Abusing Calypso phones

Sylvain Munaut

PHDays, May 30/31, 2012

About the speaker

- Linux and free software "geek" since 1999
- M.Sc. in C.S. + some E.E.
- General orientation towards low level
  - Embedded, Kernel, Drivers and such.
  - Hardware (Digital stuff, FPGA, RF, ...)
- Interest in GSM projects for about 3 years
  - OpenBTS, OpenBSC, Airprobe, Osmocom-BB, ...
  - 27C3 GSM Intercept demo
  - Mostly in my spare time

Outline

Motivation

Modify a phone to make it do what we want rather than what it was designed to.

Why ?

- Gain access to lower layers of the communication stack
  - Other projects paved the way for GSM (OpenBTS, OpenBSC, Osmocom-BB, ...)
  - However they don't all allow to go down to L1 and some depend on expensive hardware
- Create the tool allowing security research
- Just for fun: Usefulness is overrated anyway

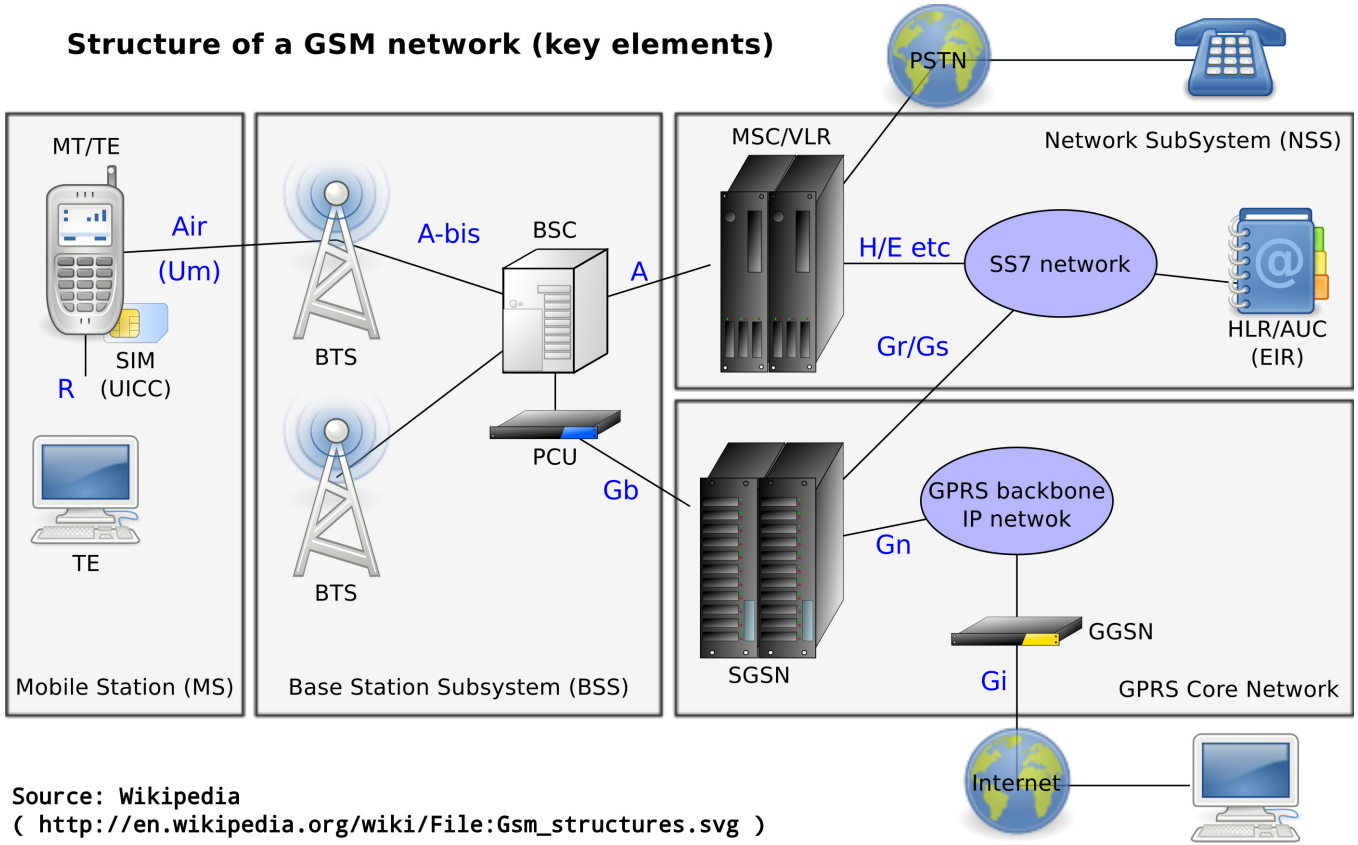
Today's target

Target hardware: Motorola C123

- Supported by Osmocom-BB
- Classic TI Calypso design
  - Lots of alternative platforms if needed
  - Some leaked sources and documentation available
- Cheap (20 EUR new, down to 1 EUR on ebay)
- Readily available

