Signal Processing Basics & Filtering

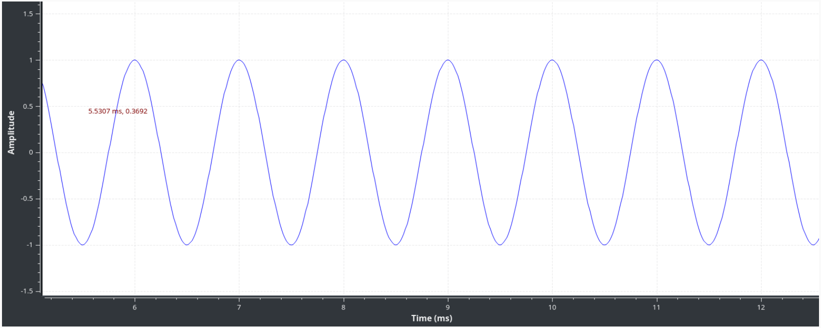
TAs Winter 2021 : Michalis Raptakis , Eleftheria Plevridi
csdp1250@csd.uoc.gr, plevridi@csd.uoc.gr
Computer Science Department, University of Crete

https://gitlab.com/surligas/sdr-tutorial
https://gitlab.com/surligas/gr-tutorial

Manolis Surligas - surligas@csd.uoc.gr

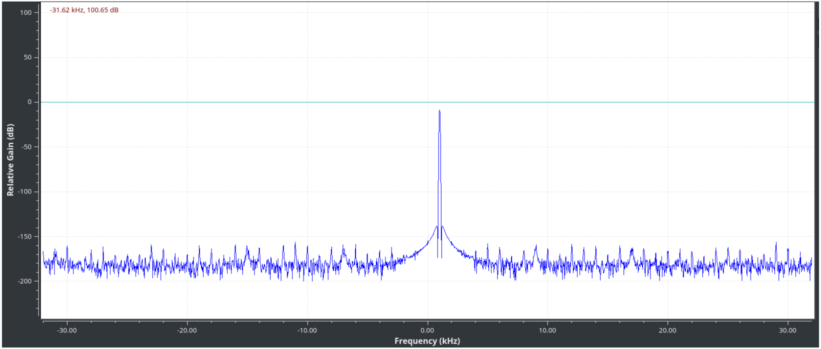
Signal Representation

Signals can be represented in the **time domain**



Signal Representation

...or in the **frequency domain**



Signal Representation

Time domain:

- X-axis represents the time
- Y-axis represents the amplitude (e.g Volts)

Frequency domain:

- X-axis represents the frequency
- Y-axis represents the power
- Commonly power is expressed in logarithmic scale ($10 \times log_{10}(x)$)

Signal Representation

Frequency domain:

- For the transition from time to frequency domain, the Discrete Fourier Transform (DFT) X_k is used

$$
\begin{aligned}
X_k &= \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn} \\
&= \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(\frac{2\pi}{N}kn\right) - i \cdot \sin\left(\frac{2\pi}{N}kn\right)\right]
\end{aligned}
$$

- In simple words, the X_k tranformation, converts the original signal into N cosines, each one with a frequency $\frac{2\pi}{N}kn$ and amplitude x_n
- For real signals the imaginary part is 0

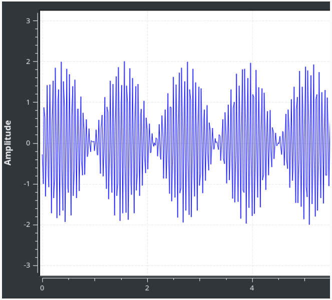
Signal Representation

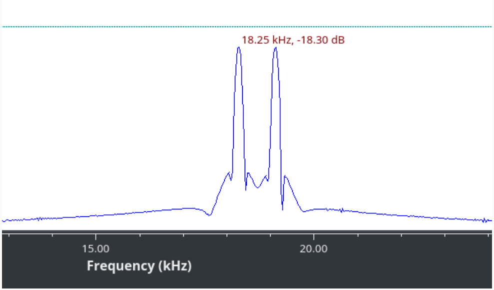
- Changing the signal in the time domain affects the frequency domain
- In most cases we use the frequency domain representation
- Different frequency components can be spotted very easily
- Signals can be spotted visually even with significant amount of noise

Use the simple_example.grc flowgraph located at the examples directory to play with time and frequency domain representation

Signal Representation

Example 1: Finding the frequency components of a mixed signal

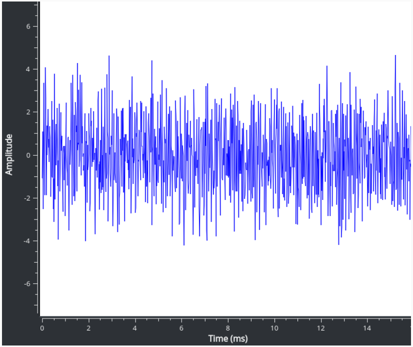


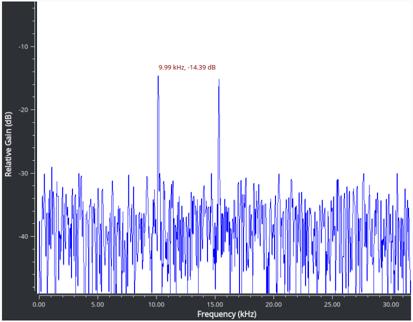


The two different sine waves can be spotted quite easily in the frequency domain comparing to the time domain

