

GNU Radio

Wireless protocol analyses approach

Alex Verduin

July 2, 2008



UNIVERSITEIT VAN AMSTERDAM

System and Network Engineering

- ▶ Research question
- ▶ What is Software Defined Radio
 - ▶ Hardware
 - ▶ Software
- ▶ What is Radio Data System
- ▶ Approach
- ▶ Flowgraph of the RDS protocol
- ▶ Demo
- ▶ Conclusion and future work
- ▶ Questions

How can a system and network engineer use the USRP and GNU Radio to fulfill a wireless protocol analyses?

SDR explained

SDR Target		Where	What
Minimize	Antenna ↓	Hardware	USRP
Maximize	↑ Bits	Software	GNU Radio

Figure: Design principles of SDR

My setup

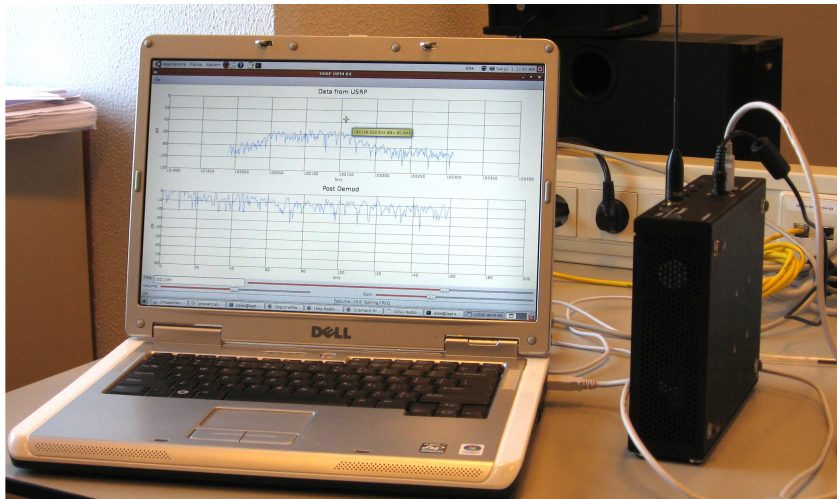


Figure: My setup

Universal Software Radio Peripheral(USRP)

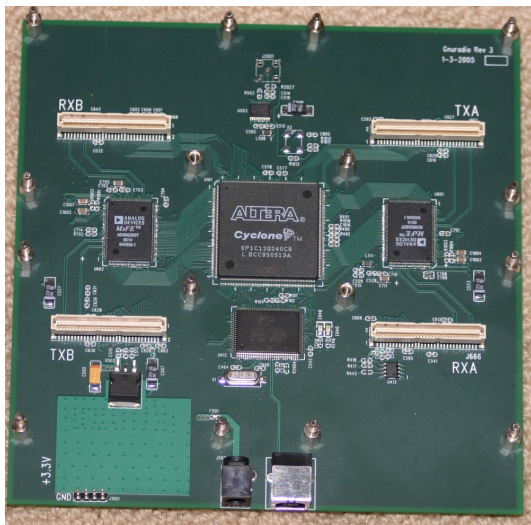
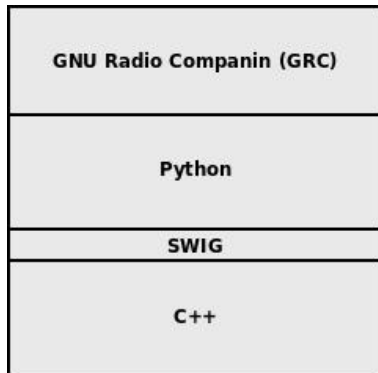
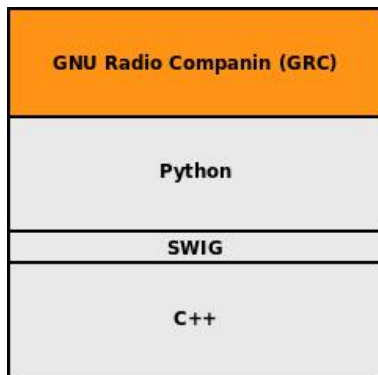


