OpenLab - Open Laboratory

Installation Instruction V3

UAS Technikum Wien
Höchstädtplatz 6, A-1200 Vienna
I: www.technikum-wien.at
E: markus.lechner@technikum-wien.at
E: patrick.schmitt@technikum-wien.at

This work was conducted in context of the public funded project OpenLab managed by MA23 under grant 17-08

Contents

1 OpenLab Signal-Toolkit Hardware Setup

This chapter will give you an overview about the currently available hardware of the OpenLab Signal-Toolkit and how to setup each tool.

1.1 OpenLab Oscilloscope Hardware Setup

Currently there exists two versions of the OpenLab Oscilloscope hardware:

- Version 2 and 3: First official OpenLab Oscilloscope hardware, small PCB, support for uC platforms, components on both sides
- Version 4: Latest official OpenLab Oscilloscope hardware, bigger PCB, much improved signal to noise ratio, supports only DE0, components only on one side

In order to continue with the setup process, it is necessary to find out which version of the Oscilloscope hardware is being used.
It is only necessary to distinguish between version 2/3 and version 4. The changes between version 2 and 3 do not affect this setup guide.

The images below should help you to recognize your oscilloscope hardware:

TODO: INSERT IMAGES OF V3 AND V4

Please continue with the subsection according to your HW version.

1.1.1 Oscilloscope V2/3

The image below illustrates an overview of the assembled OpenLab Oscilloscope (V2/3) utilizing the FPGA DEO board.
Before you start with your measurements, please connect the PCB to the male header GPIO1. **Please do not try to plug the PCB in GPIO2!**
Connect the serial FTDI cable to the hardware and subsequently to a free USB port.
Finally, the FPGA can be started by pressing the button SW10 (red button).

The software configuration is described in chapter 2 (including its sections and subsections).

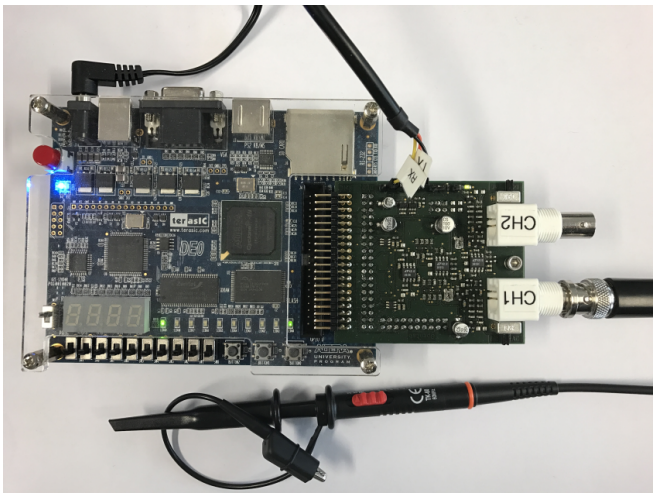
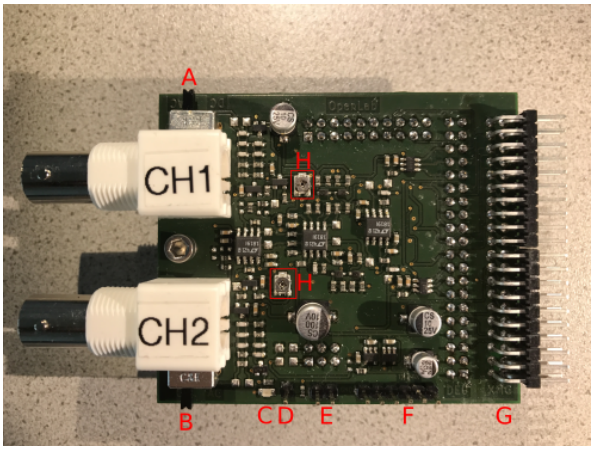
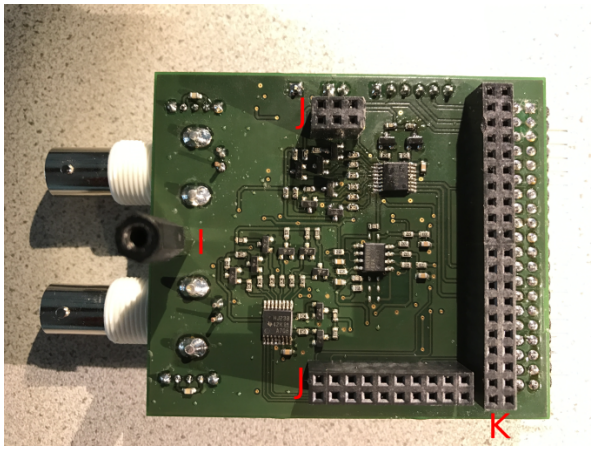


Figure 1.1: Hardware Overview (V2/3 attached to DE0 board)

The following images are intended to give an overview of the PCB.



