

Projektarbeit im SoSe2017

**Konzeption und Aufbau eines Handheld zum
Auslesen des BCCH umliegender GSM
Basisstationen**

von

Dennis Dette [Mat.Nr.: 45311]
Christian Kobiela [Matr.Nr.: 47571]



18.10.2017

Inhaltsverzeichnis

Abkürzungsverzeichnis	iii
Abbildungsverzeichnis	iv
1 Einleitung	1
2 Installationsanleitung	2
2.1 Installation von benötigter Software für gr-gsm	2
2.1.1 Installieren von Kalibrate	2
2.1.2 Zugriff auf USB-Device freischalten	2
2.1.3 Kalibrierung des RTL-SDR Gerätes	3
2.1.4 Installation von GNU Radio	5
2.1.5 Installieren von libosmocore	5
2.2 Installation von gr-gsm	6
2.3 Reinstallation	6
3 Hardwareaufbau des Handhelds	7
3.1 Druck des Cases	7
3.2 Benötigten Teile	7
3.2.1 Hardware	8

3.2.2	Werkzeuge und Ergänzendes	8
3.3	Zusammenbau	8
3.3.1	Verkabelung	9
3.4	Stromversorgung	9
4	Bedienung des Handhelds	11
4.1	Stromversorgung	12
4.2	Bedienung über die Desktopoberfläche	12
4.2.1	Den Scanner starten	13
4.3	Erweitern und Verändern	13
4.3.1	Startup	13
4.3.2	Autostartup	14
5	Anleitung zur GUI unter PyQt4	15
5.1	Allgemeiner Aufbau der GUI	15
5.2	Anleitung zum Anpassen des textuellen Inhalts	18
6	Setup des Raspberry PI	21
6.1	Das How To und interdependencies	21
7	Zusammenfassung	23

Abkürzungsverzeichnis

GSM	Global System for Mobile Communications
BTS	Base transceiver station
BCCH	Broadcast control channel
SDR	Software Defined Radio

Abbildungsverzeichnis

3.1	Zusammenbau der Caseteile [?]	8
3.2	Einbau der Elektronik [?]	9
3.3	Einsetzen des Akkus [?]	9
3.4	Verkabelung im Inneren des Case [?]	9
3.5	Pinbelegung des Raspberry [?]	9
4.1	Beschreibung der Anschlüsse	11
4.2	Einstellung der automatischen Helligkeitsregelung	12
4.3	Ansicht des Dekstops	12
5.1	Standard Fenster GUI	15
5.2	Abgeschlossener Scan	16
5.3	Info des BCCH	16
5.4	Mouseover Tooltip	16
5.5	GUI Ansicht angeklickt	17
5.6	Infos der Childnodes	17

Einleitung

Die Aufgabe bestand darin ein Handheld zu konzipieren mit welchem es möglich ist die umgebenden GSM-Base transceiver station (BTS) zu scannen. Durch das Auslesen der Informationen des Informationsstring des Broadcast control channel (BCCH) können die gefundenen Basisstationen beschrieben und charakterisiert werden.

Ein solches System bestand bereits, allerdings lief dieses unter Ubunutu und war somit an ein Notebook gebunden. Die Idee war es das Ganze mobiler zu gestalten und auf den neuesten Stand zu bringen.

Die Verwendung eines DVB-T-Sticks sowie der Software Gnuradio waren Grundlage dieses Software Defined Radio (SDR) Projektes.

Dieser Ausarbeitung liegt eine DVD mit allen Datenblättern, einem Datenträgerabbild, dem git Repository und allen sonstigen Daten die zu diesem Projekt gehören um Notfalls alles reproduzieren zu können.

Installationsanleitung

Ausführlichere Anleitung und weitere Wiki-Einträge sind hier nachzulesen:
[https://github.com/ptrkrysik/gr-gsm/wiki/
Installation-on-RaspberryPi-3](https://github.com/ptrkrysik/gr-gsm/wiki/Installation-on-RaspberryPi-3)

Installation von benötigter Software für gr-gsm

Installieren von Kalibrate

Als erstes installieren wir Kalibrate:

```
1 sudo apt-get install libtool autoconf automake libfftw3-dev
  librtlsdr0 librtlsdr-dev libusb-1.0-0 libusb-1.0-0-dev
2 git clone https://github.com/asdil12/kalibrate-rtl.git
3 cd kalibrate-rtl
4 git checkout arm_memory
5 ./bootstrap
6 ./configure
7 make
8 sudo make install
```

Zugriff auf USB-Device freischalten

Das RTL-SDR Gerät einstecken und ID mit dem Befehl `lsusb` überprüfen.
Zu sehen sollte etwas wie folgend sein:

2.1. INSTALLATION VON BENÖTIGTER SOFTWARE FÜR GR-GSM3

```
1 Bus 001 Device 004: ID **0bda:2832** Realtek Semiconductor  
  Corp. RTL2832U DVB-T  
2 Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.  
  SMSC9512/9514 Fast Ethernet Adapter  
3 Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.  
4 Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root  
  hub
```

In unserem Fall ist die ID **0bda:2832**. Anschließend öffnen wir eine rules-Datei:

```
1 sudo nano /etc/udev/rules.d/20.rtlsdr.rules
```

...in welcher dann folgende Zeile hinzugefügt werden muss:

```
1 SUBSYSTEM=="usb", ATTRS{idVendor}=="0bda", ATTRS{idProduct}==  
  "2832", GROUP="adm", MODE="0666", SYMLINK+="rtl_sdr"
```

Falls mehrere RTL-SDR Geräte verwendet werden, können mehrere Zeilen hinzugefügt werden. Die ID muss jeweils natürlich entsprechend des Gerätes abgewandelt werden.

Danach sollte der Raspberry Pi neugestartet werden: `sudo reboot`

Kalibrierung des RTL-SDR Gerätes

Jetzt können wir den Befehl ausführen um das RTL-SDR Gerät zu kalibrieren (um genau zu sein um den durchschnittlichen absoluten Fehler in ppm zu berechnen):

```
1 kal -s GSM900
```

Das Ergebnis sollte ähnlich zu diesem sein:

```

1 Found 1 device(s):
2   0: Generic RTL2832U
3
4 Using device 0: Generic RTL2832U
5 Found Rafael Micro R820T tuner
6 Exact sample rate is: 270833.002142 Hz
7 kal: Scanning for GSM-900 base stations.
8 GSM-900:
9     chan: 1 (935.2MHz - 33.430kHz) power: 55085.23
10    chan: 3 (935.6MHz - 34.130kHz) power: 63242.36
11    chan: 5 (936.0MHz - 33.970kHz) power: 41270.82
12 ...
13 ...
14     chan: 112 (957.4MHz - 32.934kHz) power:
15           498930.07
16     chan: 116 (958.2MHz - 31.859kHz) power:
17           88039.44
18     chan: 124 (959.8MHz - 32.429kHz) power:
19           247404.23

