RTL-SDR on Ubuntu

Introduction

This note is published in the hope that it will be useful. However, the reader, particularly if living in California, must accept full responsibility for any use which they may make of it.

Robin, G8DQX 2017-10-21

There are various cheap & cheerful SDR dongles (RTL dongles) available, designed for DVB-T/DAB/FM service, but which may be pressed in to service as 8-bit[1] SDR radios. These are based around the Realtek RTL2832U demodulator chip and R820T or R820T2 tuner chip. The demodulator chip has a direct sample capability[2] which permits its use as part of a general SDR chain.

There is a compatibility list[3], which recommends these sources:

- Newsky TV28T dongle[4]
- Cosycave (https://www.cosycave.co.uk/)
- Nooelec (http://www.nooelec.com/)

DVB-T dongles which use **other** chipsets are available, but such are no use for this, general SDR, application.

Further sources of information are:

- the Osmocom rtl-sdr wiki page[5]
- RTLSDR.com[6]
- LinuxTV.org[7] for dongle info

This note assumes that the reader is familiar with basic Ubuntu administration, including at least the use of sudo,
the use of the Users and Groups GUI,
and installing packages using Synaptic (or the command line)

[1] 8 bit operation implies **≤48dB** of dynamic range of the A/D converter.

[2] The capability is noted at http://rtlsdr.org/#history_and_discovery_of_rtlsdr.

[3] At https://www.reddit.com/r/RTLSDR/wiki/compatibility.

[4] More info at http://www.newskysz.com/228/101.html.

[5] At http://osmocom.org/projects/sdr/wiki/rtl-sdr.

[6] At http://www.rtlsdr.com/.

[7] At https://www.linuxtv.org/wiki/index.php/DVB-T_USB_Devices.

Installing software & configuring device recognition

Adapted from an instructables page[8]*, a short note*[9]*, and other sources. Compilation from source is generally no longer necessary*.

Set up sdr group

Access to the RTL dongle is granted to members of the **sdr** group[10].

- Establish the **sdr** group, if not already present
- Add a user to the **sdr** group if they are to be permitted to use the RTL dongle

Install GNU Radio & RTL-SDR

These may be installed from the same repositories as used for GQRX installation.

Package **gnuradio** is installed as a dependency of package **gqrx-sdr**.

Package **rtl-sdr** must be installed directly.

Blacklist kernel module

Generally, Linux will automatically load a kernel module to support the normal operation of an RTL dongle. However, for knockabout SDR use, which repurposes a mode otherwise intended only for VHF and DAB/DAB+, this kernel module must be blocked from loading.

As root, generate a new file: **`/etc/modprobe.d/no-rtl.conf`**

Make the contents[11] of the file:

```
# Blacklist RTL driver modules to allow full-range SDR application
# Remove blacklist to restore normal operation, if required
blacklist dvb_usb_rtl28xxu
blacklist dvb_usb_rtl2832u
blacklist dvb_usb_v2
blacklist e4000
blacklist fc0013
blacklist r820t
blacklist rtl2830
```

[8] Found at http://www.instructables.com/id/rtl-sdr-on-Ubuntu/.

[9] See https://ranous.files.wordpress.com/2016/03/rtl-sdr4linux_quickstartv10-16.pdf, at https://ranous.wordpress.com/.

[10] Other guides suggest using the **adm** group for this purpose, but this is a **nasty hack** which adds stray functionality to a group which has a different declared purpose.

[11] Contents from various web pages, including https://blog.jokielowie.com/en/2017/04/sledzimy-szybowce-samoloty-helikoptery-i-balony-cz-3-flarm/.

```
blacklist rtl2832
```

Set up UDEV rules for the dongle in hand

Not all suitable dongles will have the same USB identifiers. Thus the appropriate **UDEV rule** must be crafted for the specific dongle to be used.

Identify the device in use

Using **lsusb**, identify the vendor ID and product ID of the device in use:

```
lsusb
```

Which will return a line similar to:

```
Bus 003 Device 003: ID abcd:0123 ACME Dxer
```

The important details are the <vendor>:<product> identification, which in this example are `abcd:0123`.

These details are entered in to a new UDEV rule.

Create a new UDEV rule

As root, create a new file `/etc/udev/rules.d/20-rtlsdr.rules` with the following line as content, but substituting the newly discovered vendor and product IDs:

```
SUBSYSTEM=="usb", ATTRS{idVendor}=="abcd", ATTRS{idProduct}=="0123",
GROUP="sdr", MODE="0666", SYMLINK+="rtl_sdr"
```

This makes the dongle accessible to any user in the **sdr** group, and adds a symlink `/dev/rtl_sdr` when the dongle is attached.

Activate the UDEV rule

Now restart the USB device enumerator:

```
sudo service udev restart
```

UDEV rule for NooElec.com NESDR SMArt

If using a different device, this is no more than a worked example.

