Project: Innovative Open Source Courses for Computer Science

Wireless Signal Processing in GNU Radio Environment Teaching Material

Wojciech Sałabun
West Pomeranian University of Technology in Szczecin

30.05.2021

Innovative Open Source Courses for Computer Science

This syllabus was written as one of the outputs of the project "Innovative Open Source Courses for Computer Science", funded by the Erasmus+ grant no. 2019-1-PL01-KA203-065564. The project is coordinated by West Pomeranian University of Technology in Szczecin (Poland) and is implemented in partnership with Mendel University in Brno (Czech Republic) and University of Žilina (Slovak Republic). The project implementation timeline is September 2019 to December 2022.

Project information

Project was implemented under the Erasmus+.
Project name: **"Innovative Open Source courses for Computer Science curriculum"**
Project nr: **2019-1-PL01-KA203-065564**
Key Action: **KA2 – Cooperation for innovation and the exchange of good practices**
Action Type: **KA203 – Strategic Partnerships for higher education**

Consortium

ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY W SZCZECINIE
MENDELOVA UNIVERZITA V BRNE
ZILINSKA UNIVERZITA V ZILINE

Erasmus+ Disclaimer

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Copyright Notice

Co-funded by the
Erasmus+ Programme
of the European Union

Assignment 1
Introduction to the GNU Radio environment

30.05.2021

GNU Radio is free software that allows the user to design and implement signal processing without the need for hardware such as a signal generator or oscilloscope. It allows the generation of Python code and is an alternative to Matlab or LabView. Project files are saved with .grc extension.

The software can be downloaded from https://www.gnuradio.org/.

Detailed documentation is available at https://www.gnuradio.org/docs/

Once the project has been created, 2 basic components will appear in the workspace

- Design options
- Variable specifying the system sampling rate

GNU Radio basic interface element

- List of GRC environmental components - right-hand side
  - Components are added by dragging an item from the list into the main area
- Program console with declared variables - lower part of GUI
- Workspace with settings bar - upper part of the GUI
  - launching the programme
  - generate code to a Python file with the extension .py
  - aborting the programme

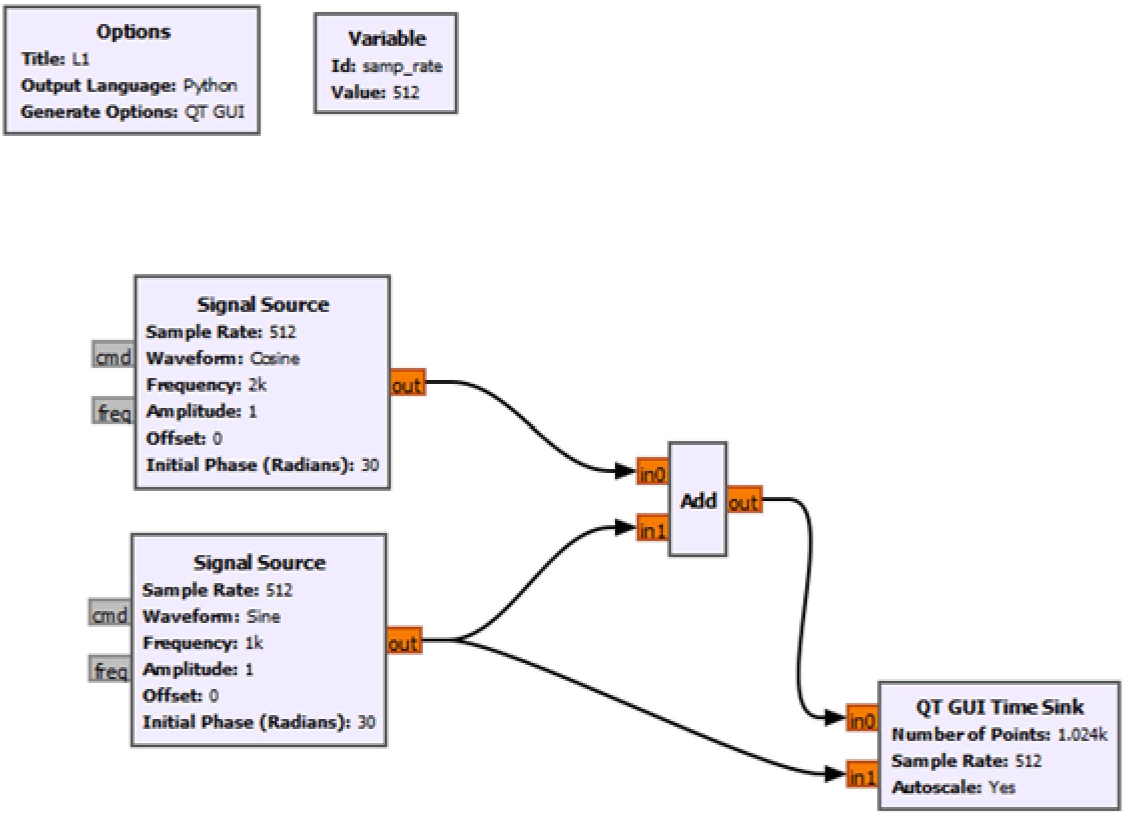
Basic components

- Options - main parameters of the program (title, description, project ID), the Generate Options field must be set, otherwise an error will be returned
- Variable - variable in the program, set the ID and value
- Signal Source (Waveform generators tab)- a block which creates a signal source with selected parameters, we define the frequency, amplitude, offset, signal shape, data type, we specify the data type as float
- Multiply - multiplies two signals, other mathematical operators like Add, Subtract, Divide and logical operators like And, Or, Xor, Not can be used analogically. Blocks are connected by a single click on one of them and then on the selected operator

- Graphics components
    - Sink - allow visualization of the designed signal, the data type must be specified as float
    - Chart settings panel
- Rational Resampler - block used to change the sampling frequency of a signal
- Throttle - block used to generate real time graphs, limits the number of bits used that pass through the component
- Audio Sink - to play audio from a signal using a sound card, typical Sample Rate is 48KHz
- Files operators
    - Wav File Source - reading data from a file
    - Wav File Sink – writing data to a file

Tasks

- Build 5 systems with different properties
    - use mathematical and logical operators
    - make various settings for the generated signal
- Use Sink to visualise layouts
    - Use different settings to display graphs



Options
Title: L1
Output Language: Python
Generate Options: QT GUI

Variable
Id: samp_rate
Value: 512

Signal Source
Sample Rate: 512
Waveform: Cosine
Frequency: 2k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 30

Signal Source
Sample Rate: 512
Waveform: Sine
Frequency: 1k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 30

Add

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 512
Autoscale: Yes

Figure 1: Example layout.