(a) Hardware Front View



(b) Hardware Back View

Figure 1.2: OpenLab Oscilloscope Hardware PCB V2/3

- A: AC/DC Switches CH1
- B: AC/DC Switches CH2
- C: Power LED
- D: Probe Compensation
- E: GND Connector
- F: Serial Interface Connector (GND | - | - | TX | RX | -)
- G: FPGA DE0 Connector
- H: Offset Correction - Do not adjust this trimmer!
- I: Threaded Standoff
- J: TIVA-C Connectors
- K: XMC4500 Connector

The FTDI cable needs to be connected to the **'Serial Interface Connector' - H**, shown in the image below.

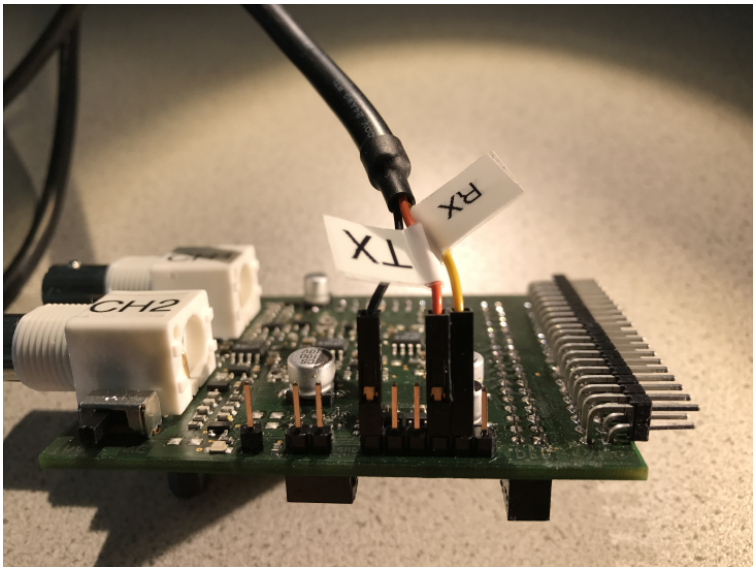


Figure 1.3: Hardware Serial Interface

1.2 OpenLab Soundcard-based Signal Generator Hardware Setup

The OpenLab soundcard-based signal generator is a single-board solution which plugs into your workstations USB- and audio-port. The USB connection is used for power delivery and transmitting settings from the graphical user interface. In order to setup the HW it is necessary to connect your soundcard using a 6.3mm jack-plug cable to the audio port of the signal generator. Furthermore connect the signal generator with your workstation using the USB cable. Finally check if the power LED is on.

The following images show the different plugs, buttons and important circuit parts of the OpenLab soundcard-based signal generator.

TODO: INSERT IMAGES OF SIGGEN

Continue with the software setup described in chapter 2 and its according sections.

1.3 OpenLab U-I Source Hardware Setup

In order to prepare the hardware of this tool, just plug the single board to your workstation using a USB cable. The red power LED will turn on if the device is ready for further operations.

The images below describe some relevant parts of the U-I source hardware.

TODO: INSERT IMAGES OF U-I SOURCE

Now you are ready to continue with the software setup shown in chapter 2.

1.4 OpenLab Multimeter Hardware Setup

TODO: WRITE SECTION

2 OpenLab Signal-Toolkit Software Setup

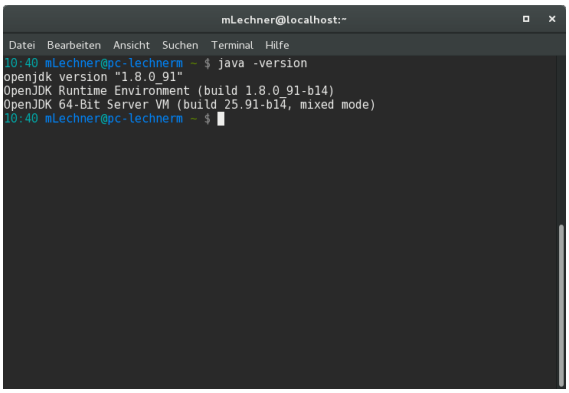
The following installation instruction explains in detail the necessary steps to get started with the Signal-Toolkit applications. The latest version of all applications support the operating systems Linux and Windows. (no official MAC support) In circumstances, the installation process can only be completed if the user has administrator privileges. (depending on the tool which should be started)

If any OpenLab application wants to be started on a computer of the UAS (University of Applied Sciences), please boot from the Embedded-Systems partition.

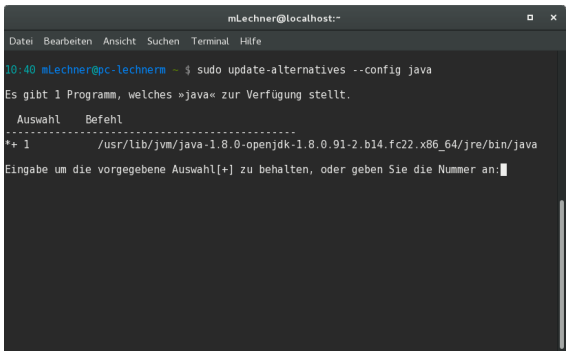
The installation process for each tool is shown in the following sections.

