Introductory Tutorial for SDR and GNU Radio Beginners

GNU Radio Conference 2023

Murat Sever

Outline

- About me
- About tutorial
- GNU Radio
- Lab: Digital Signal Processing (DSP)
- Lab: Software Defined Radio (SDR)
- Lab: Wide Band Frequency Modulation (WBFM)

About me

Part-Time Lecturer at TOBB ETU, Ankara, Turkey

- ELE361L Course / Telecom Laboratory
    - Summer 2021
    - Fall 2022
    - *Fall 2023*

Project Owner & Manager

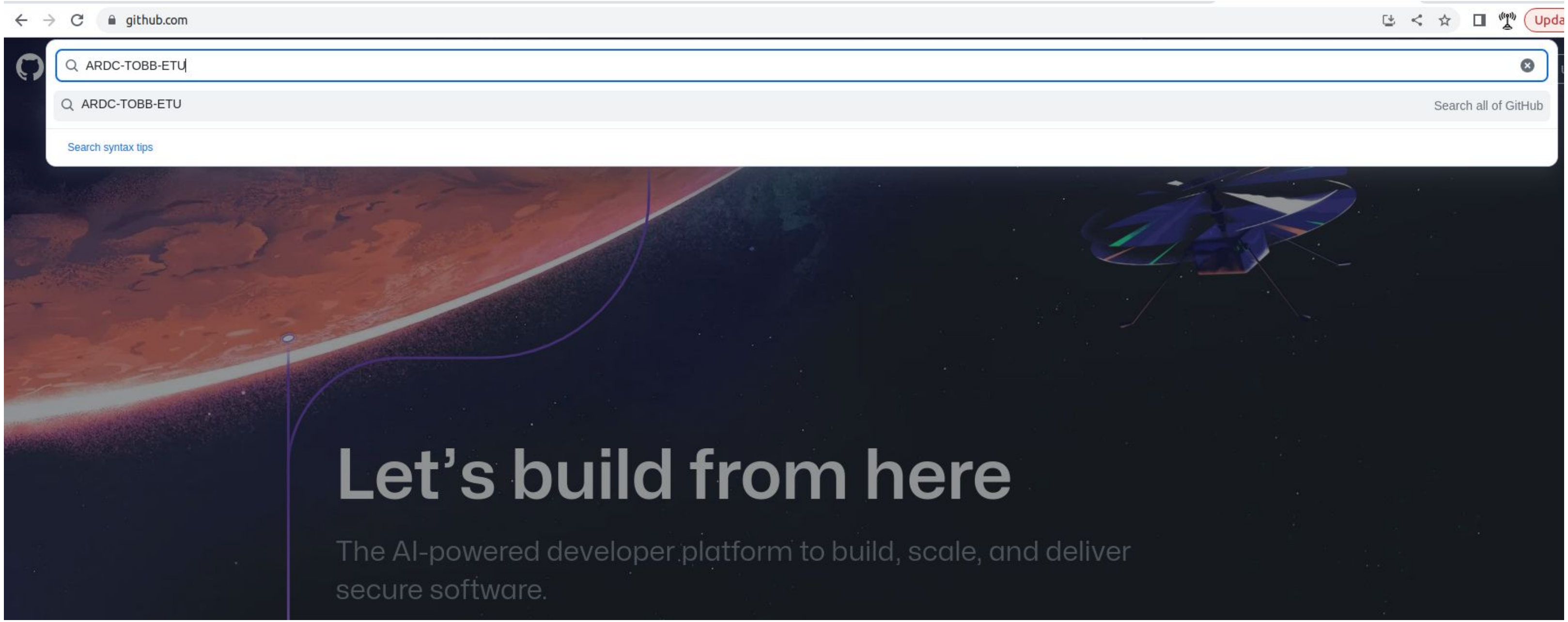
- SDR-Powered Education

About tutorial

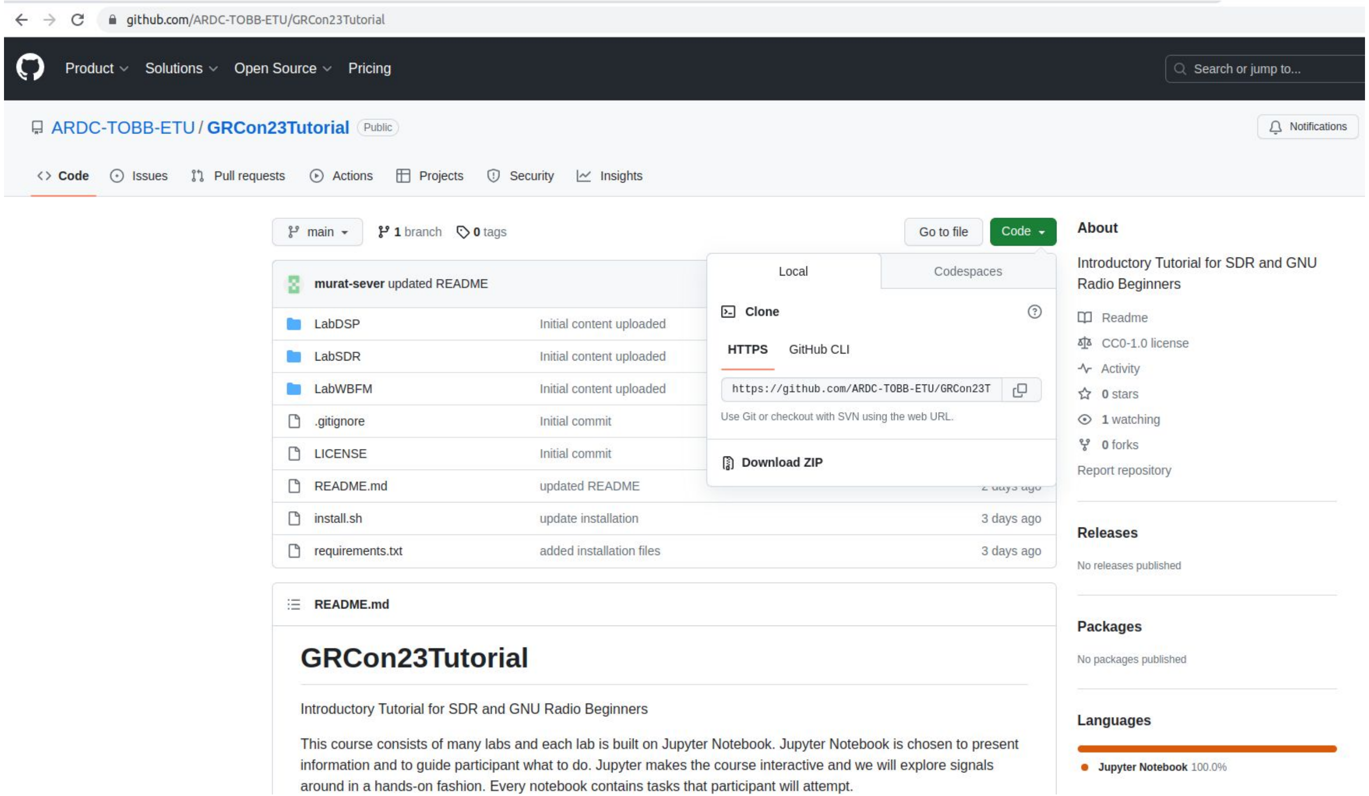
- Introduces fundamental DSP concepts and GNU Radio to new users
- Consists of the following lab modules based on Jupyter Notebooks
  - Lab DSP
  - Lab SDR
  - Lab WBFM
- We will use GNU Radio for
  - Exploring signals in simulation mode
  - Sound processing
  - Spectrum watching with RTL-SDR
  - Broadcast FM demodulation

Get the labs (if you haven't already)

- Labs available @ GitHub
  - https://github.com/ARDC-TOBB-ETU/GRCon23Tutorial



Download or clone



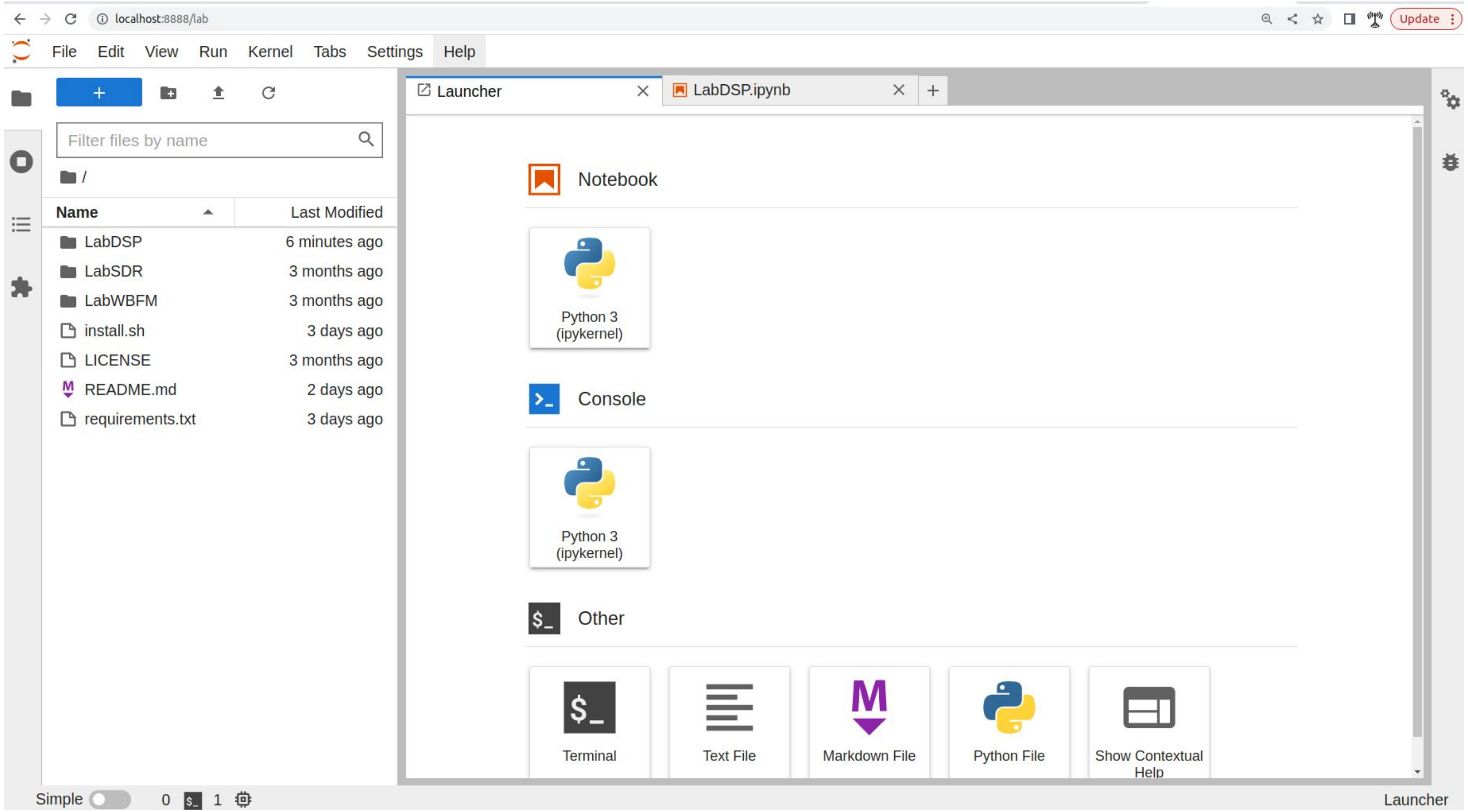
Use README to install

- Linux/Mac
  - Run ./install.sh
- Windows
  - Install miniforge
  - Create a new environment
    - conda config --append channels conda-forge
    - conda create --name GRCon23 --file requirements.txt

Opening Jupyter Notebooks/GNU Radio

- Linux/Mac
  - Run `source "${HOME}/conda/etc/profile.d/conda.sh"`
  - Activate the environment `conda activate GRCon23`
  - Run `jupyter-lab`
  - Run `gnuradio-companion`
- Windows
  - Open a miniforge prompt
  - Activate the environment `conda activate GRCon23`
  - Run `jupyter-lab`
  - Run `gnuradio-companion`

Jupyter Notebook



Outline

- About me
- About tutorial
- **GNU Radio (slides from a previous presentation)**
- Lab: Digital Signal Processing (DSP)
- Lab: Software Defined Radio (SDR)
- Lab: Wide Band Frequency Modulation (WBFM)

