Software Defined Radio - SDR

R. Zitouni

rafik.zitouni@esiee.fr

Plan

- Software Defined Radio concept
  - Motivation
  - Use cases
  - Definition and Architecture
  - Features
- Software Communication Architecture (SCA)
- Universal Software Radio Peripheral and GNU Radio
- Demonstrations and tools
- GNU Radio

Motivation

Traditional transceivers are in Hardware

All the following components are in **Hardware**

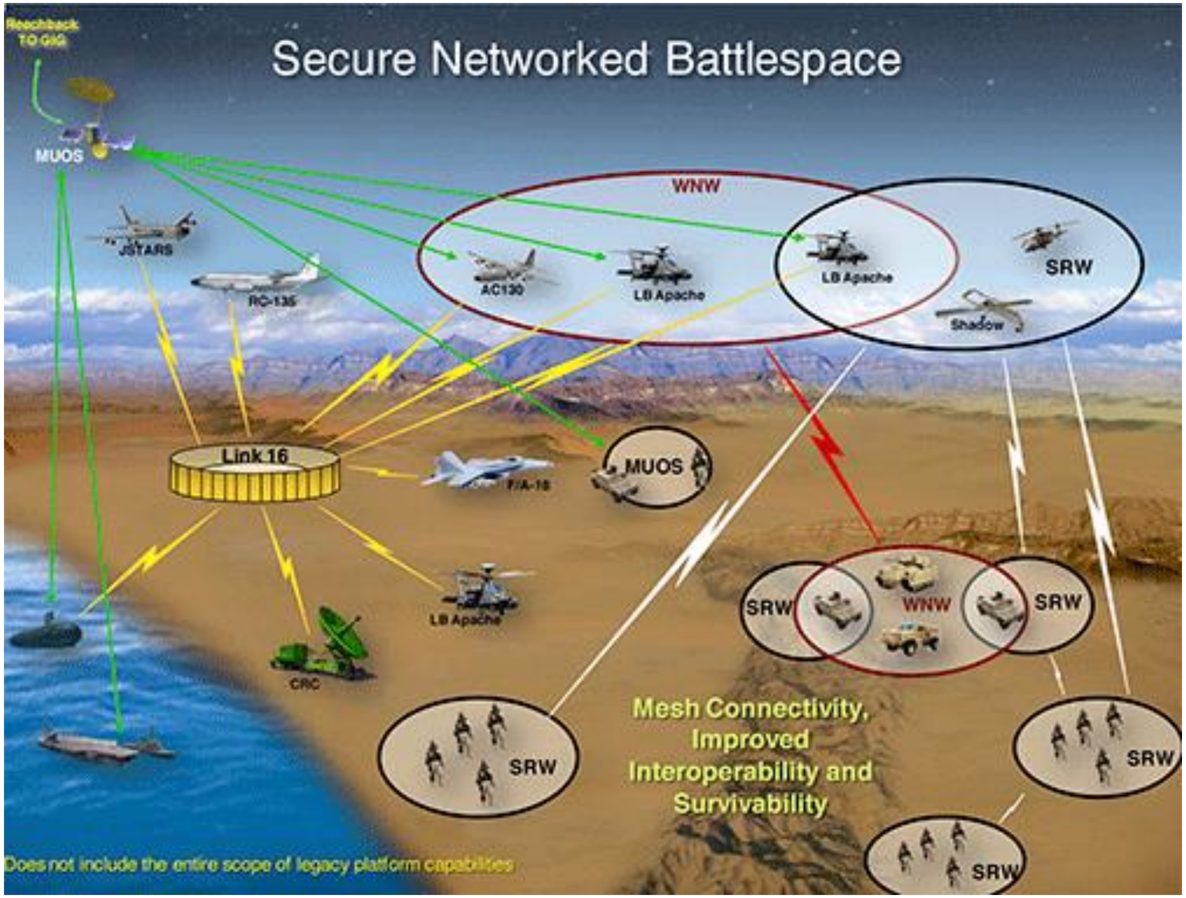
- Amplifiers
- Modulators
- Demodulators
- Mixers
- Filters
- Oscillators

  .etc

How to virtualize them in **Software**

Motivation

The U.S. Military's Joint Tactical Radio System (DoD)
Joint Tactical Radio System (JTRS) → US Military
European Secure Software Defined Radio SSOR project → Thales
CONTACT project



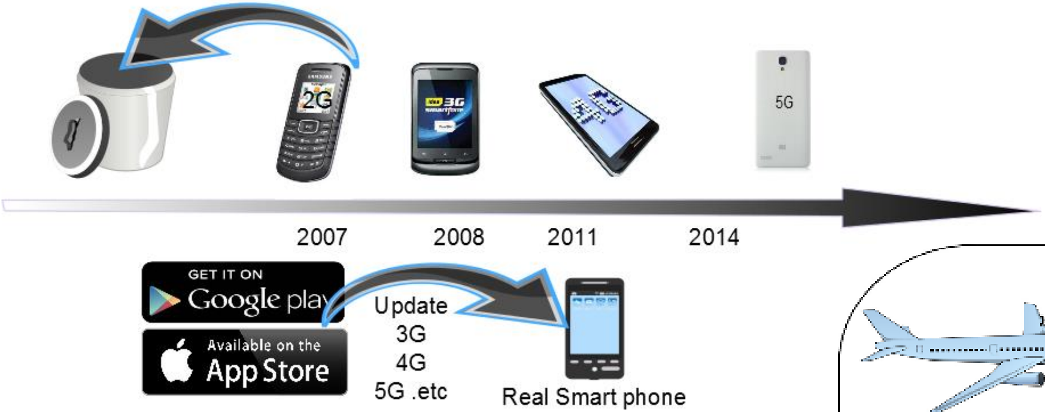
Each infrastructure has its wireless or radio network
SRW: Soldier Radio Waveform
WNW: Wide Band Networking Waveform AdHoc Netwrok
etc.

How to ensure the **interoperability** between **heterogeneous wireless networks** ?

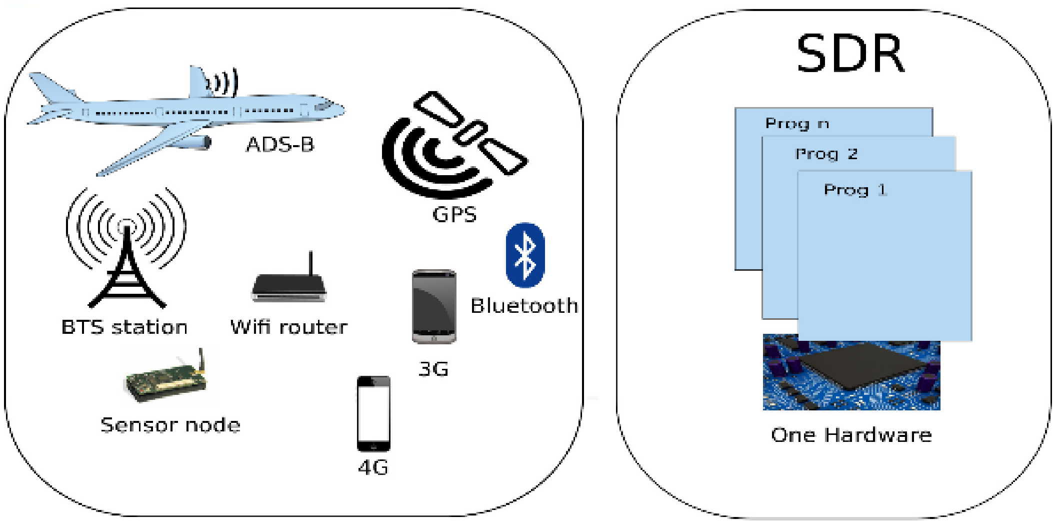
Motivation

How to implement **standard specifications** in software avoiding the waste of hardware components?

e.g. We can imagine an application of hardware updating version (3G, 4G, 5G, etc.) from App Store or Google Play.



Interoperability at Physical Layer



Maintainability of Physical Layer

Motivation

Applications:

From my youtube channel, you can find
https://www.youtube.com/watch?v=xN4CQsMqZv0

➢ SDR receiver of ADS-B transponder
Determines aircrafts positions arround Paris.

➢ GSM receiver
Sniff data beacon frames of Bouygue Telecom base stations.

Tools:

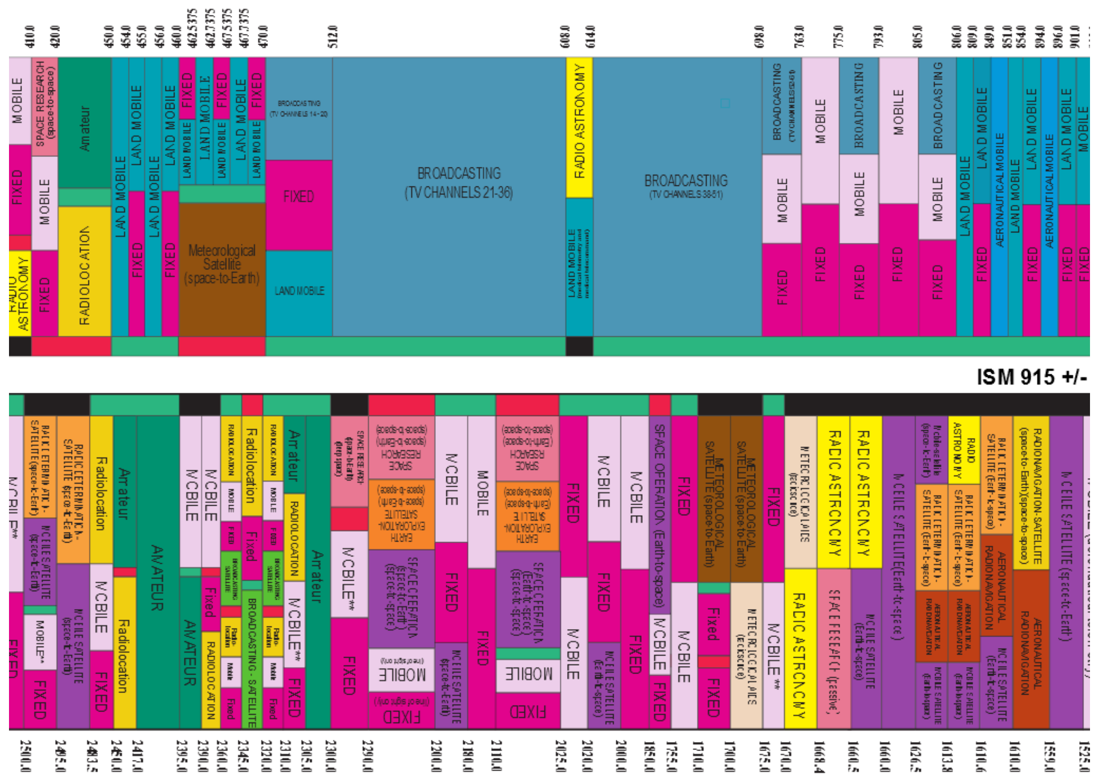
Hardware: RTL-SDR dongle
Software: GNU Radio toolkit

Needs and use cases

Needs

- **Flexibility**, compactness and low cost design
- Easy **maintainability**, **evolutivity** and **reusability**
- Efficient use of the available **scarce frequency**

The spectrum of **frequency bands is crowded** with a high number of applications and technologies.



