GNU Radio

Wireless protocol analyses approach

Alex Verduin

July 2, 2008

UNIVERSITEIT VAN AMSTERDAM

System and Network Engineering

Index

- 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

Research question

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

SDR explained

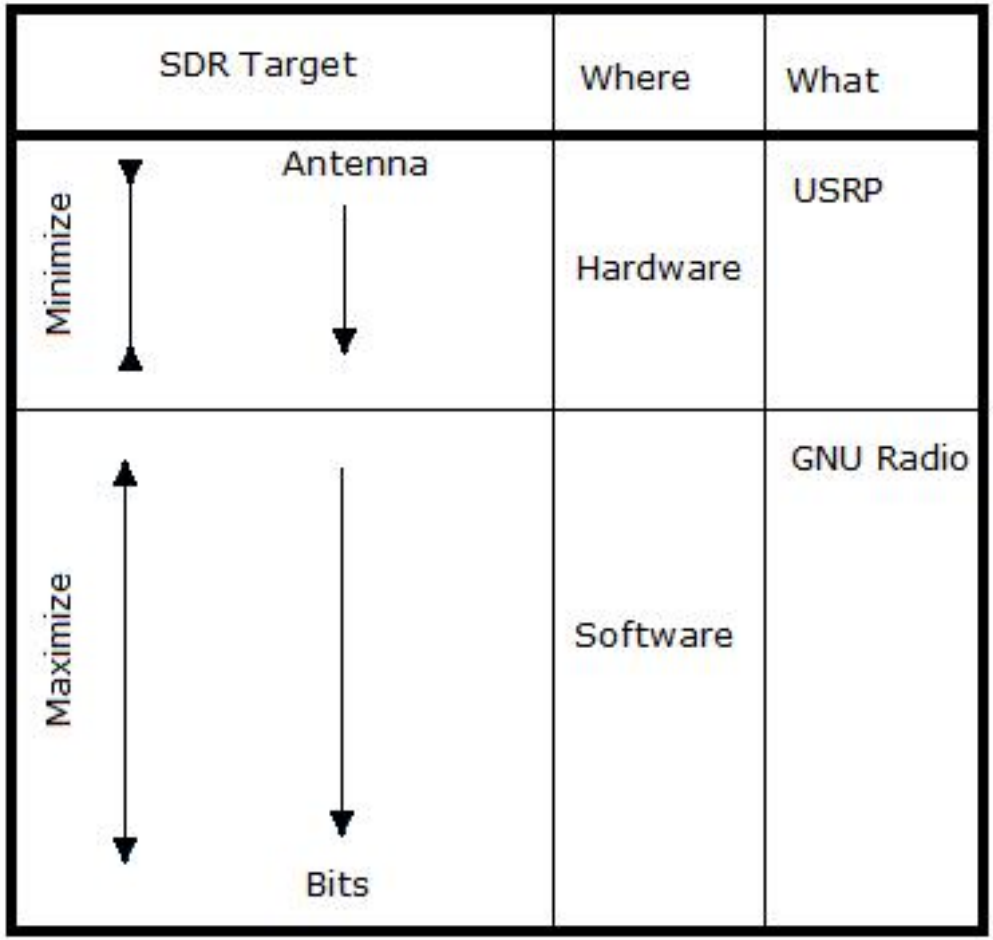


Figure: Design principles of SDR

My setup

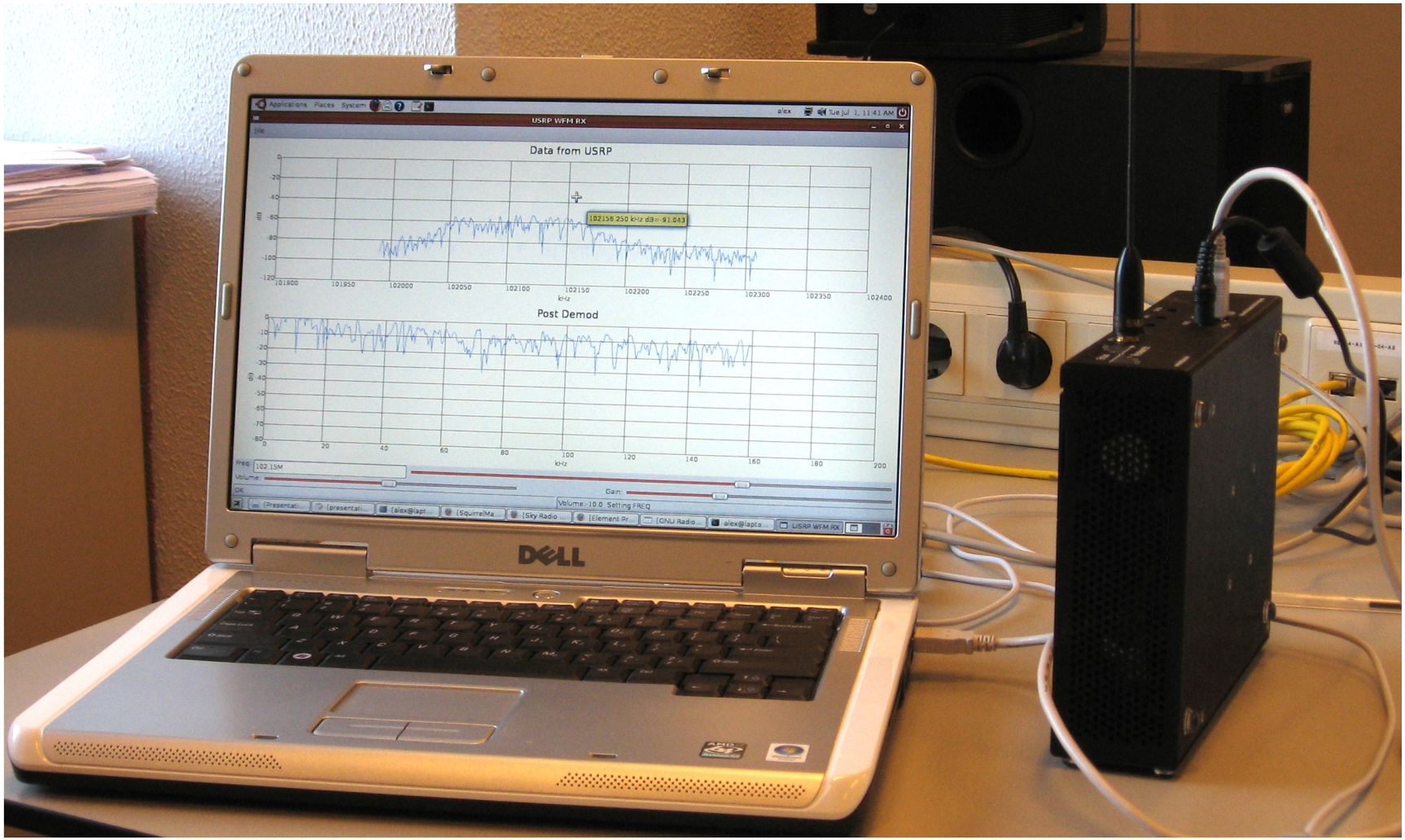


Figure: My setup

Universal Software Radio Peripheral(USPR)

