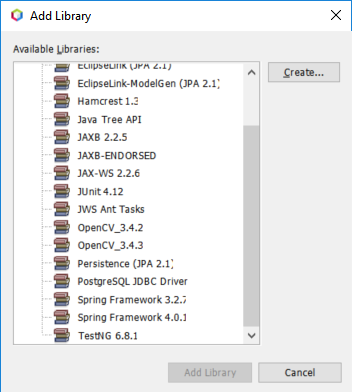
todo: paste in basic example code here

1. **Add OpenCV lib to project**

<https://www.codeproject.com/Tips/717283/How-to-Use-OpenCV-with-Java-under-NetBeans-IDE>

or this is easier …Right Click Libraries folder under the project and add OpenCV3.4.3 library







1. If you would like to test/**develop your code on PC** rather than on Pi, select the local platform. And change the Vm options to where ever you installed OpenCV on you PC

**-Djava.library.path=C:\OpenCV\_3.4.3\opencv\build\java\x64**

When deploying locally on the PC, It is looking for this file on the PC



**opencv\_java343.dll**

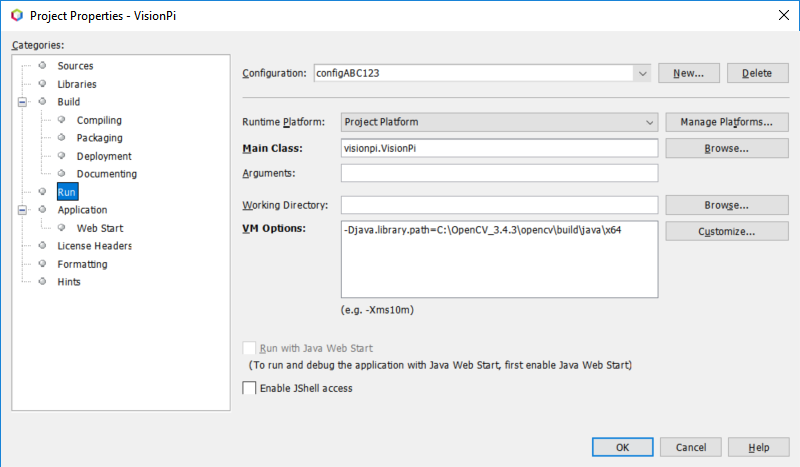
**You will need to change it back when you want to deploy to the Pi again**

**-Djava.library.path=/usr/local/share/OpenCV/java/**

When deploying remotely to the Pi, It is looking for this file on the Pi



**opencv\_**

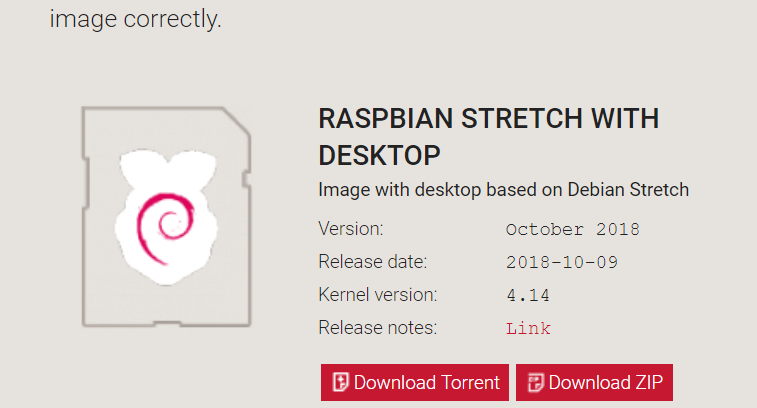
1. 



**Section 4 - Initial Setup of Raspberry Pi and OpenCV on Pi** (will take all day the first time you do it…. about four hours with some practice. Pay attention to any errors that are displayed at each step then google the errors and try to resolve them before moving to the next step. You can skip this section if you can copy an SD card from a PI that has been set up, but you won’t learn nearly as much.)

1. **Download the latest image** to your laptop from here:

#### <https://www.raspberrypi.org/downloads/raspbian/>

****



1. **Unzip the image.**
2. **Download and Install Win32 Disk Imager** from here:

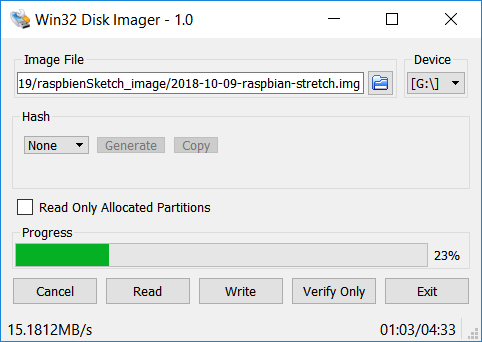
<https://sourceforge.net/projects/win32diskimager/>

****



1. Using Win32 Disk Imager, **write the unzipped image** to a 16GB or larger SD card.
   1. If you run into problems, see Win32 Disk Imager and SD Card Formatter Sections of Appendix A
   2. **Don’t format card** even when windows says you must. We are making a bootable card. You may be asked a few times. Always click **cancel**.
   3. Use a reasonable fast card. This link compares many cards for use with the Pi.

<https://www.jeffgeerling.com/blog/2018/raspberry-pi-microsd-card-performance-comparison-2018>





1. **Initial Pi Setup:** 
   1. Insert SD card, connect HDMI Display, mouse and keyboard to Pi for initial set up. Power up the Pi and you should see the Raspbian Desktop.
   2. When prompted to set select a language, keyboard type and time zone.
   3. When prompted, setpi password to “pi”
   4. When prompted, set up Wi-Fi or if you intend to use ethernet, skip this step and plug in ethernet. The Wi-Fi or Ethernet needs to be connected to the internet.
   5. When prompted to update software, click next. After minute(or as much as 10), you should see a message that “System is up to date”.
   6. When prompted, reboot and you will reboot to the desktop.
   7. Hover the mouse over the Wi-Fi icon and write down the ethernet and/or Wi-Fi IP address. Example: eth0: *192.168.1.4, wlan0: 192.168.1.5*
   8. Click the Terminal Icon at the top left of the display and enter the following to start the configuration tool.

*sudo raspi-config*

* 1. Using the configuration tool, select these settings

Boot options -> B1 Console Autologin, Automatically logged in as ‘pi’ user

Interfacing Options -> P1 Camera, enable

Interfacing Options -> P2 SSH, enable

I~~nterfacing Options -> P3 VNC, enable~~

Advanced Options -> A1 Expand File System

Select Finish and reboot

The Pi will boot to the console. Turn off the pi with

*sudo shutdown now*

*Note: Get in the habit of using the* ***sudo******shutdown now*** *command before removing power from the pi. If the pi in in the middle of a flash access when you remove power, you can corrupt the card causing strange behavior at best or make the card unbootable and need to start over. This applies during development and during an FRC match. We will use a* ***backup battery*** *on the match ready robot.* ***Make backups*** *of your SD card as needed.*

* 1. Power down the Pi; unplug the display, keyboard and mouse. From here on we will be “headless” using an SSH remote Putty session to allow us to paste in long commands from the laptop.

1. **Set up to build OpenCV**

*Note: Performing Copy and paste in a putty session is different than windows. Text highlighted in a putty window is automatically copied to the windows clipboard. Right clicking in a putty session pastes in at the curser position from the windows clipboard. Feel free to type any short commands but* ***copy and paste the long ones from this document to avoid typos****.*

* 1. Using Putty that you set up in a previous section, start session with Pi. Enter the following commands

*sudo apt-get update*

*sudo apt-get upgrade*

* 1. Free up space by removing wolfram and libreoffice.