Needs and use cases

Use cases:

- For manufacturer
  - **Rapid prototyping** for test and performance evaluation
- For research and hobbyist
  - **Replace simulation** by the true and real communications
- For teaching
  - **Pedagogical technology** to explain signal processing functions and wireless networking by an intuitive software application

Definition and Architecture

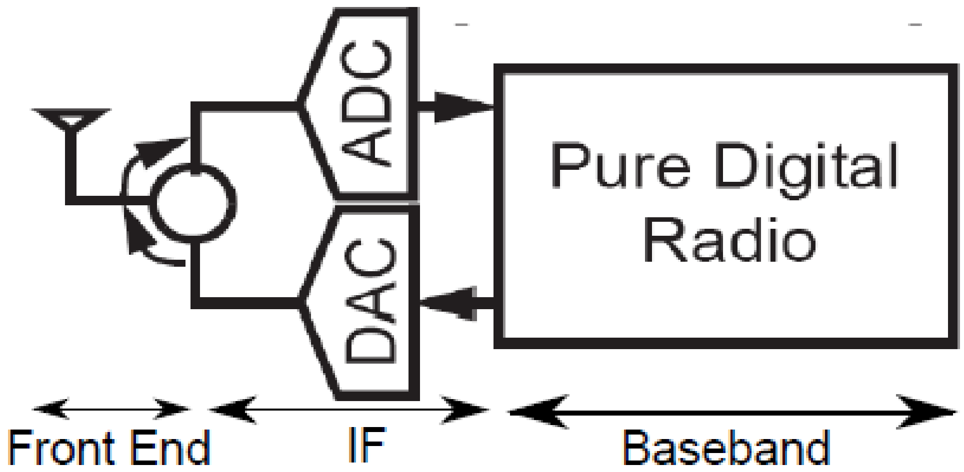
Definition and Histroy:

A **radio Tx/Rx** employing a technology that allows the **RF operating parameters** such as frequency band, modulation type, or output power .etc **to be set** or **altered** by **software.**

- Concept introduced publicly by **Joseph Mitola** in 1991.
- First SDR developed by the U.S. military : **SpeakEasy**
    - Programmable microprocessors for implementing more than **10 military** communication standards
    - Frequencies ranged from **2 MHz to 2 GHz.**

Definition and Architecture

Architecture:



The **DAC** and **ADC** should be as close as possible to the ends of the receiver and transmitter antenna

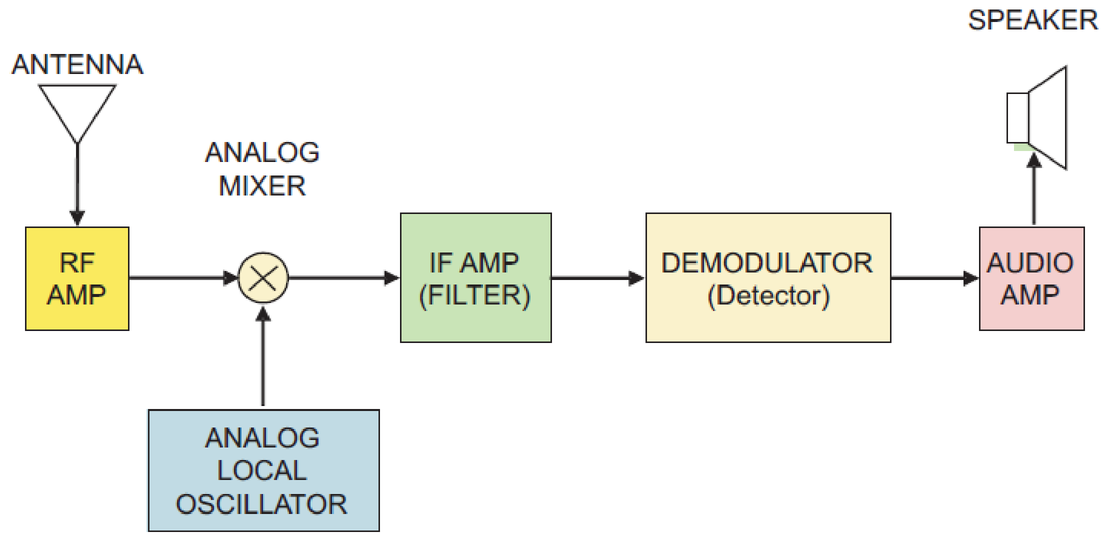
The three parts of an SDR are:

1)RF **Front End**

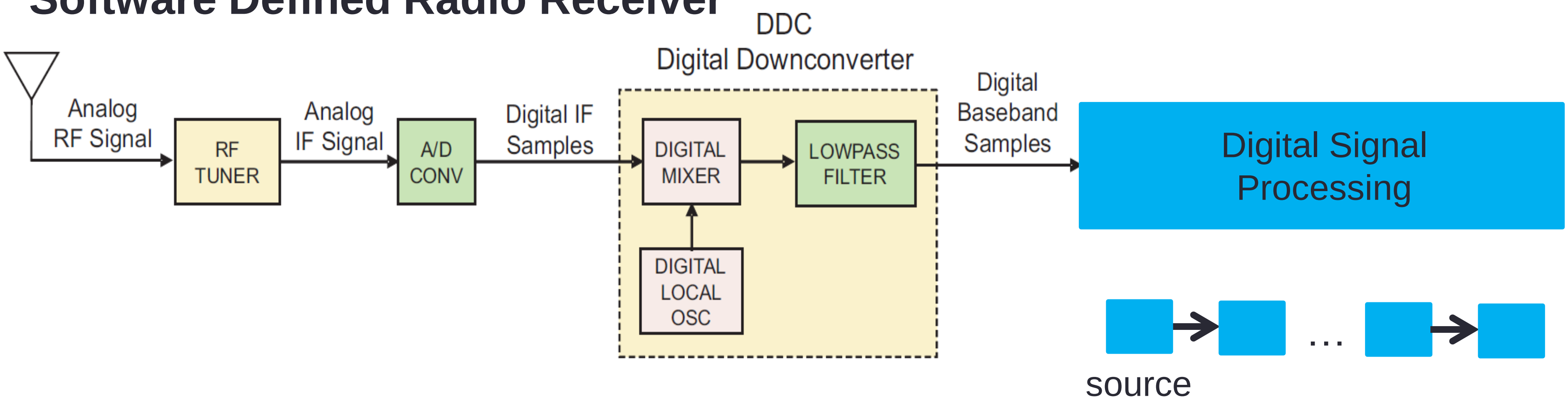
2)Intermediate Frequency **IF**

3)Baseband processing by software

Analog receiver



Software Defined Radio Receiver



Definition and Architecture

More than **60 platforms**

Classification of SDR platforms

Standardization
- **Software Communication Architecture (SCA)**
- Reconfigurable Radio System (RRS) → advanced GNU Radio & USRP

Programming model
- **General Purpose Processor (GPP)**
- Coprocessor
- Distributed Approach
- Processor centric Approach
- Programming blocks approach
- Reconfigurable Units

Processor Model
- **General Purpose Processor (GPP)**
- **FPGA**
- DSP
- ASIC

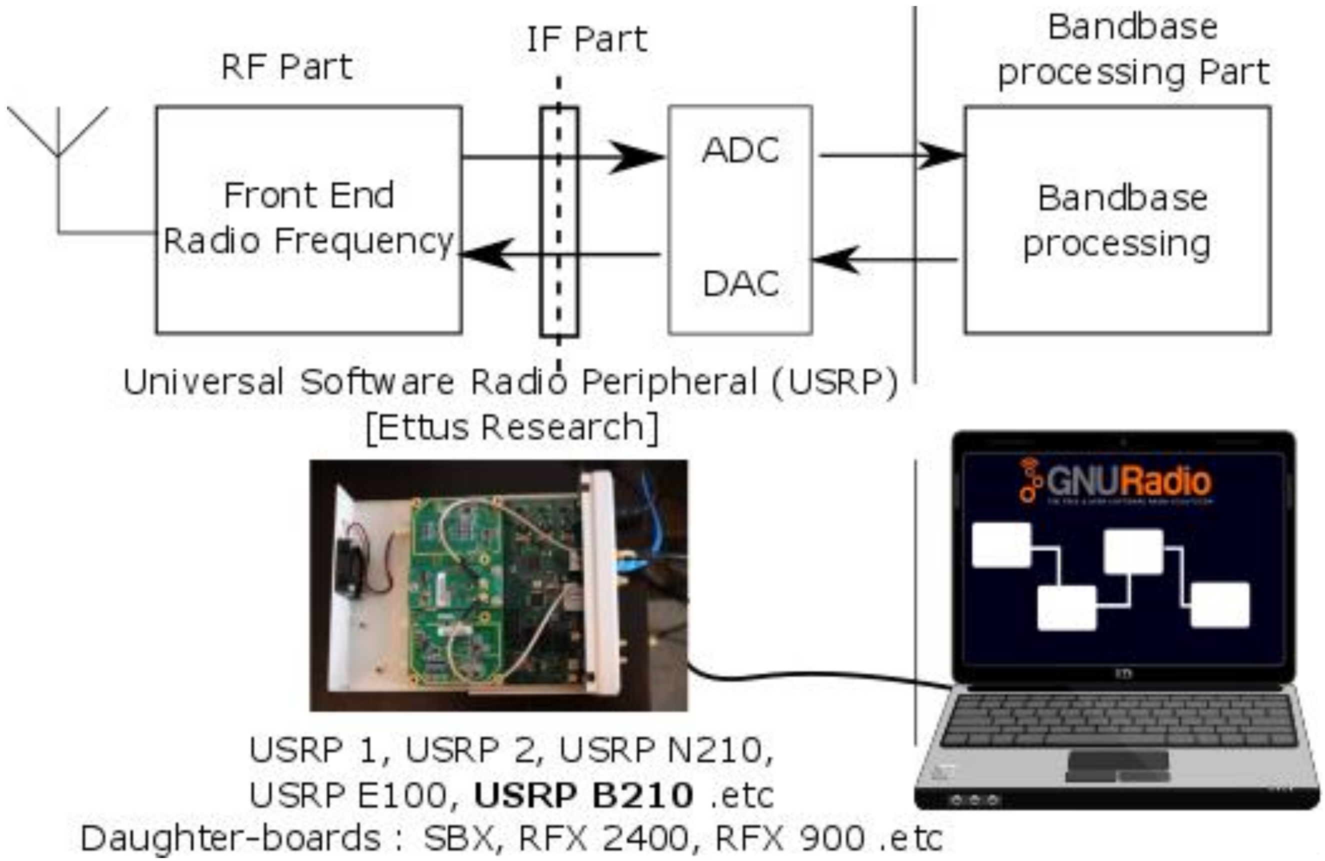
FPGA: Field Programmable Gate Array
ASIC: Application Specific Integrated Circuit