Assignment 2
Signal representations in time and frequency domain

30.05.2021

The aim of the laboratory is to investigate the performance of systems representing signals in the time domain and in the frequency domain. In addition, analyse and compare the results obtained against each other and investigate the properties of the signal at fixed parameters.

Detailed documentation is available at the link https://www.gnuradio.org/docs/

To create a circuit to visualise a signal in the time domain, use the blocks learned in previous classes. However, to create a circuit to represent a signal in the frequency domain, use the following function blocks.

- FFT - main block implementing Fast Fourier Transform
  - FFT Size - the number of samples used to calculate the FFT
  - Forward/Reverse - whether to use FFT or iFFT
  - Window - what type of window to use for each set before FFT is performed, default option is blackmanharris, describes how many points should be in each window, size must match the value of FFT Size parameter
  - Shift - whether to perform FFT shift, can only be performed for complex input data
  - Num Threads - how many threads to engage for FFT calculation
- Frequency Sink - function block for displaying signals in the frequency domain
  - FFT Size - number of samples used to calculate FFT
  - Wintype - type of window
  - Center Frequency - the frequency to which the graph will be centered
  - Bandwidth - the bandwidth of the displayed signal
  - Name - title of the chart

Basic layout using FFT:

- Input signal
- Throttle block for real-time circuit generation
- Block FFT
- Vector Sink/Virtual Sink block for visualization

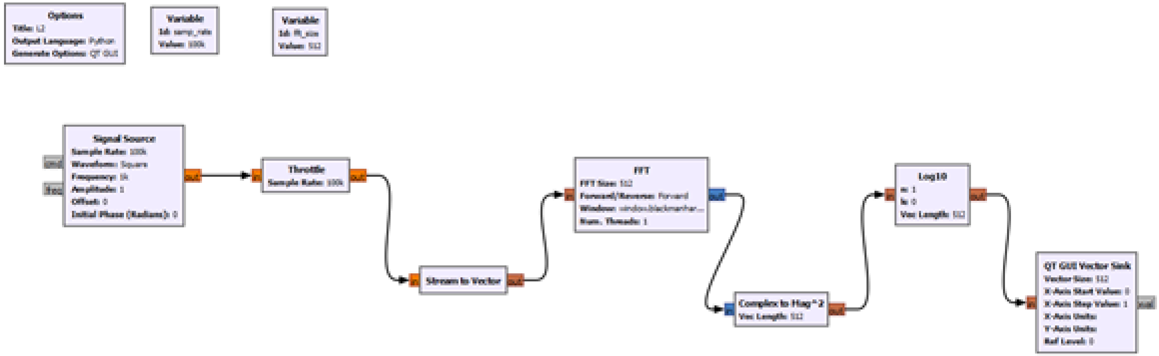
Basic layout using Frequency Sink::

- Input signal
- Throttle block for real-time circuit generation
- Frequency Sink visualization block

These systems can be extended with additional signal inputs, arithmetic/-logic operation blocks, modulation blocks (ASK, FSK, PSK)

Tasks

- Build 2 circuits with different parameters in the time domain using knowledge and function blocks from previous classes
- Add the possibility to display and visualise the designed signal
- Build 2 circuits with different parameters in the frequency domain based on function blocks from previous classes and the current instruction based on the FFT block
- Build 2 circuits with different parameters in the frequency domain based on function blocks from previous classes and from the current instruction based on the Frequency Sink block



Options
Title: L2
Output Language: Python
Generate Options: QT GUI

Variable
Id: samp_rate
Value: 100k

Variable
Id: fft_size
Value: 512

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

Throttle
Sample Rate: 100k

Stream to Vector

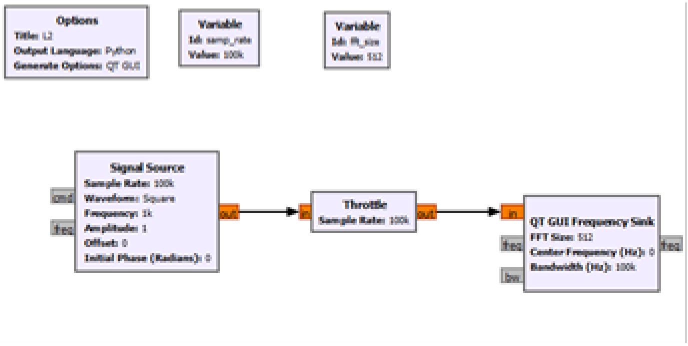
FFT
FFT Size: 512
Forward/Reverse: Forward
Window: window.blackmanhar...
Num. Threads: 1

Complex to Mag^2
Vec Length: 512

Log10
n: 1
k: 0
Vec Length: 512

QT GUI Vector Sink
Vector Size: 512
X-Axis Start Value: 0
X-Axis Step Value: 1
X-Axis Units:
Y-Axis Units:
Ref Level: 0

Figure 2: Example circuit using the FFT block.



Options
Title: L2
Output Language: Python
Generate Options: QT GUI

Variable
Id: samp_rate
Value: 100k

Variable
Id: fft_size
Value: 512

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

Throttle
Sample Rate: 100k

QT GUI Frequency Sink
FFT Size: 512
Center Frequency (Hz): 0
Bandwidth (Hz): 100k

Figure 3: Example circuit using the Frequency Sink block.