*sudo apt-get purge libreoffice\**

*sudo apt-get clean*

*sudo apt-get autoremove*

* 1. Get the OpenCV code, unzip and create a folder where we will put the build:

*wget -O opencv.zip https://github.com/opencv/opencv/archive/3.4.3.zip*

*wget -O opencv\_contrib.zip* <https://github.com/opencv/opencv_contrib/archive/3.4.3.zip>

*unzip opencv.zip*

*unzip opencv\_contrib.zip*

*cd opencv-3.4.3*

*mkdir build*

* 1. Set JAVA\_HOME environment variable on pi
     1. Open bashrc file using nano editor

*sudo nano ~/.bashrc*

* + 1. Scroll to the bottom of file with the down arrow and paste in

export JAVA\_HOME=/usr/lib/jvm/jdk-8-oracle-arm32-vfp-hflt

* + 1. ctrl+o then enter to write changes to file
    2. ctrl+x to exit nano editor
    3. reboot the pi

*sudo shutdown -r now*

* 1. Verify JAVA\_HOME is set correctly

*echo $JAVA\_HOME*

Verify path is: /usr/lib/jvm/jdk-8-oracle-arm32-vfp-hflt

If it is not set, response will be blank.

* 1. Install Apache Ant

*sudo apt-get install ant*

* 1. Install CMAKE

*sudo apt-get install cmake*

* 1. Install dependencies the OpenCV build will need

sudo apt-get install libpng-dev

sudo apt-get install libpng++-dev

~~sudo apt-get install libgstreamer-plugins-base1.0-dev~~

* 1. videodev.h will be needed during cmake step below so make it available

sudo apt-get install libv4l-dev

cd /usr/include/linux

sudo ln -s ../libv4l1-videodev.h videodev.h

* 1. The build will need lots of memory so increase the swap file size. We will change it back after the build.
     1. Open swap file configuration file using nano editor

sudo nano /etc/dphys-swapfile

* + 1. Scroll to this line using the arrow keys: CONF\_SWAPSIZE=100
    2. Change to CONF\_SWAPSIZE=1024
    3. ctrl+o then enter to write changes to file
    4. ctrl+x to exit nano editor
    5. restart swap service (Reboot would work equally as well)cd ~

sudo /etc/init.d/dphys-swapfile restart

1. **Build OpenCV**
   1. The next couple steps fully use the CPU so it will get hot. Insure the little heatsinks have been placed on the pi’s two main ICs. Remove any covers that will cause heat build up. Some suggest a fan. Without a fan, mine gets so hot that I can’t keep my finger on the heatsink which is a good rule of thumb that an IC is too hot. So far, I see no signs that I have overheated the pi. If you have a fan handy, use it.
   2. CD into this directory you created in a previous step.

cd ~/opencv-3.4.3/build

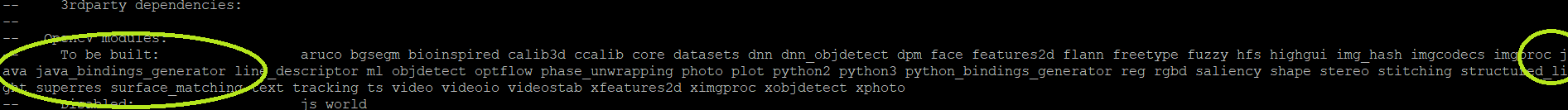
* 1. CMAKE – Sets things up for the make step that follows. Multiline strings can be hard to copy and paste into putty, so the tiny line below has the needed command on one line.

TODO: Test with Arm NEON and VFPV3 compiler switches. The following link claims 48% increase in speed. Try these opencv calls with getNumberOfCPUs and getNumThreads

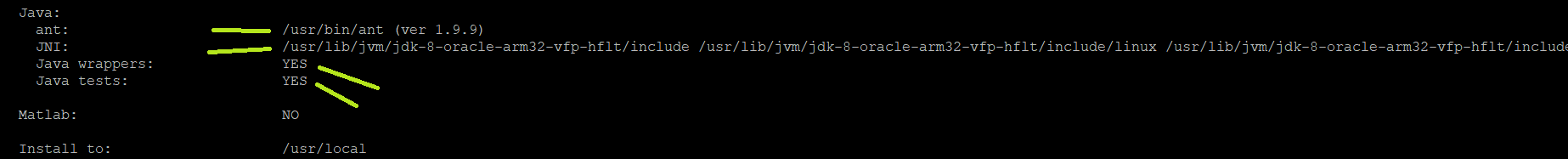
<https://www.pyimagesearch.com/2017/10/09/optimizing-opencv-on-the-raspberry-pi/>

cmake -D CMAKE\_BUILD\_TYPE=RELEASE -D CMAKE\_INSTALL\_PREFIX=/usr/local -D WITH\_LIBV4L=ON -D OPENCV\_EXTRA\_MODULES\_PATH=~/opencv\_contrib-3.4.3/modules ..

* 1. Scroll back in the cmake output, look for “***To be built***” and verify ***java and java\_bindings\_generator*** are listed



* 1. Scroll back in the cmake output, look for “***Java***” and verify all 4 say **yes** or **show a path**



* 1. Scroll back in the cmake output and compare errors to the ones in the appendix under section “Expected/acceptable failures during cmake” These should more or less be the only errors during the build.
  2. *Make – This step actually builds OpenCV. This step takes about 2 hours. Sometimes the display does not change for as long as 10 minutes. When done you should see the command prompt.*

make -j6

* 1. Now that the build is done, install it *(takes about 1 minute)*

sudo make install

* 1. Put the swap file size back to 100
     1. Open swap file configuration file using nano editor

sudo nano /etc/dphys-swapfile

* + 1. Scroll to this line using the arrow keys: CONF\_SWAPSIZE=1024
    2. Change to CONF\_SWAPSIZE=100
    3. ctrl+o then enter to write changes to file
    4. ctrl+x to exit nano editor
    5. restart swap service (Reboot would work equally as well)

sudo /etc/init.d/dphys-swapfile restart

1. Install pi4j - This laibrary will allow access to the GPIO so we can light LEDs on the Pi’s 40 pin header

curl -s get.pi4j.com | sudo bash

1. Quick test ofOpenCV

*python*

*import cv2*

*cv2.\_\_version\_\_*

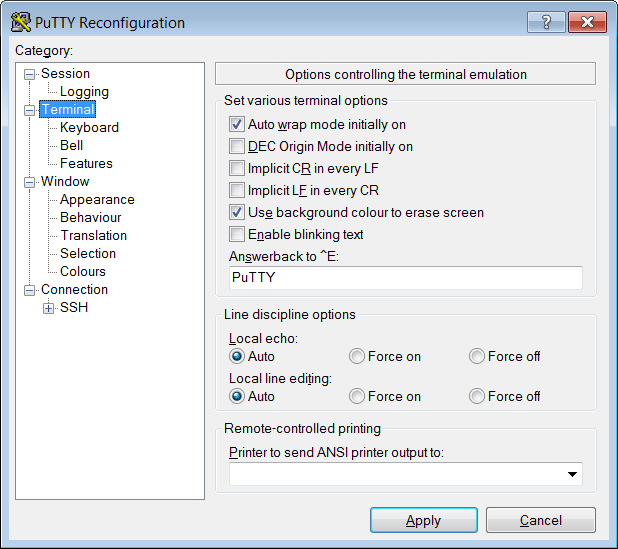
**You should see ‘3.4.3’**

Type ctrl+d to exit python**.**

**Appendix A - Various Tips**

1. **Setup Putty Window** – so cursor does not jump

The setup Chris Daham showed caused strange behavior in nano editor and leaving old text at the command prompt when simply pressing up arrow or down arrow. Uncheck Implicit CR and LF as shown below:



1. **Connect Pi directly to laptop via ethernet**

* On Pi modify dhcpcd.conf

sudo nano /etc/dhcpcd.conf

#static IP configuration

interface eth0

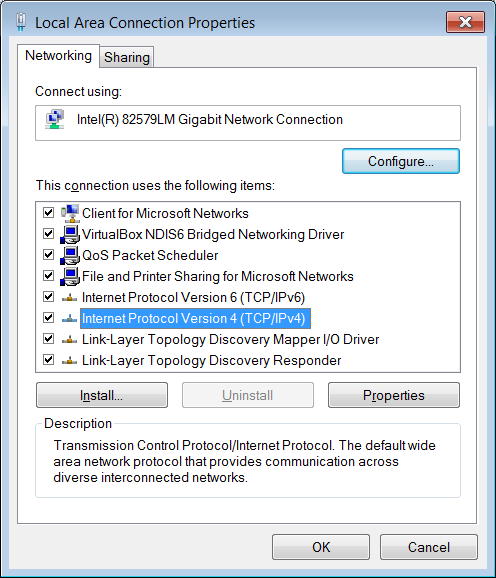
static ip\_address=192.168.1.17/24

static routers=192.168.1.1

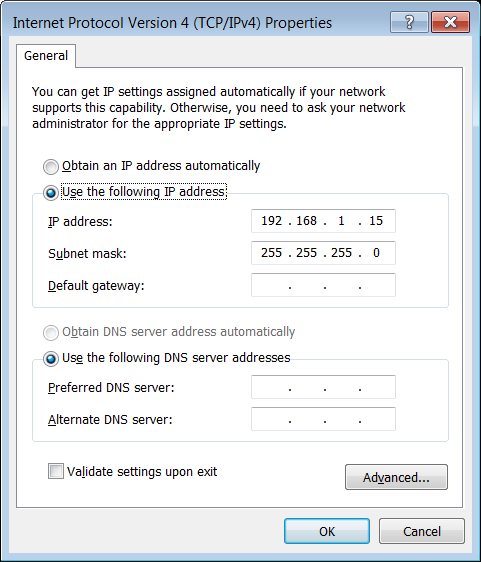
static domain\_name\_servers=192.168.1.1

* On Laptop modify ethernet setup

Control panel -> network and sharing center ->change adapter settings -> local area connections-> properties



->properties



Set static IP address and mask

1. **How to Backup SD Card** – When copying SD card images, you can run into problems when the source card is a larger than the destination card. Two seemingly identical 8GB cards may be different by a few bytes. You might expect 7GB image, for example, to fit on a 8GB card but if the source card is a larger 16, 32 or 64GB some copy approaches will not work. The latest Raspbian OS has a utility, SD Card Copier included with the image. I could not get this to work

The easiest approach I found is to load “RPI Clone” to the Pi and let the Pi make the backup.

* 1. For a new card, you may have to format the card using SD Card Formatter (Step 5 below).
  2. Follow the RPI Close Setup found here (just need the 3 commands listed below.)

<https://github.com/billw2/rpi-clone>

$ git clone https://github.com/billw2/rpi-clone.git

$ cd rpi-clone

$ sudo cp rpi-clone rpi-clone-setup /usr/local/sbin

* 1. Plug into Pi, a USB adapter with micro SD card you want to copy to.
  2. If you see a message “Do you want to unmount /media/pi/3464-3134? (yes/no):”

Answer yes then enter

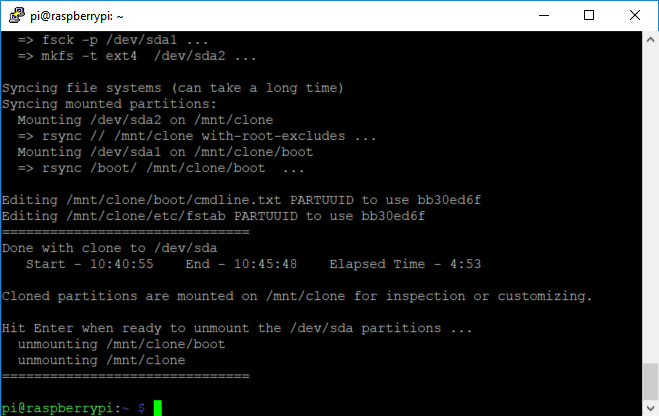
Initialize and clone to the destination disk sda? (yes/no):

Answer yes then enter

* 1. From Putty session, enter

sudo rpi-clone sda

After about 20 minutes, you should see: (Screen may appear frozen for most of the 20 minutes. The green SD activity light will be flashing rapidly.

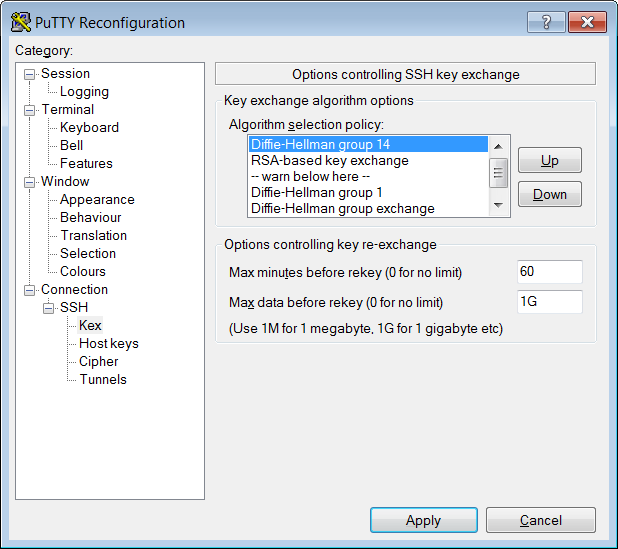




If the procedure fails. Try again (no need for step b. Just do steps a,c,d and e

1. ***Win32 Disk Imager –*** *Is used to write initial raspbien image to Pi. If the SD card is formatted wrong you will get the an “access denied” error. Follow the next step if that is the case. If you have a card with* an image that you no longer want, insert into laptop slot and DON’T format when asked. Instead erase all the files from file explorer. Then you can load a new image without needing the SD card formatter that takes a long time (30+ minutes)
2. **SD Card Formatter** – You will need to do this for a new card or if you get error: “Access denied” when writing image to SD card with Win32 Disk Imager
   1. download SD card formatter <https://www.sdcard.org/downloads/formatter_4/eula_windows/index.html>
   2. Select overwrite Format, not quick format (will take about 30 minutes for a 16GB card)
   3. Then write the image with Win32 Disk Imager or clone to the card with RPI clone
   4. **Don’t** **format** the disk when it asks. It is a boot disk
3. **Error when trying to connect Putty to Pi:** PuTTY Fatal Error: expected key exchange group packet from server

In putty settings under **connection->ssh->kex** I moved Diffie-Hellman group 14 to the top of the list and Diffie-Helman group exchange to the bottom. this solved the issue . See screen shot below: Don’t forget to save settings.



