




**kbandura** add some t's into fm. Simplify bandwidth

a9ebdf6 · 6 years ago



| Name  | Name                           | Last commit date |
|---|--------------------------------|------------------|
|  ..        |                                |                  |
|  img       | FM                             | 7 years ago      |
|  README.md | add some t's into fm. Simpl... | 6 years ago      |

## README.md



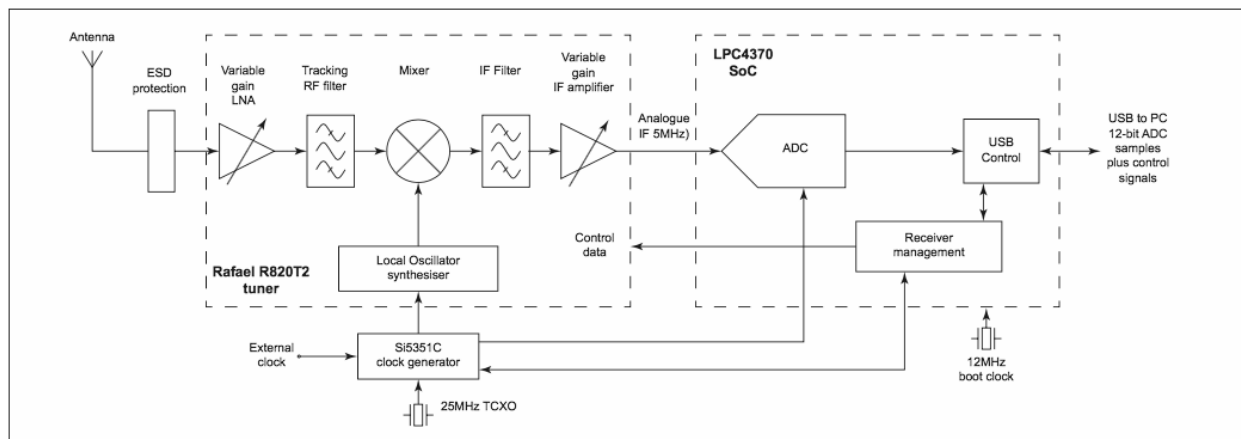
## 2. Software Defined Radio - Hardware

In [Lab 1](#) we learned how to work around GNU Radio Companion (GRC) and simulate signals and instruments. We shall now step into the real world with real signals.

- [2. Software Defined Radio - Hardware](#)
  - [2.1. Introduction](#)
  - [2.2 Frequency Correction of the SDR Dongle](#)
  - [2.3. GNURadio FM](#)
    - [2.3.1 Signal Modulation](#)
      - [2.3.1.1 Amplitude Modulation](#)
      - [2.3.1.2 Frequency Modulation](#)
    - [2.3.2 Let's Make our FM Radio](#)
  - [2.4. Fun SDR/GNU Radio things](#)

### 2.1. Introduction

A "simplified" diagram of the device which we shall be using is shown below:



Simplifying this further for a general SDR hardware including the energy conversions:

(EM Waves))))) >-(Antenna)-->(Amplifier)--->(Local Oscillators + Filters)-->(Analog to Digital Converter)-->(networking control: usually USB)--->[Computer]

Radio Waves excite electrons in the antenna and induces a current. The frequencies the antenna is most sensitive to is determined by the geometry of the antenna's design. The electric current is then initially amplified a bit. This amplifier is generally a "Low Noise Amplifier" because we want as little as possible in the antenna signal from the local electronics. Processing a signal at a fixed frequency gives a radio receiver improved performance so thus a local oscillator (LO) is used. It is an electronic oscillator used with a mixer to change the frequency of a signal. This frequency conversion process, also called heterodyning, produces the sum and difference frequencies from the frequency of the local oscillator and frequency of the input signal. The desired frequency is then filtered out and if required amplified again. The last step is the most crucial step where-in the signal is digitized to be sent to the computer to be manipulated by our gnuradio code!

[↑ Go to the Top of the Page](#)

## 2.2 Frequency Correction of the SDR Dongle

The hardware is well made, but a precision clock is quite expensive. The frequency the "tuner" tunes to may be slightly off from the actual frequency it is tuning to. We can correct for that in the software. For high-end SDR dongles this correction is virtually non-existent but some low-end dongles have higher deviations!

