EE-342 Telecommunications Systems **Lab 1**
Spring Semester
Telecommunications Circuits Laboratory
EPFL

1 GNU Radio Framework Introduction

In this lab, you will get familiar with GNU Radio, a free and open-source software development toolkit for building software-defined radios (SDRs).
GNU Radio provides a set of signal processing blocks that allow you to implement radio communication systems in software rather than hardware. Software-defined radios are widely used in both academia and industry to develop and test various wireless communication systems.

1.1 GNU Radio Companion and Flowgraphs

The easiest way to use GNU Radio is through **GNU Radio Companion (GRC)**, a graphical tool that allows you to create **signal processing flowgraphs**. A **flowgraph** is a visual representation of a signal processing chain, where each block performs a specific function.

1.1.1 Task 1: Creating a Simple Flowgraph

In this task, you will build a basic flowgraph that generates a sine wave, scales it by a constant factor, and displays the resulting signal.

Step 1: Setting Up the Flowgraph

- Open GNU Radio Companion by typing `gnuradio-companion &` in a terminal.
- Press `Ctrl+N` to create a new flowgraph.
- Double-click the **Options** block to set the **ID** and **title** of your flowgraph.
- Save your flowgraph in your home directory with `Ctrl+S`.

By default, the flowgraph includes a **Variable** block specifying the sample rate. You can leave this at its default value, as its effect will be discussed in a future lab.

Step 2: Adding Signal Processing Blocks GNU Radio Companion provides a wide range of signal processing blocks, accessible from the right-hand panel. If the panel is not visible, press `Ctrl+F` to bring it up and search for blocks by name.
To recreate the flowgraph shown in the figure below, use the following blocks:

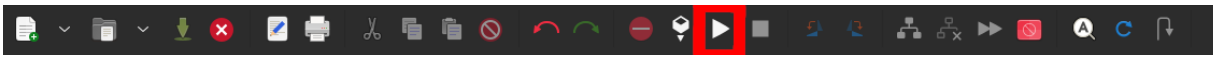
- **Signal Source**

- **Multiply Const**
- **QT GUI Range**
- **QT GUI Time Sink**
- **Throttle**

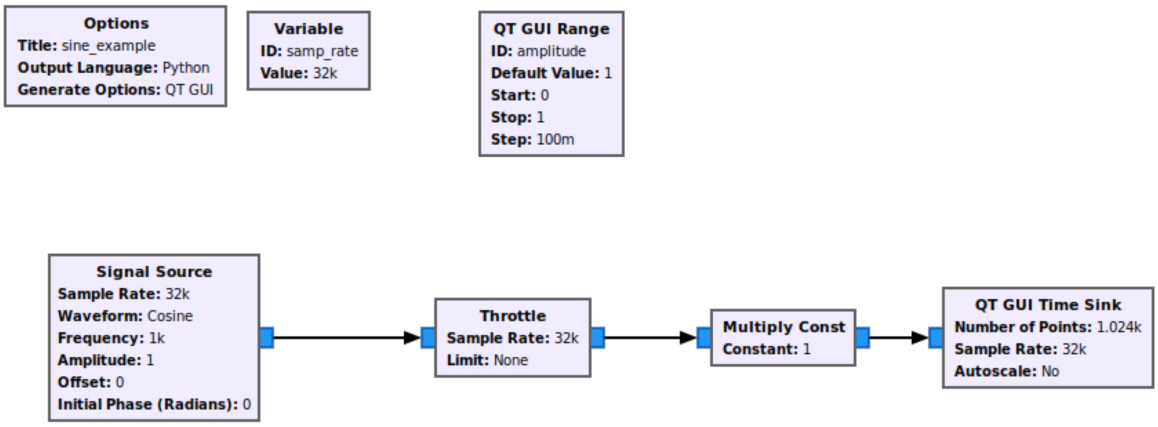
Step 3: Connecting and Configuring Blocks

- To connect blocks, either:
  - Click on two ports sequentially, or
  - Click and drag from one port to another.
- Double-click any block to modify its parameters and view its documentation.
- You can use the **ID** of a **Variable** or **QT GUI Range** block to dynamically set a parameter in another block (e.g., the **const** parameter in **Multiply Const**).

