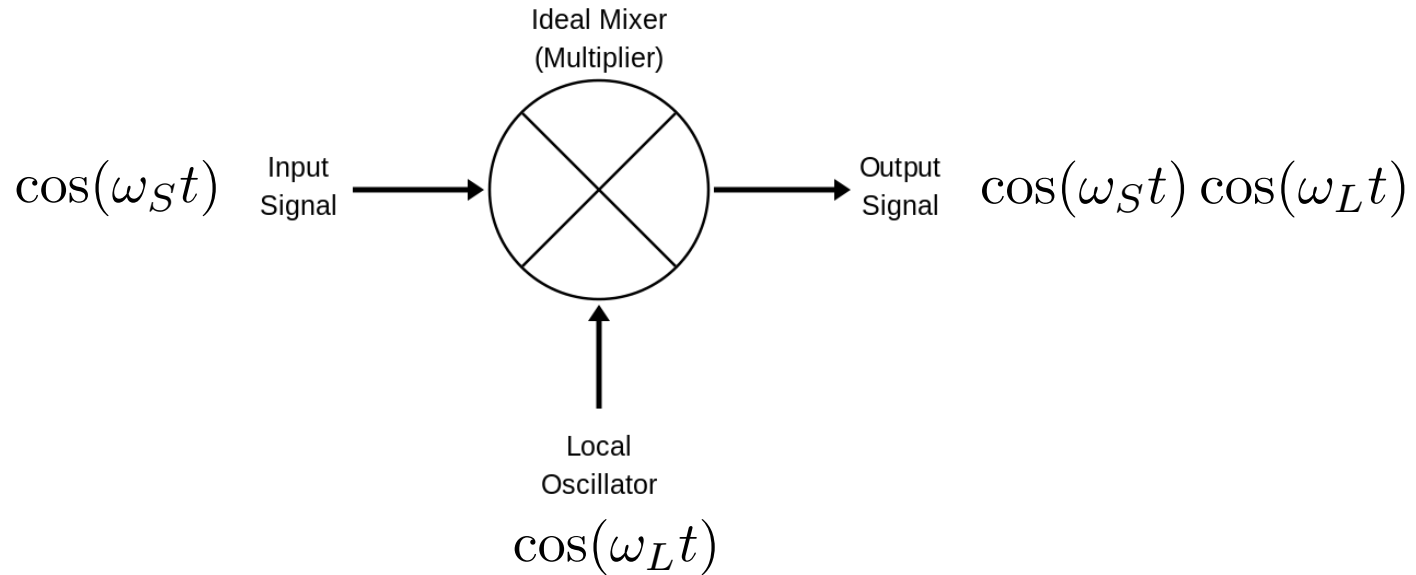


Software Defined Radio (SDR)

Edoardo Milotti

Corso di Metodi di Trattamento del Segnale

Rivelazione eterodina




$$\cos(\omega_S t) \cos(\omega_L t) = \frac{1}{2} \{ \cos[(\omega_S + \omega_L)t] + \underline{\cos[(\omega_S - \omega_L)t]} \}$$

Importanza della fase

Consideriamo un segnale modulato in ampiezze e fase (questo comprende anche la modulazione di frequenza), allora l'eterodina produce il seguente risultato


$$A(t) \cos(\omega_C t) \cos(\omega_C t + \varphi(t)) = \frac{A(t)}{2} [\cos \varphi(t) + \cos(2\omega_C t + \varphi(t))]$$

 I (in-phase)

dove la fase aggiunta corrisponde a frequenze di modulazione molto più piccole della frequenza della portante. Come si vede, se la fase aggiunta è vicina 90° , il segnale a bassa frequenza ha ampiezza molto piccola e viene quasi completamente perso.

Però se si moltiplica anche per un seno ...

$$A(t) \cos(\omega_C t) \sin(\omega_C t + \varphi(t)) = \frac{A(t)}{2} [\sin \varphi(t) + \sin(2\omega_C t + \varphi(t))]$$

 Q (quadrature)

e si vede che il segnale a bassa frequenza in questo caso NON è piccolo per una fase vicina a 90° .

A questo punto la modulazione di ampiezza si può recuperare in questo modo

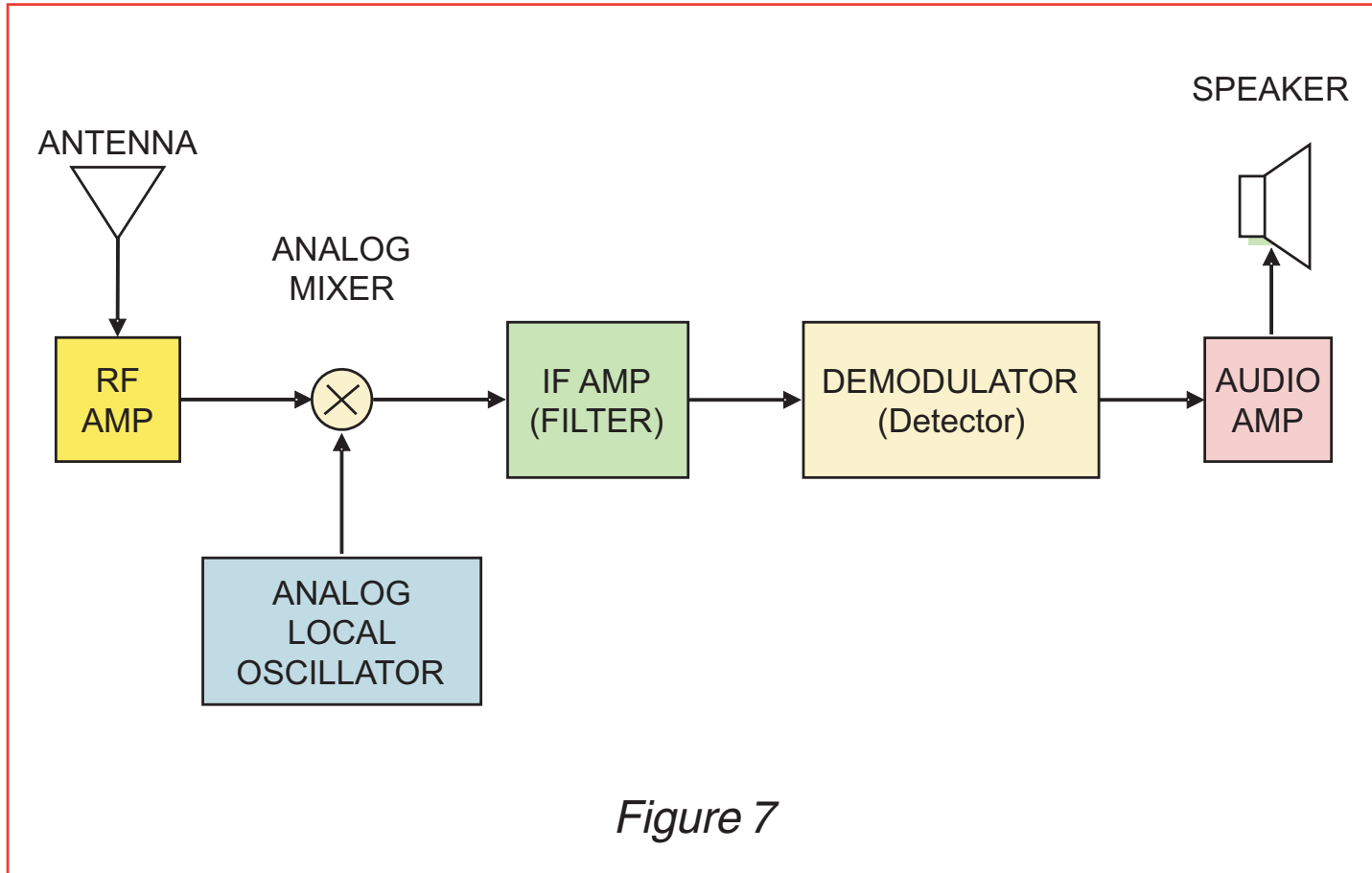
$$x = \frac{A(t)}{2} \cos \varphi$$
$$y = -\frac{A(t)}{2} \sin \varphi$$