Assignment 3
Implementation of continuous amplitude and phase modulation systems

30.05.2021

The aim of the laboratory is to investigate the effect of the continuous modulations used on the properties of the output signal. Amplitude and angle modulation will be used for this. After designing the circuits and applying the modulations, the bandwidth of the signals should be evaluated and the signals should be analysed in the frequency domain.

Detailed documentation is available at the link https://www.gnuradio.org/docs/. Amplitude modulation in GNU Radio

- Input signal sources with different parameters
- Arithmetic operation (e.g. Multiply)
- Time Sink block for displaying a signal

Phase modulation in GNU Radio

- Input signal source
- Block Frequency Mod
- Throttle block
- Block to display the signal (Time Sink)

Tasks

- Build 2 circuits for amplitude modulation
  - Apply different input parameters to the signal
  - Display designed systems in the time domain and in the frequency domain
  - Compare the results obtained
- Build 2 circuits for phase modulation
  - Apply different input parameters to the signal
  - Display designed systems in the time domain and in the frequency domain
  - Compare the results obtained

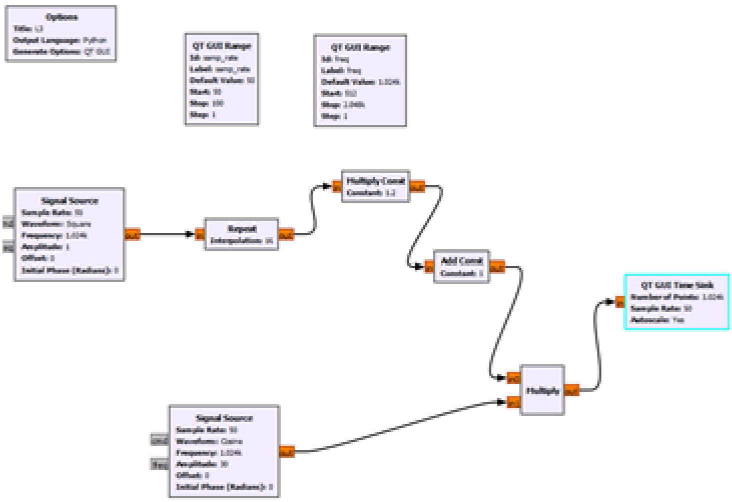


Figure 4: Example of system implementation for AM.

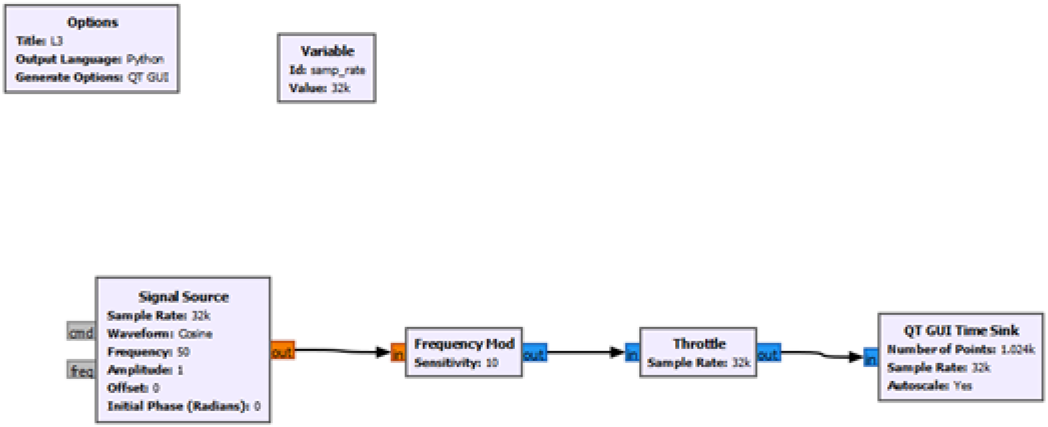


Figure 5: Example of system implementation for PM.

Assignment 3
Implementation of continuous amplitude and phase modulation systems

30.05.2021

The aim of the laboratory is to investigate the effect of the continuous modulations used on the properties of the output signal. Amplitude and angle modulation will be used for this. After designing the circuits and applying the modulations, the bandwidth of the signals should be evaluated and the signals should be analysed in the frequency domain.

Detailed documentation is available at the link https://www.gnuradio.org/docs/.
Amplitude modulation in GNU Radio

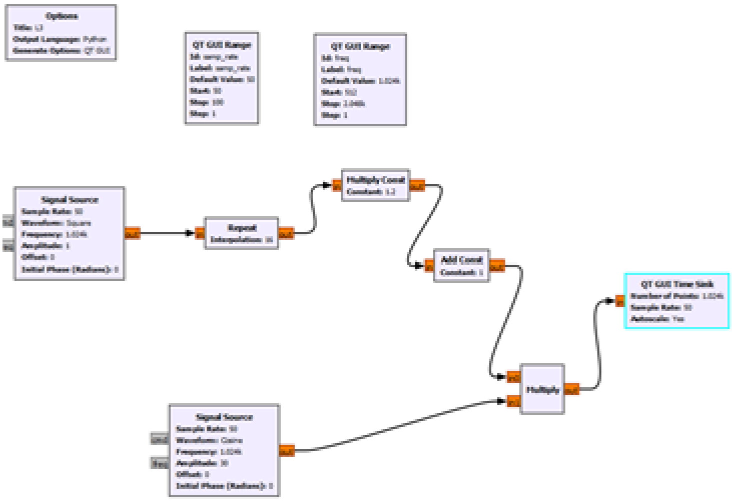
- Input signal sources with different parameters
- Arithmetic operation (e.g. Multiply)
- Time Sink block for displaying a signal

Phase modulation in GNU Radio

- Input signal source
- Block Frequency Mod
- Throttle block
- Block to display the signal (Time Sink)

Tasks

- Build 2 circuits for amplitude modulation
  - Apply different input parameters to the signal
  - Display designed systems in the time domain and in the frequency domain
  - Compare the results obtained
- Build 2 circuits for phase modulation
  - Apply different input parameters to the signal
  - Display designed systems in the time domain and in the frequency domain
  - Compare the results obtained