GNU Radio is…

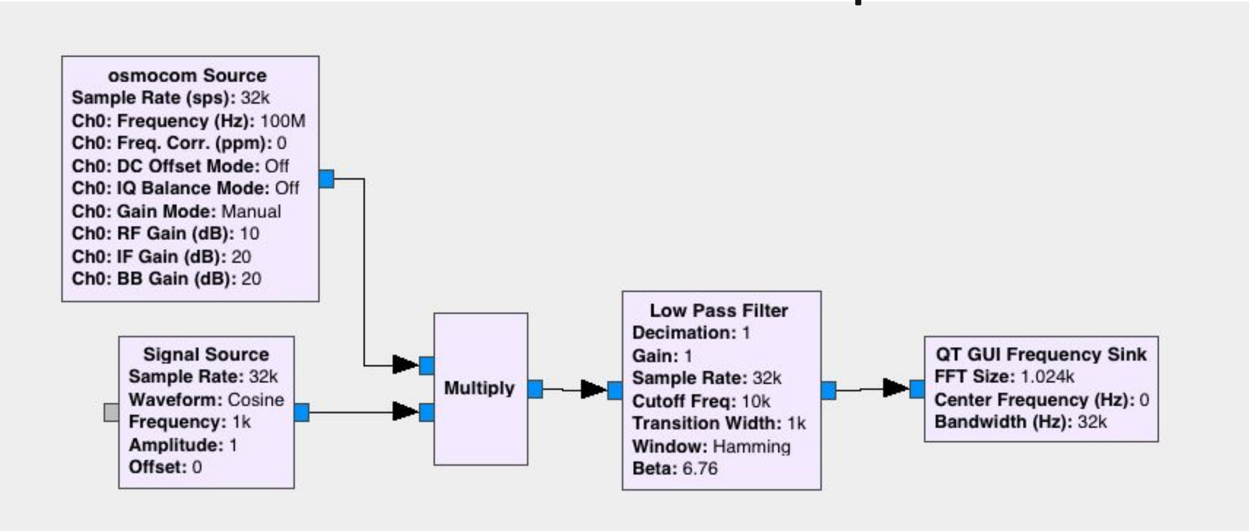
- A signal processing library
- Designed for real-time
- The software part of an SDR
- Not a radio application
- The tool to **build your own** transceivers
- **FOSS**: Free and Open Source Software

GNU Radio

- Open-source framework for SDR and signal processing
- Founded by Eric Blossom in 2001
- Block-based dataflow architecture
- Each block runs in its own thread
- Data flows through a graph called a Flowgraph
- Blocks are nodes in a Flowgraph, and perform operations and signal processing
- Signals normalized between -1.0 and +1.0
- Similar in concept to MathWorks SimulinkTM
- Running C++ and Python under-the-hood
- Can write code directly, or use the GNU Radio Companion (GRC) graphical tool

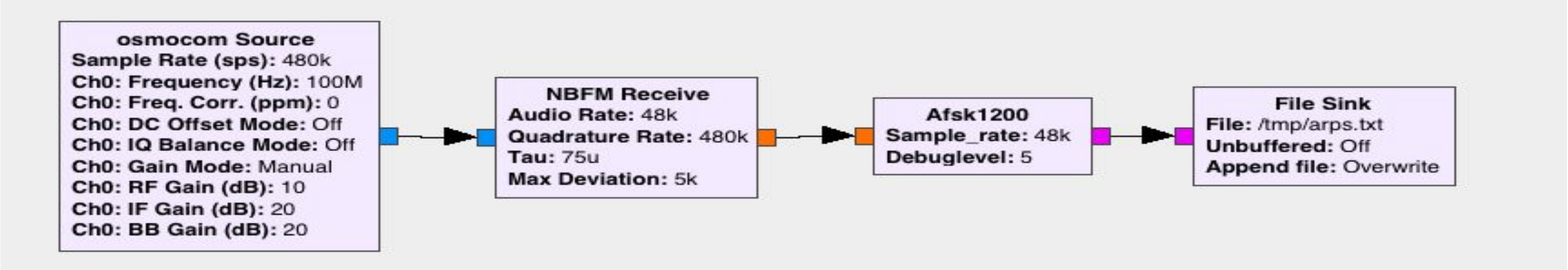
Basic Concept: Flow Graph

- Transceivers are implemented as *flow graphs*
- Similar to Simulink / schematics
- Define structure and parameters of *blocks*



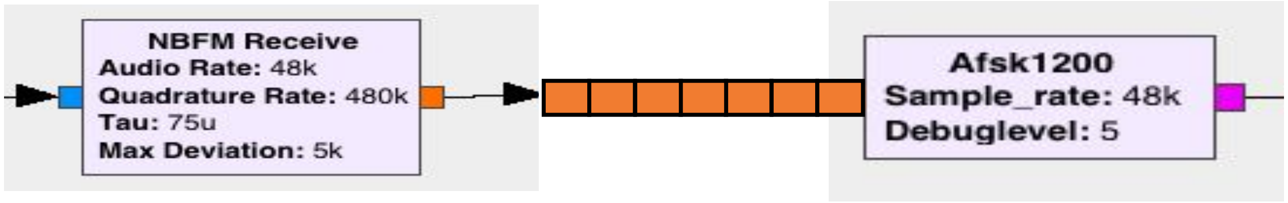
Basic Concept: Block

- Written in C++ or Python
- Implement one logical step
- Each block run in separate thread