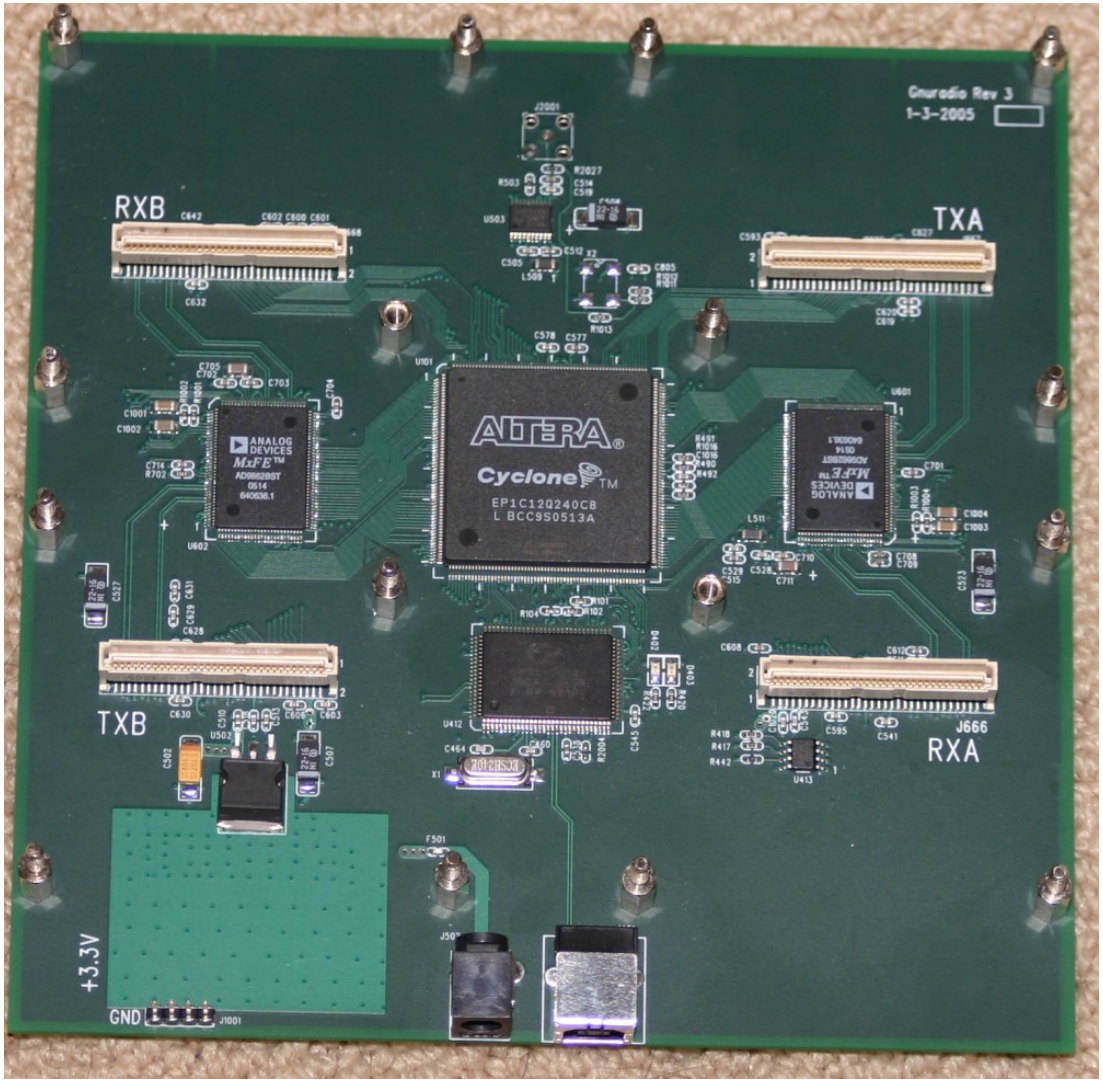
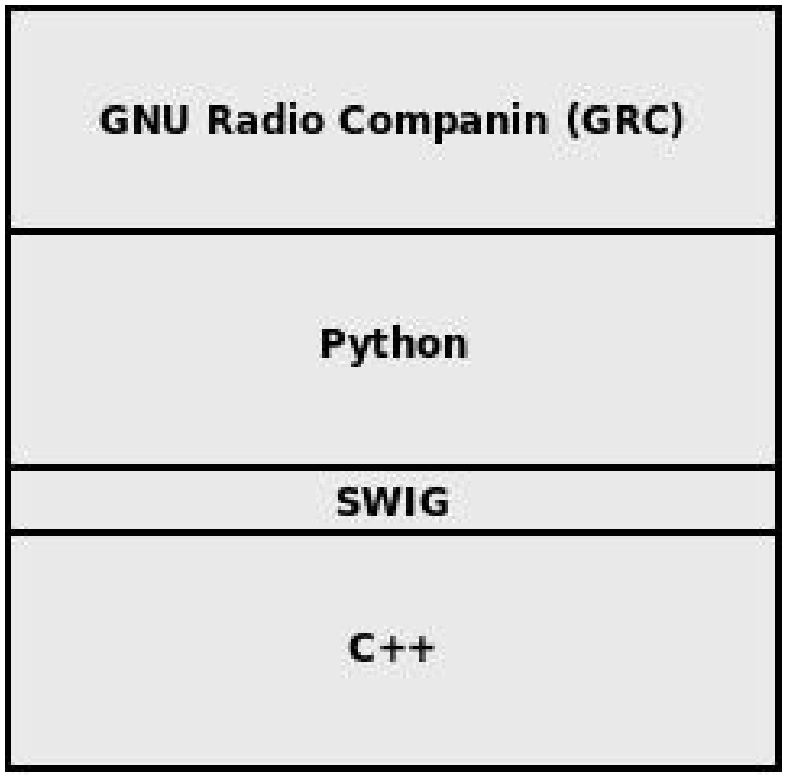