Signal Representation

Example 2: Finding the frequency components of a noisy mixed signal

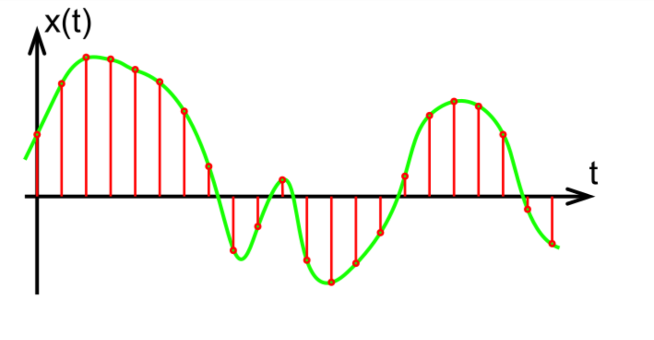




Even with the presence of significant noise, the signal components are distinguishable! (However this is not always the case ☹)

Sampling & Quantization

Sampling is the process of converting an continuous-time signal into a discrete-time signal



Nyquist Theorem

A real signal $x(t)$ band-limited to B Hz can be reconstructed without loss of information iff the sampling frequency is greater than $2 \times B$

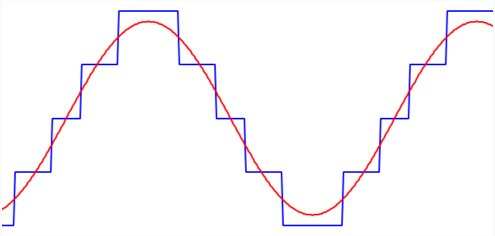
- A band-limited signal $x(t)$, is a signal that does not have spectral components above B Hz
- The minimum sampling frequency $R = 2 \times B$ is called **Nyquist** frequency

Sampling Theory

- In most cases there is a need to change the sampling rate of a signal
  - Hardware capabilities / restrictions
  - Processing maybe easier at specific sampling rates
  - Processing resources
- **Decimation** is the process of reducing the sample rate
- **Interpolation** increases the sampling rate
- Combining decimation and interpolation an arbitrary sampling rate can be achieved
- More about resampling on the next lecture!

Quantization

Quantization converts the signal amplitude into digital form samples



An approximation of the Signal-to-Quantization Noise Ratio (SQNR) for an N-bit ADC/DAC is:
$\mathrm{SQNR} = 20 \log_{10}(2^N) \approx 6.02 \cdot N \ \mathrm{dB}$

Filtering

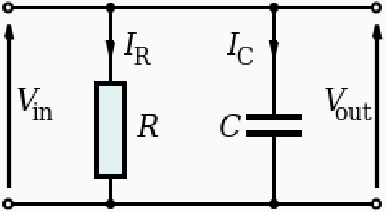
Why do we use filters?

- Filters are used to separate mixed signals
- Extract a specific band of interest
- Avoid undesired effects (aliasing, imaging e.t.c)
- Restore distorted signals
- Pulse shaping

Filter types

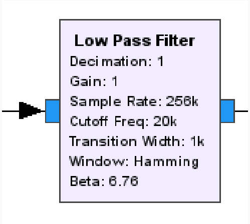
Analog Filters:

- Implemented using electronic components (resistors, capacitors, coils, transistors)
- Used in analog front-ends
- Standard performance
- Difficult to change their properties
- Application specific



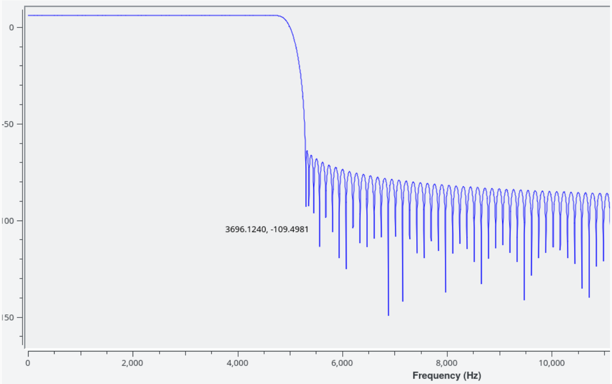
Digital Filters:

- Implemented in software
- Variable performance depending the processing resources
- Change their properties relatively easily
- Can adapt to any application