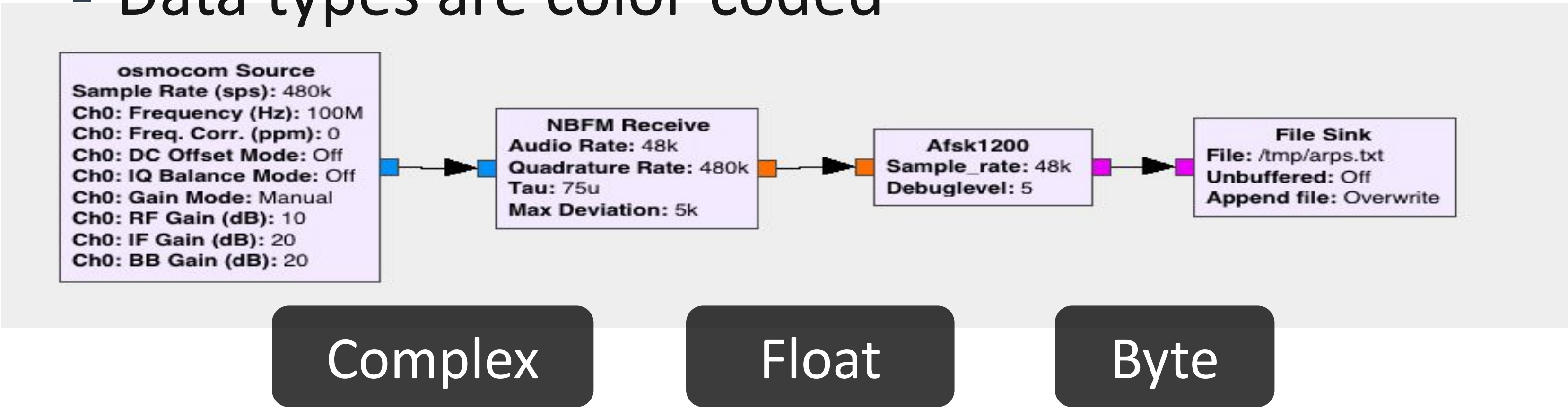


Data Streams

- Samples are buffered

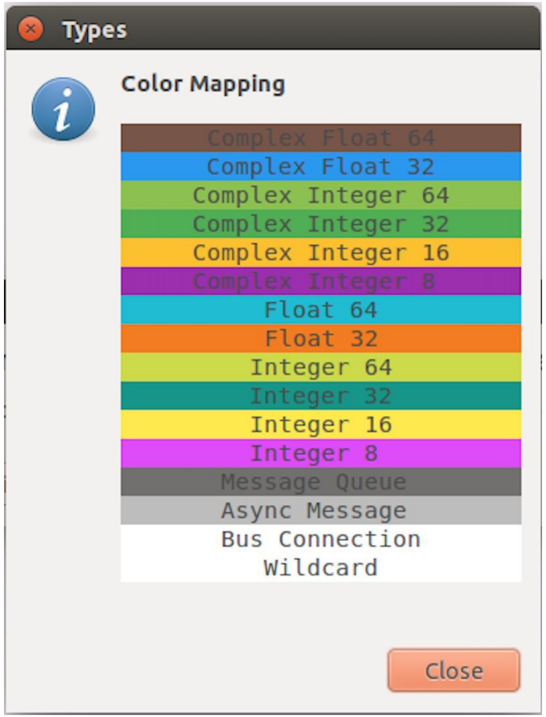


- Data types are color-coded

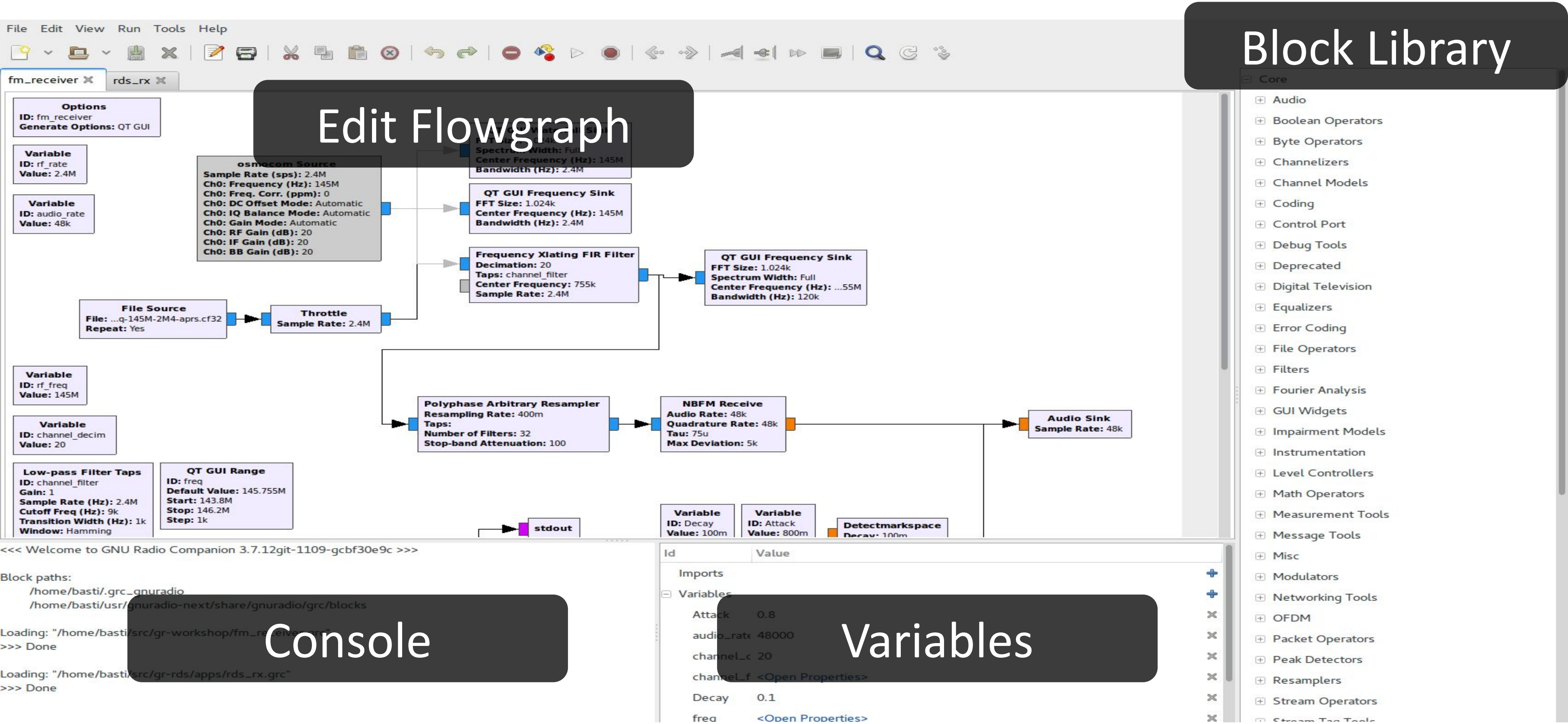


Color Types

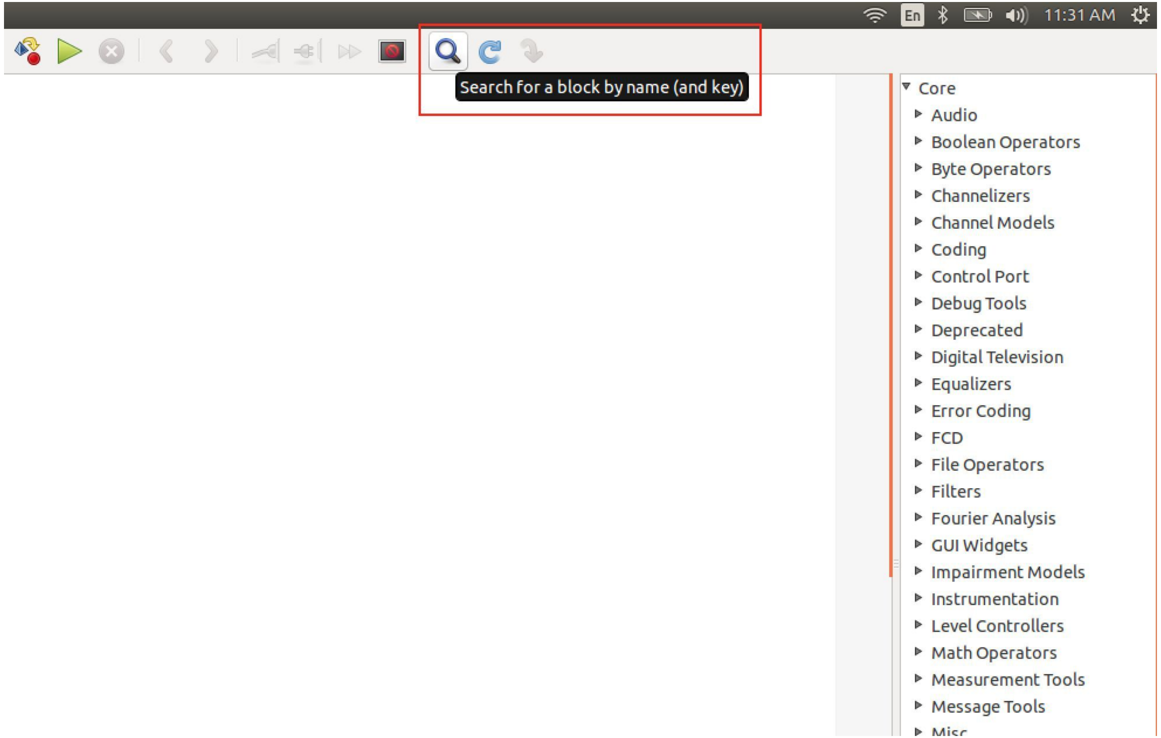
Click on menu item Help->Types



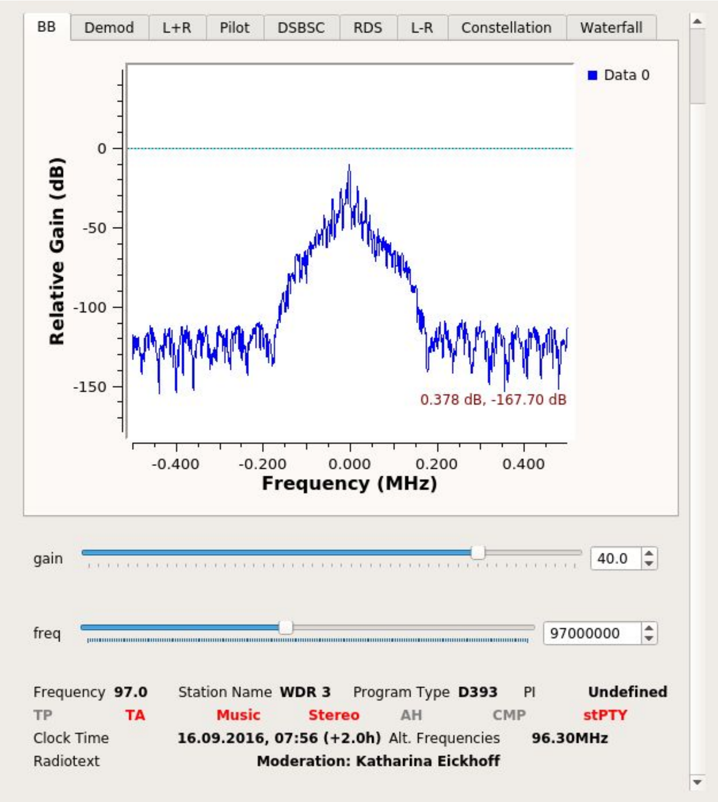
GNU Radio Companion

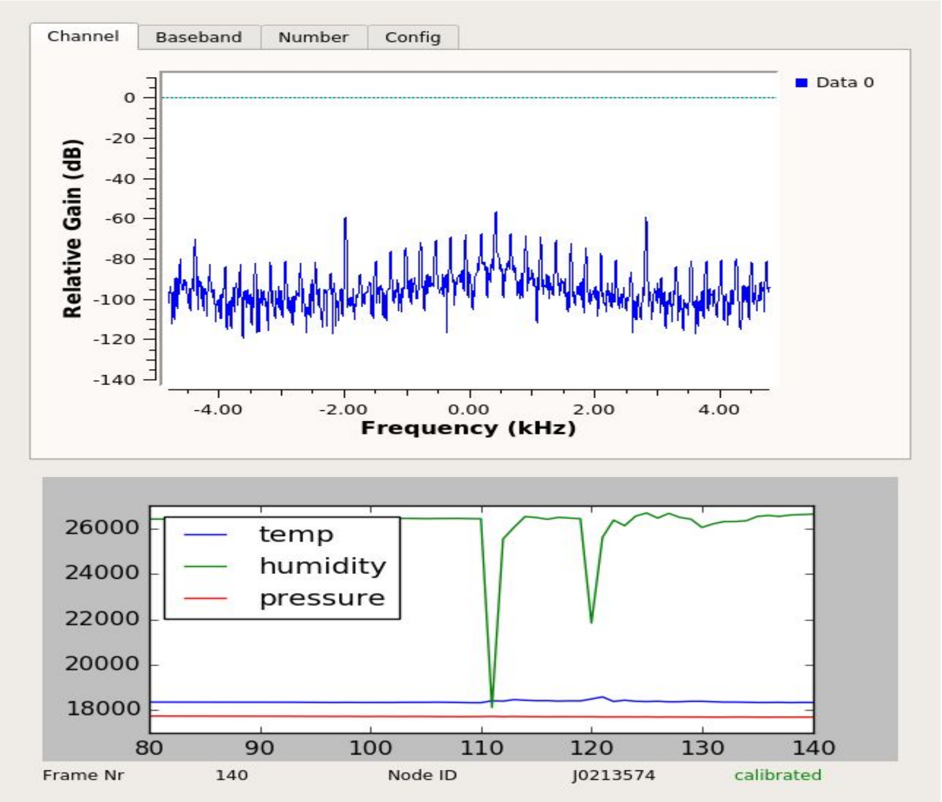


Search Blocks

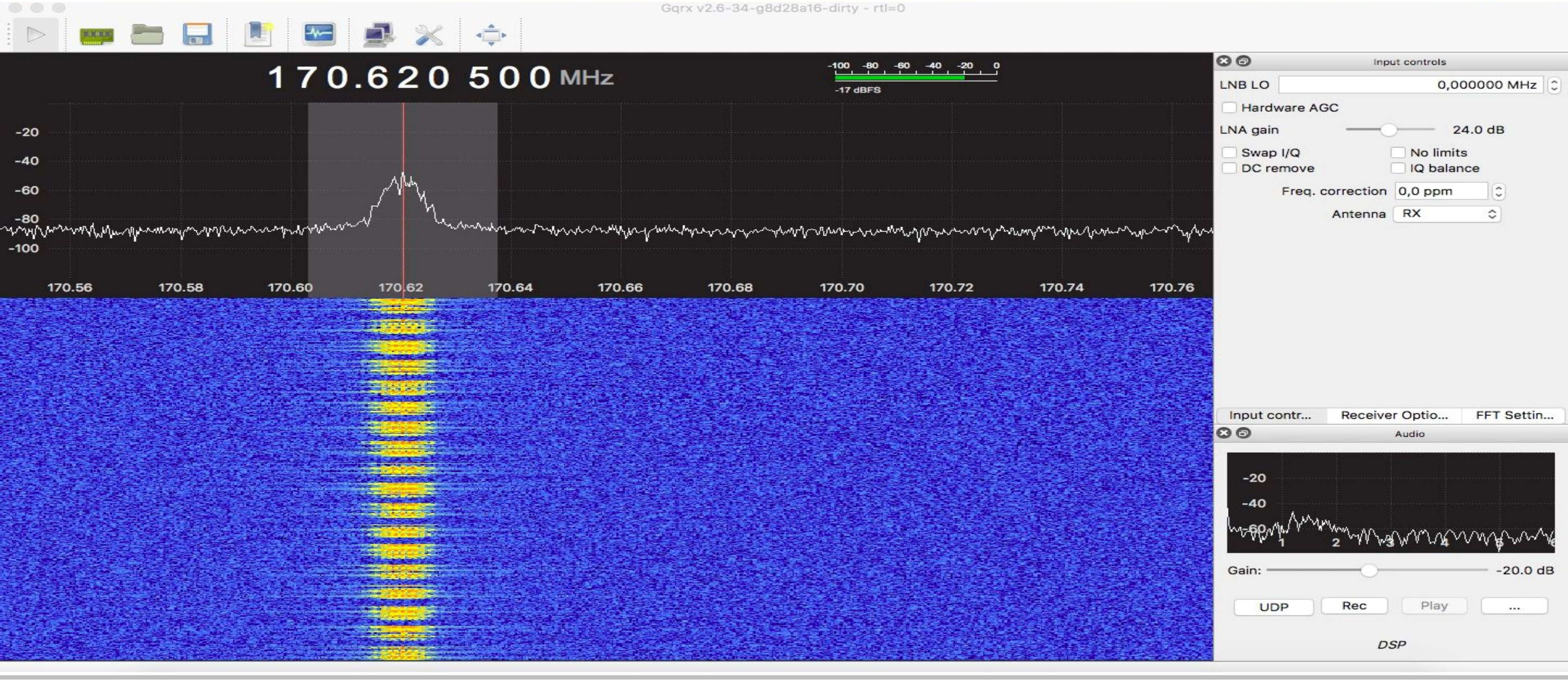


GUI Output and Instrumentation



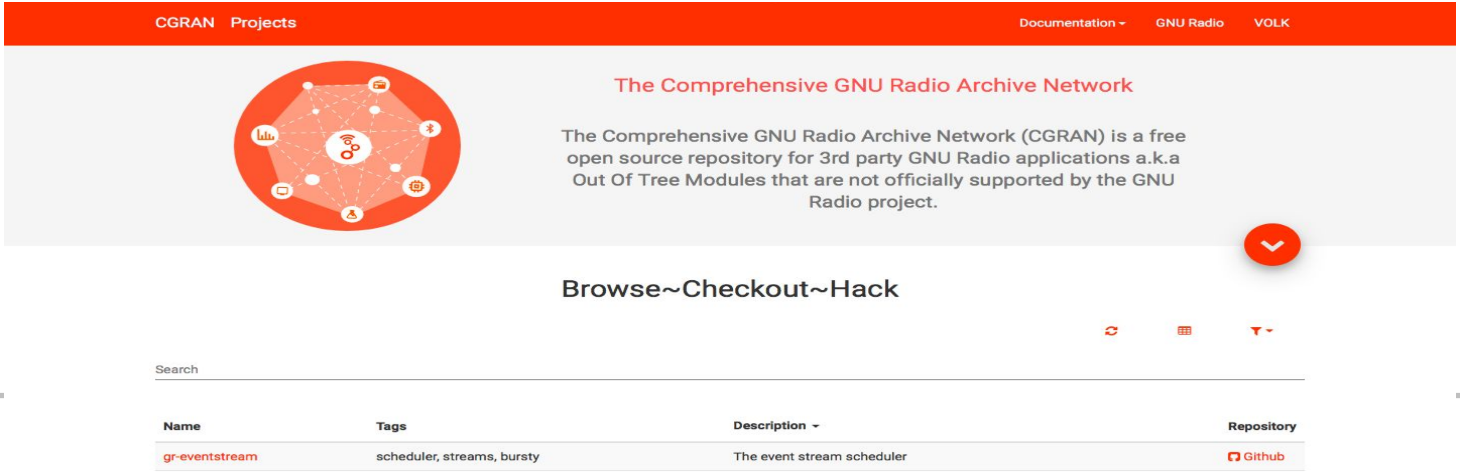


GQRX - a GNU Radio Application

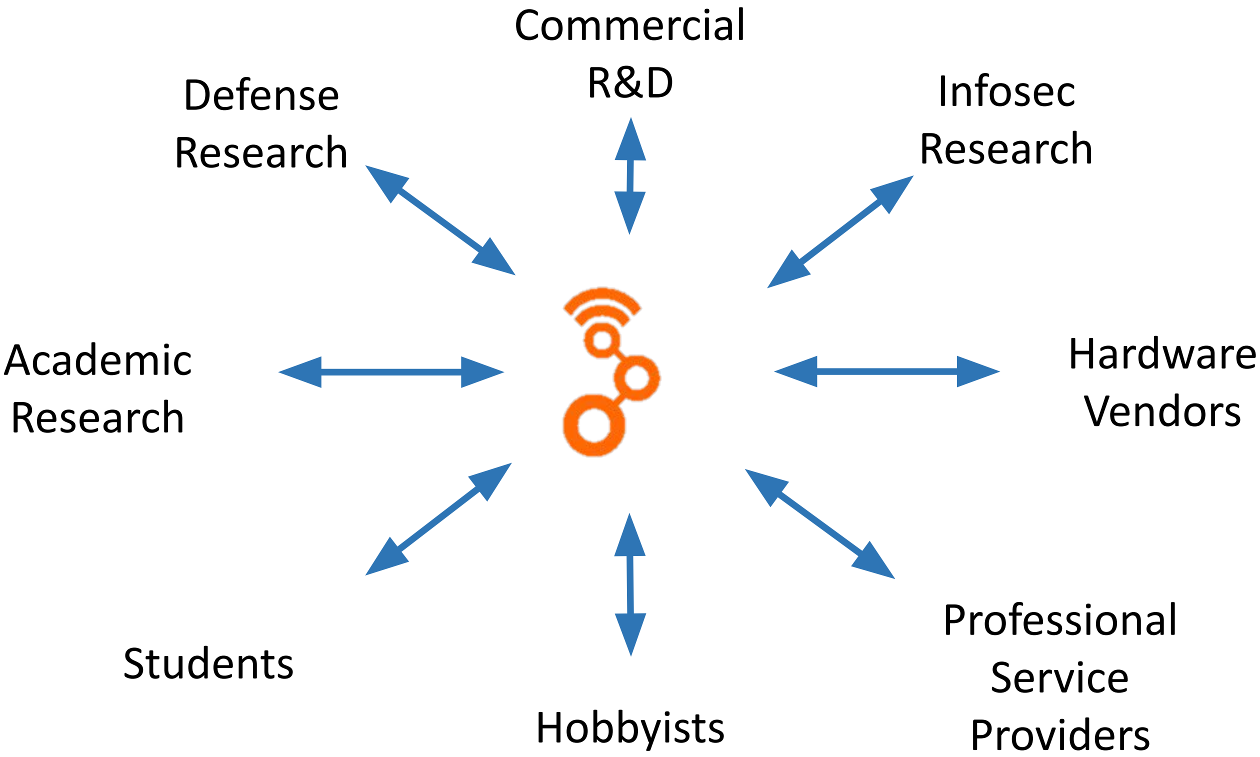


Out Of Tree Modules

- GNU Radio can be extended with OOTs
- OOTs cover more specific functionality
- There is a large number available
- CGRAN is our central database