```

Das stärkste Signal wäre in diesem Fall Kanal 112. Also führen wir die Kalibrierung auf diesem Kanal durch:

```
1 kal -c 112
```

...und erhalten ein Ergebnis wie folgt:

```

1 Found 1 device(s):
2   0: Generic RTL2832U
3
4 Using device 0: Generic RTL2832U
5 Found Rafael Micro R820T tuner
6 Exact sample rate is: 270833.002142 Hz
7 kal: Calculating clock frequency offset.
8 Using GSM-900 channel 112 (957.4MHz)
9 average          [min, max]      (range, stddev)
10 - 34.368kHz          [-34376, -34357]      (20,
11           4.697051)
12 overruns: 0
13 not found: 0
14 average absolute error: 35.897 ppm

```

Unser durchschnittlicher absoluter Fehler wäre hier also 36 ppm (35.897 ppm).

2.1. INSTALLATION VON BENÖTIGTER SOFTWARE FÜR GR-GSM5

Installation von GNU Radio

Als nächstes installieren wir GNU Radio:

```
1 sudo apt-get install gnuradio gnuradio-dev
```

Installieren von libosmocore

Libosmocore muss kompiliert werden...

```
1 sudo apt-get install cmake
2 sudo apt-get install build-essential libtool shtool autoconf
   automake git-core pkg-config make gcc
3 sudo apt-get install libpcsclite-dev libtalloc-dev
4 git clone git://git.osmocom.org/libosmocore.git
5 cd libosmocore/
6 autoreconf -i
7 ./configure
8 make
9 sudo make install
10 sudo ldconfig -i
11 cd
```

...außerdem benötigen wir noch ein paar andere Dinge.

```
1 sudo apt-get install gr-osmosdr rtl-sdr
2 sudo apt-get install libboost-dev
3 sudo apt-get install osmo-sdr libosmosdr-dev
4 sudo apt-get install libusb-1.0.0 libusb-dev
5 sudo apt-get install libboost-all-dev libcppunit-dev swig
   doxygen liblog4cpp5-dev python-scipys
```

Installation von gr-gsm

Und nun zum letzten Schritt:

```

1 git clone https://github.com/ptrkrysik/gr-gsm.git
2 cd gr-gsm
3 mkdir build
4 cd build
5 cmake ..
6 make
7 sudo make install
8 sudo ldconfig

```

Zuletzt erstellen wir noch die `./gnuradio/config.conf` config-Datei, mit `nano ./gnuradio/config.conf`. Und fügen diese zwei Zeilen hinzu (damit GNU Radio die custom Blöcke von gr-gsm finden kann):

```

1 [grc]
2 local_blocks_path=/usr/local/share/gnuradio/grc/blocks

```

Reinstallation

Sollte an irgend einem Punkt etwas dermaßen kaputt gegangen sein, dass eine Rettung nicht mehr möglich ist kann die auf der DVD hinterlegte ISO Datei auf die MikroSD Karte des Raspberry geflasht werden um den Auslieferungszustand wieder zu erlangen. Die ISO muss erstmal auf einen PC kopiert und anschließend entpackt werden.

```

1 sudo dd if=~/Path_to_Img.img of=/dev/mmcblk0

```

Als OutputFile muss die MikroSD ausgewählt werden. welche vorher mit

```

1 lsblk

```

lokalisiert wurde.

Hardwareaufbau des Handhelds

Im folgenden wird erklärt wie die verbauten Komponenten zusammen zu fügen sind, sollte der Bedarf bestehen das Handheld erneut auf zu bauen, etwas zu modifizieren oder sich ein defekt einschleichen. Die Anleitung für das Case stammt von Adafruit.com /citeAdafruit und wird als PDF Datei der DVD beiliegen.

Druck des Cases

Die CAD Dateien zum Druck des Cases sind auf der DVD hinterlegt und können dafür verwendet werden das Case erneut drucken zu lassen beziehungsweise falls notwendig zu modifizieren.

Das empfohlene Material ist PLA wobei ABS, Nylon, copperfill, bamboo fill oder PET ebenfalls verwendet werden können. Da bei Herr Strohrmann Hi-Wis nur ABS auf Lager war, wurde dieses verwendet.

Über den Sommer haben wir die Erfahrung gemacht, dass sich das Material stark verzieht wenn es Wärme ausgesetzt wird. Sollte dies wieder vorkommen, kann ein vorsichtig verwendet Föhn Abhilfe verschaffen um das Case wieder in Form zu bringen.

Benötigten Teile

Die Teile die man benötigt um das Handheld aufzubauen sind folgende:

Hardware

- Pi Foundation PiTFT - 7" Touchscreen Display
- Raspberry Pi 3
- 200mm Flex Displaykabel
- Adafruit PowerBoost 1000C
- 2500mAh LiPo Akku
- SPDT Schalter
- 16GB Micro SD Karte (r: 95MB/s, w: 60MB/s)

Werkzeuge und Ergänzendes

Zudem braucht man noch gewisses Werkzeug:

- 3D Drucker
- Filament
- Kreuzschlitzschraubenzieher
- Lot
- Litzen mit 1,5 mm²
- Kabelbinder
- M3 x .5 x 6M Schrauben x12

Zusammenbau

Der Zusammenbau des Tablets ist ziemlich selbsterklärend.

Nachdem die gedruckten Teile wie in Abbildung 3.1 zusammen gebaut wurden müssen nur noch die einzelnen Komponenten an Ihren Platz geschraubt werden, wie in Abbildung 3.2 zu sehen ist.