2.1 ALL TOOLS - Java installation (Linux)

The first step of the installation instruction is to verify the currently utilized Java version. In Figure 2.1a the Linux terminal is used with the command **java -version** to display the configured Java environment. If the output of the terminal is *java version "1.8.0_xxx"*, continue with section 2.3. The output of the terminal is *openjdk version "1.8.0_91"*, thus it is necessary to prove which Java program is available. In order to change the default Java environment, enter the following command **sudo update-alternatives --config java**. Figure 2.1b shows that only the *openjdk version "1.8.0_91"* is available. Therefore, it is mandatory to install either the Java Runtime Environment (JRE) or the Java Developer Kit (JDK).



(a) Java version: openJDK



(b) Java alternative configuration

Figure 2.1: Java version and program verification

This guide only describes how to install the JDK version. This is the recommended way. If you really wish to only install the JRE version please ask one of the OpenLab members for further assistance. *(We prefer to install the JDK package of Java in order to enable further development for your studies and it does not have any negative effects anyways)*

Before the installation of JDK, it is mandatory to determine the architecture of the used operating system. Open a terminal and enter **uname -m**. If the architecture is x86_64 (64 Bit) please

continue with the installation by downloading a 64 Bit JDK installation package for your operating system. If not, you can try to install a 32 Bit JDK but this is not recommended. You can also try to upgrade to a 64 Bit Operating System as 32 Bit systems a generally deprecated. (Contact one of the OpenLab Team members if you need assistance)

The following link will bring you to the Oracle download page: `http://www.oracle.com/technetwork/java/javase/downloads/index.html`

Next perform these steps to download the correct version:

- Search for the **Java SE 8uXXX** entry on that page. (XXX will change according to the currently available subversion)
  Hint: *Please make sure that you download Java 8! The OpenLab Signal-Toolkit currently does not run on Java 9 or Java 10!*
- There will be 3 different buttons at the right side near that entry: JDK, Server JRE and JRE –> Please click on **JDK** to continue (see figure 2.2)
- On the next page we have to search for the entry with the highest version number available (normally the entry in the middle of the page) (see figure 2.3)
- Click on the radio-button beside **Accept License Agreement**
- The table below lists all supported Operating Systems for the Java SE package. Search for **Linux x64** there should be two entries.
- For each entry there is a download link on the right side. One ends with .rpm and the other with .tar.gz
- For Red Hat based distros (Fedora, Korora,...) please click on the file name ending with **.rpm** For Debian based distros (Debian, Ubuntu, Mint, ...) please click on the file name ending with **.tar.gz**

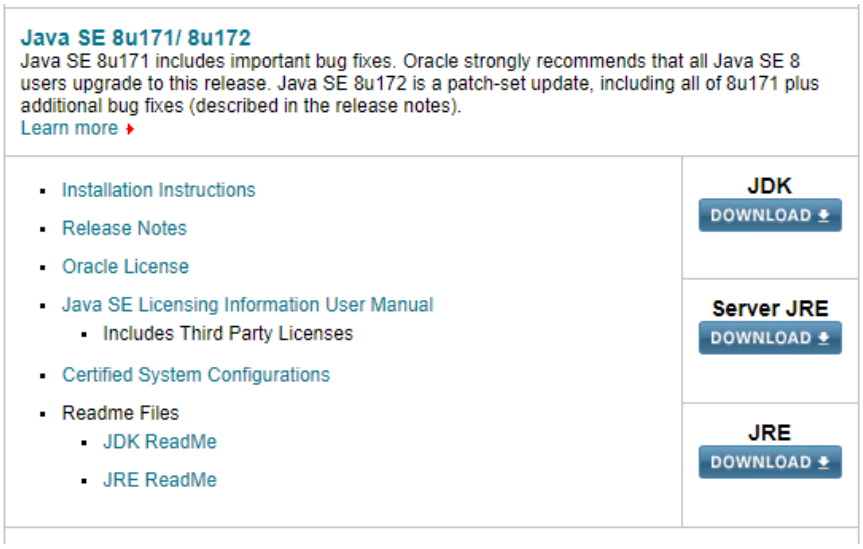


Figure 2.2: Download Java 8 JDK from Oracle - select JDK for Java 8

Java SE Development Kit 8u172

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.99 MB	jdk-8u172-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.9 MB	jdk-8u172-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.07 MB	jdk-8u172-linux-i586.rpm
Linux x86	184.91 MB	jdk-8u172-linux-i586.tar.gz
Linux x64	167.15 MB	jdk-8u172-linux-x64.rpm
Linux x64	182.08 MB	jdk-8u172-linux-x64.tar.gz
Mac OS X x64	247.87 MB	jdk-8u172-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	140.05 MB	jdk-8u172-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.35 MB	jdk-8u172-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	140.63 MB	jdk-8u172-solaris-x64.tar.Z
Solaris x64	97.06 MB	jdk-8u172-solaris-x64.tar.gz
Windows x86	199.11 MB	jdk-8u172-windows-i586.exe
Windows x64	207.3 MB	jdk-8u172-windows-x64.exe

Figure 2.3: Download Java 8 JDK from Oracle - choose correct installation package for OS

After the download is complete please continue with the corresponding subsection.

