Sniffing GSM signals for everyone

with gr-gsm and Multi-RTL

Piotr Krysik

Camp++
19 August 2016

About the speaker

whoami

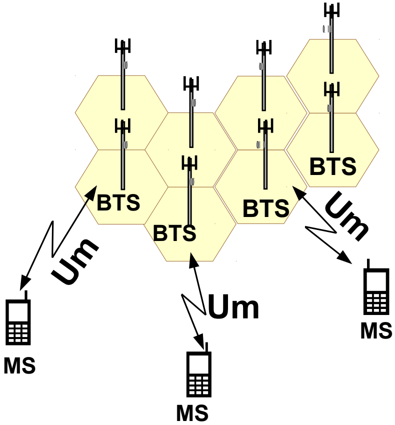
- author of the core part of gsm-receiver (most popular part of Airprobe)
- main author of gr-gsm - a GSM reception and decoding toolbox (successor of Airprobe)
- author of Multi-RTL - a RTL-SDR based multichannel receiver
- researcher at Warsaw University of Technology (Poland)
- a Free Software fan

gr-gsm

gr-gsm
What is it?

- Out-of-tree GNU Radio module
- set of tools for receiving, de-multiplexing, and decoding
- together with Wireshark enables analysis of live GSM transmission in the Um radio interface
- project's page: https://github.com/ptrkrysik/gr-gsm

gr-gsm
What is it?



Motivations to make a GSM receiver

Why people wanted to receive GSM themselves? (2007)

- more GSM terminals than nodes connected to the Internet
- no access to GSM physical layer
- knowledge of GSM - very rare, almost none in the FOSS world
- GSM network security research far behind Internet security research

GSM security

Tons of security weaknesses of GSM waiting for exploitation:

- no BTS authentication (exploited by security apparatus to this day)
- comically weak algorithm securing main Ki key - comp128v1 (replaces in early 2000's by comp128v2)
- weak design of A5/1 cipher securing signaling data and voice (based on LFSRs, relatively easily modeled mathematically, vulnerable to time-memory tradeoff, ...)
- availability of many offers of devices for breaking GSM security for government buyers
- GSM carriers living in denial of vast GSM insecurity
- difficulty of creating of a receiver presented as real a security measure

GSM security

Causes of GSM insecurity:

- long tradition of collusion of telecommunication companies with governments against the interests of end-users
  - dating back to times of national telecommunication monopolies
  - continuing to this day
  - power of telecommunication companies highly dependent on power of governments (and vice-versa)
  - example: symbiotic relation of AT&T and NSA (confirmed in 2015 after analysis of NSA's own documents that we have from Snowden)

GSM security

Causes of GSM insecurity (cont.):

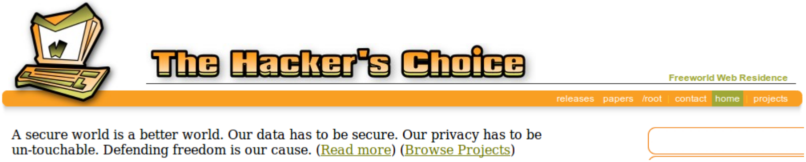
- committee that created GSM in 1980's concerned with possibility of TOO high users security *
  - British wanted 48 bit key on behalf of GCHQ and NSA
  - ended up with 54 bit key - 64 bits with last 10 bits zeroed
  - French didn't want encryption at all (A5/0 - "French Mode")
  - deliberately weakened A5/2 cipher for export to disliked countries
  - encryption could be turned off or switched to A5/2 mode, without the cell phone user knowing (by design)
- security through obscurity

* sources: account by Ross Anderson 1994
final confirmation: "We were pressured to weaken the mobile security in the 80's" 2014

Predecessors of gr-gsm - THC gsm cracking project (2007-2009)

Created three sub-projects for receiving GSM:

- GSSM by Joshua Lackey
- GSMSP
- GSM tvoid by Tempest Void - the most advanced one
  - had automatic frequency offset correction
  - simple equalization to fight inter-symbol-interferences (ISI)



