A Simple SDR for the Raspberry Pi

Rev. 1.0 by Robert Nickels W9RAN   March 31, 2015

The Raspberry Pi (original or model B) lacks the necessary computing power to run Linux-based Software Defined Radio applications, but there is one very basic SDR program that will run on any Pi – it’s called rtl-fm. Despite the name, rtl-fm includes demodulators for AM, wide and narrow-band FM, and upper/lower sideband. It is a command line program and has no user interface to speak of, however various projects have added this functionality using various “helper” programs.

Rtl-fm is distributed free by Osmocom as part of a set of basic utilities for the DVB-T dongles that use the RTL-2832 IC. All that is needed is a dongle and R-Pi with functional sound, keyboard, and network connection. A certain familiarity with Linux and troubleshooting ability is required, as this is intended to provide general guidelines rather than tested step-by-step instructions. Google is the best source of information about Linux commands and any errors that may come up.

The following assumes a Raspberry Pi has been loaded with the Rasbian Operating System. This document is intended to supplement the general rtl-sdr information found at: http://sdr.osmocom.org/trac/wiki/rtl-sdr specifically for Raspberry Pi users.

Update the Raspberry Pi

```bash
sudo apt-get update
sudo apt-get upgrade
```

The above steps will apply all the latest updates that are available.

Prevent the Raspberry from trying to install DVB-T drivers for the dongle
Edit /etc/modprobe.d/raspi-blacklist.conf file and add the following lines

```bash
blacklist dvb_usb_rtl28xxu
blacklist rtl_2832
blacklist rtl_2830
blacklist r820t
```

To edit, use the nano text editor: sudo nano raspi-blacklist.conf Enter or paste all the lines above and press Ctrl-X to exit, and then “Y” to save the file.

Then reboot the Raspberry Pi
**Build the Osmocom software to use the DVB-T dongle as an SDR**

The following steps will install needed dependencies and then build (compile) the software from source code that is obtained from Osmocom’s github server. Because many commands require root privileges, they are run with “sudo” to obtain temporary root access. If a command produces errors, running with “sudo” often fixes the problem.

```
sudo apt-get install git build-essential cmake libusb-1.0-0-dev
sudo git clone git://git.osmocom.org/rtl-sdr.git
cd rtl-sdr
sudo mkdir build
cd build
sudo cmake ../ -DDETACH_KERNEL_DRIVER=ON -DINSTALL_UDEV_RULES=ON
sudo make
```

```
sudo make install
```

```
sudo ldconfig
```

The SDR applications will be installed in the `/usr/local/bin` directory. Go there:

```
- $ cd /usr/local/bin
```

Typing “ls <enter>” will list the contents of the directory. You should see several files with that start with the letters “rtl”. Note that to run these files the next character must be “_” (underscore) even though the dash or hyphen is also used.

**Test the rtl dongle**

Use the rtl_test program to see if the Pi can communicate with the dongle by typing the following command at the console when in `/usr/local/bin` directory:

```
- $ rtl_test
```

You should get a short sign-on message that includes “Lost xx bytes”. If this is all you get, things are working right. This confirms that the Pi is able to communicate with the dongle.

**Configure Pi Audio system**

Audio can be sent to either an HDMI TV/monitor with audio capability or to an amplified speaker connected to the 3.5mm Audio connector on the Pi. Make sure you have one or the other connected, then run `sudo raspi-config`, select Advanced Options, and force the Audio output to the device you have connected. Test your audio and speakers by typing the following command from the console:

```
- $ speaker-test -t sine -f 1000
```
Be sure you can hear a 1000 Hz tone before proceeding. Press Ctrl-C to end the test.

**Connect dongle and antenna**

The short antenna supplied with most dongles is inadequate for anything but receiving very strong local stations, but it can be used for initial testing if no better antenna is available. Connect it and plug the dongle into the Pi USB port, or into a powered USB hub attached to the Pi. Early models of the Pi may not be able to provide enough power to operate the Dongle, so if problems occur, try using an powered USB hub, or a larger power supply. Use rtl_test to verify that you are able to communicate with the dongle without difficulty.

**Listen to FM Radio**

Rtl_fm is a very simple SDR program that will run on the Raspberry Pi. It doesn’t even have a tuning dial! You must specify the frequency you want by typing it into the command line, so this is a very inefficient way to listen to the radio, but it does work and it proves that the Raspberry Pi CAN do SDR. Please read the rtl_fm user guide that is found online: [http://kmkeen.com/rtl-demod-guide/](http://kmkeen.com/rtl-demod-guide/)

The best way to see if rtl-fm will work is to try tuning to a local FM broadcast station. You will need to substitute the frequency where I have entered “89.1” below, as this is the frequency of MY local NPR station. Everything else should be typed exactly as shown into the console: (including the underscore and upper case where shown)

```
$ rtl_fm -M wbfm -F 89.1M | aplay –r 32k –t raw –c 1 -f S16_LE <enter>
```

M in the frequency entry is shorthand for Mhz.

You should be rewarded with hearing the FM broadcast audio from your Pi speakers!

(Note the | character above, which is used to “pipe” the output from rtl_fm into the aplay program. If your keyboard doesn’t have this vertical line character, you can create it by holding down the ALT key and typing 0124 on the number pad.)

The example above uses a Linux audio application called “aplay” which is part of the Alsa sound system. Google for full information on aplay options, as there are many, but do not change them unless you know why you are doing so. Alsa includes a simple volume control application that can change the audio level. To run it type:

```
-$ alsamixer
```

A simple graphical screen will appear and the level of the slider can be adjusted up and down with the up/down arrows. Press ESC to exit Alsamixer.
Have fun!

You can try to hear other radio signals by pressing Ctrl C (hold Ctrl while hitting C) to stop the program, and then editing the command line. This is easiest if you hit F1 and then use the left arrow to move to the point desired. Try editing the command line to tune to the frequencies of other FM stations.

A good next step is to try listening to narrow band FM. The best station to start with is your local National Weather Service transmitter – mine is on 162.45 Mhz so you should change this to the frequency of the station nearest you. To set rtl_fm to narrowband FM on the weather channel, type the following into the console:

```bash
$ rtl_fm -M fm -f 162.45M -s 20K | aplay -r 20K -t raw -c1 -f S16_LE <enter>
```

Editing the frequency above should allow you to hear ham FM stations or repeaters, or other FM transmissions. You may be able to reduce the “20k” to a lower value such as “8k” for better audio, depending on the mode used. However since the audio is being piped from rtl_fm to aplay, it is important that the same sample rate be specified on both sides of the “|” character, otherwise the audio will be distorted.

To try other modes, see the user guide website. It is possible to change the RF gain, adjust for frequency error, engage squelch, and even scan a list or range of frequencies using various command line options.

As simple as it is, rtl_fm illustrates the principles of SDR operation on a small CPU platform using nothing but a very cheap TV dongle as the RF front-end. With the improved processing power of the new Raspberry Pi Model B more options will no doubt be available in the near future!