Step 4: Running the Flowgraph Once all blocks are connected, press the **"Play"** button at the top of the window to run your flowgraph.



You should see a window with a slider and a plot appearing. You can change the amplitude of the sine wave by moving the slider and observe the effect on the plot.



Options
Title: sine_example
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 32k

QT GUI Range
ID: amplitude
Default Value: 1
Start: 0
Stop: 1
Step: 100m

Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 1k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Throttle
Sample Rate: 32k
Limit: None

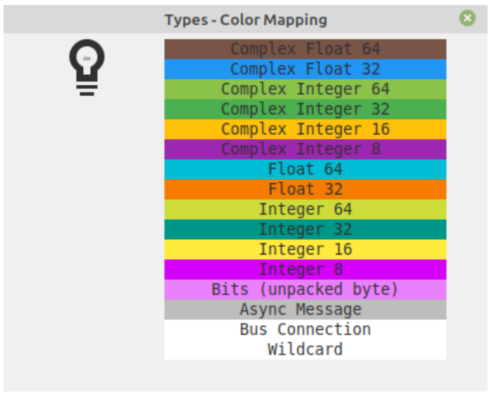
Multiply Const
Constant: 1

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 32k
Autoscale: No

1.1.2 Notes on the Blocks Used

- The color of the port of each block indicates the type of data that is expected or produced by the block (see figure below). Only ports of the same color can be connected together.
- The *Throttle* block is used to limit the rate at which the flowgraph processes samples. When set to the `samp_rate`, it ensures real-time processing. It is required to avoid overloading the CPU when the flowgraph is running without a physical radio connected.

- When modifying the parameters of a block, some parameters are underlined. This means that the parameter can be changed during runtime. If not underlined, the parameter can only be changed when the flowgraph is stopped.
- The GNU Radio Companion interface shows unit prefixes (e.g., 100m); however, you must input the full number (e.g., 0.1) in the parameter field.



Types - Color Mapping

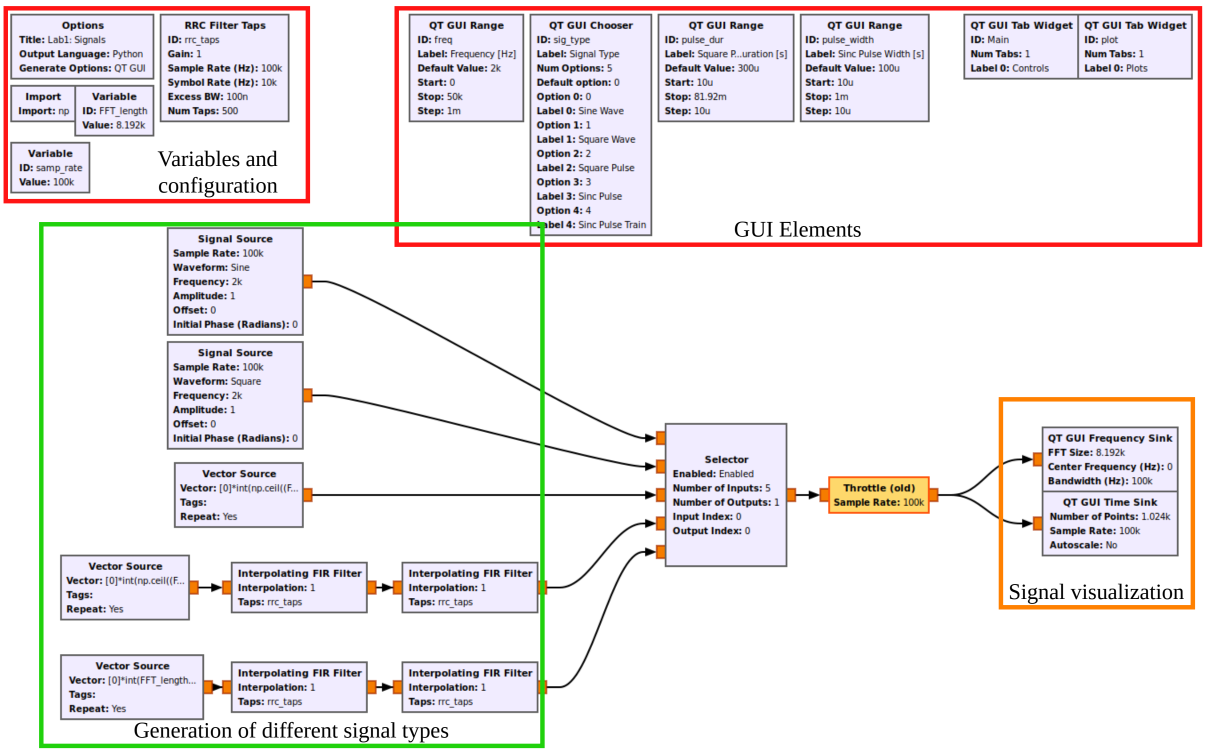
Complex Float 64
Complex Float 32
Complex Integer 64
Complex Integer 32
Complex Integer 16
Complex Integer 8
Float 64
Float 32
Integer 64
Integer 32
Integer 16
Integer 8
Bits (unpacked byte)
Async Message
Bus Connection
Wildcard