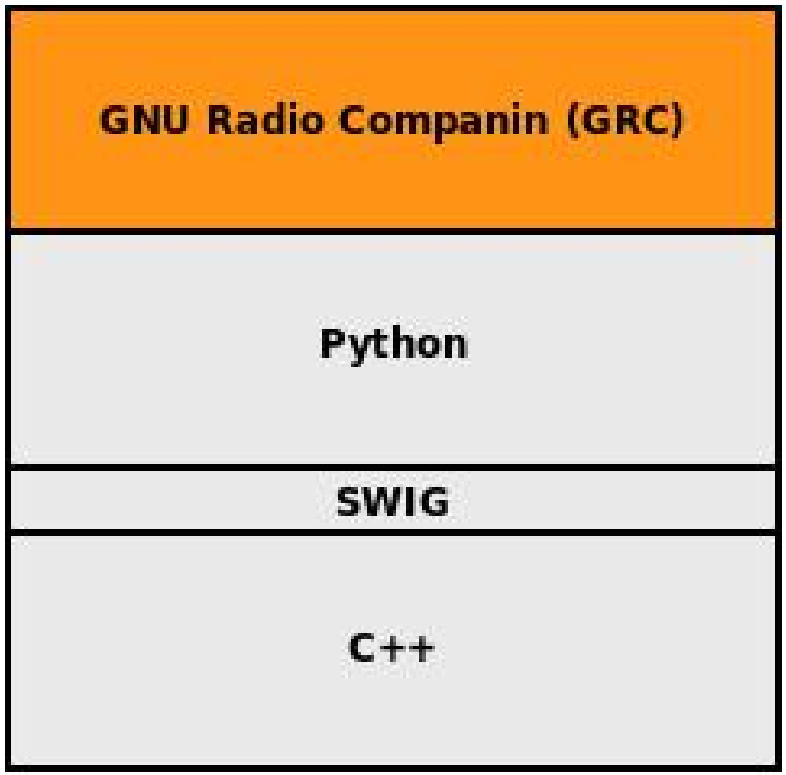


