Welcome to OsmoSDR

OsmoSDR is a 100% Free Software based small form-factor inexpensive SDR (Software Defined Radio) project.

If you are familiar with existing SDR receivers, then OsmoSDR can be thought of something in between a FunCube Dongle (only 96kHz bandwidth) and a USRP (much more expensive). For a very cheap (but inaccurate) SDR, you can use the DVB-T USB stick using the RTL2832U chip, as documented in rtl-sdr.

It consists of a USB-attached Hardware, associated Firmware as well as GrOsmoSDR gnuradio integration on the PC.

The first generation hardware (1.2 MS/s) is available to interested developers from http://shop.sysmocom.de/

We are currently doing a re-design for the second-generation hardware, featuring higher sample rate and other goodies.

Hardware schematics and firmware source code are kept in a git repository on git.osmocom.org.

- clone the repository like this: `git clone git://git.osmocom.org/osmo-sdr.git`
- web-browse the repository: http://cgit.osmocom.org/cgit/osmo-sdr/

For a complete list of local wiki pages, see TitleIndex.

Picture

This is a (slightly better) picture of the current developer preview:

GNSS-SDR officially supports rtl-sdr

The team of developers of the open source project GNSS-SDR proudly announces that the latest version of our GNSS receiver supports the realtime operation using RTL-SDR compatible dongles.

They achieved GPS position fix with what is probability the cheapest GPS receiver ever made. If you have a RTL2832U based compatible DVB-T receiver you can ...

(Read more)

Posted: 2012-08-17 21:40
Author: laforge
Categories: gnss rtlsdr gps
Comments (0)

First 16 OsmoSDR boards available for developers

There are something like 16 units of OsmoSDR that we have produce and which are able to sell to interested developers.

However, as there are only 16 units right now, and as the firmware and host software is in a barely usable but incomplete state, we would like to make sure that those 16 units get sold to people who actually have an interest (a ...

(Read more)

Posted: 2012-05-16 22:39
Author: laforge
Categories: sdr
Comments (0)

Introducing RTL-SDR

While the OsmoSDR is still not available, some Osmocom team members (notably Steve Markgraf) have been hacking away on an alternative least-cost solution: rtl-sdr.

So what is rtl-sdr? It is a creative form of using consumer-grade DVB-T USB receivers, turning them into fully-fledged software defined radios.

Those ...

(Read more)

Posted: 2012-03-18 19:00
Author: laforge
Categories: rtl sdr
Comments (0)

OsmoSDR hardware verification at 28C3

At 28c3, the OsmoSDR team was busy verifying the hardware design on the first prototypes.

The result can be summarized as:

- SAM3U is working, enumerates on USB and can be programmed via SAM-BA
- E4K tuner driver is working
- Si570 driver is working

- FPGA can be flashed via JTAG bit-banging from SAM3 ...

(Read more)

Posted: 2011-12-31 15:51

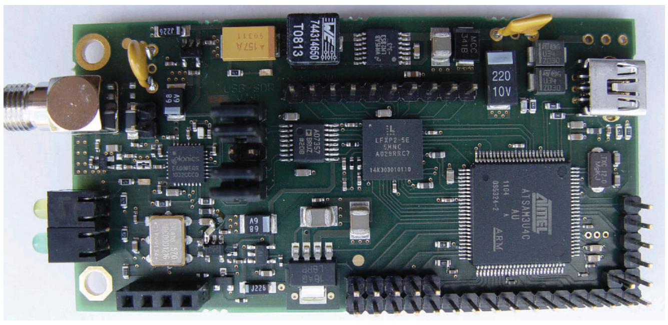
Author: laforge
Categories: hardware
Comments (0)

About OsmocomSDR

This is the blog of the OsmoSDR project, a small-size, low-cost Software Defined Radio hardware/firmware project.

Posted: 2011-12-31 15:44

Author: laforge
Categories: (none)
Comments (0)



Mailing List

We discuss both OsmoSDR as well as rtl-sdr on the following mailing list: osmocom-sdr@…. You can subscribe and/or unsubscribe via the following link: http://lists.osmocom.org/mailman/listinfo/osmocom-sdr

Credits

- Stefan Reimann of SR-Systems (electrical engineering, manufacturing)
- Christian Daniel, Matthias Kleffel and Thomas Kleffel of maintech (overall design, FPGA, rum-ba)
- Harald Welte of sysmocom (sam3u firmware development)

Attachments

- osmosdr.jpg (173.2 KB) - added by *laforge* 9 months ago. “PCB photograph of OsmoSDR”
- osmosdr-beta.jpg (145.3 KB) - added by *Hoernchen* 4 months ago. “OsmoSDR developer preview”

OsmoSDR Gnuradio Source

Primarily gr-osmosdr supports the OsmoSDR hardware, but it also offers a wrapper functionality for FunCube Dongle, Ettus UHD and rtl-sdr radios. By using gr-osmosdr source you can take advantage of a common software api in your application(s) independent of the underlying radio hardware.

Build process

The gnuradio source requires a recent gnuradio (>= v3.5.3) to be installed.

Please note: prior pulling a new version from git and compiling it, please do a "make uninstall" first to properly remove the previous version.

Building with cmake:

```
git clone git://git.osmocom.org/gr-osmosdr
cd gr-osmosdr/
mkdir build
cd build/
cmake ../
make
sudo make install
sudo ldconfig
```

NOTE: The source block will appear under 'OsmoSDR' category in GRC menu.

To build the API documentation:

```
cd build/
cmake ../ -DENABLE_DOXYGEN=1
make -C docs
```

Automated installation

Marcus D. Leech has kindly integrated the forementioned build steps into his gnuradio installation script at http://www.sbrac.org/files/build-gnuradio. This is the most user-friendly option so far.

Device specification

You can specify the source or sink device using a comma separated string of argument=value pairs.

Local file

file=<path-to-file-name>	
freq=<frequency>	in Hz
rate=<sampling-rate>	in samples/s
repeat=true|false	Default is false
throttle=true|false	Throttle flow of samples, default is false

OsmoSDR (source)

osmosdr=<device-index>	
mcr=<rate>	Master clock rate. FIXME: Setting the MCR is not supported
nchan=<channel-number>	FIXME: Values of nchan != 1 are not supported
buffers=<number-of-buffers>	Default is 32

OsmoSDR (sink)

FIXME: OsmoSDR sink is not yet implemented.

rtl-sdr

rtl=<device-index>	
rtl_xtal=<frequency>	Frequency in Hz of the crystal oscillator used for the RTL chip
tuner_xtal=<frequency>	Frequency in Hz of the crystal oscillator used for the tuner chip
buffers=<number-of-buffers>	Default is 32

NOTE: if you don't specify rtl_xtal/tuner_xtal, the underlying driver will use 28.0MHz

rtl-sdr TCP server

rtl_tcp=<hostname>:<port>	hostname defaults to "localhost", port to "1234"
psize=<payload-size>	Default is 16384 bytes

UHD

uhd	Use this argument without value
nchan=<channel-index>	
subdev=<subdev-spec>	See below

Additional argument/value pairs will be passed to the underlying driver, for more information see http://files.ettus.com/uhd_docs/manual/html/general.html#specifying-the-subdevice-to-use and http://files.ettus.com/uhd_docs/manual/html/identification.html#common-device-identifiers

FCD

Argument	Notes
fcd=<device-index>	

Known Apps

The following 3rd party applications are successfully using gr-osmosdr:

Name	**Type**	**Author**	**URL**
gr-pocsag	GRC Flowgraph	Marcus Leech	https://www.cgran.org/browser/projects/gr-pocsag/trunk
multimode RX	GRC Flowgraph	Marcus Leech	https://www.cgran.org/browser/projects/multimode/trunk
simple_fm_rvc	GRC Flowgraph	Marcus Leech	https://www.cgran.org/browser/projects/simple_fm_rcv/trunk
Wireless Temp. Sensor RX	Gnuradio App	Kevin Mehall	https://github.com/kevinmehall/rtlsdr-433m-sensor
gqrx (fork)	SDR GUI	Alexandru Csete	https://github.com/mathisschmieder/gqrx
tetra_demod_fft	SDR GUI	osmocom team	osmosdr-tetra_demod_fft.py and the HOWTO
gqrx (original)	SDR GUI	Alexandru Csete	https://github.com/csete/gqrx/tree/osmosdr
tetra_demod_fft	Trunking RX	osmocom team	osmosdr-tetra_demod_fft.py and the HOWTO
airprobe **(NEW)**	GSM sniffer	osmocom team et al	http://git.gnumonks.org/cgi-bin/gitweb.cgi?p=airprobe.git
gr-smartnet (WIP) **(NEW)**	Trunking RX	Nick Foster	http://www.reddit.com/r/RTLSDR/comments/us3yo/rtlsdr_smartnet/ Notes from the author
gr-air-modes **(NEW)**	ADS-B RX	Nick Foster	https://www.cgran.org/wiki/gr-air-modes call with --rtlsdr option

UHD - USRP Hardware Driver

UHD is the "Universal Software Radio Peripheral" hardware driver. The goal of UHD is to provide a host driver and API for current and future Ettus Research products. Users will be able to use the UHD driver standalone or with third-party applications.

Contents

Generated on: 2012-08-31 18:21 UTC.

UHD Installation (Linux)

We provide UHD binary installers for Ubuntu and Fedora users. There are two choices:

- **Unstable**: based off of UHD's **master** branch. Includes new features and bugfixes.
- **Stable**: based off of UHD's **maint** branch. Only includes releases and bugfixes.

We only support versions of Ubuntu and Fedora that are officially maintained and supported. Currently supported platforms are:

- **Ubuntu 10.04, 11.04-12.04**
- **Fedora 16-17**

The three methods of installing UHD are described below.

Install with Apt/Yum

Binaries generated from UHD's unstable and bugfix branches can be installed using our Ubuntu and Fedora repositories. Follow these instructions to install UHD and receive package updates.

Ubuntu

NOTE: Only follow one set of these instructions at a time. Our unstable and stable repositories will conflict.

- **Master/Unstable**
  - Copy and paste these commands into your terminal. This will install UHD as well as allow you to receive package updates.

```
sudo bash -c 'echo "deb http://files.ettus.com/binaries/uhd_unstable/repo/uhd/ubuntu/`lsb_release -cs` `lsb_release -cs` main" > /etc/apt/sources.list.d/ettus.list'
sudo apt-get update
sudo apt-get install -t `lsb_release -cs` uhd
```

- **Releases/Bugfixes**
  - Copy and paste these commands into your terminal. This will install UHD as well as allow you to receive package updates.

```
sudo bash -c 'echo "deb http://files.ettus.com/binaries/uhd_stable/repo/uhd/ubuntu/`lsb_release -cs` `lsb_release -cs` main" > /etc/apt/sources.list.d/ettus.list'
sudo apt-get update
sudo apt-get install -t `lsb_release -cs` uhd
```

Fedora

NOTE: Only follow one set of these instructions at a time. Our unstable and stable repositories will conflict. If you are switching between our unstable and stable repositories, run the following command before doing so:

```
yum clean metadata all
```

- **Master/Unstable**
  - Follow these instructions to install UHD and receive package updates.
  - Create the file **/etc/yum.repos.d/ettus.repo**. Copy this into the file:

```
[ettus-uhd-master-repo]
name=Ettus Research - UHD Master $releasever-$basearch
baseurl=http://files.ettus.com/binaries/uhd_unstable/repo/uhd/fedora/$releasever/$basearch
gpgcheck=0
```

  Run the following commands:

```
sudo yum --enablerepo='ettus-uhd-master-repo' install uhd
```

- **Releases/Bugfixes**
  - Follow these instructions to install UHD and receive package updates.

  - Create the file **/etc/yum.repos.d/ettus.repo**. Copy this into the file:

```
[ettus-uhd-stable-repo]
name=Ettus Research - UHD Stable $releasever-$basearch
baseurl=http://files.ettus.com/binaries/uhd_stable/repo/uhd/fedora/$releasever/$basearch
gpgcheck=0
```

    Run the following commands:

```
sudo yum --enablerepo='ettus-uhd-stable-repo' install uhd
```

Installer Packages

We provide standalone binaries for all supported platforms.

With the release of a new binary, we provide the following:

- UHD binary
- Tarball of the UHD images used when forming the binary

NOTE: Installing packages with this method will not install all necessary runtime dependencies. Run the following command to install runtime dependencies:

Ubuntu

```
sudo apt-get install python libboost-all-dev libusb-1.0-0-dev
```

Fedora

```
sudo yum install python boost-devel libusb1-devel
```

Follow these links for easy access to the latest binaries:

- Latest Unstable
- Latest Stable
- Latest Release **(003.004.003)**

Binaries of all previous releases can be downloaded below:

- <Newest>
- 003.004.003 - 2012/07/30
- 003.004.002 - 2012/05/24
- 003.004.001 - 2012/04/12
- 003.004.000 - 2012/03/21
- 003.003.002 - 2012/01/09
- 003.003.001 - 2011/11/01
- 003.003.000 - 2011/10/14
- 003.002.004 - 2011/09/23
- 003.002.003 - 2011/08/31
- 003.002.002 - 2011/08/16
- 003.002.001 - 2011/07/29
- 003.002.000 - 2011/07/28
- 003.001.002 - 2011/06/15
- 003.001.001 - 2011/06/06
- 003.001.000 - 2011/05/18
- 003.000.001 - 2011/04/01
- 003.000.000 - 2011/03/28
- <Oldest>

Source Code

Or you can build UHD from source. Follow the Build Instructions.

files.ettus.com/binaries/uhd_stable/latest_stable/

Ettus Research

Name	Last modified	Size	Description
Parent Directory		-	
uhd-images_003.004.003-release.tar.gz	06-Sep-2012 15:33	6.6M	
uhd-images_003.004.003-release.zip	06-Sep-2012 15:33	6.6M	
uhd_003.004.003-26-stable-debug_Win32.exe	06-Sep-2012 15:33	22M	
uhd_003.004.003-26-stable-debug_Win64.exe	06-Sep-2012 15:33	11M	
uhd_003.004.003-26-stable_Fedora-16-i686.rpm	06-Sep-2012 15:33	6.0M	
uhd_003.004.003-26-stable_Fedora-16-x86_64.rpm	06-Sep-2012 15:33	6.0M	
uhd_003.004.003-26-stable_Fedora-17-i686.rpm	06-Sep-2012 15:33	6.0M	
uhd_003.004.003-26-stable_Fedora-17-x86_64.rpm	06-Sep-2012 15:33	6.0M	
uhd_003.004.003-26-stable_Ubuntu-10.04-i686.deb	06-Sep-2012 15:33	8.5M	
uhd_003.004.003-26-stable_Ubuntu-10.04-x86_64.deb	06-Sep-2012 15:33	8.5M	
uhd_003.004.003-26-stable_Ubuntu-11.04-i686.deb	06-Sep-2012 15:33	8.5M	
uhd_003.004.003-26-stable_Ubuntu-11.04-x86_64.deb	06-Sep-2012 15:33	8.6M	
uhd_003.004.003-26-stable_Ubuntu-11.10-i686.deb	06-Sep-2012 15:33	8.5M	
uhd_003.004.003-26-stable_Ubuntu-11.10-x86_64.deb	06-Sep-2012 15:33	8.5M	
uhd_003.004.003-26-stable_Ubuntu-12.04-i686.deb	06-Sep-2012 15:33	8.5M	
uhd_003.004.003-26-stable_Ubuntu-12.04-x86_64.deb	06-Sep-2012 15:33	8.5M	
uhd_003.004.003-26-stable_Win32.exe	06-Sep-2012 15:33	19M	
uhd_003.004.003-26-stable_Win64.exe	06-Sep-2012 15:33	19M	

files.ettus.com/binaries/gnuradio/latest_stable/

Ettus Research

Name	Last modified	Size	Description
Parent Directory		-	
gnuradio_3.6.2_Fedora-16-i686.rpm	06-Sep-2012 15:33	10M	
gnuradio_3.6.2_Fedora-16-x86_64.rpm	06-Sep-2012 15:33	11M	
gnuradio_3.6.2_Fedora-17-i686.rpm	06-Sep-2012 15:33	11M	
gnuradio_3.6.2_Fedora-17-x86_64.rpm	06-Sep-2012 15:33	11M	
gnuradio_3.6.2_Ubuntu-10.04-i686.deb	06-Sep-2012 15:33	17M	
gnuradio_3.6.2_Ubuntu-10.04-x86_64.deb	06-Sep-2012 15:33	17M	
gnuradio_3.6.2_Ubuntu-11.04-i686.deb	06-Sep-2012 15:33	17M	
gnuradio_3.6.2_Ubuntu-11.04-x86_64.deb	06-Sep-2012 15:33	18M	
gnuradio_3.6.2_Ubuntu-11.10-i686.deb	06-Sep-2012 15:33	18M	
gnuradio_3.6.2_Ubuntu-11.10-x86_64.deb	06-Sep-2012 15:33	18M	
gnuradio_3.6.2_Ubuntu-12.04-i686.deb	06-Sep-2012 15:33	18M	
gnuradio_3.6.2_Ubuntu-12.04-x86_64.deb	06-Sep-2012 15:33	18M	
gnuradio_3.6.2_Win32.exe	12-Sep-2012 12:21	58M	

UHD Start »

UHD Installation (Windows)

We provide UHD installers for Windows users.