Definition and Architecture

Example of an Open source platform

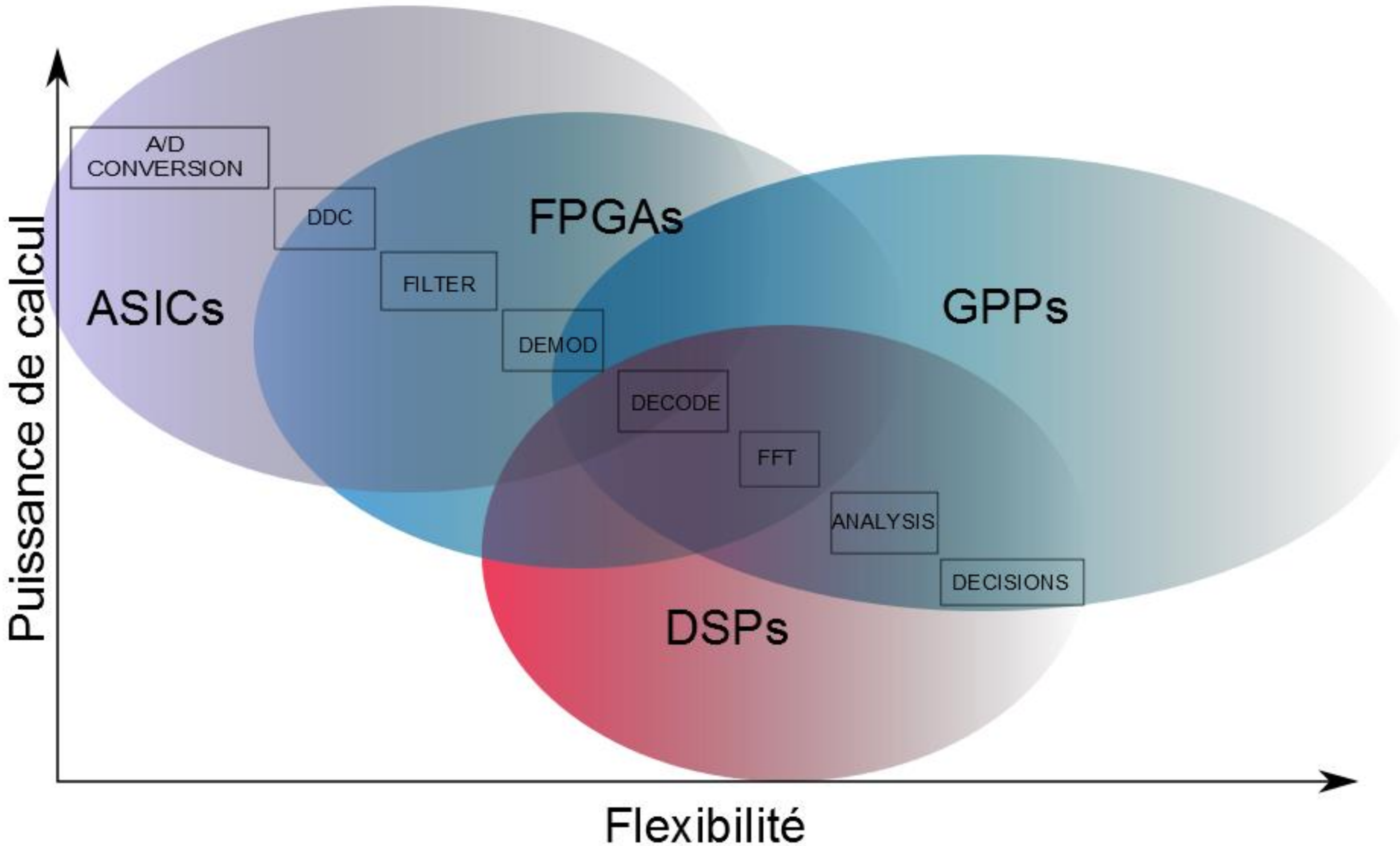
Universal **S**oftware **R**adio **P**eripheral (**USRP**) → (with an **FPGA for DDC**)

General **P**urpose **P**rocessor (**GPP**) → (run a GNU Radio toolkit)



Definition and Architecture

How to select **a suitable SDR** platform to develop a software transmitter/receiver?



SDR Features

Features of SDR

- **Recongurability** and reprogrammable of radio Tx/Rx
- **Baseband waveform** or radio functionality **stored in memory**
  - **Different modulations**, error correction coding
  - **Functional blocks can potentially be changed** in real time
- **Operating parameters** of functional blocks **can be adjusted** by designer or automated programs.

Baseband refers to the original data signal (**without up-converting with high frequency carrier**), whereas passband refers to filtered signal that was originally modulated onto a carrier.

SDR Features

- **Multifunctionality:** Same digital communication system platform which support multiple types of radio functions. *e.g*. Download software design of a radio.

- **Global Mobility:** Standards are defined for particular geographical area. With an SDR, the transceivers operations are transparent. *e.g.* ISM frequency bands are different from Europe and US.

- **Compactness and Power Efficiency:** One SDR platform supports many communication standards.

- **Ease of manufacturing:** Migrate a baseband functions from hardware to software

- **Ease of Upgrading:** Latest updates of standard specifications are considered by updating firmware software.

Software Communication Architecture (SCA)

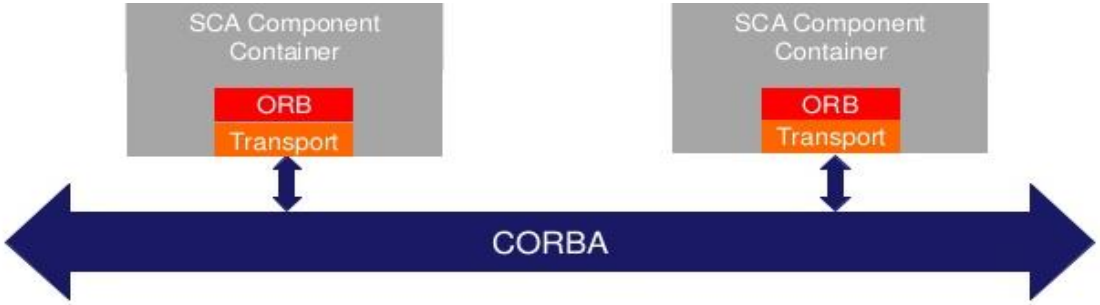
Software Communication Architecture (SCA)

Definition

Open architecture framework that tells communications systems designers how elements of hardware and software are to operate in harmony. **[Hay03]**

It enables communication platforms (e.g. **software defined radios**) to load applications (e.g. **waveforms**), **run these applications**, and be networked into an integrated system.

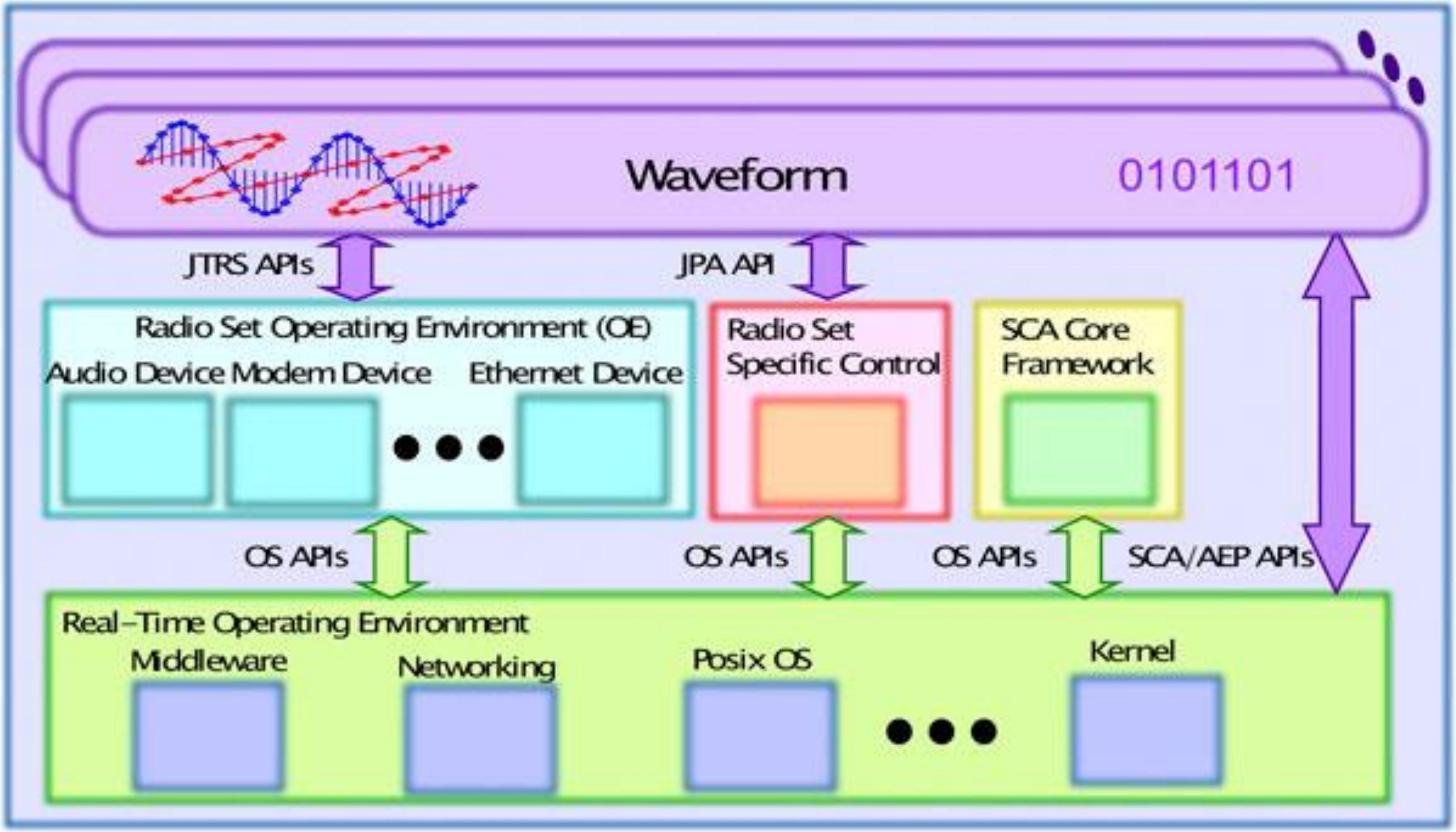
It was created for **JTRS**



Simplified Architecture

Software Communication Architecture (SCA)

Architecture



Parameters of Software Defined Radio

Hardware Device

Designer should select **Front End Hardware** device based on its performances and application needs.