GSM background

GSM
Network overview



Source: Wikipedia
(http://en.wikipedia.org/wiki/File:Gsm_structures.svg)

We'll be focusing on the GSM Air Interface: `Um`.

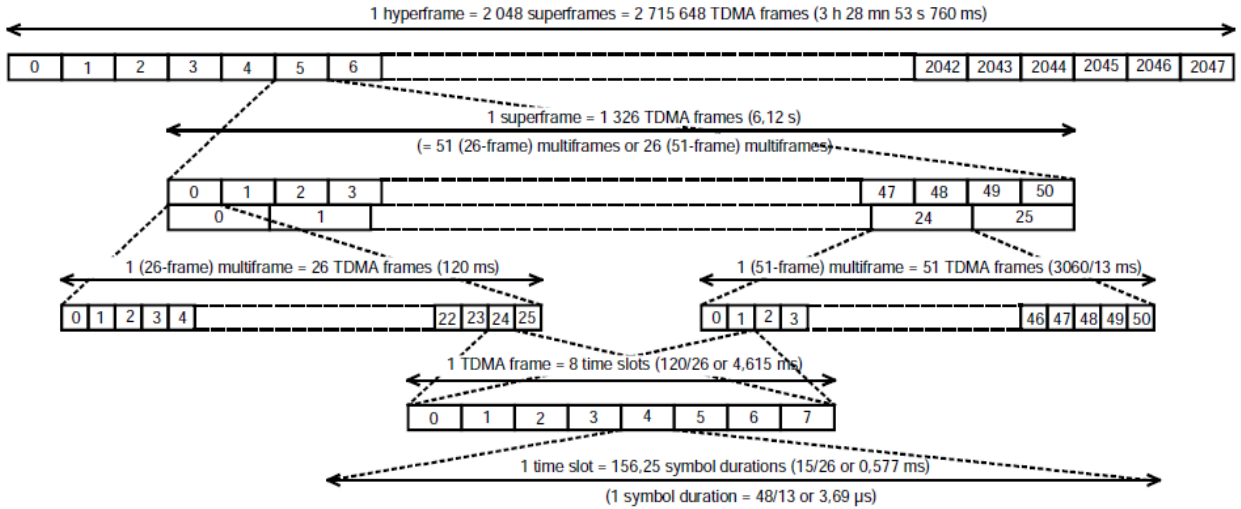
GSM Um: Layer 1

Frequencies

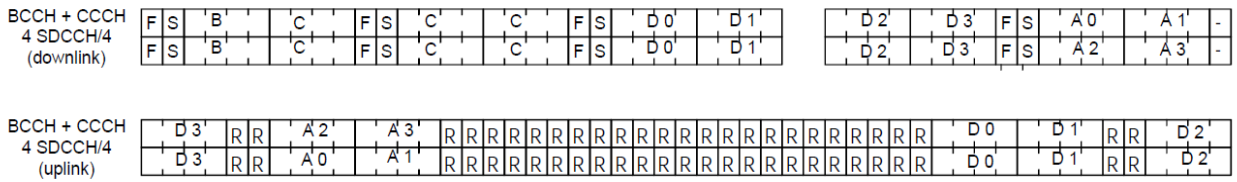
- Several bands
  - `GSM-850`, `EGSM-900`, `DCS1800`, `PCS1900`, ...
  - `http://en.wikipedia.org/wiki/GSM_frequency_bands`
- Each band has two frequency range (FDD)
  - Downlink, from Network to MS (e.g. `DCS1800`: 1710.2 to 1784.8 MHz)
  - Uplink, from MS to Network (e.g. `DCS1800`: 1805.2 to 1879.8 MHz)
- `ARFCN` = Absolute Radio-Frequency Channel Number
  - maps to a given frequency pair (UL/DL)
  - 200 kHz spacing

