# Introduction to Software Defined Radios

Manolis Surligas manolis@libre.space Libre Space Foundation & Computer Science Department, University of Crete

# This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License

https://gitlab.com/surligas/sdr-tutorial https://gitlab.com/surligas/gr-tutorial According to ITU-R SM.2152, Software-defined Radio (SDR) is:

A radio transmitter and/or receiver employing a technology that allows the RF operating parameters including, but not limited to, frequency range, modulation type, or output power to be set or altered by software, excluding changes to operating parameters which occur during the normal pre-installed and predetermined operation of a radio according to a system specification or standard."

#### The key characteristics are:

- Multiple communication standards support
- Easy and economic upgrades
- More sophisticated RF devices → Cognitive Radios
- Easy and rapid development, testing and deployment of new telecommunication standards
- Flexibility

Depending on the application requirements software may run in various execution environments

A typical SDR hardware architecture



A Superheterodyne transceiver



A Zero-IF front-end

Direct sampling receiver

#### **Complete SDR Platforms:**

- GNU Radio
- Matlab Simulink
- Pothos SDR

#### Libraries:

- Liquid-dsp
- VOLK
- itpp

- Software language can be arbitrary
- For realtime applications  $C/C{++}$ 
  - Often assisted by Single Instruction Multiple Data (SIMD)
- If the requirements are strict enough, FPGAs are used
  - High throughput
  - Low latency, low jitter
  - Predictable timings

## Introduction to GNU Radio

- GNU Radio is an open-source platform that provides signal processing blocks to implement software radios
- Core written in C/C++, some Python bindings
- Provides a GUI called GNU Radio Companion (GRC) to easily create software radio programs
- www.gnuradio.org

- It is highly recommended to install GNU Radio from the provided packages of your distribution
  - **Ubuntu-Debian:** *apt-get install gnuradio gnuradio-dev*
  - Fedora: yum install gnuradio gnuradio-devel
  - **OpenSUSE:** *zypper in gnuradio gnuradio-devel*
  - Pre-build Win64 images are also available

More info at https://wiki.gnuradio.org/index.php/InstallingGR

- Lets write our first software radio application with GNU Radio
- Firstly, open GNU Radio Companion or GRC

suntitled - GNU Radio Companion	~
Eile Edit View Run Iools Help	
🗗 🛅 🗙 📭 🛣 🗊 🖄 이 5 군 💷 😂 이 이 < > 1 🖉 🖉 이 🖹 이 이 5 군 1	
	(add)     (Eddian Operators)     (Eddian Operators)     (Obarnel Models)     (Coharnel Models )     (Deparcatel )     (Deparcatel )     (Eddians)     (Equators )     (Eddians)     (Eddi
	> [Idpc_ht]
	> [ Level Controllers ]
	> [ Math Operators ]
	> [ Measurement Tools ]
	> [ message rools ]
	> [ Misc ]
	> [ Modulators ]
	> [ Networking Tools ]

12

- This is the working area of GNU Radio
- A program based on GNU Radio is a scenario with multiple processing units connected each other. It is commonly called Flowgraph
- Each processing unit is called **Block**
- Ready to use blocks can be found at the left side of GRC window
  - **Ctrl+F** function is supported!

- The option block contains several parameters related with the flowagraph
- To reveal the properties of each block, double click on it
- The important to remember:
  - **ID:** The name of the Python executable that is going to be generated
  - Generate Options: QT GUI in case our flowgraph has a GUI element, NO GUI otherwise

÷.			Properties: Options 🗸 🗸 😣
General	Advanced	Do	cumentation
	ID Title		simple_example
	Author		
D	escription		
Canvas Size			OTGUI
00110	Run		Autostart V
Max Nu	mber of Outp	ut	0
Realtin	ne Schedulin	g	ott 🗸

- Now lets do some real work!
- Suppose we want to add two float signals into one and plot them at the time domain each one and their sum
- The first signal A will be a cosine with frequency of 2 kHz and the second signal B will be a sine of frequency 5 kHz
- Their maximum amplitude should be 1
- Search for a block called Signal Source
- Drag and drop it at the working area

• The result:

*untitled - GNU Radio Companion	
jile <u>E</u> dit <u>V</u> iew <u>R</u> un Iools Help	
그 🖿 🖶 🗙 📭 🖻 🛇   5 ਟ   🟴 은 히 은   < >   / / / @ 🖻   Q (	2 >-
	A Q sou X
Options Up to the second secon	Vaveform Generators ]
Generate Options: QT GUI	- Constant Source
	- Fast Noise Source
	- Noise Source
	- Random Source
	Signal Source
	GLFSR Source
Variable	~ [ Audio ]
U sampjace Value: 33K	- Audio Source
Signal Source	- Way File Source
Sample Rate: 32k	<ul> <li>[ Networking Tools ]</li> </ul>
Frequency: 1k	- TCP Source
Amplitude: 1 Offset: 0	UDP Source
	~ [ File Operators ]
	- File Descriptor Source
	- File Meta Source
	- File Source
	[Message Tools ]
	- Message Burst Source
	- Message Source
	~ [ Misc ]
	- Null Source
	- Vector Source
	- Bus Source
	- Bus Source Structure
	- Pad Source
	- Virtual Source
	~ [FCD]
	- Funcube Dongle Source
	~ [ieee80211]
	- char_device_source
	EEE80211 Unbuffered File 1
	[IEEE802.15.4]
	- IEEE802.15.4 SSDR DUMM
	EEEB02.15.4 SSDR PSDU
Jaing: "Mitairtni/gr-laser/examples.nast_UAQLgrc" >> Done	~ [UHD]
	UHD: USRP Async Msg Sc

16

- Drag and drop or copy and paste (yeah Ctrl+C Ctrl+V works on blocks!) and the second Signal Source
- Lets set properly their parameters by double click on each one

00	Properties: Signal Source 🛛 🗸 🔿 😣	🕴 Properties: Signal Source 🗸 🔨 😒
General Advanced	Documentation	General Advanced Documentation
ID	signal_A	ID signal_B
Output Type	Float 🗸	Output Type Float 🗸
Sample Rate	samp_rate	Sample Rate samp_rate
Waveform	Cosine 🗸	Waveform Sine v
Frequency	2e3	Frequency 5e3
Amplitude	1	Amplitude 1
Source - out(U): Port is not conn	sctad.	Source - out(D); Port is not connected.
	<u>✓ QK</u> <u>▲pply</u> <u>© cancel</u>	<u>✓ QK</u> <u>✓ Apply</u> <u>⊘ Cancel</u>

17

- Is our flowgraph ready? NO!
- Each flowgraph should have at least one source block and at least one sink
- Sources are blocks with only outputs. They only produce items
- On the other hand, sinks have only inputs. They only consume items
- We want to plot the time domain of the signals, so import a QT Time Sink block

- Make the appropriate configuration at the time sink block
- Float inputs, 3 different inputs, proper labels e.t.c

8	Properties: QT GUI Time Sink 🛛 🗸 🔨 🚫	8	Properties: QT GUI Time Sink 🛛 🗸 🔨 🗙
General Trigger Config	Advanced Documentation	General Trigger Config	Advanced Documentation
ID	qtgui_time_sink_x_0	Control Panel	Yes 🗸
Type	Float 🗸	Legend	Yes Y
Name	"Time Sink"	Line 1 Label	Circuit A
Y Axis Label	Amplitude	Line 1 Width	Signal A
Y Axis Unit	•	Line i widdi	
Number of Points	1024	Line 1 Color	Blue V
Sample Rate	samp_rate	Line 1 Style	Solid V
Grid	No v	Line 1 Marker	None 🗸
Autoscale	No v	Line 1 Alpha	1.0
Y min	1	Line 2 Label	Signal B
Ymax	1	Line 2 Width	1
Number of Inputs	3	Line 2 Color	Red 🗸
Update Period	0.10	Line 2 Style	Solid V
Disp. Tags	Yes 🗸	Line 2 Marker	None V
GUI Hint		Line 2 Alpha	1.0
		Line 3 Label	Signal A + Signal B
		Line 3 Width	1
		Line 3 Color	Green 🗸
		Line 3 Style	Solid 🗸
		Line 3 Marker	None v
		Line 3 Alpha	1.0
		Ciel: i=0/0)	

19

- Now we want to connect the output of each signal to the corresponding input of the time sink
- Piece of cake! Just click on the desired source and then at the target sink port!
- A connection is created. Move wherever you want the blocks. The connection follows!
- But wait! We want also the sum of Signal A and Signal B. No problem! Bring in an Add block.

