



Betreuer:

Nils Beyrle*

Prof. Dr. Johannes Maucher
Moritz Seltmann

Ein Projekt im Studiengang Medieninformatik an der
Hochschule der Medien Stuttgart

8. August 2010

Zusammenfassung

Das Ziel des Projektes ist es die Steuerungskanäle von Global System for Mobile Communications (**GSM**) mitzuschneiden. Bei diesen Kanälen handelt es sich um unverschlüsselte Broadcast-Kanäle, die von allen Teilnehmern in einer Zelle zur Koordinierung genutzt werden. Darüber gehen u.A. periodisch ausgestrahlte Informationen der Basisstation, Daten beim Einbuchten eines Endgerätes in ein Netz, oder beim Wechsel (Handover) einer Funkzelle. Zu diesen Kanälen zählt beispielhaft der Broadcast Control Channel (**BCCH**). Das Ergebnis des Dumps soll sich an dem für WLAN bekannten *airodump* orientieren. Dazu zählt z.B. nach einem Durchlauf eine Übersicht zu gefunden Netzen auszugeben.

Die verschiedenen mitgeschnittenen Pakete müssen dafür zuerst einmal auch als solche auf den PC kommen. Diese werden anschließend dort so aufbereitet, dass sie etwa im weit verbreiteten *Wireshark* betrachtet und näher analysiert werden können.

Das für den Dump eingesetzte Gerät muss dazu die in der Luftschnittstelle vorhandenen Informationen zunächst am Rechner verfügbar machen. Dazu werden im ersten Schritt verschiedene Geräte evaluiert:

- Telit EVK2
- Nokia 3310
- Universal Software Radio Peripheral (**USRP**)

Aus diesen Geräten wird das am besten geeignete Gerät ausgewählt und mit diesem dann der Mitschnitt und die nötige Aufbereitung der Daten realisiert.

*E-Mail: nb031@hdm-stuttgart.de

Inhaltsverzeichnis

Inhaltsverzeichnis	2
Abbildungsverzeichnis	4
1 Einleitung	5
2 Geräteauswahl	5
2.1 Voraussetzungen	5
2.2 Die Geräte	5
2.2.1 Telit EVK2	5
2.2.2 Nokia 3310	5
2.2.3 USRP (Universal Software Radio Peripheral)	6
3 Verwendung des USRP	7
3.1 Überblick über die verwendete Software	7
3.2 Versionschaos zwischen den Programmen	7
3.3 Installation der Software	8
3.3.1 GNU Radio	9
3.3.2 airprobe	10
3.3.3 Wireshark	10
3.3.4 airprobe - gssm	10
3.4 Beschreibung von airprobe	11
3.4.1 gsmdecode	11
3.4.2 gsm-tvoid	11
3.4.3 gsm-receiver	12
3.4.4 gssm	12
4 Live-CD	12
4.1 Einleitung	12
4.2 Programme	13
4.2.1 GNU Radio	13
4.2.2 airprobe	13
4.2.3 Wireshark	13
4.2.4 GSMdump	14
4.3 Beschreibung der GSMdump Skripte	14
4.3.1 gsmdump.sh	14
4.3.2 pcap4gsmdecode.py	16
4.3.3 analyse.py	17
4.3.4 summary.py	17
4.3.5 arfcn.py	18
4.3.6 gsmalive.sh	18
4.3.7 capture.sh	19
4.3.8 analysecf.sh	20
5 Weitere Software-Projekte	20
5.1 OpenBSC	20
5.2 OpenBTS	20
5.3 OsmocomBB	21

6	Fazit	22
A	Abkürzungsverzeichnis	23
B	Sequenzdiagramm	24
B.1	Paging Request	24
C	Analysierte Mitschnitte	25
C.1	BCCH	25
C.1.1	System Information Type 1	25
C.1.2	System Information Type 2	26
C.1.3	System Information Type 2ter	27
C.1.4	System Information Type 2quater	28
C.1.5	System Information Type 3	29
C.1.6	System Information Type 4	31
C.1.7	System Information Type 13	32
C.2	CCCH	33
C.2.1	Immediate Assignment	33
C.3	PCH	34
C.3.1	Paging Request Type 1 (IMSI)	34
C.3.2	Paging Request Type 1 (TMSI)	35
D	Quellcode der Live-CD	36
D.1	gsmdump.sh	36
D.2	pcap4gsmdecode.py	41
D.3	analyse.py	42
D.4	summary.py	44
D.5	arfcn.py	48
D.6	gsmalive.sh	49
D.7	capture.sh	51
D.8	analysecfi.sh	53
	Literaturverzeichnis	56

Abbildungsverzeichnis

1	Zusammenwirken der einzelnen Programme	7
2	Versionschaos zwischen den Programmen	8
3	Grafische Darstellung der Empfangsstärke bei <i>gsm-tvoid</i>	12
4	pcap Datei in <i>Wireshark</i>	14
5	Beispielausgabe eines Scans von <i>gsmdump.sh</i>	16
6	Sequenzdiagramm eines Paging Request	24

1 Einleitung

Ein Beispielmitschnitt im Wireshark-Format findet sich unter http://www.gsmdump.de/downloads/arfcn_877_1878.2M.pcap

Die aktuelle Version dieses Dokuments findet sich unter <http://www.gsmdump.de/downloads/gsmdump.pdf>

Die aktuelle Version dieses Dokuments in einer Druckversion findet sich unter http://www.gsmdump.de/downloads/gsmdump_print.pdf

Die Live-CD zum Projekt findet sich unter <http://www.gsmdump.de/downloads/gsmdump.iso>

In Beispielpaketen wurden **IMSI**s oder ähnliche Geräten/Personen zuordenbare Daten im Idealfall anonymisiert.

Im Folgenden werden zumindest grundlegende Kenntnisse über **GSM** (den Aufbau des Netzes, der Luftschnittstelle und der Signalisierung) vorausgesetzt. Zum Nachschlagen sei hier auf [9], [7] und [8] verwiesen.

2 Geräteauswahl

2.1 Voraussetzungen

Für die gestellte Aufgabe ist es wichtig, dass das Gerät nicht auf einen bestimmten Anbieter festgelegt ist. Es sollen alle von **GSM** verwendeten Frequenzen, ohne **SIM**-Karten von allen Providern zu benötigen, untersucht werden können. Das verwendete Gerät sollte zudem möglichst viele der verfügbaren Daten an seiner Schnittstelle zur Analyse bereitstellen.

2.2 Die Geräte

2.2.1 Telit EVK2

Das *Telit EVK2* besitzt neben einem **GSM**-Modul auch noch einen GPS-Empfänger. Es gibt seine Daten über vier virtuelle serielle Schnittstellen, welche über eine USB-Schnittstelle an den Rechner angebunden werden, preis. Sein primärer Einsatzzweck ist der mobile Einsatz, um etwa Karten für die Netzabdeckung zu erstellen oder Daten für eine Positionsbestimmung zu gewinnen (wurde bereits im WS 2008/2009 von Christoph Graf im Rahmen des Projektes "GSM und GPS Ortung im Vergleich" an der HdM realisiert).

Gegen eine Verwendung für GSMDump spricht, dass das Gerät eine **SIM**-Karte benötigt und damit die gewonnen Daten immer an das verwendete Netz geknüpft sind. Es ist also nicht einfach ein Wechsel auf die Frequenzen eines beliebigen Mobilfunkanbieters möglich.

2.2.2 Nokia 3310

Das *Nokia 3310* ist zwar ein schon in die Jahre gekommenes Handy, es hat gegenüber einem iPhone jedoch einen entscheidenden Vorteil: es besitzt einen

Debug-Modus über den es an einer seriellen Schnittstelle sehr viele Informationen über Paktet an sich und sein Netz bereitstellt. So erhält man darüber auch einen Mitschnitt der Übertragenen Daten, jedoch nur aus dem, durch die verwendete **SIM**-Karte festgelegt, eigenen Netz und an das im speziellen Fall verwendete Handy adressierte Daten.

2.2.3 USRP (Universal Software Radio Peripheral)

Homepage: <http://www.ettus.com/order>

Ein **USRP** bietet eine sehr flexible Hardware Plattform, die es ermöglicht per Software auf verschiedenen Funk-Frequenzen empfangen und senden zu können. So können für verschiedene Frequenzen verschiedene Empfangs-/Sendemodule in die schwarze Box eingebaut werden und somit theoretisch beliebige Frequenzen mitgehört oder Daten darauf versendet werden.

Diese Offenheit befreit auch von der bei den vorherigen Lösungen nötigen **SIM**-Karte und von der Beschränkung auf den eigenen Mobilfunkanbieter. Mit dem eingebauten *DBSRX* Modul können Frequenzen von 800 MHz bis 2400 MHz empfangen werden. Das deckt das ganze in Deutschland für **GSM** (bei 900 und 1800 MHz) verwendete Frequenzspektrum ab.

Bisher ist es mit *airprobe* nur möglich den Downlink-Kanal¹ (also alles was von der Basisstation zum Mobilteil gesendet wird) zu empfangen. Der Grund hierfür ist, dass die Software sich im ersten Schritt auf die Downlink-Frequenz synchronisieren muss, da dort die dafür benötigten Pakete vorhanden sind. Hierfür gibt es nun zwei Lösungsansätze (siehe [4,5]):

- Im Anschluss an eine erfolgreiche Synchronisierung wäre ein Wechsel auf die Uplink-Frequenz nötig, allerdings ohne die Synchronisation zu verlieren. Dies wäre praktisch fast unmöglich, da ein sofortiges umschalten nicht möglich ist und auch die Dauer eines Umschaltvorgangs nicht im Vorhinein exakt bestimmt werden kann
- Mit einem zweiten Empfangsmodul im **USRP** wäre es möglich gleichzeitig auf einer Frequenz den Downlink-Kanal und auf einer zweiten den Uplink-Kanal mitschneiden zu können und somit die Synchronisierung aus dem Downlink-Kanal auf den Uplink-Kanal zu übertragen. Diese Funktion ist bisher allerdings noch nicht in *airprobe* implementiert.

Für die Realisierung der Aufgabenstellung wird im folgenden das **USRP** verwendet.

Ausblick: Mit dem verwendeten Modul wären sogar noch Versuche im 2400 MHz Bereich von WLAN denkbar.

Mit einem weiteren Modul, um auf den **GSM**-Frequenzen zusätzlich auch senden zu können, wäre mit *OpenBTS*² damit der Betrieb einer eigenen Basisstation möglich.

¹Siehe auch Abbildung 6 auf Seite 24 eines Paging Request

²Für Details zu *OpenBTS* siehe Kapitel 5.2 auf Seite 20

3 Verwendung des USRP

3.1 Überblick über die verwendete Software

Für den Betrieb des **USRP** sind eine Reihe von verschiedenen Softwarepaketen nötig:

- *GNU Radio*: Eine Software-Sammlung, die sich um den Funklayer kümmert, also alles was irgendwie nötig ist um am Ende rein binäre Daten vorliegen zu haben. Sie spricht das **USRP** an und bereitet die Daten soweit auf, dass sie von anderen Anwendungen aus *airprobe* weiterverarbeitet werden können. *GNU Radio* implementiert große Teile der Funk-Logik in Software und fällt damit in die Kategorie *Software Defined Radio*³
- *airprobe*: Eine Sammlung verschiedener Tools um **GSM** analysieren und aufbereiten zu können. *airprobe* verarbeitet die Daten von *GNU Radio* weiter.
- *Wireshark*: Das weitverbreitete Werkzeug um Paketmitschnitte komfortabel analysieren zu können. Ist in einer angepassten Version nötig um die **GSM** Daten zu verstehen.

Abbildung 1 stellt das Zusammenwirken der verschiedenen Tools dar.

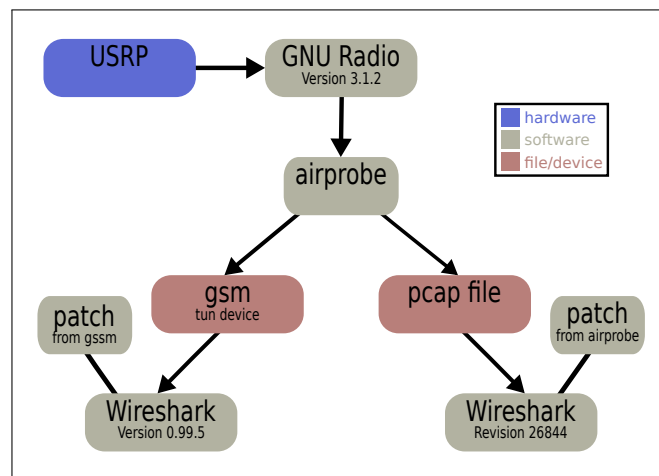


Abbildung 1: Zusammenwirken der einzelnen Programme

3.2 Versionschaos zwischen den Programmen

Die Inbetriebnahme des **USRP** stellt sich als nicht trivial heraus. Ein großes Problem bis es überhaupt irgend ein Paket zu sehen gibt ist das Versionschaos der verschiedenen verwendeten Programme. So funktionieren benötigte Patches nur bei bestimmten Versionen und Programme spielen nur mit bestimmten Versionen anderer Programme zusammen. Alte Versionen machen dann aber

³zu *Software Defined Radio* siehe auch http://en.wikipedia.org/wiki/Software-defined_radio

teilweise wieder Probleme mit neueren *GCC* Versionen (für Details siehe auch Kapitel 3.3). Abbildung 2 demonstriert in vereinfachter Form die verschiedenen Abhängigkeiten.