2.1.1 Installing JAVA using a RPM package (usually Red Hat based distributions Fedora, Korora,...)

For installing the Java developer kit (JDK), on RPM package based distributions, download the RPM file from the link mentioned in the previous section. After the download completed the package can be installed by invoking a terminal pointing to the location of the just downloaded file. By using the following command the installation process can be started: **sudo rpm -Uvh jdk-8uxxx-linux-x64.rpm**. Replace the xxx with the actual version number of the Java RPM package. An example is shown in figure 2.4.

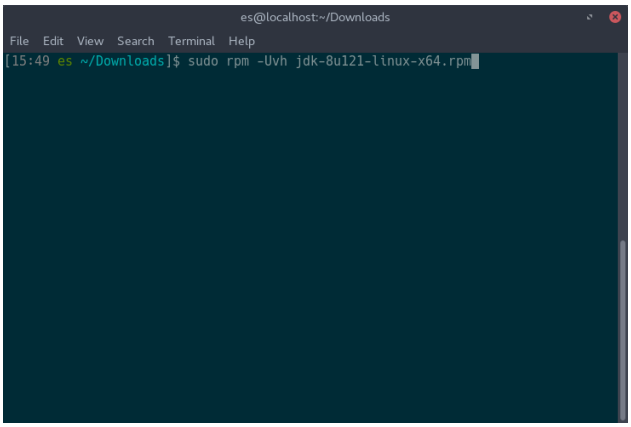


Figure 2.4: Installing the Java RPM package by invoking the RPM command

After the successful installation of JRE or JDK, it is necessary to verify again which Java programs are available, using **sudo update-alternatives --config java** illustrated in Figure 2.5.

Finally, it is only necessary to select, in this case, the second Java program which is *java version "1.8.0_111"*. The same command is entered again to verify whether the correct Java environment has been chosen.

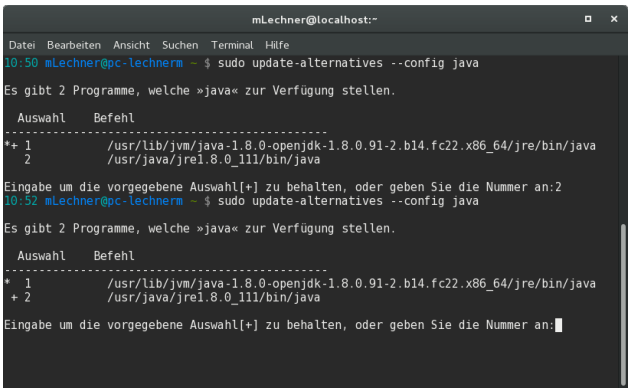


Figure 2.5: Java alternative configuration with installed JRE or JDK

A successful installation of Java can be checked by entering the **java -version** command. The output should be similar to figure 2.6. (except the version number)

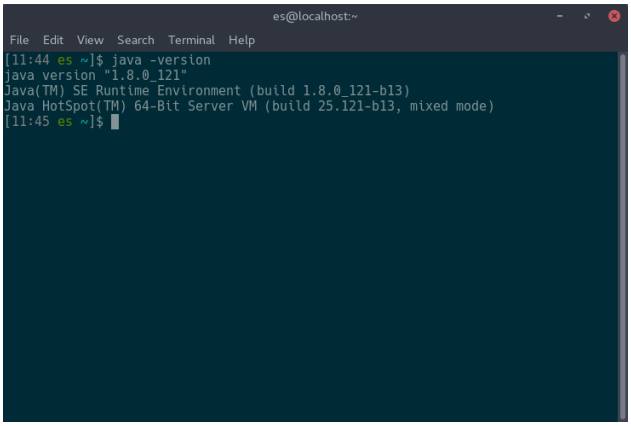


Figure 2.6: Output of "java -version" after correct installation

Verify that the output does not contain *OpenJDK* in any line.

2.1.2 Installing JAVA using a tar.gz package (usually Debian based distributions Ubuntu, Mint,...)

For installing the Java developer kit (JDK) via a tar.gz package, download the tar.gz file from the link mentioned in 2.1. In order to copy the necessary files to a proper location, a folder has to be created using the following command: **sudo mkdir /opt/jdk**

This procedure can be seen in figure 2.7.

Next, unpack the tar.gz file which was previously downloaded by entering the following command into a terminal window: **sudo tar -zxf jdk-8u121-linux-x64.tar.gz -C /opt/jdk**
(assuming the terminal already points to the folder locating the tar.gz file)
Hint: *Keep in mind that the version number could differ from the example!*

This command will take a while depending on the used machine. There is no indication of progress. The process is completed after the terminal draws a new line. Any errors will be displayed accordingly.