Predecessors of gr-gsm - gsm-receiver (2009-2010)

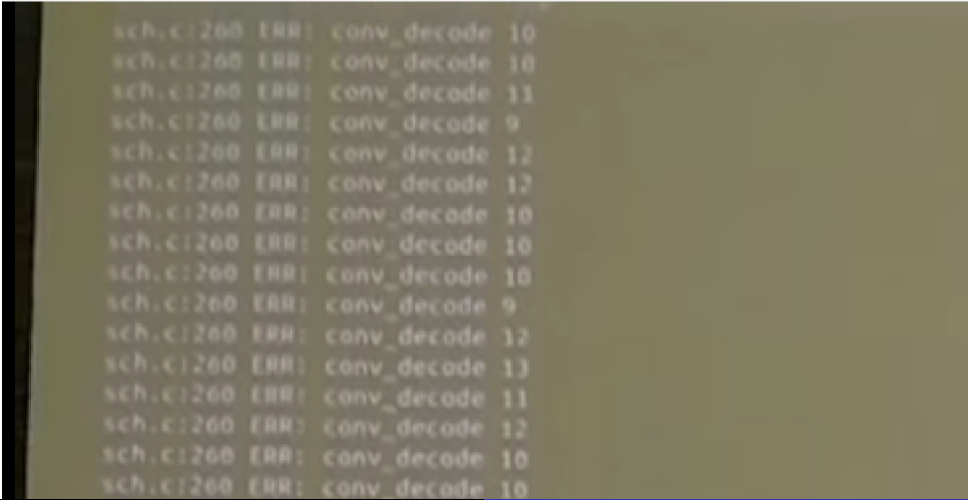
- at the beginning of 2009 THC's gsm project unexpectedly ends
- works on GSM sniffer continued under Airprobe project
- in the mid-2009 gsm-receiver was created by me and added to Airprobe
  - received GSM bursts the way it was intended (used training sequences and Viterbi equalization)
  - included CCCH, TCH/F (voice) channels decoding and decryption - all added as dirty hack that supposed to be removed promptly
  - patched by Harald Welte and Sylvain Munaut to add PCAP output and SDCCH/FACCH/SACCH decoding
  - the most popular part of Airprobe

Drawbacks of gsm-receiver

- never removed temporary hacks - gsm-receiver was meant for receiving GSM bursts only
- unresolved problems:
    - unstable frequency correction loop
    - not working synchronization of bursts inside TDMA frame with use of training sequences

Drawbacks of gsm-receiver

Drawbacks of gsm-receiver



End of gsm-receiver

- at some point I didn't have time to work on it
- in 2011 osmocom-bb project released
- osmocom-bb: implementation of gsm transceiver with use of Calypso based GSM phone (not general purpose SDR hardware)
- Calypso phones much cheaper than SDR hardware for gsm-receiver
- motivation to develop gsm-receiver project evaporated (although I made some attempts to restart)

gr-gsm - second life of gsm-receiver

end of 2013

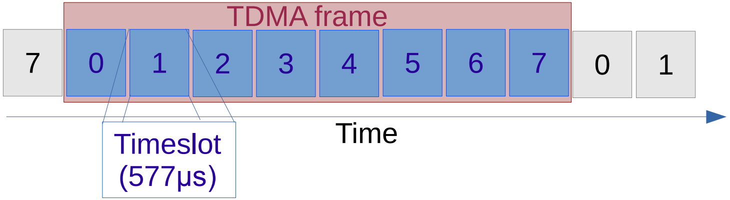
- I discovered that people still find gsm-receiver usable
- ... and are happy with it - despite its flaws
- ... and using it with cheap RTL-SDR receivers
- another attempt to correct synchronization in gsm-receiver - successful one :)
- the problem with frequency correction loop instability initially solved through removal of it
- Airprobe project unmaintained to the point it's not possible to access the repository
- April 2014 - gr-gsm project is born

gr-gsm currently

- main part - block for receiving bursts
  - without decryption and decoding related bloat
  - with support for frequency hopping and uplink demodulation
- modular design
  - separate blocks for logical channels de-multiplexing
  - separate blocks for decoding
- makes use of new GNU Radio capabilities: message passing, stream tags
- applications for:
  - scanning for base stations
  - live analysis of a single C0 channel
  - decoding of a signal file stored to a disk
- there are more capabilities in gr-gsm that are not covered by out of the box apps