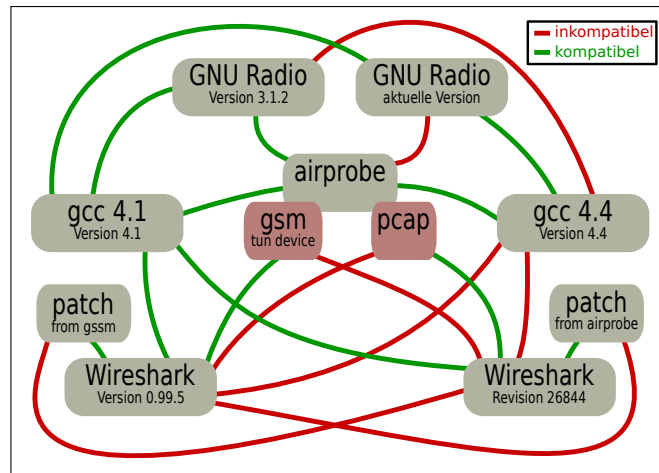


Abbildung 2: Versionschaos zwischen den Programmen

3.3 Installation der Software

Als Entwicklungs- und Test-System kommt ein Ubuntu 9.10 unter 32 Bit zum Einsatz. Es sind einige Programme zum kompilieren von eigener Software nötig. Sollte einer der folgenden Schritte daher fehlschlagen fehlt vermutlich ein benötigtes Programm oder eine benötigte Bibliothek, welche dann bei Bedarf über den Paketmanager nachzuinstallieren ist.

GNU Radio setzt in deren Wiki ⁴ die folgenden Programme voraus. Es empfiehlt sich diese für die weiteren Schritte zu installieren.

```
sudo aptitude install swig g++ automake libtool python-dev libfftw3-dev libcppunit-dev
libboost1.38-dev libusb-dev fort77 sdcc sdcc-libraries libstd1.2-dev python-wxgtk2.8
subversion git-core guile-1.8-dev libqt4-dev python-numpy ccache python-opengl libgs10-
dev python-cheetah python-lxml doxygen qt4-dev-tools libqt5-qt4-dev libqwtplot3d-qt4-
dev pyqt4-dev-tools
```

GCC wird zudem in Version 4.1 benötigt:

```
sudo aptitude install gcc-4.1
```

Sollten sich ein paar Python Skripte über fehlende libs beschweren, so liegt das daran, dass Ubuntu nicht mehr in `/usr/local/lib/python2.6/site-packages` danach sucht, z.B. *GNU Radio* hierhin jedoch installiert wurde. Zwei Symlinks dienen hier als einfache Abhilfe:

```
cd /usr/local/lib/python2.6/dist-packages
sudo ln -s ../site-packages/gnuradio gnuradio
sudo ln -s ../site-packages/usrpm usrpm
```

Anschließend sollten die entsprechenden libs vom System gefunden werden.

⁴<http://gnuradio.org/redmine/wiki/gnuradio/UbuntuInstall>

3.3.1 GNU Radio

Homepage: <http://gnuradio.org/redmine/wiki/gnuradio> [2]

Bei *GNU Radio* stellte es sich als sehr schwer heraus die zu *airprobe* passende Version zu finden. Die beiden Programme arbeiten nicht in jeder Version zusammen. Nach unzähligen Kompilier-Vorgängen zeigte sich die Version 3.1.2 als funktionsfähig. Zu all dem lässt sich die *GNU Radio* Version 3.1.2 nicht mit *GCC* in der Version 4.4 bauen, daher ist das *configure*-Skript anzuweisen den *GCC* in der Version 4.1 zu verwenden. Installieren lässt es sich dann mit:

```
wget ftp://ftp.gnu.org/gnu/gnuradio/gnuradio-3.1.2.tar.gz
tar xvfz gnuradio-3.1.2.tar.gz
cd gnuradio-3.1.2.tar.gz
./configure CC=gcc-4.1 CXX=g++-4.1
make
sudo make install
```

Wichtig ist, dass in der Auflistung nach dem *configure* die Unterstützung für das **USRP** aufgeführt wird (siehe Listing 1). Dies sollte zwar der Fall sein, ansonsten müssen jedoch die entsprechenden Meldungen durchgesehen werden, um herauszufinden an was es scheitert.

Listing 1: Ausgabe von `./configure CC=gcc-4.1 CXX=g++-4.1` bei *GNU Radio*

```
*****
The following GNU Radio components have been successfully configured:

config
omnithread
gnuradio-core
usrp
gr-usrp
gr-audio-alsa
gr-audio-oss
gr-atsc
gr-gpio
gr-gsm-fr-vocoder
gr-pager
gr-radar-mono
gr-radio-astronomy
gr-trellis
gr-video-sdl
gr-wxgui
gr-sounder
gr-utils
gnuradio-examples

You may now run the make command to build these components.

*****
The following components were skipped either because you asked not
to build them or they didn't pass configuration checks:

gr-audio-jack
gr-audio-osx
gr-audio-portaudio
gr-audio-windows
gr-comedi

These components will not be built.
```

3.3.2 airprobe

Homepage: <https://svn.berlin.ccc.de/projects/airprobe/> [1]

airprobe erhält man mit einem

```
git clone git://svn.berlin.ccc.de/airprobe
```

damit lässt sich gsmdecode

```
cd airprobe/gsmdecode
./bootstrap
./configure
make
```

und gsm-tvoid bauen

```
cd ../gsm-tvoid
./bootstrap
./configure
make
```

3.3.3 Wireshark

Homepage: <http://wiki.wireshark.org/Development> [3]

Wireshark wird in Revision 26844 benötigt, diese erhält man mit

```
svn co -r 26844 http://anonsvn.wireshark.org/wireshark/trunk/ wireshark
```

anschließend wendet man den Patch aus *airprobe* an (der Pfad muss evtl. entsprechend angepasst werden)

```
cd wireshark
patch -p0 < ~/airprobe/wireshark/wireshark-wtap-gsm.patch
```

um zum Abschluß Wireshark zu konfigurieren und zu kompilieren. Hier macht wieder der *GCC* in Version 4.4 Probleme, weshalb auf die Version 4.1 ausgewichen wird:

```
./autogen.sh
./configure CC=gcc-4.1 CXX=g++-4.1
make
```

3.3.4 airprobe - gssm

Homepage: <http://thre.at/gsm/>

gssm wird nicht immer zwangsläufig benötigt. Für einen Live-Mitschnitt ist jedoch *Wireshark* in der Version 0.99.5 und *mktun* aus *gssm* nötig (siehe auch Kapitel 3.4.4 auf Seite 12).

Leider kommt *gssm* nicht mit *GNU Radio* Version 3.1.2 klar, hierfür wird die Revision 5220 benötigt. Diese Revision bezieht sich auf SVN, inzwischen verwendet *GNU Radio* jedoch Git, dort lautet die entsprechende Revision 28259329. Zum kompilieren ist hierfür folgende Vorgehensweise notwendig:

```
git clone git://gnuradio.org/gnuradio
cd gnuradio
git checkout 28259329
./configure CC=gcc-4.1 CXX=g++-4.1
make
sudo make install
```

anschließend wird *Wireshark* in Version 0.99.5 benötigt. Hier führen folgende Befehle zum Erfolg:

```
wget http://www.wireshark.org/download/src/all-versions/wireshark-0.99.5.tar.bz2
tar xvfz wireshark-0.99.5.tar.bz2
cd wireshark-0.99.5
patch -p1 < ~/airprobe/gsm/patch/wireshark-0.99.5-gsm.patch
./configure CC=gcc-4.1 CXX=g++-4.1
make
sudo make install
```

zu letzt muss noch *gsm* kompiliert werden

```
cd airprobe/gsm
./bootstrap
./configure
make
```

3.4 Beschreibung von airprobe

3.4.1 gsmdecode

gsmdecode dient dazu die in hexadezimaler Schreibweise empfangenen Pakete in ein menschenlesbares Form zu bringen und auf der Shell auszugeben. Die Daten erhält es von *gsm-tvoid*.

3.4.2 gsm-tvoid

gsm-tvoid empfängt mithilfe von *GNU Radio* die **GSM**-Daten und leitet sie beispielsweise an *gsmdecode* zur Ausgabe weiter

```
cd airprobe/gsm-tvoid/src/python
./gsm_scan.py -R B -p d -c 24 -r e | ../../gsmdecode/src/gsmdecode -i
```

- **-R B** weist *gsm-tvoid* an das Modul B zu verwenden. Das **USRP** besitzt zwei Einschübe für Sende-/Empfangsmodule. In unserem Fall ist in Steckplatz B das *DBSRX* eingebaut.
- **-p d** gibt die Ausgabeform an. In unserem Fall d, was eine Ausgabe für *gsmdecode* in hexadezimaler Form erzeugt.
- **-c 24** gibt den **GSM-ARFCN** Kanal an, der mitgeschnitten werden soll.
- **-r e** definiert die Region. Hier e für Europa.

gsm-tvoid stellt in einer Grafik (siehe Abbildung 3 auf der nächsten Seite) zudem die Empfangsstärke der aktuell gewählten Frequenz dar.

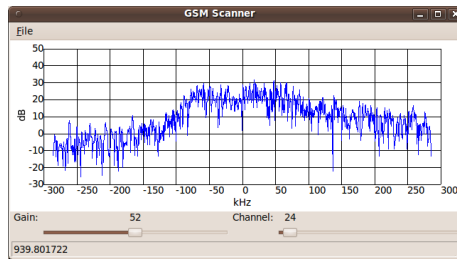


Abbildung 3: Grafische Darstellung der Empfangsstärke bei *gsm-tvoid*

3.4.3 gsm-receiver

gsm-receiver bietet eine ähnliche Funktionsweise wie *gsm-tvoid*. Viele der Funktionen wurden inzwischen auch unter den Programmen ausgetauscht, wodurch kein allzu großer Unterschied mehr zwischen ihnen besteht. In diesem Projekt wird im Allgemeinen *gsm-tvoid* verwendet.

3.4.4 gssm

Mit *gssm* ist es möglich die Daten aus der Luft live in *Wireshark* über ein TUN-Device zu analysieren.

Dafür muss der TUN-Kernel-Treiber geladen und das gsm TUN-Device erstellt werden

```
sudo modprobe tun
cd src/mktun
sudo ./mktun gsm
```

Nachdem in der Datei `../python/gssm_usrp.py` der Wert `c0` auf die gewünschte Frequenz gesetzt wurde, kann der Mitschnitt gestartet

```
cd ../python
./gssm_usrp.py
```

und in *Wireshark* das Interface gsm ausgewählt werden.

Da für *gssm* selbst eine andere *GNU Radio* Version als für *gsm-tvoid* oder *gsm-receiver* benötigt wird, kann von *gssm* auch nur *mktun* verwendet werden um das TUN-Device zu erstellen und *Wireshark* entsprechen gepatcht werden. *gsm-tvoid* kann das damit erstellte TUN-Device dann auch nutzen und ermöglicht auch so ohne *gssm* einen Live-Mitschnitt.

4 Live-CD

4.1 Einleitung

Homepage: <http://www.gsmdump.de>

Um den Einstieg möglichst einfach zu machen entstand während des Projekts eine Live-CD auf der bereits alle nötigen Programme installiert und eingerichtet sind. Somit muss nur noch diese CD gestartet, das **USRP** angeschlossen und das

entsprechende Programm gestartet werden.

Die einzelnen im Folgenden beschriebenen Programme und Skripte können über eine Konsole in einem beliebigen Pfad gestartet werden. Die Programme und Skripte sind entsprechend gelinkt.

Zum Einstieg empfiehlt sich zunächst ein Blick auf *gsmdump.sh* (siehe Kapitel 4.3.1 auf der nächsten Seite), welches einen guten Überblick über die gesamten Möglichkeiten bietet. Damit ergibt sich auch recht einfach ein erster Überblick über die empfangenen Mobilfunkstationen in der Nachbarschaft.

Die Live-CD basiert auf Ubuntu 10.04 (i386). An diesem wurden einige Änderungen vorgenommen, so wurden zahlreiche nicht direkt benötigte Pakete entfernt um Platz zu sparen und die CD unter der 700 MB Grenze einer realen CD zu halten. Die benötigte Software wurde auf einem Ubuntu 9.10 kompiliert und zu Debian-Paketen verpackt, welche dann in dem Live-System installiert wurden.

Die aktuellste Version der CD findet sich unter <http://www.gsmdump.de/downloads/gsmdump.iso>

4.2 Programme

4.2.1 GNU Radio

*GNU Radio*⁵ ist in Version 3.1.2 enthalten.

4.2.2 airprobe

Auf der Live-CD sind alle Programmteile von *airprobe*⁶ enthalten. Diese befinden sich im Verzeichnis `~/Desktop/airprobe`

4.2.3 Wireshark

*Wireshark*⁷ ist in Revision 26844 auf der Live-CD installiert. Diese Version dient dem Betrachten von zuvor aufgezeichneten pcap Dateien. Abbildung 4 auf der nächsten Seite zeigt *Wireshark* mit einer zur näheren Analyse geöffneten pcap Datei.

Weiterhin ist *Wireshark*⁸ auch in Version 0.99.5 mit einem Patch von *gssm* auf der Live-CD vorhanden. Dieser wird für Live-Mitschnitte benötigt und kann als *gsmshark* gestartet werden.

Erläuterungen zu den einzelnen Paketen finden sich in Anhang C auf Seite 25.