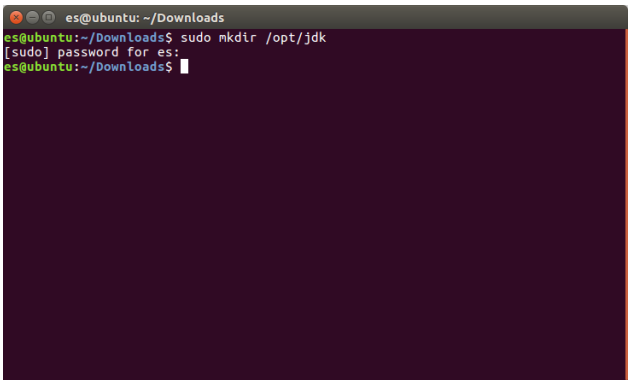


Figure 2.7: Creating a folder for the Java installation using a tar.gz package

In order to verify the successful unpack process, the following command should output a folder which is named after the installed java version: **ls /opt/jdk**

An example of those commands can be seen in figure 2.8.

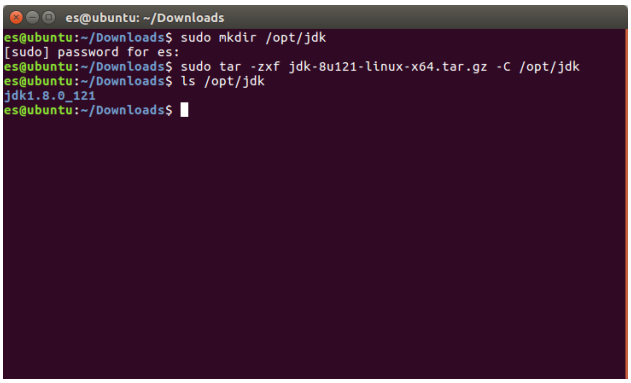


Figure 2.8: A summary of the commands used to install java utilizing a tar.gz package

The final step is to tell the system to actually use the java executable, located in the folder which was previously created, whenever the Java command is invoked. The following two commands configures the system accordingly:

- *sudo update-alternatives --install /usr/bin/java java /opt/jdk/jdk1.8.0_X/bin/java 100*
- *sudo update-alternatives --install /usr/bin/javac javac /opt/jdk/jdk1.8.0_X/bin/javac 100*

Replace X with the version number of the, to be installed, Java package.

A successful installation of Java can be checked by entering the **java -version** command. The output should be similar to figure 2.9. (except the version number)

Verify that the output does not contain *OpenJDK* in any line.

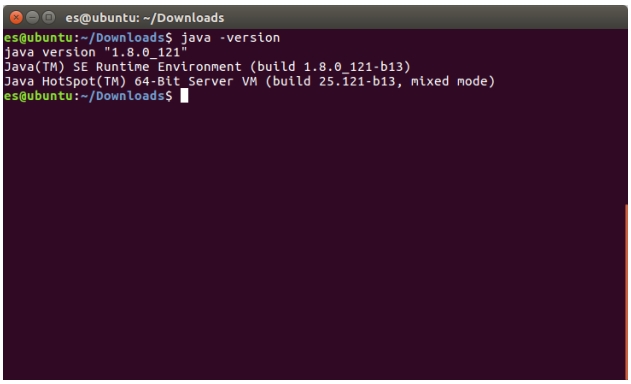


Figure 2.9: Output of "java -version" after correct installation

2.2 ALL TOOLS - Execution of the Signal-Toolkit Application (Linux)

The Signal-Toolkit application can only be launched with the command
java -jar OpenLab_SignalToolkit_VERSIONNUMBER
whereas it is necessary to be in the same directory as the Signal-Toolkit application.
(replace VERSIONNUMBER with the actual number of the filename!)

2.3 Oscilloscope and SoundCard Signal Generator - Serial Device Permissions (Linux)

After the installation process of Java, it is mandatory to allow a non-default user to use serial devices such as ttyUSBx. Hence, a terminal program needs to be opened and the following command should be entered: **sudo usermod -a -G dialout MY_USER_NAME**. To get the current user name, enter **echo "$USER"**. A reboot is necessary before this configuration will be effective.

2.4 U-I Source and Multimeter - USB HID RAW access permissions (Linux)

In order to gain access to USB HID devices (RAW-mode) as a standard user, it is necessary to create a so called udev rule. Those rules define what action should be performed if a device is connected to a linux workstation. Because linux blocks access to USB HID raw to any user by default, we need to tell the system that specific devices are allowed to be accessed without root privileges.

We only want to allow specific USB devices (in this case the OpenLab tools) to be enabled for HID raw access. To do so, we detect the OpenLab devices by searching for their specific vendor and product ID, as well as their manufacturer name and serial number. Only if those values from the connected USB device are equivalent to the values of a real OpenLab device, access is granted. With this procedure we prevent other devices from getting USB HID raw access.

