Document
SUSE / Ubuntu Linux install notes for the Java application VNA-J

Author:

Andy Eskelson G0POY

Contact: andyyahoo@g0poy.co.uk

Status Issue 2

Table of Contents

Summary

This document describes the steps necessary to install the VNA-J application on SUSE Linux. It also provides some more general information that may be useful to system administrators of other Linux distributions. Also provided is a general description of how to solve permission issues relating to the detection of serial port adaptors. A seperate section describes the installation on a Ubuntu Linux system

WARNING

SOME OPERATIONS REQUIRE ROOT ACCESS, AND THE INSTALLATION OF APPLICATIONS AND LIBRARIES INTO SYSTEM AREAS OF THE OPERATING SYSTEM. THERE IS THE POTENTIAL FOR SEVERE DAMAGE TO YOUR SYSTEM. IT IS STRONGLY ADVISED THAT A FULL BACKUP IS TAKEN BEFORE PROCEEDING WITH THIS, OR ANY MAJOR INSTALLATION.

Introduction

The miniVNA instrument by mRS http://www.miniradiosolutions.com has proved to be a popular and very useful test instrument. This instrument is a simple box with two BNC connectors and a USB connector. All the control of this instrument is performed by a software application running on the host computer.

Many people have contributed to the development of this software, but the focus has been mainly on the Microsoft Windows operating system. There was a Linux based application but this is no longer supported, and the advancement of the various Linux distributions have rendered it inoperable. Recently Dietmar, DL2SBA has developed a control application based on the Java programming language. Java was designed to be a cross-platform language, which allows THE SAME application to run on any supported Java enabled Operating System.

There are several steps that must be undertaken in order to run VNA-J on Linux, and of course there are a number of pitfalls that you can get caught out by. This document is mainly based on the SUSE 11.1 version of Linux. At the time of writing SUSE 11.2 has been released and these instructions do cover the installation on SUSE 11.2 as well, however there is a problem with SUSE 11.2 which is an identified bug, so please take note of the section dealing with SUSE 11.2. The installation is basically the same on all versions on Linux, so the general information presented here should help users of other Linux distributions get the VNA-J application running.

Please note that I use the Gnome desktop, so some of the images may look different if you use say, KDE.

Requirements

Several items of software are required, some mandatory and some possibly optional. It will depend on what you already have installed. The version of of the Java runtime environment used is 1.6 this is purely because there is a screen-shot in the VNA-J documentation that shows this version being used.

In addition to the the Java runtime, an additional Java application library is needed, this is the serial port drivers, and is called rxtx.

Finally there is the VNA-J application itself.

Some optional software, which is needed if you need to compile the rxtx libraries is the Java development SDK. It is beyond the scope of this document to describe the compilation process.

The use of the binary distribution package makes the installation much easier, and far less prone to errors.

The software is available from:

VNA-J application: http://dl2sba.wikidot.com/vna-j

rxtx library http://rxtx.qbang.org/wiki/index.php/Download
(file "rxtx-2.2pre2-bins.zip")

From your Linux software repository if not already installed on your system:
java-1_6_0-openjdk
java-1_6_0-openjdk-plugin (web plug-in, may be required for other programs)

These are the `demo` and development archives, and are not required unless you intend to compile any applications from source.

java-1_6_0-openjdk-demo
java-1_6_0-openjdk-devel
java-1_6_0-openjdk-javadoc
java-1_6_0-openjdk-src

Java Check

Check which version (if any) of java that you have installed.
Open a terminal screen (> prompt) and type

```
> java -version
```

You should get a response similar to this:

```
> java -version
java version "1.6.0_0"
OpenJDK Runtime Environment (IcedTea6 1.6.2) (suse-0.1.1-i386)
OpenJDK Server VM (build 14.0-b16, mixed mode)
```

The important line is the Java version 1.6.0_0
If you have 1.6 or higher, then you do not need to install any of the java packages, apart from the development archives should you wish to compile applications from source.

Java Installation

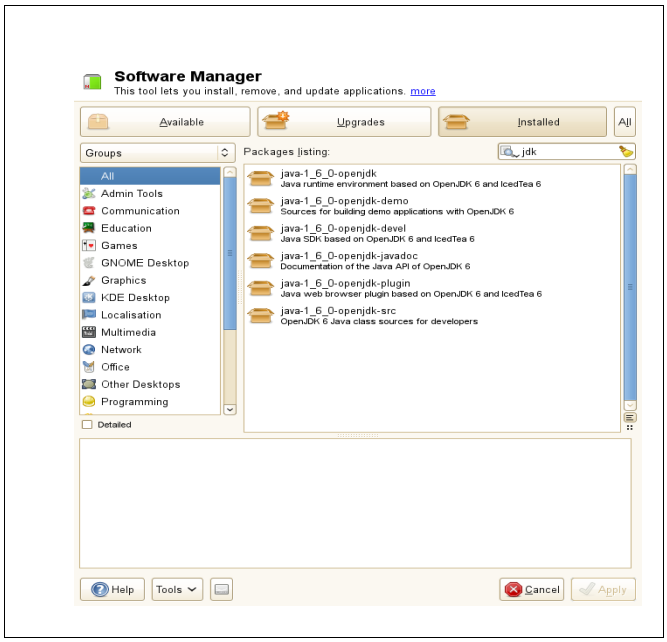
The installation of software in SUSE Linux is performed by the YAST package manager.