Options
Title: L3
Output Language: Python
Generate Options: QT GUI

QT GUI Range
Id: samp_rate
Label: samp_rate
Default Value: 50
Start: 10
Stop: 100
Step: 1

QT GUI Range
Id: freq
Label: freq
Default Value: 1.024k
Start: 512
Stop: 1.048k
Step: 1

Signal Source
Sample Rate: 50
Waveform: Square
Frequency: 1.024k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Repeat
Interpolation: 16

Multiply Const
Constant: 1.2

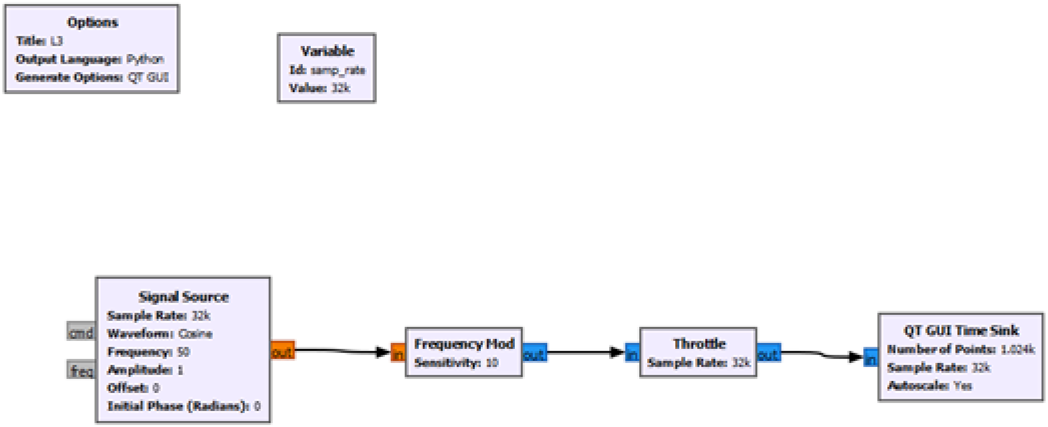
Add Const
Constant: 1

Multiply

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 50
Autoscale: Yes

Signal Source
Sample Rate: 50
Waveform: Cosine
Frequency: 1.024k
Amplitude: 10
Offset: 0
Initial Phase (Radians): 0

Figure 6: Example of system implementation for AM.



Options
Title: L3
Output Language: Python
Generate Options: QT GUI

Variable
Id: samp_rate
Value: 32k

Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 50
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Frequency Mod
Sensitivity: 10

Throttle
Sample Rate: 32k

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 32k
Autoscale: Yes

Figure 7: Example of system implementation for PM.

Assignment 4
Amplitude, phase, frequency keying

30.05.2021

The aim of the laboratory is to investigate the properties of a signal modulated using ASK, FSK and PSK modulation. Modulate and demodulate the input signal using selected keying. Then analyse the resulting signal waveforms.

Detailed documentation is available at https://www.gnuradio.org/docs/

ASK modulation:
On-Off Keying Modulation

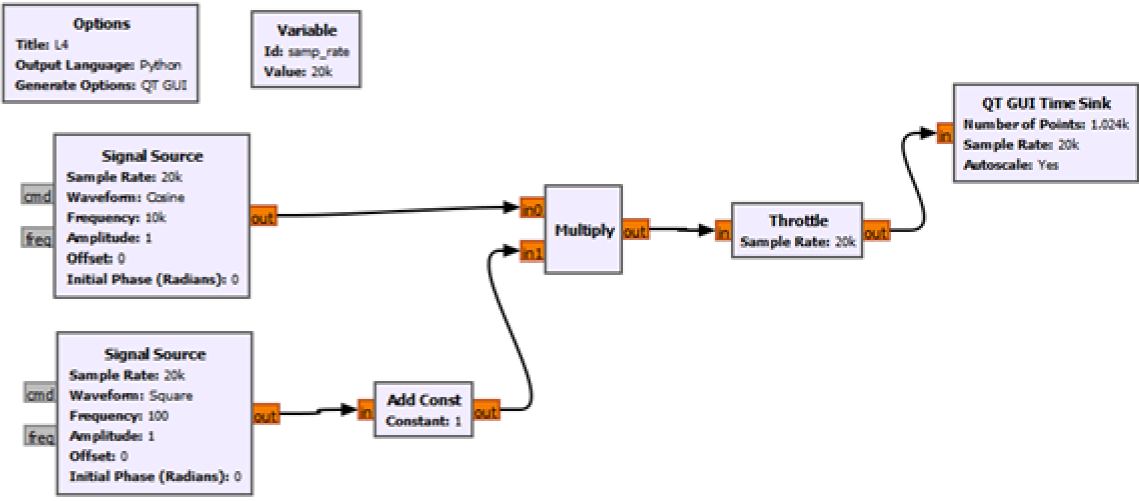
- Two inputs:
  - Carrier signal
    - Type: sinus
    - Frequency: 10KHz
  - Modulated signal
    - Type: square
    - Frequency: 100 Hz
    - A Slider block can be added for changing the values of signal parameters
  - Multiply block
  - Throttle block
  - Scope Sink visualization

ASK

- For system with On-Off Keying Modulation
  - Change the amplitude of the modulated signal (for example from 0,1 to 1,2) by adding ”Add const” blocks
  - Scope Sink visualization

Demodulation

- Blok Float To Complex to change data representation
- Blok Channel Model – to retransmit the signal
- Xlating FIR Filter block
  - Requires the definition of a variable with a given value:
    firdes.low_pass(1, samp_rate, fc, 25000, firdes.WIN_HAMMING, 6.76)
- Block Complex to Mag ^{2} to calculate the real and imaginary part



Options
Title: L4
Output Language: Python
Generate Options: QT GUI

Variable
Id: samp_rate
Value: 20k

Signal Source
Sample Rate: 20k
Waveform: Cosine
Frequency: 10k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Signal Source
Sample Rate: 20k
Waveform: Square
Frequency: 100
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Add Const
Constant: 1