GSM Um: Layer 1
TDMA

- Fully synchronous
- Described as a TDMA nightmare



- Each frame in multi-frame has a specific purpose



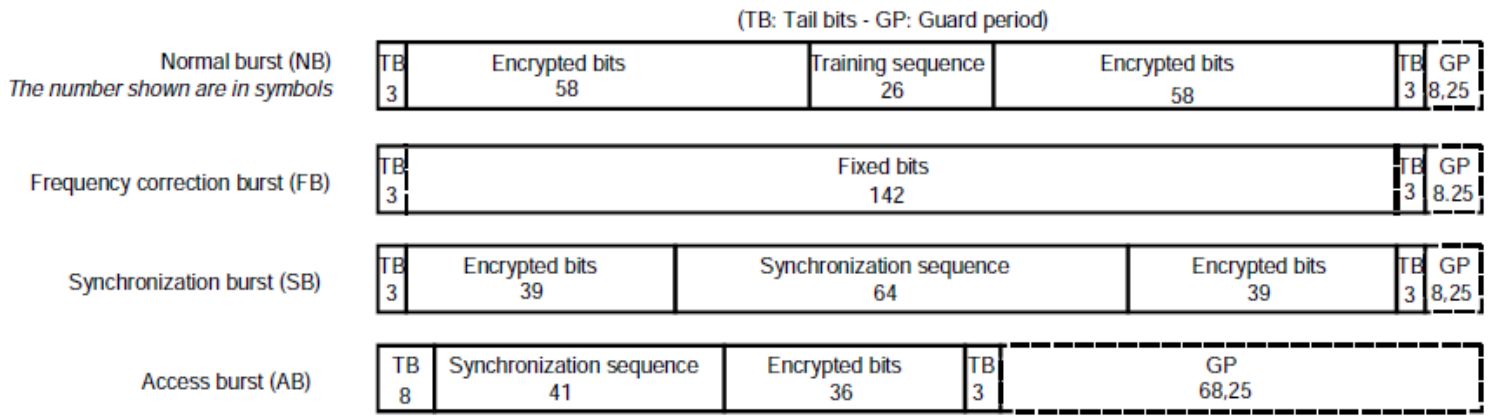
- 1 frame = 8 timeslots (bursts)
- Physical channel = 1 timeslot on 1 ARFCN

GSM Um: Layer 1

Bursts

4 types of bursts :

- Normal burst: Used to carry "real" data traffic.
- Frequency correction burst: (FCCH) Allow MS to sync its clock and coarse TDMA
- Synchronization burst: (SCH) Allow MS to preicsely sync to TDMA
- Access burst: (RACH) Used by the MS to request a dedicated channel



Passive Listening

A bit of history

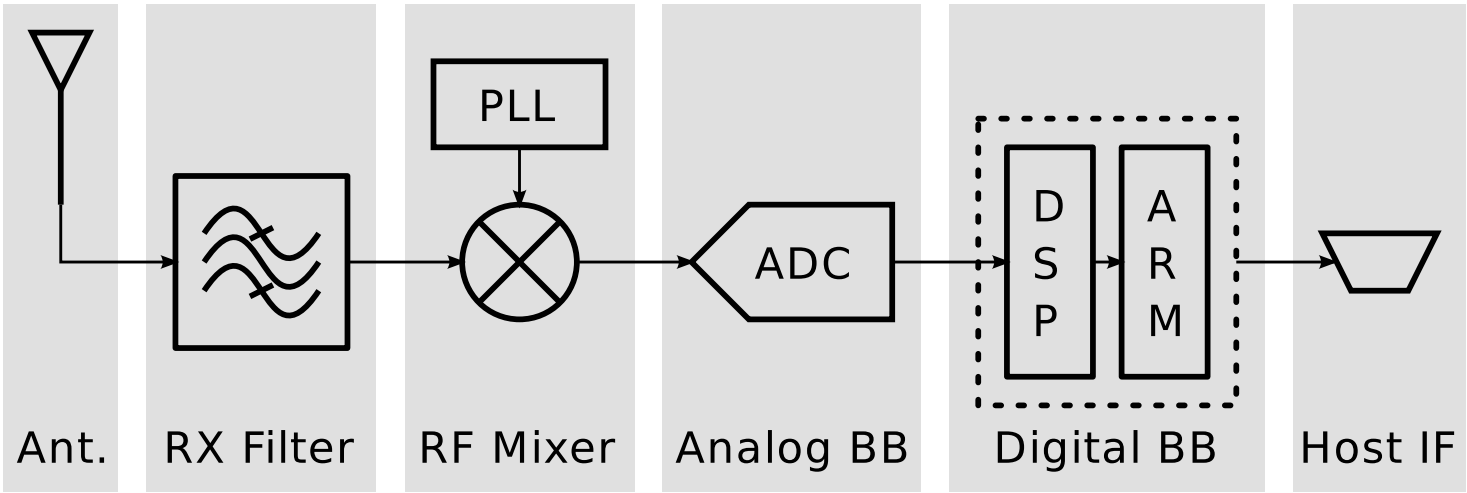
Osmocom-BB is an Free Software GSM Baseband implementation.