UHD can be installed on Windows via pre-built installer packages (compiled with Microsoft Visual Studio 2010) or compiled from source.

- **Currently Supported: XP, Vista, 7**

Installer Packages

Unstable

Unstable installer packages are built from the **master** branch of UHD. These packages include all bugfixes, as well as new features.

- **Latest Unstable Build**
- **Older Unstable builds**

Stable

Stable installer packages are built from the **maint** branch of UHD. These packages only include releases and bugfixes.

- **Latest Stable Build**
- **Older Stable Builds**

Releases

Release installer packages are built from release tags of the **maint** branch.

- **Latest Release (003.004.003)**: (Win32) (Win64)

Older Binaries of all previous releases can be downloaded below:

- <Newest>
- 003.004.003 (Win32) (Win64) - 2012/07/30
- 003.004.002 (Win32) (Win64) - 2012/05/24
- 003.004.001 (Win32) (Win64) - 2012/04/12
- 003.004.000 (Win32) (Win64) - 2012/03/21
- 003.003.002 - 2012/01/09
- 003.003.001 - 2011/11/01
- 003.003.000 - 2011/10/14
- 003.002.004 - 2011/09/23
- 003.002.003 - 2011/08/31
- 003.002.002 - 2011/08/16
- 003.002.001 - 2011/07/29
- 003.002.000 - 2011/07/28
- 003.001.002 - 2011/06/15
- 003.001.001 - 2011/06/06
- 003.001.000 - 2011/05/18
- 003.000.001 - 2011/04/01
- 003.000.000 - 2011/03/28
- <Oldest>

Post-Install Tasks

Using a USB-based device?

- Download the Windows USB driver
- USB post-installation instructions

Install the MSVC Redistributable Package:

- Download MSVC Redistributable Package

Source Code

You can build UHD from source. There are two choices of compilers for Windows users:

- Microsoft Visual Studio Express.
  - Users can develop with the **free** version.
- MinGW
  - An alternative to using a Microsoft compiler.

Follow the Build Instructions.

rtl-sdr

DVB-T dongles based on the Realtek RTL2832U can be used as a cheap SDR, since the chip allows transferring the raw I/Q samples to the host, which is officially used for DAB/DAB+/FM demodulation. The possibility of this has been discovered by the V4L/DVB kernel developer Antti Palosaari.

Specifications

The RTL2832U outputs 8-bit I/Q-samples, and the highest theoretically possible sample-rate is 3.2 MS/s, however, the highest sample-rate without lost samples that has been tested so far is 2.8 MS/s. The frequency range is highly dependent of the used tuner, **dongles that use the Elonics E4000 offer the widest possible range (64 - 1700 MHz with a gap from approx. 1100 - 1250 MHz)**. When used out-of-spec, a tuning range of approx. 50 MHz - 2.2 GHz is possible (with gap).

Supported Hardware

The following devices are known to work fine with RTLSDR software:

VID	PID	tuner	device name
0x0bda	0x2832	all of them	Generic RTL2832U (e.g. hama nano)
0x0bda	0x2838	E4000	ezcap USB 2.0 DVB-T/DAB/FM dongle
0x0ccd	0x00a9	FC0012	Terratec Cinergy T Stick Black (rev 1)
0x0ccd	0x00b3	FC0013	Terratec NOXON DAB/DAB+ USB dongle (rev 1)
0x0ccd	0x00d3	E4000	Terratec Cinergy T Stick RC (Rev.3)
0x0ccd	0x00e0	E4000	Terratec NOXON DAB/DAB+ USB dongle (rev 2)
0x185b	0x0620	E4000	Compro Videomate U620F
0x185b	0x0650	E4000	Compro Videomate U650F
0x1f4d	0xb803	FC0012	GTek T803
0x1f4d	0xc803	FC0012	Lifeview LV5TDeluxe
0x1b80	0xd3a4	FC0013	Twintech UT-40
0x1d19	0x1101	FC2580	Dexatek DK DVB-T Dongle (Logilink VG0002A)
0x1d19	0x1102	?	Dexatek DK DVB-T Dongle (MSI DigiVox? mini II V3.0)
0x1d19	0x1103	FC2580	Dexatek Technology Ltd. DK 5217 DVB-T Dongle
0x0458	0x707f	?	Genius TVGo DVB-T03 USB dongle (Ver. B)
0x1b80	0xd393	FC0012	GIGABYTE GT-U7300
0x1b80	0xd394	?	DIKOM USB-DVBT HD

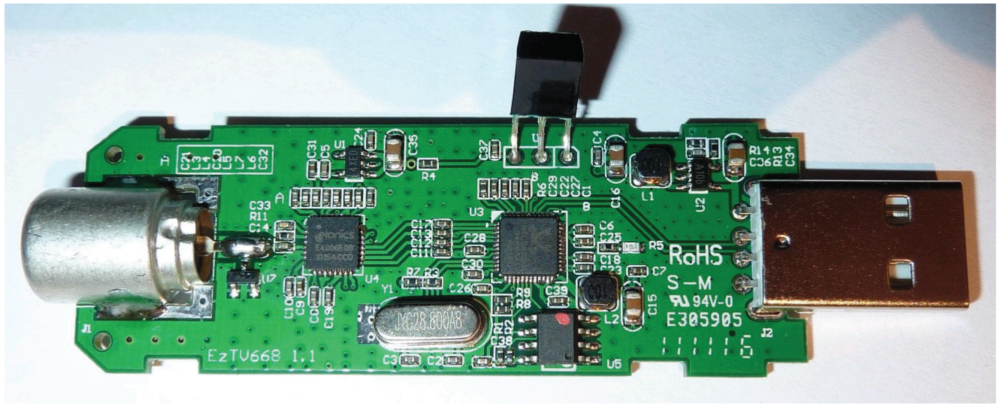
0x1b80	0xd395	FC0012	Peak 102569AGPK
0x1b80	0xd39d	FC0012	SVEON STV20 DVB-T USB & FM

If you don't know where to buy one or if you are just looking for a trustworthy source, try http://shop.sysmocom.de/t/software-defined-radio

People over at reddit are collecting a list (v2) of other devices that are compatible.

Other dongles based on the RTL2832U might be added in the future as well.

This is the PCB of the ezcap-stick:



More pictures can be found here.

Software

rtl-sdr is a commandline tool that can initialize the RTL2832, tune to a given frequency, and record the I/Q-samples to a file.

The code can be checked out with:

```
git clone git://git.osmocom.org/rtl-sdr.git
```

It can also be browsed on http://cgit.osmocom.org/cgit/rtl-sdr/

If you are going to "fork it on github" and enhance it, please contribute back and submit your patches to: osmocom-sdr at lists.osmocom.org

A GNU Radio source block for OsmoSDR **and rtlsdr** is available. **Please install a recent gnuradio (>= v3.5.3) in order to be able to use it.**

Mailing List

We discuss both OsmoSDR as well as rtl-sdr on the following mailing list: osmocom-sdr@....

You can subscribe and/or unsubscribe via the following link: http://lists.osmocom.org/mailman/listinfo/osmocom-sdr

Building the software

rtlsdr library & capture tool

You have to install development packages for libusb1.0 and can either use cmake or autotools to build the software.

Please note: prior pulling a new version from git and compiling it, please do a "make uninstall" first to properly remove the previous version.

Building with cmake:

```
cd rtl-sdr/
mkdir build
cd build
cmake ../
make
sudo make install
sudo ldconfig
```

In order to be able to use the dongle as a non-root user, you may install the appropriate udev rules file by calling cmake with -DINSTALL_UDEV_RULES=ON argument in the above build steps.

```
cmake ../ -DINSTALL_UDEV_RULES=ON
```

Building with autotools:

```
cd rtl-sdr/
autoreconf -i
./configure
make
sudo make install
sudo ldconfig
```

The built executables (rtl_sdr, rtl_tcp and rtl_test) can be found in rtl-sdr/src/.

In order to be able to use the dongle as a non-root user, you may install the appropriate udev rules file by calling

```
sudo make install-udev-rules
```

pre-built Windows version using libusb 1.0.9 and pthreads-win32 cvs

Gnuradio Source

The gnuradio source requires the rtl-sdr package and a recent gnuradio (>= v3.5.3) to be installed.

The source supports direct device operation as well as a tcp client mode when using the rtl_tcp utility as a spectrum server.

Please note: prior pulling a new version from git and compiling it, please do a "make uninstall" first to properly remove the previous version.

Please note: you always should build & **install the latest version of the dependencies (librtlsdr in this case)** before trying to build the gr source.

Building with cmake:

```
git clone git://git.osmocom.org/gr-osmosdr
cd gr-osmosdr/
mkdir build
cd build/
cmake ../
make
sudo make install
sudo ldconfig
```

NOTE: The source block (osmosdr/rtlsdr Source) will appear under 'Sources' category in GRC menu.

For initial tests we recommend the multimode receiver gnuradio companion flowgraph (see "Known Apps" table below).

You may find more detailed installation instructions in this recent tutorial.

Automated installation

Marcus D. Leech has kindly integrated the forementioned build steps into his gnuradio installation script at http://www.sbrac.org/files/build-gnuradio. This is the most user-friendly option so far.

Usage

rtl_sdr

Example: To tune to 392.0 MHz, and set the sample-rate to 1.8 MS/s, use:

```
./rtl_sdr /tmp/capture.bin -s 1.8e6 -f 392e6
```

to record samples to a file or to forward the data to a fifo.

If the device can't be opened, make sure you have the appropriate rights to access the device (install udev-rules from the repository, or run it as root).

rtl_tcp

Example:

```
rtl_tcp -a 10.0.0.2 [-p listen port (default: 1234)]
```

```
Found 1 device(s).
Found Elonics E4000 tuner
Using Generic RTL2832U (e.g. hama nano)
Tuned to 100000000 Hz.
listening...
Use the device argument 'rtl_tcp=10.0.0.2:1234' in OsmoSDR (gr-osmosdr) source
to receive samples in GRC and control rtl_tcp parameters (frequency, gain, ...).
```

use the rtl_tcp=... device argument in gr-osmosdr source to receive the samples in GRC and control the rtl settings remotely.

This application has been successfully crosscompiled for ARM and MIPS devices and is providing IQ data in a networked ADS-B setup at a rate of 2.4MSps. The gr-osmosdr source is being used together with an optimized gr-air-modes version (see Known Apps below). You can find a Makefile for OpenWRT here.

A use case is described here.

rtl_test

To check the possible tuning range (may heavily vary by some MHz depending on device and temperature), call

```
rtl_test -t
```

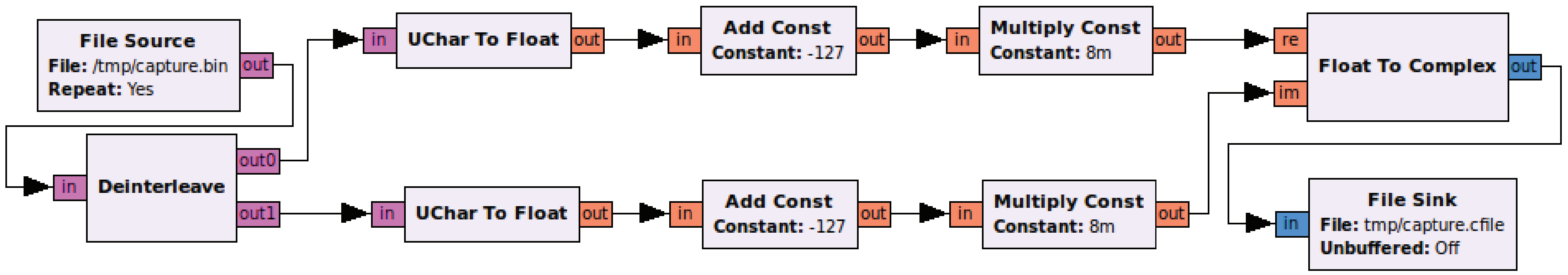
To check the maximum samplerate possible on your machine, type (change the rate down until no sample loss occurs):

```
rtl_test -s 3.2e6
```

A samplerate of 2.4e6 is known to work even over tcp connections (see rtl_tcp above). A sample rate of 2.88e6 may work without lost samples but this may depend on your PC/Laptop's host interface.

Using the data

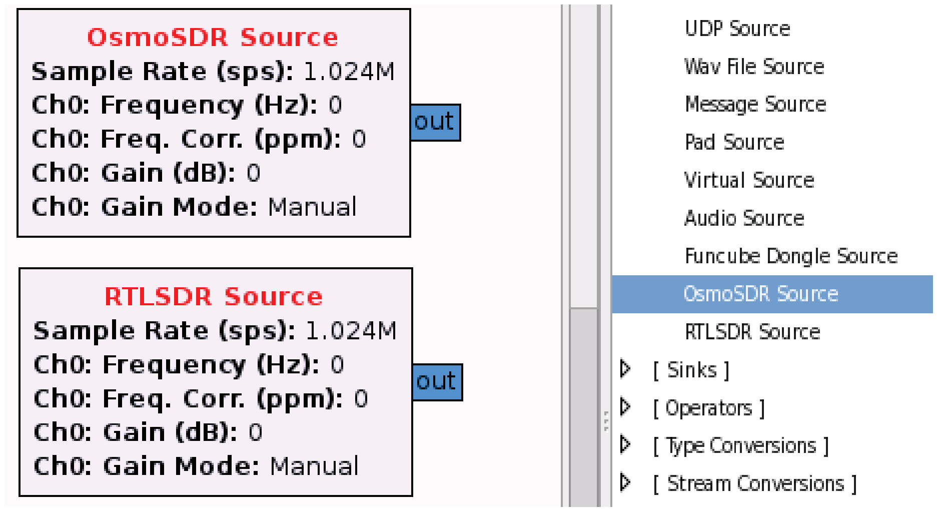
To convert the data to a standard cfile, following GNU Radio Block can be used:



The GNU Radio Companion flowgraph (rtl2832-cfile.grc) is attached to this page. It is based on the FM demodulation flowgraph posted by Alistair Buxton on this thread.

Please note: for realtime operation you may use fifos (mkfifo) to forward the iq data from the capture utility to the GRC flowgraph.

You may use any of the the following gnuradio sources (they are equivalent):


OsmoSDR Source
Sample Rate (sps): 1.024M
Ch0: Frequency (Hz): 0
Ch0: Freq. Corr. (ppm): 0
Ch0: Gain (dB): 0
Ch0: Gain Mode: Manual

RTLSDR Source
Sample Rate (sps): 1.024M
Ch0: Frequency (Hz): 0
Ch0: Freq. Corr. (ppm): 0
Ch0: Gain (dB): 0
Ch0: Gain Mode: Manual

What has been successfully tested so far is the reception of Broadcast FM and air traffic AM radio, TETRA, GMR, GSM, ADS-B and POCSAG.

Tell us your success story with other wireless protocols in ##rtlsdr channel on freenode IRC network.