GNU Radio is used by



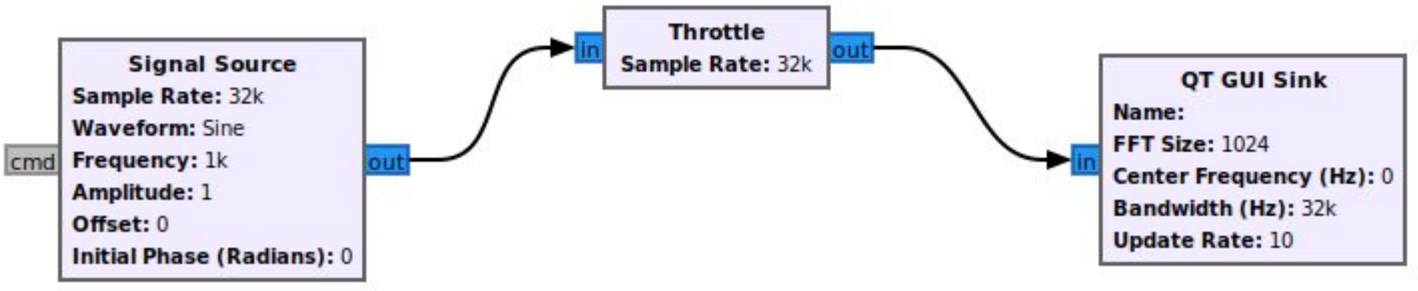
GNU Radio is an Ecosystem

- Active Open Source community since 2001
- PyBombs, OOTs
- GRCon since 2011
- GNU Radio Foundation
- FOSDEM SDR DevRoom
- GSoC, SoCIS, R&S Competition, SDR Academy
- GNU Radio Europe

Outline

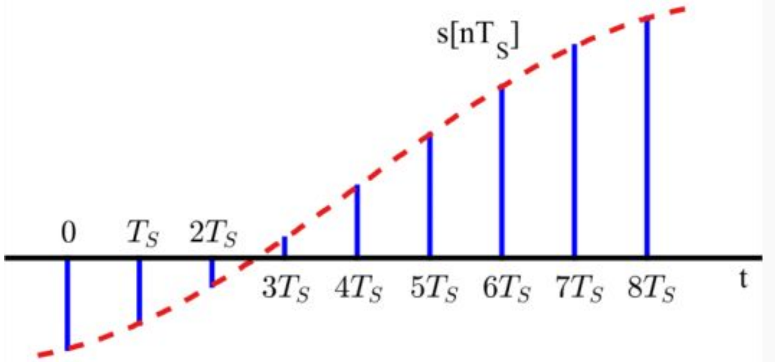
- About me
- About tutorial
- GNU Radio
- **Lab: Digital Signal Processing (DSP)**
- Lab: Software Defined Radio (SDR)
- Lab: Wide Band Frequency Modulation (WBFM)

Exploration of Signals in Frequency Domain



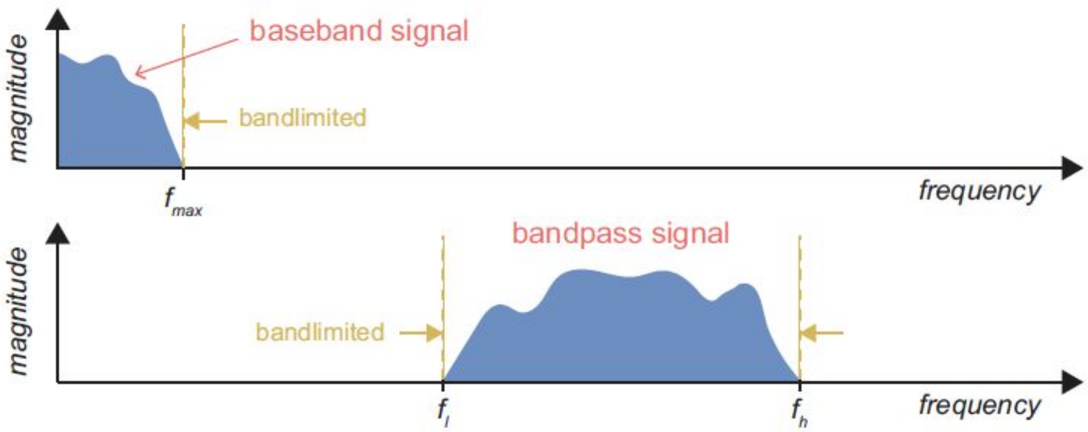
Sampling

- Communication signals are continuous-time
- We (ADCs) take samples at regular times
- Ts is sampling period
- Fs is sampling frequency



Baseband & Bandpass

- Baseband: Information signal
- Bandpass: Communication signal

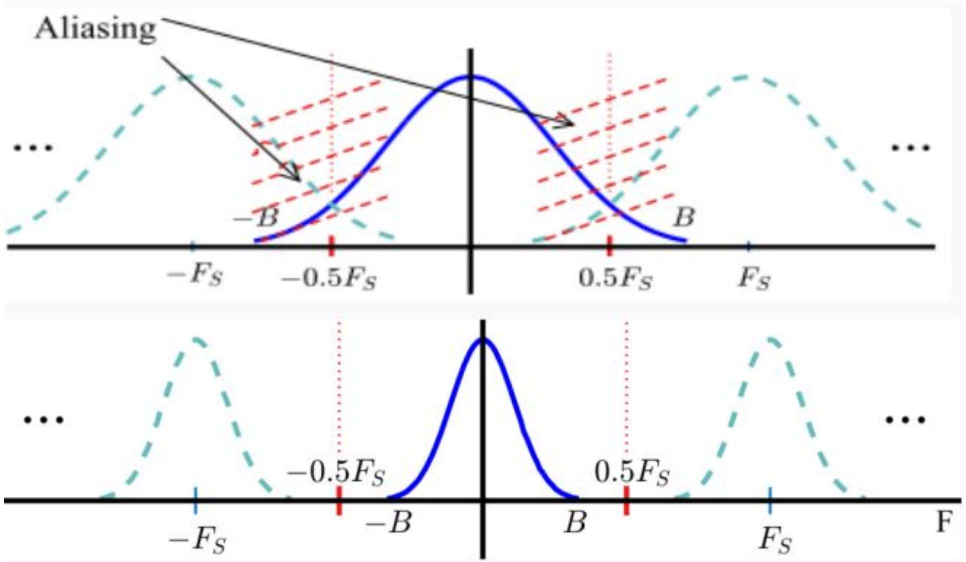


Nyquist Sampling Theorem

- The **Nyquist Sampling Theorem** states that a baseband, bandlimited signal must be sampled at **greater than twice the bandwidth** present in the signal, i.e.
  - fs > 2 * fmax
  - fs > 2 * (f_high - f_low)

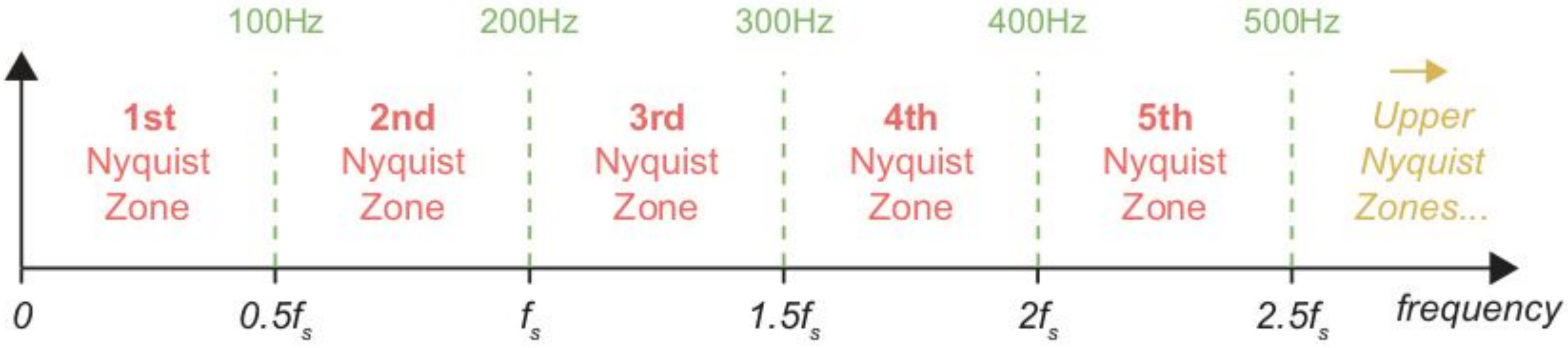
Aliasing

- Sampling produces aliases (spectral replicas)
- To prevent aliasing Fs must satisfy Fs > 2 * BW