Figure: USRP

GNU Radio Software architecture



GNU Radio Software architecture



GNU Radio Software architecture

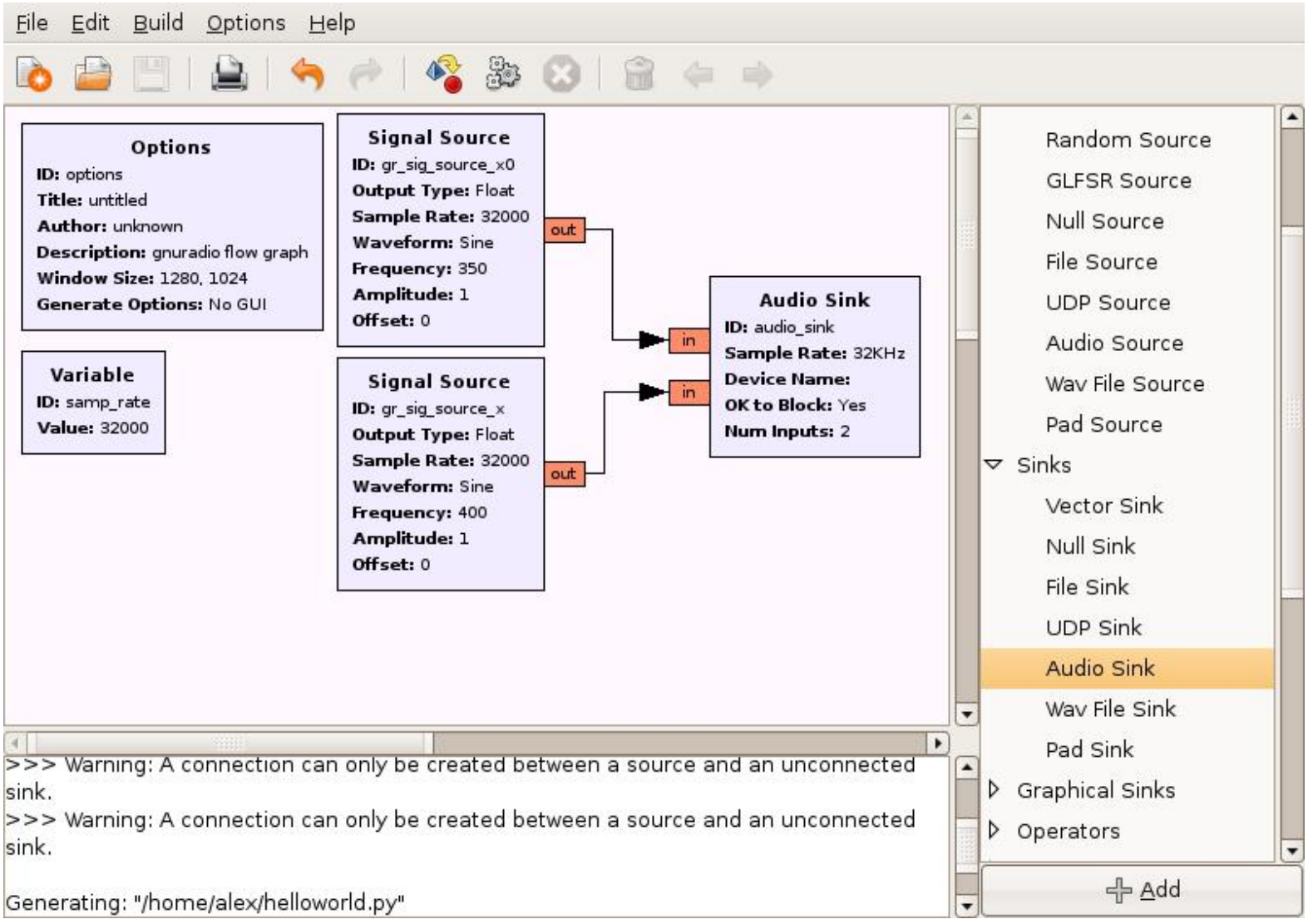
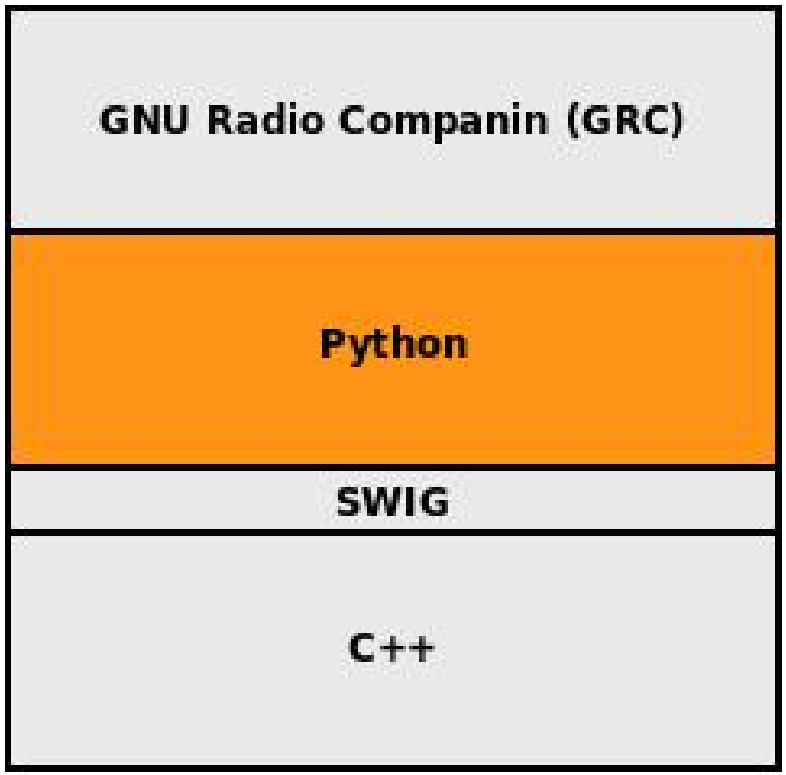