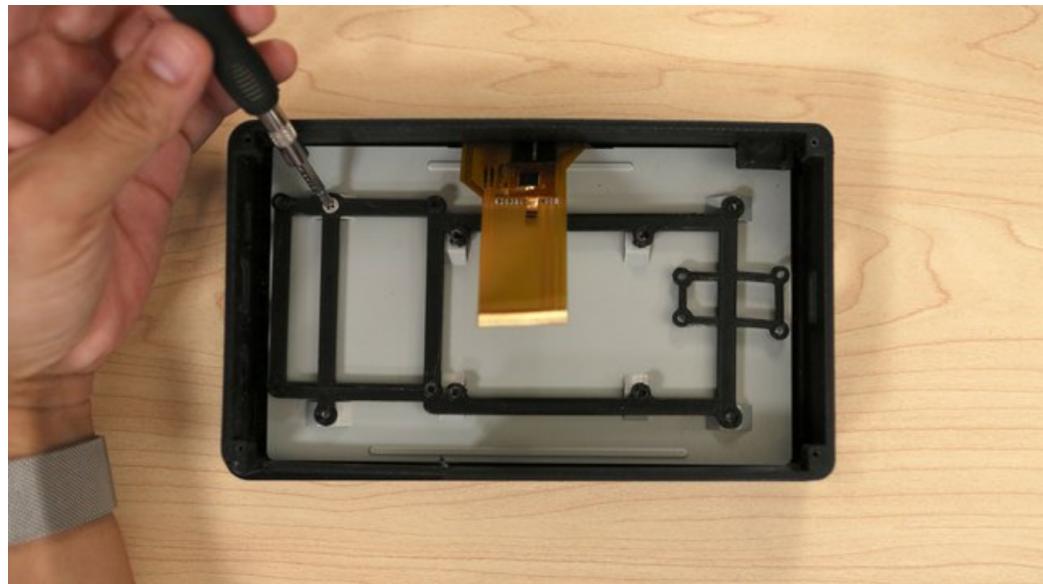


Abbildung 3.1: Zusammenbau der Caseteile [?]



Abbildung 3.2: Einbau der Elektronik [?]



Abbildung 3.3: Einsetzen des Akkus [?]

Anschließend noch den Akku mit einem Kabelbinder an seinem Rahmen befestigen und dann über dem Displaytreiber festschrauben: Abbildung 3.3

Zu guter Letzt noch das Flachbandkabel des Displays in den Raspberry stecken, hierfür zuerst die graue Lasche an beide Seiten nach oben ziehen, Kabel bis zum Anschlag einlegen und Lasche gleichmäßig wieder eindrücken. Deckel zu und fertig.

Verkabelung

EN und GND des Adafruit PowerBoost1000C werden an den Schalter herausgeleitet. GND wird hierbei mit dem mittleren Pin verbunden.

Der Akku wird über den JST Stecker an die PowerBoost1000C angeschlossen.

Der positive Ausgang des PowerBoost1000C wird mit dem GPIO # 2 und der Negative mit dem GPIO # 6 verbunden.

Der 5V Pin des Displaytreibers wird mit den GPIO # 4 und GND an GPIO # 9 des Raspberrys verbunden.

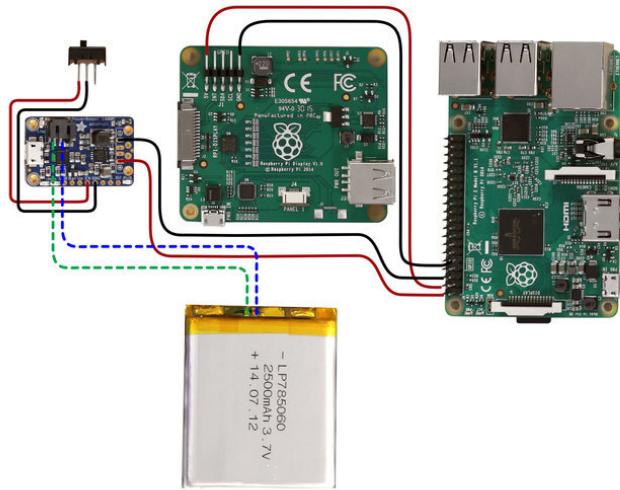


Abbildung 3.4: Verkabelung im Inneren des Case [?]

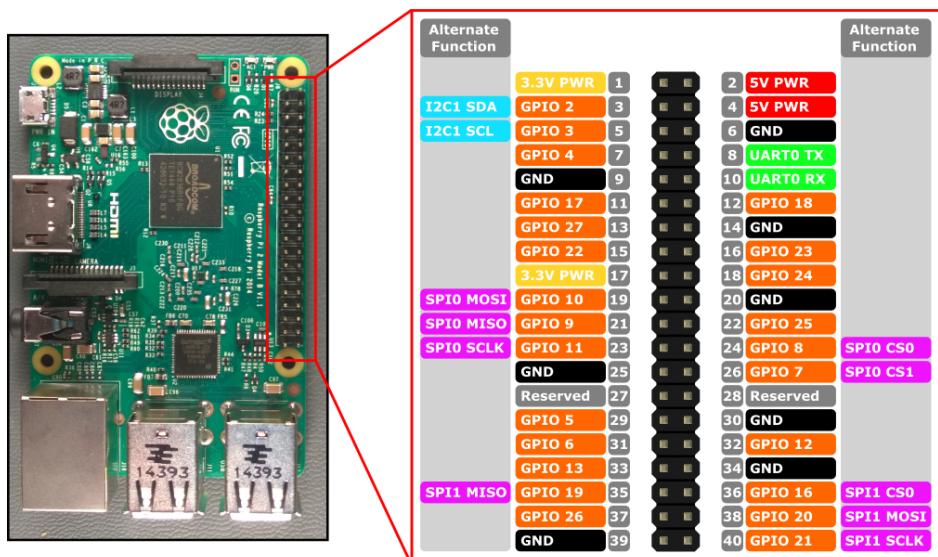


Abbildung 3.5: Pinbelegung des Raspberry [?]

Stromversorgung

Herzstück der Stromversorgung ist ein Adafruit PowerBoost1000C. Hierbei handelt es sich um eine Elektronik wie sie in vielen Powerbanks zu finden ist. Das heißt einerseits hat man einen MikroUSB Eingang über den der LiPo Akku geladen werden kann andererseits gibt es einen USB Ausgang an dem die 3,7V des Akkus auf 5V hochgeregt, ausgegeben werden. Für das Laden des Akkus ist der MCP73871 von Microchip verantwortlich, der DC/DC Boostconverter ist von Texas Instruments: TPS6109. Der USB Connector wurde nicht verbaut und die 5V werden über verlöteten Litzen direkt auf den Raspberry Pi geleitet.

Üblicherweise können 1A oder Spitzenwerte bis zu 2,5A können aus dem Akku gezogen werden. Die Maximale Stromaufnahme des Raspberry beträgt 2,5A.

Bedienung des Handhelds

In diesem Kapitel finden Sie eine Bedienungsanleitung für das Handheld und Tipps für den Umgang mit diesem.

Um einen gemeinsamen Betrachtungspunkt zu haben legen wir nun fest, dass man das Tablet im Querformat verwendet und "oben" die Seite beschreibt an der die USB Anschlüsse des Raspberrys zu sehen sind (vgl. Abbildung 4.1). In diesem Fall finden wir die Ladebuchse rechts über die das Tablet sowohl geladen als auch stationär verwendet werden kann. HDMI, AUX und einen zweiten MikroUSB Anschluss zur Direktversorgung beziehungsweise Überbrückung des Akkus, links. Der ON/OFF Schalter befindet sich auf der Unterseite rechts. Dieser ist leider wegen der Drucktoleranzen des 3D Druckers nicht all zu gut zugänglich. Deshalb empfiehlt es sich den Schalter mit einem spitzen Gegenstand zu betätigen (bspw. bietet sich hier die Antenne des DVB-T Sticks an).