$$\Rightarrow A(t) = \sqrt{x^2 + y^2}$$

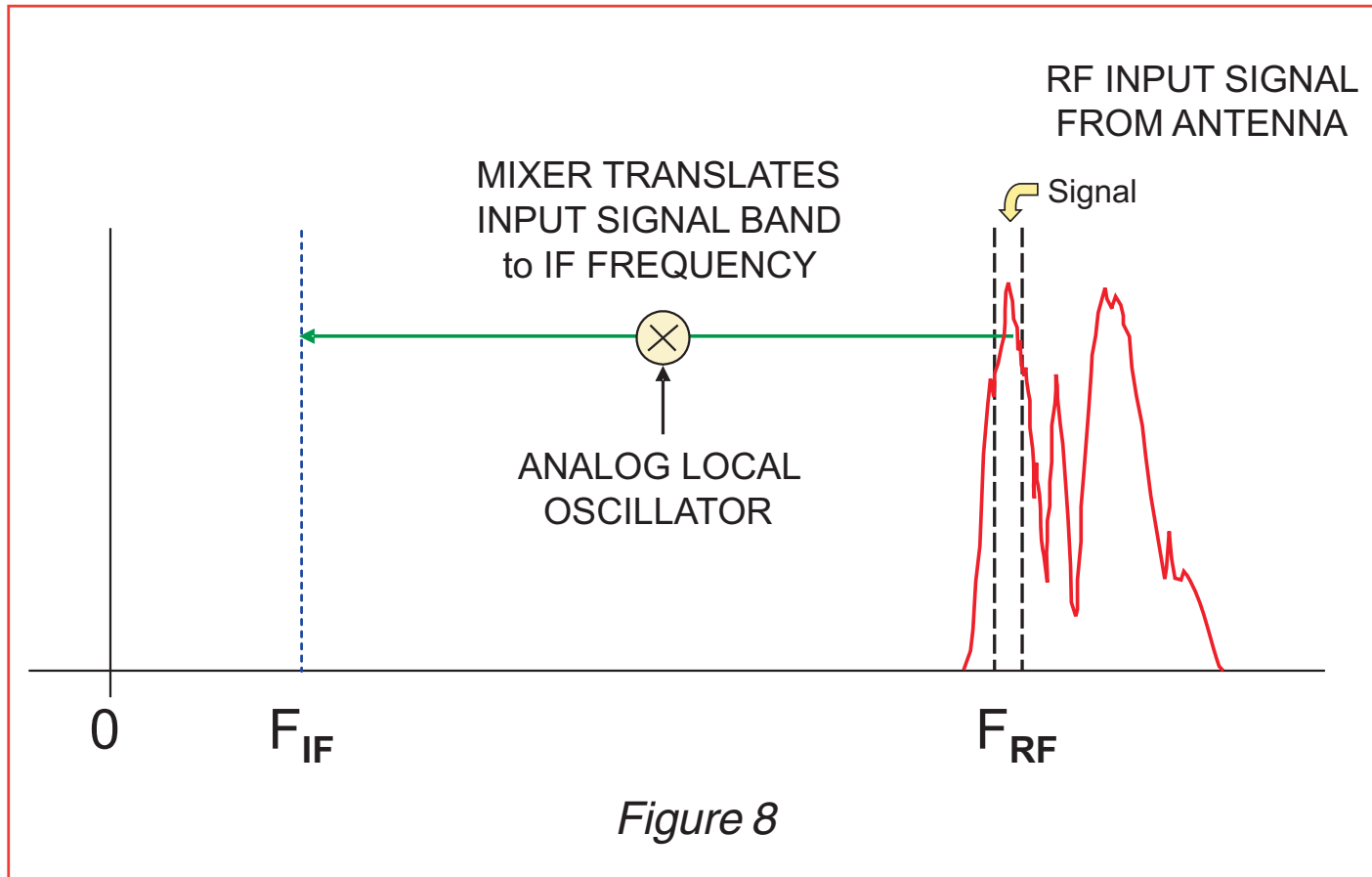
e la modulazione di fase o frequenza

$$\varphi(t) = -\arctan \frac{y}{x}$$

Analog Radio Receiver Block Diagram



Analog Radio Receiver Mixer



SDR Receiver Block Diagram

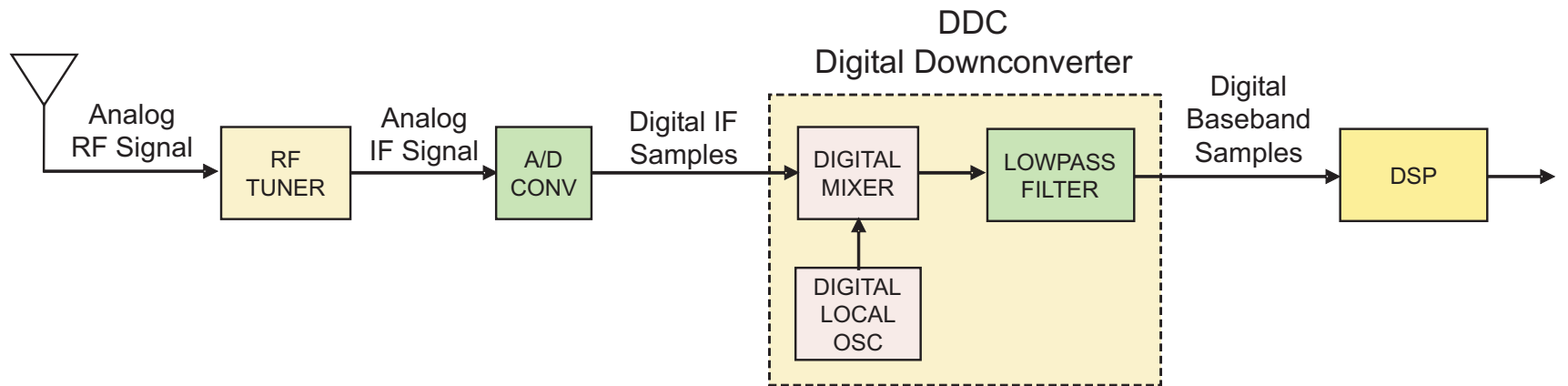


