From 0 to 6 GHz in 30 minutes – bootstrap your SDR Experience!
Start from scratch today, hack the EM spectrum tomorrow!

Marcus Müller

FOSDEM 2017

February 4, 2017
From theory to practice

Engineering in theory

- Analyze the situation
- Implement
- Improve
From theory to practice

Engineering in theory
- Analyze the situation
- Implement
- Improve

Engineering in Practice
- Get your tools ready
- Familiarize with both tools & Problem
- Implement
- Improve your Implementation, the tools, the environment
Introduction

Getting GNU Radio

From Idea to Implementation

Extending the Tools

Conclusion
Getting GNU Radio

Challenging:

- Dependencies
- Platforms
- Usage with concurrent, different versions
- Usage in cross-compilation
Getting GNU Radio

Challenging:

- Dependencies
  - Python
  - Boost
  - Qt
  - ZeroMQ
  - Thrift . . .

- Platforms

- Usage with concurrent, different versions

- Usage in cross-compilation
Getting GNU Radio

Challenging:

▶ Dependencies
▶ Platforms
  ▶ Linux
    ▶ Fedora / CentOS / RHEL
    ▶ Ubuntu / Mint / Debian
    ▶ Arch
    ▶ Gentoo
    ▶ ...
  ▶ OS X / macports
  ▶ Windows*
▶ Usage with concurrent, different versions
▶ Usage in cross- compilation
Getting GNU Radio

Challenging:

- Dependencies
- Platforms
- Usage with concurrent, different versions
  - Might want to install bleeding edge dependencies …
  - … might not want to mess up your workstation by installing bleeding edge stuff.
- Usage in cross-compilation
Optimal solution for dependency hell: **Well-kept distro packages**

- can’t keep $N_{\text{GNU Radio \ projects}} \cdot N_{\text{Distros}}$ binary packages up-to-date
- Good luck getting current Thrift e.g. on Ubuntu 14.04LTS
- Heck, this is a **Developers’ Meeting** – you’re here to build and mend things – **Offering a safe way to build from Source is a must**
Python Build & Overlay Managed Bundle System

- Manages Recipes &
- Installs those and their Dependencies
- Will install the recipes you asked for in a Prefix (think: VirtualEnv)
- Deals with
  - yum/dnf
  - apt
  - yaourt
  - macports . . .
- builds from source if required
It’s really easy to install GNU Radio into ~/prefix:

```
sudo pip install pybombs
pybombs recipes add gr-recipes
    git+https://github.com/gnuradio/gr-recipes.git
pybombs prefix init ~/prefix -a myprefix
    -R gnuradio-default
```

To activate that prefix:

```
source ~/prefix/setup_env.sh
```
I heard **Ecosystem**?! 

CGRAN (Comprehensive GNU Radio Archive Network) has lots of applications and tools other devs shared

<table>
<thead>
<tr>
<th>Name</th>
<th>Tags</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr-eventstream</td>
<td>scheduler, streams, bursty</td>
<td>The event stream scheduler</td>
</tr>
<tr>
<td>Receiver for Vaisala Weather Sonde</td>
<td></td>
<td>Receiver for Vaisala Weather Sonde</td>
</tr>
<tr>
<td>gr-pyqt</td>
<td>gui, plotting, pyqt, pyqwt</td>
<td>Python QT Plotters and Message Tools Repo</td>
</tr>
<tr>
<td>gr-pcap</td>
<td>pcap, packet</td>
<td>PCAP recording and playback</td>
</tr>
<tr>
<td>gr-microtelecom</td>
<td>hardware, source</td>
<td>Microtelecom's Perseus SDR source module</td>
</tr>
</tbody>
</table>
When we think about Signal Processing Systems, we picture something like:
GNU Radio Companion allows us to represent such logical block diagrams in hardware!
GNU Radio Companion allows us to represent such logical block diagrams in hardware!
Demo?
Let’s build a Demo!