Run the YAST package manager and select software management.
In the package search box enter “jdk” and you may be presented with a package list. By default this will be whatever is installed on your system. If the Java Check test shows a java version, then you should see that version as already installed.

Assuming that you need to install a newer version of java, then you will need to click on the Available button in the YAST software manager. That will bring up a list of the available java development kit packages that are available.

Select the java-1_6_0-openjdk, which has the comment below of: “Java runtime environment....”
An install button will appear in the bottom right hand corner of the package manager. Click on this.
If you so wish, you can select additional packages to install in the same way. A list of the packages that you have selected will appear to the right of the main software manager window.

When you have finished your selections, click on apply and the installation process will start. Once finished, if you reopen the software manager, and enter jdk in the search box you will see the packages you selected, as “installed” The screen-shot below shown the result of this action after I had installed the runtime and the development packages.



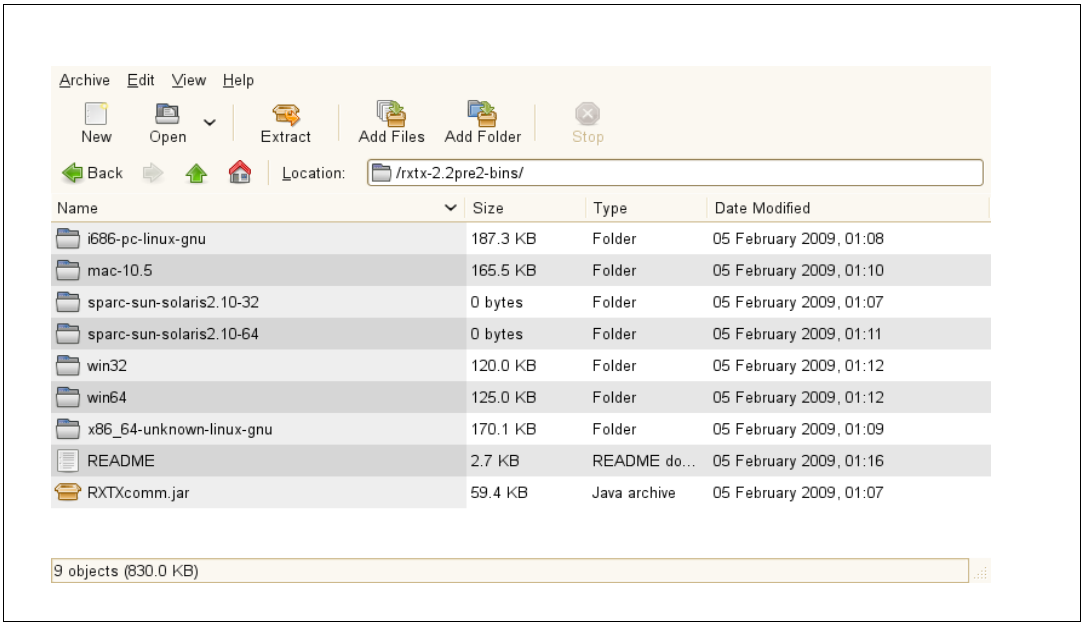
rxtx Installation

The next stage is to install the necessary libraries that drive the serial port.

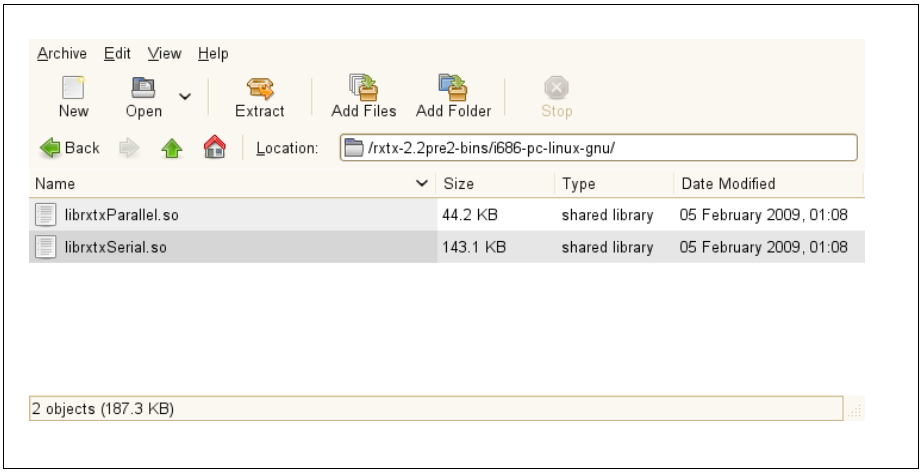
THIS REQUIRES ROOT ACCESS TO SYSTEM AREAS, BE CAREFUL!

Click on the rxtx-2.2pre2-bins.zip archive that you downloaded.
File roller will decompress the archive and present you will a single directory, rxtx-2.2pre2-bins, click into this to see the contents.

You should see a listing like this:



As you can see there are sub directories for the main OS types. For Linux we will use the topmost directory “i686-pc-linux-gnu”, so click on this. You will then see just two files



Create a directory on the desktop called “junk” and drag these files into it.

These two libraries now need to be copied elsewhere. You need to be superuser or root to be able to do this..

We need to copy these files into /usr/lib do this with the command line, or the mc (midnight commander) program. I will use the command line in this case.