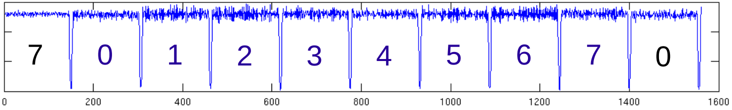
GSM radio link basics: TDMA frame

- TDMA multiplexing - signal divided into frames of 8 timeslots
  - Time-slot length = 156.25 symbols
  - Transmission rate = 270.83 kbits/s
- GMSK modulation
  - Frequency modulation
  - Constant envelope

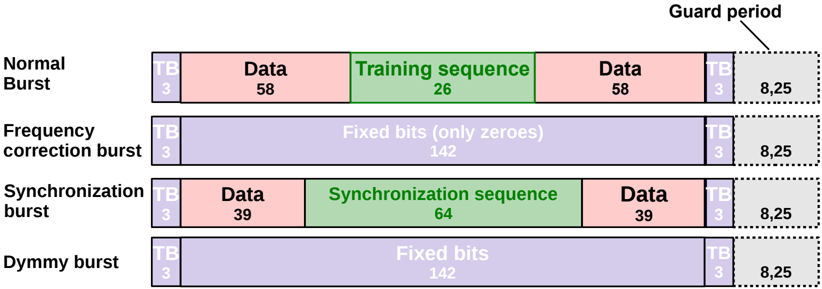


GSM radio link basics: TDMA frame

- TDMA multiplexing - signal divided into frames of 8 timeslots
  - Time-slot length = 156.25 symbols
  - Transmission rate = 270.83 kbits/s
- GMSK modulation
  - Frequency modulation
  - Constant envelope



GSM radio link basics: GSM bursts

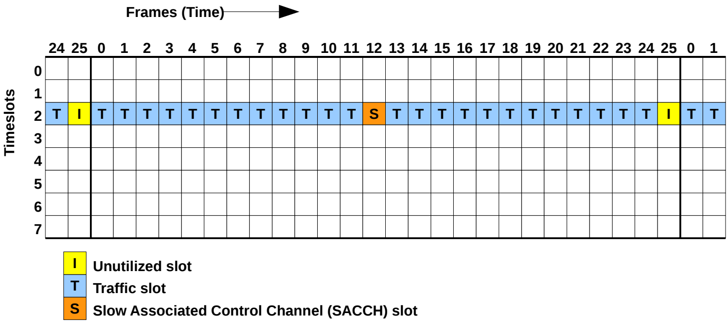


GSM radio link basics: logical channels

- Timeslot - a physical channel
- Multiple logical channels in (few) predefined groups inside timeslots