Figure 9

SDR Receiver Mixer

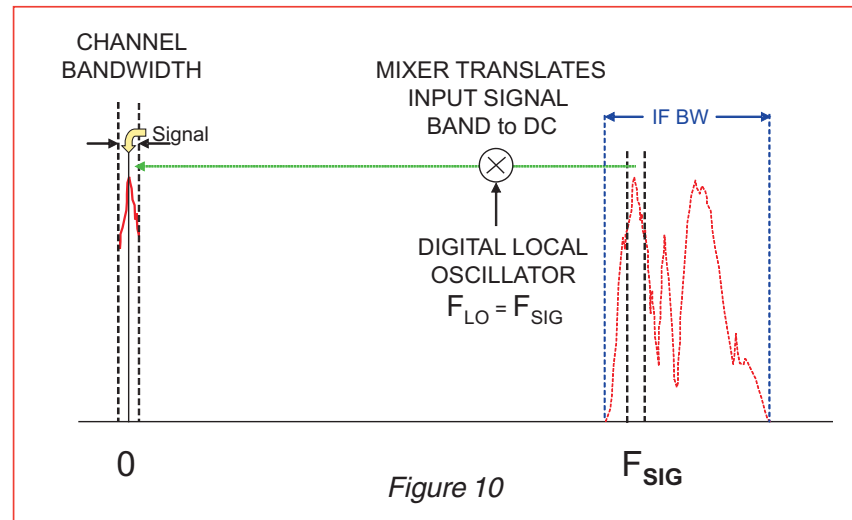


Figure 10

DDC Local Oscillator and Decimation

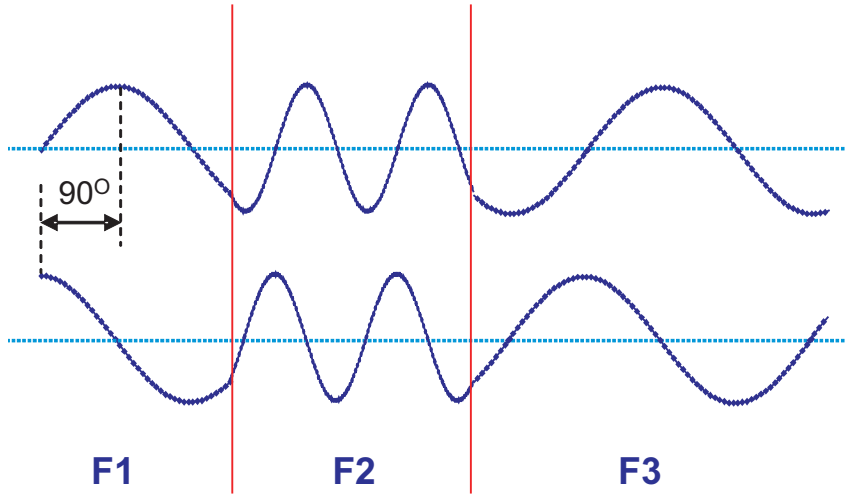


Figure 11A Local Oscillator Frequency Switching

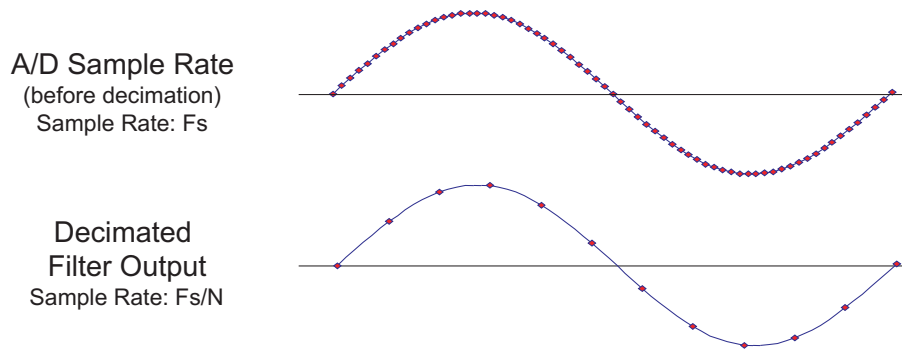