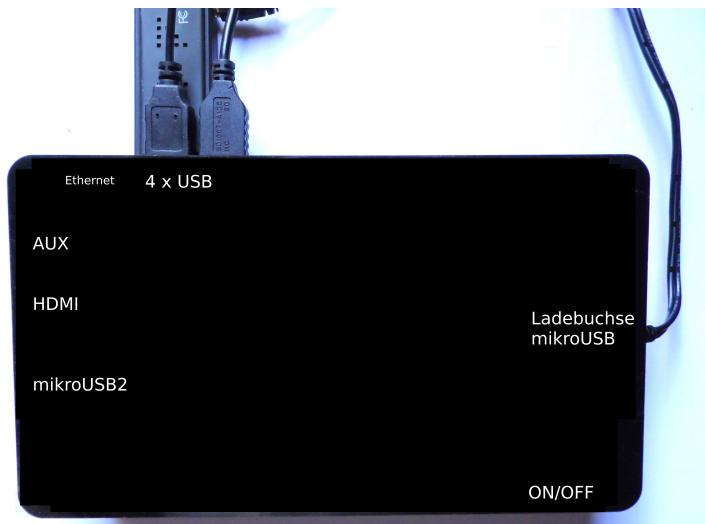


Abbildung 4.1: Beschreibung der Anschlüsse

Stromversorgung

Verbaut ist ein 2,5 Ah LiPo Akku welcher über eine Adafruit Power-boost1000c Ladeelektronik geladen und betrieben wird. Die Elektronik ist sowohl dafür zuständig den Akku aufzuladen als auch bei der stationären Verwendung ein angeschlossenes USB Netzteil als Stromquelle zu Nutzen. Da der Normalstrom der aus dem Akku gezogen wird sich um die 1A bewegt kann es durchaus zu einer Unterversorgung kommen. Aufgrund dessen wird es nicht empfohlen bei höchster Displayhelligkeit den Global System for Mobile Communications (GSM) Suchlauf durchzuführen. Die Displaybeleuchtung dunkelt sich nach 10 Sekunden ab um diesen Fall der möglichen Unterversorgung auszuschließen. Dies kann in den Einstellungen wie in Abbildung ?? geändert werden.

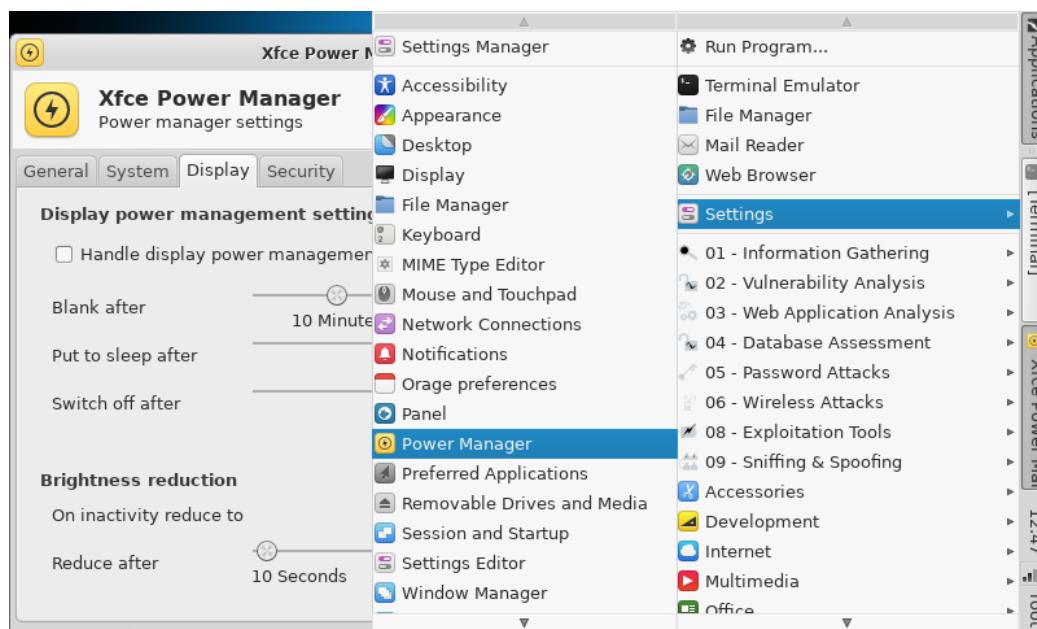


Abbildung 4.2: Einstellung der automatischen Helligkeitsregelung

Bedienung über die Desktopoberfläche

Auf dem Desktop befinden sich die wichtigsten Shortcuts für den Gebrauch des GSM Scanners.

Das Touchscreen wurde so eingestellt, dass man nur einmal klicken muss um

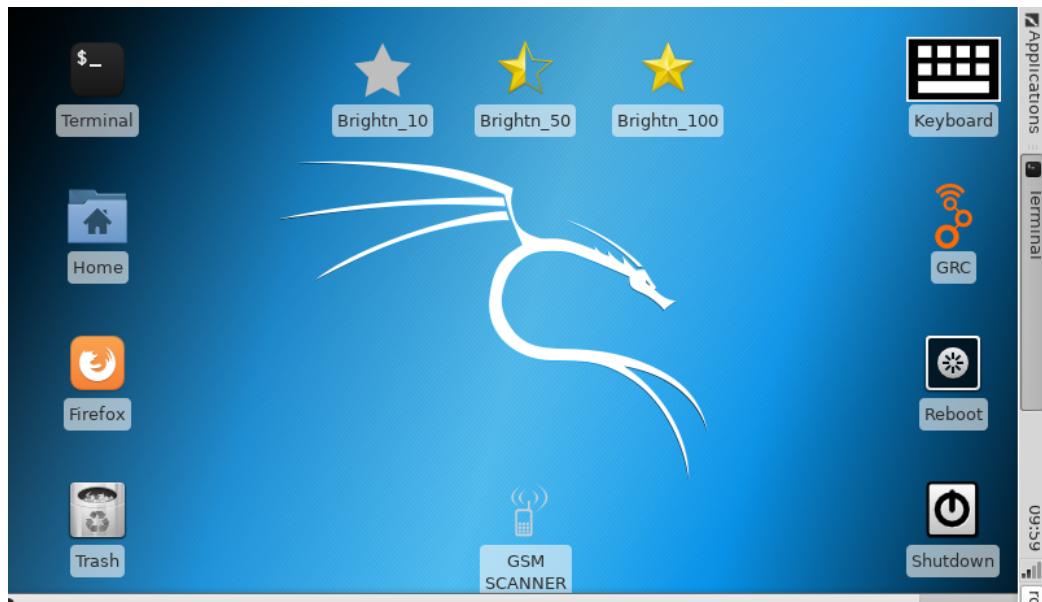


Abbildung 4.3: Ansicht des Dekstop

Programme auszuführen was eine Bedienung mit dem Finger erleichtert. Um eine einfache Einstellbarkeit der Displayhelligkeit zu realisieren haben wir auf dem Desktop Shortcuts hierfür implementiert. Die Stufen 10%, 50% und 100% können ausgewählt werden. Sind andere Stufen gewünscht, so kann man die Displayhelligkeit durch ausführen des Befehls

```
echo XXX > /sys/class/backlight/rpi_backlight/brightness
```

in einer Konsole ändern. XXX kann im Bereich von 0 (0%) bis 255 (100%) gewählt werden.