Open up a terminal and enter the following commands
Note the text in brackets [] are my comments, do not type these

cd /usr/lib	[change to /usr/lib directory]
su	[switch to superuser mode]
*****	[you will be requested to enter your root password at this point]
#	[the prompt will turn to a # to indicate SU mode]

Now we copy the files
my home directory is called g0poy, you will need to change this to whatever your home directory is called.

enter the command

```
# cp /home/g0poy/Desktop/junk/librxtx*  /usr/lib/
```

This will copy all the files in the junk folder that we placed on the desktop into the /usr/lib directory, note the the space after the *.

We now check that the files have been copied
enter the command

```
ls librxtx*
librxtxParallel.so  librxtxSerial.so
#
```

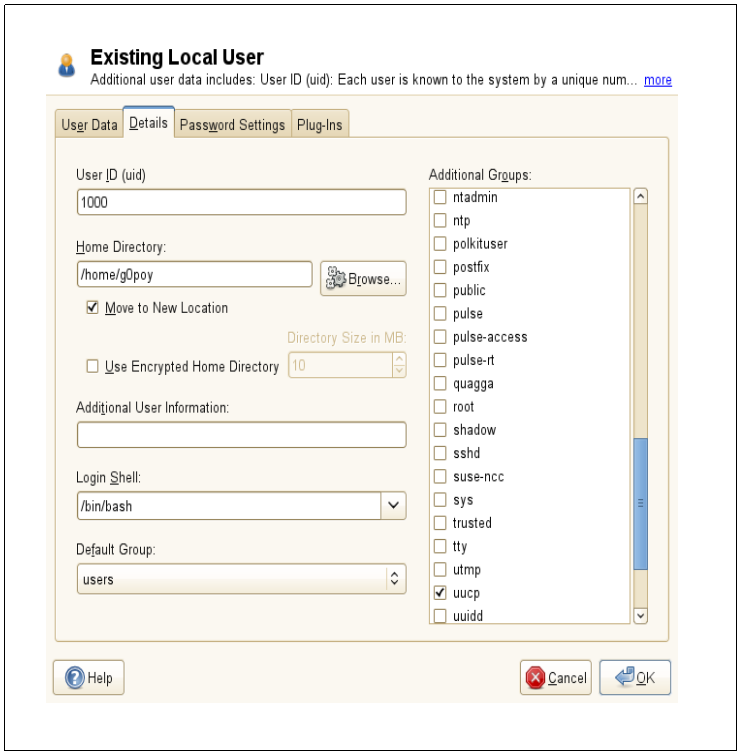
The command line will display a list of the files that we have just copied.

User configuration

In order for the rxtx libraries to work correctly your normal user needs to be given access rights to the uucp group.

Run YAST and select “User and Group Management” which is under the “Security and Users” sub-section.

Select your normal user and then:
click on “edit”
click on the “Details” tab
from the additional groups list find and select the uucp group then click on OK.
The screen-shot below shown the group uucp being added to my main user.



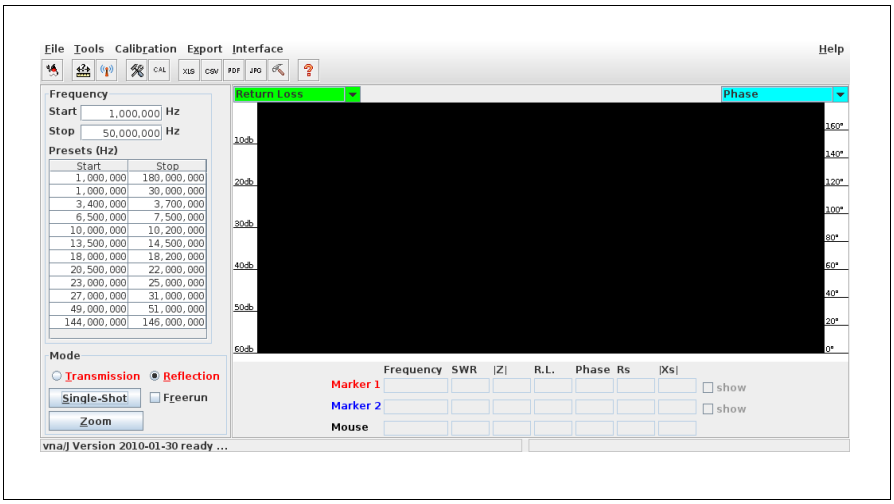
It is now time to do a little testing. As Dietmar suggests, create a VNA-J folder, and place the downloaded vna jar file in it. (At the time of writing the file is vnaJ_1_6_2.jar)

Open the VNA-J folder right click in it and select “open in terminal” a terminal window will now appear.

Enter the command

```
> java -jar vnaJ_1_6_2.jar
```

remember that pressing the tab key will auto complete the command. Press enter, and if all is well you will see the VNA-J GUI appear. You may notice an error message appear, in the terminal but this is due to the pre-release version of the libraries, it is nothing to worry about. However if you get a number of errors and possibly no GUI, then proceed to the troubleshooting section.



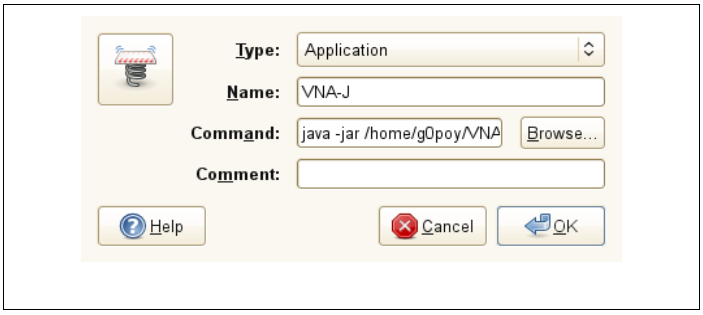
That completes the installation of the java application,

Desktop Launcher

If you wish, move the VNA-J folder off the desktop to wherever you keep your applications. (I just keep it in the root of my home directory)

Right click on the desktop and select “Create Launcher”
Set the following:
Type: Application
Name: VNA-J (or anything you want)

For the command select browse and then navigate to wherever you placed the VNA-J directory, then select and open the application .jar file. This will put the full path into the command box. AT THE FRONT of this path enter the command “java -jar” followed by a space.