Figure 11B FIR Filter Decimation

DDC Signal Processing

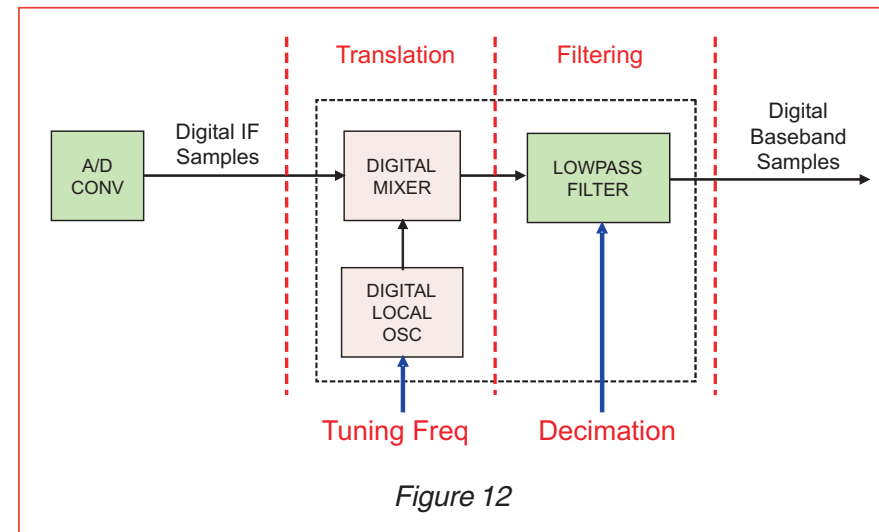
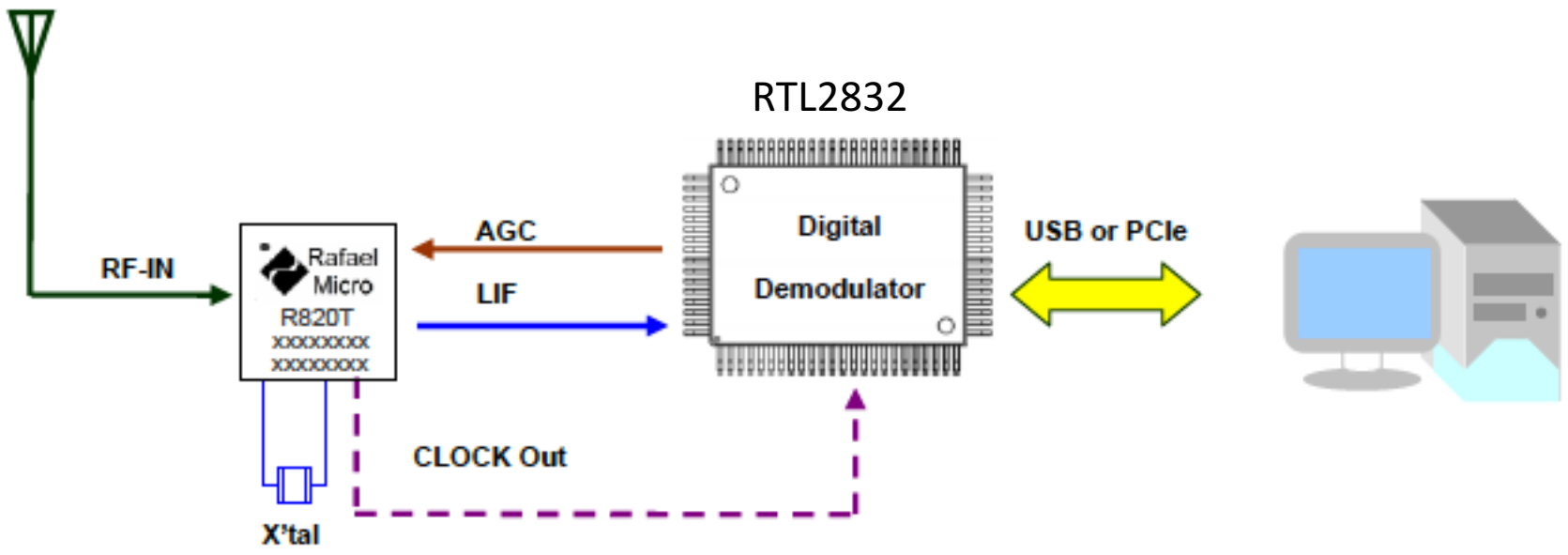
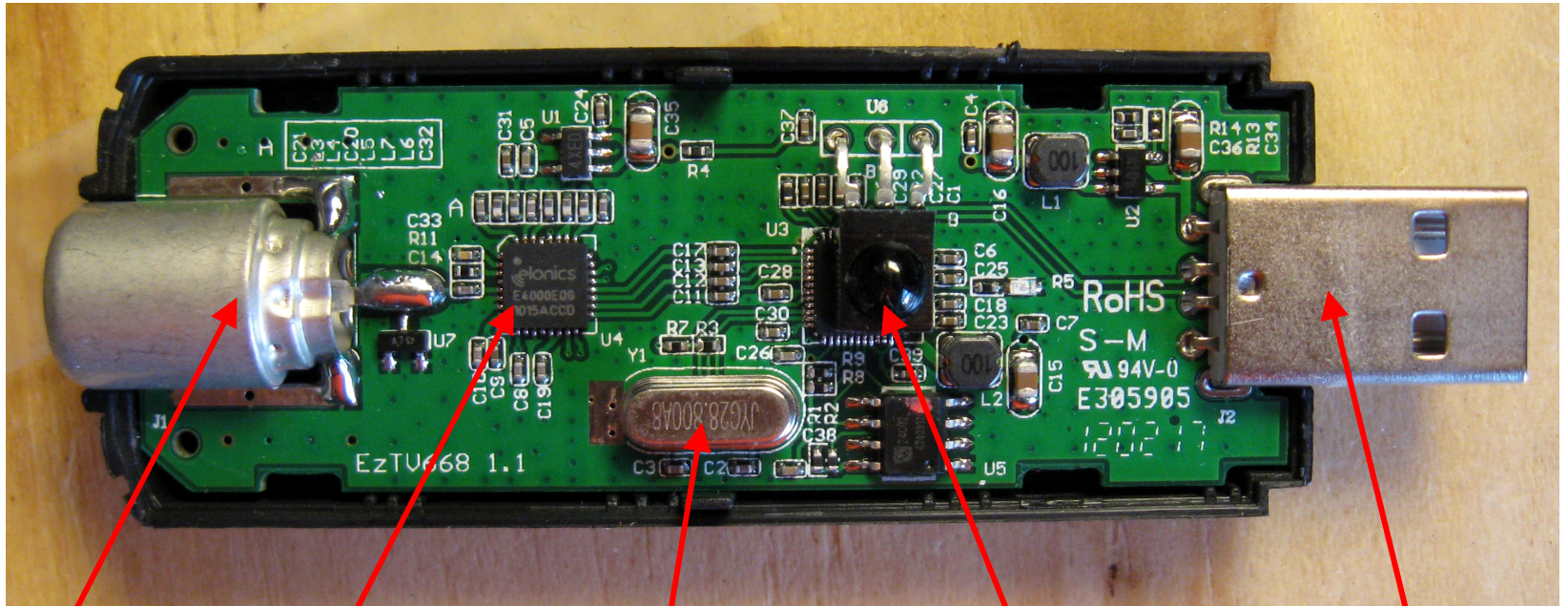


Figure 12





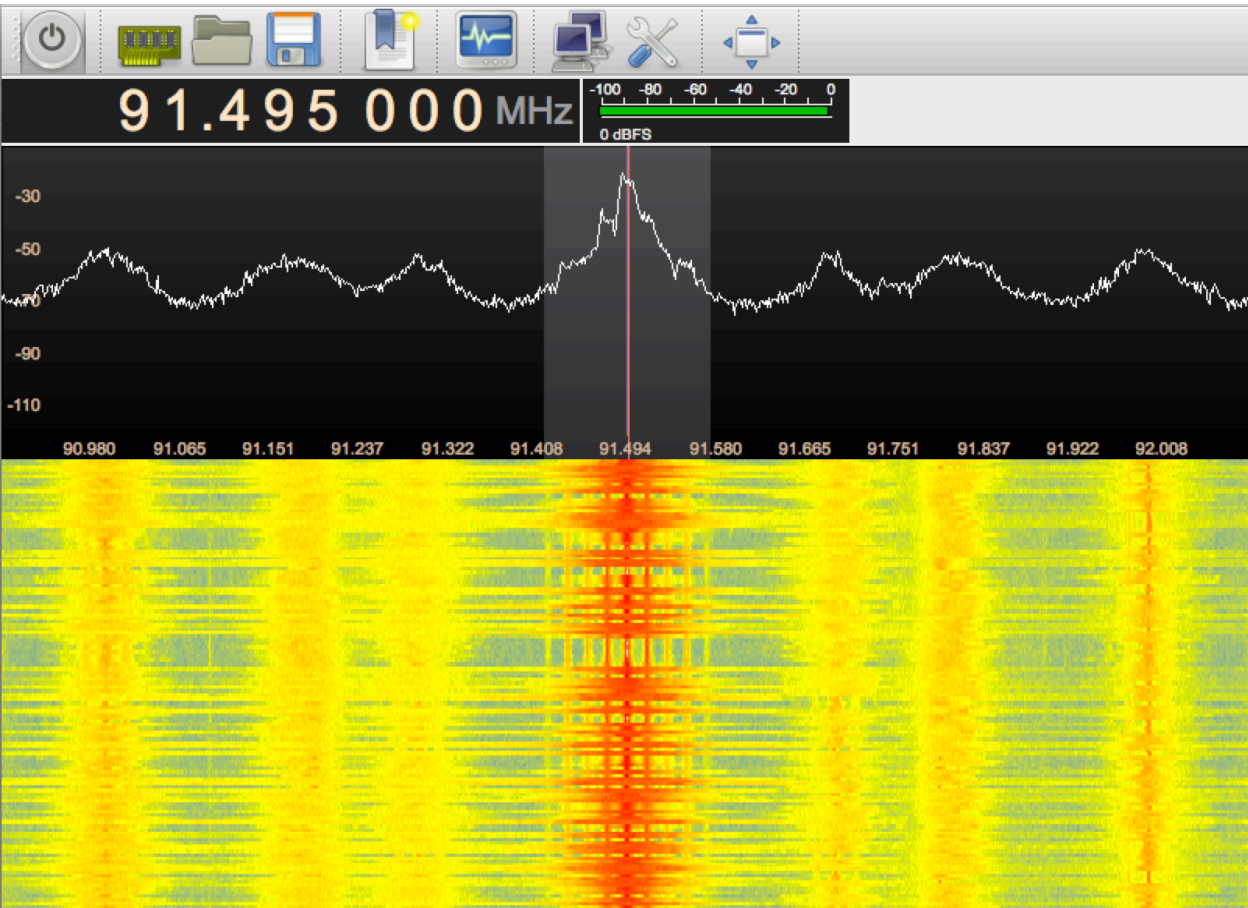
Ingresso antenna

Tuner chip (eterodina)