⁵Installation von *GNU Radio* siehe Kapitel 3.3.1 auf Seite 9

⁶Installation von *airprobe* siehe Kapitel 3.4 auf Seite 11

⁷Installation von *Wireshark* in Revision 26844 siehe Kapitel 3.3.3 auf Seite 10

⁸Installation von *Wireshark* in Version 0.99.5 siehe Kapitel 3.3.4 auf Seite 11

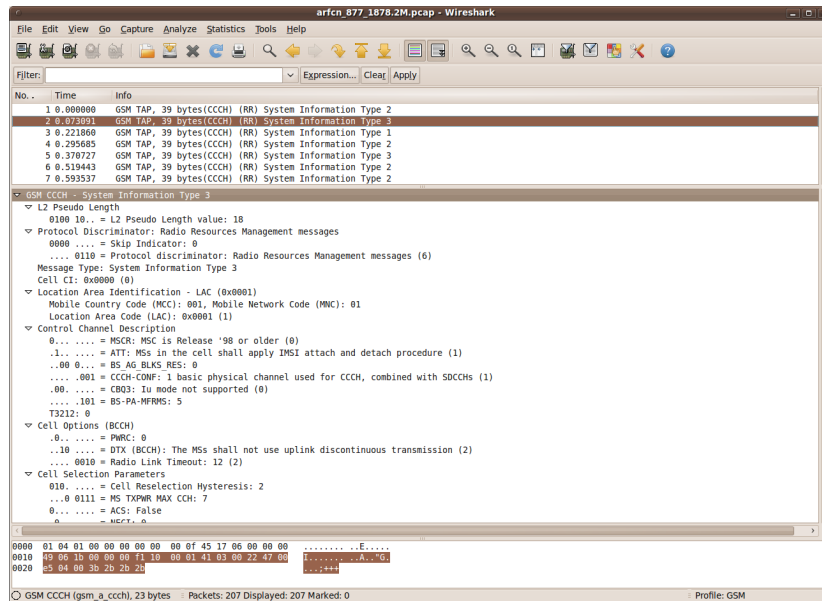


Abbildung 4: pcap Datei in *Wireshark*

4.2.4 GSMdump

Im Ordner `~/Desktop/gsmdump` befinden sich Skripte um einen einfachen Einstieg zu ermöglichen. Die Programme befinden sich alle im Pfad, können also direkt in einer Shell an jedem Ort auch ohne explizite Pfadangabe gestartet werden. Die einzelnen Programme werden in Kapitel 4.3 näher beschreiben.

4.3 Beschreibung der GSMdump Skripte

4.3.1 gsmdump.sh

Quellcode siehe Anhang D.1 auf Seite 36

gsmdump.sh erstellt eine Übersicht über alle in der Umgebung empfangbaren Basisstationen. Für den Einstieg lohnt sich zunächst ein Blick auf dieses Skript.

Dazu hört es auf allen Frequenzen nacheinander für eine definierte Zeit auf Daten und erstellt daraus Dateien im pcap Format. Diese wandelt *pcap4gsmdecode.py*⁹ daraufhin in ein hexadezimalen Format um, welches *gsmdecode*¹⁰ interpretieren kann. Die Ausgabe von *gsmdecode* wird von *analyse.py*¹¹ schließlich analysiert und in CSV Dateien abgelegt. Im letzten Schritt erstellt *summary.py*¹² aus diesen CSV Dateien lesbare Text Versionen, welche mit den pcap Dateien zu der entsprechenden Frequenz (und Absolute Radio Frequency Channel Number (**ARFCN**)) in einem nach dem aktuellen Datum benannten Ordner abgelegt werden.

⁹*pcap4gsmdecode.py* siehe Kapitel 4.3.2 auf Seite 16

¹⁰*gsmdecode* siehe Kapitel 3.4.1 auf Seite 11

¹¹*analyse.py* siehe Kapitel 4.3.3 auf Seite 17

¹²*summary.py* siehe Kapitel 4.3.4 auf Seite 17

Hilfe zur Verwendung von *gsmdump.sh* findet sich in Listing 2.

Listing 2: Verwendung von *gsmdump.sh*

```
$ gsmdump.sh -h
This script captures GSM traffic and creates a summary and pcap file. By default it iterates
all ARFCNs.

SYNOPSIS
    gsmdump.sh [ -s TMPDIR ] [ -t SECONDS ] [ -d DECIM ] [ -g GAIN ] [ -c ARFCN ] [ -R
    BOARD ] [ -o DSTDIR ] [ -k ] [ -l ] [ -h ]

OPTIONS
    -s TMPDIR
        Sets the TMPDIR.
    -t SECONDS
        Duration in SECONDS to stay on each ARFCN.
    -d DECIM
        Sets the DECIM value.
    -g GAIN
        Sets the GAIN value.
    -c ARFCN
        Scan only on ARFCN, don't iterate all.
    -R BOARD
        Specifies which BOARD (A or B) of the USRP to use.
    -o DSTDIR
        Sets the DSTDIR. Only if -l is not set.
    -k
        Keeps the raw dump files. Be carfull with this option, this can be much data.
    -l
        Run in an infinite loop. Not in combination with -o.
    -h
        Prints this help.
```

Abbildung 5 auf der nächsten Seite zeigt eine beispielhafte Ausgabe eines Scandurchgangs beim Aufruf von *gsmdump.sh*. Ohne explizite Parameter scannt das Skript dabei alle **ARFCNs** nacheinander auf vorhandene Basisstationen. Dabei sind im Ergebnis Zellen aller vier großen deutschen Mobilfunkanbieter zu sehen. Die empfangenen **IMSI**s wurden in der Abbildung geschwärzt, empfangene **TMSI**s wurden aus Platzgründen nicht aufgelistet.

Bei Aufruf dieses Skripts wird ein Ordner mit dem aktuellen Datum erstellt (z.B. 20100725204351). Darin werden die pcap Dateien (z.B. arfcn_24_939.8M.pcap) und die zugehörige Zusammenfassung in einer TXT Datei (z.B. arfcn_24_939.8M.txt) abgelegt. Zudem wird in diesem Ordner noch eine Zusammenfassung von allen **ARFCNs** erstellt (summary.txt).

networks:					
ARFCN	FREQ	Local Area Code	Cell identity	Mobile Network Code	Mobile Country Code
3	935.6M				
24	939.8M				
27	940.4M	0x7116	0x3150	T-Mobile Deutschland GmbH	Germany
45	944.0M	0x7116	0xf658	T-Mobile Deutschland GmbH	Germany
69	948.8M				
112	957.4M	0x02be	0x2c63	Vodafone D2 GmbH	Germany
645	1831.8M	0xc6ff	0x234a	02 (Germany) GmbH & Co. OHG	Germany
649	1832.6M	0xc6ff	0x250d	02 (Germany) GmbH & Co. OHG	Germany
668	1836.4M	0xc6ff	0xe875	02 (Germany) GmbH & Co. OHG	Germany
675	1837.8M	0xc6ff	0x4c35	02 (Germany) GmbH & Co. OHG	Germany
693	1841.4M				
696	1842.0M				
700	1842.8M				
713	1845.4M	0xc6ff	0x9962	02 (Germany) GmbH & Co. OHG	Germany
753	1853.4M	0x02be	0x2c63	Vodafone D2 GmbH	Germany
807	1864.2M	0x0dc9	0x580b	E-Plus Mobilfunk GmbH & Co. KG	Germany
820	1866.8M	0x2115	0xe8bb	E-Plus Mobilfunk GmbH & Co. KG	Germany
822	1867.2M	0x0dc9	0xf4f	E-Plus Mobilfunk GmbH & Co. KG	Germany
824	1867.6M				

IMSI:					
ARFCN 27:					
ARFCN 45:					
ARFCN 112:					
ARFCN 645:					
ARFCN 649:					
ARFCN 668:					
ARFCN 675:					
ARFCN 807:					
ARFCN 820:					
ARFCN 822:					
ARFCN 824:					

Abbildung 5: Beispielausgabe eines Scans von *gsmdump.sh*

4.3.2 pcap4gsmdecode.py

Quellcode siehe Anhang [D.2 auf Seite 41](#)

pcap4gsmdecode.py wandelt eine Datei im pcap Format in ein von *gsmdecode*¹³ lesbares Format um.

Dazu wird es mit der entsprechenden pcap Datei als Parameter aufgerufen und gibt diese in einem hexadezimalen Format aus. Diese Ausgabe kann dann mit *gsmdecode* weiterverarbeitet werden.

Ein Beispiel zur Verwendung von *pcap4gsmdecode.py* findet sich in Listing 3.

Listing 3: Verwendung von *pcap4gsmdecode.py*

```
$ pcap4gsmdecode.py 20100725204351/arfcn_24_939.8M.pcap | gsmdecode -i
```

¹³*gsmdecode* siehe Kapitel [3.4.1 auf Seite 11](#)

4.3.3 analyse.py

Quellcode siehe Anhang [D.3 auf Seite 42](#)

analyse.py erstellt aus der Ausgabe von *gsmdecode*¹⁴ eine Zusammenfassung über in der Nachbarschaft gefundene Netze, **IMSI**s und **TMSI**s im CSV Format. Hilfe zur Verwendung von *analyse.py* findet sich in Listing 4.

Listing 4: Verwendung von analyse.py

```
$ analyse.py -h
Usage: analyse.py [options]

Options:
  -h, --help show this help message and exit
  -a ARFCN, --arfcn=ARFCN
                        Input is an ARFCN
```

Die CSV Datei kann mit *summary.py*¹⁵ in ein für die Bildschirmausgabe optimiertes Format gebracht werden.

4.3.4 summary.py

Quellcode siehe Anhang [D.4 auf Seite 44](#)

summary.py erstellt aus den CSV Dateien von *analyse.py*¹⁶ eine menschenlesbare Ausgabe.

Hilfe zur Verwendung von *summary.py* findet sich in Listing 5.

Listing 5: Verwendung von summary.py

```
$ summary.py -h
Usage: summary.py [options]

Options:
  -h, --help show this help message and exit
  -s, --single Read from a single ARFCN file
```

¹⁴*gsmdecode* siehe Kapitel [3.4.1 auf Seite 11](#)

¹⁵*summary.py* siehe Kapitel [4.3.4](#)

¹⁶*analyse.py* siehe Kapitel [4.3.3](#)

4.3.5 arfcn.py

Quellcode siehe Anhang [D.5 auf Seite 48](#)

arfcn.py berechnet zu einer gegebenen **ARFCN** die zugehörige Frequenz, welche z.B. für die Zusammenarbeit mit *GNU Radio* benötigt wird.

Hilfe zur Verwendung von *arfcn.py* findet sich in Listing [6](#).

Listing 6: Verwendung von *arfcn.py*

```
$ arfcn.py -h
Usage: arfcn.py [options]

Options:
  -h, --help show this help message and exit
  -a ARFCN, --arfcn=ARFCN
                        Input is an ARFCN
```

Eine Beispielausgabe von *arfcn.py* findet sich in Listing [7](#).

Listing 7: Beispielausgabe von *arfcn.py -a 24*

```
$ arfcn.py -a 24
939.8M
```

4.3.6 gsmlive.sh

Quellcode siehe Anhang [D.6 auf Seite 49](#)

gsmlive.sh ermöglicht einen Live-Mitschnitt mit Wireshark. Das Programm öffnet automatisch Wireshark und startet den Mitschnitt.

Beim booten der Live-CD wird mit *mktun* aus *gssm*¹⁷ über

```
mktun gsm
```

das tun Device gsm erstellt, über welches dann die Daten live an Wireshark geleitet werden können.

Für den Live-Mitschnitt wird Wireshark (mit einem Patch von *gssm*) in Version 0.99.5 benötigt. Dieser ist auf der Live-CD zusätzlich als *gsmshark* verfügbar.

Hilfe zur Verwendung von *gsmlive.sh* findet sich in Listing [8 auf der nächsten Seite](#).

¹⁷*gssm* siehe Kapitel [3.4.4 auf Seite 12](#)

Listing 8: Verwendung von gsmlive.sh

```
$ gsmlive.sh -h
This script captures an specified ARFCN. It creates a live dump to the gsm tun device and a
pcap file for later analysis.

SYNOPSIS
    gsmlive.sh [ -c ARFCN ] [ -d DECIM ] [ -R BOARD ] [ -s ] [ -h ]

OPTIONS
    -c ARFCN
        Sets the ARFCN.
    -d DECIM
        Sets the DECIM value.
    -R BOARD
        Specifies which BOARD (A or B) of the USRP to use.
    -s
        Don't start Wireshark. So there will be an output to the shell, the pcap file and
        the gsm tun device. You can capture the gsm device with any other tool of
        course.
    -h
        Prints this help.
```

4.3.7 capture.sh

Quellcode siehe Anhang [D.7 auf Seite 51](#)

capture.sh schreibt die Rohdaten von einer bestimmten **ARFCN** oder, sollte keine explizite Angabe erfolgen, von allen **ARFCNs** in Dateien. Diese können im nächsten Schritt manuell mit *analysecfile.sh*¹⁸ ausgewertet werden.

Hilfe zur Verwendung von *capture.sh* findet sich in Listing 9.

Listing 9: Verwendung von capture.sh

```
$ capture.sh -h
This script captures GSM traffic and stores it in a cfile. By default it iterates all ARFCNs
.

SYNOPSIS
    capture.sh [ -t SECONDS ] [ -d DECIM ] [ -g GAIN ] [ -c ARFCN ] [ -R BOARD ] [ -d
    DSTDIR ] [ -h ]

OPTIONS
    -t SECONDS
        Duration in SECONDS to stay on each ARFCN.
    -d DECIM
        Sets the DECIM value.
    -g GAIN
        Sets the GAIN value.
    -c ARFCN
        Scan only on ARFCN, don't iterate all.
    -R BOARD
        Specifies which BOARD (A or B) of the USRP to use.
    -d DSTDIR
        Sets the DSTDIR.
    -h
        Prints this help.
```

¹⁸*analysecfile.sh* siehe Kapitel [4.3.8 auf der nächsten Seite](#)

4.3.8 analysefile.sh

Quellcode siehe Anhang [D.8 auf Seite 53](#)

analysefile.sh wertet mitgeschnittene Rohdaten analog zu *gsmdump.sh*¹⁹ aus.

Es eignet sich dazu zuvor mitgeschnittene Rohdaten, z.B. von *capture.sh*²⁰, auszuwerten.

Hilfe zur Verwendung von *analysefile.sh* findet sich in Listing 10.

