Compiling and Installing Sigrok

Introducation

Having recently acquired a logging multimeter and a logic analyzer, I wanted to use *sigrok* to communicate with them from a PC. I am running Linux Mint 18.1 and 18.2 and have found that the sigrok package provided from the repository was significantly out of date and I was getting segmentation errors when trying to run sigrok, which did not seem to be able to connect to my devices. The solution was to install and compile the latest versions from the sigrok website.

For most part, the instructions are provided within the README file in each of the distribution packages and the git repositories. The file README.devices in *libsigrok* was also helpful in getting *sigrok-cli* working without errors relating to permissions in user mode.

It is sometimes difficult to identify exactly what dependencies need to be installed and I received some useful help from the sigrok developers in order to get the *libsigrok* and *PulseView* compiled properly. The developpers can be contacted on IIRC at #sigrok@freenode, or via the sigrok mailing list on Sourceforge. I made some notes along the way in case I need to repeat the steps on another computer.

Downloading current versions of the software

Current versions can be downloaded from:

https://sigrok.org/wiki/Downloads

Download the following archives:

libserialport
libsigrok
libsigrokdecode
sigrok-cli
PulseView

Sigrok-meter is currently a work in progress and sources can presently only be downloaded via github. Further details can be found here:

https://sigrok.org/wiki/Sigrok-meter

Unpack each package using tar, for example:

```
tar -xzvf libserialport-0.1.1.tar.gz
```

Do the same for all other packages.

If you are using a cloned logic analyzer from the far east, you will also likely require the appropriate firmware. There is a firmware package called sigrok-firmware-fx2afw on the official website which contains a number of firmware images for such devices.

See https://sigrok.org/wiki/Fx2lafw for further details

The firmware files need to go in /usr/share/sigrok-firmware.

Uninstalling the old sigrok packages

If you have installed sigrok packages from your Linux distribution archive, then before the up-to-date packages from the official website can be installed, the old distribution packages need to be removed:

On Mint/Ubuntu/Debian it was simply a matter of doing:

```
$ sudo apt-get remove sigrok
$ sudo apt-get remove sigrok-cli
$ sudo apt-get remove pulseview
$ sudo apt-get autoremove
```

The last command will automatically remove any dependencies that are no longer required.

Pre-requisites for the main modules

Before compilation can begin, it is necessary to install the gcc compiler, if you have not already done so:

```
$ sudo apt-get update
$ sudo apt-get install gcc gcc-multilib
$ sudo apt-get install cmake g++
```

Note: cmake and g++ will be required to compile PulseView.

In order to meet the minimum dependency requirements we will also require some libraries:

```
$ sudo apt-get install python3 python3-dev
$ sudo apt-get install libglib2.0 libglib2.0-dev
$ sudo apt-get install libzip4 libzip-dev
$ sudo apt-get install libusb-0.1-4 libusb-dev
$ sudo apt-get install libftdi1-2 libftdi1-dev
$ sudo apt-get install libieee1284-3 libieee1284-3-dev
```

These are the minimum dependencies and are enough to start with. However, in order to compile modules such as PulseView and sigrok-meter, additional dependencies will be required. These are covered in each individual section.

Compiling the main packages

libserialport

Libserialport is a necessary pre-requisite for libsigrok so needs to be compiled first:

```
$ cd libserialport-0.1.1
$ ./configure
$ make
```

To install the package:

```
$ sudo make install
```

libsigrokdecode

```
$ cd libsigrokdecode-0.5.0
$ ./configure
$ make
$ sudo make install
```

libsigrok

Libsigrok is required for sigrok-cli, PulseView and sigrok-meter. At this point, we compile it to get sigrok-cli working. PulseView and Sigrok meter require further dependencies to be installed which are discussed in the appropriate section.

```
$ cd libsigrok-0.5.0
$ ./configure
$ make
$ su
$ make install
$ exit
```

Note: for some reason, installation was not successul when using sudo make install, however using su to get a root prompt (#) and making the install from there did seem to work consistently. We then go back to the user prompt ($) with exit to continue the next step.

sigrok-cli

```
$ cd sigrok-cli-0.7.0
$ ./configure
$ make
$ sudo make install
$ ln -s /usr/local/bin/sigrok-cli /usr/bin/sigrok-cli
```

Note: the last step seemed to be necessary otherwise the command would not run.