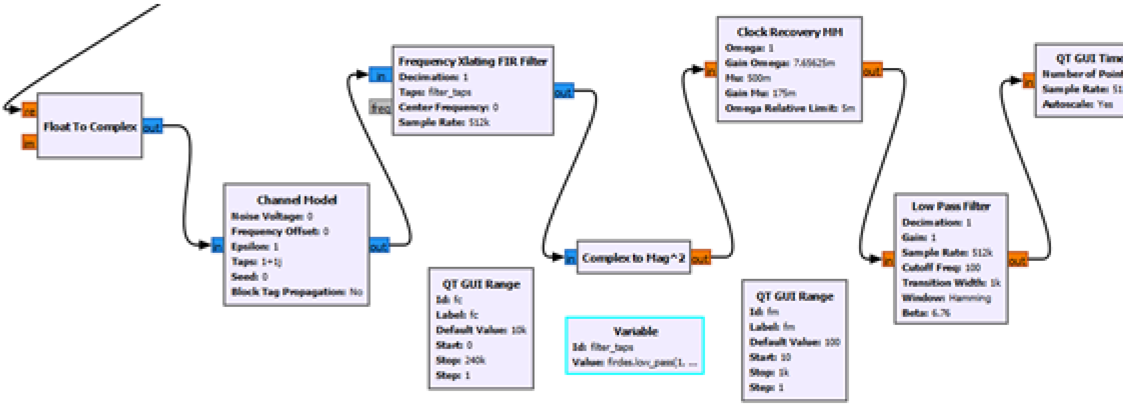
Multiply

Throttle
Sample Rate: 20k

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 20k
Autoscale: Yes

Figure 8: ASK modulation.

- Clock Recovery MM block for decoding symbols
- Low Pass Filter to filter the demodulated signal
- Scope Sink visualization



Float To Complex

Channel Model
Noise Voltage: 0
Frequency Offset: 0
Epsilon: 1
Taps: 1+1j
Seed: 0
Block Tag Propagation: No

Frequency Xlating FIR Filter
Decimation: 1
Taps: filter_taps
Center Frequency: 0
Sample Rate: 512k

QT GUI Range
Id: fc
Label: fc
Default Value: 10k
Start: 0
Stop: 240k
Step: 1

Complex to Mag^2

Variable
Id: filter_taps
Value: firdes.low_pass(1, ...

Clock Recovery MM
Omega: 1
Gain Omega: 7.65625m
Mu: 500m
Gain Mu: 175m
Omega Relative Limit: 5m

QT GUI Range
Id: fm
Label: fm
Default Value: 100
Start: 10
Stop: 1k
Step: 1

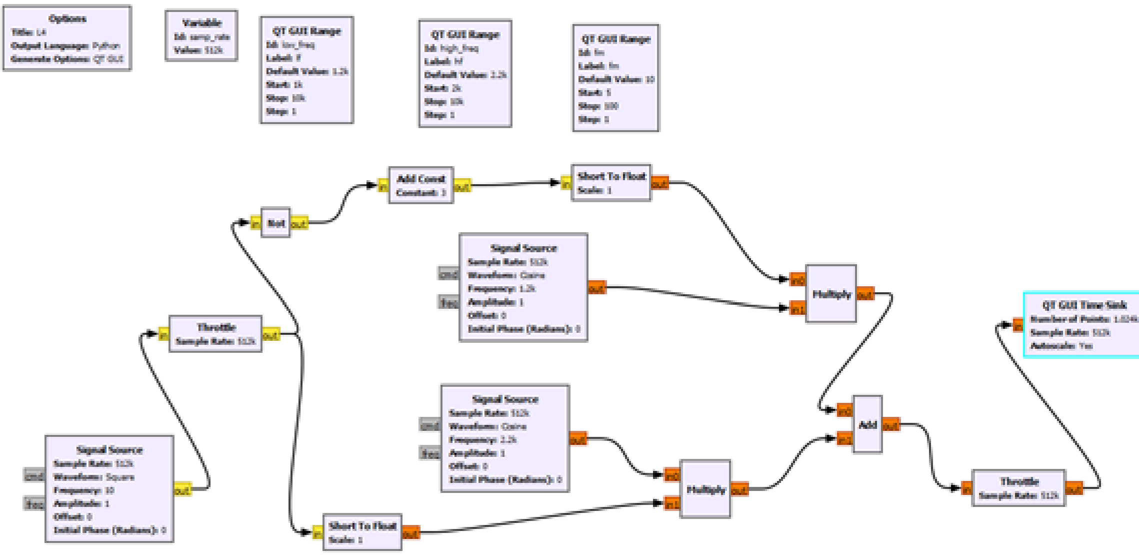
Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 512k
Cutoff Freq: 100
Transition Width: 1k
Window: Hamming
Beta: 6.76

QT GUI Time
Number of Poin
Sample Rate: 51
Autoscale: Yes

Figure 9: ASK demodulation.

FSK modulation:

- Modulated signal
  - Type: float, square
  - Frequency: 50 Hz
- Throttle block
- Short To Float to convert data
- Two input signals combined with a branched modulated signal using Multiply blocks
- Add block to sum the signals
- Throttle block
- Scope Sink visualization



Options
Title: L4
Output Language: Python
Generate Options: QT GUI

Variable
Id: samp_rate
Value: 512k

QT GUI Range
Id: low_freq
Label: lf
Default Value: 1.2k
Start: 1k
Stop: 10k
Step: 1

QT GUI Range
Id: high_freq
Label: hf
Default Value: 2.2k
Start: 2k
Stop: 10k
Step: 1

QT GUI Range
Id: fm
Label: fm
Default Value: 10
Start: 1
Stop: 100
Step: 1

Signal Source
Sample Rate: 512k
Waveform: Square
Frequency: 10
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Throttle
Sample Rate: 512k

Not

Add Const
Constant: 1

Short To Float
Scale: 1

Signal Source
Sample Rate: 512k
Waveform: Cosine
Frequency: 1.2k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Signal Source
Sample Rate: 512k
Waveform: Cosine
Frequency: 2.2k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Short To Float
Scale: 1