Figure 1.1: Port colors in GNU Radio

1.2 Signal Processing in Time and Frequency Domain

In this part of the lab, you will use a flowgraph to observe the effect of different signal processing operations on the time and frequency domain representation of a signal.
You can open the flowgraph `lab1_signals.grc`, which serves as the basis for this section of the lab.
The flowgraph consists of the following components:

- **Signal Generation**: Different types of signals are generated, including:
  - Sine wave
  - Square wave
  - Square pulse
  - Sinc pulse
  - Sinc pulse train
- **Selector**: Allows you to choose which signal to visualize.
- **QT GUI Frequency Sink**: Displays the signal in the frequency domain.
- **QT GUI Time Sink**: Displays the signal in the time domain.


Variables and configuration
GUI Elements
Generation of different signal types
Signal visualization

Options
Title: Lab1: Signals
Output Language: Python
Generate Options: QT GUI

RRC Filter Taps
ID: rrc_taps
Gain: 1
Sample Rate (Hz): 100k
Symbol Rate (Hz): 10k
Excess BW: 100n
Num Taps: 500

Import
Import: np

Variable
ID: FFT_length
Value: 8.192k

Variable
ID: samp_rate
Value: 100k

QT GUI Range
ID: freq
Label: Frequency [Hz]
Default Value: 2k
Start: 0
Stop: 50k
Step: 1m

QT GUI Chooser
ID: sig_type
Label: Signal Type
Num Options: 5
Default option: 0
Option 0: 0
Label 0: Sine Wave
Option 1: 1
Label 1: Square Wave
Option 2: 2
Label 2: Square Pulse
Option 3: 3
Label 3: Sinc Pulse
Option 4: 4
Label 4: Sinc Pulse Train

QT GUI Range
ID: pulse_dur
Label: Square P...uration [s]
Default Value: 300u
Start: 10u
Stop: 81.92m
Step: 10u

QT GUI Range
ID: pulse_width
Label: Sinc Pulse Width [s]
Default Value: 100u
Start: 10u
Stop: 1m
Step: 10u

QT GUI Tab Widget
ID: Main
Num Tabs: 1
Label 0: Controls

QT GUI Tab Widget
ID: plot
Num Tabs: 1
Label 0: Plots

Signal Source
Sample Rate: 100k
Waveform: Sine
Frequency: 2k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Signal Source
Sample Rate: 100k
Waveform: Square
Frequency: 2k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Vector Source
Vector: [0]*int(np.ceil((F...
Tags:
Repeat: Yes

Vector Source
Vector: [0]*int(np.ceil((F...
Tags:
Repeat: Yes

Interpolating FIR Filter
Interpolation: 1
Taps: rrc_taps

Interpolating FIR Filter
Interpolation: 1
Taps: rrc_taps

Vector Source
Vector: [0]*int(FFT_length...
Tags:
Repeat: Yes

Interpolating FIR Filter
Interpolation: 1
Taps: rrc_taps

Interpolating FIR Filter
Interpolation: 1
Taps: rrc_taps

Selector
Enabled: Enabled
Number of Inputs: 5
Number of Outputs: 1
Input Index: 0
Output Index: 0

Throttle (old)
Sample Rate: 100k

QT GUI Frequency Sink
FFT Size: 8.192k
Center Frequency (Hz): 0
Bandwidth (Hz): 100k

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 100k
Autoscale: No

1.2.1 Periodic Signals

Sine Wave

You can start by running the flowgraph and observe the effect of changing the frequency of the sine wave on the spectrum and time plot.
The time plot should look fine however the frequency plot might not be as expected.