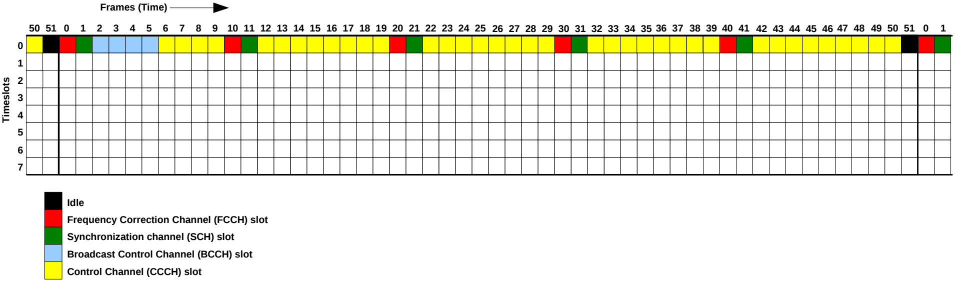
GSM radio link basics: logical channels

Traffic channel



GSM radio link basics: logical channels

Broadcast channel

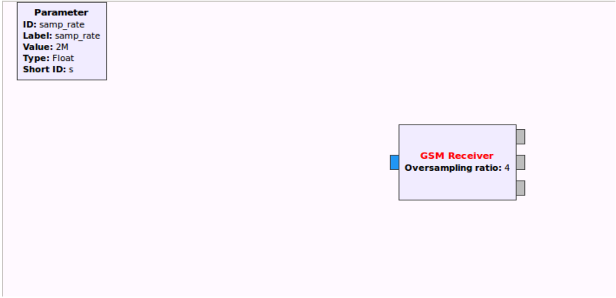


GSM components available in gr-gsm: receiver

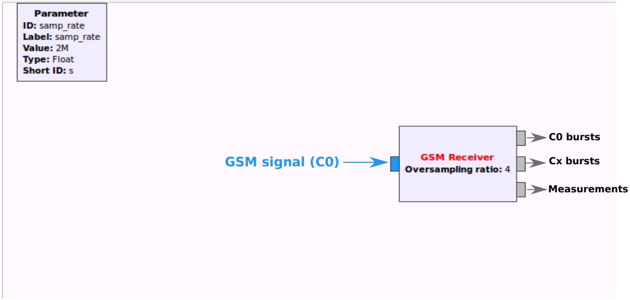
Receiver:

- transforms oversampled GSM signal (usually 4 times) into GSM bursts
- on the first input takes GSM broadcast (C0) signal
- works according to a simple algorithm
  1. find frequency correction burst (FCCH)
  2. find synchronization burst (SCH burst one frame after FCCH burst)
  3. synchronously process busts
     - use FCCH for carrier frequency offset measurements
     - use SCH bursts to keep synchronization
     - in case of synchronization loss go to (1)
- can process frequency hopping and uplink (but this isn't integrated with existing apps yet)

GSM components available in gr-gsm: receiver



GSM components available in gr-gsm: receiver

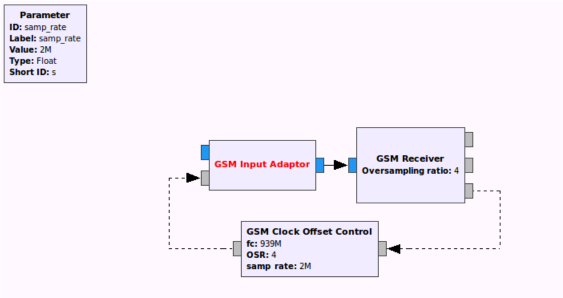


gr-gsm components: reference clock drift correction loop

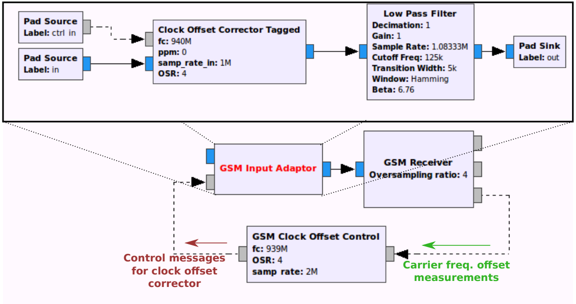
Reference clock drift correction loop

- cheap receivers usually equipped with clock sources not accurate enough for GSM reception
- for RTL-SDR's clock inaccuracy up to +/- 80ppm and changes with time/temperature
- 1ppm accuracy required for GSM reception
- the frequency correction loop corrects two effects of reference clock offset:
  - carrier frequency offset
  - sampling clock frequency offset