1. **Putty connects and provides log in but session hangs :** Putty was working yesterday and nothing changed. Login from local keyboard and display works fine. If after entering login name and password in putty, you don’t get a command prompt, **try creating a new SSH putty session with nothing other than the IP address.** After comparing the reg settings between the working session and the not working session, both session were working again.
2. **Shutdown in headless mode Concern**

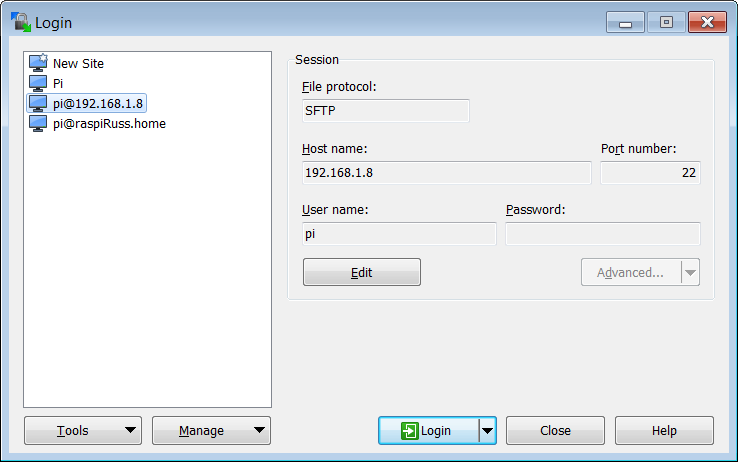
* Ask mentors
* Backup SD card
* Wait for SD act light to stop flashing
* Add power down button to GPIO with monitor service

1. **Bring up windows on laptop with XMing**
   1. Start xming on laptop
   2. Connect SSH putty session with pi
   3. Type (without sudo)

pcmanfm &

1. **How to Copy Files to Pi**

Use a USB stick or install **WinSCP** to copy files from another PC. Use IP address or hostname raspiRuss.home



1. **Display path to java**

which java

1. **View syslog**

tail -n 2000 /var/log/syslog

1. **Cursor jumps and pastes in junk when using Nano editor via Putty**

-Solution 1: use keyboard and monitor local to Pi

-Solution 2: Edit files using **WinSCP**

1. **XMing** – When XMing is running on laptop, windows will pop up on laptop as whenever they would have popped up on pi desktop.

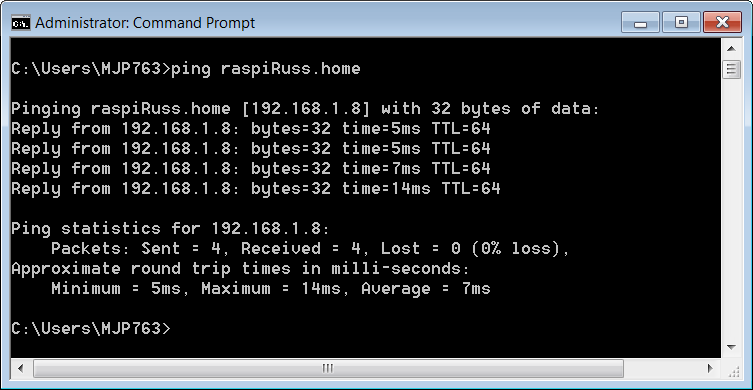
Install according to Chris Daham’s cheat sheet – Don’t forget X11 forwarding steps.

1. **What is PI’s IP address?**

**On PC, Type hostname -I**

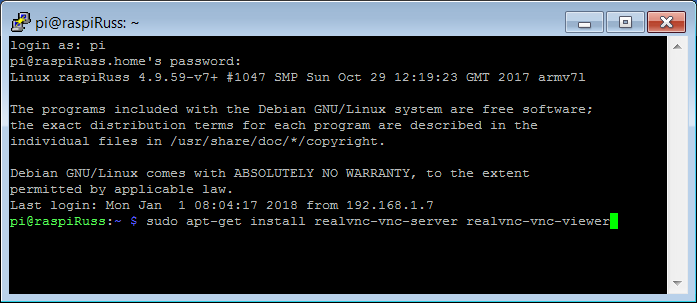
Or Ping the Pi’s host name with: ping raspiRuss.home

IP address will display.

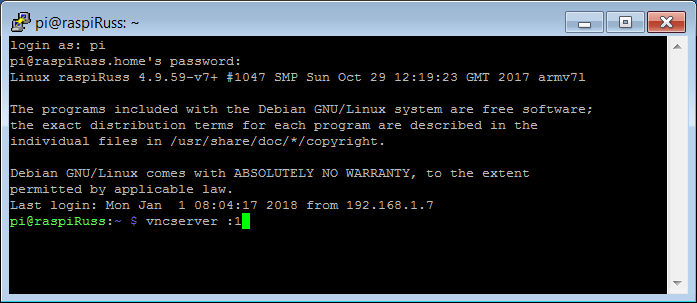


1. **Install/Setup VNC to connect laptop via ethernet (**Only do this if you are running the Pi headless and want the Raspbian desktopto show up on your laptop)

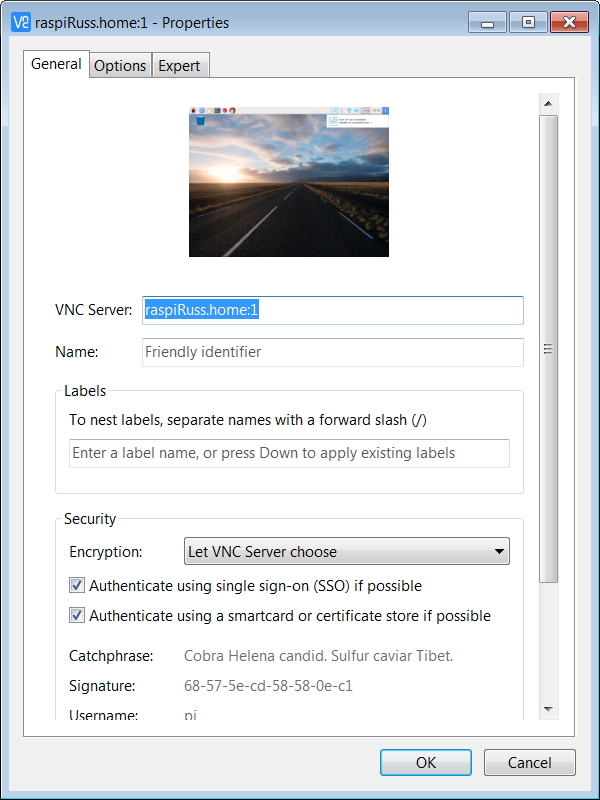
$ sudo apt-get update  
$ sudo apt-get install realvnc-vnc-server realvnc-vnc-viewer

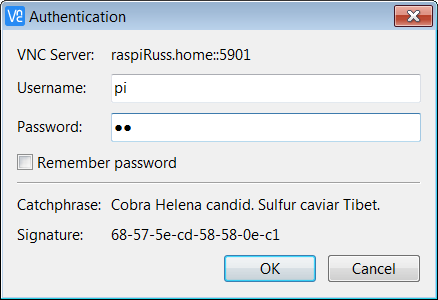


* Enable VNC on Pi in the settings menu
* //On raspberry Pi, enter vncserver :1 to start server – there is a space before the :1
* Add vncserver :1 as the last line of etc/bash.bashrc with sudo nano etc/bash.bashrc
* Reboot pi
* Tunnel in with putty
* Start vnc on laptop
* To use ethernet cable to Pi, no need for crossover cable as Pi has auto switching to x over
  + Install DHCP server on laptop: Run this bat file: RunStandAlone.bat. Install and exe lives here: C:\OpenDHCPServer
  + Pi board is getting IP as 169.254.119.77
  + 127.0.0.1:6789 not displaying server page… err in log file
  + Ctrl C to exit DHCP server



On laptop, run RealVNC ->file->new connection. Then only need to enter hostname, (ras pi Russ home) or ip address followed by :1 then open the connection. User is pi, password is pi: See below





1. **Set up Raspbian GUI boot**
   1. When you are in the command line, log in with default user password (if you did not change it)
   2. When you are logged in, run the following command: sudo raspi-config
   3. -In this configuration screen, look for the option of Enable Boot to Desktop/Scratch.
2. **Load internal camera driver for OpenCV**

sudo modprobe bcm2835-v4l2

Type the above command; the letter l is between 4 and 2 , no spaces around dash

1. **Display camer driver info after loading it in last step**

v4l2-ctl -d /dev/video0 --all (there are two dashes before all)