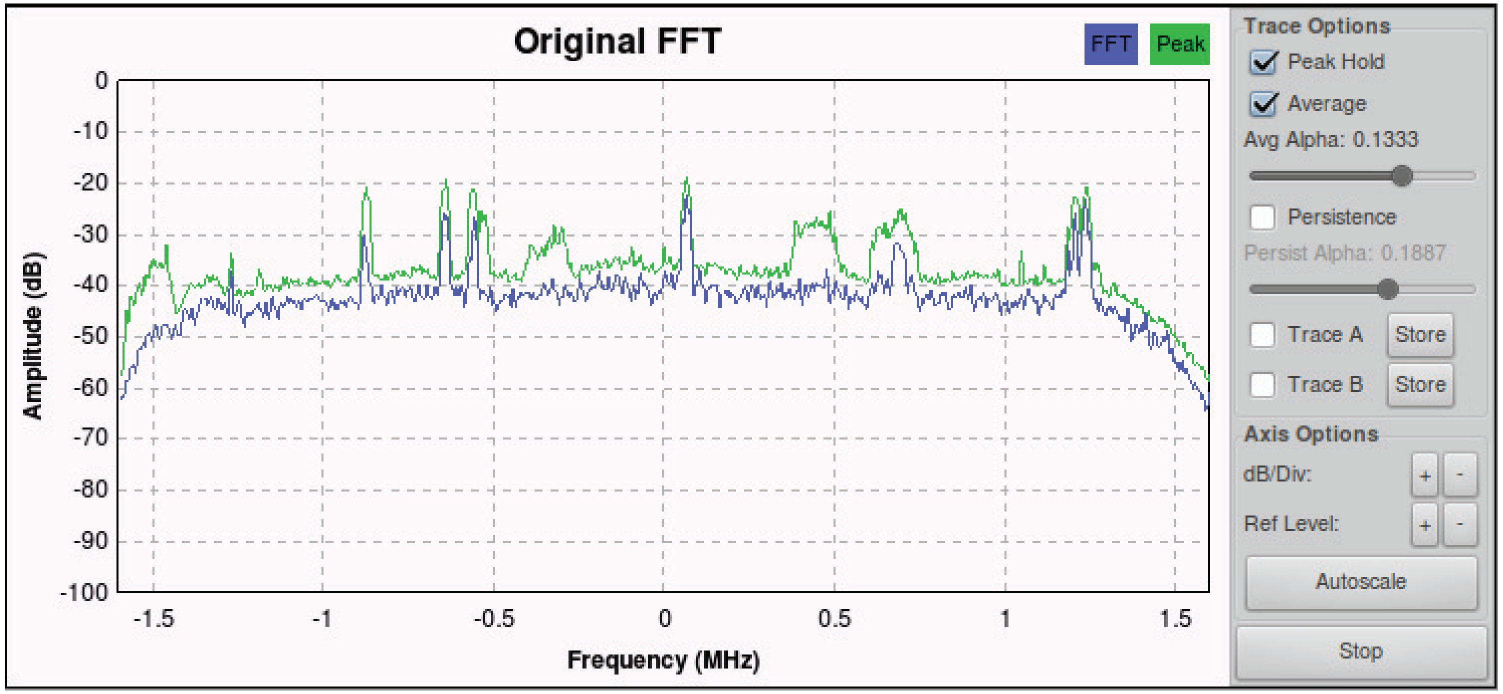
Known Apps

The following 3rd party applications and libraries are successfully using either librtlsdr directly or the corresponding gnuradio source (gr-osmosdr):

Name	Type	Author	URL
gr-pocsag	GRC Flowgraph	Marcus Leech	https://www.cgran.org/browser/projects/gr-pocsag/trunk
multimode RX (try first!)	GRC Flowgraph	Marcus Leech	https://www.cgran.org/browser/projects/multimode/trunk
simple_fm_rvc	GRC Flowgraph	Marcus Leech	https://www.cgran.org/browser/projects/simple_fm_rcv /trunk
python-librtlsdr	Python Wrapper	David Basden	https://github.com/dbasden/python-librtlsdr
pyrtlsdr	Python Wrapper	Roger	https://github.com/roger-/pyrtlsdr
rtlsdr-waterfall	Python FFT GUI	Kyle Keen	https://github.com/keenerd/rtlsdr-waterfall
Wireless Temp. Sensor RX	Gnuradio App	Kevin Mehall	https://github.com/kevinmehall/rtlsdr-433m-sensor

QtRadio	SDR GUI	Andrea Montefusco et al.	http://napan.ca/ghpsdr3/index.php/RTL-SDR
gqrx (fork)	SDR GUI	Alexandru Csete	https://github.com/mathisschmieder/gqrx
rtl_fm	SDR CLI	Kyle Keen	merged in librtlsdr master
SDR#	SDR GUI	Youssef Touil	http://sdrsharp.com/ and Windows Guide or Linux Guide
tetra_demod_fft	Trunking RX	osmocom team	osmosdr-tetra_demod_fft.py and the HOWTO
gqrx (original)	SDR GUI	Alexandru Csete	https://github.com/csete/gqrx
airprobe	GSM sniffer	osmocom team et al	http://git.gnumonks.org/cgi-bin/gitweb.cgi?p=airprobe.git
gr-smartnet (WIP)	Trunking RX	Nick Foster	http://www.reddit.com/r/RTLSDR/comments/us3yo/rtlsdr_smartnet/ Notes from the author
gr-air-modes	ADS-B RX	Nick Foster	https://www.cgran.org/wiki/gr-air-modes call with --rtlsdr option
Linrad	SDR GUI	Leif Asbrink (SM5BSZ)	http://www.nitehawk.com/sm5bsz/linuxdsp/hware/rtlsdr/rtlsdr.htm DAGC changes were applied to librtlsdr master
gr-ais (fork)	AIS RX	Nick Foster Antoine Sirinelli Christian Gagneraud	https://github.com/chgans/gr-ais
GNSS-SDR **(NEW)**	GPS RX (Realtime!)	Centre Tecnològic de Telecomunicacions de Catalunya	Documentation and http://www.gnss-sdr.org
LTE-Cell-Scanner **(NEW)**	LTE Scanner	James Peroulas	https://github.com/Evrytania/LTE-Cell-Scanner
Simulink-RTL-SDR **(NEW)**	MATLAB/Simulink wrapper	Michael Schwall Sebastian Koslowski Communication Engineering Lab (CEL) Karlsruhe Institute of Technology (KIT)	http://www.cel.kit.edu/simulink_rtl_sdr.php
gr-scan **(NEW)**	Scanner	techmeology	http://www.techmeology.co.uk/gr-scan/

Using our lib? Tell us! Don't? Tell us why! :)



Multiple GMR-carriers can be seen in a spectrum view with the full 3.2 MHz bandwidth (at 3.2 MS/s).

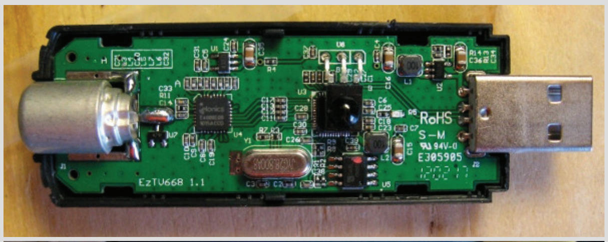
Credits

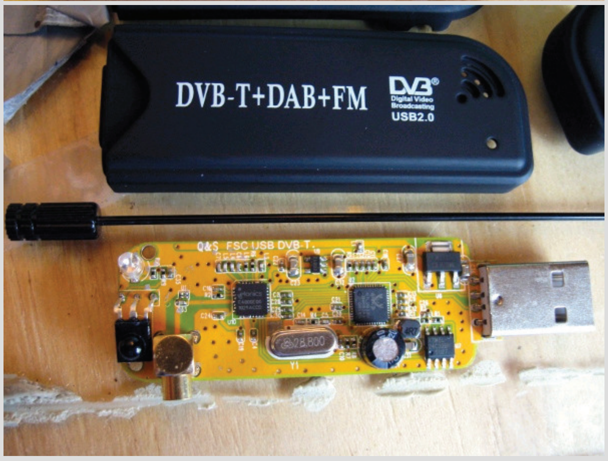
rtl-sdr is developed by Steve Markgraf and Dimitri Stolnikov, with contributions by Kyne Keen, Hoernchen, Christian Vogel and Harald Welte.

Attachments

- rtl2832-cfile.png (23.8 KB) - added by *steve-m* 7 months ago.
- ezcap_top.jpg (177.7 KB) - added by *steve-m* 7 months ago. “top view of the ezcap PCB”
- rtl2832-cfile.grc (8.6 KB) - added by *steve-m* 7 months ago. “GRC flowgraph for the RTL2832 file format”
- rtl-sdr-gmr.png (42.4 KB) - added by *steve-m* 7 months ago. “spectrum view of GMR carriers”
- EZTV666.JPG (163.3 KB) - added by *laforge* 6 months ago. “Similar but smaller EZTV 666 receiver”
- rtl-sdr.2.pdf (1.8 MB) - added by *laforge* 3 months ago. “Presentation given at FreedomHEC 2012 Taipei”
- osmosource.png (24.5 KB) - added by *horiz0n* 3 months ago. “gr-osmosdr sources”
- RelWithDebInfo.zip (1.0 MB) - added by *Hoernchen* 2 days ago. “-”

GNU Radio and rtl-sdr with Realtek RTL2832U/Elonics E4000 software defined radio receiver.





Originally meant for television reception it was discovered by V4L/DVB kernel developer Antti Palosaari that these devices can output unsigned 8bit I/Q samples at ~3.2 MS/s after oversampling and decimating the 28.8MS/s to the RTL baseband. But 2.4 MS/s is the max recommended to avoid losing samples. The dongles with an E4000 tuner can range between 54-2200 MHz (in my experience) with a gap over 1100-1250 MHz in general. The tuner error is usually <100Khz and usually more towards 50Khz, relatively stable once warmed up, and stable from day to day for a given dongle. The actual output is interleaved; so one byte I, then one byte Q with no header or metadata (timestamps). For the best information stop reading this page right now and go to the rtl-sdr wiki - OsmoSDR by the Osmocom people who created,

- librtlsdr - contains the major part of the driver
- rtl_sdr, rtl_tcp - command line capture tools
- gr-osmosdr - gnuradio source block

Freenode's ##rtlsdr and reddit.com/r/rtlsdr were also helpful in getting the software running. They put up with lots of ignorant questions. Below are my notes on learning how to use GNU Radio, GNU Radio Companion, and gr-osmosdr source or 3rd party apps using librtlsdr using libusb-1.0.

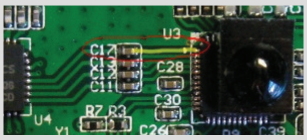
2012-08-22: The E4000 tuner datasheet has been released into the wild. Elonics-E4000-Low-Power-CMOS-Multi-Band-Tuner-Datasheet.pdf, but...

> *"All the ones that are documented in the DS are 'explained'in the driver header file ... And the rest, the datasheet call them "Ctrl2: Write 0x20 there" and no more details"*

I bought two of those rtlsdr usb dongles ($20 each). The (yellow) Newsky TV28T was from ebay and arrived within two weeks, well before the DealExtreme dongle arrived. The (green) ezcap EzTV668 was from DealExtreme, payed for on March 25th and delivered on May 20th. The antenna connector on the Newsky dongle I have is MCX. I used to think the Newsky TV28T was missing a protection diode U1 (reddit thread on protection diode replacements) but apparently it is just very, very tiny (0203 sized or smaller). The antenna connector on the ezcap is IEC-169-2, Belling-Lee. I use a PAL Male to F-Connector Female for other antenna with the ezcap. In normal (not HF) mode the ezcap has a frequency offset of about ~44 Khz (+-1Khz) or ~57 PPM from reality as determined by checking against a local 751 Mhz LTE cell using LTE Cell Scanner. Here's a live/updating plot of frequency offsets in PPM over a week.

2012-09-07: Experimental support for dongles with the Rafael Micro R820T tuner that started appearing in May has been added to rtl-sdr source base by stevem. These tuners are supposed to cover 40Mhz to roughly 1.75GHz (with no l-band gap) (locking again at 1850-1860 and 1900-2100). They also don't have the DC spike caused by the I/Q imbalance since they use a different, non-zero, IF. On the other hand, they might have image aliasing due to being superheterodine receivers. See stevem's tuner comparisons. I just ordered one selling for $11.65 shipped, it should be here by October. On 2012-09-20 the R820T datasheet was leaked to the ultra-cheap-sdr mailing list. The official range is 42-1002 Mhz with a 3.5 dB noise figure.

High Frequency (0-30Mhz) Mod



Steve Markgraf of Osmocom has created an experimental software and hardware modification to receive 0-30Mhz directly by connecting a short 2-3m wire to the right side of capacitor 17 (on EzTV668 1.1, at least) that goes to pin 1 of the RTL2832U. That's the one by the dot on the chip surface. Apparently even pressing a wet finger onto the capacitor can pick up strong AM stations. This bypasses static protection among other things so there's a chance of destroying your dongle.

As of now it'll only work at 2.048 MS/s sample rate. When using the modified rtlsdr if the library receives a frequency setting under 30Mhz it disables the E4000 tuner (high impedance on input) and does direct sampling with the realtek chip. For the default frequency ranges the E4000 is left alone and everything works normally (excepting potential interference from the HF antenna). The frequency offset during direct sampling mode will vary with a periodicity due to something resulting from the rtl2832u quadrature output direct digital synthesis frequency generator. A recent test performed by mikikg with a Marconi 2019A signal generator shows bad frequency errors below 10Mhz with decent behavior above. reddit: announcement, further discussion.

GNU Radio, RTLSDR, DSP, and Antenna Links

- rtl-sdr wiki - OsmoSDR
- Known applications - rtl-sdr wiki
- http://www.reddit.com/r/rtlsdr
- Ultra Cheap SDR - google group
- AB9IL's rtl2832 software defined radio overview

About

I enjoy recursion, dissipating local energy gradients, lipid bilayers, particle acceleration, heliophysics instrumentation and generally anything with a high rate of change in current. This site is a combination of my efforts to archive what I find interesting and my shoddy attempts to implement the aformentioned without a budget.

Recent Music

np: Deftones - Pink Maggit

- Libre.fm
- Last.fm (old)

Recently Read

- Frank Close - Neutrino
- Kim Stanley Robinson - A Short, Sharp Shock
- Stephen R Donaldson - "The Gap Cycle" series
- Terry Pratchett & Stephen Baxter - The Long Earth
- Ernest Cline - Ready Player One
- L E Modesitt Jr - Gravity Dreams
- Hannu Rajaniemi - The Quantum Thief
- Chang Goon Chung - The Fat Years
- James SA Corey - Leviathan Wakes
- Project Itoh - Harmony

Contact

email * at *superkuh.com where you use the word prefixed with the asterix for the asterix

PGP Key

TorChat: fri6mj44l5bujjyp

IRC as 'superkuh'

- Freenode
- OFTC
- ShadowWorld
- OperaNet
- Quakenet

History

This site was previously located at superkuh.ath.cx, but that subdomain system was shut down.

Then it was at superkuh.com for a while until all data was lost. Now it's back, same place, much less content.

superkuh.bit on namecoin DNS.

- Elonics E4000 reference schematic (pdf) - this is not of the Chinese produced boards
- E4000 Benefits of DigitalTune Architecture (pdf)
- rtlsdr.org wiki
- GNU Radio Wiki
- Building GNU Radio on Ubuntu Linux - GNU Radio Wiki
- http://lists.gnu.org/pipermail/discuss-gnuradio/
- patchvonbraun's https://www.cgran.org/browser/projects
- oz9aec's GRC Examples
- https://github.com/csete/gnuradio-grc-examples/tree/master/receiver
- actual television decoding driver for linux
- balint256's GNU Radio tutorial videos
- firebyte's post on r/GNURadio
  - Radioware Documentation - GNU Radio tutorials
  - CSUN/EAFB Software Defined Radio (SDR) Senior Project
  - The Scientist and Engineer's Guide to Digital Signal Processing
  - Fundamentals of Wireless Communication
  - Tutorials in Communications Engineering
- Antenna Theory
- AT&T Archives: Similiarities of Wave Behavior - impedance explained

Warning: I don't know what I am doing. There are errors. Refer to the proper documentation and original websites, applications, grc and python files first.

GNU Radio RTLSDR Setup

patchvonbraun has made setting up and compiling GNU Radio and librtlsdr with all the right options very simple on Ubuntu and Fedora. Simply run, http://www.sbrac.org/files/build-gnuradio, a shell script written to automate downloading and compiling of prequisites, libraries, correct branches, udev settings, and more. I had no problems using Ubuntu 10.04 AMD64 or 12.04 32bit. If you're an OSX user then head over to titanous' homebrew-gnuradio.

```
mkdir gnuradio
cd gnuradio
wget http://www.sbrac.org/files/build-gnuradio
chmod a+x build-gnuradio
./build-gnuradio
```

An (re)install looks like this. It might be useful to save the log output for future reference. Then test it.

```
rtl_test -t
Found 1 device(s):
  0:  ezcap USB 2.0 DVB-T/DAB/FM dongle

Using device 0: ezcap USB 2.0 DVB-T/DAB/FM dongle
Found Elonics E4000 tuner
Benchmarking E4000 PLL...
[E4K] PLL not locked!
[E4K] PLL not locked!
[E4K] PLL not locked!
[E4K] PLL not locked!
E4K range: 52 to 2210 MHz
E4K L-band gap: 1106 to 1247 MHz
```

Once GNU Radio is installed the "Known Apps" list at the rtl-sdr wiki is a good place to start. Try running a third party receiver, a python file or start up GNU Radio Companion (gnuradio-companion) and load the GRC flowcharts. If you're having "Failed to open rtlsdr device #0" errors make sure something like /etc/udev/rules.d/15-rtl-sdr.rules exists and you've rebooted.