We can transmit a signal using a known and reliable tone. Then we use our receiver set up with `gqrx` to see the signal. If the incoming signal is exactly at the expected frequency then the internal clock is working well. If not we look at the `input controls` tab in `gqrx` and change the `freq. correction` value until the peak is at the correct output. This value will be different for all dongles. It also changes with the temperature of the dongle. It is interesting to watch this change as the dongle warms up. Note your value for future purposes.

[↑ Go to the Top of the Page](#)

## 2.3. GNURadio FM

---

We used gqrx in section 1.2 to listen to FM now we shall code our own radio using GRC!

### 2.3.1 Signal Modulation

Modulation is a process of mixing a signal with a sinusoid to produce a new signal. Consider a signal represented by the function:

$$f(t) = A\sin(\omega t + \phi).$$

This sinusoid has 3 variables that can be altered to change the function  $f(t)$ . The first term,  $A$ , is called the magnitude, or amplitude of the sinusoid. The next term,  $\omega$  is known as the frequency, and the last term,  $\phi$  is known as the phase angle. We can encode our message in either of these three parameters.

The sinusoidal signal that is used in the modulation is known as the carrier signal, or simply "the carrier". The signal that is used in modulating the carrier signal (or sinusoidal signal) is known as the "data signal" or the "message signal".

In other words we can say that modulation is used because some data signals are not always suitable for direct transmission, but the modulated signal may be more suitable.

It follows from above we encode in the above three variables. Consequently, we have 3 basic types of analog modulation:

- Amplitude Modulation
- Frequency Modulation
- Phase Modulation

#### 2.3.1.1 Amplitude Modulation

For our discussion of amplitude modulation consider a carrier wave of frequency  $f_c$  and amplitude  $A$  given by:

$$c(t) = A \cdot \sin(2\pi f_c t).$$

Let  $m(t)$  represent the modulation waveform. For this example we shall take the modulation to be simply a sine wave of a frequency  $f_m$ , a much lower frequency (such as an audio frequency) than  $f_c$ :

$$m(t) = M \cdot \cos(2\pi f_m t + \phi),$$

where  $M$  is the amplitude of the modulation. If  $M > 1$  then overmodulation occurs and reconstruction of message signal from the transmitted signal is more difficult.

Amplitude modulation results when the carrier  $c(t)$  is multiplied by the positive quantity  $(1 + m(t))$ :

$$y(t) = [1 + m(t)] \cdot c(t) = [1 + M \cdot \cos(2\pi f_m t + \phi)] \cdot A \cdot \sin(2\pi f_c t)$$

Using trigonometric identities,  $y(t)$  can be shown to be the sum of three sine waves:

$$y(t) = A \cdot \sin(2\pi f_c t) + \frac{AM}{2} [\sin(2\pi(f_c + f_m)t + \phi) + \sin(2\pi(f_c - f_m)t - \phi)]$$

Therefore, the modulated signal has three components: the carrier wave  $c(t)$  which is unchanged, and two pure sine waves (known as sidebands) with frequencies slightly above and below the carrier frequency  $f_c$ .

Demodulation or extracting the message from the carrier involves simply filtering out the carrier signal. We can construct an AM radio receiver on GNU radio however our SDR dongle can only tune from ~20 MHz to ~1800 MHz.

### 2.3.1.2 Frequency Modulation

As the name suggests the message signal is encoded in the frequency variable of the carrier signal as in  $x(t) = a \sin(f(t)t + \phi)$ . If the information to be transmitted (i.e., the data/message signal is  $x_m(t)$  and the sinusoidal carrier is  $x_c(t) = A_c \cos(2\pi f_c t)$ , where  $f_c$  is the carrier's base frequency, and  $A_c$  is the carrier's amplitude, the modulator combines the carrier with the data/message signal to get the transmitted signal

$$\begin{aligned} y(t) &= A_c \cos(2\pi f(t)t) \\ &= A_c \cos(2\pi[f_c + f_\Delta x_m(t)]t) \\ &= A_c \cos(2\pi f_c t + 2\pi f_\Delta x_m(t)t) \end{aligned}$$

where  $f_\Delta$  is the sensitivity of the frequency modulator which adjusts how much bandwidth is used for the signal.