1. **Check available camera features**

Load driver 1st, with sudo modprobe bcm2835-v4l2

v4l2-ctl --list-formats-ext -d /dev/video0

video0 is 1st camera (plugged in USB or starting v4l2 driver) , Video1 is 2nd camera

1. **Set ras pi camera params**

v4l2-ctl --set-fmt-video=width=1920,height=1088,pixelformat=4

1. **Take video**

raspivid -o video.h264 -t 10000

1. **Take still pic – vertical flip and horizontal flip**

raspistill -vf -hf -o cam2.jpg

1. **List video formants supported by camera**

v4l2-ctl -d /dev/video0 --list-formats

1. **Display camera type and format**

v4l2-ctl -D

v4l2-ctl -V

1. **Can’t load onboard camera driver**

“Could not insert 'bcm2835\_v4l2': Operation not permitted”

*Use “sudo raspi-config” to enable camera under “interface Options” menu then reboot*

1. **Get IP address of Pi**

Ping raspiRuss or use advanced IP scanner

1. Linux **Screen** utility

[**https://www.youtube.com/watch?v=hB6Y72DK8mc**](https://www.youtube.com/watch?v=hB6Y72DK8mc)

This utility provides two primary functions that are nice to have.

1. Multiple screen sessions with one SSH (putty) connection that can be switched back and forth
2. If a network failure happens during a long process like building OpenCV, once the network is restored, you can connect to the session to see how the build progressed without rather than start the build again.
3. **Motion Sensing/tracking with still camera**

<https://www.youtube.com/watch?v=X6rPdRZzgjg>

# *OpenCV Tutorial: Real-Time Object Tracking Without Colour* by [Kyle Hounslow](https://www.youtube.com/channel/UCJ2b0kP6Hwc_R8ebv8P2f9w)

This only applies when the camera is still so it does not apply so much to FRC. Detecting motion from a moving camera needs a more advanced approach. Regardless it teaches some important features of OpenCV.

1. **Remove program**

**sudo apt-get --purge remove APPNAME**

**sudo apt-get autoremove –purge**

1. **Expected/acceptable failures during cmake**. (clean these up when time permits)

HAVE\_CXX\_WMISSING\_PROTOTYPES – Failed

HAVE\_CXX\_WSTRICT\_PROTOTYPES – Failed

HAVE\_C\_WSIGN\_PROMO – Failed

HAVE\_C\_WNO\_DELETE\_NON\_VIRTUAL\_DTOR – Failed

HAVE\_CXX\_WNO\_UNNAMED\_TYPE\_TEMPLATE\_ARGS – Failed

HAVE\_C\_WNO\_UNNAMED\_TYPE\_TEMPLATE\_ARGS – Failed

HAVE\_CPU\_NEON\_SUPPORT (check file: cmake/checks/cpu\_neon.cpp)

CPU\_NEON\_SUPPORT – Failed

HAVE\_CPU\_FP16\_SUPPORT (check file: cmake/checks/cpu\_fp16.cpp)

HAVE\_CPU\_FP16\_SUPPORT - Failed

HAVE\_CXX\_MFPU\_NEON\_FP16 (check file: cmake/checks/cpu\_fp16.cpp)

HAVE\_CXX\_MFPU\_NEON\_FP16 - Failed

FP16 is not supported by C++ compiler

HAVE\_C\_FVISIBILITY\_INLINES\_HIDDEN – Failed

HAVE\_C\_WNO\_IMPLICIT\_FALLTHROUGH – Failed

HAVE\_CXX\_WNO\_INCONSISTENT\_MISSING\_OVERRIDE - Failed

HAVE\_CXX\_WNO\_IMPLICIT\_FALLTHROUGH – Failed

HAVE\_CXX\_WNO\_MISSING\_PROTOTYPES – Failed

HAVE\_CXX\_WNO\_TAUTOLOGICAL\_UNDEFINED\_COMPARE – Failed

HAVE\_CXX\_WNO\_SHORTEN\_64\_TO\_32 – Failed

HAVE\_CXX\_WNO\_ENUM\_COMPARE\_SWITCH – Failed

Failed to find installed gflags CMake configuration, searching for gflags build directories exported with CMake.

-- Failed to find gflags - Failed to find an installed/exported CMake configuration for gflags, will perform search for installed gflags components.

-- Failed to find gflags - Could not find gflags include directory, set GFLAGS\_INCLUDE\_DIR to directory containing gflags/gflags.h

-- Failed to find glog - Could not find glog include directory, set GLOG\_INCLUDE\_DIR to directory containing glog/logging.h

HAVE\_CXX\_WNO\_UNUSED\_PRIVATE\_FIELD – Failed

1. **Can find the right scoping for the OpenCV function or constant you want?** Example java code often does not compile because your OpenCV library is different or your imports are different. Python and C code is even harder to get the scoping right when you convert to Java. So when you can’t find an equivalent line of code, manually search the library for what you want by expanding the library in the Projects pane or right click-> find. This may help determine the correct import.
2. **How to eject a USB drive**

sudo apt-get install eject

once installed

udisks --unmount /dev/sda

udisks --eject /dev/sda

Does not work : todo figure this out

For now shutdown and unplug the USB drive

1. **Load GPIO for java**

Google pi4j from pi and download pi4j version 1.2 or later. It will put it in the home/pi/downloads folder

$ sudo dpkg -i /home/pi/Downloads/pi4j-1.2-SNAPSHOT.deb

The libs will be here: /opt/pi4j/lib

Add them to eclipse as external jars properties->java build path->libraries ->add external jars

1. Format SD card with SD Card Formatter (not formatter that comes with any OS) <https://www.sdcard.org/downloads/formatter_4/>

**Appendix B - Rasberry Pi Vision Processing Approach**

The PI 3B+ CPU can quickly be 100% utilized with just a few OpenCV vision processing (VP)functions on a 680x480 image. When this happens, the latency from when a physical object such as a ball moves to when the frame is processed can be a couple seconds or more. This would not allow responsive feedback to control motors to drive to and pick up the object.

A faster processor would be one option. Note that trying to find the ball in the image does not need quick response but once the ball is detected and VP is used as feedback to the motion control PID, the latency must be low.

A state machine will govern the approach

* Initial implementation will use sliders for calibration then hardcode those values. When ENABLE\_GUI is true, the state will be held at State -1 with the slider values printed to the output window. When ENABLE\_GUI is false, set the initial state to 3 to use the hardcode values. When States 0, 1 and 2 are implemented, set the initial state to 0.
* **State 0**: Discard 300 frames (10 seconds at 30fps) after boot to allow camera to stbilize. During this time the target ball should be placed near the center of the camera on the ground about 2 feet away. Slow flash LED 1 during this period
* **State** **1**: 50x50 pixels will be colored sampled and averaged for 3 seconds as the target hue. Fast flash LED 1
* **State** **2**: 10 seconds delay to allow ball to be removed from view if desired. No LED flash
* **State** **3**: Detect ball using large region of interest (ROI) (680x200). If detected, change to State 5.
* **State** **4**: Detect ball using full 640x480 frame while rotating and driving in a 3’ x 3’ area. If detected, change to State 5.