Taking the procedure as above, the ID is:

```
Bus 003 Device 014: ID 0bda:2838 Realtek Semiconductor Corp. RTL2838
DVB-T
```

The rule file `/etc/udev/rules.d/20-nooelec.comNESDRSMArtsdr.rules` is then:

```
SUBSYSTEM=="usb", ATTRS{idVendor}=="0bda", ATTRS{idProduct}=="2838",
GROUP="sdr", MODE="0666", SYMLINK+="rtl_sdr"
```

Test it out

Basic testing

The first thing to do would seem to be to check out the dongle plus configuration using **rtl_test** (from the **rtl-sdr** package). According to the **rtl_fm** documentation:

> **rtl_test** should return a list of supported gain values and not produce error messages. Make sure you are using a USB 2.0 port[12] as well, otherwise you get really weird errors.

A good result should be similar to[13]:

```
:~$ rtl_test
Found 1 device(s):
  0:  Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Supported gain values (29): 0.0 0.9 1.4 2.7 3.7 7.7 8.7 12.5 14.4 15.7
16.6 19.7 20.7 22.9 25.4 28.0 29.7 32.8 33.8 36.4 37.2 38.6 40.2 42.1
43.4 43.9 44.5 48.0 49.6
[R82XX] PLL not locked!
Sampling at 2048000 S/s.

Info: This tool will continuously read from the device, and report if
samples get lost. If you observe no further output, everything is fine.

Reading samples in async mode...
^CSignal caught, exiting!

User cancel, exiting...
Samples per million lost (minimum): 0
```

Now is a good time to put some sort of **effective** aerial on to the dongle, and see how it performs. **rtl_fm** is a simple SDR receiver that demodulates:

- AM
- FM (narrowband)
- LSB
- USB
- WBFM
- raw output for test or specialist purposes

[12]USB port issues may occur with any version of port. USB 2.0 and USB 3.0 may all, in general, perform satisfactorily. There may be issues arising from poorly designed USB hardware, but that is outside the scope of this note!

[13]Possibly with a small number of lost samples, such as 108 bytes, announced at the start of the run.

There is more information in a guide[14]. A good test of FM reception might be, substituting a local strong FM station for 92.5MHz:

```
rtl_fm -M wbfm -f 92.5M | play -r 32k -t raw -e s -b 16 -c 1 -V1 -
```

The guide also suggests scanning **airband** and simple **ADS-B** reception.

Using GRC

Beware. *The GRC script[15] mentioned in the instructables website no longer works, and is best ignored as a test mechanism.*

Applications

Using GQRX

GQRX has a direct configuration of RTL dongles. The **device string**[16] should read:

`rtl=0` for VHF/UHF operation

`rtl=0,direct_samp=2` for direct sampling mode, or possibly

`rtl=0,direct_samp=1`

Spectrum analyser

There is a python-based spectrum analyser[17] available.

ADS-B

There are various ADS-B applications available[18]. The most reliable so far seems to be **dump 1090**[19]. This application must be compiled, and a library development package installed. This requires the following packages (at least), and some familiarity with the compilation process:

```
build-essential
librtlsdr-dev
```

Other applications

There is a great deal of Linux software that supports the RTL hardware[20].

[14]At http://kmkeen.com/rtl-demod-guide/.

[15]The script is to be found at https://cdn.instructables.com/ORIG/FE4/ZI8J/H1LWQJUY/FE4ZI8JH1LWQJUY.grc.

[16]The device string is part of GNU Radio.

[17]Details at https://github.com/EarToEarOak/RTLSDR-Scanner.

[18]Noted at https://www.rtl-sdr.com/adsb-aircraft-radar-with-rtl-sdr/.

[19]Available at https://github.com/MalcolmRobb/dump1090.

[20]Including a list at https://www.rtl-sdr.com/big-list-rtl-sdr-supported-software/.

Multiple Dongles with GQRX

This procedure is noted, but is not tested by the author.

Using multiple dongles with GQRX on a single host is possible if they are differentiated by **serial number**. This may be set using **rtl_eeprom**. The serial number is then quoted in a device string, set in the GQRX configuration dialogue. This is set as follows[21,22]:

RTL-SDR

Argument	Notes
rtl=<device-index>	0-based device identifier OR **serial number**
rtl_xtal=<frequency>	Frequency (Hz) used for the RTL chip, accepts scientific notation
tuner_xtal=<frequency>	Frequency (Hz) used for the tuner chip, accepts scientific notation
buffers=<number-of-buffers>	Default is 32
buflen=<length-of-buffer>	Default is 256kB, must be multiple of 512
direct_samp=0|1|2	Enable direct sampling mode on the RTL chip. 0: Disable, 1: use I channel, 2: use Q channel
offset_tune=0|1	Enable offset tune mode for E4000 tuners

If multiple dongles are simultaneously to be active, then multiple instance of GQRX must be launched, configured for the individual serial numbers.

[21]As described at http://osmocom.org/projects/sdr/wiki/GrOsmoSDR#RTL-SDRSource.

[22]A similar procedure would apply to GNU Radio.