Risonatore a quarzo

Demodulatore digitale

Uscita USB



Receiver Options

kHz

Hardware freq: 91.493900 MHz

Filter width: Normal

Filter shape: Normal

Mode: WFM (mono)

AGC: Fast

Squelch: -150.0 dBFS

Noise blanker: NB1 NB2

Input controls Receiver Options

FFT Settings

FFT size: 32768 RBW: 36.6 Hz

Rate: 30 fps Overlap: 0%

Averaging: [Slider]

Pandapter: [Slider] WF

Peak: DEL Hold

Ref. level: [Slider] -10 dB

Range: [Slider] 120 dB

Zoom: [Slider] 1x

[R] [C] [D]

Color: [White] [Fill]

FFT Settings Audio RDS

Bookmarks

Frequency	Name	Modulation	Bandwidth
0 89495000	RAI1	WFM (stereo)	160000
1 92295000	RAI2	WFM (stereo)	160000
2 94595000	RAI3	WFM (stereo)	160000
3 107895000	RTL102.5	WFM (stereo)	160000

Untagged

What is GNU Radio?

GNU Radio is a free & open-source software development toolkit that provides signal processing blocks to implement software radios. It can be used with readily-available low-cost external RF hardware to create software-defined radios, or without hardware in a simulation-like environment. It is widely used in research, industry, academia, government, and hobbyist environments to support both wireless communications research and real-world radio systems.

What is Software Radio?

The [Wikipedia page on software radio](#) provides an excellent overview. In brief, a software radio is a radio system which performs the required signal processing in software instead of using dedicated integrated circuits in hardware. The benefit is that since software can be easily replaced in the radio system, the same hardware can be used to create many kinds of radios for many different communications standards; thus, one software radio can be used for a variety of applications!

What Does GNU Radio Do?

GNU Radio performs all the signal processing. You can use it to write applications to receive and transmit data with radio hardware, or to create entirely simulation-based applications. GNU Radio has filters, channel codes, synchronisation elements, equalizers, demodulators, vocoders, decoders, and many other types of blocks which are typically found in signal processing systems. More importantly, it includes a method of connecting these blocks and then manages how data is passed from one block to another. Extending GNU Radio is also quite easy; if you find a specific block that is missing, you can quickly create and add it.

La trasformata di Hilbert

Si consideri il solito impulso esponenziale

$$h(t) = \begin{cases} \exp(-t/\tau) & t \geq 0 \\ 0 & t < 0 \end{cases}$$

la cui trasformata di Fourier è

$$H(\omega) = \int_0^{+\infty} \exp(-t/\tau) e^{-i\omega t} dt = \frac{1}{\frac{1}{\tau} + i\omega} = \frac{\tau}{1 + i\omega\tau}$$

In particolare si può anche scrivere

$$H(\omega) = \frac{1}{\frac{1}{\tau} + i\omega} = -\frac{i}{\omega - \frac{i}{\tau}}$$

Se la costante di decadimento tende all'infinito l'impulso esponenziale tende a diventare una funzione theta di Heaviside, e quindi la trasformata di Fourier della funzione theta è

$$\Theta(\omega) = \int_{-\infty}^{+\infty} \theta(t) e^{-i\omega t} dt = \lim_{\varepsilon \rightarrow 0^+} \frac{-i}{\omega - i\varepsilon}$$

e ovviamente vale anche la formula analoga che si ottiene con la sostituzione $\omega \rightarrow -\omega$

$$\int_{-\infty}^{+\infty} \theta(t) e^{i\omega t} dt = \lim_{\varepsilon \rightarrow 0^+} \frac{i}{\omega + i\varepsilon}$$

Adesso si noti che

$$\theta(t) = \int_{-\infty}^t \delta(t') dt' \quad \Rightarrow \quad \theta'(t) = \delta(t)$$

Ricordando che la rappresentazione di Fourier della funzione delta è

$$\delta(t - t_0) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{i\omega(t-t_0)} d\omega$$

e che

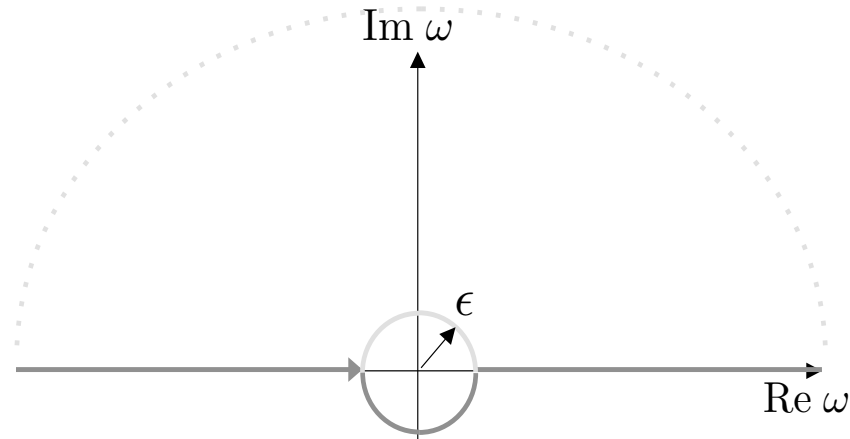
$$\frac{d}{dt} \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) e^{i\omega t} d\omega = \frac{1}{2\pi} \int_{-\infty}^{+\infty} i\omega F(\omega) e^{i\omega t} d\omega$$

si trova la seguente espressione per l' antitrasformata di Fourier della funzione theta

$$\theta(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \frac{1}{i\omega} e^{i\omega t} d\omega$$

Questa espressione integrale ha un problema di definizione intorno all'origine (nel dominio della frequenza), e l'integrale può essere definito solo tramite un processo al limite.

Si segue il cammino di integrazione mostrato nella figura seguente, chiudendolo nel semipiano superiore, e passando sopra oppure sotto la singolarità nell'origine, che dà quindi (in media!) un contributo che è uguale a metà dell'integrale fatto su tutta la circonferenza che circonda l'origine



e quindi – notando che il residuo in 0 vale 1 – l'integrale può essere scritto come

$$\begin{aligned} \theta(t) &= \frac{1}{2\pi} \int_{-\infty}^{+\infty} \frac{1}{i\omega} e^{i\omega t} d\omega = \frac{1}{2\pi i} \left[\left(\lim_{\varepsilon \rightarrow 0} \int_{-\infty}^{-\varepsilon} \frac{1}{\omega} e^{i\omega t} d\omega + \lim_{\varepsilon \rightarrow 0} \int_{+\varepsilon}^{+\infty} \frac{1}{\omega} e^{i\omega t} d\omega \right) + \frac{1}{2} 2\pi i \right] \\ &= \frac{1}{2\pi} P \int_{-\infty}^{+\infty} \frac{1}{i\omega} e^{i\omega t} d\omega + \frac{1}{2} \end{aligned}$$

L'ultima formula può anche venire riarrangiata nel modo seguente

$$P \int_{-\infty}^{+\infty} \frac{1}{\pi t} e^{-i\omega t} dt = -i[2\theta(\omega) - 1] = -i \operatorname{sgn}(\omega)$$