- Frequency Band **(MHz/GHz)**
- Output Power **(dBm, mW, Watt)**
- Sampling Rate of DAC/ADC **(Ms/s)**
- Maximum sampling Rate at application layer **(Ms/s)**
    - Frequency bandwidth **(MHz)** at application layer

USRP N210

USRP E310

USRP B210

RTL-SDR (20 $ -TV dongle)

Parameters of Software Defined Radio

Software Toolkit

- **GNU Radio:** Open source software consisting of C/C+ libraries wrapped with Python scripts
- **LabView:** Graphical programming with National Instrument devices
- **Simulink:** Blocks as GNU Radio blocks but on Matlab environment

Universal Software Radio Peripheral – GNU Radio

Universal Software Radio Peripheral

USRP B210

- ➢Frequency range: **70 MHz - 6 GHz**
- ➢Up to **56 MHz** of instantaneous bandwidth (**61.44MS/s quadrature)**
- ➢**Full duplex, MIMO (2 Tx & 2 Rx)**
- ➢Fast and convenient bus-powered USB 3.0 connectivity

Covered Applications

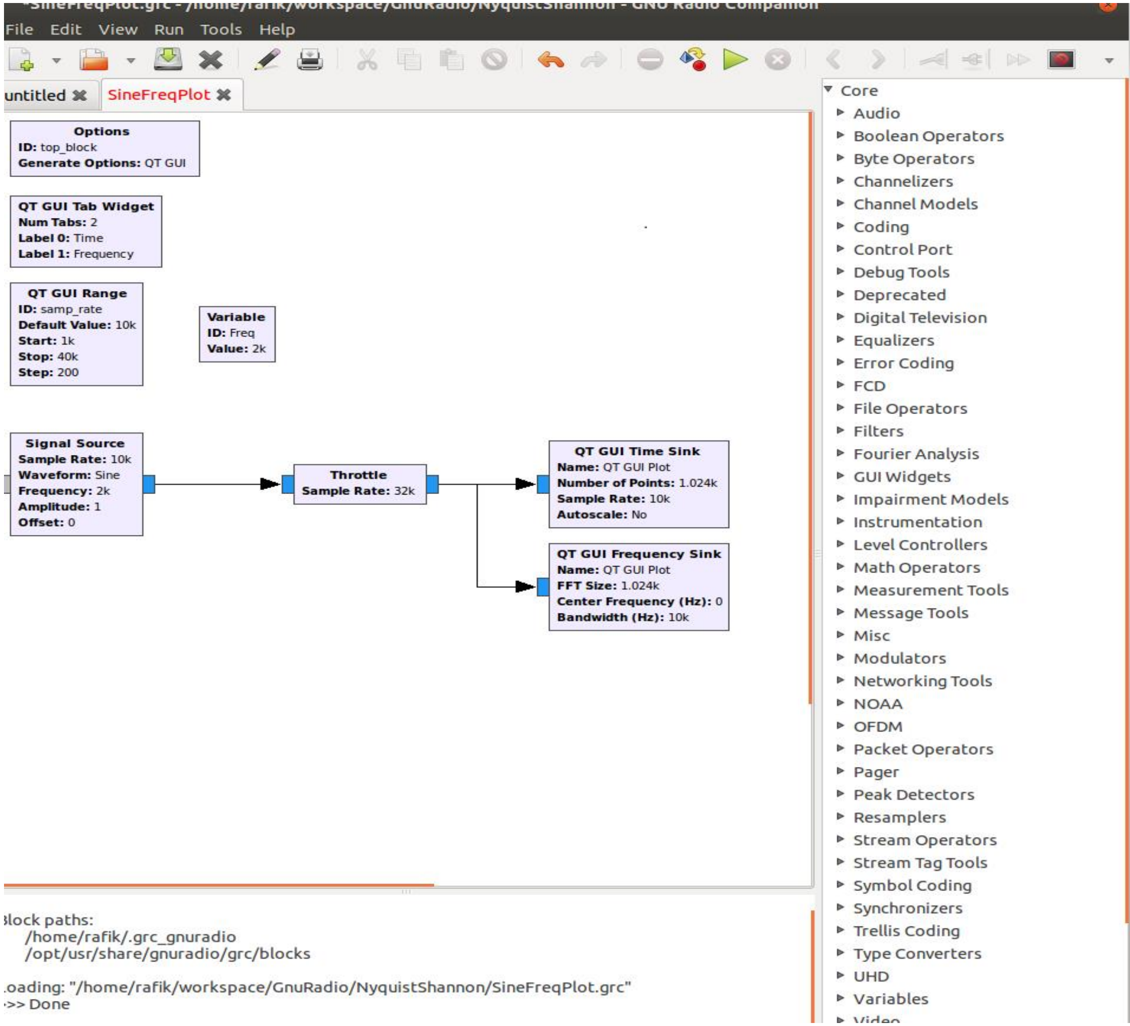
All wireless communications in the frequency band from 70 MHz o 6 GHz

- ➢ Wifi networks (IEEE 802.11 g/ac/p)
- ➢ ZigBee (IEEE 802.15.4)
- ➢ ADS-B transponder, GSM .etc

GNU Radio

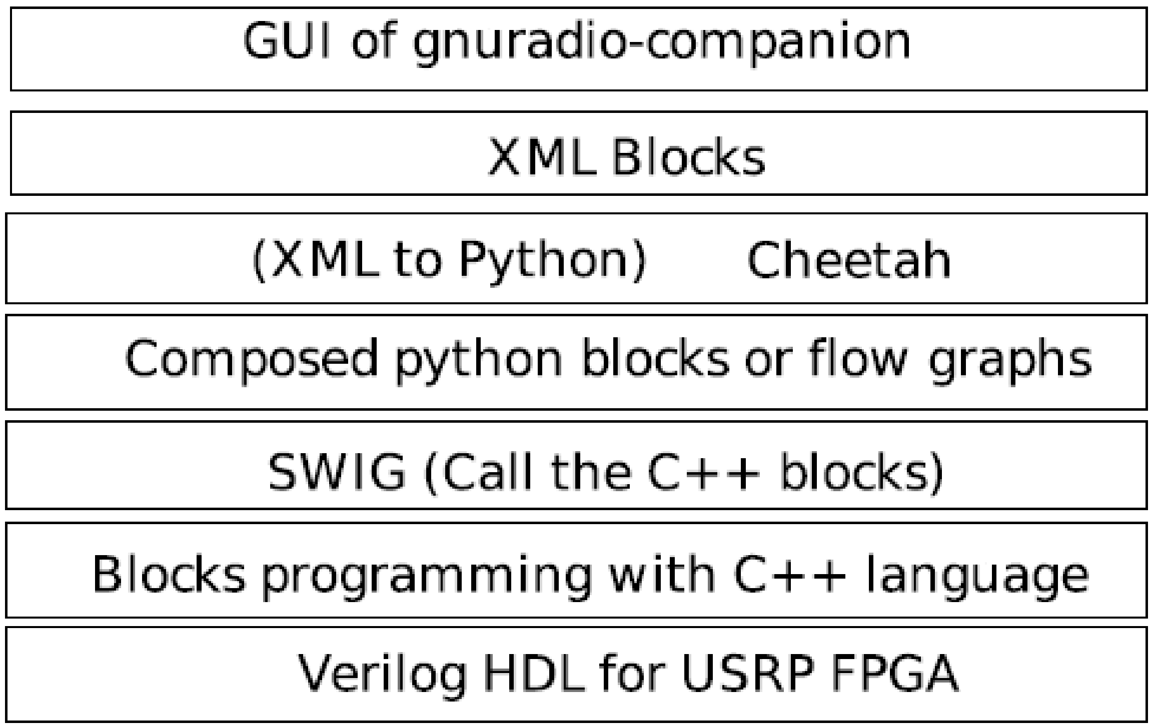
GNU Radio companion is the GUI of the GNU Radio toolkit. It offers more than **100 blocks.**