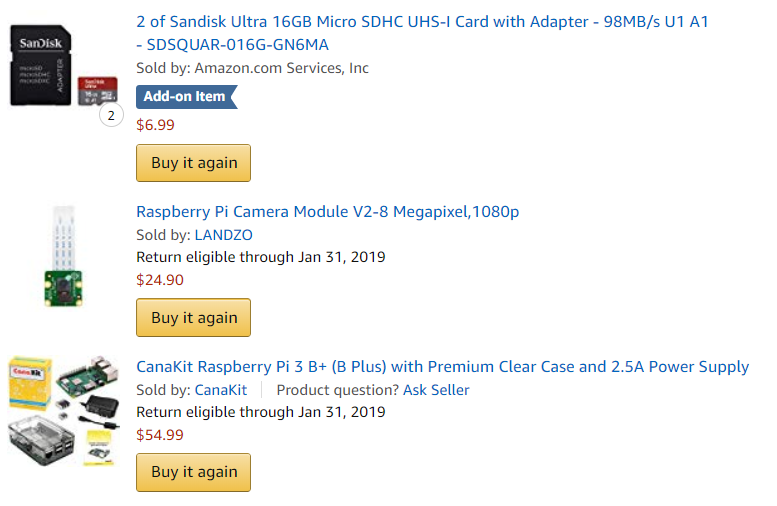
After 360 degrees rotation, change to State 3

* **State 5**: Drive to ball with reduced region of interest and limited detection scan line for H and V. If at ball, change to State 6. If detection lost, change to State 3.
* **State 6:** Attempt pick up ball. Then check if ball in gripper. It in gripper, change to State 7. If not, change to State 4.
* **State 7:** Rotate up to 360 degrees looking for drop box. If detected, change to State 8, else stop.
* **State 8:** Drive to drop box with reduced region of interest and limited detection scan lines for H and V. If at drop box, change to State 9.
* **State 9:** Drop in box; back up 2 feet; change to State 4.

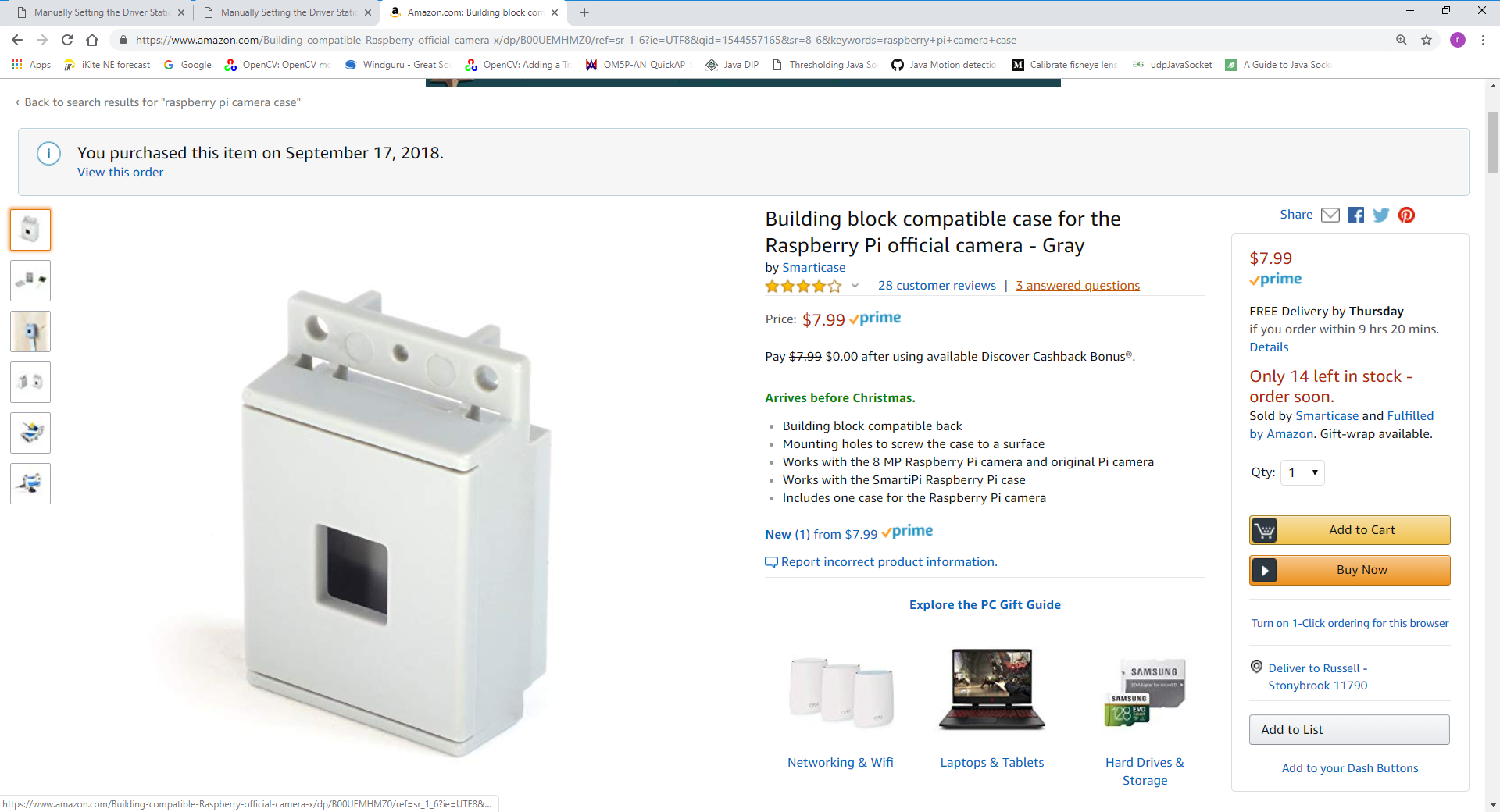
**Appendix C - Needed hardware to develop Rasberry Pi Vision Processing**

You will need the following parts totaling $94.87 from Amazon

16GB micro SD card, camera with flex, raspberry pi (version 3B+ is preferred. )

****

**After trying two other camera cases, the following is the only one that fits properly – Be real careful plugging in the camera flex. The connector on the board breaks easily. The flex and connector will not stand any abuse so make sure it is all screwed or help together with double sided tape**



**Some additional items you will need that you probably have:**

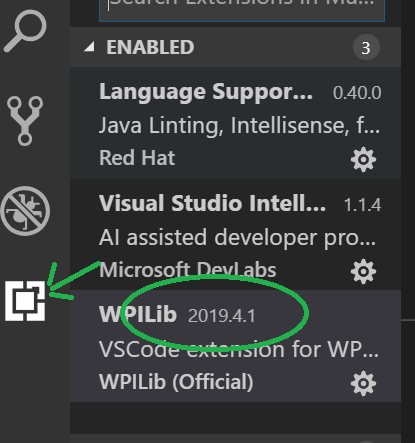
-Laptop or PC running Windows

-Network Options

* With a spare ethernet jack on your router and ethernet cable to plug your Pi into your network you wont need a monitor, keyboard or mouse connected to the Pi. Install Advanced IP Scanner on your laptop to determine the Pi’s IP address
* I have a few images you can load to an SD card depending on how much you would like to do yourself. Ideally you follow the procedure given above in Section 4 above.
  + Raspbien Sketch image (you will need an HDMI monitor, keyboard and mouse to set headless mode and set up a few other items described in section 4 above)
  + Raspbien Sketch image set up for headless (Jump right into building OpenCV as described in Section 4, Step 6 above.
  + Raspbien Sketch image set up for headless with OpenCV set up(Jump right into using OpenCV

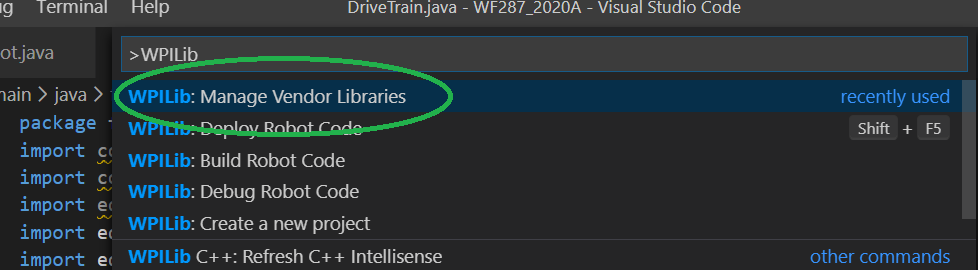
//\*\*\*\*\*\*\*\*\*\*\*TIPS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* **B: Check WPILib Version**
  + Open the VS Code project and make sure WPI lib version is **2019.4.1.** If it is not,
    - Connect to Wi-Fi with internet
    - Click the gear icon and uninstall
    - Search in search box of the same panel for WPI and install the correct version.