Observing the whole LDP433 spectrum
- 25 channels, 433.075 MHz to 434.775 MHz
- channel spacing: 25 kHz
- Modulation: FM
- receiving and visualizing the whole spectrum
- demodulation and playback of a single channel
Spectrum Visualizer

- **Low Pass Filter**
  - Decimation: 1
  - Gain: 1
  - Sample Rate: 2.205M
  - Cutoff Freq: 25k
  - Transition Width: 12.5k
  - Window: Hamming
  - Beta: 6.76

- **Power Squelch**
  - Threshold (dB): -15
  - Alpha: 100u
  - Ramp: 0
  - Gate: No

- **NBFM Receive**
  - Audio Rate: 44.1k
  - Quadrature Rate: 2.205M
  - Tau: 75u
  - Max Deviation: 5k

- **Multiply Const**
  - Constant: 100

- **Audio Sink**
  - Sample Rate: 44.1kHz

- **UHD: USRP Source**
  - Samp Rate (Sps): 2.205M
  - Ch0: Center Freq (Hz): ...75M
  - Ch0: Gain Value: 50
  - Ch0: Antenna: RX2

- **QT Fosphor sink**
  - Center Frequency (Hz): ...25M
  - Span (Hz): 2.205M

- **QT GUI Time Sink**
  - Number of Points: 4.41k
  - Sample Rate: 44.1k
  - Autoscale: No
Spectrum Visualizer
Extending the Tools

- GNU Radio’s meant to be extended
- You can add your own blocks to the library!
- (Usually) the right way: Write an Out-Of-Tree Module!
Out-Of-Tree Modules

- self-contained, buildable source tree
- uses/links against GNU Radio
- can contain anything from a clever Flow Graph . . .
- . . . to a complete complex application (gr-ieee802-15-4, gr-air-modes, . . .)!
Out-Of-Tree Modules (short: OOTs) follow a typical folder structure.
There's a tool to provide you with that structure: `gr_modtool`!
Most important subcommands:

- `gr_modtool help`: Shows help
- `gr_modtool newmod`: Make a new OOT
- `gr_modtool add`: Add a new block to an existing OOT
How do we apply this?
Let’s go for another hands-on!

A speaking clock

What? A clock that, every $N$ seconds, says the current time.
How do we apply this?
Let’s go for another hands-on!

A speaking clock

What? A clock that, every \( N \) seconds, says the current time.

How? Using an existing text-to-speech program and python blocks.
How do we apply this?
Let’s go for another hands-on!

A speaking clock

What? A clock that, every $N$ seconds, says the current time.
How? Using an existing text-to-speech program and python blocks.
Why? Yes.
How should it look like?

clock
interval in seconds: 20

Audio Sink
Sample Rate: 44.1k
Extending the Tools

Using `gr_modtool`

```
gr_modtool newmod speakingclock
Creating out-of-tree module in
./gr-speakingclock...Done.
>Use 'gr_modtool add' to add a new block to this currently empty module.
cd gr-speakinggclock
gr_modtool add
GNU Radio module name identified: speakingclock
Enter block type: source
Language (python/cpp): python
...
```
Putting in the functional code

- We’ve added a python block, so the code is in `python/`
- we locate that file, and change the `work()` method
- `mkdir build; cd build; cmake ..; make; make install`
Writing a PyBOMBS recipe

Easy:

```plaintext
category: common
depends:
  - gnuradio
description: This is my OOT
gitbranch: master
inherit: cmake
source: git+https://github.com/myuser/myrepo.git
```

Make a Pull Request against gr-etcetera/gr-recipes, done!
Wrapping things up

- GNU Radio can be a beast to set up, but PyBOMBS eases the pain
- If your problem is easy to visualize as concatenation of signal processing steps, it’s probably easy to implement using the GNU Radio Companion
- It might happen that there’s no block to do what you want, so write your own
- Blocks live in OOTs, and sharing an OOT is fun & easy
Useful Links

GNU Radio project http://gnuradio.org
Guided Tutorials http://tutorials.gnuradio.org
CGRAN http://cgran.org
PyBOMBS http://github.com/gnuradio/pybombs

GNU Radio mailing list discuss-gnuradio@gnu.org
Registration & Archive: https://lists.gnu.org/mailman/listinfo/discuss-gnuradio