The click on OK.
When you click on the VNA-J icon that is now on the desktop, the application will open.
You can amuse yourself by creating a suitable icon for the launcher....

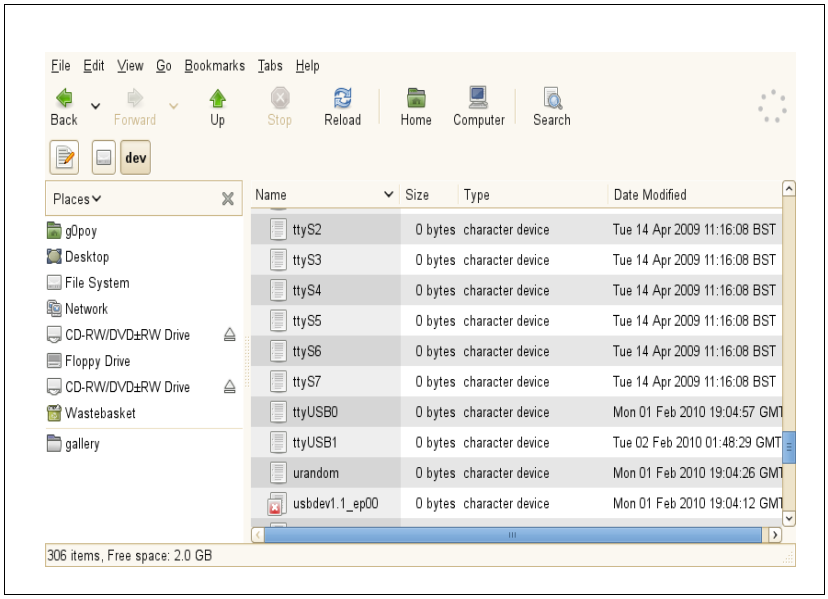
Serial Ports

Linux serial ports are rather restricted in exactly who can access them, I suspect that this is a hang over from when serial ports were the main form of external communication. So there are a couple of things that you MAY have to do. Linux distributions vary in implementation, so you just have to try things out and see what happens.

The first thing to do is identify the serial port that the miniVNA will use. This is really easy to do. The miniVNA uses the USB to serial chip manufactured by FTDI. There is much said about downloading drivers for windows and so on, there is none of that required here, support for the FTDI chip set is built into the Linux kernel. There are several methods of finding out what the serial port of the miniVNA is, in SUSE the easy method is to open up the file explorer and set the view to “list” (there are a LOT of files in /dev, and the list makes them easier to see. Navigate to /dev, and scroll down the list until you find the entries of ttyS5,6,7 etc. These are internal ports.
Now plug in your miniVNA, after a couple of seconds you should see another port appear in the list, it will be ttyUSB0,1,2 and so on. If you only have one USB serial device (the miniVNA) it will be ttyUSB0 In the screen-shot below you can see my own setup, I have an additional USB to serial adaptor, which is ttyUSB0, and in my case the miniVNA appears as ttyUSB1

Under Linux the serial ports are identified by their path, so my serial port is /dev/ttyUSB1

The screen-shot below shows file manager set to my /dev directory, where you can see the two USB serial devices that I have.



A second method to find the serial ports is to use the command line.

Open a terminal and type

```
> cd /dev
```

```
> ls ttyU*
```

then plug in the miniVNA and repeat the command

```
> ls ttyU*
```

you should then see a difference. The additional port is the miniVNA. The output will look something like this.

```
> cd /dev
> ls ttyU*
ttyUSB0
> ls ttyU*
ttyUSB0  ttyUSB1
>
```

Now that you know the serial port that the miniVNA uses, open up the VNA-J application and set the interface to that of the miniVNA. The VNA-J application will automatically list the serial ports that the miniVNA uses, you just have to select the correct one.

Try a single shot run, and see what happens. If it works, then the install is complete.

You may get a port error, this is not too surprising due to the way that Linux restricts access to the serial ports. If you enter the command ls -al ttyUSB* you will get a listing like this

```
> ls -al ttyUSB*
crw-rw-rw- 1 root uucp 188, 0 2010-02-01 19:04 ttyUSB0
crw-rw-rw- 1 root uucp 188, 1 2010-02-04 18:29 ttyUSB1
>
```

The letters at the start define the “privilege” that the “owner”, “group” and “others” have on the port. Each group has the rw- flags set, that means that all three groups have both read and write access to the port. By default the final group of rw- is set to --- That means that “others” have no access to the port. As you can guess, the application is treated as “others”.

This is very easy to correct for a test. You must be root in order to do this so open a terminal, cd to /dev then type su. To become root. The prompt will change to #, indicating that you are the root user.
Type chmod 666 ttyUSB0.(or whichever is your miniVNA) Use the ls -al command to see the change. The output below shows before and after the chmod command is issued.

```
 # ls -al ttyUSB1
crw-rw---- 1 root uucp 188, 1 2010-02-04 18:29 ttyUSB1
 # chmod 666 ttyUSB1
 # ls -al ttyUSB1
crw-rw-rw- 1 root uucp 188, 1 2010-02-04 18:29 ttyUSB1
 #
```

As you can see the final group of rw has been added to the permissions.