In order to create such udev rules we need to define them by adding a .rules file to */etc/udev/rules.d/*.
But first we need to investigate which .rules files are already present in this directory. In order to do so, please start a terminal window and enter:

ls -la /etc/udev/rules.d/
Dependent on your linux configuration, you will see the output shown in figure 2.10:

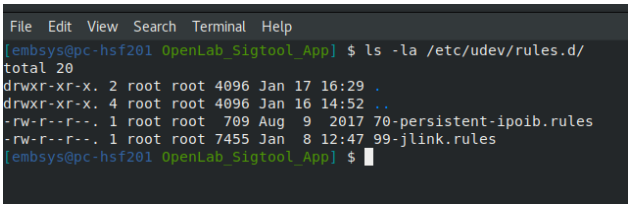


Figure 2.10: Shows the content of the "/etc/udev/rules.d"-folder

The output shows that two .rules files are already present. A new file hast to start with a different number at the beginning of the filename. However, the number should not be higher than 99. If 99 is already taken we prefer to choose the next available lower number. (in this case 98)

To install a new .rules file choose between two methods:

2.4.1 Method 1: Create a new udev rule by copying a pre-configured OpenLab file

First open a terminal and change to the *OpenLab_Sigtool_App* folder using **cd**. Then check the content of this folder by using **ls -la**. There should be a folder called *Setup_files*, change into that folder.
Now enter **sudo cp 98-hidrawOPENLAB.rules /etc/udev/rules.d/**
Hint: *Please check before you copy the .rules file that the number in the beginning of the file name (98 if unchanged) is not already taken by another .rules file!*

Your terminal window should look similar to figure 2.11:

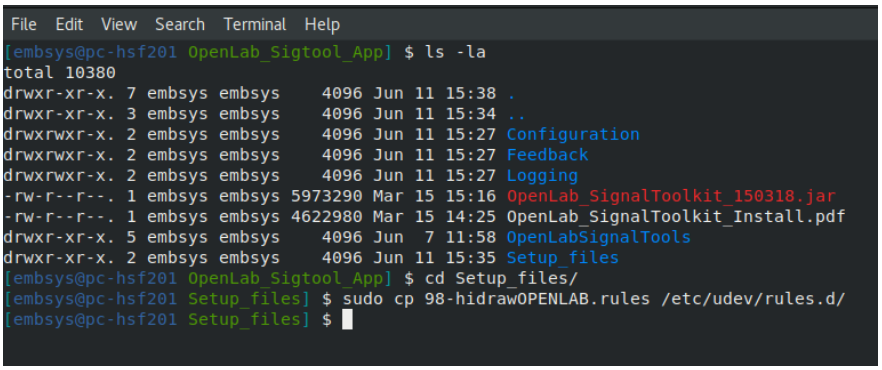


Figure 2.11: Shows the commands entered for copying the OpenLab USB HID .rules file

Continue by entering your sudo password (there will be no characters visible during typing)
Finish up by restarting your system and launching the desired OpenLab application. You should now have access to the OpenLab tools which use HID raw transfer.

2.4.2 Method 2: Create your own udev .rules file using nano (or other editor)

In order to write your own .rules file, open a terminal window and enter:
sudo nano /etc/udev/rules.d/99-hidrawOPENLAB.rules

Hint: *Change the beginning number of the file name (in the above example "99") if this number is already taken by another .rules file as described in 2.4! If 99 is taken choose 98 and so on...*

Now the system will ask you for your sudo password. Enter the password (there will be no characters displayed doing typing) and hit Enter. The terminal now loads the nano editor with a new empty file. In this file we write the following two lines in order to allow access to the OpenLab devices:

```
KERNEL=="hidraw*", ATTRSmanufacturer=="OPENLAB FHTW ", ATTRSidVendor=="1fc9", ATTRSidProduct=="0003", MODE="0666"
KERNEL=="hidraw*", ATTRSserial=="OpenLab-Multimeter", ATTRSidVendor=="1fc9", ATTRSidProduct=="0081", MODE="0666"
```

Your terminal window should now look like figure 2.12.

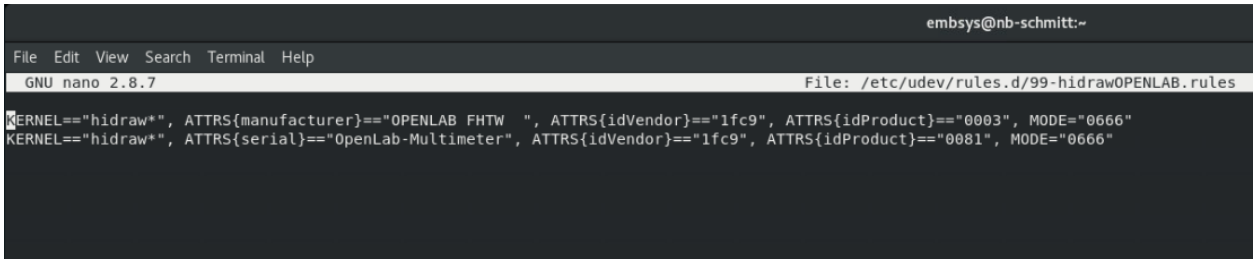


Figure 2.12: Shows the content of the newly created OpenLab USB HID .rules file

In order to close nano and safe the file press **CTRL+X** and answer with **y** or **Y**. Nano will now ask for a filename. This should already be correctly set by the program. Just hit **ENTER** to continue. The .rules file is now created. In order to check the correct creation of the file, we can enter **ls -la /etc/udev/rules.d/**. Now there should be a XX.hidrawOPENLAB.rules file (XX is the placeholder for your number) listed in the output. The output should be similar to figure 2.13.