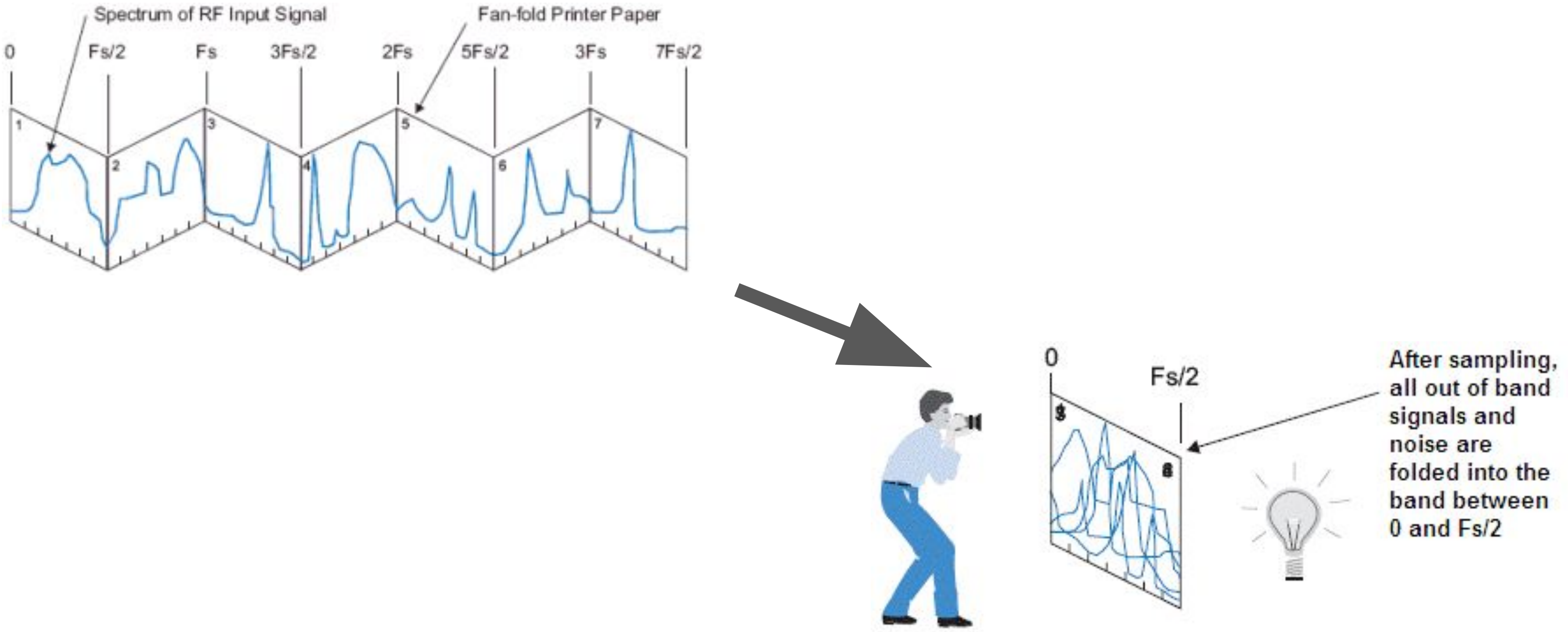


Nyquist Zones

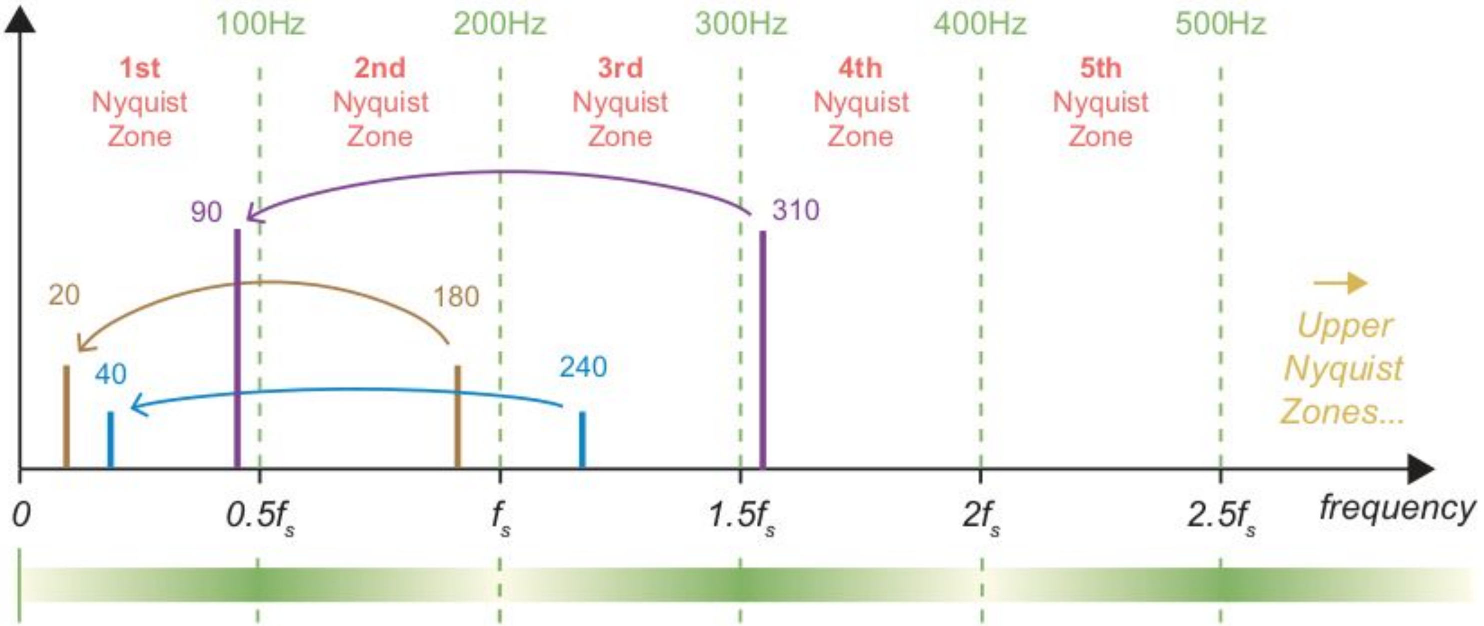
- Partitions of bandwidth 0.5f s in the frequency domain
- Any signal components present in higher Nyquist Zones are ‘folded’ down into the 1st Nyquist Zone as a result of aliasing



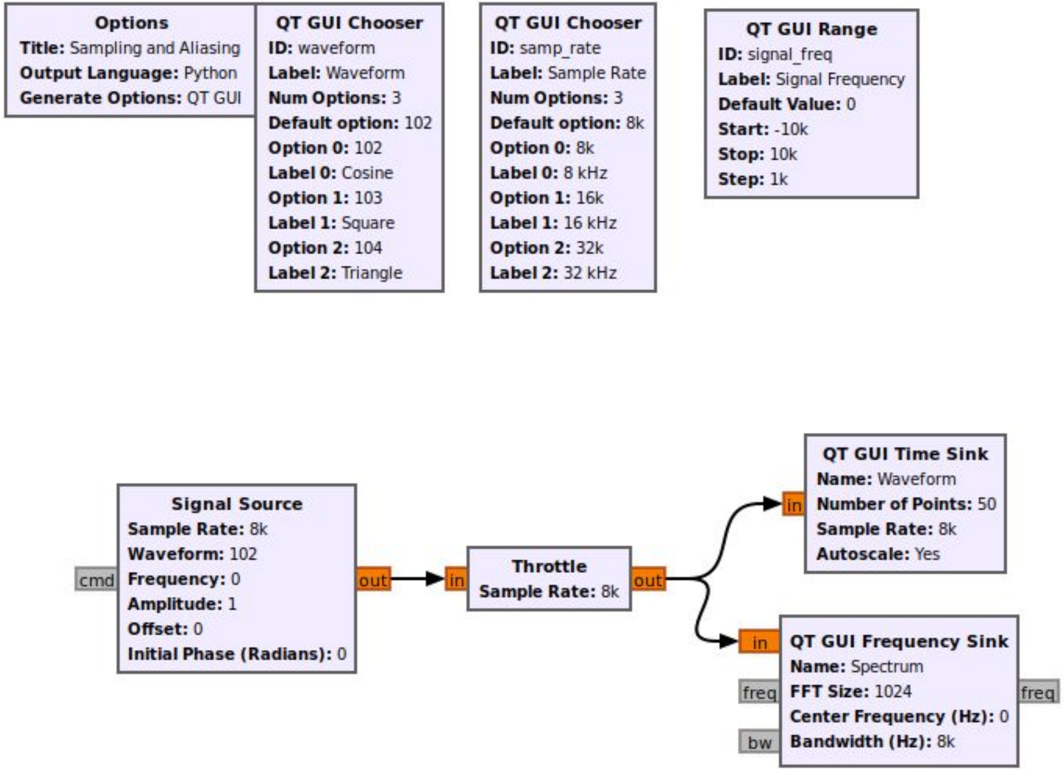
Folded Spectrum View



Examples of aliasing with reference to Nyquist Zones

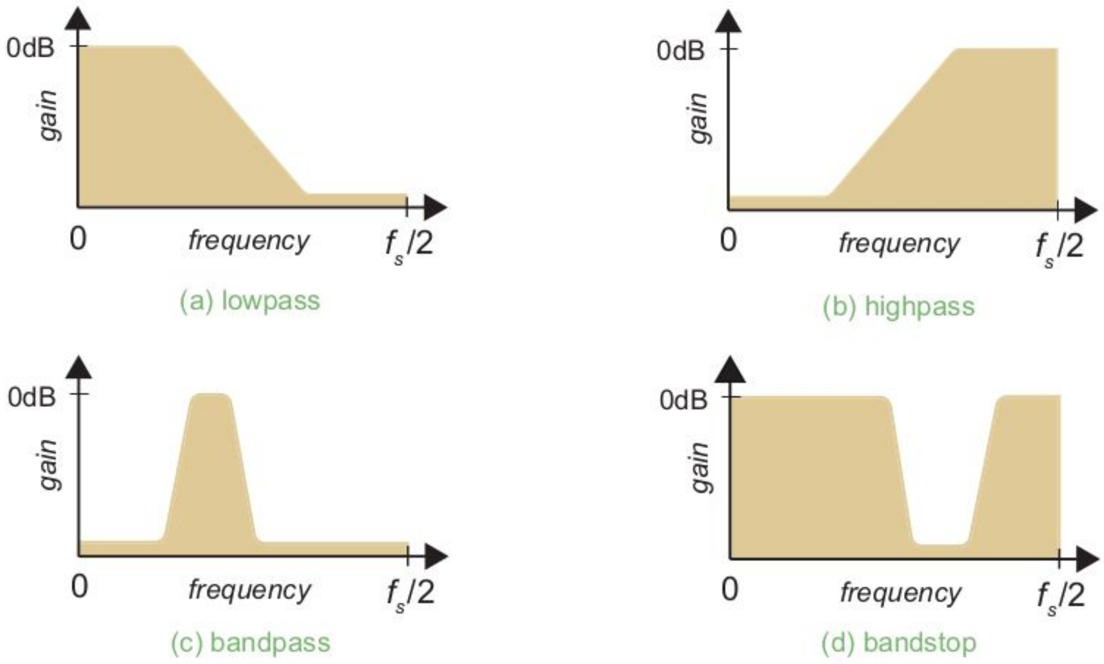


Sampling and Aliasing

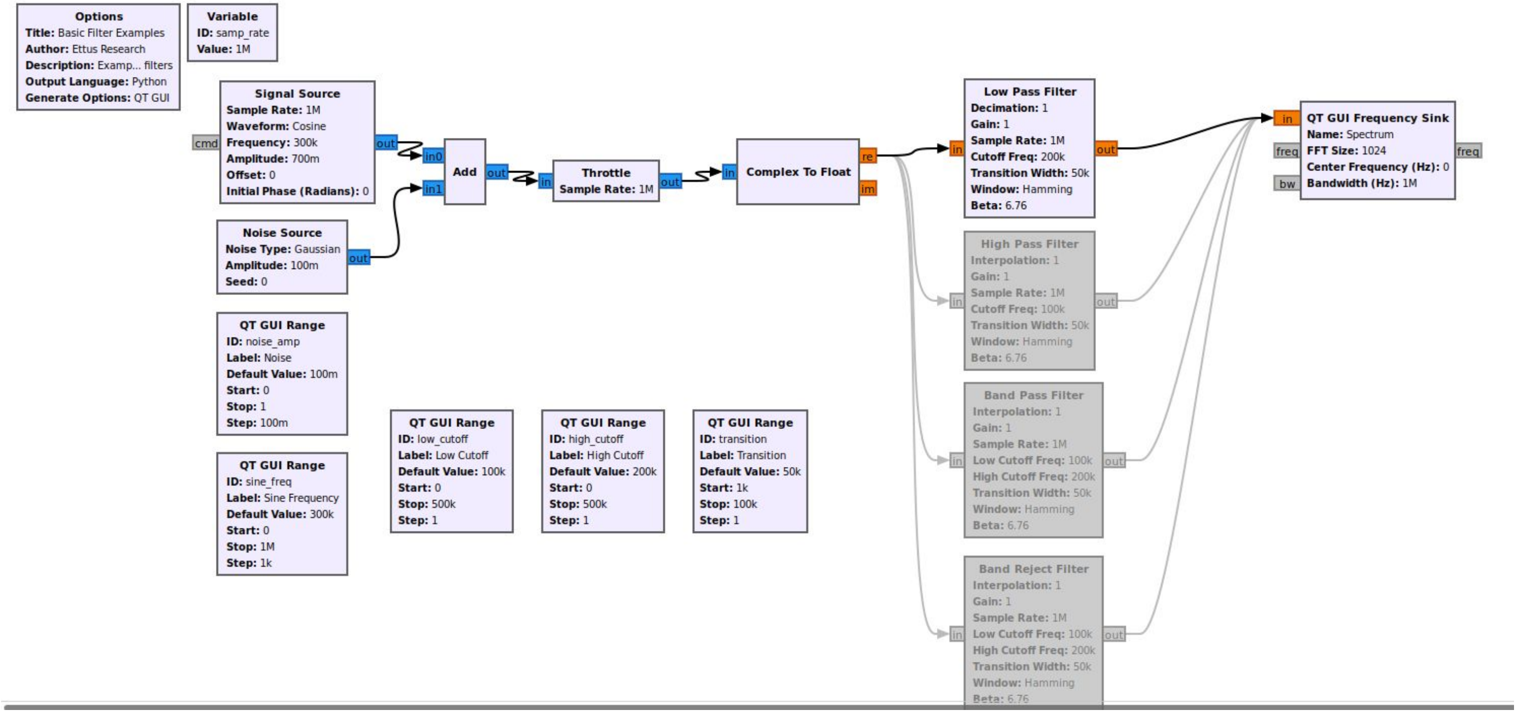


Digital Filters

- A filter modifies the frequency contents of an input signal
- Types
  - LPF
  - HPF
  - BPF
  - Notch