Listing 10: Verwendung von analysefile.sh

```
$ analysefile.sh -h
This script creates a summary and pcap file. It reads from a captured cfile.

SYNOPSIS
    analysefile.sh [ -s TMPDIR ] [ -d DECIM ] [ -h ] CFILE

OPTIONS
    -s TMPDIR
        Sets the TMPDIR.
    -d DECIM
        Sets the DECIM value.
    -h
        Prints this help.
```

5 Weitere Software-Projekte

5.1 OpenBSC

Homepage: <http://openbsc.osmocom.org/trac/>

OpenBSC ist eine offene Implementierung eines Base Station Controller (**BSC**). Es kann im aktuellen Entwicklungsstand zwei verschiedene Base Transceiver Station (**BTS**), die *Siemens BS-11 microBTS* und die *ip.access nanoBTS*, ansprechen. Netzseitig implementiert *OpenBSC* zudem in Software noch das Mobile Switching Center (**MSC**), Home Location Register (**HLR**), Authentication Center (**AuC**), Visitor Location Register (**VLR**) und Equipment Identity Register (**EIR**).

Das Projekt möchte eine Plattform schaffen um mit **GSM** Netzseitig experimentieren zu können und etwa die Sicherheit von **GSM** zu analysieren. Es möchte nicht **GSM** bis ins letzte Detail implementieren oder eine produktiv nutzbare Lösung schaffen.

5.2 OpenBTS

Homepage: <http://openbts.sourceforge.net/>

¹⁹ *gsmdump.sh* siehe Kapitel [4.3.1 auf Seite 14](#)

²⁰ *capture.sh* siehe Kapitel [4.3.7 auf der vorherigen Seite](#)

OpenBTS bietet eine Schnittstelle zwischen **GSM** und der Software-Telefonanlage Asterisk. Es verwendet mit *GNU Radio* und einem **USRP** auch Hardwareseitig eine offene und sehr flexible Plattform. Das Projekt möchte im Vergleich zu *OpenBSC* mehr eine günstige Lösung für **GSM**-Netze bieten als eine möglichst Standardkonforme Lösung zum experimentieren mit **GSM** sein. So lagert es z.B. das **HLR** und **MSC** komplett an Asterisk aus.

5.3 OsmocomBB

Homepage: <http://bb.osmocom.org/trac/>

OsmocomBB möchte eine komplett freie Implementierung eines Treibers für den Chip und den zugehörigen **GSM**-Stack für ein Telefon realisieren. Das Projekt konzentriert sich dabei auf Telefone mit Ti Calypso/Iota/Rita Chipsätzen, z.B. dem MotorolaC123. Im Gegensatz zu *OpenBSC* und *OpenBTS* ist die Arbeit hier auf die Mobilseite gerichtet.

6 Fazit

Mit *GSMdump* ist es möglich den **BCCH**, Paging channel (**PCH**) und Access grant channel (**AGCH**) von **GSM** mitzuschneiden und anschließend zu analysieren. Dabei handelt es sich um Informationen einer Basisstation an ein bestimmtes Handy (z.B. über einen aus dem Netz kommenden anstehenden Anruf) oder auch an alle empfangenden Handys im Umkreis (z.B. über die Eigenschaften und Fähigkeiten der betreffenden Zelle, oder in der Nachbarschaft vorhandene weitere alternative Zellen). Es ist kein Werkzeug um unerlaubt fremde Telefongespräche mithören zu können, dazu sind Momentan auch noch einige technische Schwierigkeiten zu bewältigen, man darf aber gespannt sein was sich in dieser Richtung in Zukunft noch so alles Bewegt.

Die während des Projekts entstandenen Skripte vereinfachen die Nutzung der verwendeten Programme wie *airprobe*, da sie sich im Hintergrund um die Zusammenarbeit der einzelnen beteiligten Tools kümmern. Die Daten werden dabei entsprechend umgewandelt und aufbereitet, um sie anschließend komfortabel analysieren zu können.

Durch die entstandene Live-CD ist es möglich ohne viel Vorarbeit einen GSM Mitschnitt zu realisieren, ohne das dafür Änderungen auf einem eigenen System nötig sind oder zuerst Software kompiliert und installiert werden muss und die ganzen zuvor schon im Detail erwähnten Besonderheiten und Stolpersteine dabei berücksichtigt werden müssen. Die für Mitschnitt und Analyse benötigten Programme sind dazu bereits auf der Live-CD installiert und eingerichtet. Zudem sind bereits alle nötigen Schritte am System vorgenommen um direkt loslegen zu können.

An dieser Stelle vielen Dank an meine Betreuer Moritz Seltmann und Prof. Dr. Johannes Maucher für deren Unterstützung.

A Abkürzungsverzeichnis

AGCH	Access grant channel
ARFCN	Absolute Radio Frequency Channel Number
AuC	Authentication Center
BCCH	Broadcast Control Channel
BSC	Base Station Controller
BTS	Base Transceiver Station
CCCH	Common Control Channel
EIR	Equipment Identity Register
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HLR	Home Location Register
HSN	Hopping Sequence Number
IMSI	International Mobile Subscriber Identity
LAC	Location Area Code
MAIO	Mobile Allocation Index Offset
MNC	Mobile Network Code
MOC	Mobile Country Code
MS	Mobile Station
MSC	Mobile Switching Center
NCC	National Color Code/Network Color Code
PCH	Paging channel
RACH	Random Access Channel
SACCH	Slow associated control channel
SDCCH	Stand alone dedicated control channel
SIM	Subscriber Identity Module
TMSI	Temporary Mobile Subscriber Identity
USRP	Universal Software Radio Peripheral
VLR	Visitor Location Register

B Sequenzdiagramm

B.1 Paging Request

Abbildung 6 zeigt einen Paging Request für einen vom Mobilfunknetz kommenden Anruf. Pakete in Downlink-Richtung können mit dem **USRP** empfangen werden, in der Abbildung sind diese zusätzlich mit einem * gekennzeichnet.

Die Grafik wurde mit Hilfe des Nokia 3310²¹ erstellt, da dort auch die Pakete vom Telefon in Uplink-Richtung zur **BTS** sichtbar sind.

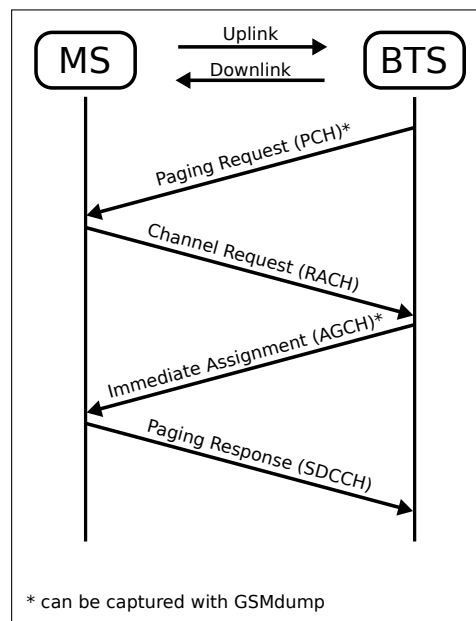


Abbildung 6: Sequenzdiagramm eines Paging Request

²¹siehe Kapitel 2.2.2 auf Seite 5

C Analysierte Mitschnitte

Im folgenden werden nun einige empfangene Pakete und deren Details näher erläutert.

C.1 BCCH

C.1.1 System Information Type 1

Zellweiter Broadcast vier mal in der Sekunde auf dem **BCCH**.

```
1 GSM CCCH - System Information Type 1
2   L2 Pseudo Length
3     0101 01.. = L2 Pseudo Length value: 21
4   Protocol Discriminator: Radio Resources Management messages
5     0000 .... = Skip Indicator: 0
6     .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
7   Message Type: System Information Type 1
8     00.. 000. = Format Identifier: bit map 0 (0x00)
9   List of ARFCNs = 83 24
10  RACH Control Parameters
11    10.. .... = Max retrans: Maximum 4 retransmissions (2)
12    ..10 01.. = Tx-integer: 12 slots used to spread transmission (9)
13    .... ..0. = CELL_BARR_ACCESS: The cell is not barred (0)
14    .... ...1 = RE: True
15    0000 0000 0000 0000 = ACC: 0x0000
16  SI 1 Rest Octets
17    NCH position: not present
18    Band Indicator: 1800
```

- **Zeile 9:** Traffic Channel auf **ARFCN** 83
- **Zeile 9:** Kanalnummer des **BCCH** auf **ARFCN** 24
- **Zeile 10ff:** RACH Control Parameters
 - **Zeile 11:** max. Wiederholungen des Zugriffsversuchs: 4
 - **Zeile 12:** zugewiesene Zufallszahl zwischen 3 und 50 für Warte-Zeitschlitz: 12
 - **Zeile 13:** erlaubte Nutzerklassen: 0 (alle)
 - **Zeile 14:** Verbindungsaufbau nach Abriss in selber Zelle erlaubt: Ja

Siehe auch [7, Seite 46ff]

C.1.2 System Information Type 2

Zellweiter Broadcast auf dem **BCCH** der die Frequenzen der Nachbarzellen des Providers (der selbe National Color Code/Network Color Code (**NCC**)) enthält.

```
1 GSM CCCH - System Information Type 2
2   L2 Pseudo Length
3     0101 10.. = L2 Pseudo Length value: 22
4   Protocol Discriminator: Radio Resources Management messages
5     0000 .... = Skip Indicator: 0
6     .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
7   Message Type: System Information Type 2
8   BCCH Frequency List
9     ..0. .... = EXT-IND: The information element carries the complete BA (0)
10    ...0 .... = BA-IND: 0
11    00.. 000. = Format Identifier: bit map 0 (0x00)
12    List of ARFCNs = 102 92 90 45 33 29 27 24
13    NCC Permitted
14      0000 1000 = NCC Permitted: 0x08
15    RACH Control Parameters
16      10.. .... = Max retrans: Maximum 4 retransmissions (2)
17      ..10 01.. = Tx-integer: 12 slots used to spread transmission (9)
18      .... ..0. = CELL_BARR_ACCESS: The cell is not barred (0)
19      .... ...1 = RE: True
20      0000 0000 0000 0000 = ACC: 0x0000
```

- **Zeile 12:** Liste benachbarter **ARFCNs**: 102 92 90 45 33 29 27 24
- **Zeile 15ff:** RACH analog zu System Information Type 1, siehe Kapitel **C.1.1 auf der vorherigen Seite**

Siehe auch [7, Seite 48]

C.1.3 System Information Type 2ter

Zellweiter Broadcast auf dem **BCCH**, der Informationen zu den Nachbarzellen enthält.

```
1 GSM CCCH - System Information Type 2ter
2   L2 Pseudo Length
3     0000 00.. = L2 Pseudo Length value: 0
4   Protocol Discriminator: Radio Resources Management messages
5     0000 .... = Skip Indicator: 0
6     .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
7   Message Type: System Information Type 2ter
8   Extended BCCH Frequency List
9     .00. .... = Multiband Reporting: 0
10    ...0 .... = BA-IND: 0
11    10.. 111. = Format Identifier: variable bit map (0x47)
12   List of ARFCNs = 596
13   SI 2ter Rest Octets
14     Empty
```

Siehe auch [6, Seite 295f]

C.1.4 System Information Type 2quater

Zellweiter Broadcast auf dem **BCCH**, der Informationen zu den Nachbarzellen enthält.

```
1 GSM CCCH - System Information Type 2quater
2   L2 Pseudo Length
3     0000 00.. = L2 Pseudo Length value: 0
4   Protocol Discriminator: Radio Resources Management messages
5     0000 .... = Skip Indicator: 0
6     .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
7   Message Type: System Information Type 2quater
8   SI 2quater Rest Octets
9     0... .... = BA-IND: 0
10    .0... .... = 3G BA-IND: 0
11    ...1. .... = Measurement Parameter Change Mark: 1
12    ...0 000. = SI2quater Index: 0
13    .... ..0 000. .... = SI2quater Count: 0
14   3G Neighbour Cell Description
15     UTRAN FDD Description
16       .101 0100 1010 100. = FDD UARFCN: 10836
17       FDD Indic0: 0
18       Nr of FDD Cells : 31
19       UTRAN FDD Description
20         Field is 0 bits long
21   3G Measurement Parameters Description
22     0111 .... = Qsearch I: Always (7)
23     .... 0... = QSearch C Initial: use Qsearch I
24     .... ..00 00.. .... = FDD Qoffset: always select a cell if acceptable (0)
25     ...0. .... = FDD Rep Quant: RSCP
26     ...0 0... = FDD Multirat Reporting: 0
27     .... .101 = FDD Qmin: -10 dB (5)
28   GPRS 3G Measurement Parameters Description
29     ..01 11.. = Qsearch P: Always (7)
30     .... ..1. = 3G Search Prio: 3G cells may be searched when BSIC decoding is
        required
31   3G Additional Measurement Parameters Description
32     .... .000 = FDD Qmin Offset: 0 dB (0)
33     0111 .... = FDD RSCPmin: -100 dBm (7)
```