Figure: USRP



GNU Radio Software architecture



GNU Radio Software architecture

The screenshot displays the GNU Radio Companion (GRC) interface. The main workspace contains a flow graph with the following components:

- Options** (ID: options):
 - Title: untitled
 - Author: unknown
 - Description: gnuradio flow graph
 - Window Size: 1280, 1024
 - Generate Options: No GUI
- Variable** (ID: samp_rate):
 - Value: 32000
- Signal Source 1** (ID: gr_sig_source_x0):
 - Output Type: Float
 - Sample Rate: 32000
 - Waveform: Sine
 - Frequency: 350
 - Amplitude: 1
 - Offset: 0
- Signal Source 2** (ID: gr_sig_source_x):
 - Output Type: Float
 - Sample Rate: 32000
 - Waveform: Sine
 - Frequency: 400
 - Amplitude: 1
 - Offset: 0
- Audio Sink** (ID: audio_sink):
 - Sample Rate: 32KHz
 - Device Name:
 - OK to Block: Yes
 - Num Inputs: 2

The two signal sources are connected to the two input ports of the audio sink. The top source is connected to the top 'in' port, and the bottom source is connected to the bottom 'in' port. The 'out' ports of the sources are connected to the 'in' ports of the sink.

The right-hand side of the interface shows a component palette with the following categories and items:

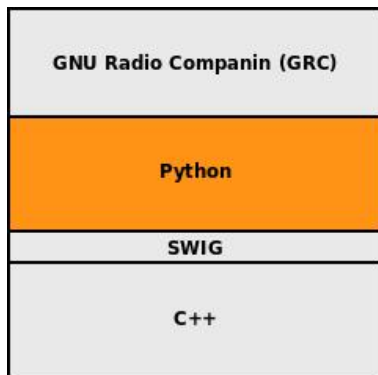
- Random Source
- GLFSR Source
- Null Source
- File Source
- UDP Source
- Audio Source
- Wav File Source
- Pad Source
- ▼ Sinks
 - Vector Sink
 - Null Sink
 - File Sink
 - UDP Sink
 - Audio Sink** (highlighted)
 - Wav File Sink
 - Pad Sink
- ▶ Graphical Sinks
- ▶ Operators
- + Add

The bottom status bar shows the following text:

```
>>> Warning: A connection can only be created between a source and an unconnected sink.  
>>> Warning: A connection can only be created between a source and an unconnected sink.  
Generating: "/home/alex/helloworld.py"
```

Figure: GNU Radio Comagnion

GNU Radio Software architecture



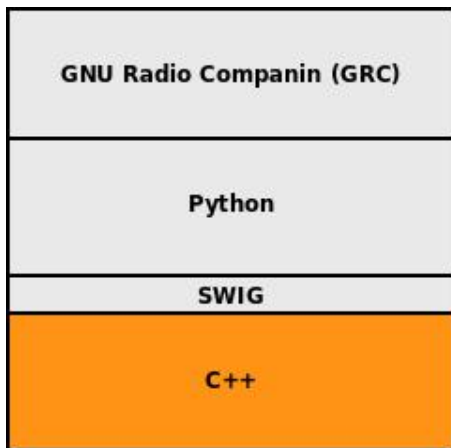
```
#!/usr/bin/env python

from gnuradio import gr
from gnuradio import audio

sampling_freq = 48000
ampl = 0.1

fg = gr.top_block ()
src0 = gr.sig_source_f (sampling_freq, gr.GR_SIN_WAVE, 350, ampl)
src1 = gr.sig_source_f (sampling_freq, gr.GR_SIN_WAVE, 440, ampl)
dst = audio.sink (sampling_freq)
fg.connect (src0, (dst, 0))
fg.connect (src1, (dst, 1))
fg.start ()
```