Figure: GNU Radio Comagnion

GNU Radio Software architecture



GNU Radio Software architecture

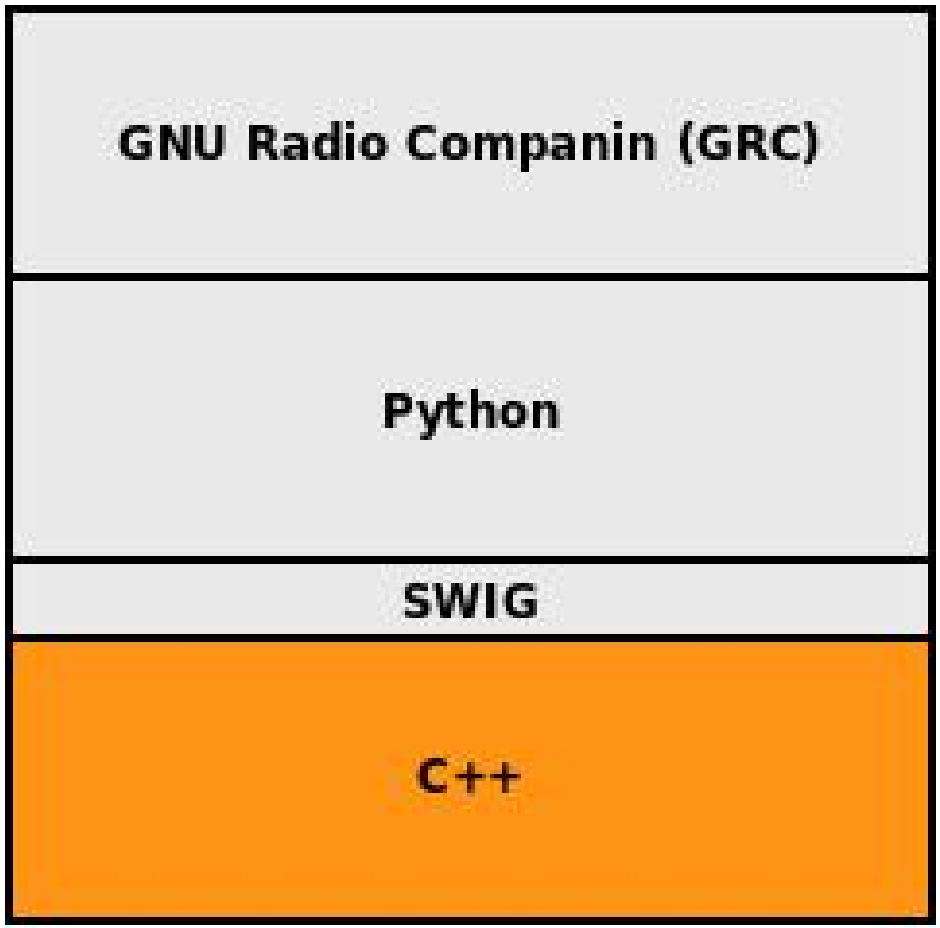
```
#!/usr/bin/env python

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;
```