It should now be possible to run sigrok-cli and test your device.

Note: On one of my computers, the following error resulted:

```
$ sigrok-cli
sigrok-cli: error while loading shared libraries: libsigrok.so.4: cannot open
shared object file: No such file or directory
```

Running cat /etc/ld.so.conf.d/* showed that the directory /usr/local/lib was present in the ldconfig list:

```
.
# libc default configuration
/usr/local/lib
.
```

Nevertheless, running the following command fixed the problem:

```
$ sudo ldconfig
```

Running LD config is the preferred option, but as a workaround it should also be possible to fix this with one the following:

- export LD_LIBRARY_PATH = /usr/local/lib
- add /usr/local/lib to /etc/environment
- add /usr/local/lib to ~/bashrc

Testing sigrok

Connect the device to the computer. If its a USB device then check whether the device has been detected:

```
$ lsusb
Bus 002 Device 004: ID 10f1:1a28 Importek
Bus 002 Device 003: ID 0a5c:21b4 Broadcom Corp. BCM2070 Bluetooth 2.1 + EDR
Bus 002 Device 002: ID 8087:0020 Intel Corp. Integrated Rate Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 138a:0005 Validity Sensors, Inc. VFS301 Fingerprint
Reader
Bus 001 Device 004: ID 0820:0001
Bus 001 Device 002: ID 8087:0020 Intel Corp. Integrated Rate Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

In the example above, a Brymen BM869s multimeter was connected using the official BU86X cable. The device is detected with the vendor/device ID (ID 0820:0001) provided, but no description is displayed.

Next run sigrok-cli. First lest see if the device is detected by sigrok:

```
$ sigrok-cli --scan
The following devices were found:
brymen-bm86x - Brymen BM869 with 2 channels: P1 P2
demo - Demo device with 12 channels: D0 D1 D2 D3 D4 D5 D6 D7 A0 A1 A2 A3
```

The demo device indicates that sigrok is working properly. In addition, our Brymen multimeter has been recognized as ‘Brymen BM869’ and the driver required has been identified as ‘brymen-bm86x’. The meter has two channels – P1 and P2. To get more information about the device we can run:

```
$ sudo sigrok-cli -d brymen-bm86x:conn=0820.0001 --show
[sudo] password for johnc:
Driver functions:
    Multimeter
    Continuous sampling
    Sample limit
    Time limit
Scan options:
    conn
brymen-bm86x - Brymen BM869 with 2 channels: P1 P2
Supported configuration options:
    (null):
    continuous: on, off
    limit_samples: 0 (current)
    limit_time: 0 (current)
```

This command ennumerates further details about the meter. We specified the identified driver using the -d flag. We also specified the connection parameter using the ‘conn=’ attribute. The parameters used are the detected vendor and device ID, but notice that a period (.) rather than a colon (:) is used as a separator. To capture some samples we should now be able to run:

[Example 1]

```
$ sudo sigrok-cli -d brymen-bm86x:conn=0820.0001 --samples 5
[sudo] password for xxxxx:
P1: 0 mHz AUTO
P2: 20 mV AC
P1: 0 mHz AUTO
P2: 20 mV AC
P1: 0 mHz AUTO
P2: 20 mV AC
P1: 0 mHz AUTO
P2: 20 mV AC
P1: 0 mHz AUTO
P2: 20 mV AC
```

The meter now returns some readings. To capture readings from a specific channel we can specify as follows:

[Example 2]

```
$ sudo sigrok-cli -d brymen-bm86x:conn=0820.0001 --samples 5 --channels P2
P2: 20 mV AC
P2: 20 mV AC
P2: 20 mV AC
P2: 20 mV AC
P2: 20 mV AC
```

Compiling libsigrok from git

When trying to communicate with the meter using a specific channel, it was discovered that data for both channels was returned regardless of the channel specified. The output shown in the first example above was obtained even when the ‘--channels P2’ option was specified. This problem was reported to the developpers via the sigrok-devel mailing list. They established that this was an omission and a fix was provided.

Eventually the fix will be made part of the official distribution, but in the meantime it is now available in the git repository. To test it, it was neccessary to clone (download) and compile libsigrok from the git repository. This also required installing additional dependencies and compiling the sources.

The repository is found here:

https://sigrok.org/gitweb/

The first step is to make sure all the required tools are installed:

```
$ sudo apt-get install git
$ sudo apt-get install automake
$ sudo apt-get install python-numpy python-setuptools
$ sudo apt-get install libtool qt5-default
```

The above is all that was required at this point. The details of all dependencies and the build process is provided in the libsigrok readme file. Additional dependencies may be required on your system.

The buid process is stated as follows:

```
Building and installing
  -----------------------

  In order to get the libsigrok source code and build it, run:

  $ git clone git://sigrok.org/libsigrok
  $ cd libsigrok
  $ ./autogen.sh
  $ ./configure
  $ make
```

First create a suitable directory to hold the sources (use any name you like):

```
$ mkdir libsigrok-git
```

Then go ahead and clone the repository:

```
$ git clone git://sigrok.org/libsigrok libsigrok-git
```

Naviagate to this directory:

```
$ cd libsigrok-git
```

Now follow the rest of the process:

```
$ ./autogen.sh
```

This should create the configure script, a Makefle.in, and m4 directory and some other bits. After this, run the configure script:

```
$ ./configure
```

If all goes well, this should complete successfully. If not, then examine the output carefully to determine whether any dependencies have been missed and install these as required, then re-run the scripts.

When running autogen.sh, I received some warnings:

```
configure.ac:388: warning: Missing macro AX_PYTHON_MODULE: no check for
setuptools and numpy
configure.ac:388: warning: Missing macro AX_PYTHON_MODULE: no check for
setuptools and numpy
autoreconf: configure.ac: tracing
configure.ac:388: warning: Missing macro AX_PYTHON_MODULE: no check for
setuptools and numpy
```

It seems that these can be safely ignored.

In the output of the configure script, under the ‘Enabled hardware drivers’ section, ensure that the driver for your equipment has a ‘yes’ next to it. If not, then install the indicated missing dependency and run both the autogen.sh and configure scripts again.

Once configure has been run successfully, we can then proceed to build the library:

```
$ make
```

I got the following towards the end:

```
copying selected object files to avoid basename conflicts...
ar: `u' modifier ignored since `D' is the default (see `U')
```

However this can be safely ignored.
After this, run the install script as root:

```
$ su
$ make install
$ exit
```

The last command returns back to user mode.

You can now connect the meter again and test. At this point the ‘--channels P2’ parameter now worked correctly and produced the output in the second example.

Compiling PulseView

Since I have a logic analyzer I waned to get the latest pulseview running. The procedure from the INSTALL file is:

```
Building and installing
-----------------------

In order to get the PulseView source code and build it, run:

 $ git clone git://sigrok.org/pulseview
 $ cd pulseview
 $ cmake .
 $ make

For installing PulseView:

 $ sudo make install
```

But first, we need to get some dependencies installed:

```
$ sudo apt-get install libtool libboost-dev
$ sudo apt-get install libboost-filesystem-dev
$ sudo apt-get install libboost-serialization-dev
$ sudo apt-get install libboost-test-dev
$ sudo apt-get install qt5-default libqt5svg5 libqt5svg5-dev
$ sudo apt-get install libglibmm-2.4-dev
$ sudo apt-get install python-gobject python-gobject-dev
$ sudo apt-get install swig doxygen
```

Once these are installed, the libsigrok library needs to be re-compiled in order to generate the neccessary supporting resources, i.e. the needed language bindings dependencies. After running ./configure, in the ‘Detected Libraries’ section, the following libraries should be flagged ‘yes’: glibmm, python 2.7, pygobject. In the language bindings dependenciies section, the C++ and Python entries must be flagged ‘yes’. If any are flagged ‘no’, the following steps to compile PulseView will not work. Once everything needed is flagged as ‘Yes’, do a *make* and *make install*.

Now go back to the pulseview directory and run:

```
$ cmake .
```

Once that is done run:

```
$ make
```

And the finally:

```
$ sudo make install
```

Pulseview should now run from the command line:

```
$ pulseview &
```

Note: if you get an error as follows the check that C++ is flagged ‘Yes’ in the language bindings dependenciies section.