C.1.5 System Information Type 3

Zellweiter Broadcast auf dem **BCCH** vier mal in der Sekunde.

```
1 GSM CCCH - System Information Type 3
2 L2 Pseudo Length
3 0100 10.. = L2 Pseudo Length value: 18
4 Protocol Discriminator: Radio Resources Management messages
5 0000 .... = Skip Indicator: 0
6 .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
7 Message Type: System Information Type 3
8 Cell CI: 0xf657 (63063)
9 Location Area Identification - LAC (0x7116)
10 Mobile Country Code (MCC): 262, Mobile Network Code (MNC): 01
11 Location Area Code (LAC): 0x7116 (28950)
12 Control Channel Description
13 1... .... = MSCR: MSC is Release '99 onwards (1)
14 .1... .... = ATT: MSs in the cell shall apply IMSI attach and detach procedure (1)
15 ..01 1... = BS_AG_BLKs_RES: 3
16 .... .000 = CCCH-CONF: 1 basic physical channel used for CCCH, not combined with
SDCCHs (0)
17 .00. .... = CBQ3: Iu mode not supported (0)
18 .... .110 = BS-PA-MFRMS: 6
19 T3212: 60
20 Cell Options (BCCH)
21 .1... .... = PWR: 1
22 ..01 .... = DTX (BCCH): The MSs shall use uplink discontinuous transmission (1)
23 .... 0101 = Radio Link Timeout: 24 (5)
24 Cell Selection Parameters
25 011. .... = Cell Reselection Hysteresis: 3
26 ...0 0101 = MS TXPWR MAX CCH: 5
27 0... .... = ACS: False
28 .0... .... = NECI: 0
29 ..00 0100 = RXLEV-ACCESS-MIN: -107 <= x < -106 dBm (4)
30 RACH Control Parameters
31 10.. .... = Max retrans: Maximum 4 retransmissions (2)
32 ..10 01.. = Tx-integer: 12 slots used to spread transmission (9)
33 .... .0. = CELL_BARR_ACCESS: The cell is not barred (0)
34 .... .1 = RE: True
35 0000 0000 0000 0000 = ACC: 0x0000
36 SI 3 Rest Octets
37 SYSTEM INFORMATION TYPE 2ter message is available
38 Early Classmark Sending is allowed
39 GPRS Indicator
40 RA Colour: 6
41 .0... .... = SI13 Position: SYSTEM INFORMATION TYPE 13 message is sent on BCCH
Norm (0)
42 3G Early Classmark Sending Restriction: Neither UTRAN, CDMA2000 nor GERAN IU MODE
CLASSMARK CHANGE message shall be sent with the Early classmark sending
43 .... 0... = SI2quater Position: SYSTEM INFORMATION TYPE 2 quater message is sent on
BCCH Norm
```

- **Zeile 8:** Die vom Provider für diese Zelle vergebene ID: 0xf657
- **Zeile 10:** Der Mobile Country Code (**MCC**) definiert das Land, hier 262 für Deutschland, der Mobile Network Code (**MNC**) den Provider, hier 01 für T-Mobile
- **Zeile 11:** Der Location Area Code (**LAC**) definiert die Location in welcher sich die Basisstation befindet: 0x7116
- **Zeile 14:** Mobiltelefone sollen sich in der Zelle an-/abmelden (ATT)
- **Zeile 15:** Anzahl der verwendeten Paging Kanäle für **AGCH** (BS_AG_BLKs_RES)
- **Zeile 16:** Konfiguration des Common Control Channel (**CCCH**) (CCCH-CONF): im **CCCH** werden keine Zeitschlitz für einen Stand alone dedicated control channel (**SDCCH**) reserviert

- **Zeile 18:** Zugewiesene Paging-Untergruppe: 6 Multiframe-Perioden (BS-PA-MFRMS)
- **Zeile 19:** Zeit bis zur Aktualisierung des Standorts: 60 Stunden
- **Zeile 21:** Leistung des MS wird geregelt (PWRC)
- **Zeile 22:** Diskontinuierliche Übertragung im Uplink, während Gesprächspausen Strom sparen und nicht senden
- **Zeile 23:** Verbindung abbrechen nach 24 nicht dekodierbaren Slow associated control channel (**SACCH**) Rahmen (Radio Link Timeout)
- **Zeile 25:** Beharrungsfaktor um an Zellgrenzen nicht hin und her zu wechseln (Cell Reselection Hysteresis): 3 dBm
- **Zeile 26:** max. Sendeleistung die ein Mobile auf Random Access Channel (**RACH**) ausstrahlen darf (MS TXPWR MAX CCH). Wert in dBm.
- **Zeile 29:** nötiger Pegel für das Mobile um auf **BTS** zuzugreifen (RXLEV-ACCESS-MIN): -107 dBm
- **Zeile 30ff:** **RACH** analog zu System Information Type 1, siehe Kapitel **C.1.1** auf Seite 25
- **Zeile 36ff:** Informationen zur General Packet Radio Service (**GPRS**) Fähigkeit der Zelle (SI 3 Rest Octets)

Siehe auch [7, Seite 49ff]

C.1.6 System Information Type 4

```
1 GSM CCCH - System Information Type 4
2   L2 Pseudo Length
3     0011 00.. = L2 Pseudo Length value: 12
4   Protocol Discriminator: Radio Resources Management messages
5     0000 .... = Skip Indicator: 0
6     .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
7   Message Type: System Information Type 4
8   Location Area Identification - LAC (0x7116)
9     Mobile Country Code (MCC): 262, Mobile Network Code (MNC): 01
10    Location Area Code (LAC): 0x7116 (28950)
11   Cell Selection Parameters
12     011. .... = Cell Reselection Hysteresis: 3
13     ...0 0101 = MS TXPWR MAX CCH: 5
14     0... .... = ACS: False
15     .0.. .... = NECI: 0
16     ..00 0100 = RXLEV-ACCESS-MIN: -107 <= x < -106 dBm (4)
17   RACH Control Parameters
18     10.. .... = Max retrans: Maximum 4 retransmissions (2)
19     ..10 01.. = Tx-integer: 12 slots used to spread transmission (9)
20     .... ..0. = CELL_BARR_ACCESS: The cell is not barred (0)
21     .... ...1 = RE: True
22     0000 0000 0000 0000 = ACC: 0x0000
23   SI 4 Rest Octets
24     SI4 Rest Octets_0
25       GPRS Indicator
26         RA Colour: 6
27         .... ..0. = SI13 Position: SYSTEM INFORMATION TYPE 13 message is sent on BCCH
28           Norm (0)
           Break Indicator: Additional parameters "SI4 Rest Octets_S" are not sent in SYSTEM
             INFORMATION TYPE 7 and 8
```

Analog zu System Information Type 3, siehe Kapitel [C.1.5 auf Seite 29](#)

Siehe auch [\[7, Seite 54f\]](#)

C.1.7 System Information Type 13

Broadcast zu GPRS Fähigkeiten der BTS

```
1 GSM CCCH - System Information Type 13
2 L2 Pseudo Length
3 0000 00.. = L2 Pseudo Length value: 0
4 Protocol Discriminator: Radio Resources Management messages
5 0000 .... = Skip Indicator: 0
6 .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
7 Message Type: System Information Type 13
8 SI 13 Rest Octets
9 .100 .... = BCCH Change Mark: 4
10 .... 0000 = SI Change Field: Update of unspecified SI message or SI messages (0)
11 .01. .... = SI13 Change Mark: 1
12 GPRS Mobile Allocation
13 ...0 0000 1... .... = HSN: 1
14 ...0 0000 1... .... = MA Length: 1
15 MA Bitmap: 11
16 .... 0000 0001 .... = RAC: 1
17 .... 0... = SPGC CCCH Sup: SPLIT_PG_CYCLE is not supported on CCCH in this cell
18 .... .110 = Priority Access Thr: Packet access is allowed for priority level 1 to 4
    (6)
19 00... .... = Network Control Order: NCO (0)
20 GPRS Cell Options
21 ..00 .... = NMO: Network Mode of Operation I (0)
22 .... 011. = T3168: 2000 ms (3)
23 .... ...1 11.. .... = T3192: 200 ms (7)
24 ..01 1... = DRX Timer Max: 4 s (3)
25 .... .0.. = Access Burst Type: 8-bit format shall be used
26 .... ..0. = Control Ack Type: Default format is four access bursts
27 .... ...1 001. .... = BS CV Max: 9
28 .... 001. = PAN Dec: 1
29 .... ...0 10.. .... = PAN Inc: 2
30 ..00 0... = PAN Max: maximum value allowed for counter N3102 is 4 (0)
31 GPRS Cell Options Extension Information
32 Extension Length: 14
33 .... .0.. = EGPRS Packet Channel Request: Use EGPRS PACKET CHANNEL REQUEST
    message for uplink TBF establishment on the PRACH
34 .... ..01 10.. .... = BEP Period: 10 (6)
35 ..0. .... = PFC Feature Mode: The network does not support packet flow context
    procedures
36 .... .0.. = DTM Support: The cell does not support DTM procedures
37 .... 0... = BSS Paging Coordination: The cell does not support Circuit-
    Switched paging coordination
38 .... .1.. = CCN Active: CCN is enabled in the cell
39 .... ...1. = NW Ext UTBF: The extended uplink TBF mode is supported by the
    network
40 .... ...0 = Multiple TBF Capability: The cell does not support multiple TBF
    procedures
41 1... .... = Ext UTBF No Data: The mobile station may refrain from sending a
    PACKET UPLINK DUMMY CONTROL BLOCK message when there is no other RLC/MAC
    block ready to send in an uplink radio block allocated by the network
42 ..0.. .... = DTM Enhancements Capability: The cell does not support enhanced
    DTM CS establishment and enhanced DTM CS release procedures
43 GPRS Power Control Parameters
44 ..000 0... = Alpha: 0.0 (0)
45 .... .001 01.. .... = T Avg W:  $2^{(5/2)}$  / 6 multiframe (5)
46 ..00 011. = T Avg T:  $2^{(3/2)}$  / 6 multiframe (3)
47 .... ...0 = PC Meas Chan: Downlink measurements for power control shall be made
    on BCCH
48 0001 .... = N Avg I:  $2^{(1/2)}$  (1)
```


C.2 CCCH

C.2.1 Immediate Assignment

Die **BTS** weist dem Mobile einen **SDCCH** Kanal zu, nachdem dieses auf dem RACH einen Channel Request gestellt hat. Den kompletten Ablauf zeigt Abbildung 6 auf Seite 24.

```
1 GSM CCCH - Immediate Assignment
2 L2 Pseudo Length
3   0011 00.. = L2 Pseudo Length value: 12
4 Protocol Discriminator: Radio Resources Management messages
5   0000 .... = Skip Indicator: 0
6   .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
7 Message Type: Immediate Assignment
8 Page Mode
9   .... ..11 = Page Mode: Same as before (3)
10 Dedicated mode or TBF
11   .000 .... = Dedicated mode or TBF: This message assigns a dedicated mode resource (0)
12 Channel Description
13   0100 0... = SDCCH/8 + SACCH/C8 or CBCH (SDCCH/8), Subchannel 1
14   .... .001 = Timeslot: 1
15   010. .... = Training Sequence: 2
16   ...1 .... = Hopping channel: Yes
17 Hopping channel: MAIO 0
18 Hopping channel: HSN 1
19 Request Reference
20   Random Access Information (RA): 226
21   0010 1... = T1': 5
22   .... .010 101. .... = T3: 21
23   ...0 1001 = T2: 9
24   [RFN: 7263]
25 Timing advance value: 2
26 Mobile Allocation
27   Length: 1
28   Bitmap of increasing ARFCNs included in the Mobile Allocation: 11000000
29 IA Rest Octets
30   Data(Not decoded)
```

- **Zeile 11:** fest zugewiesener Kanal, nicht wie GPRS nur zeitweilig (Dedicated mode or TBF)
- **Zeile 13:** Kanalkonfiguration (Channel Description): SDCCH/8 + SACCH/C8 or CBCH (SDCCH/8), Subchannel 1
- **Zeile 14:** Zeitschlitz: 1
- **Zeile 16:** Frequenzsprungverfahren wird verwendet (Hopping channel): Yes
- **Zeile 17f:** Reihenfolge der nacheinander verwendeten Frequenzen und Zeitschlitz (Mobile Allocation Index Offset (**MAIO**)/Hopping Sequence Number (**HSN**))
 - **Zeile 17:** Zahl max. zur Verfügung stehender Frequenzen (**MAIO**)
 - **Zeile 18:** **HSN** zwischen 0-63 für Frequenzsprunggenerator: 1
- **Zeile 20:** aktuelle Framenummer der BTS wird dem Mobile mitgeteilt (Random Access Information)
- **Zeile 25:** Entfernungsabhängiger Korrekturfaktor für Paketlaufzeit (Timing advance value): 2

Siehe auch [7, Seite 58ff]

C.3 PCH

C.3.1 Paging Request Type 1 (**IMSI**)

Anruf aus dem Mobilfunknetz an das Mobile. Den kompletten Ablauf zeigt Abbildung 6 auf Seite 24.