Radio Data System (RDS)

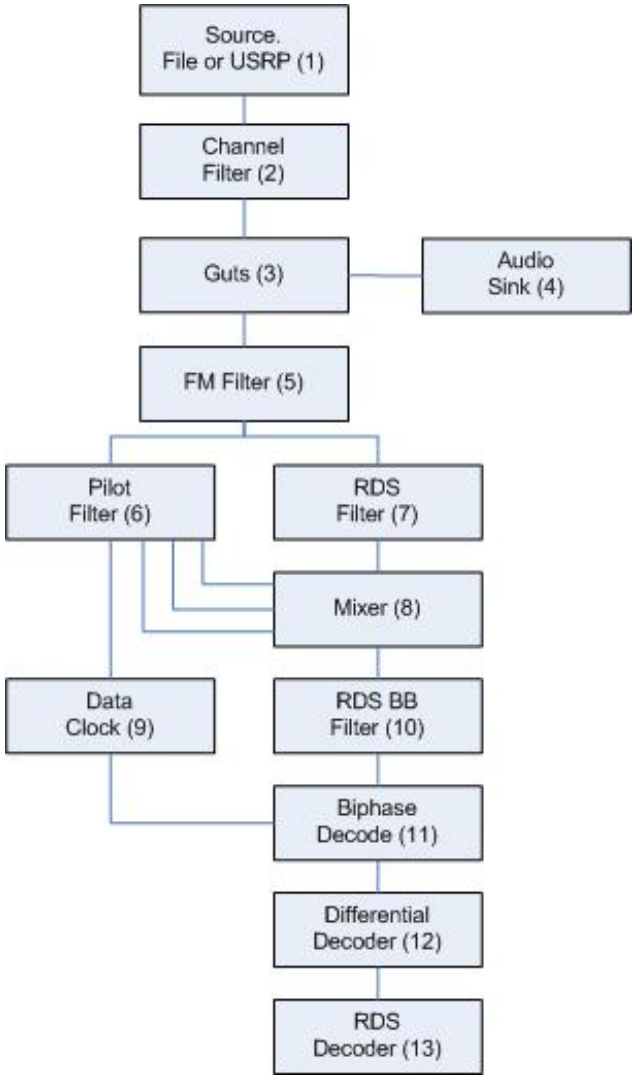
Figure: RDS logo

Figure: RDS example

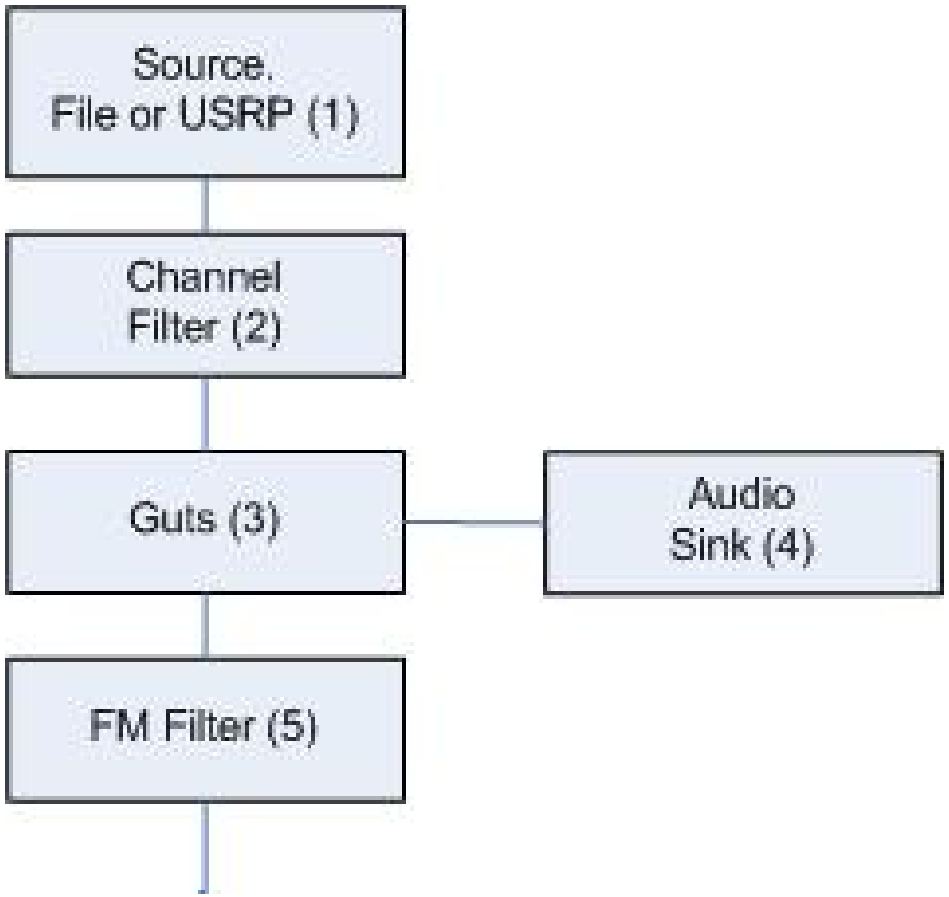
Approach

- 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