Sample rate and Gain

When setting sample rates try to use one of the values shown in, http://cgit.osmocom.org/cgit/gr-osmosdr/tree/lib/rtl/rtl_source_c.cc#n319

```
osmosdr::meta_range_t rtl_source_c::get_sample_rates()
{
  osmosdr::meta_range_t range;

  range += osmosdr::range_t( 250000 ); // known to work
  range += osmosdr::range_t( 1000000 ); // known to work
  range += osmosdr::range_t( 1024000 ); // known to work
  range += osmosdr::range_t( 1800000 ); // known to work
  range += osmosdr::range_t( 1920000 ); // known to work
  range += osmosdr::range_t( 2048000 ); // known to work
  range += osmosdr::range_t( 2400000 ); // known to work
  range += osmosdr::range_t( 2600000 ); // may work
  range += osmosdr::range_t( 2800000 ); // may work
  range += osmosdr::range_t( 3000000 ); // may work
  range += osmosdr::range_t( 3200000 ); // max rate

  return range;
}
```

stevem also did gain measurement tests with a few dongles using some equipment he had to transmit a GSM FCCH peak, "which is a pure tone." This includes the E4000 and R820T tuners. In addition he measured the mixer, IF and LNA for the R820T.

Updating

When updating you can just repeat the install instructions (or just gnuradio (no UHD) with 'gnuradio_build') which is simple but long. patchvonbraun updates the build-gnuradio script regularly so be sure to get the latest. This'll do things like ldconfig for you. If you are installing libraries for gnuradio manually make sure to 'sudo ldconfig'.

---

rtl-sdr supporting receivers, associated tools

- patchvonbraun (Marcus Leech)'s multimode:
  - AM, FM, USB, LSB , WFM. TV-FM, PAL-FM. Very nice, easy to use (screenshots: main, scanning). It has an automated scanning and spectral zoom features with callbacks to click on the spectrogram or panorama to tune to the frequency of interest. There's a toggle for active gain control too. The way to get it is,

    ```
    svn co https://www.cgran.org/svn/projects/multimode
    ```

    then instead of using GRC, just run the multimode.py as is.

    ```
    make install
    python multimode.py
    ```

    If you run it outside of the svn created directory you might need to append ~/bin to pythonpath to find the helper script. If you used build-gnuradio it'll tell you what this is at the end of the install.

    ```
    set PYTHONPATH=$PYTHONPATH:/usr/local/lib/python2.6/dist-packages:~/bin
    ```

    Alternately set it in your ~/.bashrc. If you do the below make sure to reload in the terminal by "source ~/.bashrc")

    ```
    PYTHONPATH=/usr/local/lib/python2.6/dist-packages:~/bin
    export PYTHONPATH;
    ```

    When setting the sample rate it is rounded-down to a multiple of 200 Ksps so the decimation math works out.

    ```
    python multimode.py --srate=2.4M
    python multimode.py --help
    ```

    If you have overruns like "OOOOoo..." then try reducing the sample rate or pausing the waterfall or spectrum displays.

    > *"The audio subsystem uses 'a' as the identifier, and UHD uses 'u'. With RTLSDR, it'll issue 'O' when it experiences an overrun. Which means that your machine isn't keeping up with the data stream. Sometimes buffering helps, but only if your machine is right on the edge of working properly. If it really can't, on average "keep up", no amount of buffering will help."*

If you have overruns like "aUaUaUaUa" or just "aaa" then your audio subsystem isn't working quite right. Use "aplay -l" to get a list of the devices on your system.

```
aplay -l
```

The hw:X,Y comes from this mapping of your hardware -- in this case, X is the card number, while Y is the device number. Or you can use "pulse" for pulseaudio. Try specifying,

```
python multimode.py--ahw hw:0,0
```

- gqrx:
  - gqrx install notes and edit - Originally written by Alexandru Csete OZ9AEC "gqrx is an experimental AM, FM and SSB software defined receiver". The original version did not have librtlsdr support so changes were made by a number of others to add it. A couple weeks later Csete added gr-osmosdr support to the original.

```
git clone https://github.com/csete/gqrx.git
cd gqrx
# on Ubuntu/Debian, sudo apt-get install qtcreator , if you don't have it.
qtcreator gqrx.pro      # press the build button (the hammer)
# or avoid qtcreator and do it manually.
qmake
make
```

- SDR#:
  - Written by prog (Youssef) for Windows. It is very nice at the rtlsdr.org wiki there is a guide to running the original C# version under linux with mono. Mono is a little slow, but if you restrict the sample rate it works fine. It's probably the easiest program to use with good I/Q correction to mitigate the DC hump. If your platform supports it, use it. In 0.25 MS/s mode it can even run on my old 1.8Ghz P4 laptop. On linux the waterfall doesn't "fall" for me. Instead the entire vertical space just shows the current frame. Others report resizing the window sometimes fixes this but I haven't been able to find the sweet spot. To use it run rtl_tcp first then connect to the stream in sdr#. To use the USB plugin make sure it's in the same directory and then uncomment it's line in SDRSharp.exe.config. Whenever you're doing any kind of 'digital modes' make sure the 'minoutputsamplerate' in the sdrsharp.exe.config file is 48000khz and uncheck filter audio in the audio section.

- gr-air-modes:
  - A decoder of aviation transponder Mode S including ads-b reports near 1090 Mhz. It can be coupled to software to show plane positions in near real time (ex: VirtualRadar). This works under mono on Ubuntu 12.04 but not 10.04. Originally written by Nick Foster (bistromath) and adapted to rtlsdr devices first by Steve Markgraf (stevem), bistromath later added rtlsdr support. Here's an example of basic decoding done with the stock antenna on the early version by stevem. Nowdays it's better to use bistromaths'.

```
git clone https://github.com/bistromath/gr-air-modes.git
```

Here's an example of install process and first run looks like.

```
# -d stands for rtlsdr dongle, location is "North,East"
uhd_modes.py -d --location "45,-90"
```

To use with virtual radar output add the below -P switch. Then open up virtualradar with mono and go to tools->options->basestation and put in the IP of the computer running uhd_modes. There are not many compatible planes in the USA so far so even if you are seeing lots of Mode-S broadcast in uhd_modes you might not see anything in virtualradar. Sometimes my server is running at superkuh.com:81/VirtualRadar/GoogleMap.htm.

```
uhd_modes.py -d --location "45,-90" -P
mono VirtualRadar.exe
```

Modesbeast has a diagram of a colinear antenna for this purpose.

- linrad:
  - A guide on how to use linrad with rtl-sdr, and a modification to the the librtlsdr e4000 tuner code to disable digital active gain! reddit thread

> *"I tried various bits blindly and found a setting that eliminates the AGC in the RTL2832 chip. That is a significant part of the performance improvement."*

As of 2012-07-07 this feature was added to the main (librtlsdr) driver as well.

- LTE Cell Scanner:
  - *"This is an LTE cell searcher that scans a set of downlink frequencies and reports any LTE cells that were identified. A cell is considered identified if the MIB can be decoded and passes the CRC check."*

The author had only tested it on Ubuntu 12.04 but with some frustrating work replacing cmake files and compiling dependencies I made it work on 10.04. This software is very useful to get your dongle's frequency offset reliably.

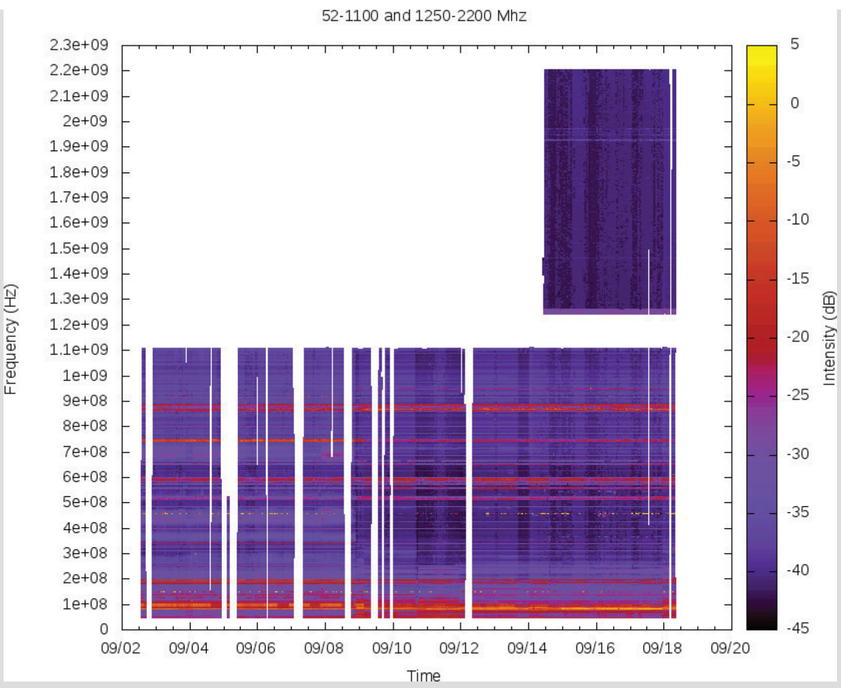
- my DongleLogger:
  - These scripts do automatic generation of spectrograms, frequency strength logging, time series plots, and spectral maps over arbitrary frequency ranges with frequency calibration using cell signals.

You can see the gallery output at, http://erewhon.superkuh.com/gnuradio/live/

Other tools

- Thomas Sailer's multimon, "Linux Radio Transmission Decoder" which I use to (try to) decode pager transmissions around 930Mhz. And more recently Dekar's multimonNG, a fork with improved error correction, more supported modes, and *nix/osx/windows support. Dekar also supplied a GRC receiver for pagers to decode pager transmissions in real-time using fifos.

DongleLogger: my pyrtlsdr lib based spectrogram and signal strength log and plotter



Purpose:

Automatic generation of and html gallery creation of spectrograms, frequency strength logging, time series plot, and full range spectral maps over arbitrary frequency ranges with frequency calibration using cell signals.

http://erewhon.superkuh.com/gnuradio/live/ - click spectrograms for time series plot

Details:

I don't know much python but the python wrapper for librtlsdr pyrtlsdr was a bit easier to work with than gnu radio when I wanted to do simple things.

```
git clone https://github.com/roger-/pyrtlsdr.git
```

I have used the "test.py" matplotlib graphical spectrogram generator that came with pyrtlsdr as a seed from which to conglomerate my own program for spectrum observation and logging. Since I am not very good with python I had to pull a lot of the logic out into a perl script. So everything is modular. As of now the python script generates the spectrogram pngs and records signal strength (and metadata) in frequency named logs. It is passed lots of arguments.

These arguments can be made however you want, but I wrote a perl script to automate it along with a few other useful things. It can generate a simple html gallery of the most recent full spectral map and spectrograms with each linked to the log of past signal levels. Or it can additionally generate gnuplot time series pngs (example) and link those intead of the raw logs. It also calls LTE Cell Scanner and parses out the frequency offset for passing to graphfreq.py for correction. I have it running on my home machine but because of the bandwidth required I have rsync updating the public mirror with a big pipe every ~40 minutes. I switched to a spiral antenna on day 253 of 2012.

Modifying pyrltsdr

As it is pyrtlsdr does not have the get/set functions for frequency correction even if I sent the PPM correct from the perl script. Since the hooks (?) were already in librtlsdr.py (line 60-66) but just not pythonized in rtlsdr.py they were easy to add to the library. These changes are required to use frequency correction and make the int variable "err_ppm" available.

I forked roger-'s pyrtlsdr on github and added them there for review or use, https://github.com/superkuh/pyrtlsdr/commit/ffba3611cf0071dee7e1efec5c1a582e1e344c61.

Download:

- graphfreqs.py - the pyrltsdr using script; it does the sampling and logging
- radioscan.pl - manages graphfreqs, parses logs, makes plots, gets frequency corrections, generates gallery

Or you can download everything together (even the modified pyrtlsdr library) in a compressed archive: donglelogger.tar.gz

Running it:

graphfreqs.py

You have to have the modified pyrtlsdr with the get/set functions for frequency correction. LTE Cell Scanner should also be installed so the "CellSearch" binary is available. Then download the two scripts above and put them in the same directory.

To call the spectrogram/log generator by itself for 431.2 Mhz at 2.4MS/s with a gain of 30 and frequency correction of 58 PPM use it like,

```
python graphfreqs.py 431200000 2400000 30 58
```

I've disabled the matplotlib (python) per frequency spectrogram plots for frequencies over 1 Ghz because there's not much going on up there. Also, the x-axis ticks and labels become inaccurate for some reason.

Logs and format

The signal strength logs, named by frequency (e.g. 53200000.log), use unix time and are comma seperated with newlines after each entry. In order of columns it is: unix time , relative signal level , gain in dB, PPM correction.

```
1345667680.28 , -34.65 , 29 , 57
1345667955.59 , -34.67 , 29 , 57
1345668004.37 , -34.55 , 29 , 57
1345668110.06 , -33.88 , 29 , 57
```

It also generates a log file with all frequencies for use with gnuplot, all.log. This file has unixtime first, then frequency, then gain and ppm error.

```
1347532002.5 52000000 -14.84 29.0 58
1347532004.88 53200000 -17.84 29.0 58
1347532007.04 54400000 -17.98 29.0 58
1347532009.04 55600000 -19.78 29.0 58
1347532011.02 56800000 -24.04 29.0 58
1347532012.98 58000000 -26.21 29.0 58
1347532014.92 59200000 -25.10 29.0 58
```

radioscan.pl

The radioscan.pl script is used to automate calling graphfreqs in arbitrary steps. To generate plots and signal strength for 52 Mhz to 1108 Mhz with a gain of 30, sample rate of 2.4MS/s, and an interval between center frequencies of 1.2 Mhz, call it like,

```
$ perl radioscan.pl -flist "52-1108,1248-2200" -g 30 -r 2400000 -s 1.2
```

cli switches/options

```
-flist "52-1108,1248-2200"  :: sets of frequency ranges to scan.
-g 30                       :: gain
-s                          :: interval between center frequencies
-r 2400000                  :: sample rate
-d1 /path/here              :: path to where the scripts are if now pwd
-d2 /path/here              :: path to the directory to put logs, plots, gallery
-c 751                      :: LTC Cell scanner frequency offset correction, takes freq in Mhz of base cell
-w                          :: turn on web gallery generation
-p                          :: turn on gnuplot time series charts for every freq
-m                          :: generate full range spectral map using all.log
-mr "52-1108,1248-1400,1900-2200"       :: set of frequency ranges to plot as a another spectral map
```

Because I can use the default directories I keep it running like the below, but anyone else should make sure to set -d2.