Multiply

Multiply

Add

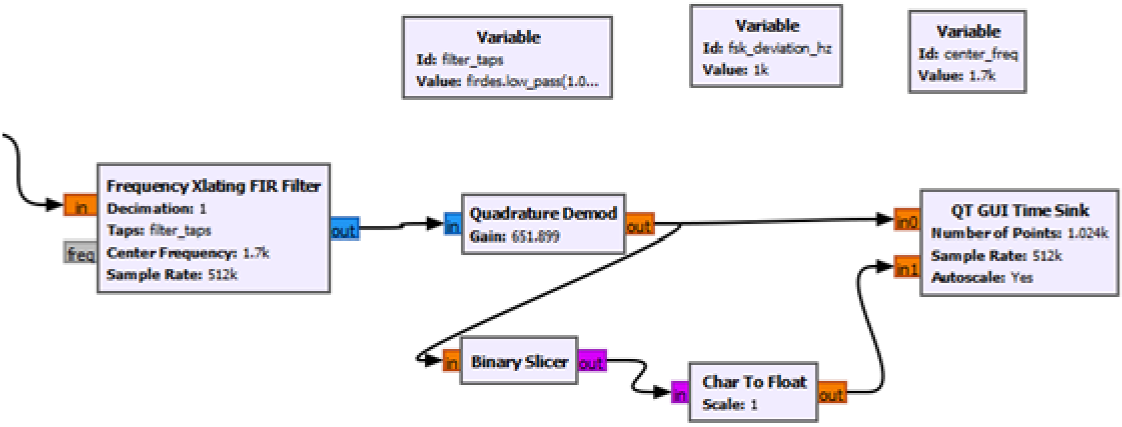
Throttle
Sample Rate: 512k

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 512k
Autoscale: Yes

Figure 10: FSK modulation.

Demodulation

- Xlating FIR Filter block
  - Requires the definition of variables with a given value:
  - firdes.low_pass(1.0,samp_rate,1000,400)
  - CenterFrequency = (Frequceny_Symbol_1 + Frequceny_Symbol_0) / 2
  - Where Frequency_Symbol_1/0 are variable names, e.g. for the Slicer_ID block
  - fsk_deviation_hz = Frequceny_Symbol_1 - Frequceny_Symbol_0
- Quadrature Demod block
- Binary Slicer block
- Char to Float conversion
- Scope Sink visualization with inputs
  - Quadrature Demod
  - Signal after data type conversion



Variable
Id: filter_taps
Value: firdes.low_pass(1.0...

Variable
Id: fsk_deviation_hz
Value: 1k

Variable
Id: center_freq
Value: 1.7k

Frequency Xlating FIR Filter
Decimation: 1
Taps: filter_taps
Center Frequency: 1.7k
Sample Rate: 512k

Quadrature Demod
Gain: 651.899

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 512k
Autoscale: Yes

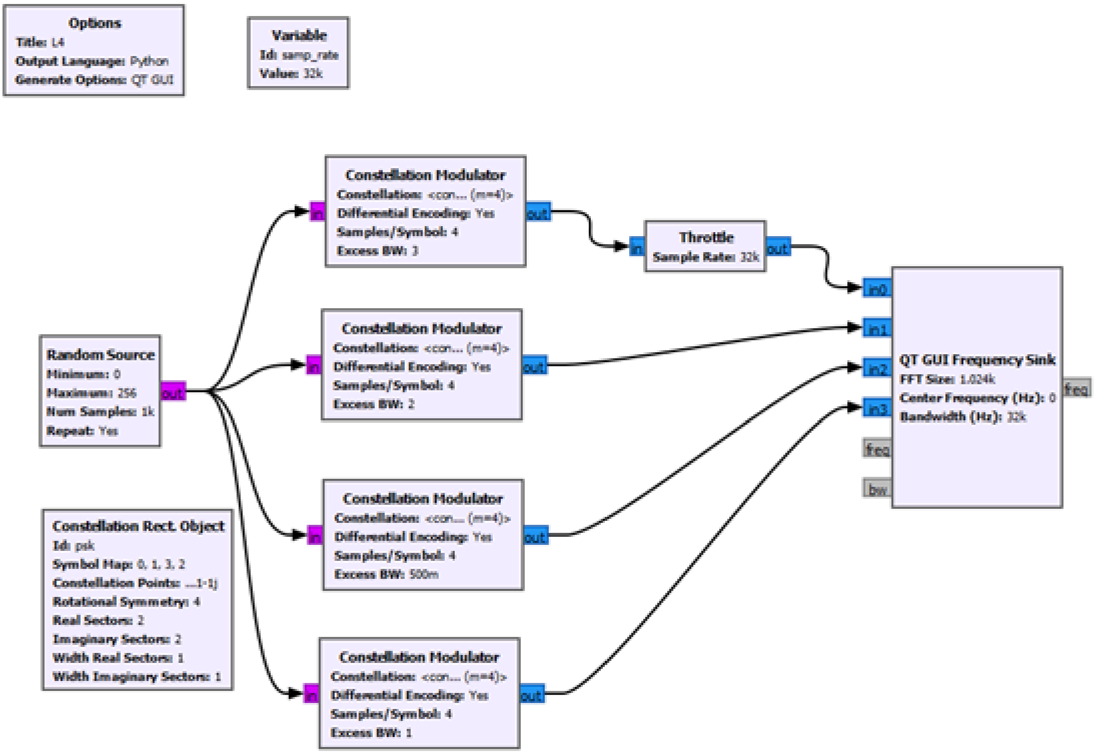
Binary Slicer

Char To Float
Scale: 1

Figure 11: FSK demodulation.