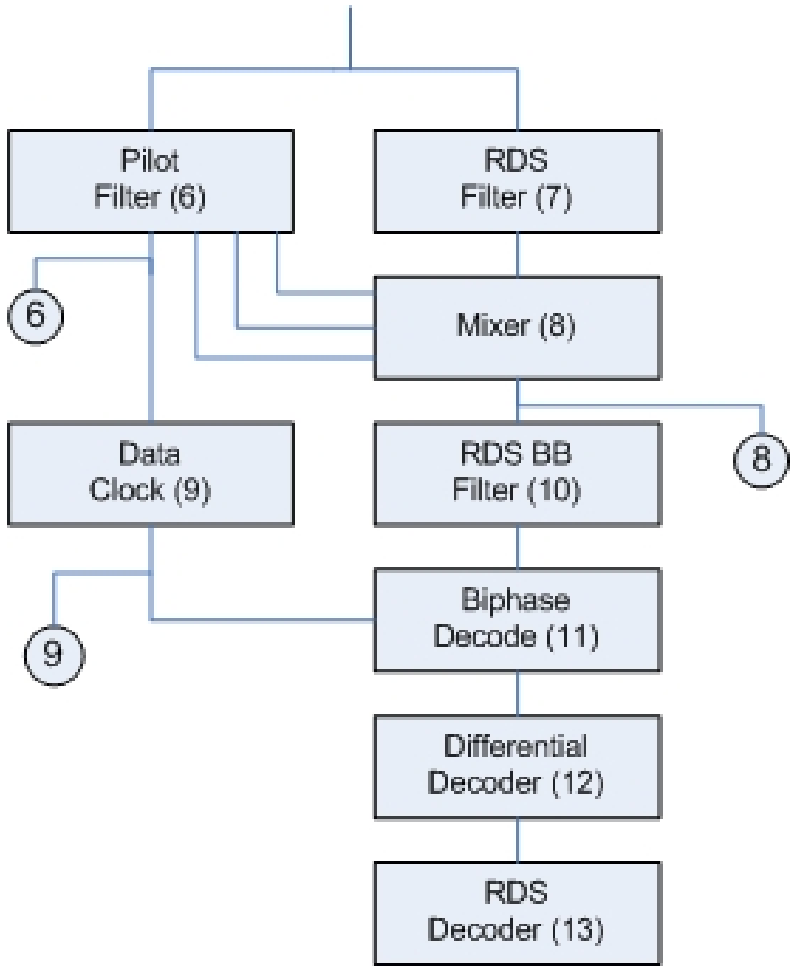
RDS Flow graph 1/3



RDS Flow graph 2/3



RDS Flow graph 3/3



Demo

Lets do some demoing!!

Demo

Conclusion and future work

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

Questions:

- .....?