GNU Radio Software architecture



GNU Radio Software architecture

```
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
gr_rds_freq_divider.cc
60 */
61 gr_rds_freq_divider::gr_rds_freq_divider (int divider)
62 : gr_sync_block ("gr_rds_freq_divider",
63               gr_make_io_signature (MIN_IN, MAX_IN, sizeof (float)),
64               gr_make_io_signature (MIN_OUT, MAX_OUT, sizeof (float)))
65 {
66     d_divider = 0;
67     DIVIDER = divider;
68     d_sign_last = d_sign_current = false;
69     d_out = 1;
70 }
71 /*
72 * The virtual destructor.
73 */
74 gr_rds_freq_divider::~gr_rds_freq_divider ()
75 {
76     // -|
77 }
78 int
79 gr_rds_freq_divider::work (int noutput_items,
80                          gr_vector_const_void_star &input_items,
81                          gr_vector_void_star &output_items)
82 {
83     const float *in = (const float *) input_items[0];
84     float *out = (float *) output_items[0];
85     for (int i = 0; i < noutput_items; i++){
86
87         d_sign_current = (in[i] > 0 ? true : false);
88
89         if(d_sign_current != d_sign_last) {
90             // A zero cross
91             if(++d_divider == DIVIDER) {
92                 d_out *= -1;
93                 d_divider = 0;
94             }
95         }
96     }
97 }
```

Radio Data System (RDS)



Figure: RDS logo

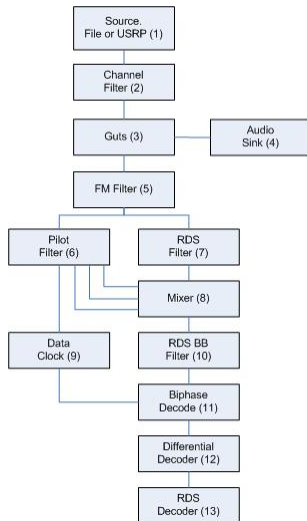


Figure: RDS example

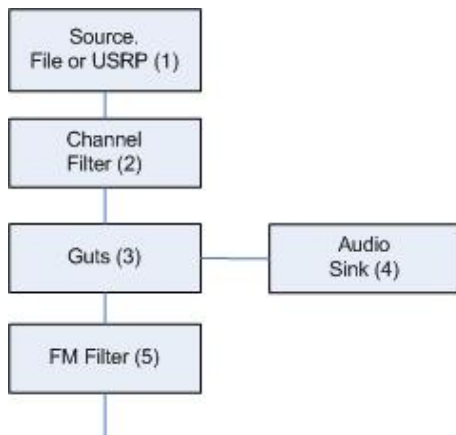
Approach

- ▶ Understand the design of the USPR and GNU Radio
- ▶ Install and try code examples
- ▶ Study protocol specifications and search for existing GNU Radio code
- ▶ Create flow graph
- ▶ Create testbed
- ▶ Capture raw samples
- ▶ Analyse the protocol

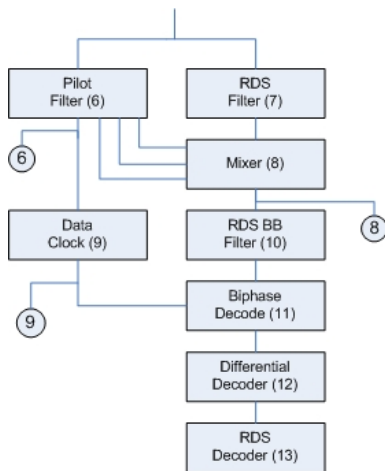
RDS Flow graph 1/3



RDS Flow graph 2/3



RDS Flow graph 3/3



Lets do some demoing!!

Conclusions:

- ▶ The defined approach works.
- ▶ Writing code, easy with Python, even more with GRC, difficult in C++
- ▶ The SNE'er needs some radio and SDR knowledge.
- ▶ Not all protocols can be fully analysed.

Future work:

- ▶ Analyse more protocols
- ▶ Extend the research with transmitting

Questions:

▶?