Type 1 kann gleichzeitig bis zu zwei Teilnehmer rufen. Kann aber auch leer sein.

```
1 GSM CCCH - Paging Request Type 1
2 L2 Pseudo Length
3 0011 00.. = L2 Pseudo Length value: 12
4 Protocol Discriminator: Radio Resources Management messages
5 0000 .... = Skip Indicator: 0
6 .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
7 Message Type: Paging Request Type 1
8 Page Mode
9 .... ..00 = Page Mode: Normal paging (0)
10 Channel Needed
11 ..00 .... = Channel 1: Any channel (0)
12 00.. .... = Channel 2: Any channel (0)
13 Mobile Identity - Mobile Identity 1 - IMSI (262010123456789)
14 Length: 8
15 0010 .... : Identity Digit 1: 2
16 .... 1... = Odd/even indication: Odd number of identity digits (1)
17 .... .001 = Mobile Identity Type: IMSI (1)
18 BCD Digits: 262010123456789
19 P1 Rest Octets
20 Packet Page Indication 1: Paging procedure for RR connection establishment
21 Packet Page Indication 2: Paging procedure for RR connection establishment
```

- **Zeile 17:** Das Gerät wird anhand seiner **IMSI** gerufen
- **Zeile 18:** Die **IMSI**: 262 steht ist die Länderkennung für Deutschland, 01 der Mobilfunkanbieter (T-Mobile), der Rest bildet die Teilnehmernummer

Siehe auch [7, Seite 45]

C.3.2 Paging Request Type 1 (TMSI)

Anruf aus dem Mobilfunknetz an das Mobile. Den kompletten Ablauf zeigt Abbildung 6 auf Seite 24.

```
1 GSM CCCH - Paging Request Type 1
2   L2 Pseudo Length
3     0010 01.. = L2 Pseudo Length value: 9
4   Protocol Discriminator: Radio Resources Management messages
5     0000 .... = Skip Indicator: 0
6     .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
7   Message Type: Paging Request Type 1
8   Page Mode
9     .... ..00 = Page Mode: Normal paging (0)
10  Channel Needed
11    ..00 .... = Channel 1: Any channel (0)
12    00.. .... = Channel 2: Any channel (0)
13  Mobile Identity - Mobile Identity 1 - TMSI/P-TMSI (0x12345678)
14    Length: 5
15    1111 .... : Unused
16    .... 0... = Odd/even indication: Even number of identity digits (0)
17    .... .100 = Mobile Identity Type: TMSI/P-TMSI (4)
18    TMSI/P-TMSI: 0x12345678
19  P1 Rest Octets
20    Packet Page Indication 1: Paging procedure for RR connection establishment
21    Packet Page Indication 2: Paging procedure for RR connection establishment
```

- Zeile 17: Das Gerät wird anhand seiner TMSI gerufen
- Zeile 18: Die gerufenen TMSI

Siehe auch [7, Seite 45]

D Quellcode der Live-CD

D.1 gsmdump.sh

```
1  #!/bin/bash
2  #
3  # This script captures GSM traffic and creates a summary and pcap file. By default it
4  #
5  #
6  # predefine config values
7  TMP=/tmp/gsmdump
8  DURATION=4
9  DECIM=112
10 GAIN=52
11 ARFCN=""
12 BOARD="B"
13 AKTPWD='pwd'
14 DST=$AKTPWD/"$(date +%Y%m%d%H%M%S)'
15 KEEP=1
16 LOOP=1
17 DSTSET=1
18 ARFCNSET=1
19
20 # check for parameters and change config values
21 while getopts "s:t:d:g:c:R:o:klh" optname; do
22     case "$optname" in
23         "s")
24             TMP=$OPTARG
25             ;;
26         "t")
27             DURATION=$OPTARG
28             ;;
29         "d")
30             DECIM=$OPTARG
31             ;;
32         "g")
33             GAIN=$OPTARG
34             ;;
35         "c")
36             ARFCN=$OPTARG
37             ARFCNSET=0
38             ;;
39         "R")
40             BOARD=$OPTARG
41             ;;
42         "o")
43             DST=$OPTARG
44             DSTSET=0
45             ;;
46         "k")
47             KEEP=0
48             ;;
49         "l")
50             LOOP=0
51             ;;
52         "h")
53             echo "This script captures GSM traffic and creates a summary and pcap
54             file. By default it iterates all ARFCNs."
55             echo
56             echo "SYNOPSIS"
57             echo " gsmdump.sh [ -s TMPDIR ] [ -t SECONDS ] [ -d DECIM ] [ -g GAIN ]
58             [ -c ARFCN ] [ -R BOARD ] [ -o DSTDIR ] [ -k ] [ -l ] [ -h ]"
59             echo
60             echo "OPTIONS"
61             echo " -s TMPDIR"
62             echo " Sets the TMPDIR."
63             echo " -t SECONDS"
64             echo " Duration in SECONDS to stay on each ARFCN."
65             echo " -d DECIM"
66             echo " Sets the DECIM value."
67             echo " -g GAIN"
```

```

66         echo " Sets the GAIN value."
67         echo " -c ARFCN"
68         echo " Scan only on ARFCN, don't iterate all."
69         echo " -R BOARD"
70         echo " Specifies which BOARD (A or B) of the USRP to use."
71         echo " -o DSTDIR"
72         echo " Sets the DSTDIR. Only if -l is not set."
73         echo " -k"
74         echo " Keeps the raw dump files. Be carfull with this option, this can
        be much data."
75         echo " -l"
76         echo " Run in an infinite loop. Not in combination with -o."
77         echo " -h"
78         echo " Prints this help."
79         exit 0
80     ;;
81     "?")
82         echo "Unknown option $OPTARG"
83         echo
84         $0 -h
85         exit 1
86     ;;
87     ":")
88         echo "No argument value for option $OPTARG"
89         echo
90         $0 -h
91         exit 1
92     ;;
93     *)
94         echo "Unknown error while processing options"
95         echo
96         $0 -h
97         exit 1
98     ;;
99     esac
100 done
101
102 # check parameter
103 if [ $DSTSET -eq 0 ] && [ $LOOP -eq 0 ]; then
104     echo "ERROR: -o and -l set at the same time."
105     echo
106     $0 -h
107     exit 1
108 fi
109
110 # DST is relative
111 if ! [ 'echo -n $DST | grep -e "/.*"' ]; then
112     DST=$AKTPWD/"$DST"
113 fi
114
115 # check or create the destination
116 if ! [ -d $DST ]; then
117     mkdir -p $DST
118 fi
119
120 # check or create the tmp
121 if ! [ -d $TMP ]; then
122     mkdir -p $TMP
123 fi
124 cd $TMP
125
126 # Look for usrp_rx_cfile.py
127 USRP_PROG=usrp_rx_cfile.py
128 while :; do
129     which "$USRP_PROG" > /dev/null
130     if [ $? -eq 0 ]; then
131         break
132     fi
133     USRP_PROG=/usr/share/gnuradio/usrp/usrp_rx_cfile.py
134     which "$USRP_PROG" > /dev/null
135     if [ $? -eq 0 ]; then
136         break
137     fi
138

```

```

139     echo "ERROR: usrp_rx_cfile.py not found. Make sure it's in your PATH!"
140     exit 1
141 done
142
143 # calculate samples for the specified duration
144 SAMPLES='expr 64000000 / $DECIM '*' $DURATION'
145
146 # rm old netlists
147 if [ -f netlist.nets ]; then
148     rm netlist.nets
149 fi
150 if [ -f netlist.imsis ]; then
151     rm netlist.imsis
152 fi
153 if [ -f netlist.tmsis ]; then
154     rm netlist.tmsis
155 fi
156
157 # loop forever?
158 LOOPING=0
159 while [ $LOOPING -eq 0 ]; do
160
161     # no single ARFCN specified
162     if [ $ARFCNSET -eq 1 ]; then
163
164         # Now iterate all, about 1000 RFCS, each for -t seconds
165         # GSM 900 (ARFCN 1 - 124) && GSM 1800 (ARFCN 512 885)
166         for ARFCN in `seq 1 124 && seq 512 885`; do
167
168             FREQ=$(arfcn.py -a $ARFCN)
169             FILE="$TMP/arfcn_${ARFCN}_${FREQ}.cfile"
170             echo "capturing for $DURATION seconds ($SAMPLES samples) on module
171                 $BOARD on ARFCN $ARFCN ($FREQ)Hz to $FILE"
172             $USRP_PROG -R $BOARD -g $GAIN -d "$DECIM" -f "$FREQ" -N $SAMPLES $FILE
173                 > /dev/null 2> /dev/null
174             if [ $? -gt 0 ]; then
175                 echo "WARNING: something went wrong while capturing (ARFCN
176                     $ARFCN), continuing on the next ARFCN anyway ..."
177             fi
178             gsm_scan_light.py -SN -pd -d "$DECIM" -I "$FILE" > /dev/null 2> /dev/
179                 null
180             if [ -e tvoid.pcap ] && [ -e tvoid-burst.pcap ]; then
181                 if [ `stat -c %s tvoid.pcap` -gt 24 ]; then
182                     mv tvoid.pcap $DST/arfcn_${ARFCN}_${FREQ}.pcap
183                     mv tvoid-burst.pcap $DST/arfcn_${ARFCN}_${FREQ}-burst.pcap
184                     pcap4gsmdecode.py $DST/arfcn_${ARFCN}_${FREQ}.pcap |
185                         gsmdecode -i | analyse.py -a $ARFCN
186                     # check if there are any networks
187                     if [ -f netlist.single.nets ]; then
188                         summary.py -s > $DST/arfcn_${ARFCN}_${FREQ}.txt
189                         rm netlist.single.nets
190                     fi
191                 fi
192             fi
193             if [ $KEEP -eq 0 ]; then
194                 mv $FILE $DST/.
195             else
196                 rm $FILE
197             fi
198         done
199     fi
200
201     # capture only a single ARFCN
202     else
203         FREQ=$(arfcn.py -a $ARFCN)
204         FILE="$TMP/arfcn_${ARFCN}_${FREQ}.cfile"
205         echo "capturing for $DURATION seconds ($SAMPLES samples) on module $BOARD on
206             ARFCN $ARFCN ($FREQ)Hz to $FILE"
207         $USRP_PROG -R $BOARD -g $GAIN -d "$DECIM" -f "$FREQ" -N $SAMPLES $FILE > /dev/
208             null 2> /dev/null
209         if [ $? -gt 0 ]; then
210             echo "WARNING: something went wrong while capturing (ARFCN $ARFCN) ..."
211         fi
212     fi
213 done

```

```

206         fi
207         gsm_scan_light.py -SN -pd -d "$DECIM" -I "$FILE" > /dev/null 2> /dev/null
208         if [ -e tvoid.pcap ] && [ -e tvoid-burst.pcap ]; then
209             if [ 'stat -c %s tvoid.pcap' -gt 24 ]; then
210                 mv tvoid.pcap $DST/arfcn_${ARFCN}_$FREQ.pcap
211                 mv tvoid-burst.pcap $DST/arfcn_${ARFCN}_$FREQ-burst.pcap
212                 pcap4gsmdecode.py $DST/arfcn_${ARFCN}_$FREQ.pcap | gsmdecode -i
213                     | analyse.py -a $ARFCN
214                 # check if there are any networks
215                 if [ -f netlist.single.nets ]; then
216                     summary.py -s > $DST/arfcn_${ARFCN}_$FREQ.txt
217                     rm netlist.single.nets
218             fi
219         fi
220
221         if [ $KEEP -eq 0 ]; then
222             mv $FILE $DST/.
223         else
224             rm $FILE
225         fi
226
227     fi
228
229     # create a complete summary, if something was found
230     if [ -f netlist.nets ]; then
231         summary.py | tee $DST/summary.txt
232     else
233         echo
234         echo "I'm sorry, but no network was found"
235         echo
236     fi
237
238     # rm old netlists
239     if [ -f netlist.nets ]; then
240         rm netlist.nets
241     fi
242     if [ -f netlist.imsis ]; then
243         rm netlist.imsis
244     fi
245     if [ -f netlist.tmsis ]; then
246         rm netlist.tmsis
247     fi
248     if [ -f netlist.single.nets ]; then
249         rm netlist.single.nets
250     fi
251     if [ -f netlist.single.imsis ]; then
252         rm netlist.single.imsis
253     fi
254     if [ -f netlist.single.tmsis ]; then
255         rm netlist.single.tmsis
256     fi
257     if [ -f gsm-receiver.pcap ]; then
258         rm gsm-receiver.pcap
259     fi
260     if [ -f gsm-receiver-burst.pcap ]; then
261         rm gsm-receiver-burst.pcap
262     fi
263     if [ -f speech.gsm ]; then
264         rm speech.gsm
265     fi
266     if [ -f cfile2.out ]; then
267         rm cfile2.out
268     fi
269
270     # check if loop forever?
271     if [ $LOOP -eq 0 ]; then
272         # set new DST
273         DST=$AKTPWD/"$(date +%Y%m%d%H%M%S)"
274         # check or create the destination
275         if ! [ -d $DST ]; then
276             mkdir -p $DST
277         fi
278     else

```

```
279         # no loop
280         LOOPING=1
281     fi
282
283 done
284
285 exit 0
```


D.2 pcap4gsmdecode.py

```
1  #!/usr/bin/env python
2  #
3  # This script converts a pcap file so gsmdecode can read it.
4  #
5
6  import sys
7  import pcap
8
9  def print_packet(pktlen, data, timestamp):
10     # check if len(data) is < 17, cause the next for starts at position 16
11     if not data or len(data) < 17:
12         return
13
14     out = ""
15     for i in range(16, len(data)-1):
16         if len(hex(ord(data[i]))[2:]) == 1:
17             # convert the data in HEX, and add zero before
18             out = out + " 0" + str(hex(ord(data[i]))[2:])
19         else:
20             # same, but no zero is needed
21             out = out + " " + str(hex(ord(data[i]))[2:])
22
23     print out
24
25  def main():
26
27     # help the user
28     if len(sys.argv) < 2:
29         print 'usage: pcap4gsmdecode.py <file.pcap>'
30         sys.exit(1)
31
32     # create a pcap Object
33     p = pcap.pcapObject()
34     # and use it for the given file
35     p.open_offline(sys.argv[1])
36
37     pkt = (1,2)
38     pkt = p.next()
39     # iterate the packages, if there are more packages, print them
40     while type(pkt) is tuple:
41         apply(print_packet, pkt)
42         pkt = p.next()
43
44  if __name__ == '__main__':
45     main()
```