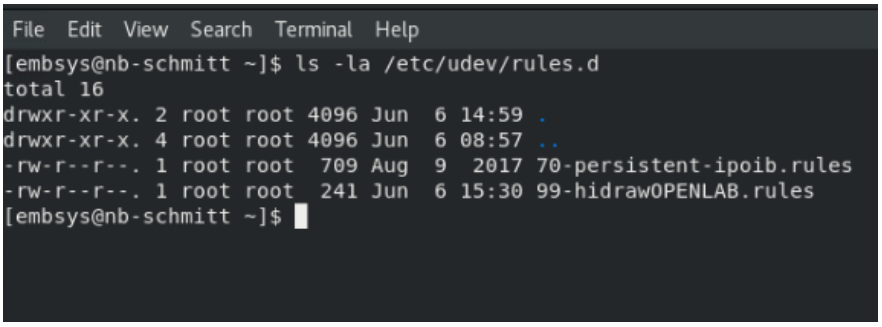
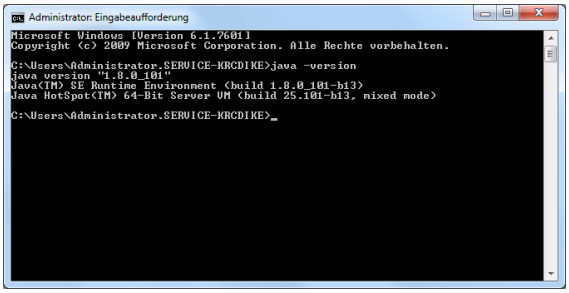


Figure 2.13: Shows the content of the "/etc/udev/rules.d"-folder after creating the .rules file

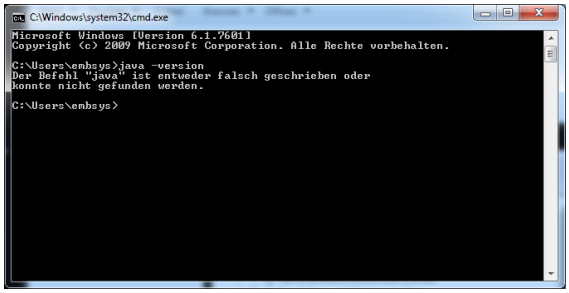
Finish up by restarting your system and launching the desired OpenLab application. You should now have access to the OpenLab tools which use HID raw transfer.

2.5 ALL TOOLS - Java Installation (Windows)

The first step of the installation instruction is to verify the currently utilized Java version. In Figure 2.14, the Windows command prompt is used with the command **java -version** to display the configured Java environment. If the output of the prompt is *java version "1.8.0_xxx"*, shown in Figure 2.14a, continue with section 2.6.



(a) Java version: "1.8.0_101"



(b) Java program not available

Figure 2.14: Java version verification

In case no Java environment is installed, the prompt outputs that the command could not be found, shown in Figure 2.14b.
Therefore, it is mandatory to install either the Java Runtime Environment (JRE) or the Java Developer Kit (JDK).

This guide only describes how to install the JDK version. This is the recommended way. If you really wish to only install the JRE version please ask one of the OpenLab members for further assistance. *(We prefer to install the JDK package of Java in order to enable further development for your studies and it does not have any negative effects anyways)*

The following link will bring you to the Oracle download page: `http://www.oracle.com/technetwork/java/javase/downloads/index.html`

Next perform these steps to download the correct version:

- Search for the **Java SE 8uXXX** entry on that page. (XXX will change according to the currently available subversion)
  Hint: *Please make sure that you download Java 8! The OpenLab Signal-Toolkit currently does not run on Java 9 or Java 10!*
- There will be 3 different buttons at the right side near that entry: JDK, Server JRE and JRE –> Please click on **JDK** to continue (see figure 2.15)
- On the next page we have to search for the entry with the highest version number available (normally the entry in the middle of the page) (see figure 2.16)
- Click on the radio-button beside **Accept License Agreement**
- The table below lists all supported Operating Systems for the Java SE package. Search for **Windows x64** there should be one entry.
- For this entry there is a download link on the right side. It ends with windows-x64.exe –> click on the file to download the installation package