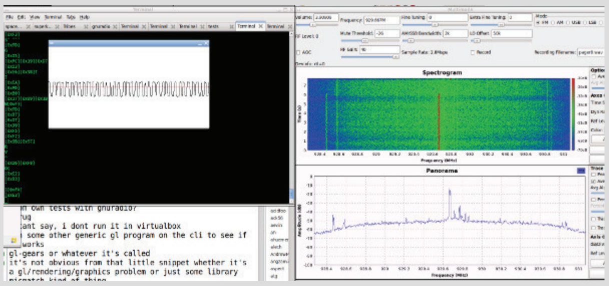
```
$ while true; do perl radioscan.pl -flist "52-1108,1248-2200" -g 30 -r 2400000 -s 1.2 -w -c 751 -p -m -mr "52-1108"; sleep 1; done;

Running pyrtl graphfreq batch job 52 to 1108 Mhz at 1.2 Mhz spacings.
Using LTE Cell Scanner to find frequency offset from 751 Mhz station...Found Elonics E4000 tuner
42.6k frequency offset. Correcting 56 PPM.
Generating spectral map.
Generating another spectral map over only 52-1108.

python ./graphfreqs_offset.py 52000000 2400000 30 56
Found Elonics E4000 tuner
python ./graphfreqs_offset.py 53200000 2400000 30 56
Found Elonics E4000 tuner
python ./graphfreqs_offset.py 54400000 2400000 30 56
Found Elonics E4000 tuner
...
python ./graphfreqs_gnuplot.py 316000000 2400000 30 56
Found Elonics E4000 tuner
Dongle froze, reseting it's USB device...
Resetting USB device /dev/bus/usb/001/017
Reset successful
python ./graphfreqs_gnuplot.py 317200000 2400000 30 56
Found Elonics E4000 tuner
..
Generating page, moving images.

starting rsync...
```

Tuner/USB freeze solution with unplugging

Since graphfreqs.py's initializing and calling of rtl-sdr happens so frequently there are sometimes freezes. To fix these the USB device has to be reset. In the past I would accomplish this by un and re-plugging the cord manually. But that meant lots of downtime when I was away or sleeping. So, I've added in a small C program to the perl script using Inline::C that exposes a function, resetusb(). It is used if the eval loop around the graphfreqs call takes more than 10 seconds. *This means you need Inline::C to run this script*.

```
sub donglefrozen {
        my $usbreset;
        my @devices = split("\n",`lsusb`);
        foreach my $line (@devices) {
                if ($line =~ /\w+\s(\d+)\s\w+\s(\d+):.+Realtek Semiconductor Corp\./) {
                        $usbreset = "/dev/bus/usb/$1/$2";
                        resetusb($usbreset);
}}}

__END__
__C__
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/ioctl.h>
#include <linux/usbdevice_fs.h>

int resetusb(char *dongleaddress)
{
        const char *filename;
        int fd;
        int rc;
        filename = dongleaddress;
        fd = open(filename, O_WRONLY);
        if (fd < 0) {
                perror("Error opening output file");
                return 1;
        }
        printf("Resetting USB device %s\n", filename);
        rc = ioctl(fd, USBDEVFS_RESET, 0);
        if (rc < 0) {
                perror("Error in ioctl");
                return 1;
        }
        printf("Reset successful\n");
        close(fd);
        return 0;
}
```

Decoding Pager Data with multimon and GRC receivers



```
n own tests with gnuradio?
ug
ant say, i dont run it in virtualbox
 some other generic gl program on the cli to see if
 works
gl-gears or whatever it's called
it's not obvious from that little snippet whether it's
a gl/rendering/graphics problem or just some library
mismatch kind of thing
```

Written by Thomas Sailer, HB9JNX/AE4WA, multimon (multimon.tar.bz2) supports decoding a large number of pager modulations.

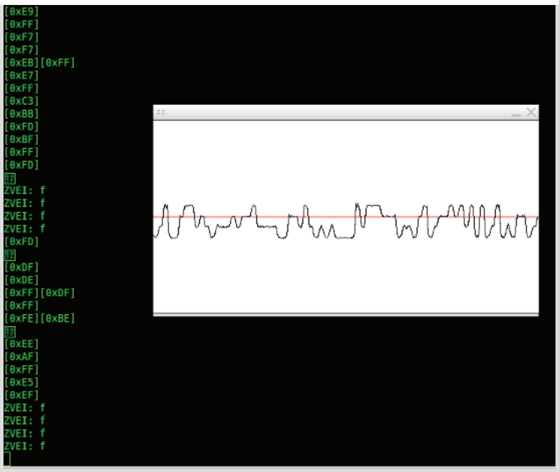
On June 29th 2012 dekar told me about his updated fork of multimon, multimonNG, with better error correction and more modulations supported. As of right this instant those on 64bit linux should just use the existing makefile and *not* qmake or qt-creator to compile it. For the windows users (or anyone wanting more info) there's a precompiled version and blog post.

There are a few options for receivers to use with the multimon decoder. I've used patchvonbraun's multimode to save .wavs and a pager example GRC of unknown origin I modified for OsmoSDR sources linked below for raw, real time decoding.

```
./multimon /dev/audio  # something about my configuration prevents it from working properly
```

When I run it off /dev/audio something about my configuration (or lack thereof) prevents it from working. It works fine for post-processing when using patchvonbraun's GNU Radio "multimode" script to save the FM decode audio to disk as wav.

```
./multimon -t wav pager.wav
```



```
[0xE9]
[0xFF]
[0xF7]
[0xF7]
[0xEB][0xFF]
[0xE7]
[0xFF]
[0xC3]
[0xBB]
[0xFD]
[0xBF]
[0xFF]
[0xFD]
▯
ZVEI: f
ZVEI: f
ZVEI: f
ZVEI: f
[0xFD]
▯
[0xDF]
[0xDE]
[0xFF][0xDF]
[0xFF]
[0xFE][0xBE]
▯
[0xEE]
[0xAF]
[0xFF]
[0xE5]
[0xEF]
ZVEI: f
ZVEI: f
ZVEI: f
ZVEI: f
```

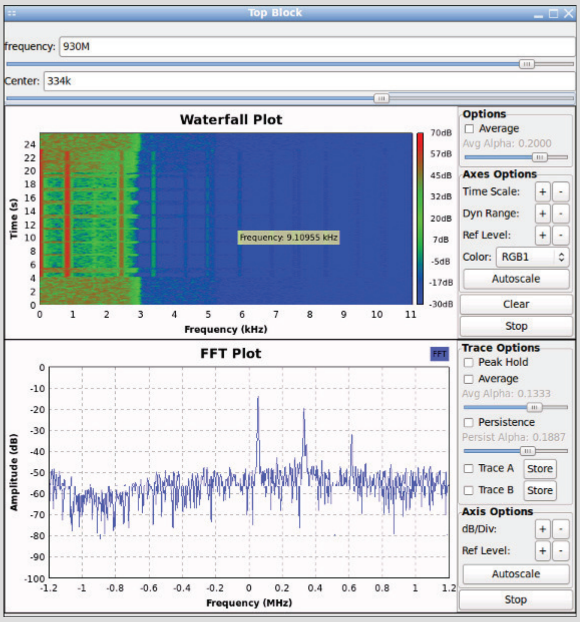
So far I get non-printable characters and repeated instances of "ZVEI: f" so those might be properly decoded Selcall tone set. The "f" would indicate, depending on ZVEI version either 680 Hz or 2600 Hz. Since it seemed to be ZWEI, I then used,

```
./multimon -a ZVEI -t wav pager.wav
```

That returns "ZWEI: f" over and over. I haven't tried decoding known good sample recordings from other people.

A better way: real time decoding w/dekar's pager_fifo

Dekar's multimonNG, a fork with improved error correction, more supported modes, and *nix/osx/windows support. In the screenshots below the signal is not pocsag. I thought it might be zwei but now I'm not so sure it's even pager data. Test samples of pocsag that Dekar links on his blog decode just fine.



Top Block

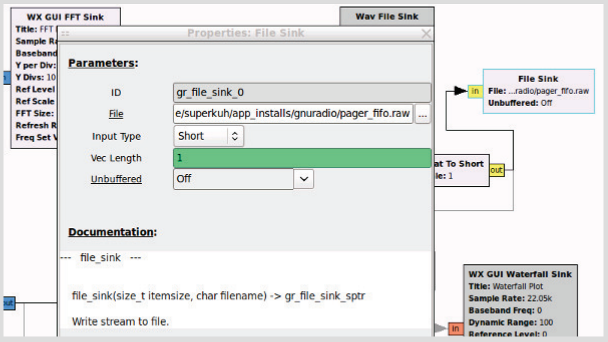
frequency: 930M

Center: 334k

pager_fifo_web.grc

```
mkfifo /tmp/pager_fifo.raw
./multimonNG -t raw /tmp/pager_fifo.raw
gnuradio-companion pager_fifo_web.grc
```

In order to decode the pager data in real time you should use a first-in first-out file (fifo). Dekar's pager_fifo is designed to do that but you'll need to set the correct file paths for the File Sink yourself. In the copy downloadable here the File Sink's path is set to "/tmp/pager_fifo.raw". You should be able to run it without editing once you've made that fifo. Make sure to start multimon reading the fifo before you begin GRC and execute the receiver.



Properties: File Sink

Parameters:

ID: gr_file_sink_0

File: e/superkuh/app_installs/gnuradio/pager_fifo.raw

Input Type: Short

Vec Length: 1

Unbuffered: Off

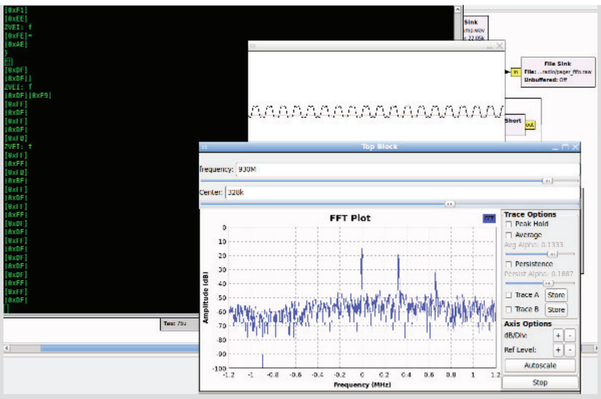
Documentation:

--- file_sink ---

file_sink(size_t itemsize, char filename) -> gr_file_sink_sptr

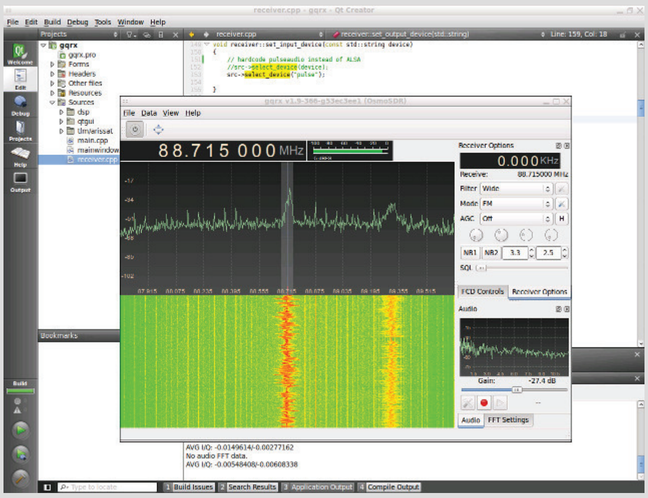
Write stream to file.

In my personal copy of dekar's pager_fifo the file and audio sinks are enabled while the waterfall, wav, and other sinks are disabled. To enable the disabled (grey) block select them and press 'e' ('d to disable). The audio sink is set to pulseaudio ("pulse").



---

gqrx install notes



When I wrote this up the original version by csete didn't support the hardware yet but mathis_, phirsch, Hoernchen, and perhaps others I've missed from ##rtlsdr on freenode had added librtlsdr support to gqrx; their repos are still listed by commented out. These days csete has added in rtlsdr support so you can use his original repository.

```
#git clone https://github.com/phirsch/gqrx
# matthis is the only version I've personally tested (old)
git clone https://github.com/mathisschmieder/gqrx
# the original; supposed to have librtlsdr support now (new)
git clone https://github.com/csete/gqrx.git
cd gqrx
# on Ubuntu, sudo apt-get install qtcreator , if you don't have it.
qtcreator gqrx.pro      # press the build button (the hammer)
# Avoid qtcreator doing it manually.
qmake
make

./gqrx
```

Use with Ubuntu 10.04 and distros with old Qt < 4.7

You will almost certainly not get this error. But, someone might, so I'm leaving it here to be indexed.

If you're like me and run an older distribution then your Qt libraries will be out of date and lack a function required for generating the name of the files to be saved when recording.

```
/home/superkuh/app_installs/gnuradio/gqrx/gqrx/qtgui/dockaudio.cpp:100: error: ‘currentDateTimeUtc’ is not a member of ‘QDateTime’
```

Initially I thought it was a qtcreator thing so I tried to get more information by doing it manually,

qmake
make
g++ -c -pipe -O2 -I/usr/local/include/gnuradio -I/usr/local/include -I/usr/local/include -I/usr/local/include/gnuradio -D_REENTRANT -D_REENTRANT -I/usr/include/libusb-1.0 -Wall -W -D_REENTRANT -DQT_NO_DEBUG -DQT_NO_DEBUG_OUTPUT -DVERSION="\"0.0\"" -DQT_NO_DEBUG -DQT_GUI_LIB -DQT_CORE_LIB -DQT_SHARED -I/usr/share/qt4/mkspecs/linux-g++ -I. -I/usr/include/qt4/QtCore -I/usr/include/qt4/QtGui -I/usr /include/qt4 -I. -I. -o dockaudio.o qtgui/dockaudio.cpp
qtgui/dockaudio.cpp: In member function ‘void DockAudio::on_audioRecButton_clicked(bool)’:
qtgui/dockaudio.cpp:100: error: ‘currentDateTimeUtc’ is not a member of ‘QDateTime’

make: *** [dockaudio.o] Error 1

To get it to compile on these systems you'll have to do the below. (edit: This little change is now added into phirsch's.)

Ubuntu 10.04 has old Qt libs and gqrx uses a function call not in them. So, while I was waiting for Qt 4.74 to compile I decided to try a hack. I removed that function call with a static string of text. [edit] I later found comparable functions for Qt 4.6 and older.

If you are using qtcreator like the docs suggest you can double click on the error and go to the line. If not, it was in ./Sources/qtgui/dockaudio.cpp replace,

```
void DockAudio::on_audioRecButton_clicked(bool checked)
{
    if (checked) {
        // FIXME: option to use local time
        lastAudio = QDateTime::currentDateTimeUtc().toString("gqrx-yyyyMMdd-hhmmss.'wav'");
```

With something like this.

```
void DockAudio::on_audioRecButton_clicked(bool checked)
{
    if (checked) {
        // FIXME: option to use local time
        // use functions compatible with older versions of Qt.
        lastAudio = QDateTime::currentDateTime().toUTC().toString("gqrx-yyyyMMdd-hhmmss.'wav'");
```

And it'll compile and run correctly on my specific machine.

Compiling LTE Cell Scanner on Ubuntu 10.04

Before starting make sure to have a fortran compiler, FFTW, BLAS, and LAPACK libraries installed from the repositories.

```
sudo apt-get install automake autoconf libtool libfftw3-3 libfftw3-dev gfortran libblas3gf libblas-dev liblapack3gf liblapack-dev libatlas-base-dev
```

If you're using 12.04 just follow the instructions on the github page and everything is trivial. For 10.04 (lucid) users the the initial hurdle is cmake. LTE Cell Scanner requires cmake 2.8.8 and Ubuntu 10.04 only has 2.8 the finding of BLAS and LAPACK libraries will fail like,

```
cmake ..
-- Found ITPP: /usr/lib64/libitpp.so
CMake Error at /usr/share/cmake-2.8/Modules/FindBLAS.cmake:45 (message):
  FindBLAS is Fortran-only so Fortran must be enabled.
Call Stack (most recent call first):
  CMakeLists.txt:29 (FIND_PACKAGE)
```

You can see my installation notes before I figured it out. To fix it I searched for people complaining of similar problems on other projects and then replaced my *system* files with theirs, FindBLAS.cmake.

```
sudo cp /usr/share/cmake-2.8/Modules/FindBLAS.cmake /usr/share/cmake-2.8/Modules/FindBLAS.cmake.bak
sudo cp FindBLAS.cmake /usr/share/cmake-2.8/Modules/
```

LAPACK will also fail this way. I used this arbitray cmake file, http://code.google.com/p/qmcpack/source/browse/trunk/CMake/FindLapack.cmake?r=5383

```
sudo cp /usr/share/cmake-2.8/Modules/FindLAPACK.cmake /usr/share/cmake-2.8/Modules/FindLAPACK.cmake.bak
sudo FindLAPACK.cmake /usr/share/cmake-2.8/Modules/FindLAPACK.cmake
```

After fixing the cmake issues compile and install the latest IT++ (ITPP 4.2). Make sure to completely remove the old ITPP 4.0.7 libraries from the Ubuntu repository. When LTE Cell scanner compiles you can go back and restore the .bak cmake files. The rate of scan is about 0.1 Mhz per 10 seconds.

```
./CellSearch -v -s 751e6 -e 751e6
LTE CellSearch v0.1.0 (release) beginning
  Search frequency: 751 MHz
  PPM: 100
  correction: 1
Found Elonics E4000 tuner
Waiting for AGC to converge...
Examining center frequency 751 MHz ...
Capturing live data
  Calculating PSS correlations
  Searching for and examining correlation peaks...
  Detected a cell!
    cell ID: 414
    RX power level: -17.0733 dB
    residual frequency offset: 43592.8 Hz
  Detected a cell!
    cell ID: 415
    RX power level: -20.8041 dB
    residual frequency offset: 43592.3 Hz
  Detected a cell!
    cell ID: 209
    RX power level: -28.8524 dB
    residual frequency offset: 43581.2 Hz
Detected the following cells:
C: CP type ; P: PHICH duration ; PR: PHICH resource type
CID      fc   foff RXPWR C nRB P  PR CrystalCorrectionFactor
414    751M  43.6k -17.1 N  50 N one 1.0000580496943698439
415    751M  43.6k -20.8 N  50 N one 1.0000580490797574829
209    751M  43.6k -28.9 N  50 N one 1.0000580342133056355
```

To disambiguate: I have only seen positive frequency offsets, but people on ##rtlsdr have reported negative offsets.

