

GPN #7

An **Introduction to  
GNU Radio  
Programming**

***Axel Rengstorf***  
***<axel.rengstorf@web.de>***

~ \$ whoami

- ***axel.rengstorf@web.de***
- freelancing security consultant
- “hacking” since 1995
- main focus on security of telephone systems

# Overview

- ***What is the GNU Radio***
- The Old Python/C++ way
- Message-Blocks (m-Block)

# What is the GNU Radio ?

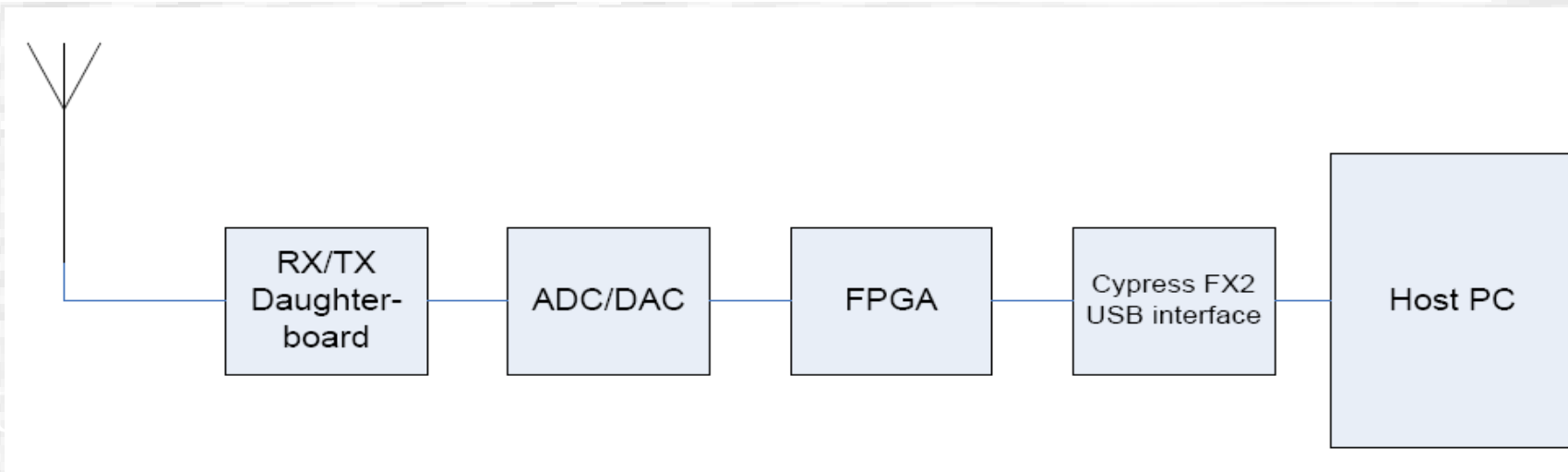
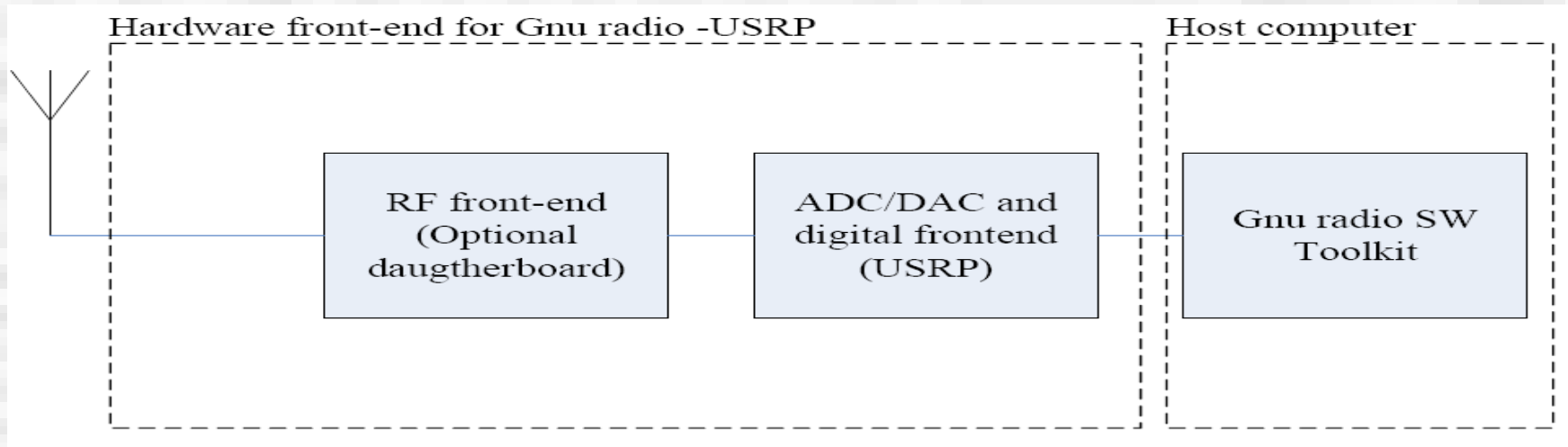
- Classic radio communication systems designed for speical purposes
- Lower layers like PHY, DLC can't be changed

## **Software Definied Radios** (SDR)

- ⇒ More flexible
- ⇒ Allows faster design of wireless protocols
  - ⇒ Easier academic research

***GNU Radio*** is a set of **software** signal processing building blocks that allow users to create their own software radio.