D.3 analyse.py

```
1  #!/usr/bin/env python
2  #
3  # This script analyses the output of gsmdecode for a summary and writes found networks,
4  IMSI and TMSI into a CSVs.
5  #
6  import sys
7  import re
8  from optparse import OptionParser
9
10 def parse_options():
11     parser = OptionParser()
12     parser.add_option("-a", "--arfcn", type="int", dest="arfcn",
13                       help="Input is an ARFCN")
14
15     (options, args) = parser.parse_args()
16     if (len(args) != 0):
17         parser.print_help()
18         sys.exit(1)
19
20     return options
21
22 def freq(arfcn):
23
24     # GSM 900
25     if arfcn > 0 and arfcn <= 124:
26         freq = 890 + 0.2*arfcn + 45
27
28     # GSM 1800
29     elif arfcn >= 512 and arfcn <= 885:
30         freq = 1710.2 + 0.2*(arfcn - 512) + 95
31
32     else:
33         freq = 0
34
35     return freq
36
37 def main():
38
39     # get parameter
40     options = parse_options()
41
42     # check options
43     # nothing to check
44
45     input=sys.stdin.readlines()
46     nets = []
47     imsis = []
48     tmsis = []
49     file_nets = ""
50     file_imsis = ""
51     file_tmsis = ""
52
53     for i in range(0, len(input)):
54         input[i] = re.sub("\n", "", input[i])
55     for i in range(0, len(input)):
56         if re.search("Cell identity", input[i]):
57             nets = input[i:i+4]
58             if re.search("Type of identity: IMSI", input[i]):
59                 imsis.append(re.sub(r".*ID\{7/odd\}: (.*)$", r"\1", input[i+1]))
60             if re.search("Type of identity: TMSI", input[i]):
61                 tmsis.append(re.sub(r".*ID\{4/even\}: (.*)$", r"\1", input[i+1]))
62     if nets == []:
63         if not (options.arfcn):
64             file_nets = "?" + ";" + "?" + ";;;"
65         else:
66             file_nets = str(options.arfcn) + ";" + str(freq(options.arfcn)) + "M" +
67                 ";;;"
68     else:
69         if not (options.arfcn):
```

```

69         file_nets = "?" + ";" + "?" + ";" + re.sub(r".*\[[^\]]*\]\ Cell
            identity.*",r"\1",nets[0]) + ";" + re.sub(r".*: [^\]* ([^\])* *
            Mobile Country Code .*$",r"\1",nets[1]) + ";" + re.sub(r".*Mobile
            Country Code \((.*)\)$",r"\1",nets[1]) + ";" + re.sub(r".*: [^\
            ]* ([^\])* *Mobile Network Code .*$",r"\1",nets[2]) + ";" + re.
            sub(r".*Mobile Network Code \((.*)\)$",r"\1",nets[2]) + ";" + re.
            sub(r".*\[[^\]]*\]\ Local Area Code.*",r"\1",nets[3])
70     else:
71         file_nets = str(options.arfcn) + ";" + str(freq(options.arfcn)) + "M" +
            ";" + re.sub(r".*\[[^\]]*\]\ Cell identity.*",r"\1",nets[0]) + "
            "; + re.sub(r".*: [^\]* ([^\])* *Mobile Country Code .*$",r"\1",
            nets[1]) + ";" + re.sub(r".*Mobile Country Code \((.*)\)$",r"\1",
            nets[1]) + ";" + re.sub(r".*: [^\]* ([^\])* *Mobile Network Code
            .*$",r"\1",nets[2]) + ";" + re.sub(r".*Mobile Network Code \((.*)
            \)$",r"\1",nets[2]) + ";" + re.sub(r".*\[[^\]]*\]\ Local Area
            Code.*",r"\1",nets[3])
72
73     # collect IMSIs
74     for i in range(0, len(imsis)):
75         file_imsis = file_imsis + str(imsis[i])
76         if i < len(imsis) - 1:
77             file_imsis = file_imsis + ";"
78
79     # collect TMSIs
80     for i in range(0, len(tmsis)):
81         file_tmsis = file_tmsis + str(tmsis[i])
82         if i < len(tmsis) - 1:
83             file_tmsis = file_tmsis + ";"
84
85     # write nets to file
86     file = open("netlist.nets", "a")
87     file.write(file_nets + "\n")
88     file.close()
89     file = open("netlist.single.nets", "w")
90     file.write(file_nets + "\n")
91     file.close()
92
93     # write IMSIs to file
94     file = open("netlist.imsis", "a")
95     file.write(file_imsis + "\n")
96     file.close()
97     file = open("netlist.single.imsis", "w")
98     file.write(file_imsis + "\n")
99     file.close()
100
101     # write TMSIs to file
102     file = open("netlist.tmsis", "a")
103     file.write(file_tmsis + "\n")
104     file.close()
105     file = open("netlist.single.tmsis", "w")
106     file.write(file_tmsis + "\n")
107     file.close()
108
109     if __name__ == '__main__':
110         main()

```

D.4 summary.py

```
1  #!/usr/bin/env python
2  #
3  # This script produces a fancy summary output, it's maybe sometimes a little bit dirty, but
4  # the output looks OK.
5  # The script looks for the width of your shell and modifies the output, so it fits the screen
6  #
7  # The getTerminalSize() function is from http://stackoverflow.com/questions/566746/how-to-get-console-window-width-in-python
8  #
9  import sys
10 import re
11 from optparse import OptionParser
12
13 def getTerminalSize():
14     def ioctl_GWINSZ(fd):
15         try:
16             import fcntl, termios, struct, os
17             cr = struct.unpack('hh', fcntl.ioctl(fd, termios.TIOCGWINSZ, '1234'))
18         except:
19             return None
20         return cr
21     cr = ioctl_GWINSZ(0) or ioctl_GWINSZ(1) or ioctl_GWINSZ(2)
22     if not cr:
23         try:
24             fd = os.open(os.ctermid(), os.O_RDONLY)
25             cr = ioctl_GWINSZ(fd)
26             os.close(fd)
27         except:
28             pass
29     if not cr:
30         try:
31             cr = (env['LINES'], env['COLUMNS'])
32         except:
33             cr = (25, 80)
34     return int(cr[1]), int(cr[0])
35
36 def single_true():
37     return
38
39 def parse_options():
40     parser = OptionParser()
41     parser.add_option("-s", "--single", action="store_true", dest="single", default=False
42                                     ,
43                                     help="Read from a single ARFCN file")
44
45     (options, args) = parser.parse_args()
46     if (len(args) != 0):
47         parser.print_help()
48         sys.exit(1)
49
50     return options
51
52 def main():
53     # get parameter
54     options = parse_options()
55
56     arfcns = []
57
58     # is the input a single ARFCN or the complete list?
59     if options.single:
60         # read netlist from file
61         try:
62             file = open("netlist.single.nets", "r")
63         except IOError:
64             print("ERROR: the netlist.single.nets file could not be found")
65             sys.exit(1)
66         liste = file.read()
67         file.close()
```

```

68     liste = liste.split("\n")
69     liste.pop(len(liste) - 1)
70     for i in range(0, len(liste)):
71         liste[i] = liste[i].split(";")
72
73     # read IMSISs from file
74     try:
75         file = open("netlist.single.imsis", "r")
76     except IOError:
77         print("ERROR: the netlist.single.imsis file could not be found")
78         sys.exit(1)
79     imsis = file.read()
80     file.close()
81     imsis = imsis.split("\n")
82     imsis.pop(len(imsis) - 1)
83     for i in range(0, len(imsis)):
84         imsis[i] = imsis[i].split(";")
85
86     # read TMSISs from file
87     try:
88         file = open("netlist.single.tmsis", "r")
89     except IOError:
90         print("ERROR: the netlist.single.tmsis file could not be found")
91         sys.exit(1)
92     tmsis = file.read()
93     file.close()
94     tmsis = tmsis.split("\n")
95     tmsis.pop(len(tmsis) - 1)
96     for i in range(0, len(tmsis)):
97         tmsis[i] = tmsis[i].split(";")
98
99 else:
100     # read netlist from file
101     try:
102         file = open("netlist.nets", "r")
103     except IOError:
104         print("ERROR: the netlist.nets file could not be found")
105         sys.exit(1)
106     liste = file.read()
107     file.close()
108     liste = liste.split("\n")
109     liste.pop(len(liste) - 1)
110     for i in range(0, len(liste)):
111         liste[i] = liste[i].split(";")
112
113     # read IMSISs from file
114     try:
115         file = open("netlist.imsis", "r")
116     except IOError:
117         print("ERROR: the netlist.imsis file could not be found")
118         sys.exit(1)
119     imsis = file.read()
120     file.close()
121     imsis = imsis.split("\n")
122     imsis.pop(len(imsis) - 1)
123     for i in range(0, len(imsis)):
124         imsis[i] = imsis[i].split(";")
125
126     # read TMSISs from file
127     try:
128         file = open("netlist.tmsis", "r")
129     except IOError:
130         print("ERROR: the netlist.tmsis file could not be found")
131         sys.exit(1)
132     tmsis = file.read()
133     file.close()
134     tmsis = tmsis.split("\n")
135     tmsis.pop(len(tmsis) - 1)
136     for i in range(0, len(tmsis)):
137         tmsis[i] = tmsis[i].split(";")
138
139 # set widths of the rows
140 width0 = max([len(row[0]) for row in liste])
141 width1 = max([len(row[1]) for row in liste])
142 width2 = max([len(row[2]) for row in liste])

```

```

142 width3 = max([len(row[3]) for row in liste])
143 width4 = max([len(row[4]) for row in liste])
144 width5 = max([len(row[5]) for row in liste])
145 width6 = max([len(row[6]) for row in liste])
146 width7 = max([len(row[7]) for row in liste])
147
148 if width0 < 5:
149     width_0 = 5
150 else:
151     width_0 = width0
152 if width1 < 4:
153     width_1 = 4
154 else:
155     width_1 = width1
156 if width2 < 13:
157     width_3 = 13
158 else:
159     width_3 = width2
160 if (width3 + width4 + 3) < 19:
161     width_5 = 19
162 else:
163     width_5 = (width3 + width4 + 3)
164 if (width5 + width6 + 3) < 19:
165     width_4 = 19
166 else:
167     width_4 = (width5 + width6 + 3)
168 if width7 < 15:
169     width_2 = 15
170 else:
171     width_2 = width7
172
173 # network overview
174 sys.stdout.write("\n")
175 print "networks:"
176 sys.stdout.write("\n")
177 print " ARFCN" + " " * (width_0 - 5) + " | FREQ" + " " * (width_1 - 4) + " | Local
    Area Code" + " " * (width_2 - 15) + " | Cell identity" + " " * (width_3 - 13) +
    " | Mobile Network Code" + " " * (width_4 - 19) + " | Mobile Country Code"
178 print "-" * (width_0 + 2) + "+" + "-" * (width_1 + 2) + "+" + "-" * (width_2 + 2) +
    "+" + "-" * (width_3 + 2) + "+" + "-" * (width_4 + 2) + "+" + "-" * (width_5 + 2)
179 for row in liste:
180     print " " + " " * (3 - len(row[0])) + row[0] + " " * (width_0 - len(row[0]) -
        (3 - len(row[0]))) + " | " + row[1] + " " * (width_1 - len(row[1])) + " |
        " + row[7] + " " * (width_2 - len(row[7])) + " | " + row[2] + " " * (
            width_3 - len(row[2])) + " | " + row[5] + " (" + row[6] + ")" + " " * (
                width_4 - len(row[5]) - len(row[6]) - 3) + " | " + row[3] + " (" + row[4]
                + ")"
181         arfcns.append(row[0])
182
183 # create fancy output for found IMSIs ...
184 sys.stdout.write("\n")
185 sys.stdout.write(" " * (getTerminalSize()[0])); sys.stdout.write("\n")
186 sys.stdout.write("\n")
187 print("IMSI:")
188 sys.stdout.write("\n")
189 arfcn_c = 0
190 for row in imsis:
191     if not row == []:
192         sys.stdout.write("ARFCN " + " " * (3 - len(arfcns[arfcn_c])) + arfcns[
            arfcn_c] + ": ")
193         width_c = (getTerminalSize()[0]-11)/16
194         for i in range(0, len(row)):
195             sys.stdout.write(row[i])
196             if not i == len(row) - 1:
197                 sys.stdout.write(" ")
198             if (i + 1) % width_c == 0:
199                 # problem with complete full lines
200                 if not (getTerminalSize()[0] - 11) % 16 == 0:
201                     # last item?
202                     if not i == len(row) - 1:
203                         sys.stdout.write("\n")
204                 if not i == len(row) - 1:
205                     sys.stdout.write(" " * 11)
206                 if not i == len(row) - 1 or not (getTerminalSize()[0] - 11) % 16 == 0:

```

```

207         sys.stdout.write("\n")
208         arfcn_c = arfcn_c + 1
209
210     # ... and of course TIMSIs
211     sys.stdout.write("\n")
212     sys.stdout.write("-" * (getTerminalSize()[0])); sys.stdout.write("\n")
213     sys.stdout.write("\n")
214     print("TMSIs:")
215     sys.stdout.write("\n")
216     arfcn_c = 0
217     for row in tmsis:
218         if not row == ['']:
219             sys.stdout.write("ARFCN " + " " * (3 - len(arfcns[arfcn_c])) + arfcns[
220                 arfcn_c] + ": ")
221             width_c = (getTerminalSize()[0]-11)/9
222             for i in range(0, len(row)):
223                 sys.stdout.write(row[i])
224                 if not i == len(row) - 1:
225                     sys.stdout.write(" ")
226                 if (i + 1) % width_c == 0:
227                     # problem with complete full lines
228                     if not (getTerminalSize()[0] - 11) % 9 == 0:
229                         # last item?
230                         if not i == len(row) - 1:
231                             sys.stdout.write("\n")
232                     if not i == len(row) - 1:
233                         sys.stdout.write(" " * 11)
234             if not i == len(row) - 1 or not (getTerminalSize()[0] - 11) % 9 == 0:
235                 sys.stdout.write("\n")
236             arfcn_c = arfcn_c + 1
237
238     sys.stdout.write("\n")
239
240 if __name__ == '__main__':
241     main()