Slightly altered GNU Radio Companion flowcharts

"The GUI stuff in Gnu Radio was rather an afterthought. Nobody really expected that you'd use it to build actual applications, but rather just use it as a way of making "test jigs" for your signal flows."

This section is my notes on how I made basic examples work, and how I edited those examples in very simple and often broken ways. Also, since gqrx, multimode, and other intergrated receivers came out I don't see any need to update these as things change. Most of this is very old.

While there are links to the originals in the summaries, these descriptions are of the versions modified by me; usually just sample rate and GUI stuff. While the sample rate or tuner width I set may be some large number, it'll become obvious what the limits of each other as you scan about and see the signal folding or mirroring. **Using sample rates above 2.4 MS/s with gr-osmosdr is not recommended**.

- FM:
  - patchvonbraun's simple (stereo) fm receiver - harder setup, best reception, best sound, 2.048 MS/s, +-600Khz fine tune
  - lindi's fm receiver easy setup, good reception, good sound, 3.2 MS/s, +-600Khz fine tune
  - superkuh's offset fm receiver - easy setup, okay reception, okay sound, 2.8 MS/s, +-900Khz tune, +-50Khz fine.
  - 2h20's beginner fm (mono) receiver - easy to understand, easy setup, okay sound, 2.8 MS/s, no fine tune
- SSB:
  - OZ9AEC's SSB Receiver - SSB rx and record to disk, seperate playback script. 1 MS/s, +-1k fine tune.

Tips

If it comes with a python file, try that first before generating one from the GRC file. When tuning, make sure to hit enter again if it doesn't work the first time or tunes to the wrong frequency. Always hit autoscale to start, and for FFT displays try using the "average" settings. I have set all audio sinks to "pulse" (pulseaudio) instead of say, "hw:0,0" (ALSA). You might have to change that. To get a list of hardwareuse "aplay -l". That'll show the various cards and devices. Use the format, "hw:X,Y" where "hw:CARD=X,DEV=Y". Some flowcharts have variables for it, others put it directly in the Audio Sink element. If you hear something interesting you can try comparing it to indentified samples from http://www.kb9ukd.com/digital/. or the windows program, Signals Analyzer. Check http://www.radioreference.com/ or http://wireless2.fcc.gov/UlsApp/UlsSearch/searchAdvanced.jsp to see what's in the USA area at a given frequency.

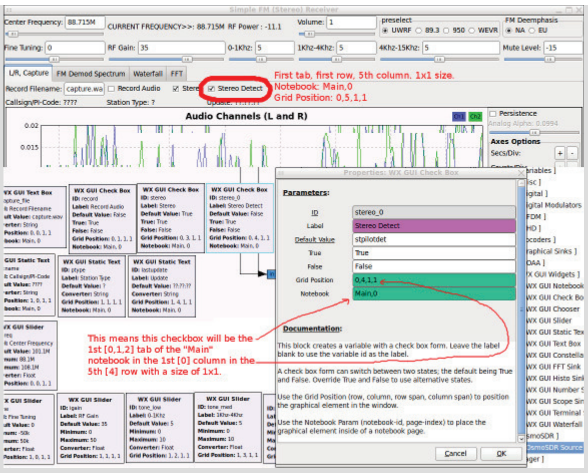
Editing

To enable a block, select it and press 'e'. To disable a block select it and press 'd'. When disabled blocks will appear darker gray.

It's easier to type in 1e6 than 1000000 so use scientific notation when you can in variable fields. If you double click on an element in a flowchart it usually includes a helpful "Documentation:" of most the variables to be set at the bottom. The GUI element grid position is a set of two pairs of numbers: "y,x,a,b" where the first pair "y,x," is position (y row, x column) and "a,b" is the span of the box. If you enter a Grid Position and it overlaps with another element it'll turn red and report the error and where the origin is of the element it overlaps with.

```
Use the Grid Position (row, column, row span, column span) to position the graphical element in the window.
```

The tab effect is done with notebooks.


First tab, first row, 5th column, 1x1 size.
Notebook: Main,0
Grid Position: 0,5,1,1

This means this checkbox will be the 1st [0,1,2] tab of the "Main" notebook in the 1st [0] column in the 5th [4] row with a size of 1x1.

Parameters:

ID	stereo_0
Label	Stereo Detect
Default Value	stpilotdet
True	True
False	False
Grid Position	0,4,1,1
Notebook	Main,0

Documentation:

This block creates a variable with a check box form. Leave the label blank to use the variable id as the label.

A check box form can switch between two states; the default being True and False. Override True and False to use alternative states.

Use the Grid Position (row, column, row span, column span) to position the graphical element in the window.

Use the Notebook Param (notebook-id, page-index) to place the graphical element inside of a notebook page.

For RTL2832 Source the minimum sample rate is ~800KS/s, it's gr-baz(?) and generally not updated. Use OsmoSDR source. It's under "OsmoSDR", not "Sources" on the right panel). It has a 1MS/s minimum sample rate. It's not recommended to use sample rates above 2.4M.

In older versions of gr-osmosdr and rtl-sdr I think automagic gain control (AGC) was on all the time so you didn't have to set the gain explicitly in the source in GRC. New versions require that and also require setting the chan 0. freq to something.

The dongles seem to have noise at their 0Hz center frequency so the best performance is from selecting a band 100-200Khz offset from the center (depending on signal type). patchvonbraun's simple_fm_rcv is a great example of that.

---

patchvonbraun's simple_fm_rcv

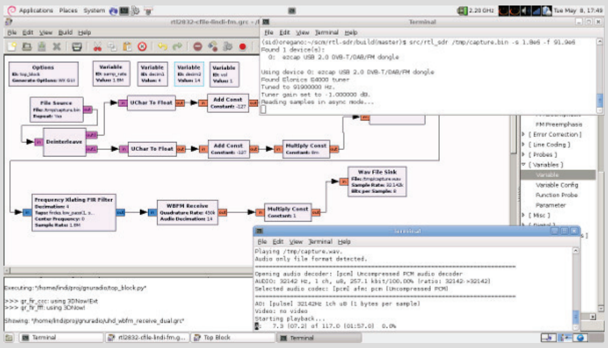
The best sounding software I've found for listening to FM is patchvonbraun's Simple FM (Stereo) Receiver. I don't think it is very simple; it includes many advanced FM specific features like extraction of the 19k (pilot) tone next to some commercial FM broadcasts. It used to do RDS, I hear, and older versions checked into CGRAN still have it, but it is removed for simplicity in this version.

```
svn co https://www.cgran.org/svn/projects/simple_fm_rcv
cd simple_fm_rcv/
cd trunk
less README
make
make install
## it'll install to ~/bin/, so I use ~/superkuh/bin below
set PYTHONPATH=/usr/local/lib/python2.6/dist-packages:/home/superkuh/bin
# run the python script
python simple_fm_rcv.py
# or edit it
gnuradio-companion simple_fm_rcv.grc
```

---

lindi's FM receiver

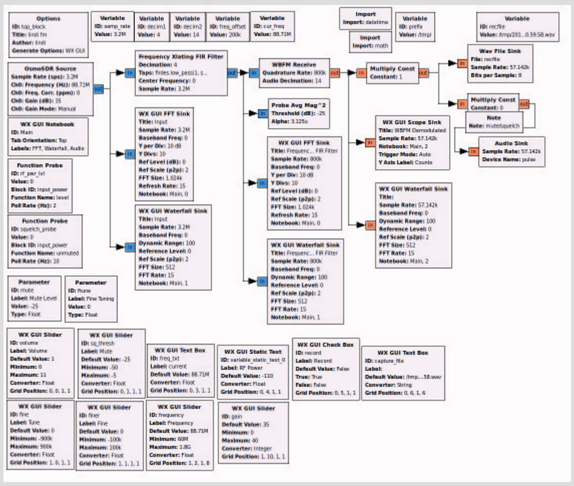
Original: http://lindi.iki.fi/lindi/gnuradio/rtl2832-cfile-lindi-fm.grc , this was an example posted to ##rtlsdr by lindi. It used a file source which was decoded to wav and saved to disk. Seen in the screenshot above.

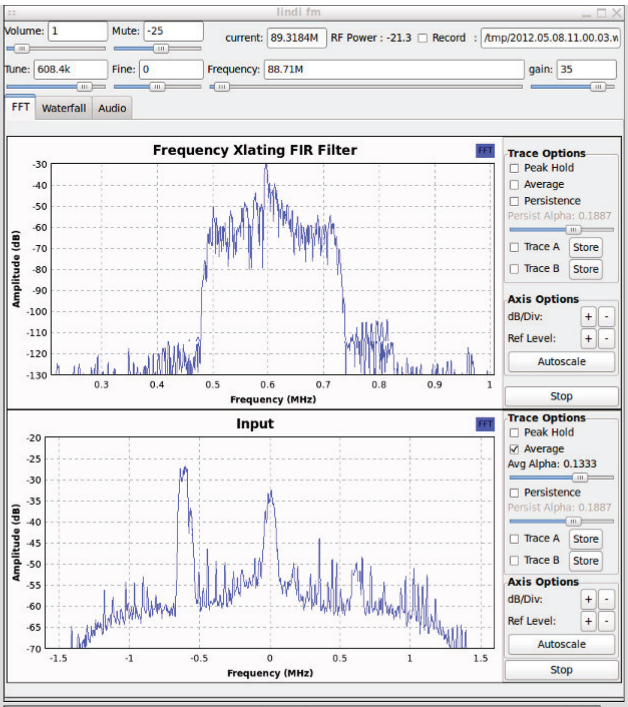

```
(sid)oregano:~/scm/rtl-sdr/build(master)$ src/rtl_sdr /tmp/capture.bin -s 1.8e6 -f 91.9e6
Found 1 device(s):
  0:  ezcap USB 2.0 DVB-T/DAB/FM dongle

Using device 0: ezcap USB 2.0 DVB-T/DAB/FM dongle
Found Elonics E4000 tuner
Tuned to 91900000 Hz.
Tuner gain set to -1.000000 dB.
Reading samples in async mode...
```

```
Playing /tmp/capture.wav.
Audio only file format detected.
==========================================================================
Opening audio decoder: [pcm] Uncompressed PCM audio decoder
AUDIO: 32142 Hz, 1 ch, u8, 257.1 kbit/100.00% (ratio: 32142->32142)
Selected audio codec: [pcm] afm: pcm (Uncompressed PCM)
==========================================================================
AO: [pulse] 32142Hz 1ch u8 (1 bytes per sample)
Video: no video
Starting playback...
A:   7.3 (07.2) of 117.0 (01:57.0)  0.0%
```

Modified: http://superkuh.com/rtl2832-cfile-lindi-fm_edit.grc , had a frontend GUI and an increased sample rate. Right now the rate of the audio files saved out is... not very useful. But it sounds fine. Seen below.

3.2 MS/s field of view, tune +-900Khz

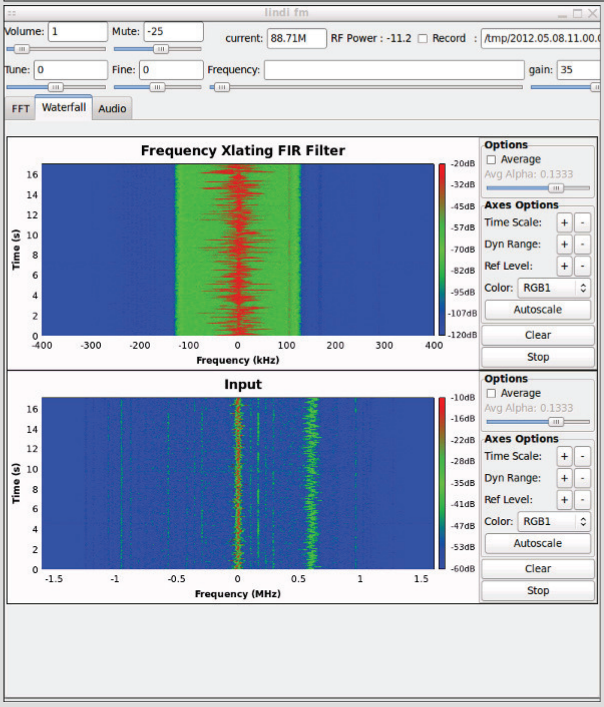




lindi fm

Volume: 1 Mute: -25 current: 89.3184M RF Power : -21.3 ☐ Record : /tmp/2012.05.08.11.00.03.w

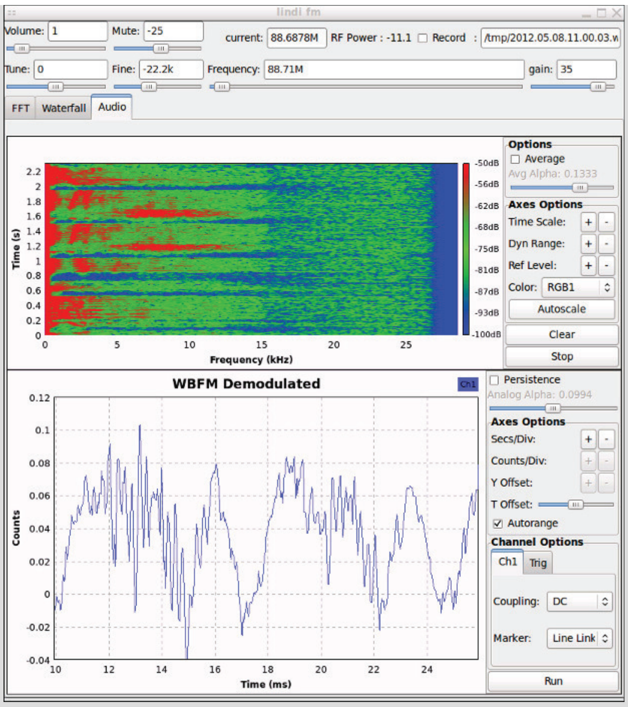
Tune: 608.4k Fine: 0 Frequency: 88.71M gain: 35



lindi fm

Volume: 1 Mute: -25 current: 88.71M RF Power : -11.2 ☐ Record : /tmp/2012.05.08.11.00.0

Tune: 0 Fine: 0 Frequency: gain: 35



Volume: 1 Mute: -25 current: 88.6878M RF Power : -11.1 Record : /tmp/2012.05.08.11.00.03.w

Tune: 0 Fine: -22.2k Frequency: 88.71M gain: 35

Average: off, Avg Alpha: 0.1333, Color: RGB1

Persistence: off, Analog Alpha: 0.0994, Autorange: on, Coupling: DC, Marker: Line Link

---

2h20's simple fm receiver

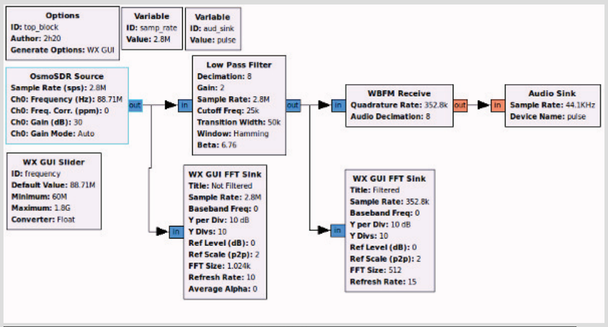
2.8 MS/s field of view, no fine tuning.

2h20 made available, with thorough explaination, a bare bones FM (mono) receiver to learn how to use GNU Radio. This was the first one I managed to get to work. Because the original h202's uses the RTL2832 Source and not the OsmoSDR Source you might experience tuner crashes if you scan too quickly. Make sure to un/replug in the dongle after these. It's best just to enter the frequency as a number.

[Be aware this section of this page was written many months ago when rtl-sdr was different and I had little idea of what I was doing. xzero has since manually added signal seeking to this example.]

My edit of 2h20's simple receiver does not add much, but I did replace the RTL2832 source with an OsmoSDR source to avoid tuner crashes. I also increased the sample rate to 2.8MS/s (to see more spectrum) and then increased the decimation in the filter from 4 to 8 to compensate so everything still decodes/sounds right. I also remove the superfluous throttle block.