- Early timeline (2010):
  - Early February: Osmocom-BB is initiated
  - Late February: Osmocom-BB is announced publicly
    - BCCH reception mostly
  - March-July: Progressive work to get TX, SDCCH, LUR, ...
  - August: First phone call
- Already a **big** advance
  - Full L2 & L3 control on the MS side
- But I wanted more ;)

Goal

- Turn a phone into a passive listener
  - **Raw bursts** data
  - Uplink and Downlink
  - Frequency Hopping
- Timeline
  - Work started almost directly after Osmocom-BB was initiated
  - First prototype in Q3 2010
    - Shown at Deepsec 2010 & 27C3

Typical RX path



- Antenna: not an issue, can be replaced if needed
- RX filter: not an issue for lab tests, can be removed if needed
- RF mixer: tests shows it works just fine tuning at UL/DL
- Analog baseband: not an issue
- DSP core: ROM based and limited. Need a solution.
- ARM core: firmware under our control thanks to osmocom-bb
- Host interface: serial can be made fast enough

DSP
The problem

- ROM based firmware
  - But supports executing code from RAM
  - Official firmwares load 'patches' somehow (fix bugs, ...)
- The ARM schedules "tasks" to be executed by the DSP
- No existing tasks does what we want
  - DSP converts from L2 packets to L1 bursts internally
- Need to patch it
  - Dump ROM
  - Analyze it and figure how patching works
  - Write custom "tasks" to do what we want

DSP
Dumping (1)

- Architecture
  - Distinct program, data & IO address space
    - Different instructions to access them
    - Some zones mapped in both program and data space
  - Communicates with the ARM by shared memory zone
    - Called `API` RAM
    - Mapped in both program and data address space
- ROM Bootloader
  - Leaked TSM30 sources hinted at ROM bootloader
  - TI documention for similar DSP provided the details
  - Allows to download custom code/data and jump to it
- Reading ROM
  - Upload custom stub to copy chunk of ROM to `API` RAM
  - But it didn't work ... only read `0xffff`
  - Security feature: code executing from RAM can't read ROM

DSP
Dumping (2)

If we can't read the ROM from code executing from RAM, we'll have to read it from code executing from ROM ...

- There has to be a `memcpy` equivalent somewhere
  - Look at known DSP code for this architecture
  - Often inlined, so only part will be usable
- Looking for:
  - `mvdd *AR?, *AR?` for data space
  - `reada *AR?` for program space
- Bruteforce it
  1. Use bootloader to launch stub
  2. Setup registers with a 'guess'
  3. Jump to a location
  4. Halt the DSP from the ARM a bit later
  5. Check for result in `API` RAM
  6. Retry ...

DSP
Dumping (3)

Program space

```
.prom0:00007213 7E92                        reada   *AR2+
.prom0:00007214
.prom0:00007214                 loc_7214:
.prom0:00007214 F000 0001                   add     #1, A
.prom0:00007216 FC00                        ret
```

Data space

```
.pdrom:0000E4B8 E598                        mvdd    *AR3+, *AR2+
.pdrom:0000E4B9 FC00                        ret
```

The `ret` instructions are added bonuses

DSP
Analyzing (1)

- CPU supported by IDA Pro Advanced
  - Added support for IO port definitions and memory mappings
  - Now in mainline
- Entry point is known
- Mix of C and hand-crafted assembly
  - No clear conventions
- Lots of indirect calls
  - Using function pointers in RAM copied from ROM at startup
    - We can replace those by our own !
    - This is how to add custom tasks, extend the DSP, ...
  - Screws a bit with IDA autoanalysis
  - Several different tables and call mechanisms

DSP
Analyzing (2)

Use interrupts and IO access to trace important functions

- Frame interrupt: Tasks
- DMA interrupt: IQ samples buffer and demodulation
- A5 unit IO: Cipher setup
- DMA unit IO: Burst RX setup
- RIF unit IO: Burst TX buffer

And finally write custom task to do what we want ...