Test the VNA-J application again, this time you should not have any port errors, and the application should be able to control the miniVNA.

If you wish you can leave things like this, and just remember to set the permissions every time manually. If you want the system to do this automatically, it is necessary to create what is known as a udev rule. This is not hard, and is well worth doing. Please see the section “udev rules”

SUSE11.2

During the course of creating this document, it has been discovered that there is a bug in the latest release, SUSE 11.2. This bug is not too serious, but it has caused complications with the operation of the rxtx libraries. The bug is reported in Bugzilla

https://bugzilla.novell.com/show_bug.cgi?id=552095

The bare facts of the bug are that:
Serial devices are now owned by root with a group of dialout
The var/lock directory is now owned by root with a group of root

Previously serial devices were root:uucp and var/lock was root:uucp

From this you will see that it is very difficult for anything other than root to write to var/lock, this makes the filelocking rather problematic to say the least. There are several solutions, most of which require changing the way the ports are assigned, and perhaps making var/lock world writeable. I am not happy doing this, as things could change if an update happens due to this bug. If not things will almost certainly change with the 11.3 release.

So the solution that I propose, is to use rxtx libraries that have been compiled NOT to use file locking. There is a configure option within the rxtx distribution that sets this option. As some users will not be happy or know how to re-compile the libraries, I will place a copy of the re-compiled libraries in the miniVNA group file store, along with a copy of this document.

Troubleshooting

Linux is a COMPLEX operating system, and the number of ways that things can interact is staggering. So finding out what is going on can be a challenge. Here are a few tips to help you get over some of the more common problems.

When running applications, IF they generate errors, they need somewhere to send those errors to. Applications that are started via a desktop icon often throw away such error messages. It therefore follows that running the application from a terminal via a command line is the best way to see if there are any error messages. In this case we use the command

```
> java -jar application.jar
```

where application.jar is the VNA-J application.

The next most obvious thing to do is read the error messages.
Below is the output on my installation when run from the terminal.

```
> java -jar vnaJ_1_6_2.jar
WARNING:  RXTX Version mismatch
      Jar version = RXTX-2.2pre1
```

```
      native lib Version = RXTX-2.2pre2
>
```

This error is not critical and is a result of the pre release nature of this version of rxtx.

Error messages relating to file lock issues.
There are two main causes of this to be aware of, the first is that you have not set your normal user to have rights to the uucp group. Also I have found that you often have to logout and login again to make these rights active. (I more often reboot the computer just to really make sure)

The second reason is that you are running SUSE 11.2 and are running foul of the bug. If you are not running SUSE then you may have a Linux distribution that has a similar problem.

IF you cannot modify the system to accommodate the file locking requirements, then as for the SUSE 11.2 issue, use the no flielocking libs.

Running an application as su, is not always the same as running as a normal user!

When the terminal prompt is > you are a normal user, when it is a # you are root. A silly mistake to make, but I've made it myself often enough.

USB ports sometimes work sometimes don't. This will depend on what the permissions on the ports are. If you over-ride them as root from the terminal, they will stay that way until the computer is rebooted, OR the miniVNA is unplugged and plugged back in. To overcome this problem see the section on udev rules.

Some Linux distributions do not have the root account enabled (Ubuntu is one such) In this case root commands have to be preceded with the command sudo. If this becomes tedious, then the command sudo su will create a root terminal from which you can execute commands at root level without the tedium of the sudo command.

Without doubt there are many many more possible errors than I have stated here. So be be prepared to work through them and don't be afraid to ask for help in the group.

Quick Install

This section is intended for experienced Linux system administrators, and so it is rather terse.

Obtain software.

Check java version, install or update to version 1.6.

Place rxtx libraries into /usr/lib

Add uucp group to primary user.

Run the jar application with java -jar from console.

Check for file lock issues.

Resolve any lock issues, may require no file locking libraries.

Check serial port permissions.
Test with chmod 666 via terminal.

Create and install udev rule to automatically fix serial port permissions.

Create desktop launcher

Done.

udev rules

When devices of all sorts are connected to a Linux system, they all need to be inserted into the filesystem, and given names and permissions. It is the function of a set of rules, known as “udev rules” to do this. The system is very easy to manipulate and this provides us with a means to change the permissions of a device when it is detected. As things like the miniVNA are detected as soon as they are plugged in, this is very useful to us.

There is an excellent guide to writing and using udev rules here:
http://www.reactivated.net/writing_udev_rules.html

udev rules are stored in the /etc/udev/rules.d/ directory.
You will see that the rules have a .rules suffix, and a numerical start. udev rules are actioned is numeric order, lowest first.. the contents of my udev.d directory is

```
> ls
10-local.rules                  56-sane-backends-autoconfig.rules
40-alsa.rules                   60-pcmcia.rules
40-bluetooth.rules              65-wacom.rules
40-xen.rules                    70-kpartx.rules
41-soundfont.rules              70-persistent-cd.rules
```

```
51-lirc.rules                  70-persistent-net.rules
51-packagekit-firmware.rules   71-multipath.rules
52-irda.rules                  77-network.rules
55-hpmud.rules                 79-yast2-drivers.rules
55-libsane.rules               99-pcsc_lite.rules
56-idedma.rules
>
```

