Introduction to GNU Radio

Computer Science Department, University of Crete

Manolis Surligas surligas@csd.uoc.gr
September 22, 2016
What is GNU Radio?

- GNU Radio is an open-source platform that provides signal processing blocks to implement software radios
- Its Core is written in C/C++, some python bindings
- Provides a GUI called GNU Radio companion to easily create software radio programs
- www.gnuradio.org
What is Software Radio?

- A radio system which performs the required signal processing in software instead of using dedicated integrated circuits in hardware

**Advantages:**

- Easy and fast development (*C/C++ code versus chip manufacturing*)
- Great simulation platform
- One software radio can be used for a variety of applications
- One (generic) hardware to rule them all!
- New standard? No problem! Just write the code and apply the patch!
What is Software Radio?

- A radio system which performs the required signal processing in software instead of using dedicated integrated circuits in hardware

**Disadvantages:**

- CPU hungry
- Greater latency
How I install GNU Radio?

- 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`

- For Windows users...

```bash
http://gnuradio.org/redmine/projects/gnuradio/wiki/WindowsInstall
```

Good luck!
How I install GNU Radio?

- 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`

- For Windows users...
  - **Good luck!**
The lab session

- Ten PCs with Linux and everything you may need
- You **prepare** the assignments **at home**
- At the lab you just run them, ask questions, prove that you know what you are doing, get a grade

**Attendance:**
- Lab sessions are **obligatory**
- It is ok to loose one, but not two or more

**Cheating:**
- Plagiarism in the submitted code leads to a grade of **zero**
- If you help a classmate during the lab session, the grade of **both** cheaters is halved
The first GNU Radio application

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

- 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 first GNU Radio application

- 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

```
<table>
<thead>
<tr>
<th>General</th>
<th>Advanced</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Title</td>
<td>Author</td>
</tr>
<tr>
<td>Title</td>
<td>Description</td>
<td>Canvas Size</td>
</tr>
<tr>
<td>Author</td>
<td>Generate Options</td>
<td>Run</td>
</tr>
<tr>
<td>Description</td>
<td>Generate Options</td>
<td>Max Number of Output</td>
</tr>
<tr>
<td>Canvas Size</td>
<td>Run</td>
<td>0</td>
</tr>
<tr>
<td>Generate Options</td>
<td>Autostart</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Manolis Surligas (CSD, UoC)  
Introduction to GNU Radio  
September 22, 2016  
9 / 29
The first GNU Radio application

• 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 first GNU Radio application

- The result:
The first GNU Radio application

- 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
The first GNU Radio application

- 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**
The first GNU Radio application

- Make the appropriate configuration at the time sink block

- Float inputs, 3 different inputs, proper labels e.t.c
The first GNU Radio application

- 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.
The first GNU Radio application

- 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 cannot be generated into an executable
The first GNU Radio application

- 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 **Help ⇒ Types**
The first GNU Radio application

- Just alter the data type of the addition block by changing its properties

- Input/output data types can be altered also by selecting the desired block and pressing the ↑↓ keys
The first GNU Radio application: Throttling

- 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⇒Throttle block to your flow graph to avoid CPU congestion.
The first GNU Radio application: Throttling

- 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.
The first GNU Radio application: Throttling

- 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
The first GNU Radio application: Throttling

- 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!
The first GNU Radio application

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?
The first GNU Radio application: Interacting with user input

- Ok, that was a nice first example but a little boring
- Let's 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
The first GNU Radio application: Interacting with user input

- 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
The first GNU Radio application: Interacting with user input

Check "$\text{start} \leq \text{value} \leq \text{stop}\$" did not evaluate.

Check "$\text{start} < \text{stop}\$" did not evaluate.

Param: Stop(stop):
Value "????????" cannot be evaluated:
invalid syntax (<string>, line 1)
The first GNU Radio application: Interacting with user input

Question
Which should be the stop frequency at the slider properties?
The course code repository

- We will use a Git repository hosted on Github to provide code and existing flowgraphs

- It will updated frequently, so stay tuned!

- We will try for each lecture to provide corresponding GNU Radio flowgraphs

- https://github.com/surligas/gr-cs330
The course code repository

- We will use a Git repository hosted on Github to provide code and existing flowgraphs

- Folder *examples* contains complete, working examples

- Folder *apps* will be used for the assignments, may contain incomplete no working flowgraphs

- **Feel free to push your own interesting/boring/whatever flowgraphs**
 Lets hear some music!

- Use the fm_receiver flowgraph in the examples folder