gr-gsm components: reference clock drift correction loop



gr-gsm components: reference clock drift correction loop

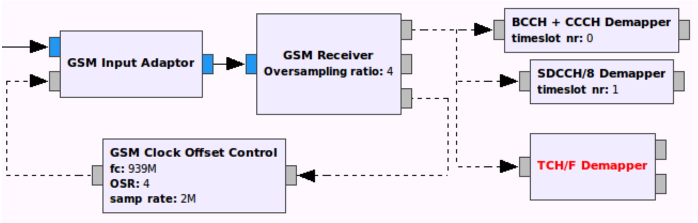


gr-gsm components: demappers

Demappers

- demultiplexing of logical channels
    - look into burst header
    - filter bursts coming from given timeslot
    - set correct channel types
- grouping of bursts for decoders

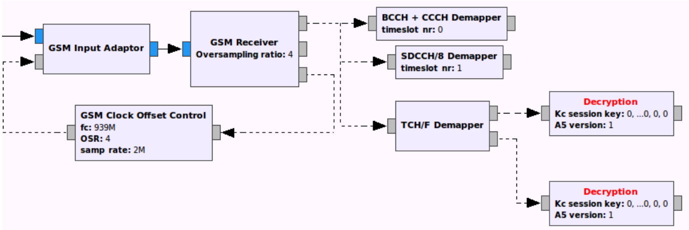
gr-gsm components: demappers



gr-gsm components: decryption block

- decrypts encrypted bursts
- supports A5/1, A5/2 and A5/3 algorithms
- implementation from libosmocore

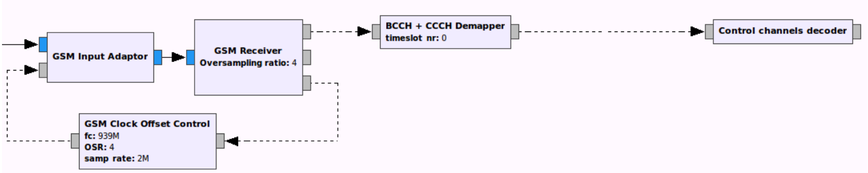
gr-gsm components: decryption block



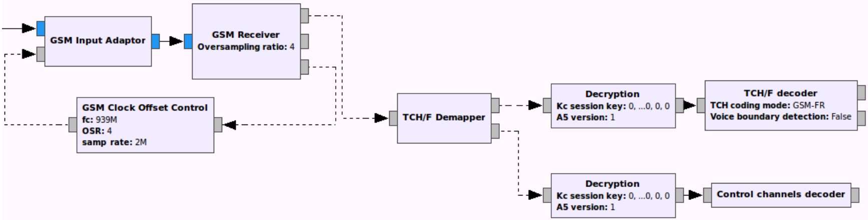
gr-gsm components: decoders

- decoding of logical channels
- currently supported:
    - control channels (BCCH, CCCH, SDCCH, SACCH, FACCH)
    - traffic channels (TCH/F)

gr-gsm components: decoders



gr-gsm components: decoders

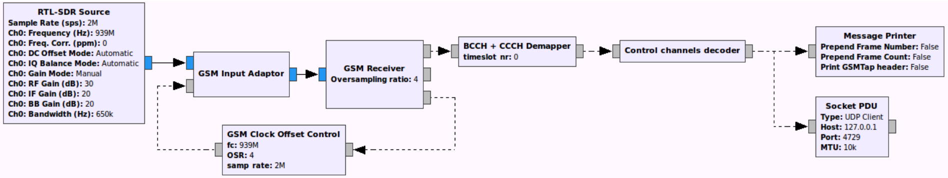


gr-gsm components: Sources and sinks

GNU Radio sources and sinks used with gr-gsm:

- any signal sources providing GSM signal:
  - file source
  - osmocom source
  - UHD source (Ettus USRP)
- Socket PDU for message analysis in Wireshark
- file sink for voice data
- gr-gsm's blocks for printing messages/bursts to stdout