Infine questo significa che la trasformata di Hilbert, che è definita dalla formula

$$H(f)(t) = \frac{1}{\pi} P \int_{-\infty}^{+\infty} \frac{f(\tau)}{t - \tau} d\tau$$

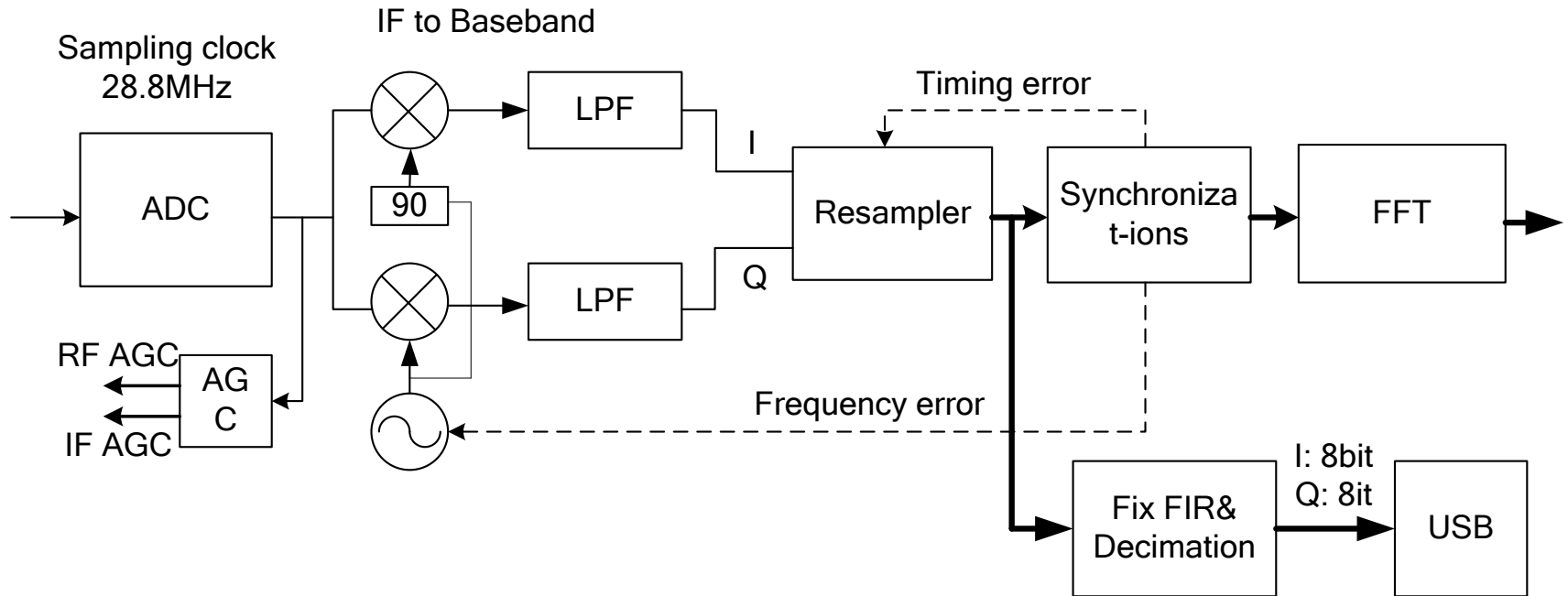
e che corrisponde ad una convoluzione, può essere scritta come segue nel dominio della frequenza

$$-i \operatorname{sgn}(\omega) F(\omega) = \begin{cases} e^{i\pi/2} F(\omega) & \omega < 0 \\ e^{-i\pi/2} F(\omega) & \omega > 0 \end{cases}$$

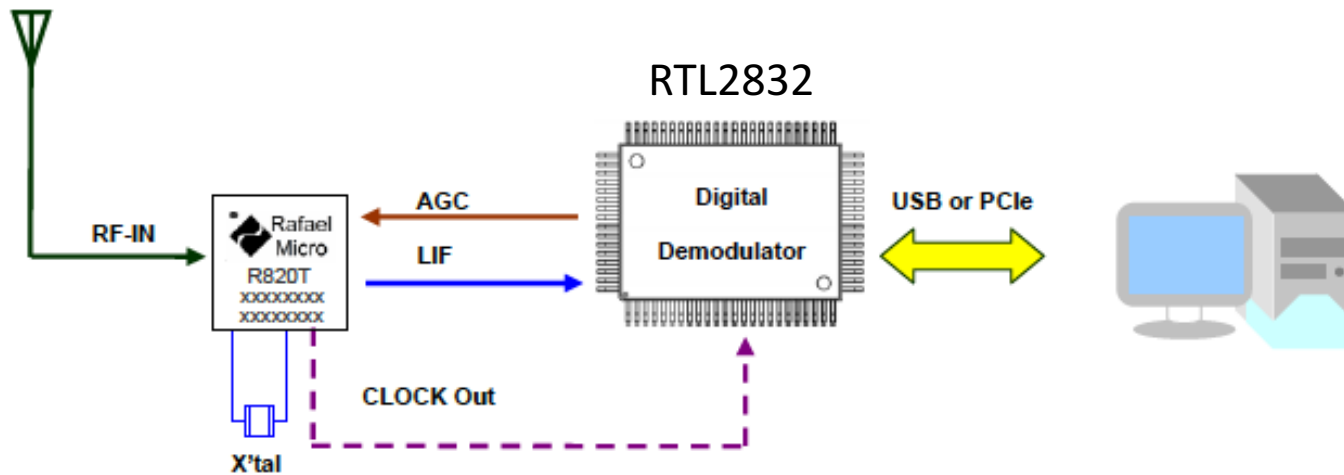
quindi la trasformata di Hilbert realizza uno shift di fase di $\pm 90^\circ$ che dipende solo dal segno della frequenza.

Modulazione SSB ... (v. nota di R. Lyons)