* **C: Install latest JSON files for third party libraries**

1. Connect to WiFi with internet access
2. From VSCode
3. Click on WPI Command Palette
4. Click on “Manage Vendor Libraries”

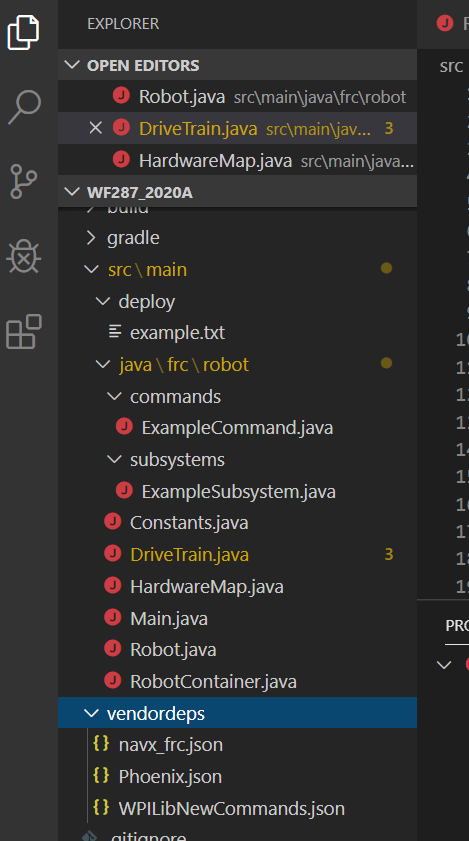
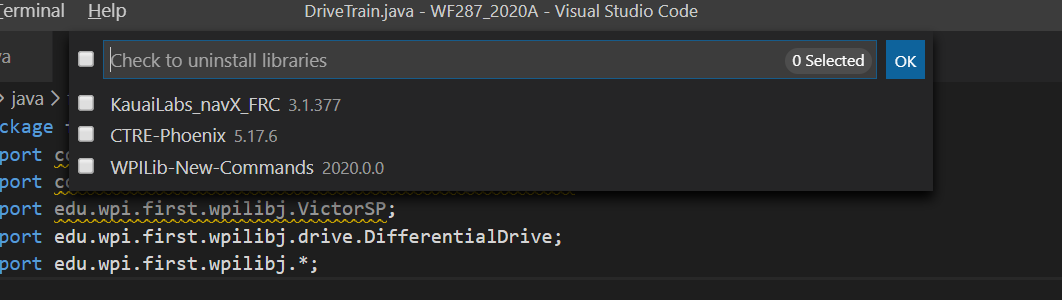


1. Uninstall Pheonix and NavX if they exist
2. Click install new libraries (**online**)” and add the following URLs for each vendor.

Pheonix -  <http://devsite.ctr-electronics.com/maven/release/com/ctre/phoenix/Phoenix-latest.json>

navX - <https://www.kauailabs.com/dist/frc/2020/navx_frc.json>

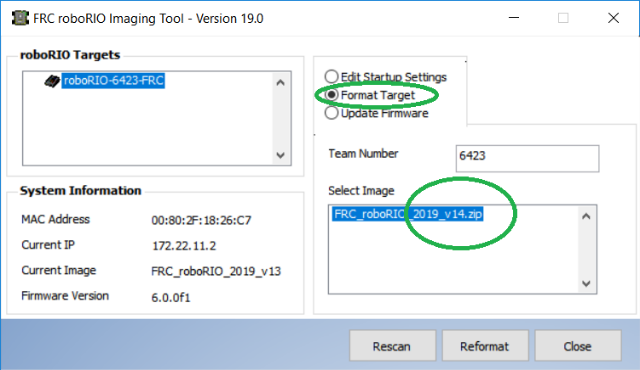
Then click on “Manage Vendor Libraries again… should see the CTRE and navX libs: You will also see them from the explorer as shown below:



* **D: Install 2019\_v14 firmware on Roborio**
  + Connect to the Roborio over WiFi (ESSID should be 6423) and deploy VS code project using the WPI command pallet(or Shift F5).
    - Skip this step if you have the correct version
    - If not, you will get the following error:

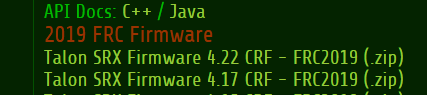
*RoboRIO Image invalid! RoboRIO: 2019\_v13, allowed: [2019\_v14]*

* + Connect laptop to Roborio using USB cable (not WiFi)
  + Start the Roborio imaging tool (Version 19.0)
  + Click “format Target” then “Reformat”
  + This takes about 20 minutes

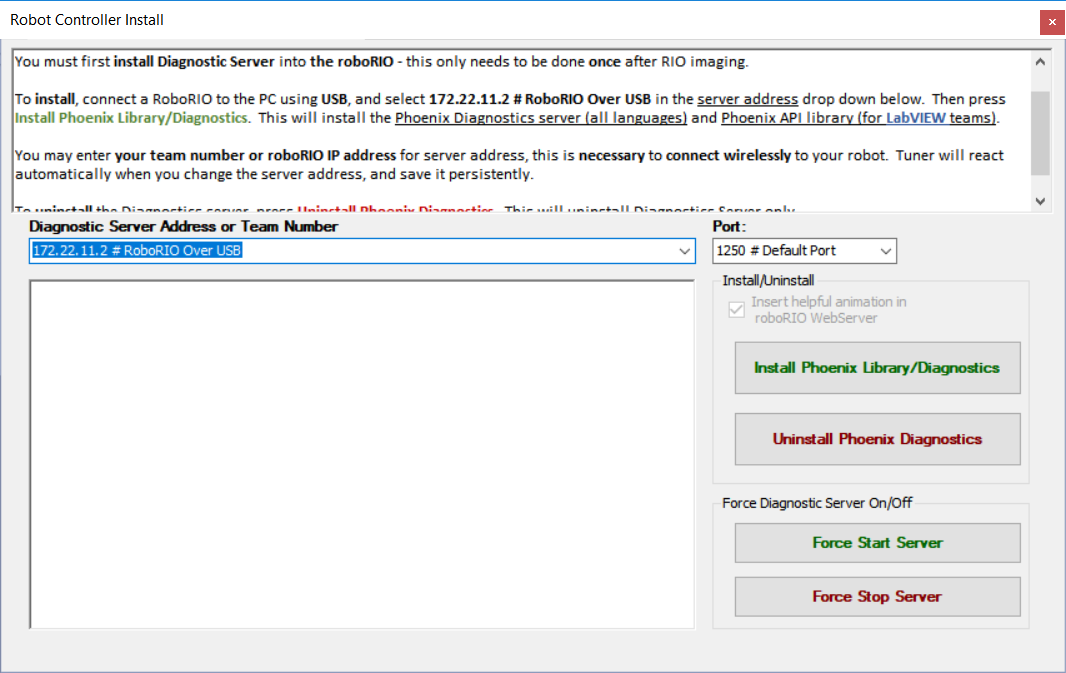


* **E: Update Speed controller firmware**
  + Copy the speed controller firmware to your laptop
    - Connect to Wi-Fi with internet
    - download the talon **4.22** and **4.17** CRF zip files from here.