These files are simple text files, and the first file, 10-local.rules is the file I created to hold my own rules. Currently there is only one rule, which is to sort out the permissions on the USB serial ports.

```
#
# udev rules for local use
# Created by A.Eskelson
# Main use is to allow access to the serial port hardware by users
# Found this to be necessary for WxToImg to control the receiver
#
# This rule sets the permissions of any ttyUSB device, i.e. ttyUSB0,
ttyUSB1
# To 666
#
KERNEL=="ttyUSB*", MODE="0666"
```

That's all it is, 9 lines of comment and one line of rule ! (Be aware of the line wrap on line 7)

It is the line starting KERNEL== that does all the work. All it is doing is looking for any device that when plugged in is assigned the name ttyUSB* the star as a wild card for the number, ensure that it's permissions are set to 0666 (4 digits are required)

If required it is also possible to set the group ownership of the device, to do this the GROUP keyword is used, for example:

```
KERNEL=="ttyUSB*", GROUP="uucp", MODE="0666"
```

However in my system I have not found the GROUP change to be necessary..

All of the following must be done as root.

Open a terminal and cd to the udev rules directory
Create a rule file called 10-local.rules
Edit this file to have the same rule as in my rules, changing the comments to suit your own needs.
Save and exit.

There are several editors available that will work from the root terminal, the old and perfectly functional vi will do the job, also if you type gedit from the command line the normal gnome editor will open in superuser mode.

The simple way to create a file is to “touch” it

.

```
>cd:/etc/udev/rules.d
```

```
su
# touch 10-local.rules
```
edit with vi
```
# vi 10-local.rules
```
or edit with gedit
```
#gedit 10-local.rules
```

Ubuntu Linux

Ubuntu Linux is a very popular Linux distdistro which has gained a large user following, mainly because it is a very much a “install and forget” type of system. Much of the underlying complexity of Linux is hidden from the user. While this is often seen as a good thing, it can make life a little hard in some cases. The installation of the VNA-J application on Ubuntu is fairly easy when looked at in terms of a more standard Linux install, but I must give the normal warnings that the installation involves running with root privilege and as such you can cause havoc with your system, so take images/backups and be careful.

Much of the detail in the SUSE install applies to Linux, so I will not repeat everything again. You will need:

The latest VNA-J application, the rxtx library files librxtxSerial.so and librxtxParallel.so

For the purposes of this install, a totally clean install was performed using Ubuntu 10.0.4 The system was then allowed to perform an on-line update.

Next two desktop folders were created, one called “VNA-J”, and the other called “vvv”

Obtain the latest vnaJ file, at the time of writing it is:
vnaJ.2.5.10.jar. Copy or move this file into the Desktop VNA-J folder.

Obtain the rxtx libraries librxtxSerial.so and librxtxParallel.so, and copy or move these into the Desktop “vvv” folder.

Note that the “vvv” folder is only a temporary folder, and will be deleted once the install is finished.

Now we need to install the libraries. This is where Ubuntu gets a little difficult. In the standard install NO root account is created, so in order to perform root privilege commands or operations they must be preceded with the command sudo. In this case a more convenient method is to open a terminal window and assess the root prompt from there.

So open a terminal and type:

```
# sudo su
```

Enter your password when prompted.
You will e presented with a # at the end of the terminal name, this indicates that you are running as root.

we need to install the libs in /usr/lib

so change directory to /usr/lib

```
# cd  /usr/lib
```

Now check that the libs are not already installed.

```
# ls librxtx*
```

You should get nothing found.

Now copy the libs

```
# cp /home/youruser/Desktop/vvv/*.so /usr/lib/
```

This will copy the libs, to double check repeat the ls command
Replace the "youruser" with whatever your logon name is

```
# ls librxtx*
```

You should see the libs listed on the terminal.

Remember that during this process the TAB key acts as an auto complete and that you also have a history buffer, where you can use the arrow keys to go back to a previously used command.

Type **`exit`** to exit from the root prompt.

Now we need to check and install the java system.

Type

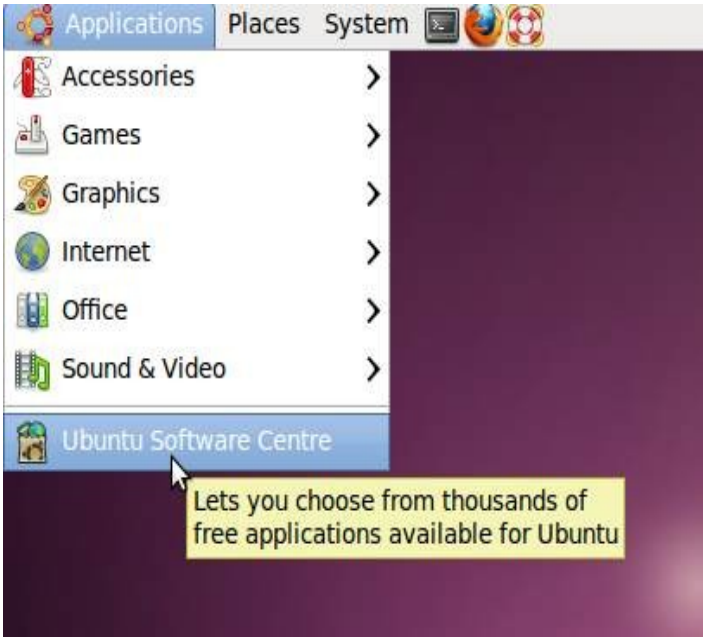
```
java -version
```

You should get a message stating that no java is installed and a message saying that several packages contain java, and these can be installed with the "apt-get" command. DO NOT USE THIS COMMAND.