# What is the GNU Radio



# Overview

- What is the GNU Radio
- ***The Old Python/C++ way***
- Message-Blocks (m-Block)

# Oldschool way

- Flow of data realized as **flowgraph** as in graph theory
  - realized in Python
  - Edge = data
  - Vertex = GR Block realized in C++
  - dataflow from source to sink
  - SWIG library used for calling C++ from Python
- USRP related routines in Python
- Code running in GR Block can not interact with „Python-Space“

# Hello World

Hello World Example: Dial Tone Output

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

from gnuradio import gr
from gnuradio import audio

def build_graph ():
    sampling_freq = 48000
    ampl = 0.1

    fg = gr.flow_graph ()
    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, 0), (dst, 0))
    fg.connect ((src1, 0), (dst, 1))

    return fg

if __name__ == '__main__':
    fg = build_graph ()
    fg.start ()
    raw_input ('Press Enter to quit: ')
    fg.stop ()
```



# Oldschool way

- `gr.topblock()`
- `usrp.source_c`
- GR filter blocks
- GR blocks for modulation/demodulation
- Hierarchical blocks for aggregation

# Overview

- What is the GNU Radio
- The Old Python/C++ way
- ***Message-Blocks (m-Block)***

# Message Blocks (mblock)

- Motivation: associate metadata to data
  - describes the data
  - Example: timestamps, modulation schemes, power level
- Data as flow of messages
- no “Python-Space” anymore
- “real-time-scheduling” of message flows
- support for aggregation
- behaviour of mblock realized as

FSM

# Creating mblocks

- own mblock derived from mb\_block class
- must implement handle\_message()
- register the mblock with

**REGISTER\_MBLOCK\_CLASS(string of  
classname)**

# Ports

- ***#include <mb\_port.h>***
- provide I/O for a mblock
- mostly defined in the constructor with `define_port()`

```
mb_port_sptr  
define_port(const std::string &port_name,  
            const std::string &protocol_class_name,  
            bool conjugated,  
            mb_port::port_type_t port_type);
```

- `protocol_class_name`: specify what kind of data may come in/out of that port, functions in `mb_protocol_class.h`

# Protocol classes

- Command and status („status“)
- Data path („data“)
- Signalling („control“)
  - Event notification
  - Arrival of a packet
  - Notification from other entities

```
#include <mb_protocol_class.h>
```

```
mb_make_protocol_class(  
    pmt_intern("qa-send-cs"), // name  
    pmt_intern("status"),    // incoming  
    pmt_intern("control"),    // outgoing  
);
```

# Subcomponents

```
// create KEY->VALUE mapping structure
pmt_t usrp_dict = pmt_make_dict()

// Specify the RBF to use
pmt_dict_set(usrp_dict,
    pmt_intern("rbf"), // KEY
    pmt_intern("inband_2rxhb_2tx.rbf")); // VALUE

. . .

// define subcomponent of mblock
// (string component_name, string class_name, pmt_t user_arg)
define_component("server", "usrp_server", usrp_dict);
```

# USRP Server

- mblock\_class (usrp\_server)
- Part of inband-library
- Provides functionality for
  - Allocation/deallocation of physical channels
  - Receiving/sending of unmodulated raw samples
  - FPGA code handling
  - Handles the USB-interface



# Connecting the ports

```
Test_usrp_tx::test_usrp_tx(...)
{
    . . .
    // (src_name, src_portname, dst_name, dst_portname)
    connect("self", "tx0", "server", "tx0");
    connect("self", "cs", "server", "cs");
    . . .
}
```

- „self“ = special name for current mblock
- Connections can be changed dynamically

# Handle\_message()

- Must be defined in own mblock code
- Executed when there is data available at the port/s
- Implements the behavioral aspect of mblock
- `#include <mb_message.h>`

# Handle\_message(pmt\_data)

```
void cmac::handle_mac_message(mb_message_sptr msg)
{
    pmt_t event = msg->signal();          // type of message
    pmt_t data = msg->data();              // the associated data
    pmt_t port_id = msg->port_id();       // the port the msg was
    received on

    . . .
    switch (state)
    {
        . . .
        case (WAIT_PREAMBLE):
            if (pmt_eq(event, s_response_rcv_raw_samples)){
                // do some fancy stuff with data
            }
            break;
        case (WAIT_HEADER):
            . . .
    }
}
```

# Working with data

- `pmt_nth(pmt_t data, int n) // nth element`
- `pmt_to_XXX; convert data to type XXX (int, long, ...)`
- Uniform numeric vectors:
  - `pmt_t pmt_make_u8vector(size_t k, uint8_t fill);`
  - `pmt_t pmt_init_u8vector(size_t k, const uint8_t *data),`

# mb\_port::send() through ports

```
#include <mb_port.h>

class mb_port {
    . . .
    virtual void
        send(pmt_t signal,
             pmt_t data = PMT_F,
             pmt_t metadata = PMT_F,
             mb_pri_t priority = MB_PRI_DEFAULT) = 0;
```

# Start the whole thing...

- `#include <mb_runtime.h>`

```
int
main (int argc, char **argv)
{
    mb_runtime_sptr rt = mb_make_runtime();
    pmt_t result = PMT_NIL;

    pmt_t args = pmt_list2(
        pmt_from_long(strtol(argv[1], NULL, 10)),
        pmt_from_long(strtol(argv[2], NULL, 10))
    );

    rt->run("top", „test_usrp_rx“, args, &result);
}
```

# More info at ...

- mblock classes and headers: `gnuradio/mblock/*`
- PMT parameter handling: `gnuradio/pmt/*`
- Code examples: `gnuradio/usrp/host/apps-inband`

# Thats all folks...

# Questions?