Question: Can you explain why the spectrum is not two diracs in $\pm f_c$?

Question: Can you write the criterion that should be satisfied to obtain a line spectrum in the frequency domain based on the sine frequency f_c, the sampling rate f_s, and the DFT size L?

Question: Can you propose a set of parameters that will allow you to achieve a line spectrum (i.e.,only diracs in the frequency domain) given that this flowgraph uses $f_s = 100\,\text{kHz}$ and $L = 8192$?

If yes, you can use that configuration for the next section related to the periodic squared signal. If not, you can check for a possible solution in the comment section of the block *Signal Source* generating the Sine. To access the comments of a block, double click on the block and go under the "Advanced" tab.

Even though your choice of parameter is correct, you might still see some spectral components that are not expected (though very small $-150\,\text{dB}$). You can mostly ignore them given their very small contribution.

Question: However, can you think of a reason for their presence?

Square Wave

You can now change the signal to a square wave.
By looking at the spectrum of a square signal there seems to be a logic in the frequency of the spectral components.

Question: Knowing that the sum of Fourier transform is a linear operation, can you express the square signal as a sum of sine waves that relates to the frequency parameter f_c? (In the expression you propose, you can keep the amplitude of each sine you use as a variable a_n. We are only interested in the frequency components.)

One of the spectral component of the square wave doesn't fit easily in an single $\sum$ expression.

Question: Can you add some operations to the flowgraph to make it disappear? **Hint**: The blocks that may be useful are under the tab "Core">>"Math Operator" in the right part of the gnuradio-companion window. If the list of block is not visible just use `Ctrl+B` to make it appear

1.2.2 Square Pulse

We will now look at the properties of a square pulses. Try different pulse duration and observe how the spectrum shape changes.

Question: Does the zero-crossing bandwidth changes according to relation seen during the lecture? How do those two parameters relate? (stay below 400μs to see the effect easily).

Question: What happens when you use a pulse of length equal to the FFT window? Is it expected?

In the flowgraph, add a new *signal source* generating a sine wave at the frequency set by the variable *freq*. Note that the output type should be set to ”float”. You can then multiply the pulse with the sine wave using a *Multiply* block and observe the effect in the time and frequency domain.

Question: When multiplying a signal with a sine in the time domain, what are you expecting to happen in the frequency domain? Can you observe it on the spectrum plot?

1.2.3 Sinc Pulse

Instead of using a square pulse which has a very wide bandwidth, you can use a sinc pulse which in theory has a limited bandwidth (a box function). Change the signal type to sinc pulse and observe that the overall shape of the spectrum is effectively far more constrained.
However, it is again not a perfect box function, as one could expect from the spectrum of a sinc.

Question: Can you, similarly to the periodic signal part, find a set of parameters that would allow you to obtain a perfect box function in the frequency domain?

Question: How does the width of the sinc pulse is supposed to relate to the width of the spectrum?

Question: In the plot, how much dB of attenuation do you have at the theoretical limit of the bandwidth?

Question: What happens when you use a sinc pulse width to the minimum value, why?

1.2.4 Sinc Pulse Train

Finally you can change the signal to a train of sinc pulses.

Question: By looking at the flowgraph in gnuradio-companion, can you explain how the pulse train are generated? (**Hint**: have a look at the documentation of the blocks used)

Question: Can you explain how the spectrum of a single pulse relates to the one of a train of the same reference pulse? (**Hint**: move back and forth between using the signal type ”Sinc pulse” and ”Sinc pulse train”)

Knowing that there are some important relationships between multiplication and convolutions between time and frequency domain.

Question: Can you express the sinc pulse train in a form composed only of the base pulse $g(t)$, simple functions which Fourier transform is well known (in standard Fourier pairs), and simple operations (i.e. addition, multiplication, and convolution)?

Question: Based on the expression found, can you explain intuitively the effect observed in the frequency domain?