```
$ pulseview: error while loading shared libraries: libsigrokcxx.so.4: cannot
open shared object file: No such file or directory
```

If it is, then try running:

```
$ sudo ldconfig
```

This resolved the issue on my system.

Compiling sigrok-meter from git

The sigrok-meter tool is a work in progress and currently needs to be cloned from the git to a local directory:

```
$ mkdir sigrok-meter-git
$ git clone git://sigrok.org/sigrok-meter sigrok-meter-git
```

Install the required dependencies:

```
$ sudo apt-get install doxygen
$ sudo apt-get install python-pyside python-pyside.qtcore python-pyside.qtgui
$ sudo apt-get install python-pyqtgraph
$ sudo apt-get install pyqt4-dev-tools
$ sudo apt-get install python-gobject python-gobject-dev
```

Now Recompile and re-install the libsigrok library. This will install the needed language bindings dependencies.

Navigate to the sigrok-meter local directory and run:

```
$ make
```

This should return one line indicating that the necessary resources have been compiled:

```
pyrcc4 -py3 -o resources.py resources.qrc
```

To run the meter program in demo mode type:

```
$ ./sigrok-meter &
```

To run it with your meter, specify the driver:

```
$ ./sigrok-meter -d brymen-bm86x &
```

Note: substitude ‘brymen-bm86x’ for whatever driver your instrument requires.

Note: if the pyqt4-dev-tools package is not installed then running sigrok-meter gives a dialogue with an error:

> Unable to use the sigrok Python bindings:
> No module named sigrok-core

Note: if you get an error that relates to the program being unable to use the libsigrokcxx.so.4 library, check that C++ is flagged ‘Yes’ in the language bindings dependenciies section. If it is, then try running:

```
$ sudo ldconfig
```

This resolved the issue on my system.

Running in user mode

When running sigrok-cli in user mode, access to the USB device is restricted and it is necessary to precede each *sigrok-cli* command with *sudo* in order to provide the library with approporiate access to the device hardware.

In order to be able to run *sigrok-cli* commands as a user, a UDEV rule needs to be specified for the device being used. The sigrok project provides a file that contains UDEV rules for devices supported by sigrok. This file can be found in the libsigrok source directory at libsigrok/contrib/z60_libsigrok.rules. UDEV is a device manager for the Linux kernel. It handles user space events raised and can provide access to USB devices using a set of user defined rules.

Udev should already be installed, but if not, then it can be installed with:

```
$ sudo apt-get install udev
```

Copy the z60_libsigrok.rules file to the appropriate place for your distribution. Linux Mint contains a number of pre-defined rules in /lib/udev/rules.d, but the proper place for user provided rules is /etc/udev/rules.d

So in my case it was appropriate to:

```
$ cp z60_libsigrok.rules /etc/udev/rules.d/60-libsigrok.rules
```

Note: remove the preceding ‘z’. I have also changed the underscore to a hyphen for consistency with the other rules on my system.

In the case of my Brymen meter, the necessary rule had not been included, so I added a rule based on one of the existing ones, keeping things in alphabetical order:

```
# Brymen BU-86X
ATTRS{idVendor}=="0820", ATTRS{idProduct}=="0001", MODE="660", GROUP="plugdev",
TAG+="uaccess"
```

Note: The sigrok project developpers have now added this rule to the file in the official distribution so it should now be available in the file you download from the git.

Restart udev:

```
$ sudo /etc/init.d/udev restart
```

Unplug the USB cable if you already had it plugged in. Now plug in the USB cable again. UDEV should now apply the rule upon detecting the device.

When sigrok-cli is now run as user, the ‘sr: usb: Failed to open device: LIBUSB_ERROR_ACCESS’ should no longer appear and the device should work as normal.

Conclusion

I hope my notes may be of some help so someone attempting to compile the sigrok modules. Of course, you may encounter errors other than the ones I encountered, and most often, if ./configure or make does not succeed it is usually because of one or more missing dependencies. Carefully examine the output from the ./configure and make commands and look for clues to the missing libraries or components that may be required.

Since not eveyone will need to compile all of the sigrok modules, I have inclided additional dependencies as required in each section. This does require going back to and re-compiling *libsigrok*, however if you intend to compile both *PulseView* and s*igrok-meter* as well, then all of the dependencies mentioned in this article could be installed at the very beginning which would probably save a bit of time.

With grateful thanks to the sigrok developpers for their assistance and acknowledging and fixing the omissions I identified.