Ferner findet sich ein virtuelles Keyboard auf dem Desktop falls man mobil etwas schreiben möchte. Reboot und Shutdown Shortcuts sind ebenso zu finden. Bitte beachten Sie: Nach dem Shutdown muss die Stromversorgung zusätzlich am ON/OFF Schalter getrennt werden da weiterhin Spannung am Display anliegt.

Den Scanner starten

Der GSM Scanner hat ebenfalls ein Desktop Shortcut welches mit einem Klick die Suche nach GSM Basisstationen ermöglicht. Um den Hintergrund zu

verstehen werden im folgenden Kapitel die Einstellmöglichkeiten, mit denen ein Scan gestartet werden kann, erläutert.

Erweitern und Verändern

Startup

```

1 Options:
2   -h, --help                  show this help message and exit
3   -b BAND, --band=BAND      Specify the GSM band for the
4                                frequency. Available
5                                bands are: GSM900, DCS1800, GSM850,
6                                PCS1900, GSM450,
7                                GSM480, GSM-R
8   -s SAMP_RATE, --samp-rate=SAMP_RATE
9                                Set sample rate [default=2000000.0] -
10                               allowed values
11                               even_number*0.2e6
12   -p PPM, --ppm=PPM          Set frequency correction in ppm [
13                               default=0]
14   -g GAIN, --gain=GAIN      Set gain [default=24.0]
15   --args=ARGS                Set device arguments [default=]
16   --speed=SPEED              Scan speed [default=4]. Value range
17                               0-5.
18   -v, --verbose              If set, verbose information output is
19                               printed: ccch
20
21
22 gsm_scanner -g 50 -p 29

```

Mit diesen Übergabeparametern wird der GSM Scanner aufgerufen und sucht im default BAND GSM900 das Netz von 925 Mhz bis 960 Mhz ab. Dies entspricht dem E-GSM 900 Netz. Ein Gain von 50 dB ist der Maximalwert und ermöglicht somit die größte "Ausbeute" was die Ergebnisliste angeht. Der Offset des Quarzes bei Betriebstemperatur wurde in Kapitel 2.1.3 durch eine Kalibrierung des DVB-T Sticks berechnet und ist immer mit anzugeben.

Das zugehörige Desktop Shortcut, wie auch die Shortcuts zur Helligkeitsregulierung, führen Shell Skripte aus, welche unter

```
1 /root/GrGsm-Gui
```

hinterlegt sind. Hier kann man eingreifen falls es gewünscht ist etwas zu ändern. Zum Beispiel wenn ein neuer Stick mit einem anderen Quarz Offset verwendet werden soll. Am besten öffnet man die Skripte über das Terminal mit "nano" da sonst kein Textverarbeitungsprogramm installiert ist.

Alle weiteren von uns geschriebenen Komponenten sind ebenfalls im Ordner

```
1 /root/GrGsm-Gui
```

zu finden. Da es sich hierbei um ein Git Repository handelt kann dieses auch über

```
1 cd GrGsm-Gui/
2 git pull
```

auf den neusten Stand gebracht werden, sollten Veränderungen vorgenommen werden.

Autostartup

Nach dem Anschalten am ON/OFF Schalter auf der Unterseite des Handhelds fährt dieses hoch ohne, dass eine Anmeldung erforderlich ist. Sollte dies geändert werden, so müssen zwei Befehle in der Datei lightdm.conf auskommentiert werden.

```
1 cd /etc/lightdm
2 nano lightdm.conf
3
4 ****
5 autologin-user=root
6 autologin-user-timeout=0
7 ****
8
9 ^ muessen durch voransetzen eines "#" auskommentiert werden
```

Anleitung zur GUI unter PyQt4

Im folgenden Kapitel soll die GUI und deren Inhalt anhand einiger Bildbeispiele verdeutlicht werden. Nachfolgend wird erläutert wie und wo im Code es möglich ist, diese Inhalte anzupassen bzw. zu verändern.

Zuerst jedoch noch ein paar Erklärungen zu dem Aufbau unseres modifizierten grgsm-scanners.

Zu Beginn des Programm, wird in der Main-Funktion unsere GUI aufgerufen und aufgebaut, während diese nun im Hintergrund auf einem Thread läuft, wird auf einem anderen Thread das Scanning fortgeführt. In den folgenden Abschnitten wird nun näher erläutert wie genau die GUI diese Informationen darstellt und den Scan-Vorgang kommuniziert.

Allgemeiner Aufbau der GUI

Starten des grgsm-scanner initiiert die GUI wie zu sehen in Abbildung:5.1. Das GUI-Fenster ist in zwei Felder aufgeteilt, ein kleineres Feld auf der linken Seite und ein größeres auf der Rechten. Im linken Feld werden die Kanäle dargestellt, welche nach und nach gescannt und hinzugefügt werden. Im rechten Feld werden zu Beginn Informationen zur Projektarbeit ausgegeben. Diese wird durch die Informationen von ausgewählten Kanälen überschrieben dazu jedoch später mehr.