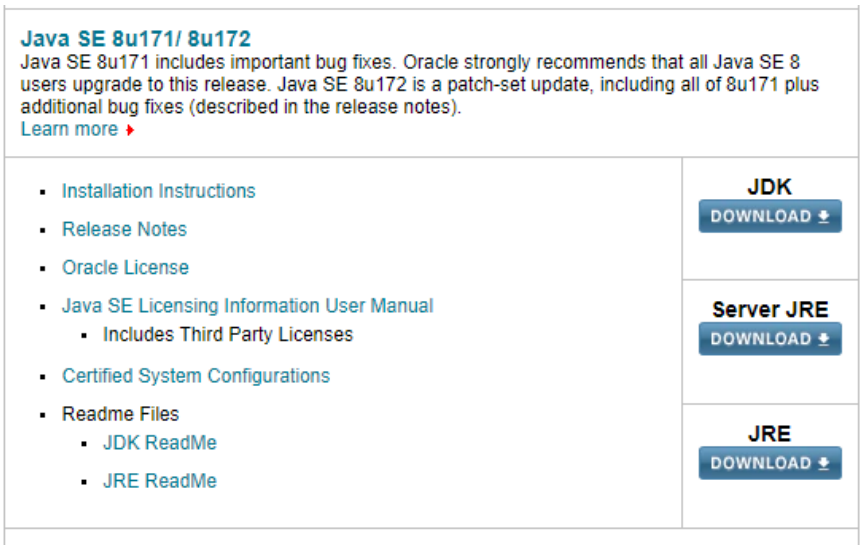


Figure 2.15: Download Java 8 JDK from Oracle - select JDK for Java 8

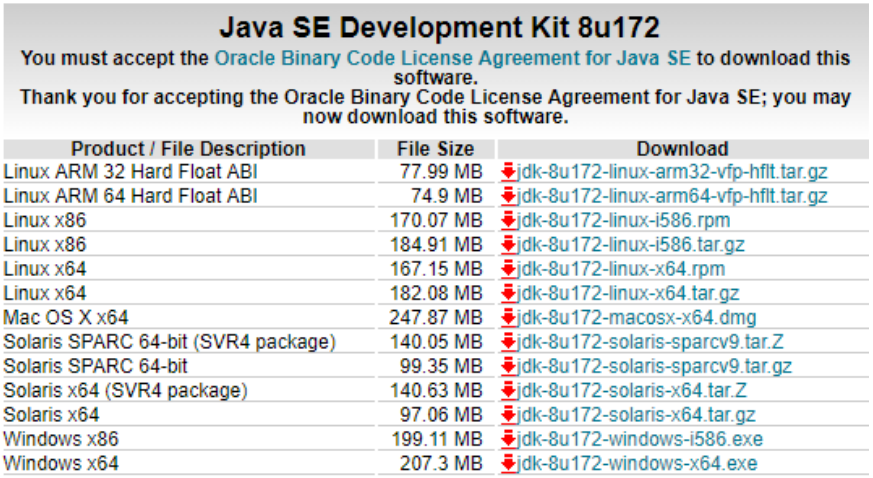


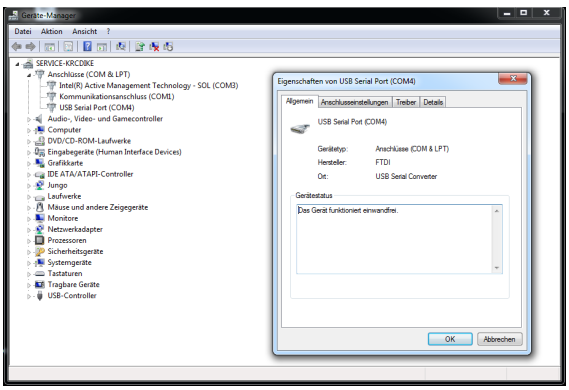
Figure 2.16: Download Java 8 JDK from Oracle - choose correct installation package for OS

After the download is complete please install the software by double clicking on the .exe file and following the instructions of the installer.

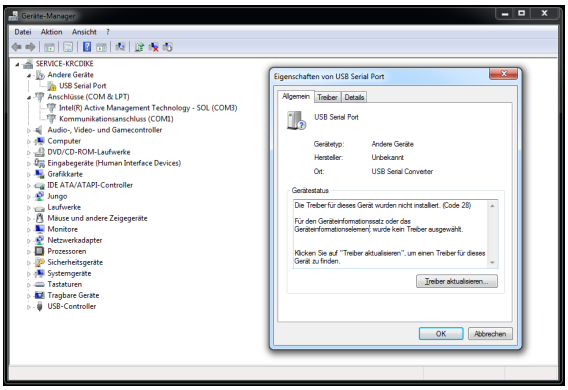
2.6 Oscilloscope - FTDI Driver Verification (Windows)

The following steps are necessary to verify whether the FTDI driver has been installed correctly. Plug the FTDI cable in a free USB port and make sure that the TXD, RXD, and GND cables are not connected to any device. In Figure 2.17a, the FTDI driver was installed correctly and the installation instruction can be continued with section 2.7.

Figure 2.17b illustrates that the USB serial device could not be installed. Therefore, it is necessary to install the FTDI driver manually. Open the device manager and make a right click on the unknown device (USB Serial Port) and select 'Update Driver Software' from the context menu. Subsequently, click 'Browse my computer for driver software'. The driver files are in the folder 'FTDI_Driver'. After the installation process, the serial device should work without any problems.



(a) Successful FTDI driver installation



(b) FTDI driver installation failed

Figure 2.17: Device Manager - FTDI driver installation verification

2.7 ALL TOOLS - Execution of the Signal-Toolkit Application (Windows)

The Signal-Toolkit application can be opened with a double click. If the Signal-Toolkit is not opened correctly, make sure you utilize the *Java(TM) Platform SE binary* program to execute Java applications. The Signal-Toolkit application can also be launched with the command
java -jar OpenLab_SignalToolkit_VERSIONNUMBER
whereas it is necessary to be in the same directory as the Signal-Toolkit application.
(replace VERSIONNUMBER with the actual number of the filename!)