Putting everything together
application for decoding BCCH channel



gr-gsm - plans for the future

- packaging the project (almost there)
- writing documentation (doxygen and tutorials) - much needed
- adding half rate TCH channels support:
  - missing demapping and decoding
  - fairly easy as soon decoding will be in libosmocore
- changing decoders to soft-input

gr-gsm - plans for the future

- improving the applications
  - improving quality of the scanning app
  - improving speed of scanning (through improving speed of the receiver)
  - adding support for decoding uplink and hopping channels
- making the receiver more modular (some initial works done, left due to lack of time)
- benchmarking BER rate of the demodulation
- improving frequency correction burst detection algorithm
- improving carrier frequency estimation
- adding ability to transmit bursts - long term plan

Multi-RTL

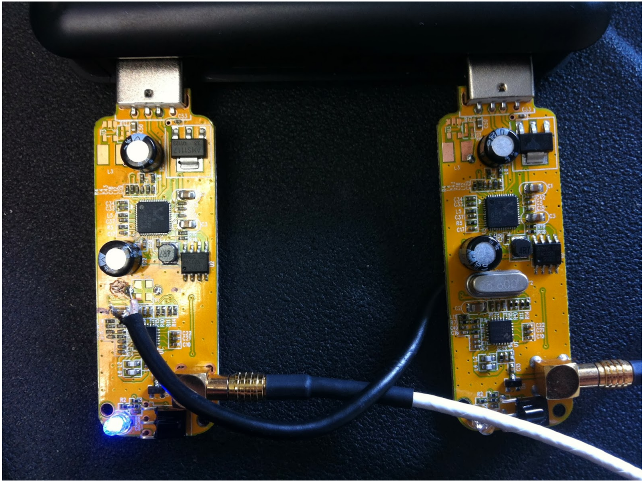
Motivation to work on a multi channel receiver based on RTL-SDR

- need to receive multiple bands synchronously (i.e. for GSM uplink reception, frequency hopping)
- only possible with relatively expensive wide-band/multi-channel equipment
- how to let everyone do this without spending hundreds $/€?

Looking for a solution...

- Juha Vierinen presented a method to synchronize two RTL-SDR dongles in frequency

Looking for a solution...



Credit: *Juha Vierinen, 2013*

Looking for a solution...

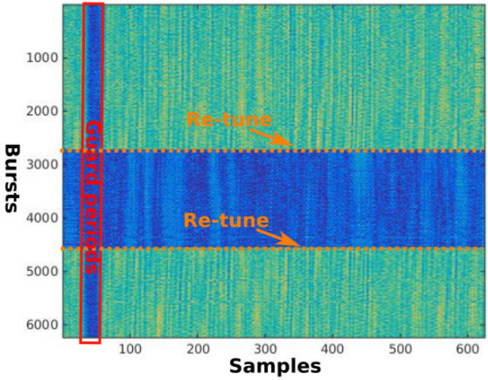
- Juha Vierinen presented a method to synchronize two RTL-SDR dongles in frequency
- no synchronization in time
- main ideas how to add time-sync - additional electronic circuit generating and injecting noise to all receivers
- possible applications of such solution:
  - direction finding
  - beamforming
  - passive radar
- not enough for receiving in multiple bands

What if RTL-SDR keep synchronization after changing central frequency

Idea how to check it:

- GSM signal has very regular amplitude due to guard period in each timeslot
- start with receiving on a frequency with GSM signal
- switch to a frequency without any signal
- switch back to the frequency with GSM signal
- look if guard periods are still where expected

What if RTL-SDR keep synchronization after changing central frequency



Synchronization is kept. Hurray!