By default Ubuntu does not come with java so it has to be installed. The later versions of Ubuntu use the Openjava system. You should install this via the Ubuntu Software Centre

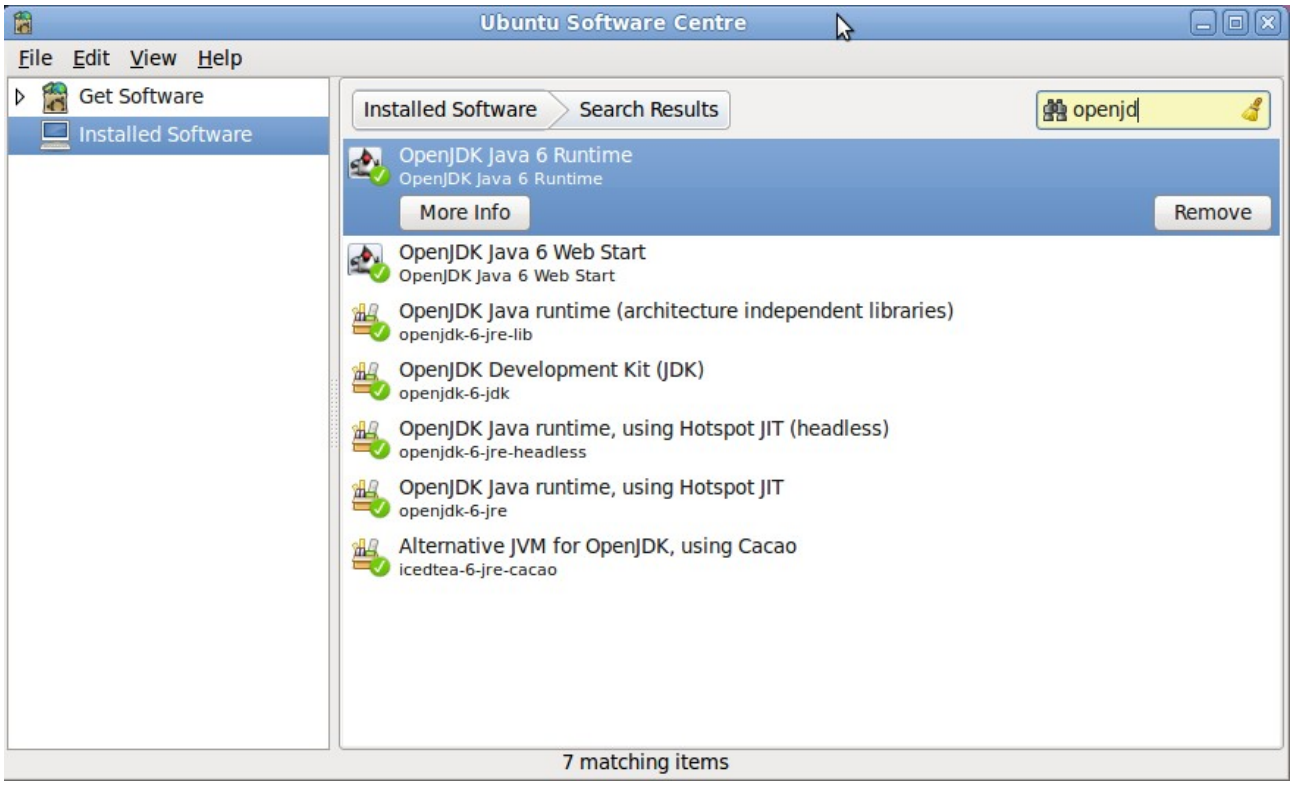
In order to obtain all the necessary libraries, the simplest method is to install the java software development kit,

To access the Ubuntu Software Centre, select applications on the main menu bar, and the software centre is right at the bottom of the list. Select this.



The software centre window will open, type in the search box at the top right hand corner "javaSDK"

You will get a screen that looks something like this. NOTE that this is the INSTALLED software screen, as I have already installed the software. You will see this on the "Get Software" screen, which is the default when the centre opens.



As you can see there will be several options, and the first is always highlighted. THIS IS NOT THE PACKAGE YOU NEED. You need to select the package:

OpenjDK Development Kit (JDK)
openjdk-6-jdk

This is the 4th item down in the above screen shot.

Select this package and then click the install button.

You will be asked for you password during the install process.

Once the install is complete check that you have an active java package. Open a terminal and type

```
java -version
```

You should get a response listing the openjdk as the installed java system.

Once you get to this point the install is essentially complete, so you can test the vnaj application.

In a terminal type
```
cd /home/youruser/desktop/VNA-J
```

This will move you to the folder where the vnaj application is. Than type

```
java -jar vnaJ.2.5.10.jar
```

The VNA-J application should open. There will be an error message regarding rxtx, ignore this.

From this point you have to:
select the correct vna hardware, select the correct com port then follow the calibration setup.