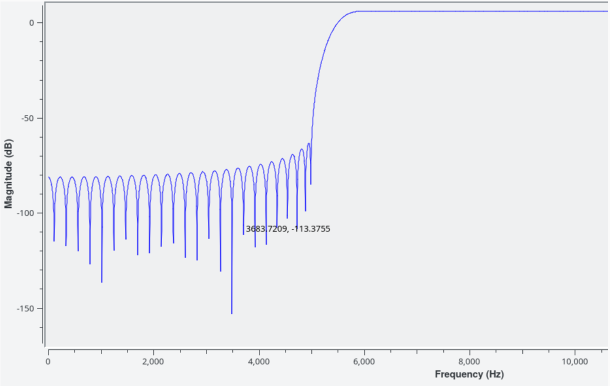
Filter types

- Low Pass Filters
  - Frequencies above a threshold do not pass the filter



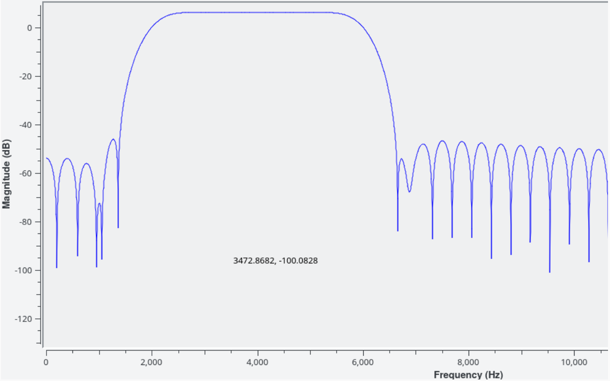
Filter types

- High Pass Filters
  - Frequencies below a threshold do not pass the filter



Filter types

- Band Pass Filters
  - Frequencies of a specific band pass the filter

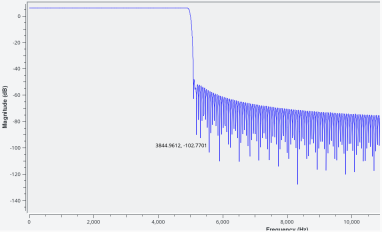


Filter parameters

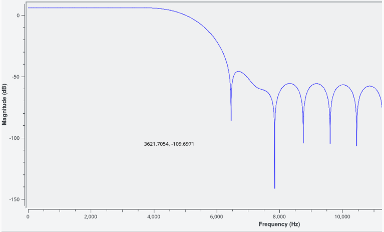
- **Cutoff frequency:** The frequency above (low pass filter) or below (high pass filter) which, the filter starts to attenuate the signal
- **Transition width:** Control how steep is the attenuation of the signal above (low pass filter) or below (high pass filter) the cutoff frequency

Filter parameters

- Small transition width:



- Large transition width:



Filtering and CPU resources

- Transition width specifies the number of taps of the filter
- Steep filters have more taps than relaxed ones
- Designing a filter is always a balance between RF performance and CPU utilization
- Sampling rate plays a key role too

Filtering and CPU resources

Example

- Assume a sampling rate of 1 MSPS
- We are interested in a range of only 10 kHz
- The requirements demand a very steep filter and we have limited CPU resources (e.g RPi3)
- How???

Filtering and CPU resources

Example

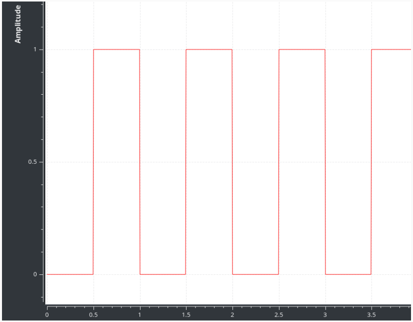
- Assume a sampling rate of 1 MSPS
- We are interested in a range of only 10 kHz
- The requirements demand a very steep filter and we have limited CPU resources (e.g RPi3)
- How???
- Use a relaxed first stage relaxed filter
- Decimate the signal by a factor of e.g 50
- Apply a second stage steep filtering

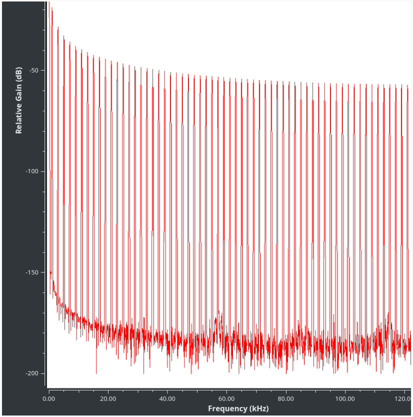
Pulse Shaping Filtering

- Filters can be used also for pulse shaping
- The goal is to minimize spurious signals and increase the spectral efficiency
- Spurious signals can be generated by extreme transitions of the source signal
- A simple square wave generated from a bit stream has an infinite bandwidth ➔non practical to transmit

Pulse Shaping Filtering

- A simple square wave generated from a bit stream has an infinite bandwidth ➔non practical to transmit
- A square wave is an infinite sum of sine waves





Pulse Shaping Filtering

- Commonly for shaping we use the Root Raised Cosine filter (RRC)
- Smooths the extreme transitions thus reducing the spurious