```

D.5 arfcn.py

```
1  #!/usr/bin/env python
2  #
3  # This script calculates the frequency to a given ARFCN
4  #
5
6  import sys
7  from optparse import OptionParser
8
9  def parse_options():
10     parser = OptionParser()
11     parser.add_option("-a", "--arfcn", type="int", dest="arfcn",
12                       help="Input is an ARFCN")
13
14     (options, args) = parser.parse_args()
15     if (len(args) != 0):
16         parser.print_help()
17         sys.exit(1)
18
19     return options
20
21 def freq(arfcn):
22
23     # GSM 900
24     if arfcn > 0 and arfcn <= 124:
25         freq = 890 + 0.2*arfcn + 45
26
27     # GSM 1800
28     elif arfcn >= 512 and arfcn <= 885:
29         freq = 1710.2 + 0.2*(arfcn - 512) + 95
30
31     else:
32         freq = 0
33
34     return freq
35
36 def main():
37
38     # get parameter
39     options = parse_options()
40
41     # check options
42     # wrong usage
43     if not (options.arfcn):
44         print "ERROR: no ARFCN is set, it seems i've nothing to do"
45         sys.exit(1)
46
47     # correct usage
48     if (options.arfcn):
49         print str(freq(options.arfcn))+ 'M'
50
51 #####
52 if __name__ == '__main__':
53     main()
```


D.6 gsmlive.sh

```
1  #!/bin/bash
2  #
3  # This script captures an specified ARFCN. It creates a live dump to the gsm tun device and
4  # a pcap file for later analysis.
5  #
6  ARFCN=""
7  DECIM=112
8  BOARD="B"
9  WIRESHARK=0
10
11 # check for parameters and change config values
12 while getopts ":c:d:R:sh" optname; do
13     case "$optname" in
14         "c")
15             ARFCN=$OPTARG
16             ;;
17         "d")
18             DECIM=$OPTARG
19             ;;
20         "R")
21             BOARD=$OPTARG
22             ;;
23         "s")
24             WIRESHARK=1
25             ;;
26         "h")
27             echo "This script captures an specified ARFCN. It creates a live dump
28                 to the gsm tun device and a pcap file for later analysis."
29             echo "SYNOPSIS"
30             echo " gsmlive.sh [ -c ARFCN ] [ -d DECIM ] [ -R BOARD ] [ -s ] [ -h ]"
31             echo "OPTIONS"
32             echo " -c ARFCN"
33             echo " Sets the ARFCN."
34             echo " -d DECIM"
35             echo " Sets the DECIM value."
36             echo " -R BOARD"
37             echo " Specifies which BOARD (A or B) of the USRP to use."
38             echo " -s"
39             echo " Don't start Wireshark. So there will be an output to the shell,
40                 the pcap file and the gsm tun device. You can capture the gsm
41                 device with any other tool of course."
42             echo " -h"
43             echo " Prints this help."
44             exit 0
45         ;;
46         "?")
47             echo "Unknown option $OPTARG"
48             $0 -h
49             exit 1
50         ;;
51         ":.")
52             echo "No argument value for option $OPTARG"
53             echo " -h"
54             $0 -h
55             exit 1
56         ;;
57         *)
58             echo "Unknown error while processing options"
59             echo " -h"
60             $0 -h
61             exit 1
62         ;;
63     esac
64 done
65
66 # check if ARFCN is set?
67 if [ "$ARFCN" == "" ]; then
```

```

68     echo "ERROR: no ARFCN set."
69     echo
70     $0 -h
71     exit 1
72 fi
73
74 # start wireshark?
75 if [ $WIRESHARK -eq 0 ]; then
76     if [[ $EUID -ne 0 ]]; then
77         echo "ERROR: you need to be root to capture with Wireshark."
78         exit 1
79     else
80         gsmshark -i gsm -k &
81     fi
82 fi
83
84 gsm_scan.py -R $BOARD -S I -p d -d $DECIM -c $ARFCN -r e | gsmdecode -i
85
86 exit 0

```

D.7 capture.sh

```
1 #!/bin/bash
2 #
3 # This script captures GSM traffic and stores it in a cfile. By default it iterates all
4 # ARFCNs.
5
6 # predefine config values
7 DURATION=4
8 DECIM=112
9 GAIN=52
10 ARFCN=""
11 BOARD="B"
12 DST='pwd'"/"date +%Y/%m/%d/%H/%M/%S'
13
14 # check for parameters and change config values
15 while getopts ":t:d:g:c:R:d:h" optname; do
16     case "$optname" in
17         "t")
18             DURATION=$OPTARG
19             ;;
20         "d")
21             DECIM=$OPTARG
22             ;;
23         "g")
24             GAIN=$OPTARG
25             ;;
26         "c")
27             ARFCN=$OPTARG
28             ;;
29         "R")
30             BOARD=$OPTARG
31             ;;
32         "d")
33             DST=$OPTARG
34             ;;
35         "h")
36             echo "This script captures GSM traffic and stores it in a cfile. By
37                 default it iterates all ARFCNs."
38             echo "SYNOPSIS"
39             echo " capture.sh [ -t SECONDS ] [ -d DECIM ] [ -g GAIN ] [ -c ARFCN ]
40                 [ -R BOARD ] [ -d DSTDIR ] [ -h ]"
41             echo "OPTIONS"
42             echo " -t SECONDS"
43             echo " Duration in SECONDS to stay on each ARFCN."
44             echo " -d DECIM"
45             echo " Sets the DECIM value."
46             echo " -g GAIN"
47             echo " Sets the GAIN value."
48             echo " -c ARFCN"
49             echo " Scan only on ARFCN, don't iterate all."
50             echo " -R BOARD"
51             echo " Specifies which BOARD (A or B) of the USRP to use."
52             echo " -d DSTDIR"
53             echo " Sets the DSTDIR."
54             echo " -h"
55             echo " Prints this help."
56             exit 0
57         ;;
58         "?")
59             echo "Unknown option $OPTARG"
60             echo
61             $0 -h
62             exit 1
63         ;;
64         ":")
65             echo "No argument value for option $OPTARG"
66             echo
67             $0 -h
68             exit 1
69     esac
70 done
```

```

69         ;;
70         *)
71             echo "Unknown error while processing options"
72             echo
73             $0 -h
74             exit 1
75     ;;
76 done
77
78 # check or create the destination
79 if ! [ -d $DST ]; then
80     mkdir -p $DST
81 fi
82
83 # Look for usrp_rx_cfile.py
84 USRP_PROG=usrp_rx_cfile.py
85 while :; do
86     which "$USRP_PROG" > /dev/null
87     if [ $? -eq 0 ]; then
88         break
89     fi
90     USRP_PROG=/usr/share/gnuradio/usrp/usrp_rx_cfile.py
91     which "$USRP_PROG" > /dev/null
92     if [ $? -eq 0 ]; then
93         break
94     fi
95
96     echo "ERROR: usrp_rx_cfile.py not found. Make sure it's in your PATH!"
97     exit 1
98 done
99
100 # calculate samples for the specified duration
101 SAMPLES='expr 64000000 / $DECIM '*' $DURATION'
102
103 # no single ARFCN specified
104 if [ "$ARFCN" == "" ]; then
105     # Now iterate all, about 1000 RFCNS, each for -t seconds
106     # GSM 900 (ARFCN 1 - 124) && GSM 1800 (ARFCN 512 885)
107     for ARFCN in `seq 1 124 && seq 512 885`; do
108         FREQ=$(arfcn.py -a $ARFCN)
109         FILE="$DST/arfcn_${ARFCN}_${FREQ}.cfile"
110         echo "capturing for $DURATION seconds ($SAMPLES samples) on module $BOARD on
111             ARFCN $ARFCN (${FREQ}Hz) to $FILE"
112         $USRP_PROG -R $BOARD -g $GAIN -d "$DECIM" -f "$FREQ" -N $SAMPLES $FILE > /dev/
113             null 2> /dev/null
114         if [ $? -gt 0 ]; then
115             echo "WARNING: something went wrong while capturing (ARFCN $ARFCN),
116                 continuing on the next ARFCN anyway ..."
117         fi
118     done
119
120 # capture only a single ARFCN
121 else
122     FREQ=$(arfcn.py -a $ARFCN)
123     FILE="$DST/arfcn_${ARFCN}_${FREQ}.cfile"
124     echo "capturing for $DURATION seconds ($SAMPLES samples) on module $BOARD on ARFCN
125         $ARFCN (${FREQ}Hz) to $FILE"
126     $USRP_PROG -R $BOARD -g $GAIN -d "$DECIM" -f "$FREQ" -N $SAMPLES $FILE > /dev/null 2>
127         /dev/null
128     if [ $? -gt 0 ]; then
129         echo "WARNING: something went wrong while capturing (ARFCN $ARFCN) ..."
130     fi
131 fi
132
133 exit 0

```

D.8 analysecf.sh

```
1  #!/bin/bash
2  #
3  # This script creates a summary and pcap file. It reads from a captured cfile.
4  #
5
6  # predefine config values
7  TMP=/tmp/gsmdump
8  DECIM=112
9
10 # check for parameters and change config values
11 while getopts ":s:d:h" optname; do
12     case "$optname" in
13         "s")
14             TMP=$OPTARG
15             ;;
16         "d")
17             DECIM=$OPTARG
18             ;;
19         "h")
20             echo "This script creates a summary and pcap file. It reads from a
                captured cfile."
21             echo
22             echo "SYNOPSIS"
23             echo " analysecf.sh [ -s TMPDIR ] [ -d DECIM ] [ -h ] CFILE"
24             echo
25             echo "OPTIONS"
26             echo " -s TMPDIR"
27             echo " Sets the TMPDIR."
28             echo " -d DECIM"
29             echo " Sets the DECIM value."
30             echo " -h"
31             echo " Prints this help."
32             exit 0
33         ;;
34         "?")
35             echo "Unknown option $OPTARG"
36             echo
37             $0 -h
38             exit 1
39         ;;
40         ":")
41             echo "No argument value for option $OPTARG"
42             echo
43             $0 -h
44             exit 1
45         ;;
46         *)
47             echo "Unknown error while processing options"
48             echo
49             $0 -h
50             exit 1
51         ;;
52     esac
53 done
54
55 # check for CFILE
56 for p in "${@:OPTIND}"; do
57     if [ "$CFILE" == "" ]; then
58         CFILE="$p"
59     else
60         echo "ERROR: Too much files specified."
61         echo
62         $0 -h
63         exit 1
64     fi
65 done
66
67 # no CFILE was set
68 if [ "$CFILE" == "" ]; then
69     echo "ERROR: No file specified."
70     echo
```

```

71     $0 -h
72     exit 1
73 fi
74
75 # CFILE is relative
76 if ! [ 'echo -n $CFILE | grep -e "~/.*" ]; then
77     CFILE='pwd'/"$CFILE
78 fi
79
80 # check or create the tmp
81 if ! [ -d $TMP ]; then
82     mkdir -p $TMP
83 fi
84 cd $TMP
85
86 # rm old netlists
87 if [ -f netlist.nets ]; then
88     rm netlist.nets
89 fi
90 if [ -f netlist.imsis ]; then
91     rm netlist.imsis
92 fi
93 if [ -f netlist.tmsis ]; then
94     rm netlist.tmsis
95 fi
96
97 FILE='echo -n $CFILE | sed -e "s/(.*)\.[^.]*$/1/"
98 gsm_scan_light.py -SN -pd -d "$DECIM" -I "$CFILE" > /dev/null 2> /dev/null
99 if [ -e tvoid.pcap ] && [ -e tvoid-burst.pcap ]; then
100     if [ 'stat -c %s tvoid.pcap' -gt 24 ]; then
101         mv tvoid.pcap $FILE.pcap
102         mv tvoid-burst.pcap $FILE-burst.pcap
103         pcap4gsmdecode.py $FILE.pcap | gsmdecode -i | analyse.py
104         # check if there are any networks
105         if [ -f netlist.single.nets ]; then
106             summary.py -s | tee $FILE.txt
107             rm netlist.single.nets
108         else
109             echo
110             echo "I'm sorry, but no network was found"
111             echo
112         fi
113     fi
114 fi
115
116 # rm old netlists
117 if [ -f netlist.nets ]; then
118     rm netlist.nets
119 fi
120 if [ -f netlist.imsis ]; then
121     rm netlist.imsis
122 fi
123 if [ -f netlist.tmsis ]; then
124     rm netlist.tmsis
125 fi
126 if [ -f netlist.single.nets ]; then
127     rm netlist.single.nets
128 fi
129 if [ -f netlist.single.imsis ]; then
130     rm netlist.single.imsis
131 fi
132 if [ -f netlist.single.tmsis ]; then
133     rm netlist.single.tmsis
134 fi
135 if [ -f gsm-receiver.pcap ]; then
136     rm gsm-receiver.pcap
137 fi
138 if [ -f gsm-receiver-burst.pcap ]; then
139     rm gsm-receiver-burst.pcap
140 fi
141 if [ -f speech.gsm ]; then
142     rm speech.gsm
143 fi
144 fi

```

```
145 if [ -f cfile2.out ]; then  
146     rm cfile2.out  
147 fi  
148  
149 exit 0
```

Literaturverzeichnis

- [1] *airprobe Projekt-Homepage*. <https://svn.berlin.ccc.de/projects/airprobe/>.
- [2] *GNU Radio Projekt-Homepage*. <http://gnuradio.org/redmine/wiki/gnuradio>.
- [3] *Wireshark Projekt-Homepage*. <http://wiki.wireshark.org/Development>.
- [4] *Mail 419: Re: [airprobe-main] Listening to the uplink with Airprobe gsm*. airprobe Mailingliste, 12 2009. <https://svn.berlin.ccc.de/projects/airprobe/wiki/MailingLists>.
- [5] *Mail 626: Re: [airprobe-main] Capture uplink GSM packets*. airprobe Mailingliste, 05 2010. <https://svn.berlin.ccc.de/projects/airprobe/wiki/MailingLists>.
- [6] ETSI: *GSM 04.08 version 7.7.1 Release 1998*.
- [7] GÖLLER, JOACHIM: *Die GSM-DM-Kanäle im Dialog. Einstieg in die Mobilfunk-Signalisierung*. Epv, 3 2003.
- [8] GÖLLER, JOACHIM: *Signaling in Mobile Radio Communication: According the the GSM standard*. Epv, 1 2006.
- [9] MAUCHER, JOHANNES: *Skript zur Vorlesung Mobile Communications Systems*.