- After connecting the addition block you may end in a situation depicted in the figure below
- Connections marked with read arrows are wrong and the flowgraph can not be generated into an executable



- In GNU Radio two connected ports MUST have the same size and type
- Each port's data type is marked with a different color
- To see the color mapping go to  $\boldsymbol{\mathsf{Help}}\Rightarrow \boldsymbol{\mathsf{Types}}$



- Just alter the data type of the addition block by changing its properties
- Input/output data types can by altered also by selecting the desired block and pressing the ↑↓keys



- No we are ready to generate the executable of the flowgraph
- To do this, click the **Generate** button
- You may need to save the flowgraph file first
- Unfortunately, during the generation of the executable a warning message appears

Warning: This flow graph may not have flow control: no audio or RF hardware blocks found. Add a  $Misc \Rightarrow Throttle$  block to your flow graph to avoid CPU congestion.

- Lets explain this warning
- The flowgraph does not include any hardware device with a specific rate of producing or consuming samples
- There is no way to slow down the flowgraph. It will execute in maximum speed taking all the CPU resources
- With all the CPU resources saturated, the host computer becomes unusable
- The solution is the use of a Throttle Block

#### Note!

When performing simulations, each flowgraph should have at least one throttle block.

- Throttle block will slow down each sample at the specified sampling period
- How it works:
  - Assume a sampling rate of 32 KSPS (Kilo-Samples per Second)
  - This means that the system should be able process 32000 samples each second
  - If the CPU freely executed the flowgraph may produce more samples per second
  - Throttle block, slows down the processing of samples by sleeping an amount of time after each sample
  - In our case the sample duration is  $\frac{1}{32000} = 31.25$  microseconds



- Add the throttle block and generate the flowgraph again
- Execute the flowgraph either pressing the **Execute** button, or running the generated python file form command line
- Show time!

#### Question 1

Almost every block takes as argument the sampling rate. Why? How the sampling rate is chosen?

#### Question 2

If the sampling rate is increased, how the throttle block will react? How about the CPU?

- Ok, that was a nice first example but a little boring
- Lets take as parameter the frequency of each signal
- To achieve that insert two QT GUI Range widgets
- Each one will specify the frequency of the corresponding signal source

- ID is used as variable name
- At the desired block, place the ID of the corresponding GUI widget at the parameter field
- As user changes from the graphical slider the frequency, the new value is automatically passed to the corresponding block

<b>*</b>		Properties: QT GUI Range 🛛 🗸 🔨 🚫			
General	Advanced	Documentation			
	<u>ID</u>	freq_a			
	Label	Signal A Frequency			
	Туре	Float 🗸			
De	fault Value	1e3			
	Start	1e2			
	Stop	???????			
	Step	1			
	Widget	Counter + Slider 🗸			
Mini	mum Length	200			
	GUI Hint				
Check "\$start <= \$value <= \$stop" did not evaluate. Check "\$start < \$stop" did not evaluate. Param - Stop(stop): Value "?972??? annot be evaluated: invalid syntax ( <string>, line 1)</string>					
		<u>✓ QK</u> <u>✓ Apply</u> <u>⊘ Cancel</u>			

8			Propertie	s: Signal Source		$\sim$ $\sim$ $\otimes$
General	Advanced	Doo	umentation			
	ID		signal_A			
OL	itput Type		Float	•		
Sa	mple Rate		samp_rate			
M	/aveform		Cosine		~	
F	equency		freq_a			
<u>A</u>	mplitude		1			
	Offset		0			
<u> </u>				1.04		
				<u> </u>	✓ Apply	⊘ <u>C</u> ancel

#### Question

Which should be the stop frequency at the slider properties?