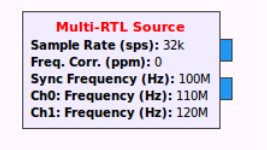
How to avoid building additional electronics

- there are many signals available in the air (GSM, DVB-T, FM)
- knowing the previous finding - we are not bound to a given frequency to perform time-synchronization
- possibility to use any signal with accuracy of time-difference estimation good enough for a particular application
- after gaining time-sync - switch receiving channels to any frequency

Multi-RTL - putting it all together

Multi-RTL

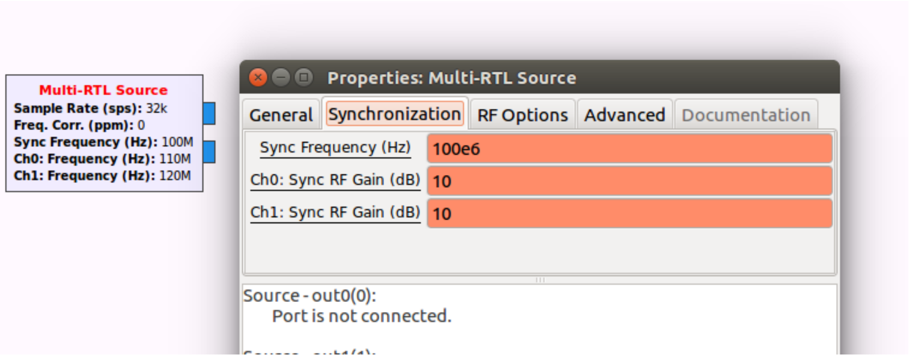
- a GNU Radio hierarchical block
- multiple Osmocom RTL-SDR source blocks under the hood
- similar set of options as Osmocom source
- project's page: https://github.com/ptrkrysik/multi-rtl



Multi-RTL - putting it all together

Multi-RTL cont.

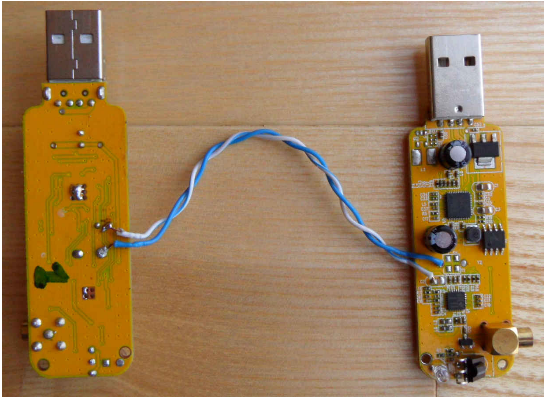
- additionl set of synchronization options



Multi-RTL - putting it all together

Multi-RTL cont.

- hardware - RTL-SDR's sharing common clock



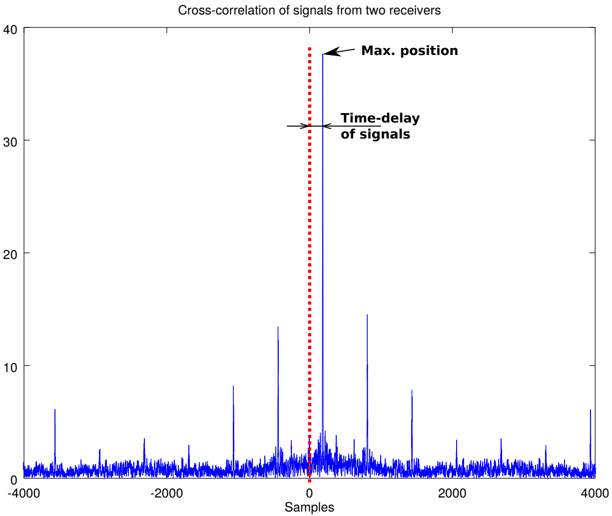
Multi-RTL - putting it all together

Multi-RTL cont.