$ gnuradio-companion



GNU Radio

Software Layres of the GNU Radio toolkit

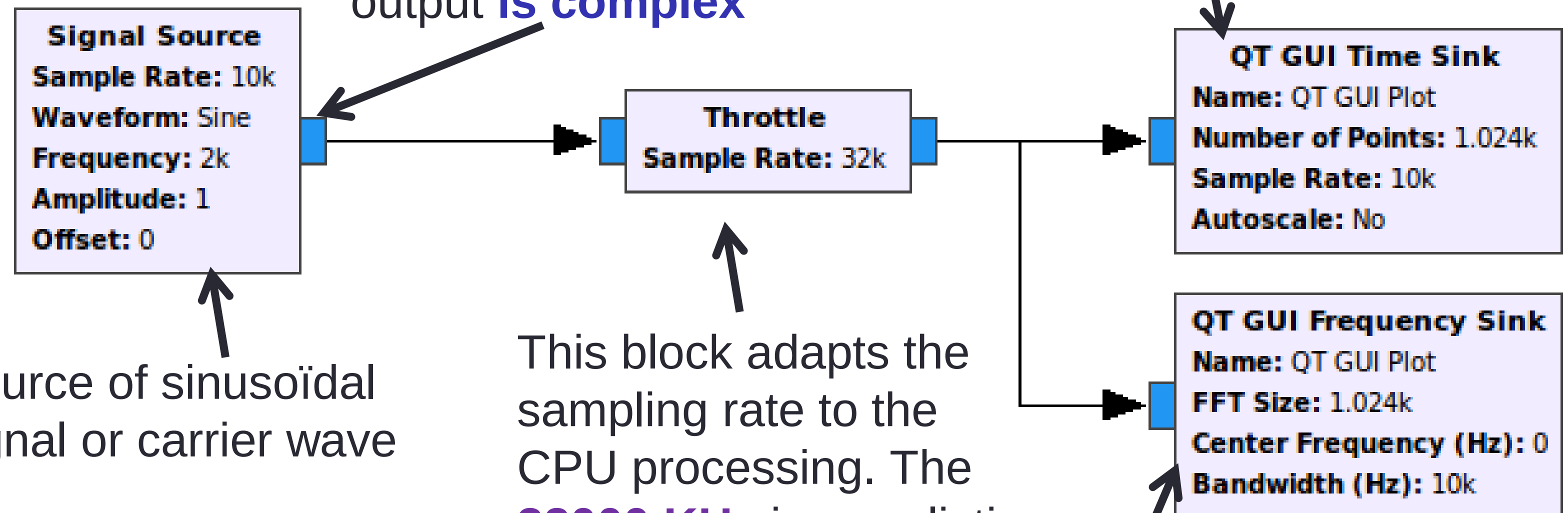


Examples of GNU Radio programs

Flow graphs are TX/RX chains or GNU Radio Programs

The type of the output **is complex**

The time domain of our waveform



Source of sinusoïdal signal or carrier wave

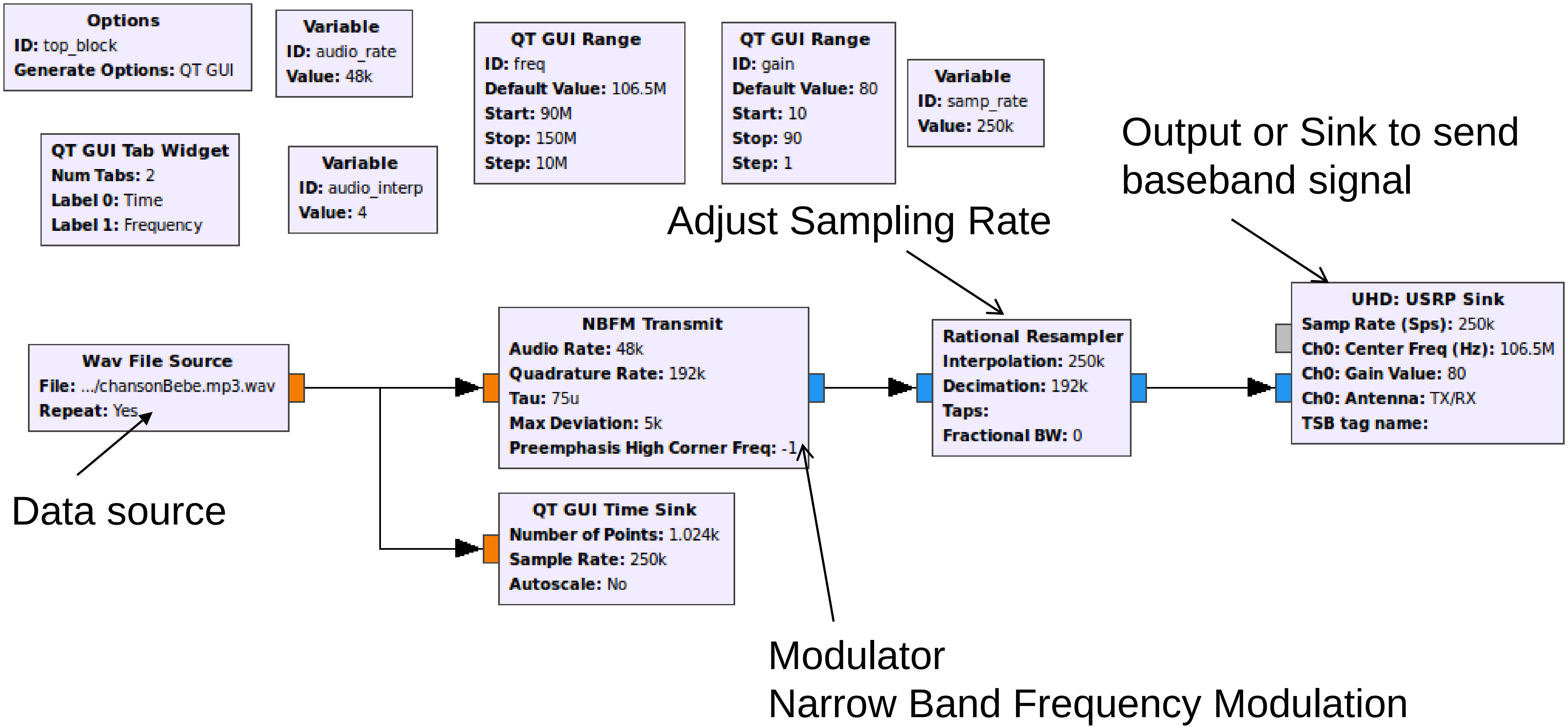
This block adapts the sampling rate to the CPU processing. The **32000 KHz** is a realistic sampling rate

The waveform in frequency domain

Flow graphs

Example of a transmitter's flow graph (.grc)

We start from **a signal or data source** to a **hardware sink block**



Examples of Applications

➢ **Flow graph illustrating theoretical concepts**

What is the concept illustrated in the following demonstration?

➢ **Sniff Wireless Local Area Networks (WLAN) WiFi IEEE 802.11 g/a/p**

Name 6 type of messages exchanged in WiFi networks?

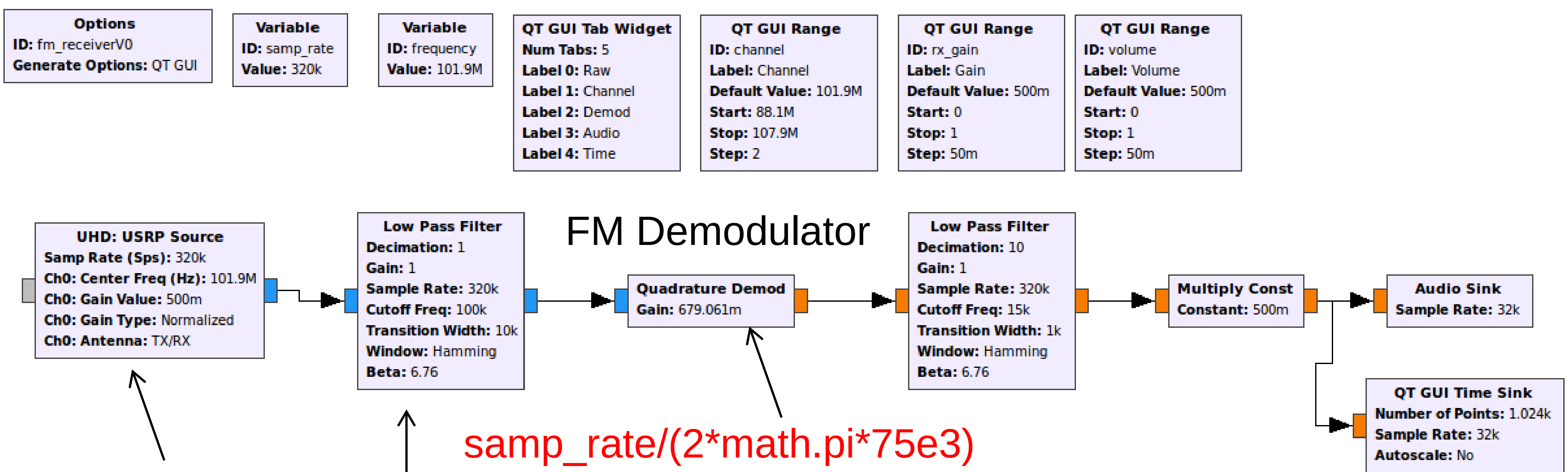
➢ **Radio amateur hobbyist application (AM/FM/ transmitter and receiver)**

Give the main condition related to the sampling rate that I shall verify in order to decode the received signal and hear a radio station?

What is the difference between **the Interpolation and the decimation**?

Flow graphs

Example of a Receiver's flow graph (.grc)



Digital Signal Receiver (or source of signal)

Low Pass Filter with a cutoff frequency at 100 KHz

Expression written in python

Audio sink

Flow graphs (Python description)

Python source code of the Receiver's flow graph
Generated using **gnuradio-companion**

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
##################################################
# GNU Radio Python Flow Graph
# Title: Fm Receiver
# Generated: Wed Dec 14 21:26:53 2016
##################################################

if __name__ == '__main__':
    import ctypes
    import sys
    if sys.platform.startswith('linux'):
        try:
            x11 = ctypes.cdll.LoadLibrary('libX11.so')
            x11.XInitThreads()
        except:
            print "Warning: failed to XInitThreads()"
```

```
        self.top_grid_layout.addWidget(self._channel_win, 1,0,1,1)
        self.blocks_multiply_const_vxx_0 = blocks.multiply_const_vff((volume, ))
        self.audio_sink_0 = audio.sink(samp_rate/10, '', True)
        self.analog_quadrature_demod_cf_0 = analog.quadrature_demod_cf(samp_rate/(2*math.pi*75e3))

        ##################################################
        # Connections
        ##################################################
        self.connect((self.analog_quadrature_demod_cf_0, 0), (self.low_pass_filter_1, 0))
        self.connect((self.analog_quadrature_demod_cf_0, 0), (self.qtgui_waterfall_sink_x_0,
```

Parameter defined in the quadrature_demod block

Flow graphs (Python description)

Example of a flow graph in python

```
#!/usr/bin/env python

#Import or include the libraries or packages
from gnuradio import gr
from gnuradio import audio, analog

#Object oriented programming with a definition of a new class
class my_top_block(gr.top_block):
    #Constructor of a class
    def __init__(self):
        gr.top_block.__init__(self)

        sample_rate = 32000
        ampl = 0.1

        src0 = analog.sig_source_f(sample_rate, analog.GR_SIN_WAVE, 350, ampl)
        src1 = analog.sig_source_f(sample_rate, analog.GR_SIN_WAVE, 440, ampl)
        dst = audio.sink(sample_rate, "")
        self.connect(src0, (dst, 0))
        self.connect(src1, (dst, 1))

#Main function as main in c language
#It is the first code to be excuted by the interpreter
if __name__ == '__main__':
    try:
        my_top_block().run()
    except [[KeyboardInterrupt]]:
        pass
```

Flow graphs (Python description)

What is the flow graph's fonction ?

```python
#!/usr/bin/env python

#Import or include the libraries or packages
from gnuradio import gr
from gnuradio import audio, analog

#Object oriented programming with a definition of a new class
class my_top_block(gr.top_block):
    #Constructor of a class
    def __init__(self):
        gr.top_block.__init__(self)

        sample_rate = 32000
        ampl = 0.1

        src0 = analog.sig_source_f(sample_rate, analog.GR_SIN_WAVE, 350, ampl)
        src1 = analog.sig_source_f(sample_rate, analog.GR_SIN_WAVE, 440, ampl)
        dst = audio.sink(sample_rate, "")
        self.connect(src0, (dst, 0))
        self.connect(src1, (dst, 1))