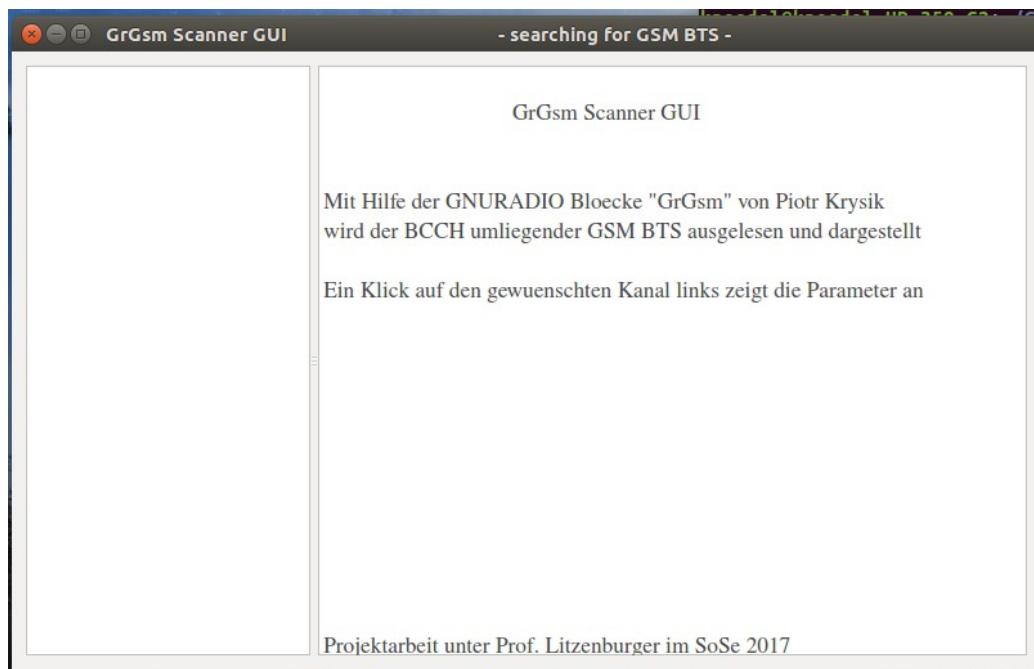


Abbildung 5.1: Standard Fenster GUI

Solange die Suche nach Kanälen noch nicht abgeschlossen wurde, wird oben im Fensterrahmen **-searching for GSM BTS-** dargestellt (siehe Abbildung: 5.1). Falls der Scan abgeschlossen ist, wird dies durch ein **-DONE!**– angezeigt, ersichtlich aus Bild 5.2. Außerdem sollten nun alle gefundenen Kanäle im linken Feld zu sehen sein.

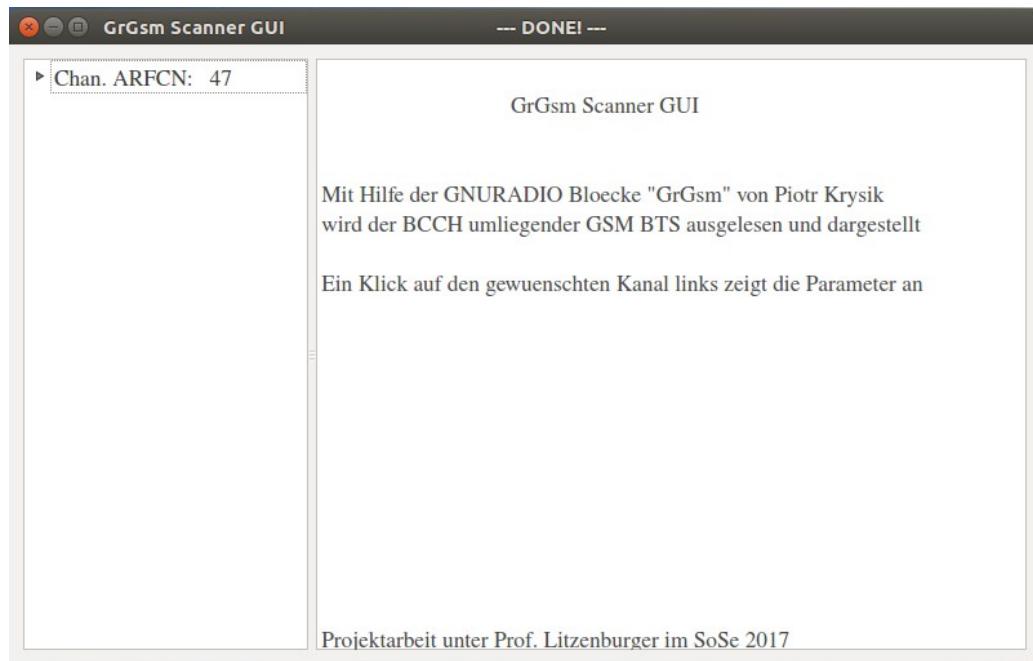


Abbildung 5.2: Abgeschlossener Scan

Falls Kanäle schon gescannt wurden und somit links auftauchen, können diese ausgewählt werden um Informationen der einzelnen Parameter einzusehen. Unter den Parametern befinden sich der ARFCN, Frequenz, LAC, MCC, MNC, CID und Power. Zusätzlich werden ein paar weitere Informationen ausgegeben (siehe Abbildung:5.3). Sollte eine Maus angeschlossen sein können die Informationen auch durch eine Art Tooltip über mouse-hovering auf dem jeweiligen Kanal ausgegeben werden (siehe Abbildung:5.4).