PSK modulation:

- Input signal source
- Block Constellation Rect Object
- Block Constellation Modulator
- Throttle block
- Scope Sink / Frequency Sink visualization



Options
Title: L4
Output Language: Python
Generate Options: QT GUI

Variable
Id: samp_rate
Value: 32k

Random Source
Minimum: 0
Maximum: 256
Num Samples: 1k
Repeat: Yes

Constellation Rect. Object
Id: psk
Symbol Map: 0, 1, 3, 2
Constellation Points: ...1-1j
Rotational Symmetry: 4
Real Sectors: 2
Imaginary Sectors: 2
Width Real Sectors: 1
Width Imaginary Sectors: 1

Constellation Modulator
Constellation: <con... (m=4)>
Differential Encoding: Yes
Samples/Symbol: 4
Excess BW: 3

Constellation Modulator
Constellation: <con... (m=4)>
Differential Encoding: Yes
Samples/Symbol: 4
Excess BW: 2

Constellation Modulator
Constellation: <con... (m=4)>
Differential Encoding: Yes
Samples/Symbol: 4
Excess BW: 500m

Constellation Modulator
Constellation: <con... (m=4)>
Differential Encoding: Yes
Samples/Symbol: 4
Excess BW: 1

Throttle
Sample Rate: 32k

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 32k

Figure 12: PSK modulation.

Tasks

- Implement the ASK modulation and demodulation circuit and examine the effect of changing the set parameters on the output
- Implement the FSK modulation and demodulation circuit and examine the effect of changing the set parameters on the output
- Implement the PSK modulation system and examine the effect of changing the Excess_BW parameter on the final result

Assignment 5
Distortion in the transmission channel

30.05.2021

The aim of the laboratory is to investigate the effect of distortion in the transmission channel. Test the efficiency of data transmission using the keying mechanisms already learned.

Detailed documentation is available at the link https://www.gnuradio.org/docs/.

The Channel Model function block can be used to generate interference and noise in the transmitted signal, where one of the adjustable parameters is the amount of noise.

Tasks

- Design an ASK modulation circuit based on the previous laboratories
  - Investigate the effect of changing the value of the Noise Voltage parameter on the reproduced signal
  - Remove Low Pass Filter block and check program operation
  - Remove Clock Recovery MM block and check the operation of the program
  - Remove Frequency Xlating FIR Filter block and check program operation
  - What was the effect of removing these blocks?
- Design an FSK modulation circuit based on the previous laboratories
  - Add a Channel Model block in the appropriate place to introduce noise into the reproduced signal
  - Examine the changes of the function used for the generated signal on the results
  - Examine the effect of the amount of noise on the resulting graphs
  - Examine the effect of changing the Low Frequency, High Frequency values on the results

Assignment 6
QPSK quadrature modulator

30.05.2021

The objective of the lab is to develop a circuit for digital modulation using code constellations. As part of the lab, the QPSK modulation circuit should be realised. In addition, the effect of the input parameters on the results should be examined.

Detailed documentation is available at the link https://www.gnuradio.org/docs/

Constellation Sink block - block for visualization of constellation points

To build a QPSK system you will need blocks:

- Generating a byte signal
- Converting bytes to floating point numbers
- Adding a constant value to a signal
- Root Raised Cosine Filter block
- Throttle block
- Converting floating point numbers to a complex number where one input is from one generated signal and the other input is from the next generated signal
- Constellation visualisation and visualisation in frequency domain

Tasks

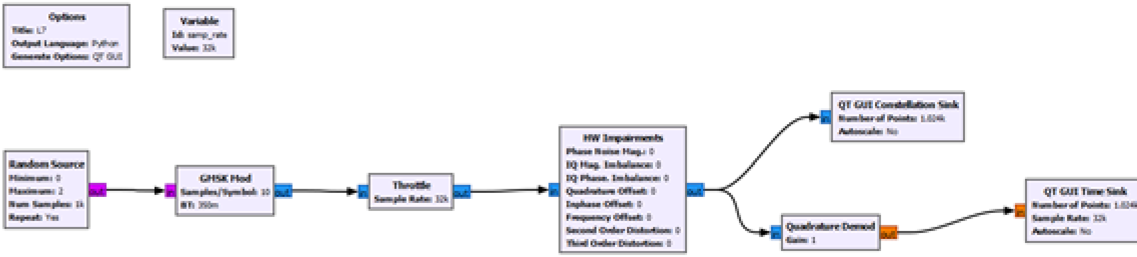
- Based on the components listed, build a QPSK system
  - Sample rate - 25k - used in Throttle, Root Raised Cosine Filter and Frequency Sink
  - Bit rate - 5k - used in Root Raised Cosine Filter
- Investigate the effect of changing parameters on system performance
- To generate the signal use
  - Signal Source
  - Random Source
  - Compare the signal sources in terms of the graphs obtained

Assignment 7
GMSK modulation

30.05.2021

The aim of the laboratory is to develop a GMSK (Gauusian Minimal Shift Keying) modulation system. The system should be designed and the influence of the input parameters on the obtained results should be investigated. In addition, the obtained results should be compared with selected modulations in the frequency domain.

Detailed documentation is available at the link https://www.gnuradio.org/docs/.



Options
Title: L7
Output Language: Python
Generate Options: QT GUI

Variable
Id: samp_rate
Value: 32k

Random Source
Minimum: 0
Maximum: 2
Num Samples: 1k
Repeat: Yes

GMSK Mod
Samples/Symbol: 10
BT: 350m

Throttle
Sample Rate: 32k

HW Impairments
Phase Noise Mag.: 0
IQ Mag. Imbalance: 0
IQ Phase. Imbalance: 0
Quadrature Offset: 0
Inphase Offset: 0
Frequency Offset: 0
Second Order Distortion: 0
Third Order Distortion: 0

QT GUI Constellation Sink
Number of Points: 1.024k
Autoscale: No

Quadrature Demod
Gain: 1

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

Figure 13: GMSK modulation system.

Tasks

- Design the layout shown above
- Insert user-modifiable variables into HW Impairments block
- Investigate the effect of changing parameter values on the graphs obtained
- Compare the results obtained with the modulations implemented in previous keying assignments

Assignment 8
FHSS spread spectrum system. Pseudo-field string generators

30.05.2021

The aim of the laboratory is to implement a spectrum spreading system using the FHSS method. In addition, implement pseudo-random sequence generators using linear shift registers LFSRs. LFSRs are binary string generators whose structure has good statistical properties.