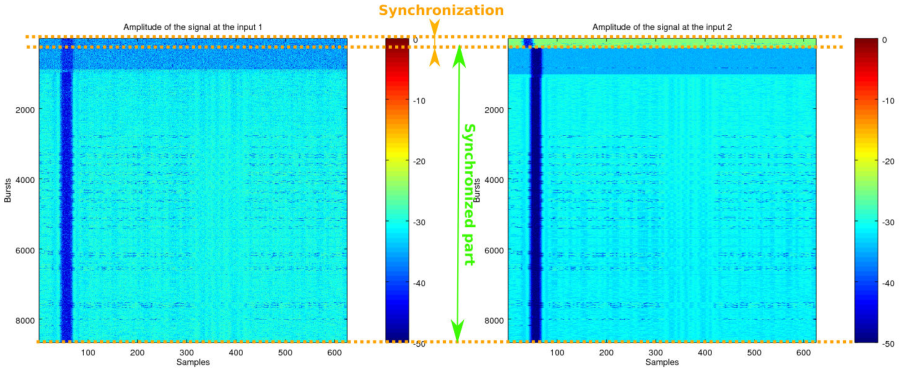
- for up to three channels receiver no additional electronics required - only Juha's hardware mod needed
- automatic synchronization procedure at start of the block:
  1. tuning the RTL-SDR dongles to the same frequency with sync. signal
  2. recording a short signals with all of the dongles
  3. computing cross-correlation of the signals
  4. co-channels delay estimation - finding positions of cross-correlation maximums
  5. correcting the delays
  6. switching the receivers to target frequencies
  7. changing other parameters of the channels (like gains) to target values

Multi-RTL - putting it all together

Channels' delay estimation



Amplitude of GSM signal recorded by two channel Multi-RTL



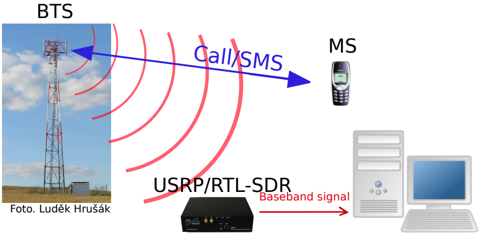
Multi-RTL - plans for the future

- making it fully coherent for everyone - porting of changes from keenerd's branch of RTL-SDR driver required
- making it resync automatically on synchronization loss - changes to the osmocom source required

Demos

Getting test data

- test data recorded with SDR hardware (USRP or RTL-SDR)
- a phone receives/makes a call or sends an SMS
- the phone - old Nokia 3310 with FBUS cable makes test captures much easier



Getting test data

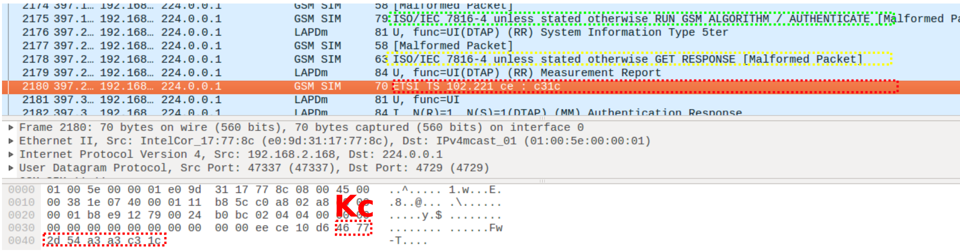
Obtaining Kc encryption key

Many possible ways:

- generating it with use of SimCard from RAND sent in Authentication Request
- cracking it with Kraken
- getting it from Nokia 3310's log obtained of dct3-gsmtap (osmocom):
  - find in Wireshark GSM Algorithm SimCard message: gsm_sim.apdu.ins == 0x88
  - find response to subsequent GET RESPONSE request
  - last 8 bytes of the response - Kc key

Obtaining Kc encryption key

Nokia 3310's log



Demos

- getting voice from GSM downlink
- getting SMS from GSM uplink (recorded with Multi-RTL)
- getting voice from uplink and downlink with frequency hopping

Obtaining hopping parameters

- System Information Type 1 message
  - contains Cell Allocation - ARFCN's used by the cell
- Immediate Assigment/Assigment Command
  - contains hopping parameters (if hopping used)
  - contains Mobile Allocation - set of ARFCN's for hopping

Questions and Answers