#Main function as main in c language
#It is the first code to be excuted by the interpreter
if __name__ == '__main__':
    try:
        my_top_block().run()
    except [[KeyboardInterrupt]]:
        pass
```

Import libraries as include in C language

Object Oriented Programming

Set parameters

Blocks creation

Blocks' interconnection

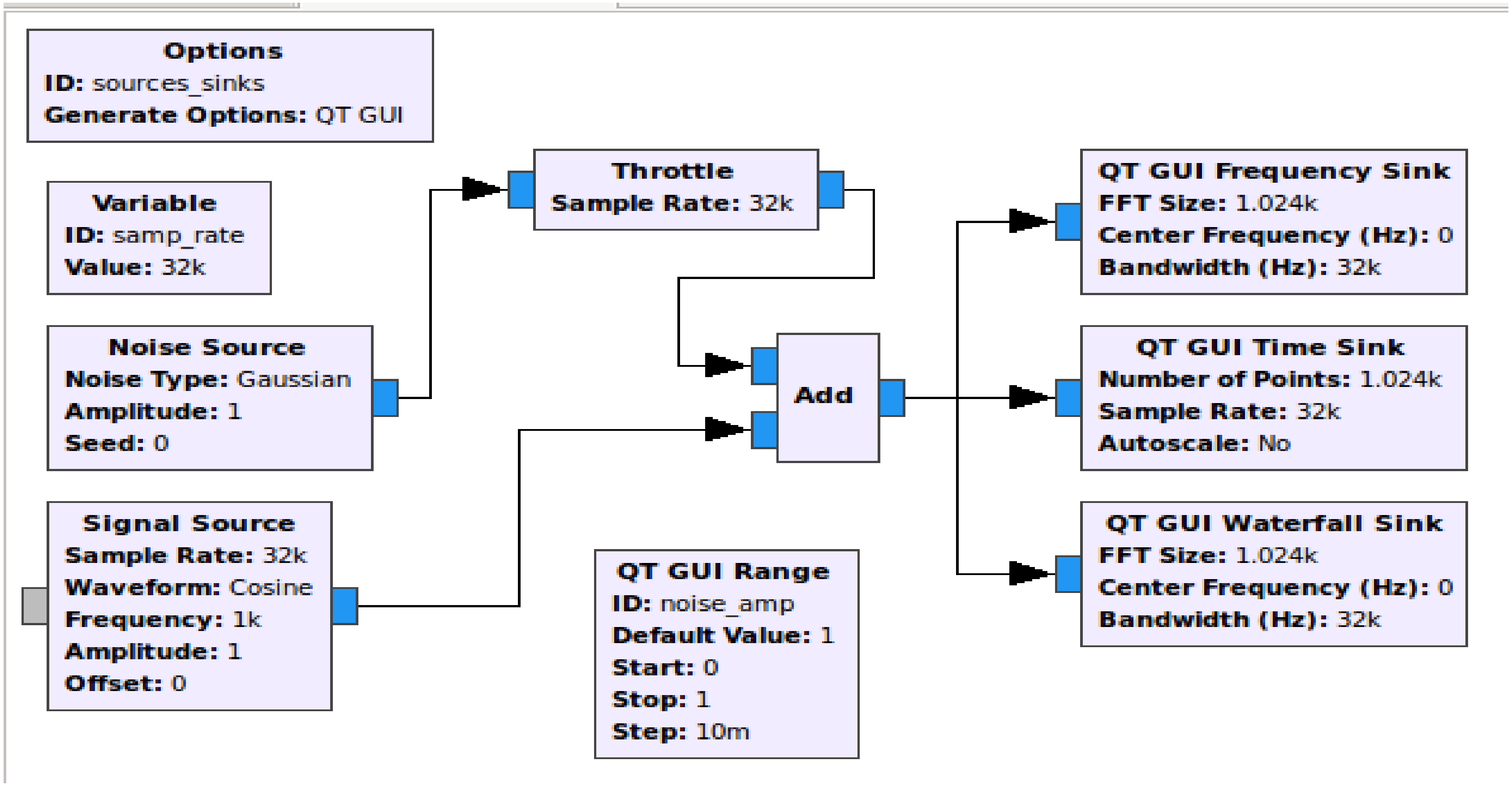
Run the flow graph

Display waveforms (Digital Oscilloscope)

Digital modulation and demodulation

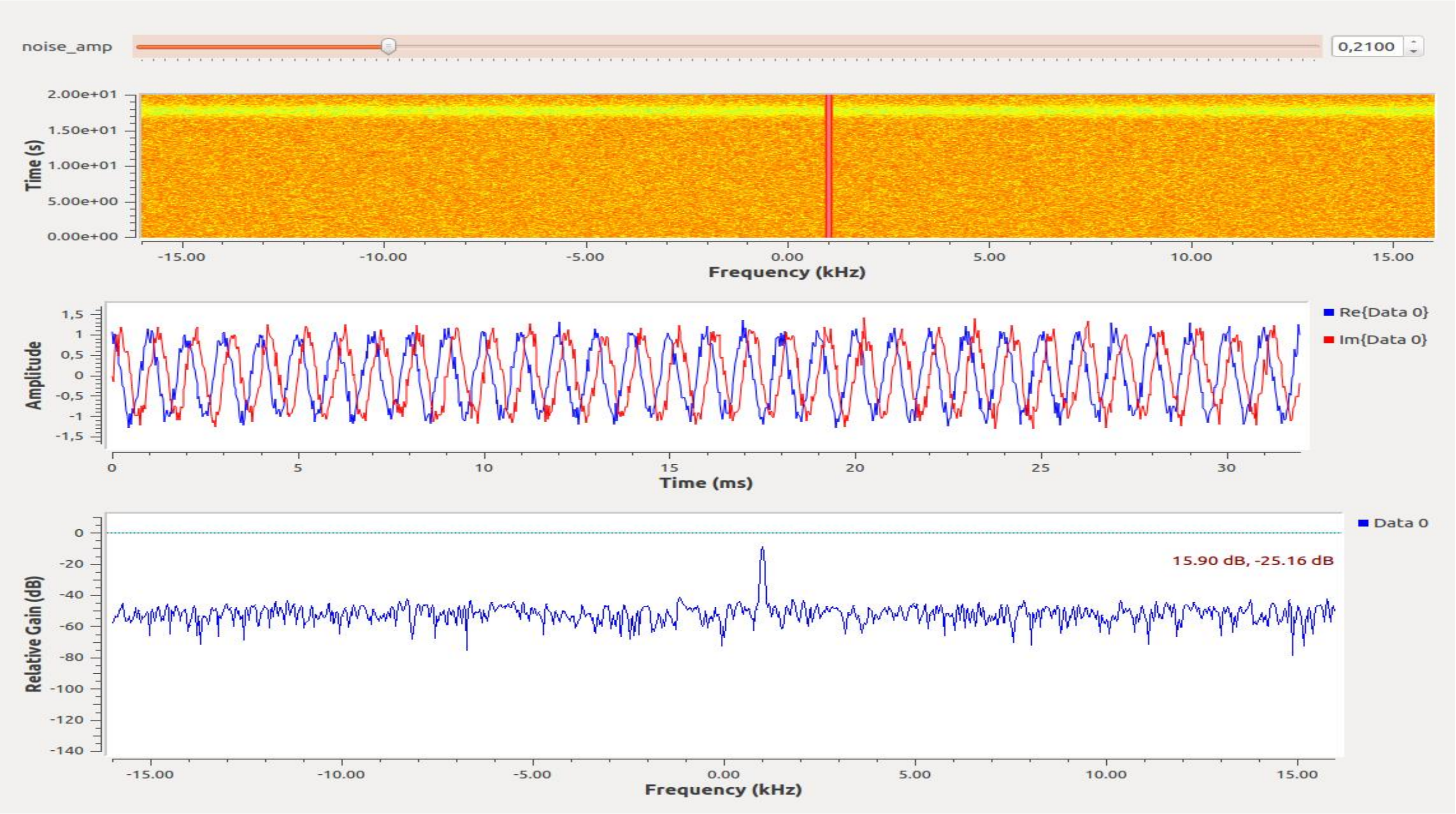
How to use different sources and how to display their waveforms?

Use of multiple sources to create a noisy sine wave and multiple views in different domains.



Display Waveforms (Digital Oscilloscope)

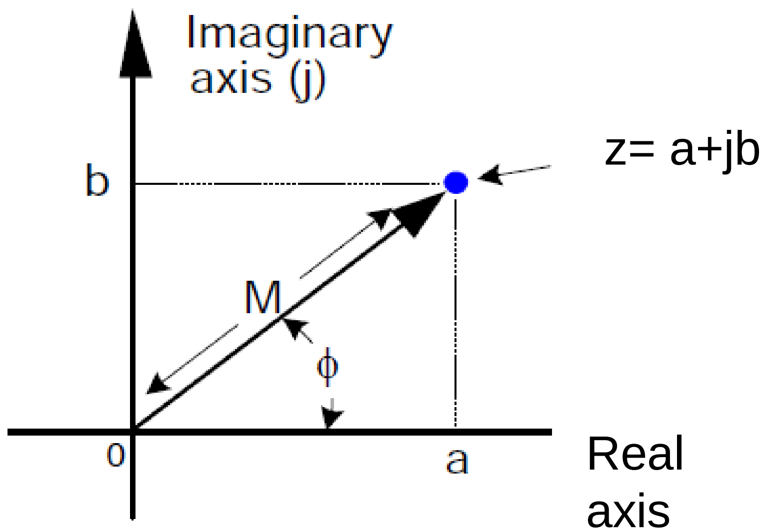
- Power Spectrum Density
- Time domain
- Waterfall spectrum usage during a period of time

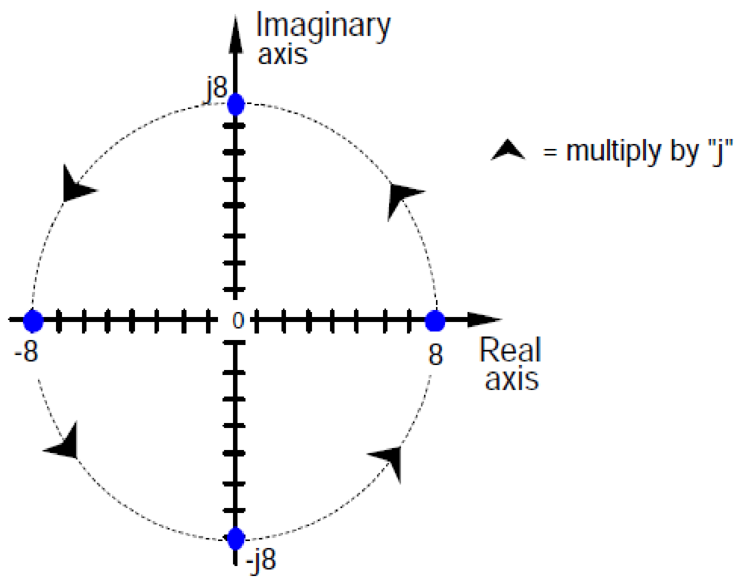


Complex samples

Why we use complex numbers?

Rectangular form: $z = a + jb$
Trigonometric form: $z = M[\cos(\phi) + j\sin(\phi)]$
Polar form : $z = M \exp(j\phi)$ % $M e^{j\varphi}$





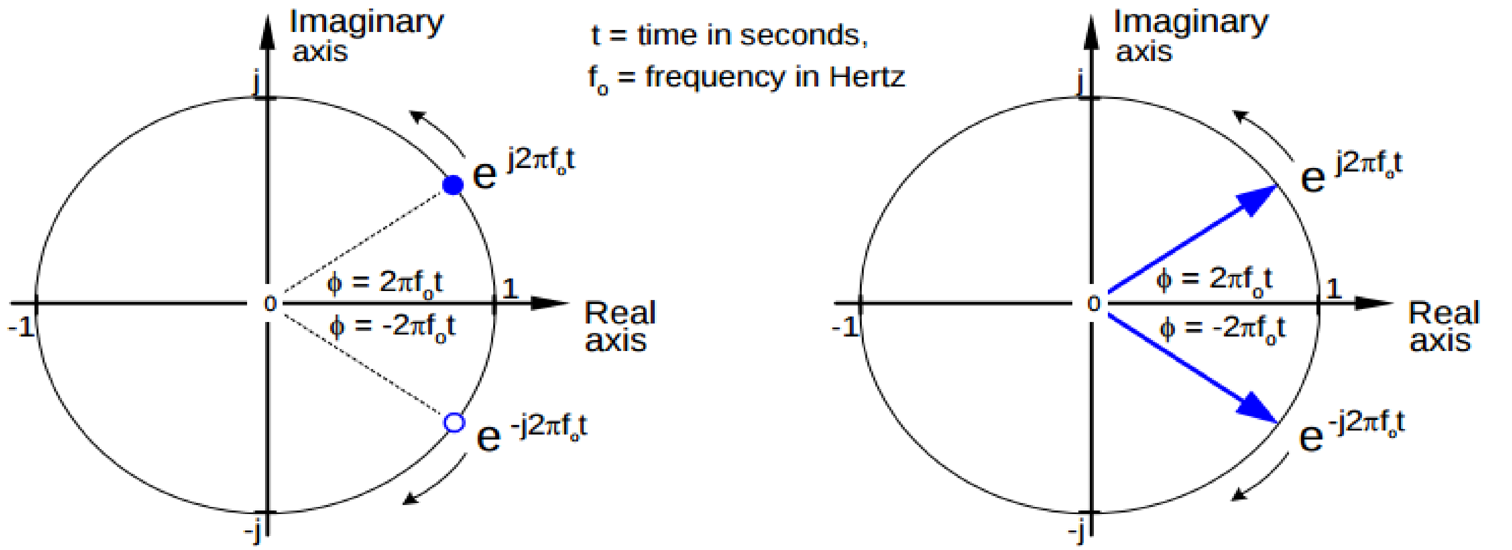
Complex samples

Complex numbers that are the function of time.

$$e^{j2\pi f_0 t}$$

Complex number whose magnitude M is one, and whose phase angle increase with time.
$2\pi f_0$: frequency in radians/second, which corresponds to a frequency f_0 cycles/second measured in Hertz.