Detailed documentation is available at https://www.gnuradio.org/docs/.

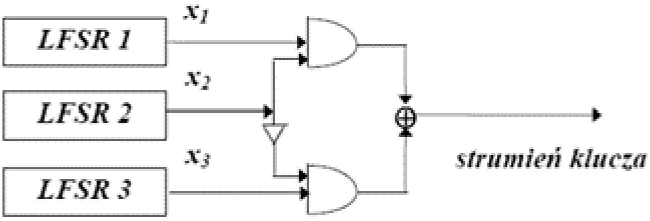


Figure 14: Structure of LFSR generators.

Tasks

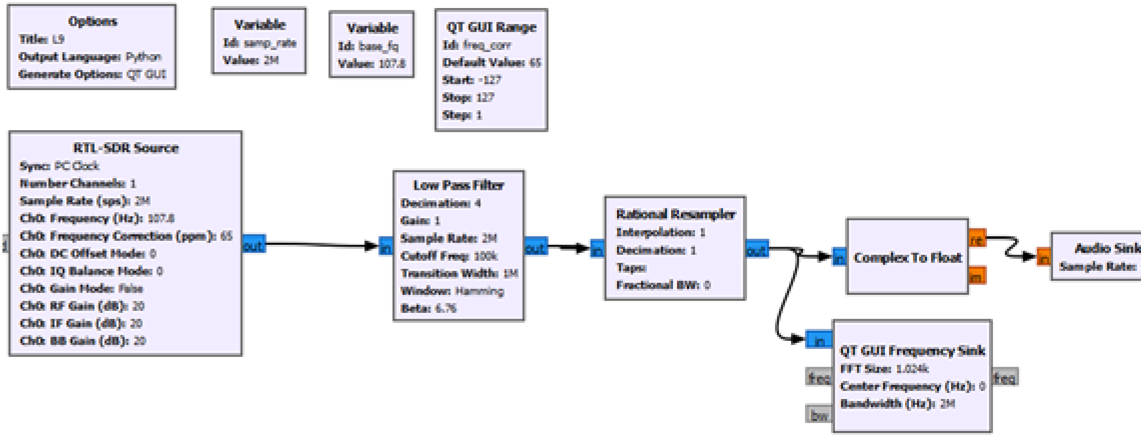
- Implement the Geffe generator
  - Use logical operation blocks
  - Use 2 different types of blocks as a signal source
    - Signal Source
    - Random Source
  - Compare the operation of the generator depending on the input used
  - Use Time Sink to visualization
  - Perform operations on the byte type
- Use the built-in GLFSR block and compare the results with the implemented Geffe generator

Assignment 9
Simulation of a radio circuit

30.05.2021

The aim of the laboratory is to simulate an example radio transmission with selected modulation schemes. In addition, analyse the selected digital modulation groups in terms of the results obtained.

Detailed documentation is available at the link https://www.gnuradio.org/docs/.



Options
Title: L9
Output Language: Python
Generate Options: QT GUI

Variable
Id: samp_rate
Value: 2M

Variable
Id: base_fq
Value: 107.8

QT GUI Range
Id: freq_corr
Default Value: 65
Start: -127
Stop: 127
Step: 1

RTL-SDR Source
Sync: PC Clock
Number Channels: 1
Sample Rate (sps): 2M
Ch0: Frequency (Hz): 107.8
Ch0: Frequency Correction (ppm): 65
Ch0: DC Offset Mode: 0
Ch0: IQ Balance Mode: 0
Ch0: Gain Mode: False
Ch0: RF Gain (dB): 20
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20

Low Pass Filter
Decimation: 4
Gain: 1
Sample Rate: 2M
Cutoff Freq: 100k
Transition Width: 1M
Window: Hamming
Beta: 6.76

Rational Resampler
Interpolation: 1
Decimation: 1
Taps:
Fractional BW: 0

Complex To Float

Audio Sink
Sample Rate:

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 2M

Figure 15: Basic radio transmission layout using RTL-SDR block.

Tasks

- Implement a basic radio circuit
- Examine the changes of input parameters on the result
- Add blocks corresponding to two selected digital modulations to the basic circuit
- Compare the results obtained for the implemented digital modulations

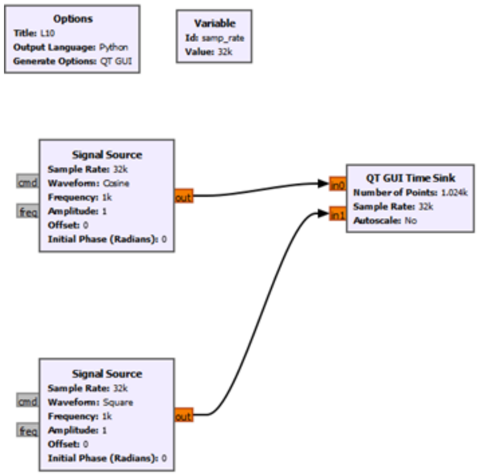
Assignment 10
GNU Radio transmission system in Python

30.05.2021

The aim of the lab is to implement a transmission system designed in GNU Radio, then generate code into Python and make changes to the model created.

Detailed documentation is available at the link https://www.gnuradio.org/docs/.

Depending on the operating system used (Windows or Linux), implement the following layout and then generate and review the code in Python language.



Options
Title: L10
Output Language: Python
Generate Options: QT GUI

Variable
Id: samp_rate
Value: 32k

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

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

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

Figure 16: Simple system example.

On Windows, this file will create itself when you run the designed model in GnuRadio. The .py file will be available under the name indicated in the project options in the location indicated earlier.

On Linux, after implementing the layout, press the button responsible for generating code for the Python language. This file will also be located in the previously indicated location where you saved your previous projects.

Tasks

- Implement the layout indicated above
- Analyze the generated code in Python
- Change the adjustable parameters in the generated code and run it and compare it with the results obtained from the GNU Radio program
    - In the absence of Python modules, install them

- Then choose 2 assignments made in this course
    - generate code in Python language
    - change selected parameters
    - compare the results obtained with those obtained by running the models in GNU Radio