### 2.3.2 Let's Make our FM Radio

One way to demodulate the signal is to extract the message encoded in the frequency of the sinusoid outside the sinusoid. That can be achieved by "fast" differentiating the sine wave, treating the message as a constant. Consider the following:

$$x(t) = a \sin(f(t)t + \phi) \frac{dx(t)}{dt} = a f(t) \cos(f(t)t + \phi) = A(t) \cos(f(t) + \phi)$$

For the FM signal

$$y(t) = A_c \cos(2\pi f_c t + 2\pi f_\Delta x_m(t)) = A_c \cos(\theta(t))$$

$$\begin{aligned} y'(t) &= -A_c \theta'(t) \sin(\theta(t)) \\ &= -2\pi A_c (f_c + f_\Delta x_m(t)) \sin(\theta(t)) \end{aligned}$$

We observe that we converted the FM signal into the form  $y(t) = [1 + m(t)] \cdot c(t)$  which is an AM signal. We can easily demodulate this AM signal by filtering out the AM "carrier". It follows the following flow:

FM ---->|Differentiator|---->|Envelope Detector|----> Signal

A similar operation can be achieved in GNU radio using the following flow:

FM ---> |Filter out the signal of interest| ---> |Resample Signal| ---> |Quadrature Demodulator|--->|Lowpass Filter| ---> Audio Signal

The quadrature demodulator uses a different technique than differentiating the signal since the incoming data is complex, but the end result is the same, where the output is proportional to the change in frequency of the input. (That gnuradio block actually has a good explanation of the math in the description. )

#### Hints:

**Source:** Since we are using a hardware source we have to use the appropriate block. Search for the `osmocom Source` block. The Device arguments should be `airspy=0`.

**NOTE:** The Sample rate supported by this dongle is either 2.5 MHz or 10 MHz. We shall set our `samp_freq` variable to `2500000`. The `ch0: Frequency (Hz)` is the frequency you want to tune to.

The screenshot shows a dialog box titled "Properties: osmocom Source" with three tabs: "General", "Advanced", and "Documentation". The "Advanced" tab is selected. The dialog contains various configuration options for the "osmosdr\_source\_0" device. The options are as follows:

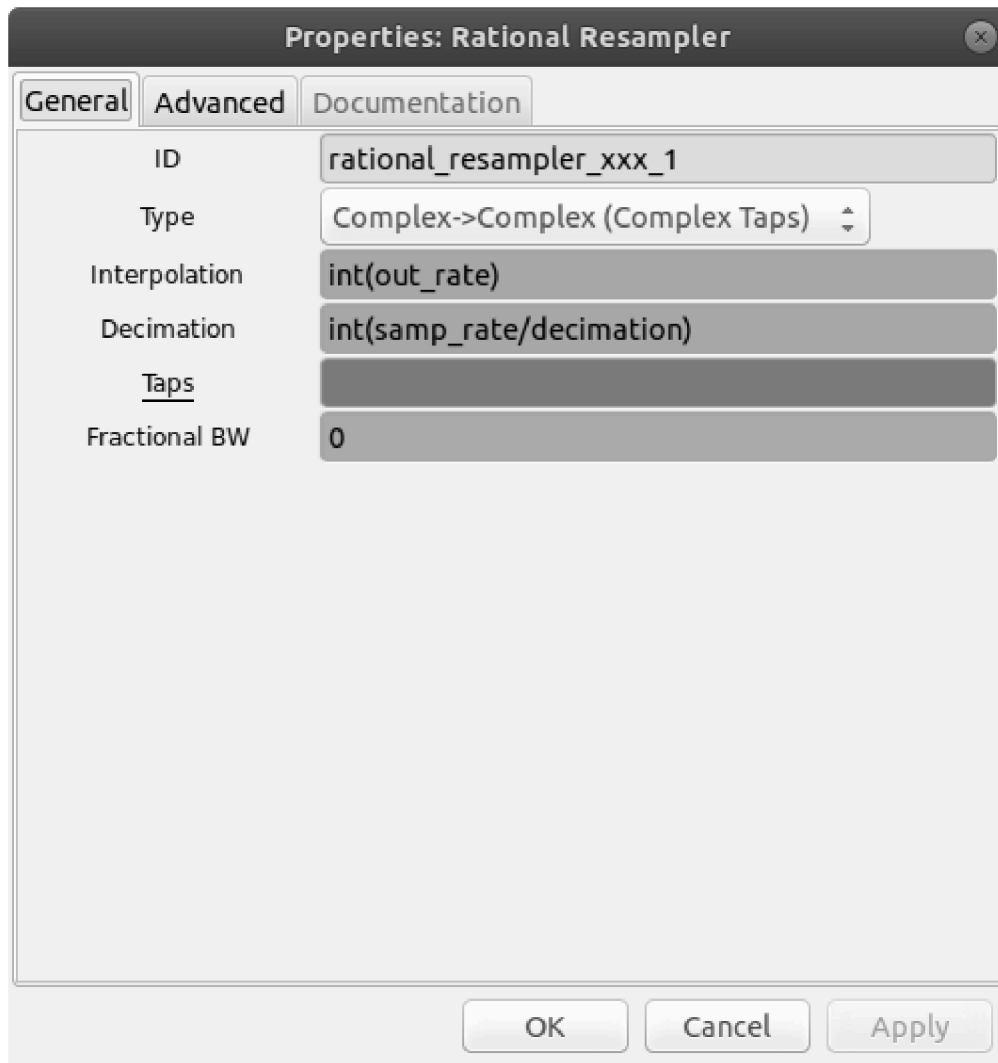
| Property               | Value            |
|------------------------|------------------|
| ID                     | osmosdr_source_0 |
| Output Type            | Complex float32  |
| Device Arguments       | airspy=0         |
| Sync                   | don't sync       |
| Num Mboards            | 1                |
| Mb0: Clock Source      | Default          |
| Mb0: Time Source       | Default          |
| Num Channels           | 1                |
| Sample Rate (sps)      | samp_rate        |
| Ch0: Frequency (Hz)    | center_freq      |
| Ch0: Freq. Corr. (ppm) | 0                |
| Ch0: DC Offset Mode    | Off              |
| Ch0: IQ Balance Mode   | Off              |
| Ch0: Gain Mode         | Manual           |
| Ch0: RF Gain (dB)      | 50               |
| Ch0: IF Gain (dB)      | 20               |
| Ch0: BB Gain (dB)      | 20               |
| Ch0: Antenna           |                  |
| Ch0: Bandwidth (Hz)    | 0                |

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

**Low Pass Filter:** This filters out all the frequencies apart from the one we want to tune our radio to. Note that I have another variable called `channel_width` which is equal to  $200\text{e}3$ . It is to filter out at a data rate 200kHz.