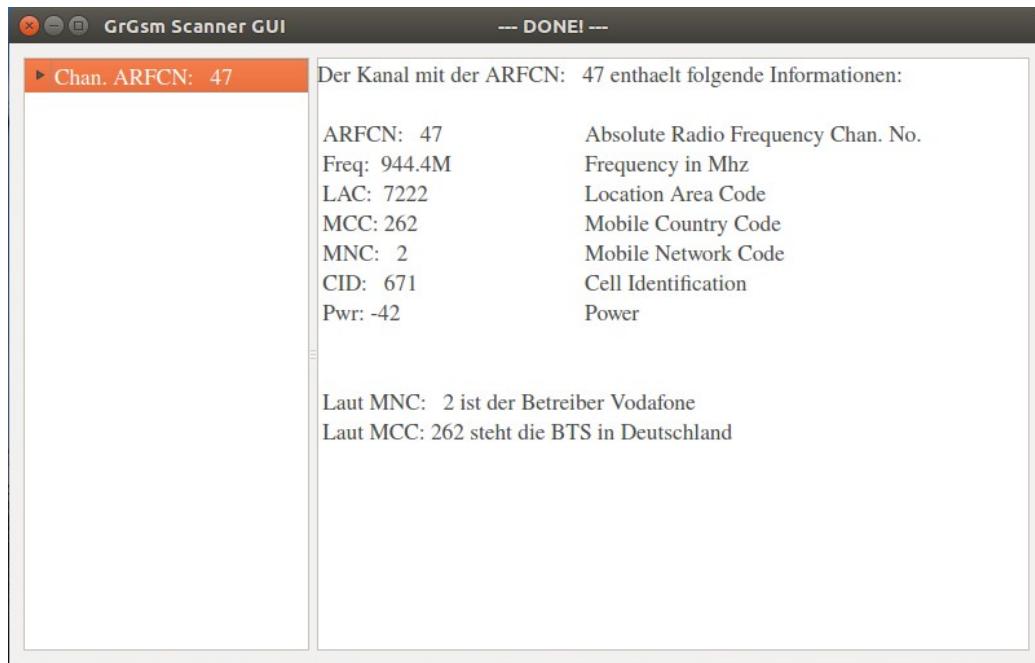


Abbildung 5.3: Info des BCCH

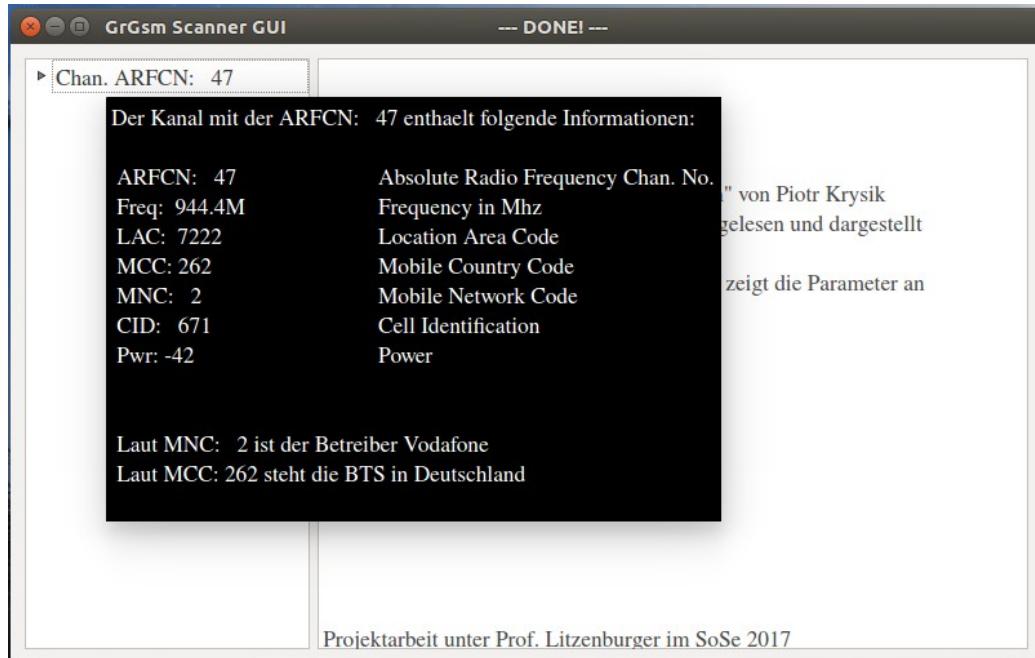


Abbildung 5.4: Mouseover Tooltip

Bei jedem Kanal ist es möglich durch einen kleinen Pfeil links daneben, diesen aufzurollen um die Parameter einzeln zu betrachten (siehe Abbildung:5.5). Durch anklicken wieder auf die einzelnen Parameter wird die jeweilige Information dargestellt (siehe Abbildung:5.6).