Work In Progress

Phone as a BTS

Goal

- Attempt to convert a phone into a working BTS
  - Not full featured, not compliant with specs, ...
  - Provide minimal service
- Motivation
  - Another cheap tool for GSM research
  - Fuzz cell phones
  - Portable fake BTS
  - Just prove it's doable
- First post on the mailing list about this about 2 years ago
  - Only the base idea, not real work done
  - First very rough work at CCCamp 11
  - Idea popped up again at OsmoDevCon 2012

Phone as a BTS

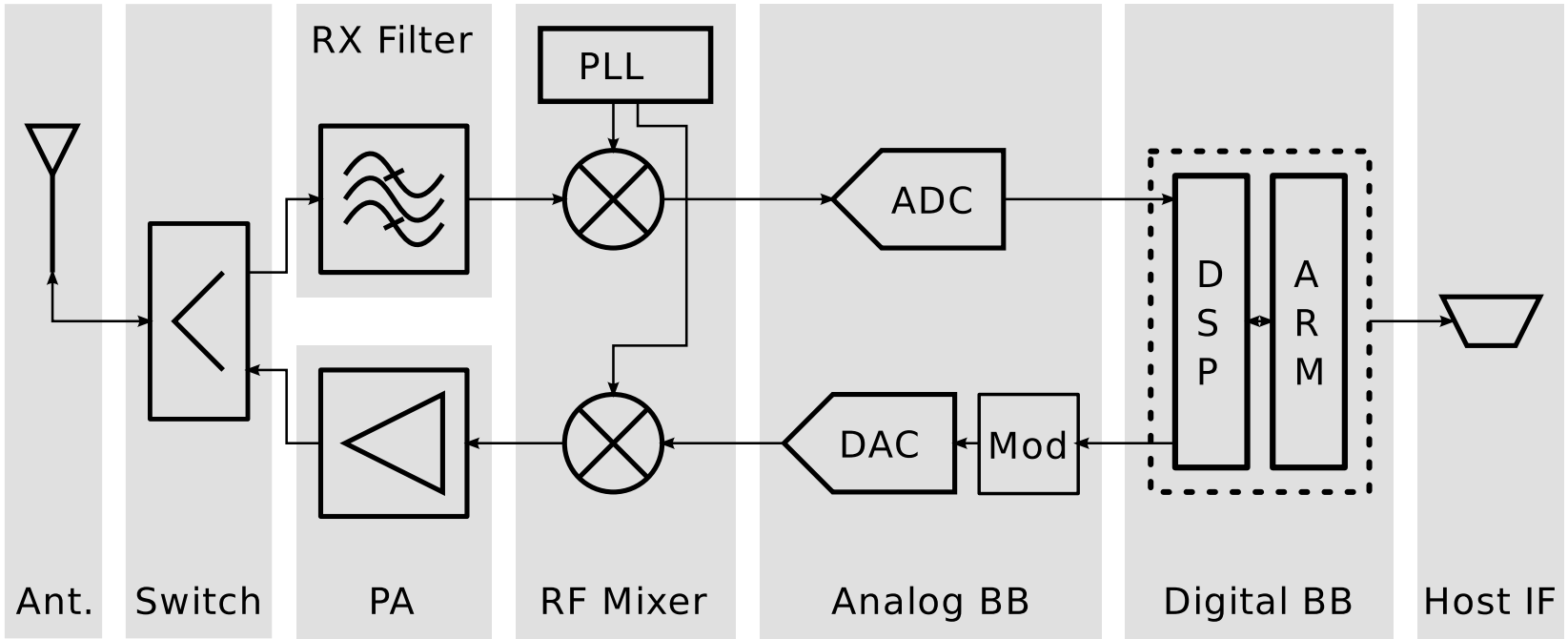
Differences between MS & BTS

What does a BTS do that a phone doesn't ?