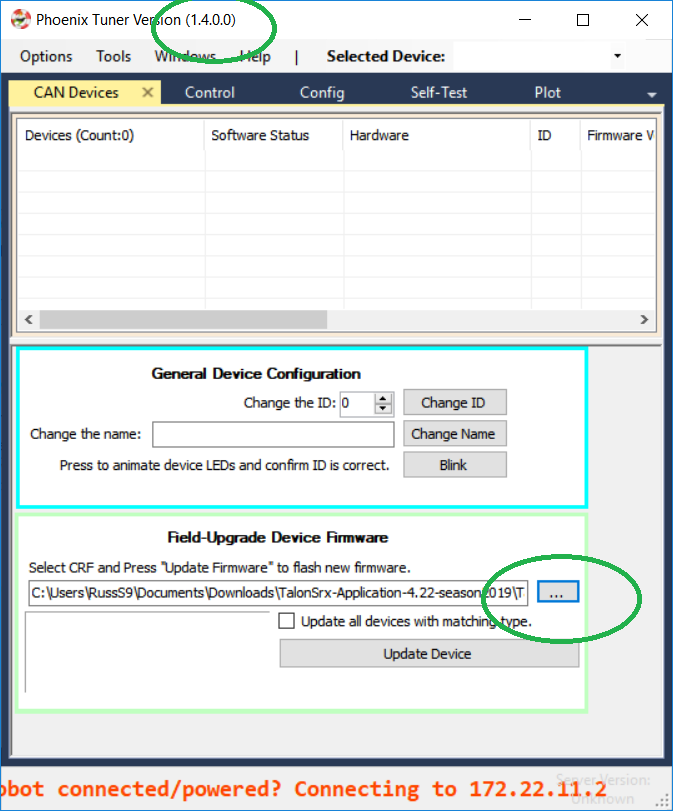
<http://www.ctr-electronics.com/talon-srx.html#product_tabs_technical_resources>



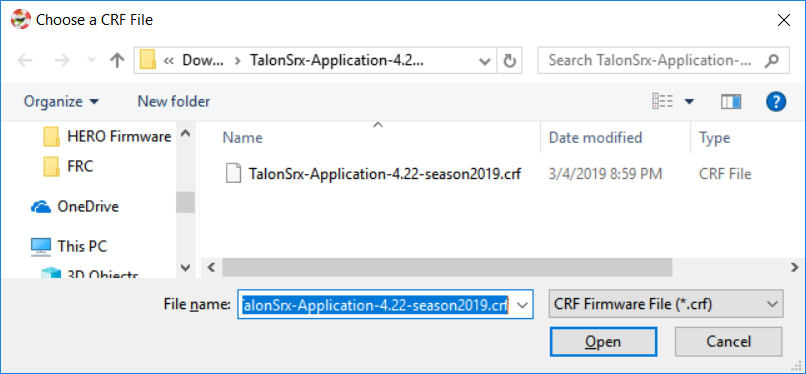
* + **Load the Talon SRX app on the Roborio** so it can communicate with speed controllers on the CAN bus. The Talon SRX app is loaded by the Phoenix Tuner app on laptop
    - *Connect to Roborio via USB*, start Phoenix Tuner version 1.4.0.0 on laptop *Warning: The window shown below may need to be dragger bigger to see the buttons.*
    - Click “Install Phoenix Library /Diagnostics” to load the app to the Roborio



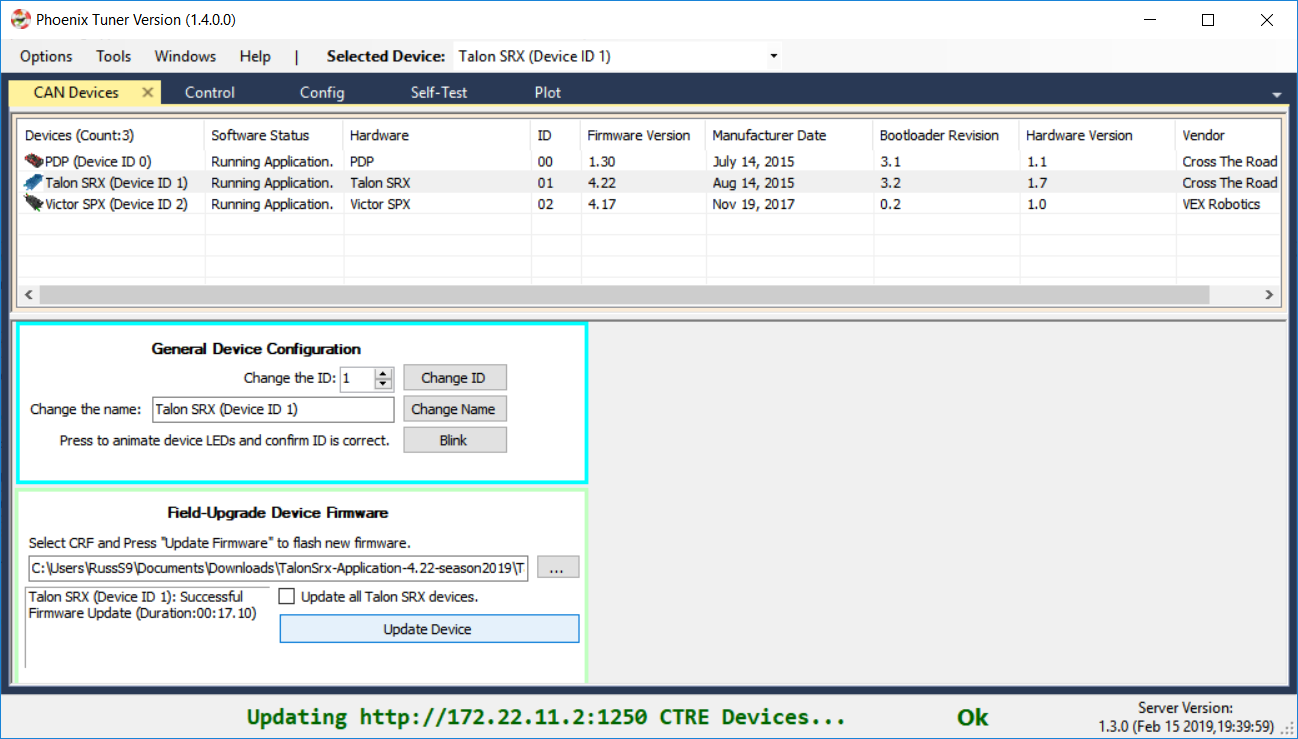
* + **Finally update the actual firmware on the Speed controllers**
    - While still connected via USB, using the other Phoenix Tuner window, click the ellipsis (three dots)



* + - Select the 4.22 CRF file. Select Talon SRX and Victor SPX speed controllers and update them



* + - All Talon **SRX devices should be updated to version 4.22**
    - The **Victor SPX can’t be updated** unless we find the correct CRF file. The Victor SPX sample I have has version 4.17 and seems to work correctly. If you try to load the CRF file for the Talon SRX to the Victor SPX, you will get a “Invalid CRF Wrong Product” error
* **F: Set the CAN IDs**
  + With Roborio still connected via USB and Phoenix Tuner still running, set the CAN IDs to match the HardwareMap class found in the **HardwareMap.java** file of the VS Code project. The summary is also listed above in step A.
  + The procedure is a bit of a dance involving changing the IDs to arbitrary high numbers so they are all different then blinking the LEDs with the blink button (LEDs only flash for a couple seconds, so be ready to observe LEDs) Then set the IDs to the desired values.



* **G: Possible Warnings** when code is 1st deployed and Teleop is enabled from driver station. Once the code is running, expect no warnings or errors. (These warnings are not critical, but we should research them if we have time. There should be no errors.)
  + Watchdog not fed within 0.020000s
  + Loop time of 0.02s overrun