Filters Using GNU Radio

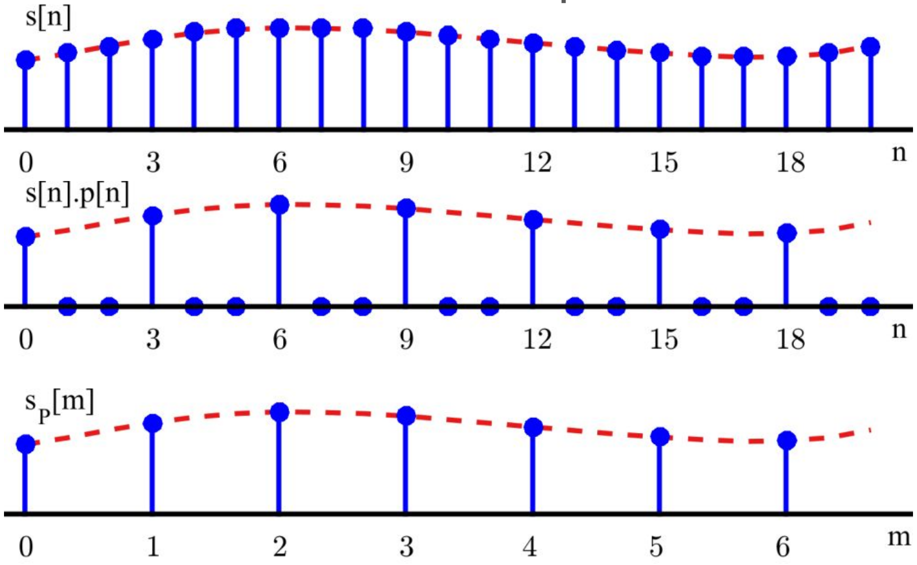


Multirate Signal Processing

- Multirate operations are required to change the sampling rate in a DSP system to optimise computational efficiency
- Some example scenarios
  - To match the sampling rates of two signal paths that will be combined
  - To adjust the sampling rate closer to Nyquist when the signal bandwidth changes
  - To match the sampling rate of an external interface, such as a DAC
  - To ease analogue anti-alias or image-rejection filter requirements

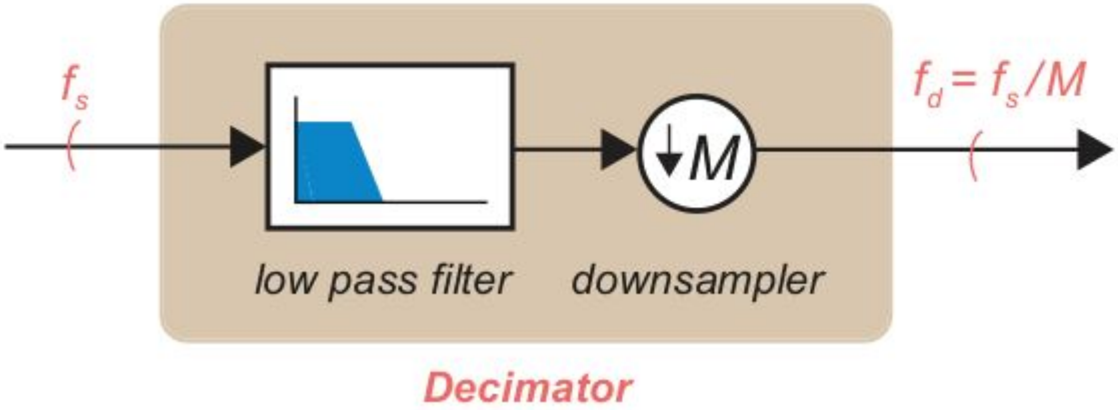
Decimation

- Reducing the sample rate by an integer factor
- Retain every Pth sample and discard the remaining samples
- The new slower sample rate is $1/P$ of the original faster sample rate



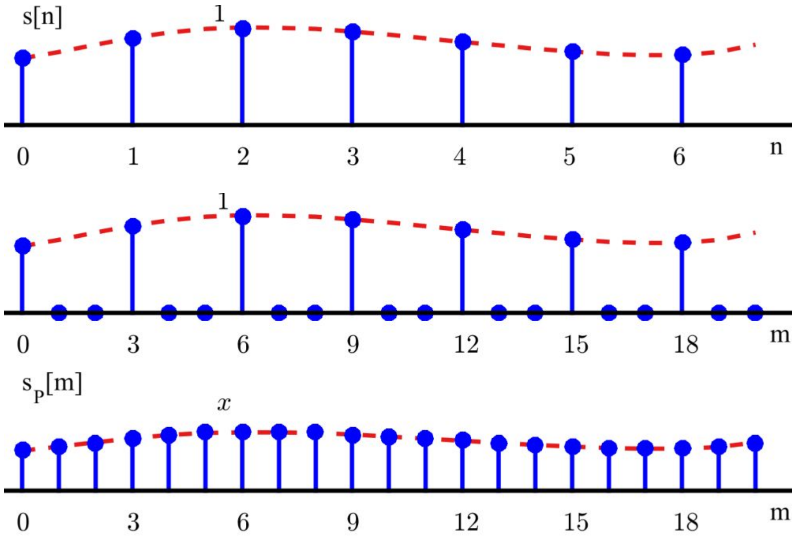
Decimation

- Decimation involves two processes:
  - anti-alias low pass filtering, followed by
  - downsampling



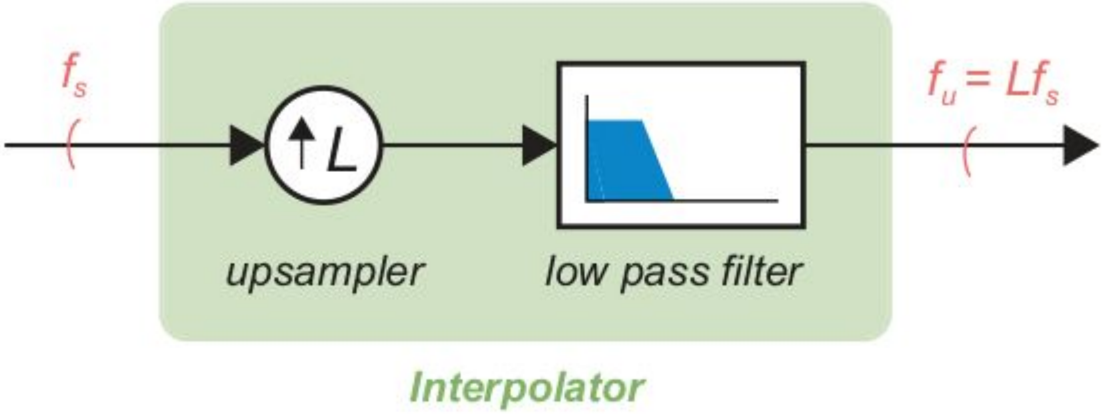
Interpolation

- Increasing the sample rate by an integer factor
- Insert $P - 1$ zeros between the original input samples and interpolate
- The new faster sample rate is P times the original slower sample rate



Interpolation

- An interpolator is composed of
  - an upsampling operation, followed by
  - a low pass image rejection filter

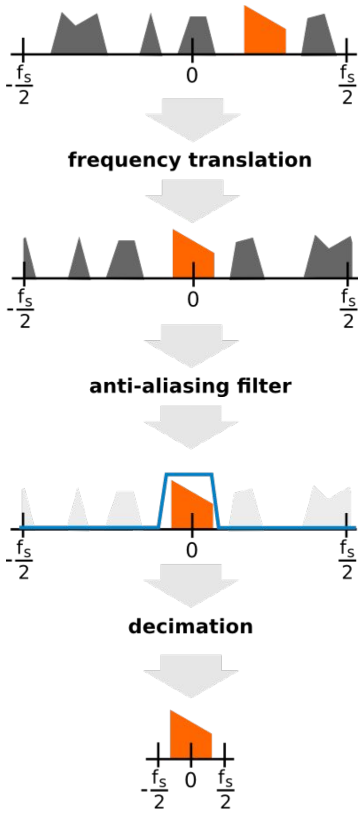


Other Multirate Operations

- There are other types of operation to be aware of, beyond simple decimation and interpolation by integer factors
- Resampling a signal by a **rational fraction**
  - If the sampling rate is to be changed by the ratio of two integers, e.g. a rate change from 100 MHz to 150 MHz could be expressed as $R = 3 / 2$. Rational fractional rate changes can be achieved using a **cascade** of an interpolator and decimator, e.g. $L = 3$ and $M = 2$ in this example. The resulting structure can be optimised using polyphase methods.
- Resampling a signal by an **irrational fraction**, or by a factor that changes over time
  - Where there is no convenient integer-based expression for the resampling ratio, or where it is dynamic, a different type of approach is required. Popular methods include highly oversampled polyphase filters, and Farrow structures.

Frequency Xlating FIR Filter

- Frequency Xlating FIR Filter is a block that:
  - performs **frequency translation** on the signal,
  - **downsamples** the signal by running a **decimating FIR filter** on it.
- It can be used as a **channelizer**:
  - it can select a narrow bandwidth channel from the wideband receiver input.



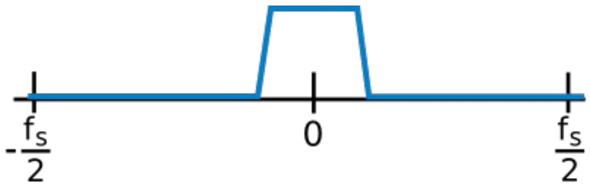
Suppose this is the stations in FM radio example!

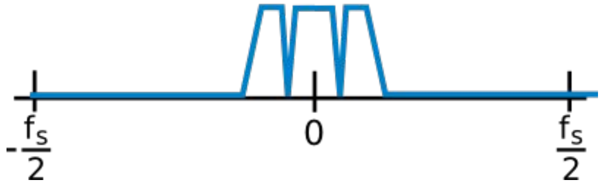
Our aim is to select only one channel

Frequency Xlating FIR Filter

- If you have Real taps, then your FIR filter will be symmetric in the frequency domain.

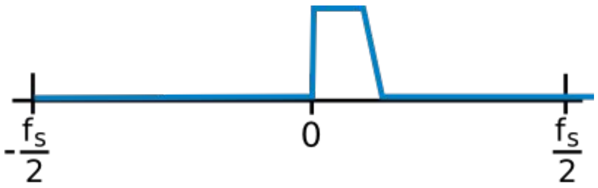
```
firdes.low_pass(1,samp_rate,samp_rate/(2*decimation), transition_bw)
```

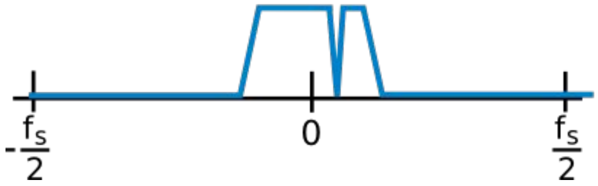




- If you have Complex taps, then your FIR filter will not have to be symmetric in the frequency domain.

```
firdes.complex_band_pass(1, samp_rate,
-samp_rate/(2*decimation),
samp_rate/(2*decimation), transition_bw)
```

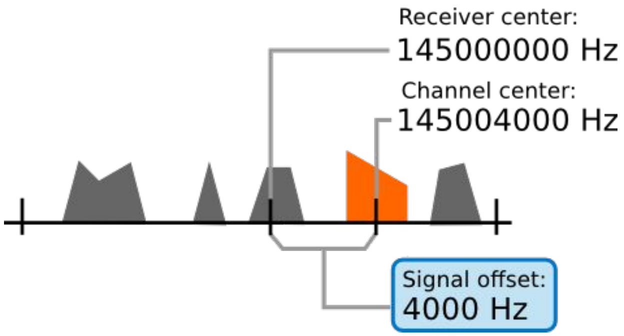




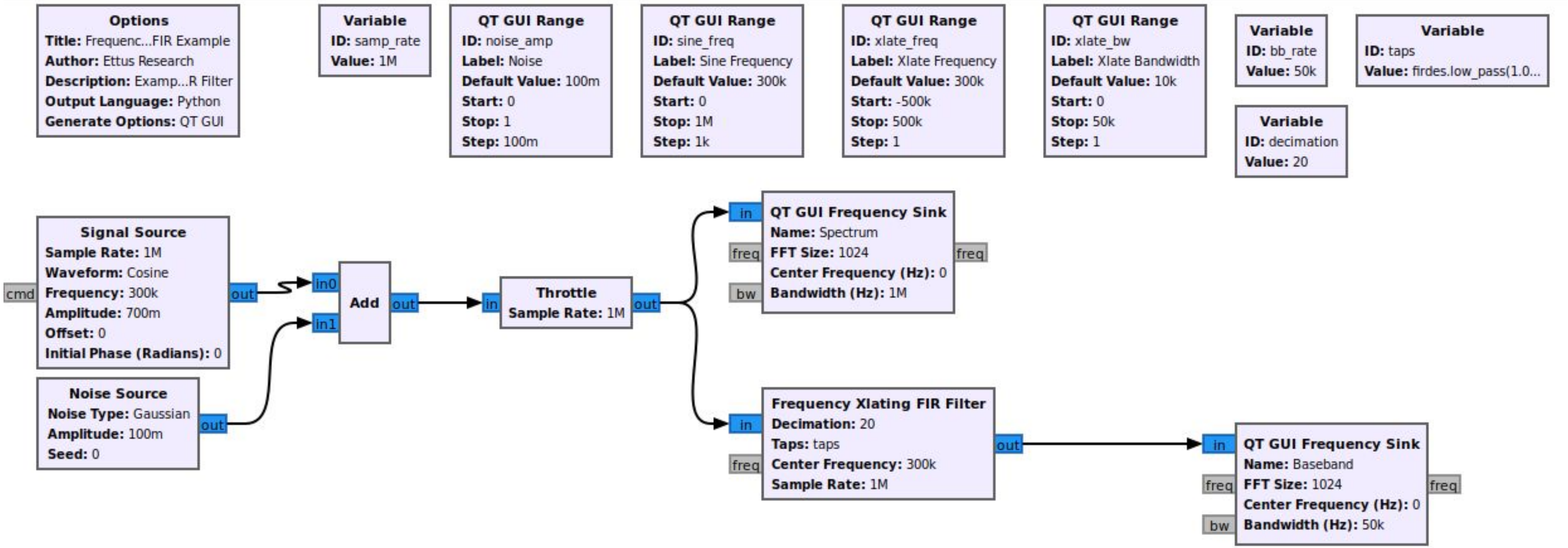
Frequency Xlating FIR Filter

- **Decimation**: the integer ratio between the input and the output signal's sampling rate.
- Example:
    - Input sample rate = 240000
    - Decimation factor = 5
    - Output sample rate = 240000 ÷ 5 = 48000