- Layer 1:
  - Uplink / Downlink frequencies
  - Simultaneous RX & TX
    - Continuous C0 beacon to allow phone to 'find' the cell
    - MS usually TX 3 timeslots after RX
  - Transmit FCCH / SCH
  - Receive RACH
  - Clock master
- Layer 2 & 3: Role swapped

Phone as a BTS

Typical TX & RX path



Phone as a BTS

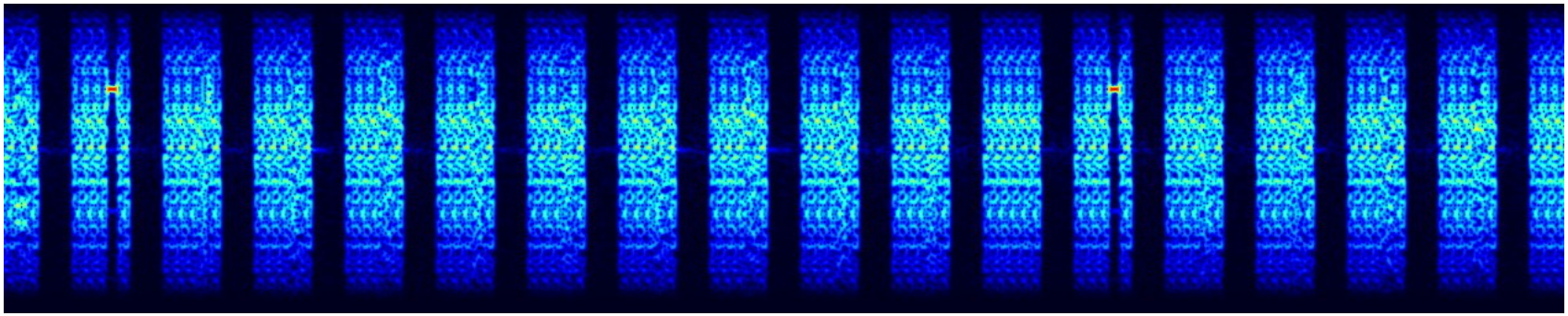
Proof of concept

- DSP patch
  - FCCH, SCH, NB & Dummy TX
  - Multi slot TX
  - RACH detection (detect with power and send IQ samples to host)
- Use OpenBTS
  - Already split between main OpenBTS and actual radio interface
  - Replace the `transceiver`
- Attempt half duplex operation
  - Timeslot layout: `Tt_R_ttt`
- Use commercial cell as timing reference

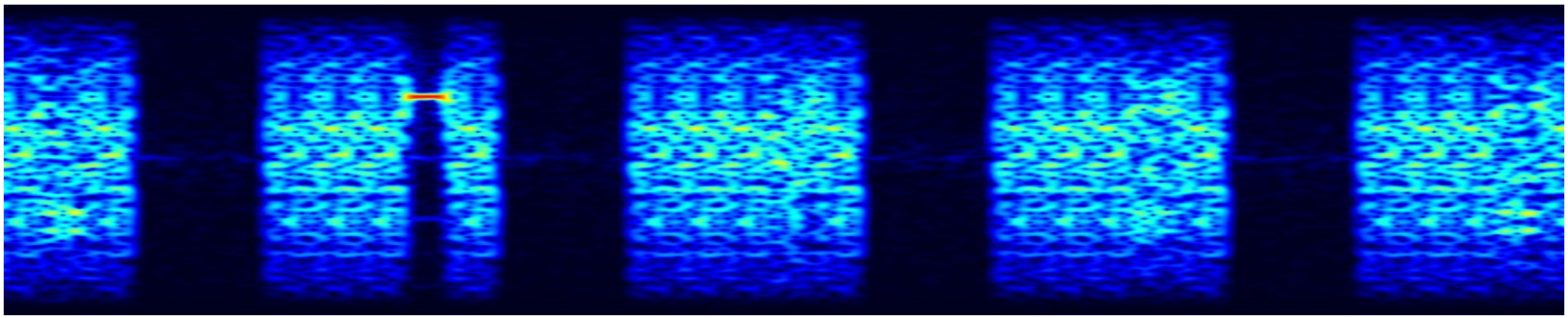
Phone as a BTS

Spectrum view

Multiframe



Zoom



Phone as a BTS

Demonstration

Hopefully, it'll work ...

Keep in mind :

- Just a proof of concept
- Long time to go to clean up and make it usable and reliable

Thanks

Thanks to anyone contributing to the various Open Source GSM / GSM security projects. Most notably here :

- Harald "LaF0rge" Welte
- Dieter Spaar
- David Burgess and his team at KestrelSP
- Andreas "jolly" Eversberg
- Steve "steve-m" Markgraf

And of course, thanks to the PHDays team for having me here.

Further reading

Airprobe	http://airprobe.org/
OsmocomBB	http://bb.osmocom.org/
OpenBSC	http://openbsc.osmocom.org/
OpenBTS	http://openbts.sourceforge.net/
GSM Specs	http://webapp.etsi.org/key/queryform.asp