Example : if **f_0= 2 Hz**, the complex point would rotate around the circle two times per second

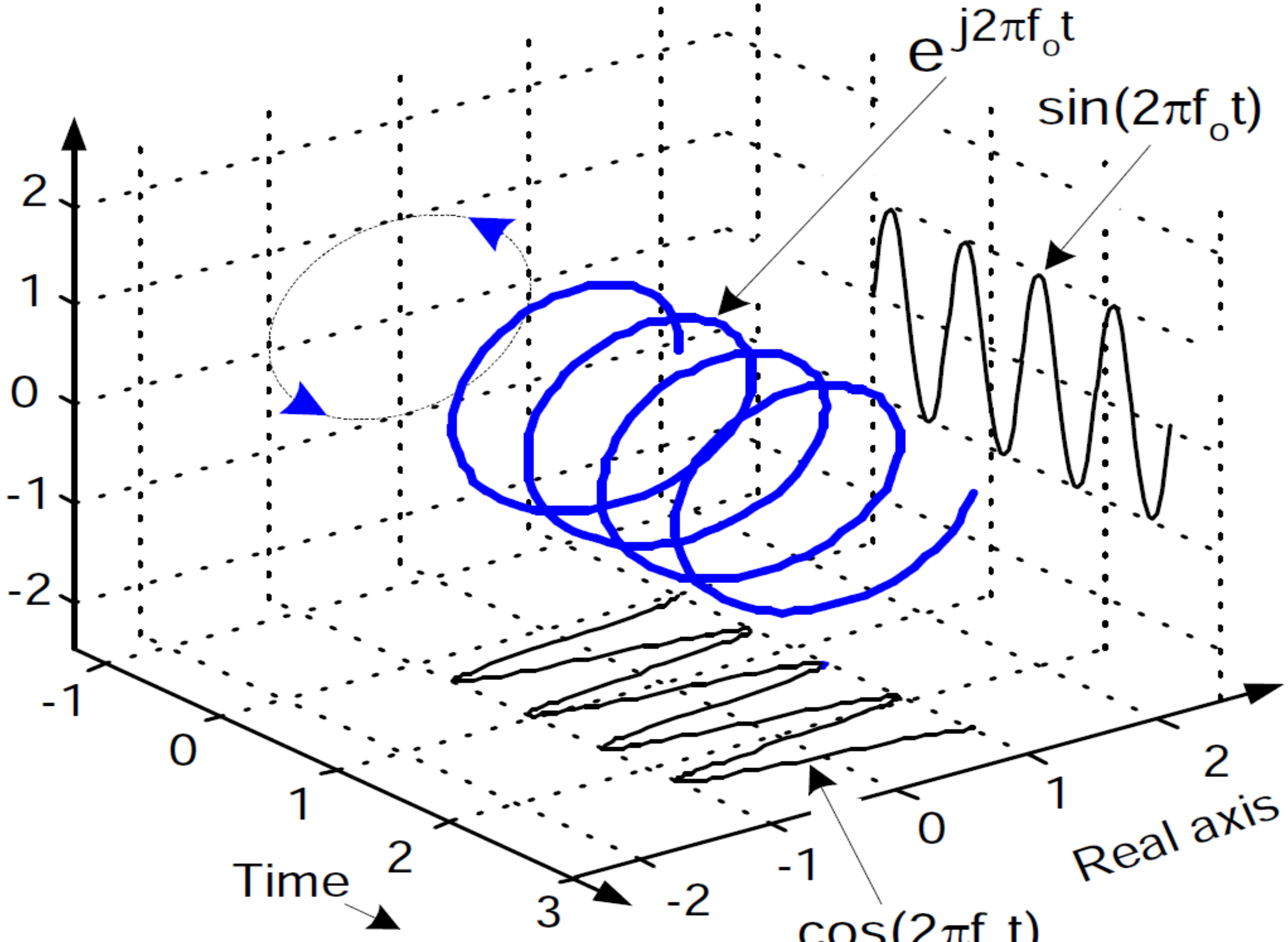


Complex samples

The real part of the spectrum is called the *In-phase*

The imaginary part is called the *quadrature* component

Imaginary



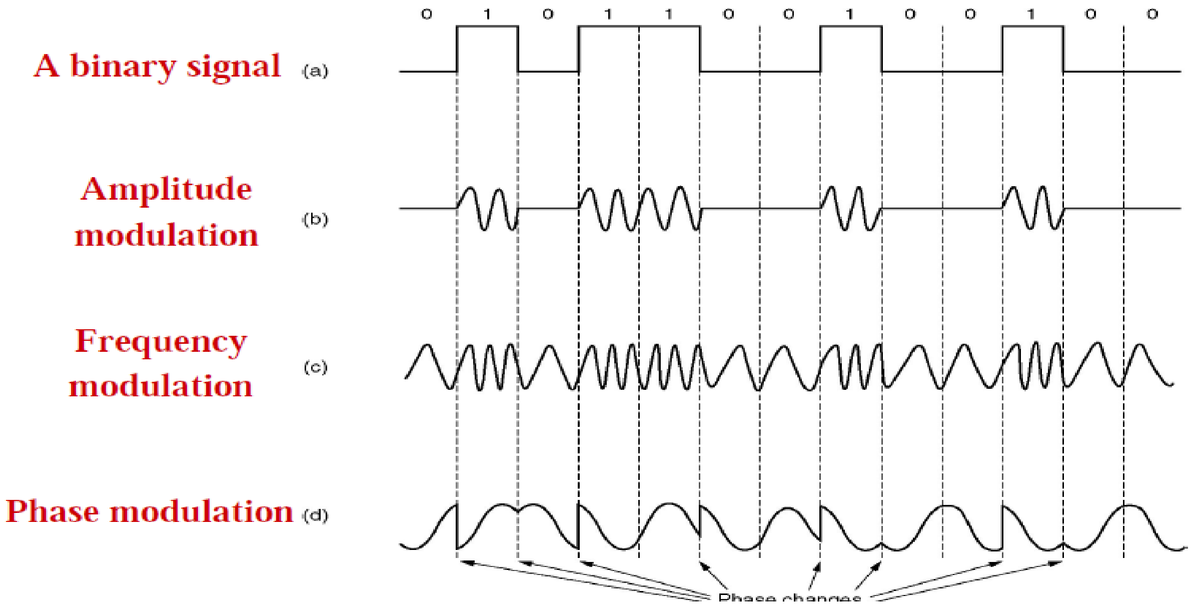
Complex samples

Complex numbers that are the function of time.

$z(t) = x(t) \cos(2\pi\, \mathbf{f}(t)\, t + \phi(t)) + j\, y(t) \sin(2\pi\, \mathbf{f}(t)\, t + \phi(t))$

$z(t) = M\mathbf{(t)} \exp(-j2\, \pi\, \mathbf{f(t)}t + \boldsymbol{\phi}\mathbf{(t)})$ % Euler's identities

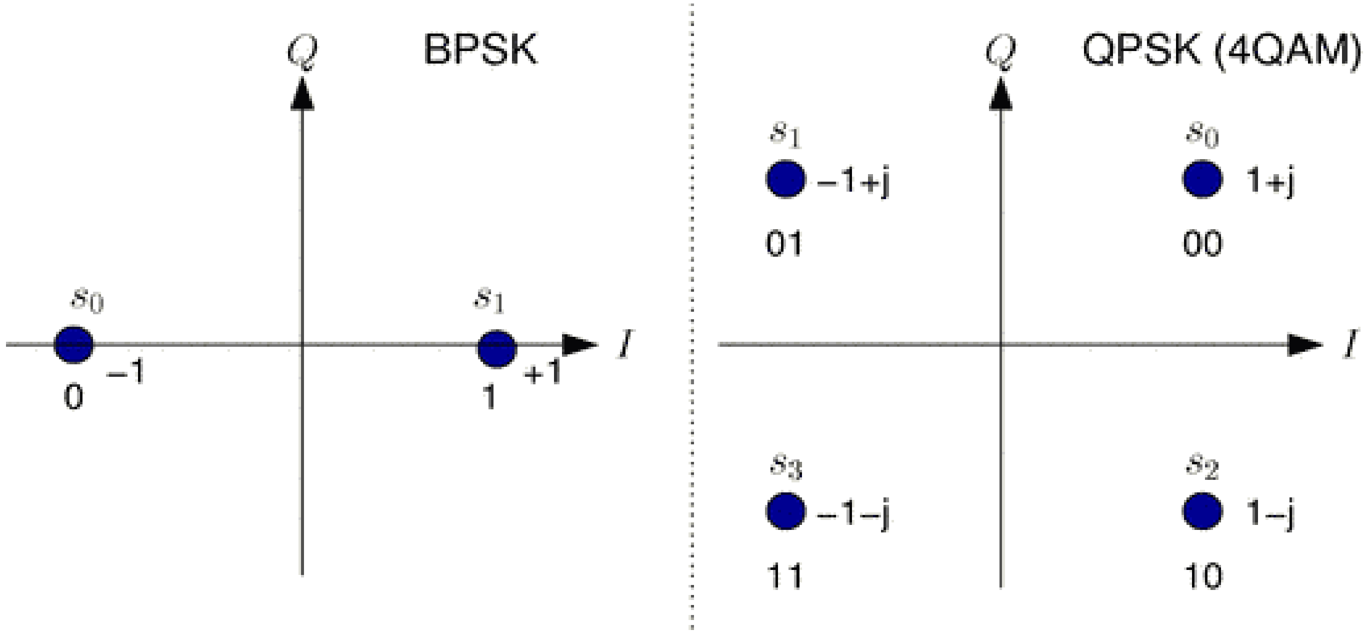
Data can be encoded in M**(t)**, f**(t)** and **ϕ(t) or their combination.**



Wireless Networks and Modulations

Modulation used by IEEE 802.15.4 (LoRa, ZigBee) transmitter :

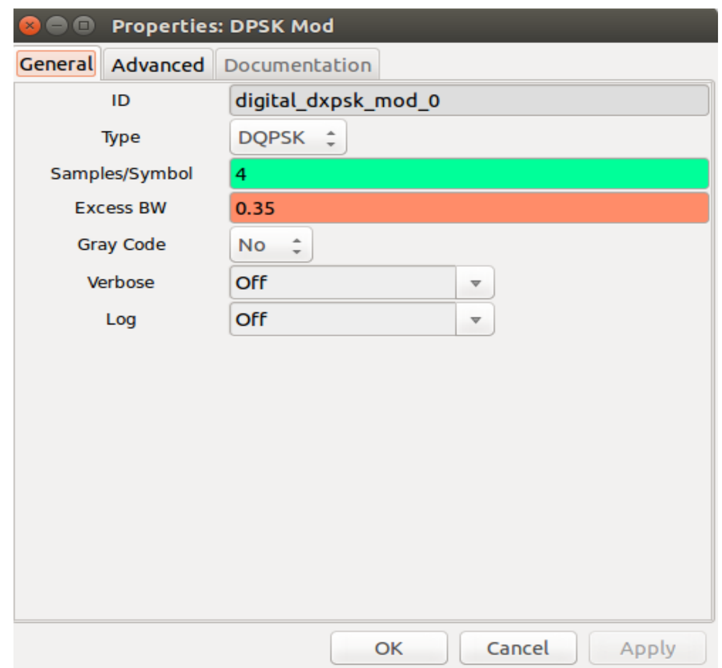
Differential Binary Phase Shift Keying (DBPSK modulation)
Offset Phase Shift Keying (OQPSK modulation)



Wireless Networks and Modulations

What is a bit rate (bits/s) of your transmitter **(DQPSK)** with a sampling rate equal to 1 M sample per second (**SPs =1 MS/s**)?

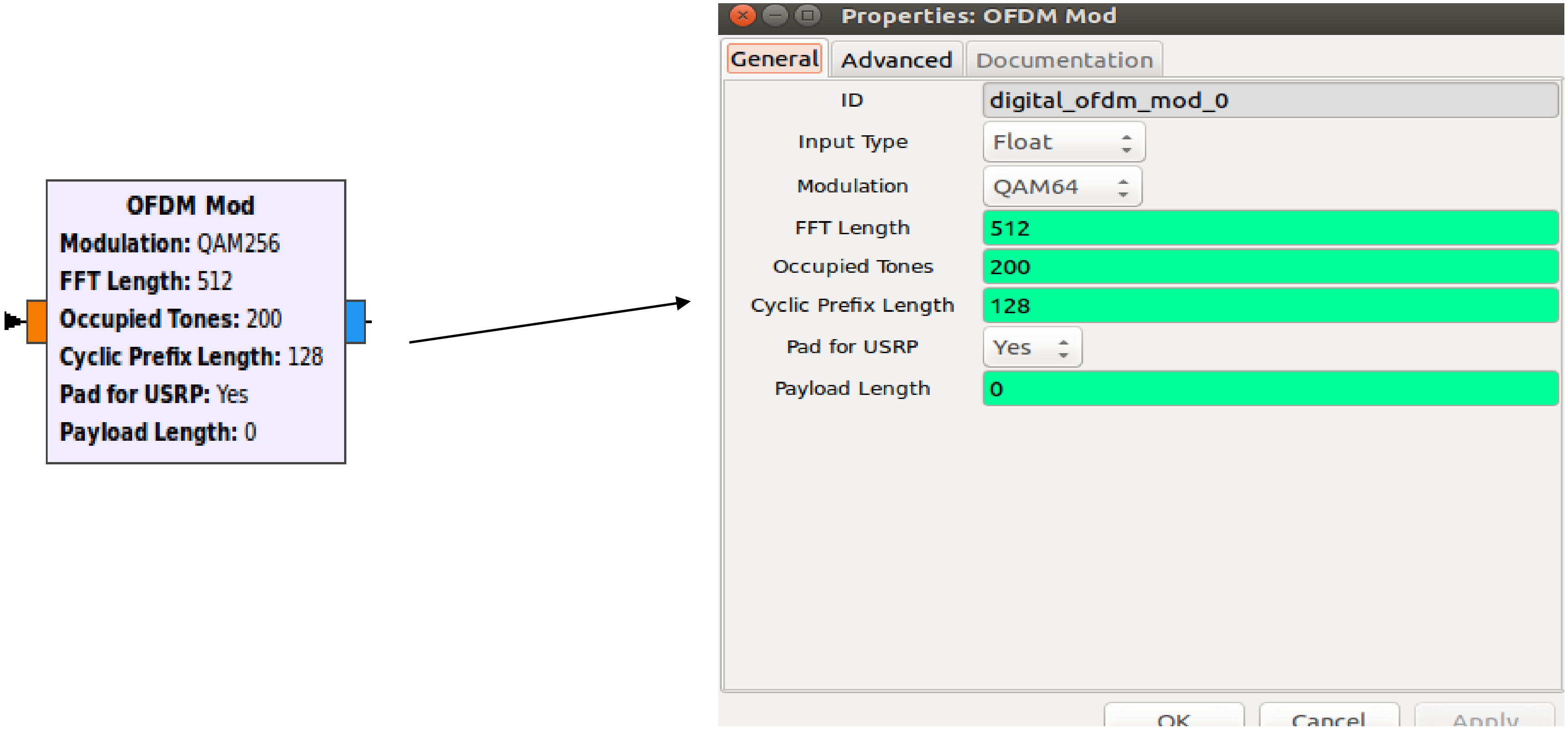
Each symbol needs 4 samples , so we can encode only **250 Symbles**. Since my modulator is DQPSK, we obtain only **500 kb/s.**



Wireless Networks and Modulations