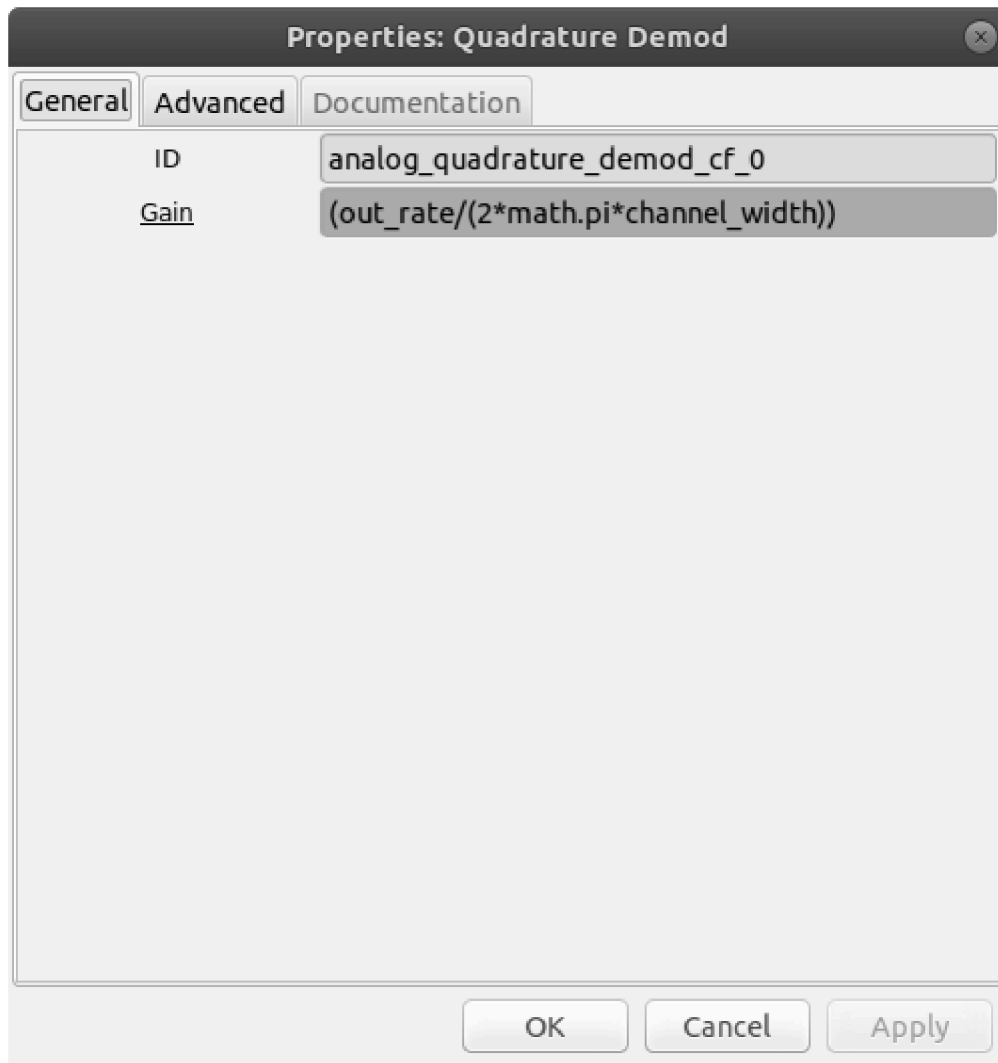


**Resampling Signal:** Use the 'Rational Resampler' block. Resample the signal such that the frequency of the signal is a multiple of output frequency. The output frequency ( 'out\_rate') is the frequency at which the sound card accepts samples i.e. 48 kHz to play audio. The output frequency should still be near to the bandwidth of the message i.e. 200kHz. 192kHz ie 'out\_rate' is the closest multiple of 48kHz to it. As noticed on the screen shot here is a new variable 'decimation == int(samp\_rate/(2\*channel\_width))'



**Quadrature demodulation:** This block extracts the time dependend frequency component of the signal which is the audio signal. Use the quad demod block and fill in ' $(out\_rate/(2 \cdot pichannel\_width))$ ' in the gain field





**Lowpass Filter:** Use a lowpass filter with the cutoff frequency at 18 kHz ( because human audio perfection has an upper limit close to it). Enter the decimation value to downsample the signal coming in from 192kHz to 48kHz the rate at which the sound card works.



### *Play audio from an audio sink*

Lets capture some sweet tunes!

[↑ Go to the Top of the Page](#)

## 2.4. Fun SDR/GNU Radio things

1. AM Radio! (see above)
2. Narrow Band FM ( same are FM but a narrower filter passband)
3. [Listen to and get airplain ADS-B data](#) To chekc it out on your own get this software: [dump1090](#)
4. Listen to HAM radio chatter ( usually amplitude modulated )
5. EMS and police and local services radio. [local scanners and frequencies](#)
6. WeatherFAX. Get latest images of weather data from naval bases! <http://www.rtl-sdr.com/receiving-weather-rtty-rtl-sdr/>

7. Get satellite data (Receive and decode live satellite images of earth): These satellite transmit that these frequencies: NOAA 15 – 137.6200 MHz NOAA 18 – 137.9125 MHz NOAA 19 – 137.1000 MHz

8. If transmitted nearby get a newspaper over the radio!

9. Decode high definition radio

10. Build your own radio astronomy observatory! ( ok we are totally doing that!)

[↑ Go to the Top of the Page](#) .....[Next Lab](#)