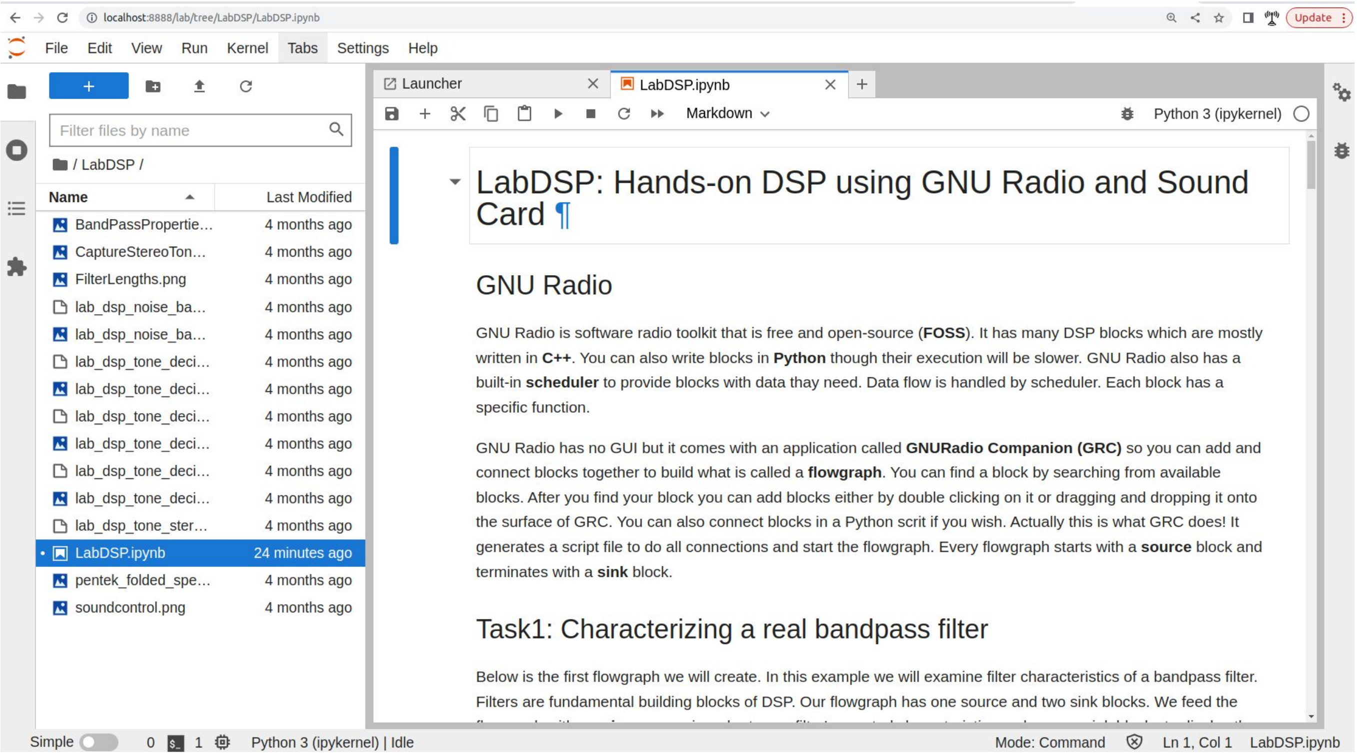
- **Center frequency**: the frequency translation offset frequency.
- In practice, it is the frequency offset of the signal if interest to be selected from the input.



Frequency Xlating FIR Filter



LabDSP.ipynb



Outline

- About me
- About tutorial
- GNU Radio
- Lab: Digital Signal Processing (DSP)
- **Lab: Software Defined Radio (SDR)**
- Lab: Wide Band Frequency Modulation (WBFM)

What is Software Defined Radio (SDR)?

“A radio in which aspects of functionality are implemented in, or controlled by, software.”

- Flexible functionality
  - the operation of a radio can be changed without making any physical alterations to the device
- Algorithms from DSP and communications theory running as real-time software on a CPU, GPU and/or FPGA
- Joe Mitola first coined the term in 1991

Why SDR?

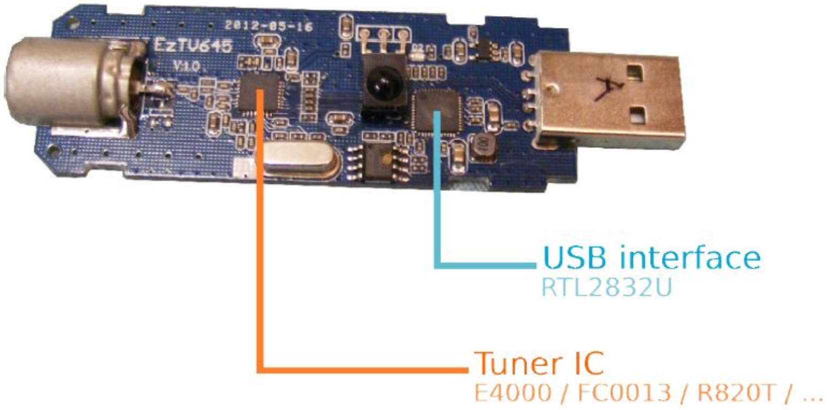
- Traditional radios are hard-wired to specific frequency bands and communication protocols
  - Fixed-function, Black Box
  - Can’t be easily modified, can’t easily access internal values and states
- SDR provides:
  - Flexibility
  - Upgradability
  - Reconfigurability
  - Lower Cost

Key SDR Parameters (Features)

- Frequency (Tuning) Range
- Instantanous Bandwidth
- Bit resolution
- Interface (USB, Ethernet, PCIe)
- Rx/Tx, half-duplex, full-duplex, MIMO
- Preselectors
- Budget: 50$-...k$

RTL-SDR

- “I smell a very cheap poor man’s SDR here☺”
- Cheap man’s radio since 2012
- Hams, DIY, hackers, makers, students,…
- Demodulator
  - Named by RTL2832U chip, DVB-T
- Tuner
  - **R820T**: 24-1766MHz
  - **E4000**: 52-2200MHz



RTL-SDR

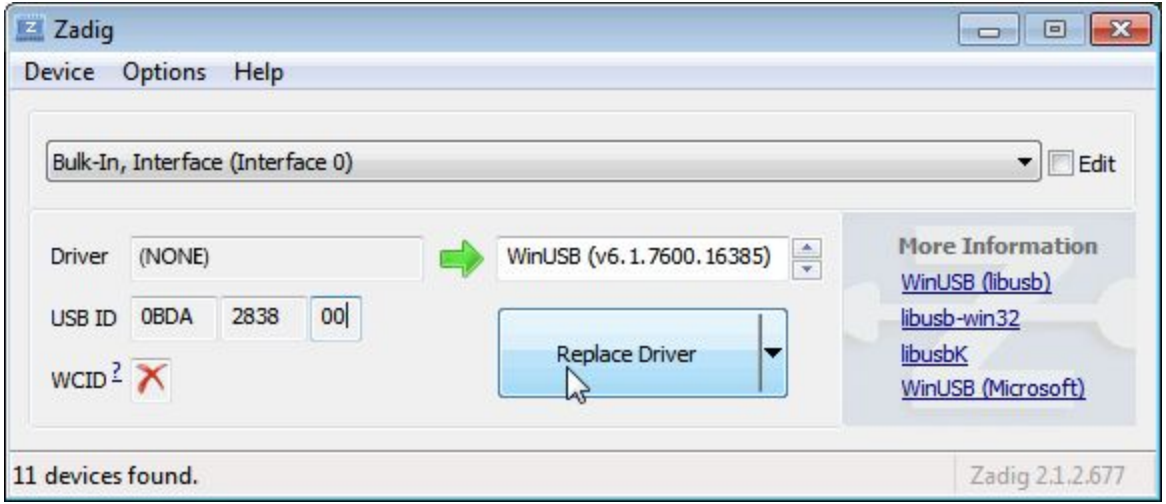
- Receive-only
- 8-bit ADC
- 24MHz-1.75GHz (depends on tuner chip)
- 2.4MSPS BW (stable) upto 3.2M
- “*HamItUp*” upconverter
  - HF coverage

RTL-SDR Driver Installation#1 - Windows

- Plug in your dongle
- Right click zadig.exe file and select "Run as administrator".
- In Zadig, go to "Options->List All Devices" and make sure this option is checked. If you are using Windows 10 or 11, in some cases you may need to also uncheck "Ignore Hubs or Composite Parents".

RTL-SDR Driver Installation#2 - Windows

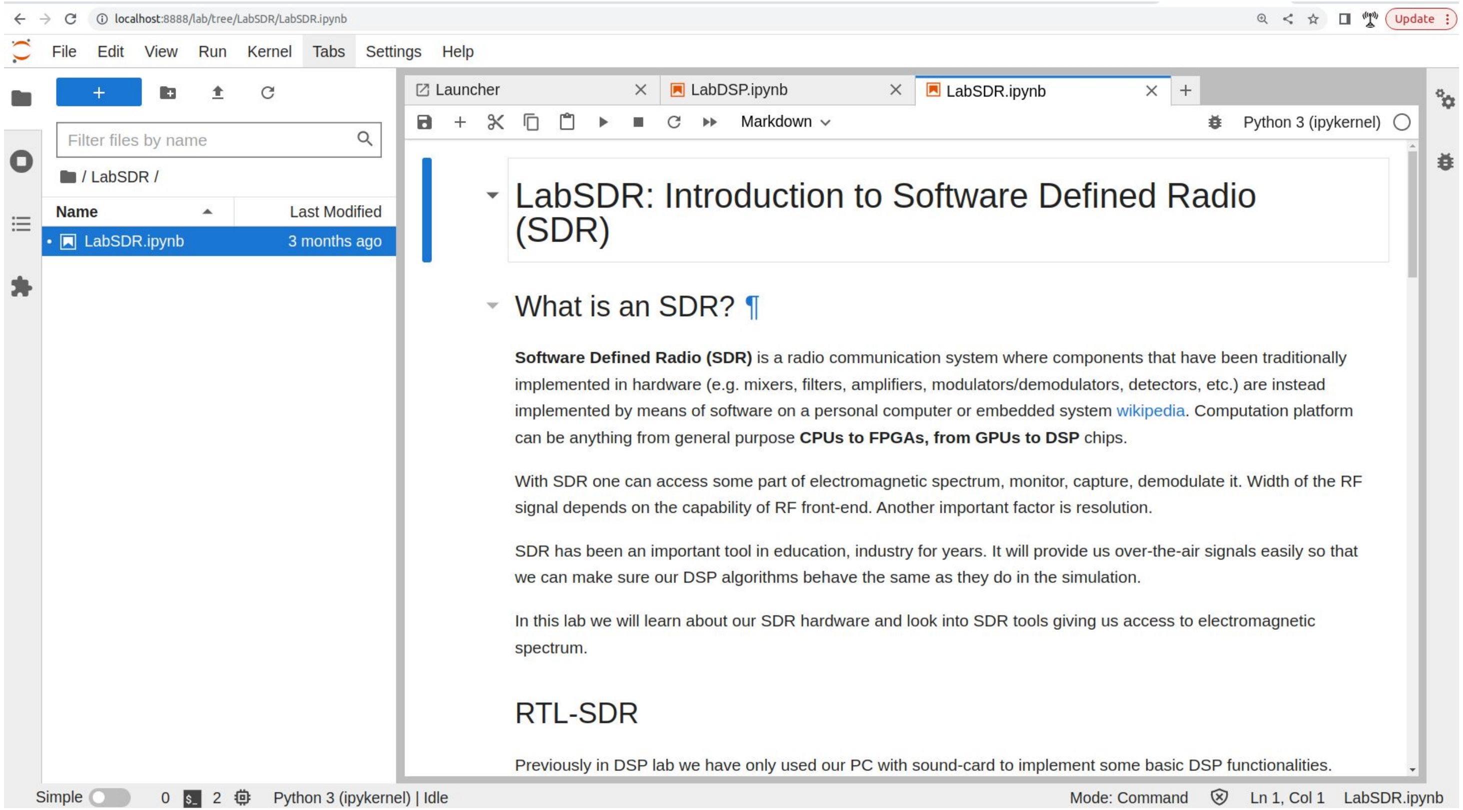
- Select "Bulk-In, Interface (Interface 0)" from the drop down list. Make sure it is Interface 0 (ZERO), and not "1".