Modulation used by IEEE 802.11 or Wifi transmitter:

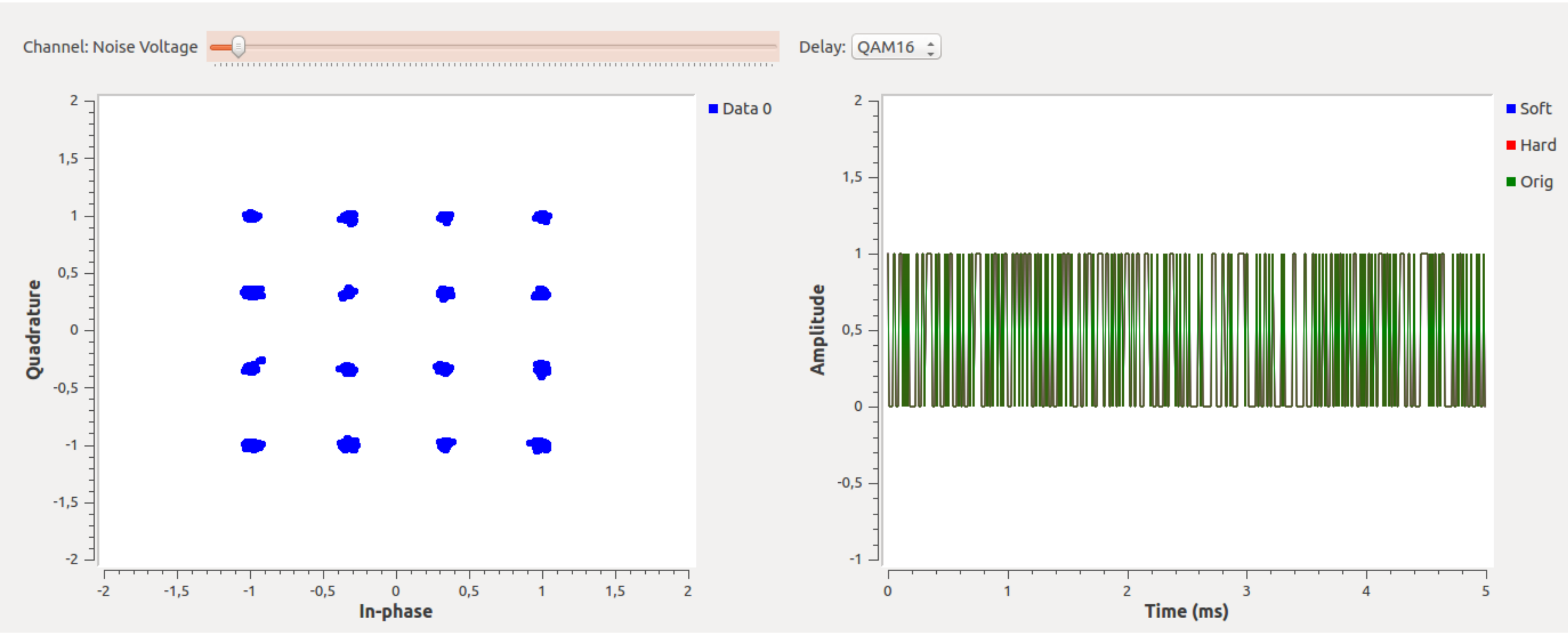
OFDM **O**rthogonal **F**requency **D**ivision **M**ultiplexing
For each carrier a **Q**uadrature **A**mplitude **M**odulation (QAM).
From Binary Phase Shift Keying to QAM-256



Wireless Networks and Modulations

IEEE 802.11 or Wifi:

OFDM Orthogonal Frequency Division Multiplexing
For each carrier a Quadrature Amplitude Modulation (QAM)
From BPSK and QPSK to QAM-256



References

[KF10] S.Katz, J. Flynn, *Senior Design and Graduate Projects Using Software Defined Radio (SDR),* 2010.

[Hay03] N. Hayes, *Software Communication Architecture*, Report of the Boeing Company, 2003

[ECE4305] A M. Wyglinski, *Software Defined Radio Systems and analysis,* Courses given in IWP. Indiana University, US. 2014

[Ettus] Ettus Research, https://kb.ettus.com/B200/B210/B200mini/B205mini

[Zit15] R. Zitouni, Software Defined Radio for Cognitive Wireless Sensor Networks : A reconfigurable IEEE 802.15.4 standard, University of Paris-Est. France, 2015

[Wiki] Wiki of GNU Radio community http://gnuradio.org/redmine/projects/gnuradio/wiki/Guided_Tutorials

[Tom] Tutorials of Tom Rondeau: http://www.trondeau.com/gr-tutorial/

[Lyo10] Richard Lyons, Quadrature Signals: Complex, but not complicated.

Richard Lyons, Understanding Digital Signal Processing, 3rd Edition, **ISBN-13:** 978-0137027415, 2010