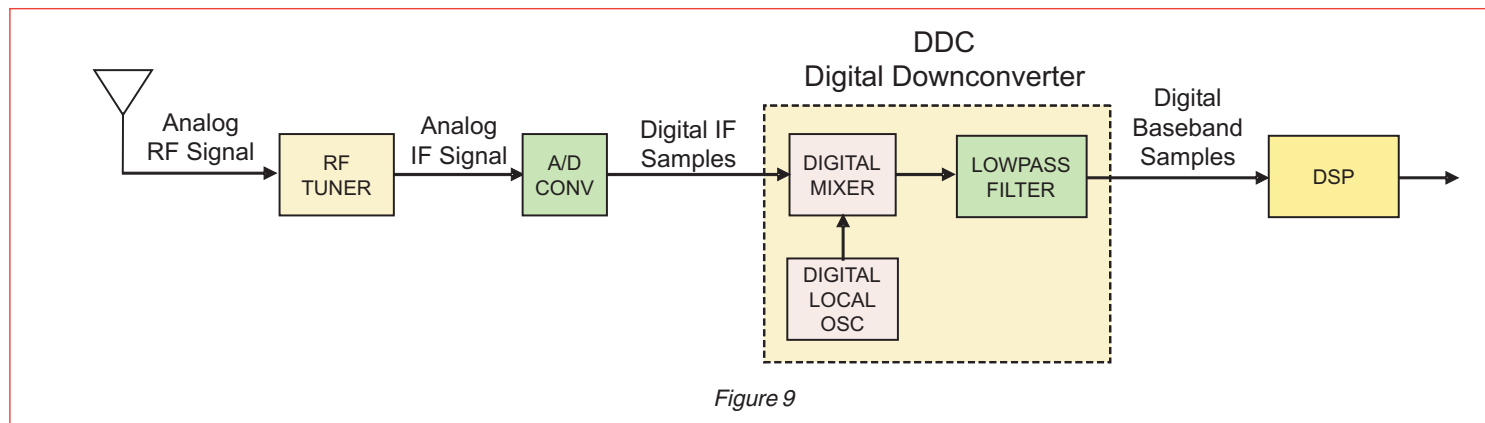
Struttura del microcircuito RTL2832

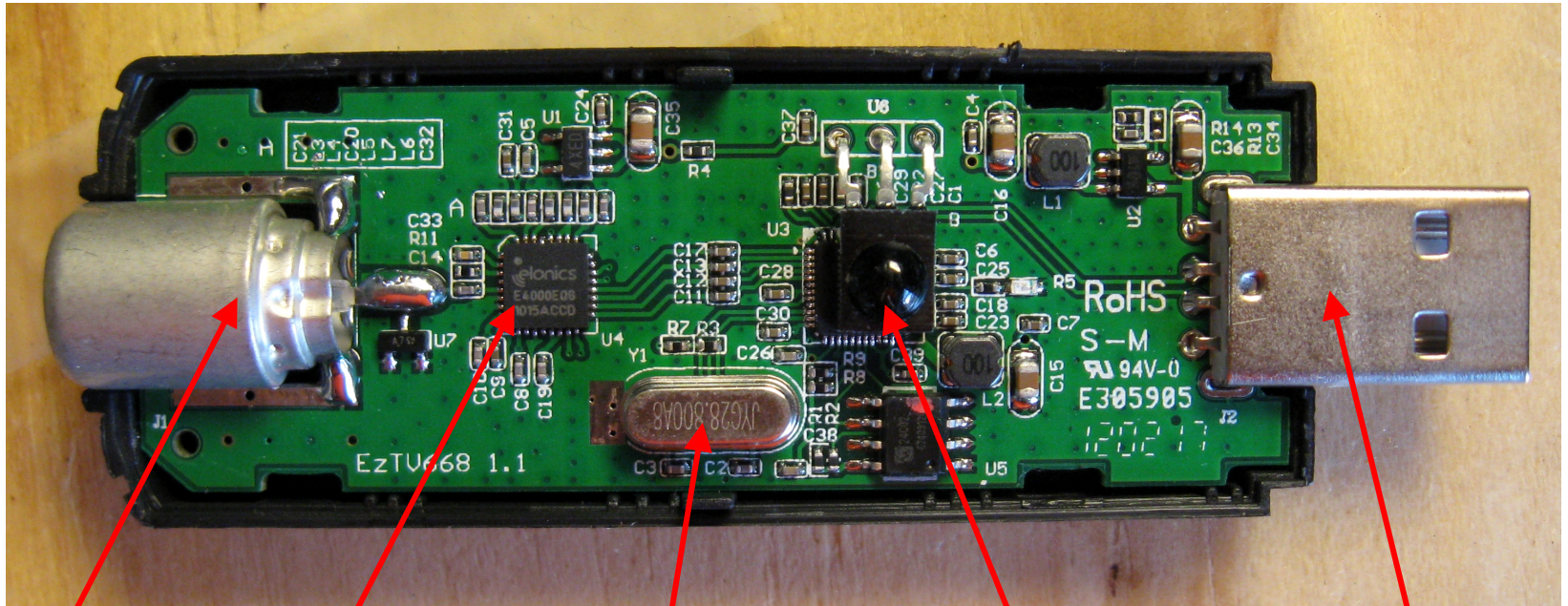


Radio digitale basata sul microcircuito RTL2832 (ADC veloce e demodulatore)



SDR Receiver Block Diagram





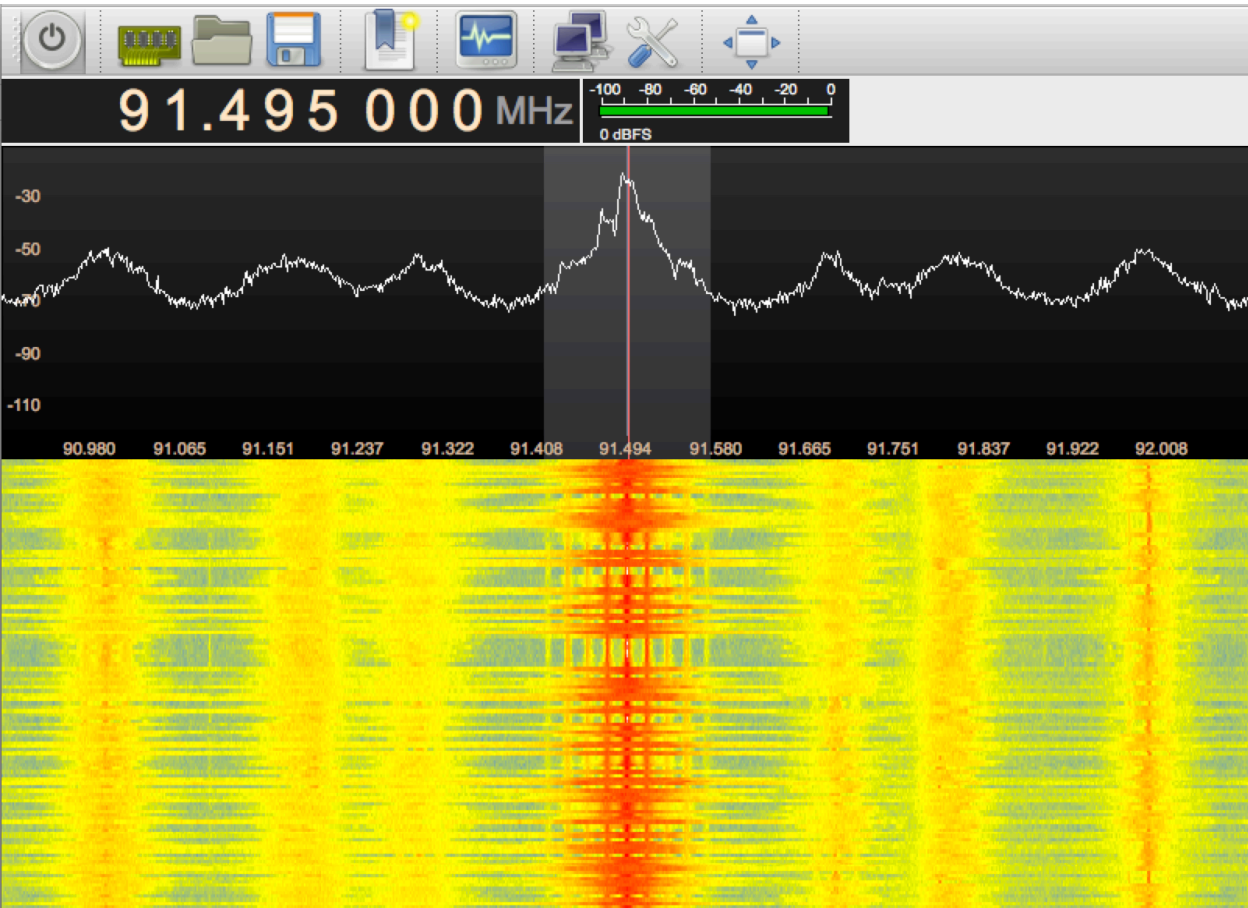
Ingresso antenna

Tuner chip (eterodina)