RTL-SDR Driver Installation - Linux

- Linux users may blacklist RTL so that default DVB-T driver is not loaded when dongle is plugged in.
  - # cd /etc/modprobe.d/
  - # sudo gedit blacklist-rtl.conf
  - # append: blacklist dvb_usb_rtl28xxu
  - OR
  - # echo "blacklist dvb_usb_rtl28xxu" >> /etc/modprobe.d/blacklist.conf

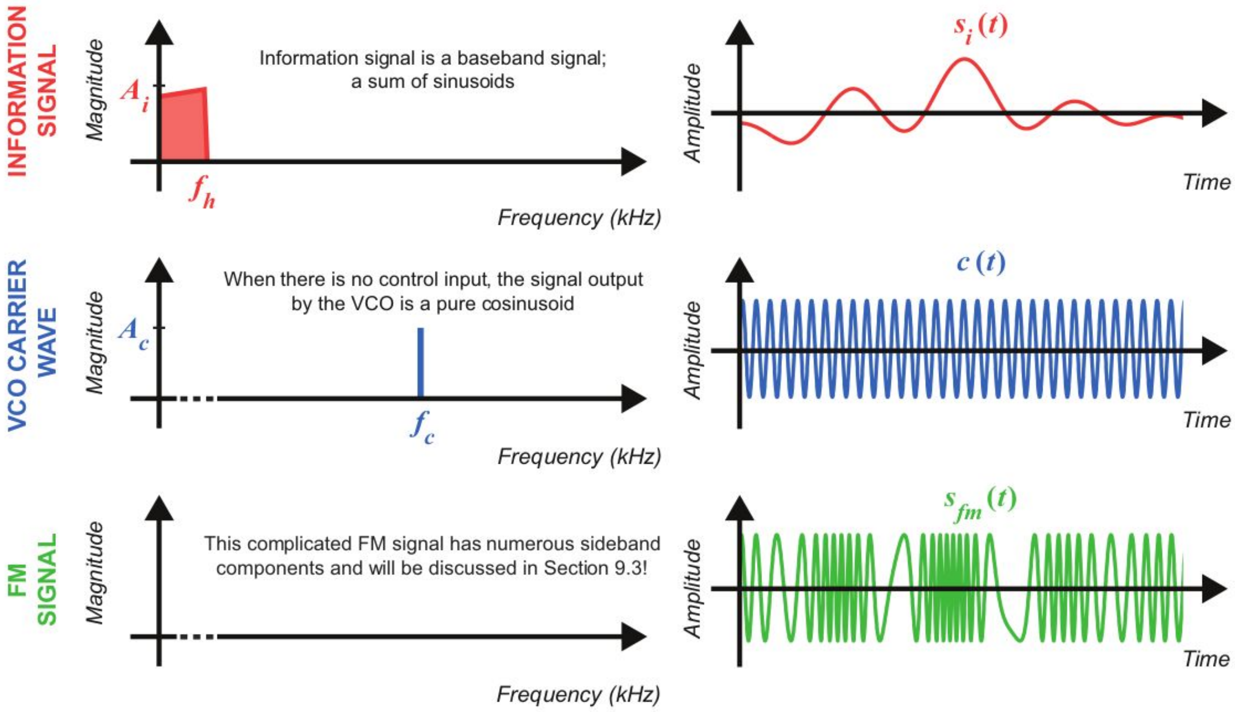
LabSDR.ipynb



Outline

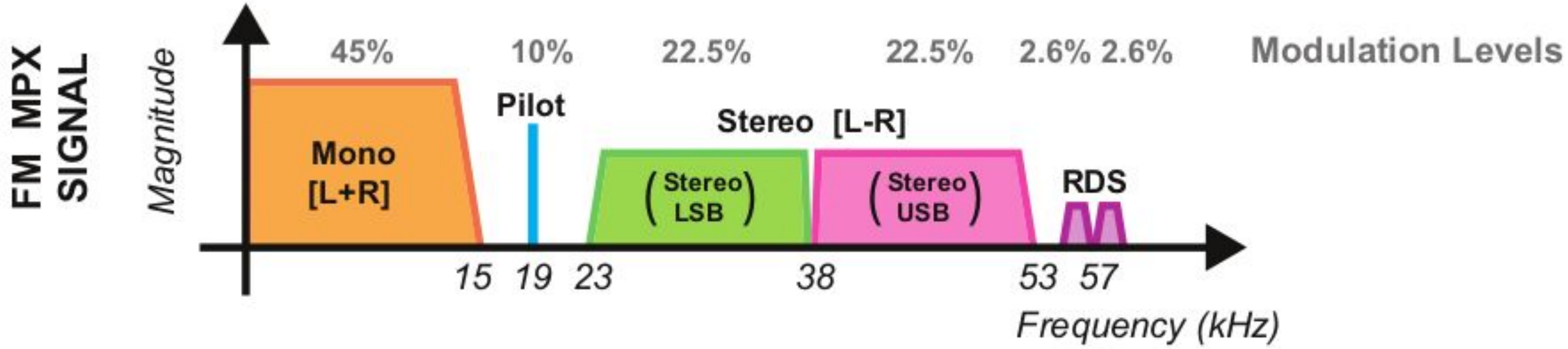
- About me
- About tutorial
- GNU Radio
- Lab: Digital Signal Processing (DSP)
- Lab: Software Defined Radio (SDR)
- **Lab: Wide Band Frequency Modulation (WBFM)**

Frequency Modulation (FM)

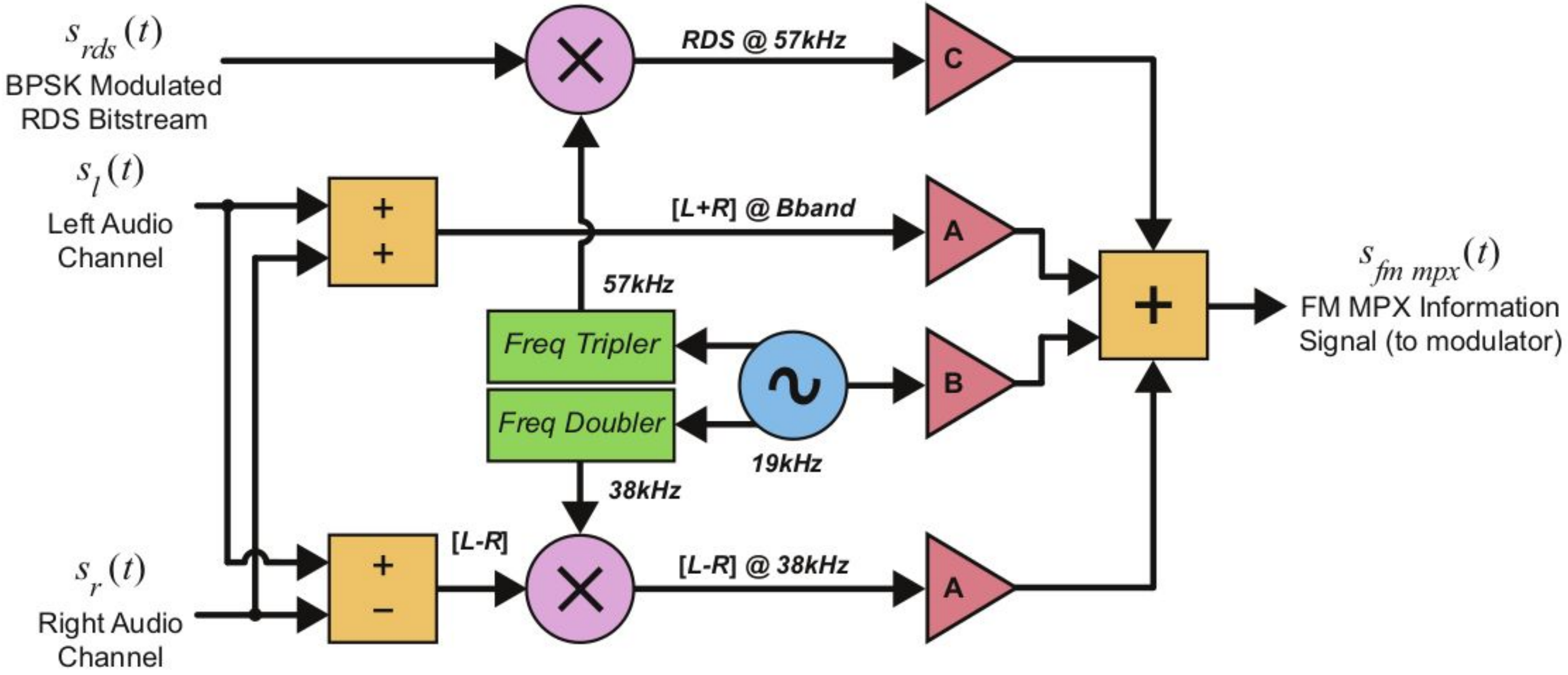


FM Radio Multiplex

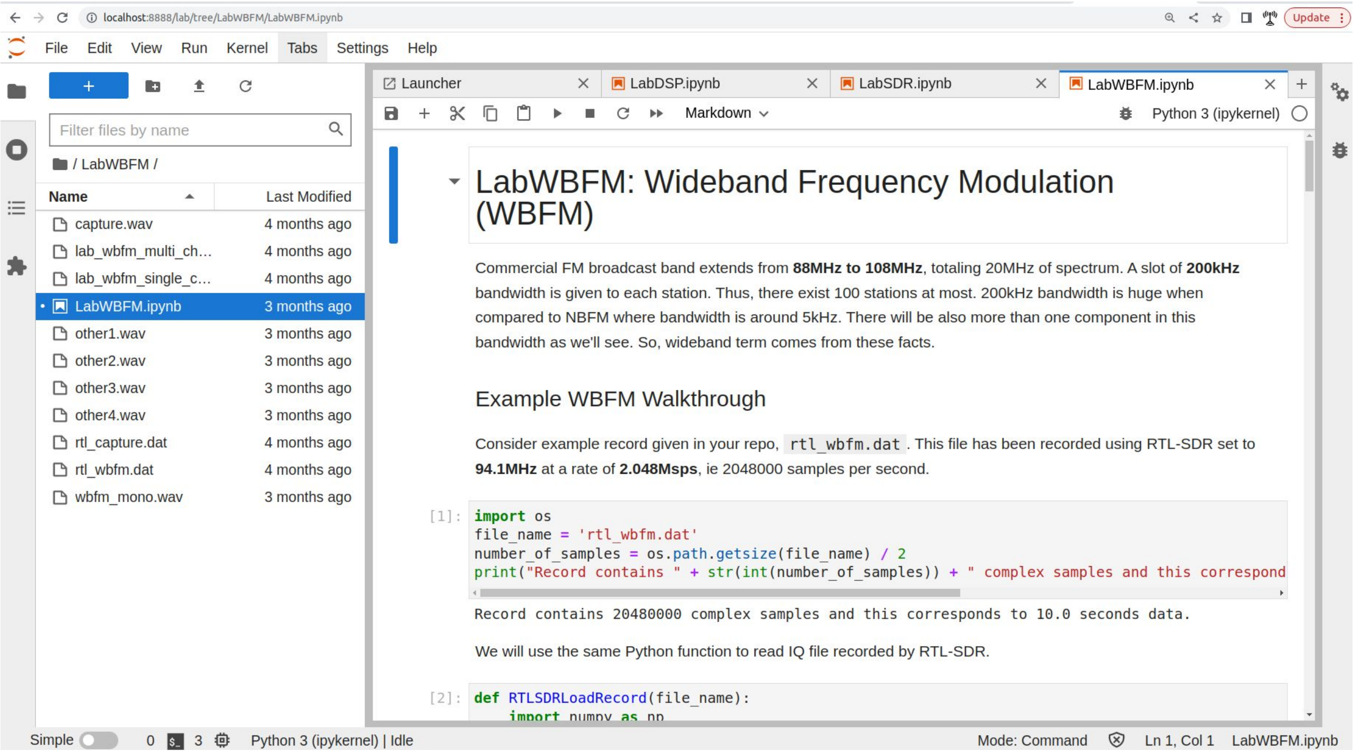
- It is common practice to multiplex multiple information signals together before performing modulation, as this allows for multi-channel transmission using one carrier.



Broadcast FM (WBFM)



LabWBFM.ipynb



Thanks!

murat-sever@live.com