- Modified 2h20's Mono FM Receiver (.grc)



Options
ID: top_block
Author: 2h20
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 2.8M

Variable
ID: aud_sink
Value: pulse

OsmoSDR Source
Sample Rate (sps): 2.8M
Ch0: Frequency (Hz): 88.71M
Ch0: Freq. Corr. (ppm): 0
Ch0: Gain (dB): 30
Ch0: Gain Mode: Auto

Low Pass Filter
Decimation: 8
Gain: 2
Sample Rate: 2.8M
Cutoff Freq: 25k
Transition Width: 50k
Window: Hamming
Beta: 6.76

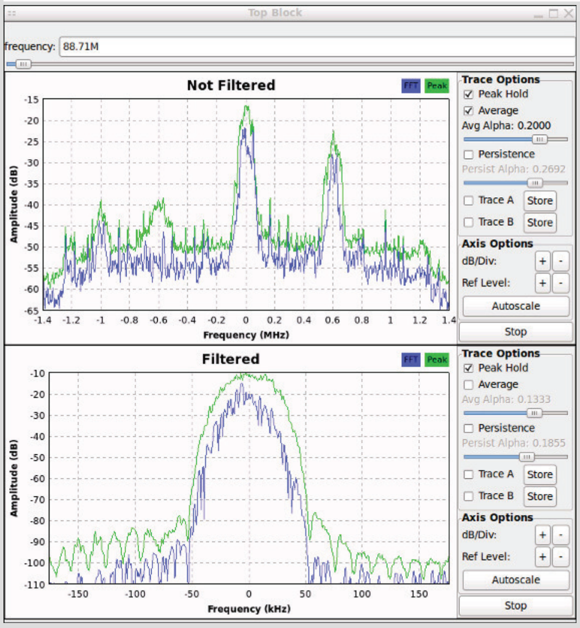
WBFM Receive
Quadrature Rate: 352.8k
Audio Decimation: 8

Audio Sink
Sample Rate: 44.1KHz
Device Name: pulse

WX GUI Slider
ID: frequency
Default Value: 88.71M
Minimum: 60M
Maximum: 1.8G
Converter: Float

WX GUI FFT Sink
Title: Not Filtered
Sample Rate: 2.8M
Baseband Freq: 0
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 10
Average Alpha: 0

WX GUI FFT Sink
Title: Filtered
Sample Rate: 352.8k
Baseband Freq: 0
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 512
Refresh Rate: 15



frequency: 88.71M

Not Filtered: Peak Hold: on, Average: on, Avg Alpha: 0.2000, Persistence: off, Persist Alpha: 0.2692

Filtered: Peak Hold: on, Average: off, Avg Alpha: 0.1333, Persistence: off, Persist Alpha: 0.1855

---

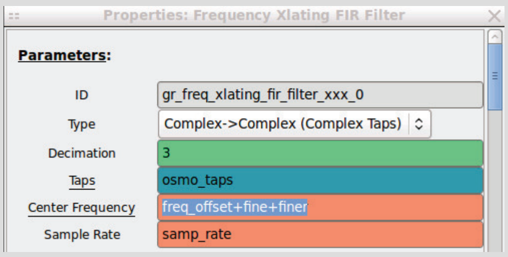
my offset tuning + recording example

2.8 MS/s field of view, +-900Khz tuning, +-50Khz fine.

This takes parts from a bunch of the other example receivers and repurposes them in presumably incorrect but seemingly working ways. It is a basic example of how to offset the tuner 200khz away from the center to avoid the noise there. I started with 2h20's simple tuner's GUI framework and removed almost all of the content. I copied, with inaccurate trial and error changes of sample rate and filter offset, sections of the offset tuning and other advanced bits from simple_fm_rcv and wfm_rx.grc. The tuner is tuned +200Khz. The freq_xlating filter is tuned +200Khz. The the bandpass filter is specified in a variable,

```
firdes.complex_band_pass(1.0,1.024e6,-95e3,95e3,45e3,firdes.WIN_HAMMING,6.76)
```

The net result is that the frequency of interest comes out of the tuner 200Khz below DC, and the freq_xlater "lifts it up" by 200Khz, and then it's bandpassed.


Properties: Frequency Xlating FIR Filter

Parameters:

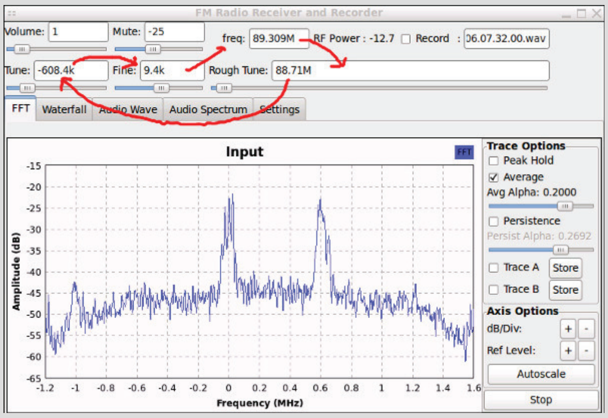
ID	gr_freq_xlating_fir_filter_xxx_0
Type	Complex->Complex (Complex Taps)
Decimation	3
Taps	osmo_taps
Center Frequency	freq_offset+fine+finer
Sample Rate	samp_rate

I also blindly copied the RF power display, a toggle for saving the audio files out to disk, and a +-900khz tuning slider from other receivers. I added a second 'fine' tune +-50Khz. This is done by setting the frequency of the Xlating FIR filter to,

```
freq_offset+fine+finer
```

where freq_offset is the frequency offset from center (200Khz in this case), fine is the ID of a wx gui slider for regular tuning, and finer is the same for fine tuning. In order for the frequency display to show the proper value it was correspondingly set to a variable ID cur_freq,

```
frequency-fine-finer
```


FM Radio Receiver and Recorder

Volume: 1 Mute: -25 freq: 89.309M RF Power : -12.7 Record : 06.07.32.00.wav

Tune: -608.4k Fine: 9.4k Rough Tune: 88.71M

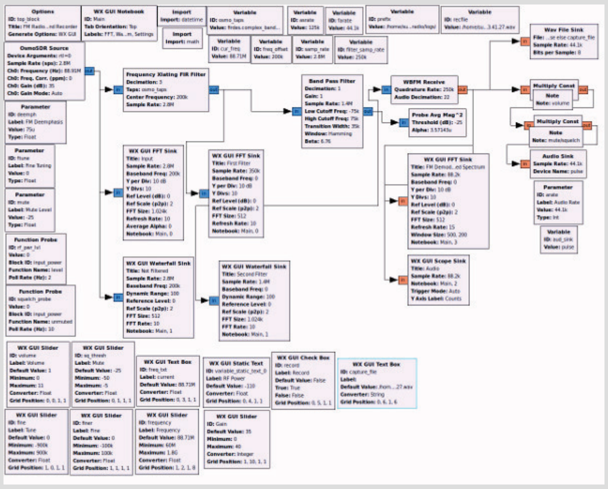
I also made the current frequency display a editable text field so you can tune, copy, and paste. There are good examples of how notebook positioning works and includes simple scripting examples for the file field. This flowchart is simple enough to learn from but includes many elements pulled out of the very complex simple_fm_rcv from patchvonbraun. Without his explanation of the offset process I wouldn't have figured it out. All blocks are layed out by type and GUI elements in the same order as they appear when run. This should help you figure out Grid and Notebook positioning.

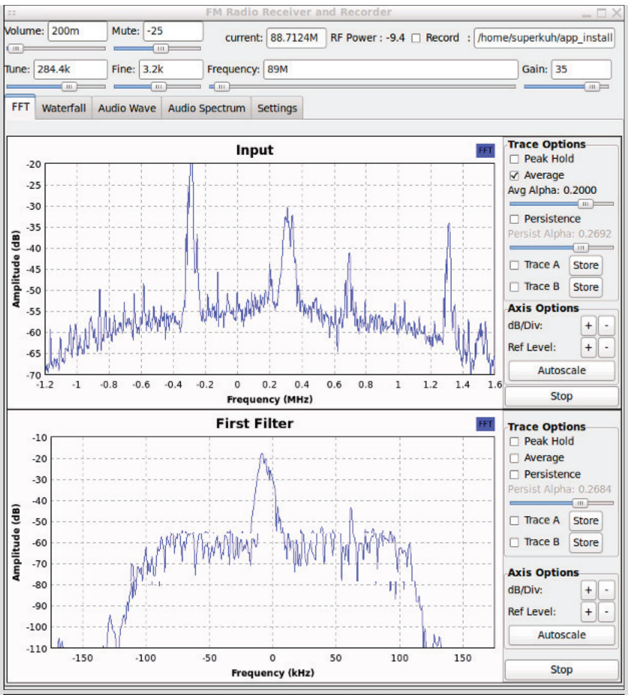
The sound is only "okay". I think the signal is being clipped off at the edges a little bit. I am not sure if it is required to install patchvonbraun's simple_fm_rcv to use this, I do use some of his custom filter stuff.

Usage

Use the Waterfall for scanning through channels. Once located, look at the offset from 0 on the bottom display. Use that to set the tuning (and fine) slider and wiggle it till you get the signal crossing the band in the "Second Filter" top display. Switch to FFT view and look at the bottom "First Filter" display, use tuning and fine tuning to center the peak on the "First Filter" display. Or the other way around. It's personal preference. Ignore the noise you see at higher frequencies (900Mhz) at +0.2Mhz baseband. Although sometimes it gets folded in depending on tuning.

- superkuh's FM w/offset tuning, fine tuning, and recording (.grc)



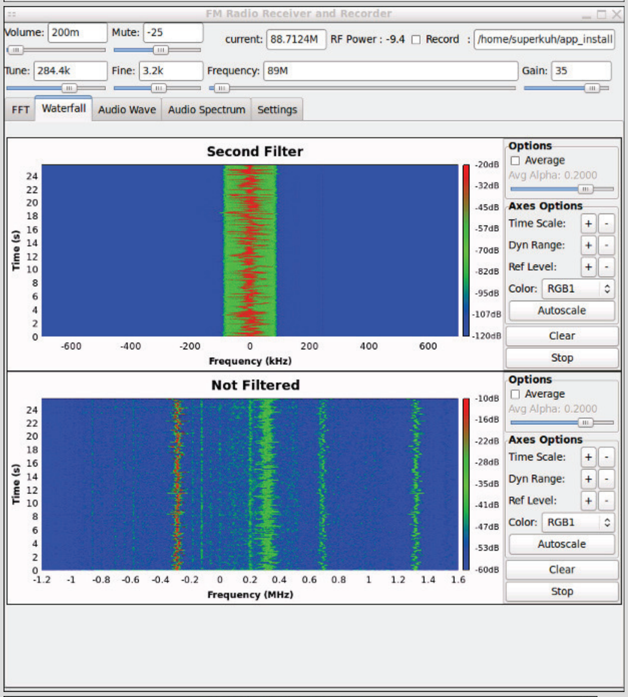


Volume: 200m Mute: -25 current: 88.7124M RF Power : -9.4 Record : /home/superkuh/app_install

Tune: 284.4k Fine: 3.2k Frequency: 89M Gain: 35

Input — Average ✓ Avg Alpha: 0.2000 Persist Alpha: 0.2692

First Filter — Persist Alpha: 0.2684

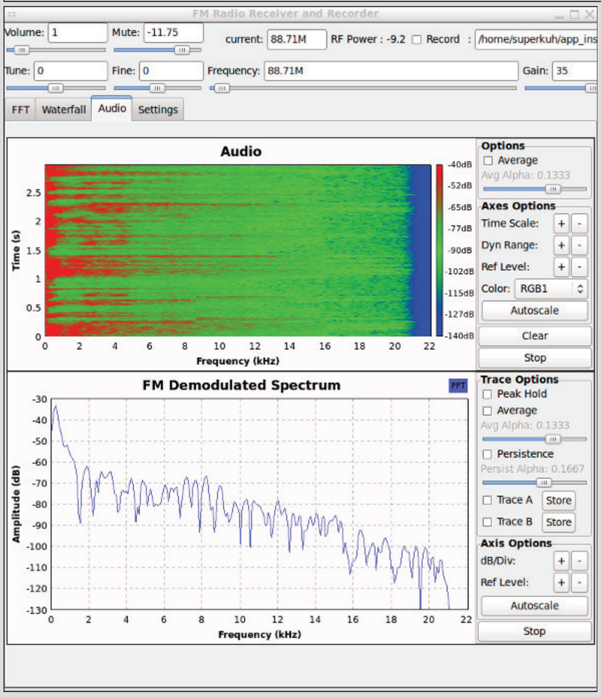


Volume: 200m Mute: -25 current: 88.7124M RF Power : -9.4 Record : /home/superkuh/app_install

Tune: 284.4k Fine: 3.2k Frequency: 89M Gain: 35

Second Filter — Avg Alpha: 0.2000 Color: RGB1

Not Filtered — Avg Alpha: 0.2000 Color: RGB1



Volume: 1 Mute: -11.75 current: 88.71M RF Power : -9.2 Record : /home/superkuh/app_ins

Tune: 0 Fine: 0 Frequency: 88.71M Gain: 35

Audio — Avg Alpha: 0.1333 Color: RGB1

FM Demodulated Spectrum — Avg Alpha: 0.1333 Persist Alpha: 0.1667

SSB Receiver and data Recorder

Created by Alexandru Csete OZ9AEC the notes say, "Simple SSB receiver prototype". This comes from the GNU Radio GRC examples repository over at https://github.com/csete/gnuradio-grc-examples/tree/master/receiver

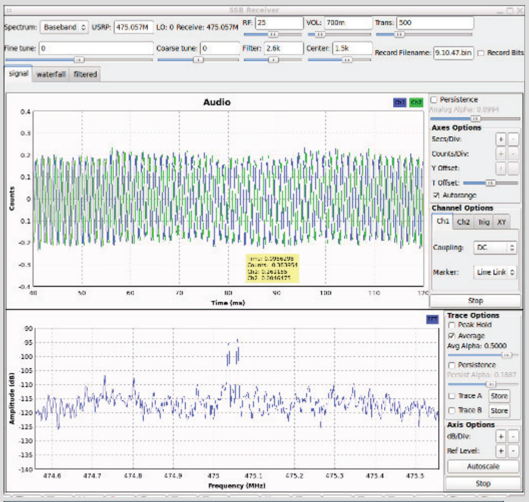
```
git clone https://github.com/csete/gnuradio-grc-examples.git
```

I changed the way it saves samples for the sister decoder program by adding automatic generation of file names and an on/off tickbox toggle for recording. You might want to change the default directory by editing the variable "prefix". The key was

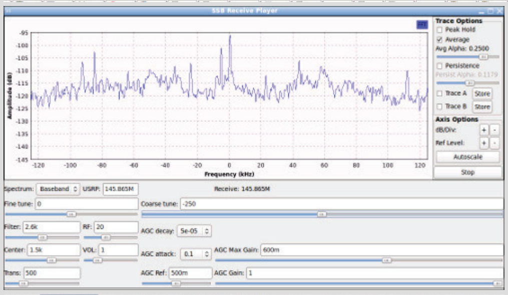
```
"/dev/null" if record == False else capture_file
```

in the File Sink 'file' field. I also changed the GUI so it was easier to find signals. Use the saved .bin files with ssb_rx_play to hear. Jumping around in frequency is a lot smoother when reading from disk instead of the dongle.