Risonatore a quarzo

Demodulatore digitale

Uscita USB



Receiver Options

kHz

Hardware freq: 91.493900 MHz

Filter width: Normal

Filter shape: Normal

Mode: WFM (mono)

AGC: Fast

Squelch: -150.0 dBFS

Noise blanker: NB1 NB2

Input controls | **Receiver Options**

FFT Settings

FFT size: 32768 RBW: 36.6 Hz

Rate: 30 fps Overlap: 0%

Averaging: [Slider]

Pandapter: [Slider] WF

Peak: DEL Hold

Ref. level: [Slider] -10 dB

Range: [Slider] 120 dB

Zoom: [Slider] 1x

[R] [C] [D]

Color: [White] [Fill]

FFT Settings | Audio | RDS

Bookmarks

Frequency	Name	Modulation	Bandwidth
0 89495000	RAI1	WFM (stereo)	160000
1 92295000	RAI2	WFM (stereo)	160000
2 94595000	RAI3	WFM (stereo)	160000
3 107895000	RTL102.5	WFM (stereo)	160000

Untagged

RTL-SDR.COM

RTL-SDR (RTL2832U) and software defined radio news and projects. Also featuring Airspy, HackRF, FCD, SDRplay and more.

HOME

ABOUT RTL-SDR

QUICK START GUIDE

FEATURED ARTICLES ▾

SOFTWARE ▾

SIGNAL ID WIKI

FORUM

RTL-SDR STORE

FEBRUARY 11, 2014

THE BIG LIST OF RTL-SDR SUPPORTED SOFTWARE

There are now dozens of software defined radio packages that support the ultra cheap [RTL-SDR](#). On this page we will attempt to list, categorize and provide a brief overview of each software program. We categorize the programs into general purpose software, single purpose software, research software and software compatible with audio piping.

If you know of a program that is missing please leave a comment in the comments section at the bottom of the page.

13/02/2014 – Added Sodira, gr-wmbus, rtl_sdr_waterfall, QTRadio, multimon, sdrangelove, lte-scanner, rtl_tcp, rtl_sdr_FS20_decoder.

17/02/2014 – Updated the Linrad description.

28/04/2014 – Added Modesdeco and Trunk88.

30/05/2014 – Added RTL Panorama, RTL SDR Panoramic Spectrum Analyzer, Chrome Radio Receiver, SeeDeR, DAB Player, RTL SDR Installer, PD/Max Wrapper, SDRWeather, LTR Analyzer, softEOT/softDPU and ScanEyes.

26/07/2014 – Added PiAware, OOK-Decoder, rtl_fm_python, rtl_power heatmap viewer, RTL Bridge, threejs-spectrum, CANFI Software, PNAIS, FLARM Decoder, Xastir, RTLSDR-Airband, SDRTrunk.

13/11/2014 – Added Touchstone, RFAalyzer, RTL1090 XHSI Interface, Parus Decoder, PlotRTL1090, LRPT Decoder.

05/02/2015 – Added rtl_tool_kit, CubicSDR, OregonWeather, FreqWatch.

15/04/2015 – Added ADSBox, YouSDR, FlightAware Flight Feeder, Frequensea, Track your flight EUROPE, QSpectrumAnalyzer, Doppler & Demod, Redsea, rtl_heatmap, gr-gsm, driveby, SDRRecord.

23/12/2015 – Added Remote rtl_udp, AISRec, dump978, AISDeco2, SDRrecorder, OpenWebRX, dsame, RTL-Widespectrum,

ADS-B

ADS-B is a surveillance technology incorporating air and ground aspects that provide Air Traffic Control (ATC) with a more accurate picture of the aircraft's three-dimensional position in the enroute, terminal, approach and surface environments. The aircraft provides the airborne portion in the form of a broadcast of its identification, position, altitude, velocity, and other information.

The ground portion is comprised of ADS-B ground stations which receive these broadcasts and direct them to ATC automation systems for presentation on a controller's display similar in nature to a radar return. ADS-B is automatic because no external interrogation is required. It is dependent because it relies on onboard position sources and broadcast transmission systems to provide surveillance information to ATC.

ADS-B allows ATC to monitor and separate aircraft efficiently, and with more precision. Because it uses GPS signals, it expands surveillance services into areas where little or no radar coverage exists. The technology provides improved situational awareness to pilots and ATC.

Providing a flexible and expandable platform to accommodate future air-traffic growth, ADS-B is designed to improve the safety, capacity and efficiency of the airspace around the world.

ADS-B equipment is currently mandatory in portions of Australian airspace, the United States requires some aircraft to be equipped by 2020 and the equipment will be mandatory for some aircraft in Europe from 2017. Canada is already using ADS-B for air traffic control.



Overview

GNU Radio is a free software development toolkit that provides the signal processing runtime and processing blocks to implement software radios using readily-available, low-cost external RF hardware and commodity processors. It is widely used in hobbyist, academic and commercial environments to support wireless communications research as well as to implement real-world radio systems.

GNU Radio applications are primarily written using the Python programming language, while the supplied, performance-critical signal processing path is implemented in C++ using processor floating point extensions where available. Thus, the developer is able to implement real-time, high-throughput radio systems in a simple-to-use, rapid-application-development environment.

If you're looking for information on GNU Radio, **start with the wiki**. To browse the git repositories use <http://gnuradio.org/cgit>

- Subprojects: **GRCon14**

Issue tracking

- **Bug**: 59 open / 551
- **Feature**: 16 open / 277
- **Support**: 0 open / 0

Members

Manager: **Eric Blossom, Johnathan Corgan, Martin Braun, Matt Ettus, Nathan West, Tom Rondeau**

Developer: **Achilleas Anastasopoulos, Alexandru Csete, Balint Seeber, Ben Hilburn, Ben Reynwar, Dimitrios Symeonidis, Doug Geiger, John Malsbury, Martin Braun, Martin Dvh, Michael Dickens, Moritz Fischer, n4hy McGwier, Nathan West, Neel Pandeya, Nicholas Corgan, Nick Foster, Patrick Strasser, Philip Balister, Sebastian Koslowski, Seth Hitefield, Sylvain Munaut, Tim O'Shea**

Reporter: **Al Whaley, Alex Dusowitz, Alex Ruffell, Alexander Chemeris, Alexander Limonov, Alexandre Pierrot, Alexandru Csete, Alfredo Muniz, Alon Levy, Amarnath alapati, Andre Puschmann, Andrej Lajovic, andres lucena, Andrew Back, Andrew Davis, andy shi, Ankit Kaushik, anton komarov, Antonio Cantoni, Anupama Purohit, aravindan arun, Arturo Rinaldi, Asier Alonso, Axelle Apvrille, Ayman Shalaby, Balint Seeber, Bastian Bloessl, Ben Hilburn, Ben Reynwar, Bill Pretty, Brian Padalino, Catalin P, Cauresew Cauresew, Chaouki KASMI, Chris Paget, Christoph Hausl, Christophe Devine, Chí-Thanh Christopher Nguyễn, Daniel Bovensiepen, Daniel Mundall, David Burgess, David Lawrence, Devang M, Diane Bruce, Dick Carrillo, Dimitri Stolnikov, Don**