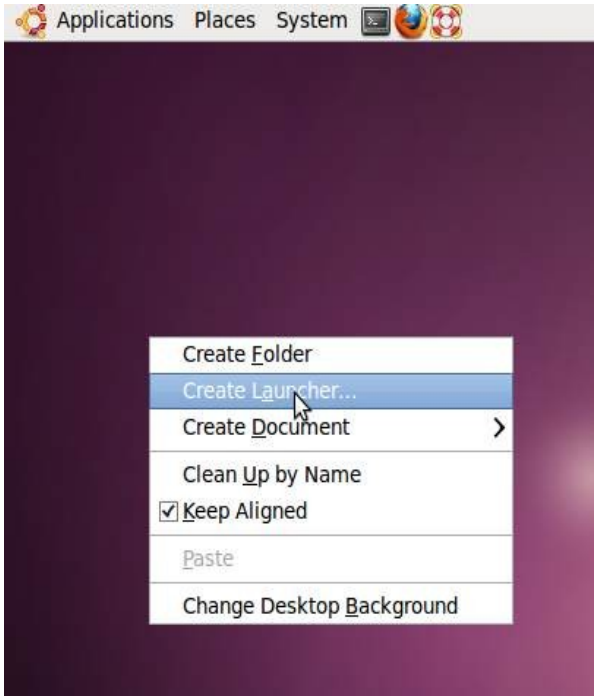
Should you have multiple java installations on your system already. The commands

```
update-alternatives --config java
update-alternatives --config javac
```

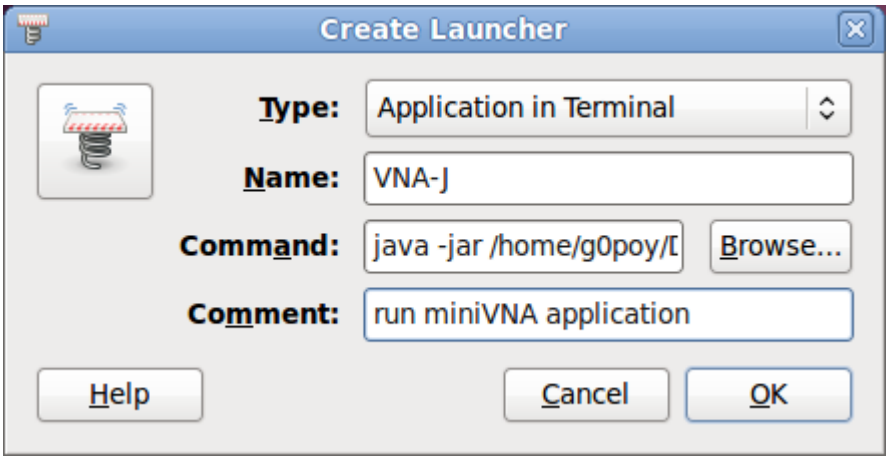
Can be used to switch between them. These commands run as root.
BOTH commands need to be used.

Ubuntu seems to work the com ports in a slightly different way to SUSE, and I have not found it necessary to create udev rules in order to make the ports work. However see the SUSE section for information on this if you run into problems of this sort. Also Ubuntu runs with the normal rxtx libs, if you get errors try the non file locking libs. Again see the SUSE section for more information on this.

It is more convenient to launch the application via a desktop icon. To create this RIGHT CLICK on the desktop and select create launcher



This will open the following window

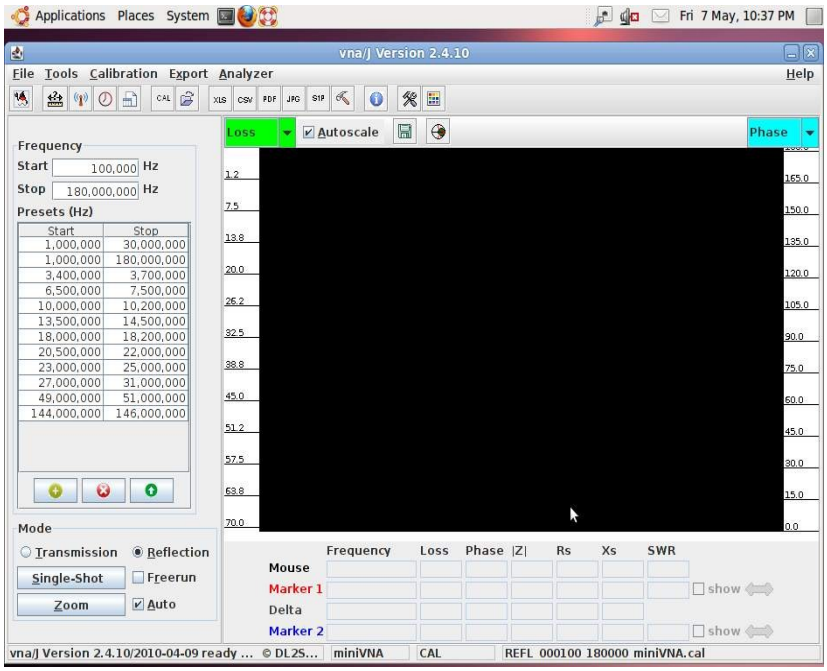


Set the Type to application in terminal
Set the Name to VNA-J
Set the Command to java -jar /home/youruser/Desktop/VNA-J/vnaJ.2.5.19
Set the Comment to whatever you want or leave it blank
Click on OK

You will now see a small icon called VNA-J

You can have fun designing a new icon for it yourself!

Clicking on this launcher will bring up the application. Here it is running:



Acknowledgements

Dietmar Krause DL2SBA	The VNA-J applications
Erik Jakobsen	starting me off creating this document
mRS	The miniVNA
Yahoo Group analyzer_iw3hev	A friendly user group for the miniVNA
rxtx group at qbang.org	provided some helpful comments on the SUSE bug