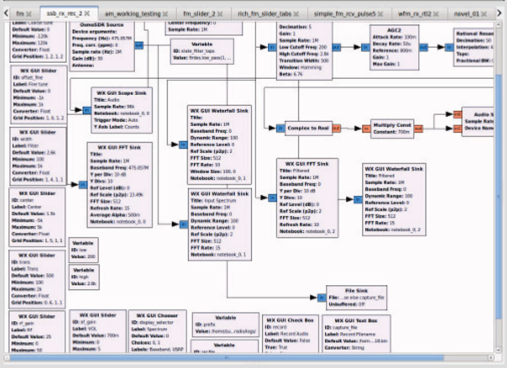
- Modified OZ9AEC SSB Receiver (.grc)
- ssb_rx_play (.grc)



Spectrum: Baseband USRP: 475.057M LO: 0 Receive: 475.057M RF: 25 VOL: 700m Trans: 500
Fine tune: 0 Coarse tune: 0 Filter: 2.6k Center: 1.5k Record Filename: 9.10.47.bin ☐ Record Bits



Spectrum: Baseband USRP: 145.865M Receive: 145.865M
Fine tune: 0 Coarse tune: -250
Filter: 2.6k RF: 20 AGC decay: 5e-05
Center: 1.5k VOL: 1 AGC attack: 0.1 AGC Max Gain: 600m
Trans: 500 AGC Ref: 500m AGC Gain: 1



```
git clone https://github.com/dbasden/python-librtlsdr.git
cd python-librtlsdr/
./sweep.py test.log

Found Elonics E4000 tuner
Using ezcap USB 2.0 DVB-T/DAB/FM dongle
50.0 MHz 50.5 MHz 51.0 MHz 51.5 MHz 52.0 ...
... 998.0 MHz 998.5 MHz 999.0 MHz 999.5 MHz

ls -lathr
total 1.4G
drwxr-xr-x 3 superkuh superkuh 4.0K 2012-05-04 01:11 ..
-rwxr-xr-x 1 superkuh superkuh  624 2012-05-04 01:11 sweep.py
-rw-r--r-- 1 superkuh superkuh  454 2012-05-04 01:11 README
drwxr-xr-x 8 superkuh superkuh 4.0K 2012-05-04 01:11 .git
-rwxr-xr-x 1 superkuh superkuh  707 2012-05-04 01:11 example.py
drwxr-xr-x 2 superkuh superkuh 4.0K 2012-05-04 01:12 rtlsdr
drwxr-xr-x 4 superkuh superkuh 4.0K 2012-05-04 01:12 .
-rw-r--r-- 1 superkuh superkuh 1.4G 2012-05-04 01:28 test.log
-rw-r--r-- 1 superkuh superkuh 1.4G 2012-05-04 01:30 test2.log
```

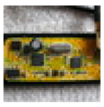
Antenna, USB extenders, etc

The 10m long active USB extension cable I bought seems to work. In fact it decreased the noise floor as compared to directly hooking up to my computer's USB ports.

rtl-sdr on Ubuntu

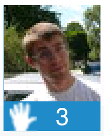
VIEW ALL STEPS ON ONE PAGE

PDF



8 Followers

Follow



3

Author: PRO braingram

It was recently discovered that several cheap DVB-T usb dongles could be configured to be used as cheap ham radio receivers.
Follow along with the discussions here: http://www.reddit.com/r/rtlsdr or find some of the useful software here: http://sdr.osmocom.org/trac/wiki/rtl-sdr

I picked up one of the compatible receivers from ali-express:
http://www.aliexpress.com/snapshot/106450763.html

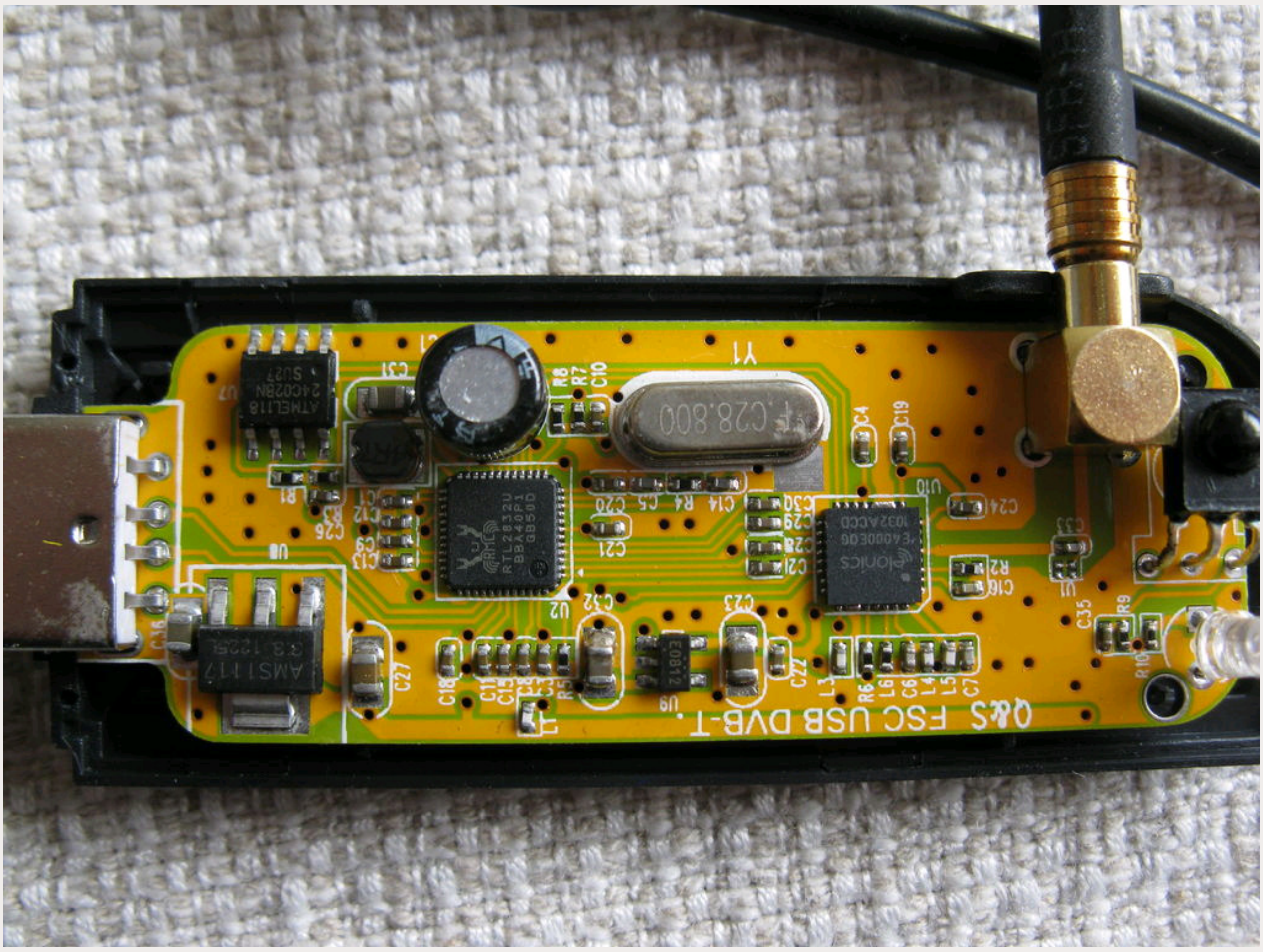
and will document here the steps I took to set it up with gnuradio on ubuntu 12.04.

There are a few things that I won't cover in this instructable, because they deserve an instructable of their own. These include:

1) using the Ubuntu command line (terminal)
2) cloning a git repository
3) building software from source on Ubuntu (especially with cmake)

If anyone has links to stellar, class A++ quality tutorials on any of these things, please add them to the comment and I will link them here.

Step 1 Install gnu-radio



It looks like there are two options for installing an up-to-date gnuradio on ubuntu:

1) use an install script
2) install from source

Don't ask me why, but I tried the second option (probably the harder). After a first attempt that failed, I went for the first (and recommended option) to much success! So... lesson learned. Note to self, follow advice of program developers when installing software.

My advice, use the install script.

Step 2 Install rtl-sdr tools

Next, you want to install any missing rtl-sdr specific tools from here:
http://sdr.osmocom.org/trac/wiki/rtl-sdr

Again, rather than duplicating instructions, follow the steps outlined here to install the rtlsdr library & capture tool: git://git.osmocom.org/rtl-sdr.git

Although this page also mentions a gnu radio module (git://git.osmocom.org/gr-osmosdr) I was unable to get it to work. Instead, I installed:
https://github.com/balint256/gr-baz

as recommended from here: http://2h2o.tumblr.com/

the basic steps are:

```
git clone https://github.com/balint256/gr-baz
```

```
cd gr-baz
sh bootstrap
./configure
make
sudo make install
sudo ldconfig
```

Now you should have all the necessary software to communicate with the DVB-T dongle using gnuradio.

Step 3 Setup udev rules

Next, you need to add some udev rules to make the dongle available for the non-root users. First you want to find the vendor id and product id for your dongle.

The way I did this was to run:

```
lsusb
```

The last line was the Realtek dongle:
Bus 001 Device 008: ID 0bda:2838 Realtek Semiconductor Corp.

The important parts are "0bda" (the vendor id) and "2838" (the product id).

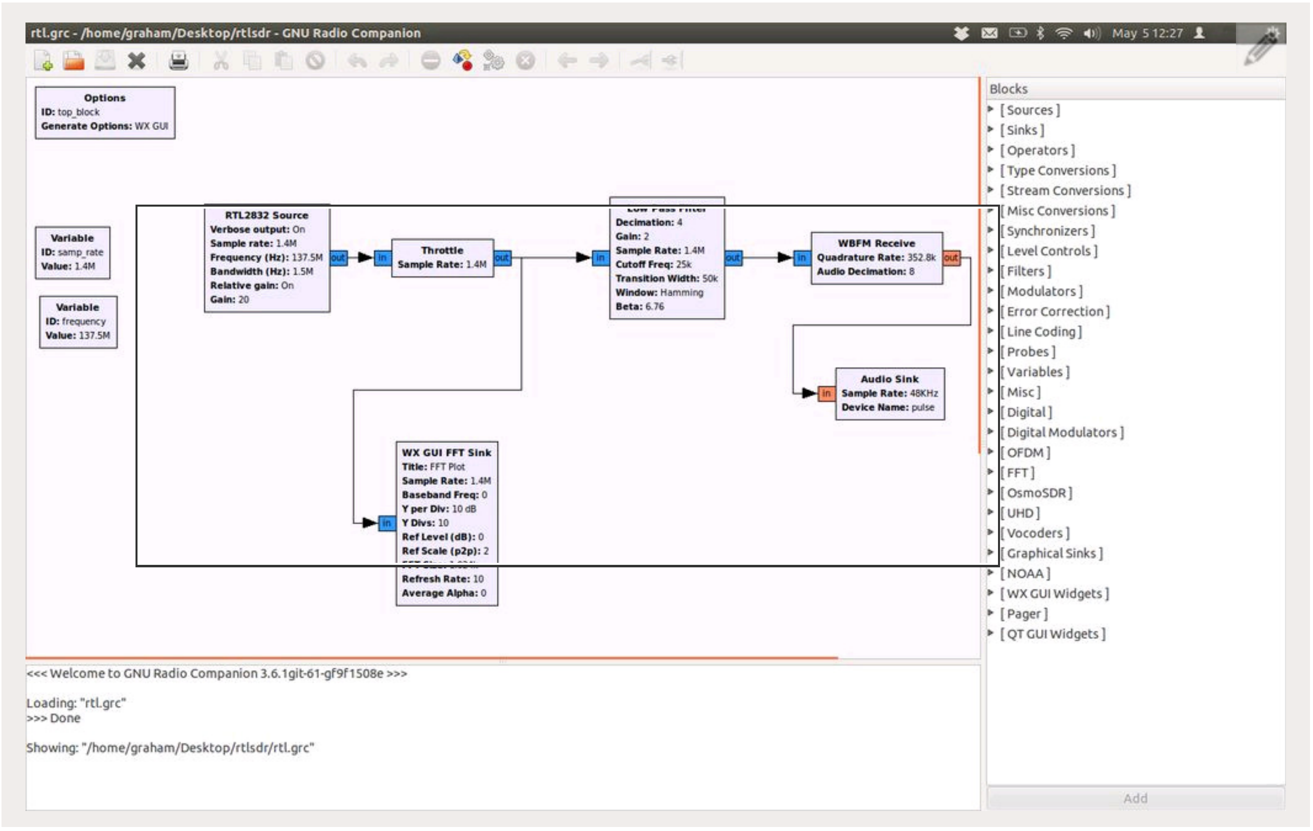
Create a new file as root named /etc/udev/rules.d/20.rtlsdr.rules that contains the following line:

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

With the vendor and product ids for your particular dongle. This should make the dongle accessible to any user in the adm group. and add a /dev/rtl_sdr symlink when the dongle is attached.

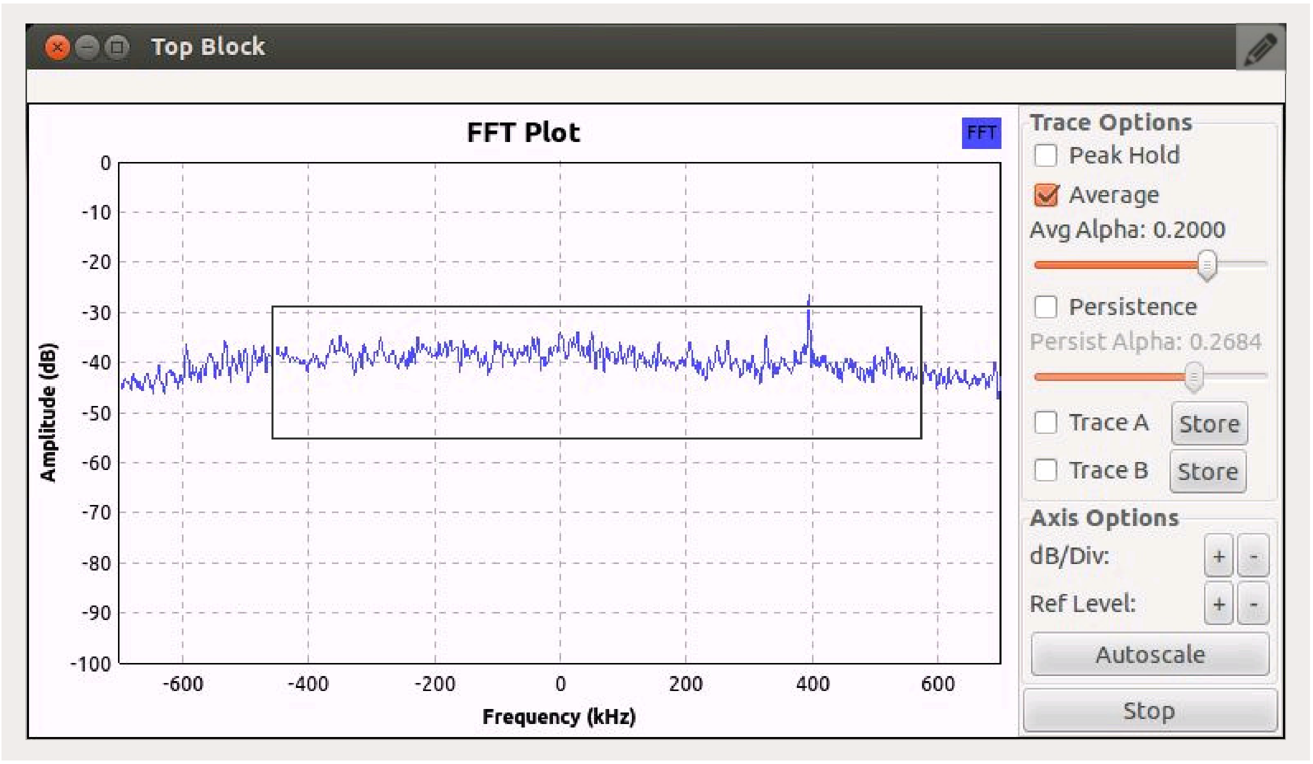
It's probably a good idea to unplug the dongle, restart udev (sudo restart udev) and re-plug in the dongle at this point.

Step 4 Test it out



```
Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 1.4M

Variable
ID: frequency
Value: 137.5M

RTL2832 Source
Verbose output: On
Sample rate: 1.4M
Frequency (Hz): 137.5M
Bandwidth (Hz): 1.5M
Relative gain: On
Gain: 20

Throttle
Sample Rate: 1.4M

Low Pass Filter
Decimation: 4
Gain: 2
Sample Rate: 1.4M
Cutoff Freq: 25k
Transition Width: 50k
Window: Hamming
Beta: 6.76

WBFM Receive
Quadrature Rate: 352.8k
Audio Decimation: 8

Audio Sink
Sample Rate: 48KHz
Device Name: pulse

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 1.4M
Baseband Freq: 0
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
Refresh Rate: 10
Average Alpha: 0
```

```
<<< Welcome to GNU Radio Companion 3.6.1git-61-gf9f1508e >>>

Loading: "rtl.grc"
>>> Done

Showing: "/home/graham/Desktop/rtlsdr/rtl.grc"
```



Now, plug in the dongle and run the following command to test out rtl-sdr:

rtl_sdr capture.bin -s 1.8e6 -f 392e6

Ctrl-C the program after a second or so. If you saw no errors, you should see a file named capture.bin in your current directory. If the program complains about not being able to open the device, try sudo-ing the command. If that helps, the udev rules are probably incorrect.

If all is still going well, try out the attached rtl.grc gnu-radio graph by downloading it and opening it with:

gnuradio-companion ~/Downloads/rtl.grc

It is a modification of the graph from http://2h2o.tumblr.com/ to use a pulse audio (Ubuntu default) rather than an ALSA audio sink.

If you've reached this point with no errors, you are now as far as I am :)

Enjoy! and please add any resources you have on using gnuradio to the comments.

rtl.grc 9 KB

6 comments **Add Comment**

Machine says: Aug 15, 2012. 7:57 AM

I'm new to SDR radio (Software Defined Radio) and have no idea what "RTL" is and have no idea what "gnuradio" is too.

OK, I went digging and found this:
http://en.wikipedia.org/wiki/GNU_Radio

Is RTL a reference to the Realtex USB DVB-T receiver?

Reply

PRO **braingram** (author) says:
Correct!