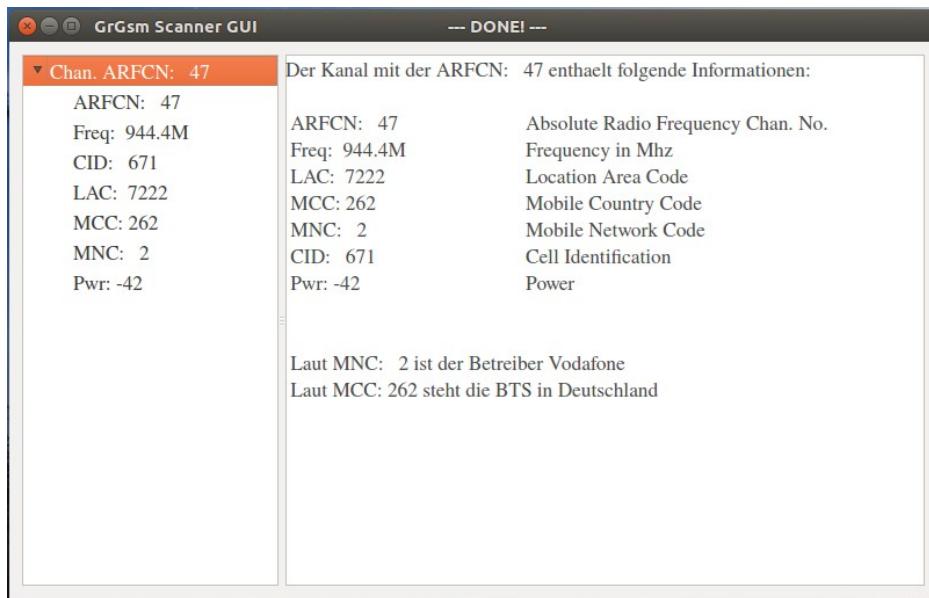


Abbildung 5.5: GUI Ansicht angeklickt

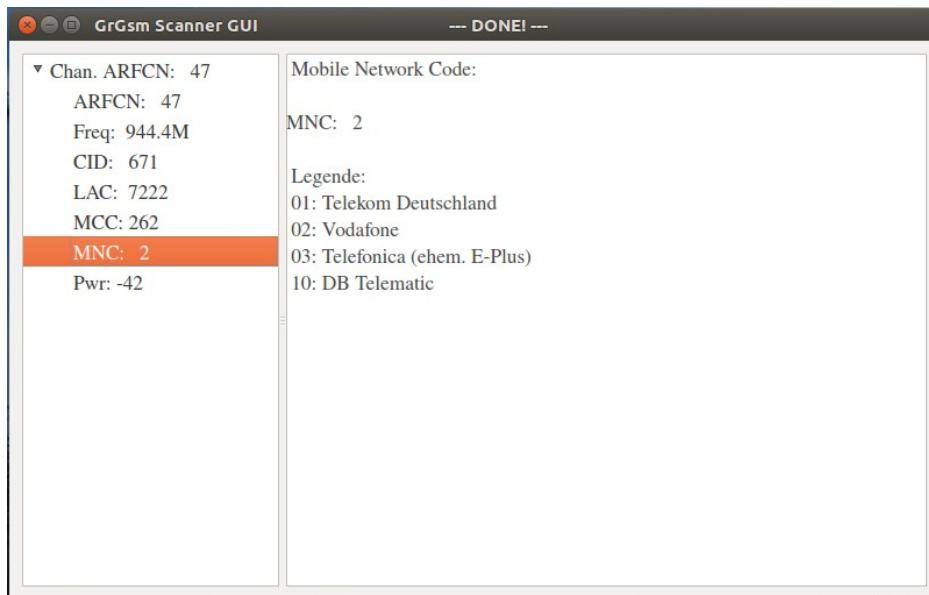


Abbildung 5.6: Infos der Childnodes

Anleitung zum Anpassen des textuellen Inhalts

Anpassung der Titelseite

Die Titelseite, wie im vorherigen Kapitel bereits erwähnt, welche zu sehen ist am Programmstart, kann durch Änderungen im folgenden Programmcode (Listing: 5.1) angepasst werden. Die Zeilenangaben links verdeutlichen zusätzlich, wo man dies im Programmcode finden kann.

```

100 self.text = QtGui.QLabel("\n"
101                 " \t\t GrGsm Scanner GUI\n\n"
102                 " Mit Hilfe der GNURADIO Bloecke \""
103                 " GrGsm\" von Piotr Krysik\n"
104                 " wird der BCCH umliegender GSM BTS"
105                 " ausgelesen und dargestellt\n\n"
106                 " Ein Klick auf den gewuenschten"
107                 " Kanal links zeigt die Parameter
108                 " an\n\n\n\n\n\n\n\n\n\n\n\n\n\n"
109
110                 " Projektarbeit unter Prof."
111                 " Litzenburger im SoSe 2017\n"
112                 " von Dennis Dette und Christian"
113                 " Kobiela\n\n"
114                 , self.right)

```

Listing 5.1: Titelseite

Anpassung der Parentnode

Die Informationen, welche angezeigt werden wenn man einen gefundenen Kanal anklickt, können in den folgenden Codezeilen geändert werden (Listing: 5.2). Bisher werden wie zu sehen ist die einzelnen Parameter mit Werten dargestellt und Angaben über Land und Betreiber der BTS.

```

187 parent_item.setToolTip(column,"Der Kanal mit der " +
188     Buffer_Channel.ARFCN +
189         " enthaelt folgende Informationen: \n\n"
190         + " " +Buffer_Channel.ARFCN + " \t\tAbsolute
191             Radio Frequency Chan. No.\n"
192         + " " +Buffer_Channel.FREQ + " \t\tFrequency
193             in Mhz\n"
194         + " " +Buffer_Channel.LAC + " \t\tLocation
195             Area Code\n"
196         + " " +Buffer_Channel.MCC + " \t\tMobile
197             Country Code\n"
198         + " " +Buffer_Channel.MNC + " \t\tMobile
199             Network Code\n"
200         + " " +Buffer_Channel.CID + " \t\tCell
201             Identification\n"
202         + " " +Buffer_Channel.PWR+ " \t\tPower\n\n\n"
203         + " Laut " + Buffer_Channel.MNC + " ist der
204             Betreiber " + Betreiber
205         + "\n Laut " + Buffer_Channel.MCC + " steht
206             die BTS in " + str(Land)
207     )

```

Listing 5.2: Kanalinformation

Anpassung der Childnodes

Nun zum letzten Teil, dessen Text angepasst werden kann, den Parameterinformationen. Diese sind einzusehen, wenn ein Kanal aufklappt ist. Hier werden durch Auswählen der einzelnen Parameter die Informationen dargestellt, welche im Code 5.3 zu sehen sind. Die sieben Parameterinformationen werden in den verschiedenen If-Fällen definiert.

```

204 if k==0:
205     item.setToolTip(column, " Absolute Radio Frequency
206         Channel Number:\n\n " + Buffer_Channel.ARFCN
207         + "\n\n In GSM cellular networks, an absolute radio-
208             frequency channel number\n ""(ARFCN) is a code
209                 that specifies a pair of physical radio carriers
210                     used for\n ""transmission and reception in a land
211                         mobile radio system, one for the uplink\n ""signal
212                             and one for the downlink signal.\n\n "
213 "ARFCN for GSM 900 \n "
214 "ARFCN = (Chan. Freq - 45 Mhz- 890 Mhz)/200\n\n "
215 "ARFCN for E GSM 900\n "
216 "ARFCN = 1024 + (Chan. Freq - 45 Mhz - 890 Mhz)/200\n
217         ")
218 elif k==1:
219     item.setToolTip(column, " Frequency:\n\n " +
220         Buffer_Channel.FREQ + "\n\n GSM Downlink Frequency
221         ")
222 elif k==2:
223     item.setToolTip(column, " Cell Identification:\n\n "
224         + Buffer_Channel.CID +
225         "\n\n A GSM Cell ID (CID) is a generally unique
226             number used\n "
227             "to identify each base transceiver station (BTS) or\n
228                 "
229             "sector of a BTS within a location area code (LAC) \n
230                 "
231             "if not within a GSM network.")
232 elif k==3:
233     item.setToolTip(column, " Location Area Code:\n\n " +
234         Buffer_Channel.LAC+
235         "\n\n A location area is a set of base stations that
236             are\n "
237             "grouped together to optimise signalling.\n "
238             "To each location area, a unique number called a
239                 location area code is assigned.\n ")
240 elif k==4:
241     item.setToolTip(column, " Mobile Country Code:\n\n "
242         + Buffer_Channel.MCC +
243         "\n\n The mobile country code consists of 3 decimal
244             digits and the mobile\n "
245             "network code consists of 2 or 3 decimal digits. The
246                 first \"2\" in 262\n "
247             "stands for Europe, 62 for Germany")
248 elif k==5:
249     item.setToolTip(column, " Mobile Network Code:\n\n " +
250         Buffer_Channel.MNC +
251         "\n\n Legende:\n 01: Telekom Deutschland\n 02:
252             Vodafone\n 03: Telefonica (ehem. E-Plus)\n 10: DB
253                 Telematic")
254 elif k==6:
255     item.setToolTip(column, " Power:\n\n " +
256         Buffer_Channel.PWR +
257         "\n\n GSM needs at least a Power of -102dBm " )

```

Listing 5.3: Parameterinformationen

Setup des Raspberry PI

Im folgenden wird beschrieben wie wir den Raspberry aufgesetzt haben und welche Probleme uns dabei begegnet sind.

Das How To und interdependencies

gr-gsm wird hauptsächlich von Piotr Krysik entwickelt und wird von diesem auch gepflegt. Es gibt auch einen vorgeschlagenen Weg wie man gr-gsm auf einem Raspberry Pi 3 zu installieren hat. Dieser Weg hat sich allerdings als Sackgasse erwiesen. Grund dafür war, dass in Zwischenzeit GnuRadio weiterentwickelt wurde und gr-gsm sich dieser Weiterentwicklung angepasst hatte. Die Weiterentwicklung betraf leider nicht Raspbian Jessie was zu folgenden Problemen führte.

gr-gsm benötigte damals eine Version von GnuRadio > 3.7.9. Für Raspbian stand allerdings nur die Version 3.7.6 zur Verfügung. Der Versuch auf instabile Testversionen 3.7.10+ auszuweichen erwies sich ebenso als aussichtslos. Das Installieren von diesen Versionen führte nur dazu, dass wiederum Bibliotheken von denen GnuRadio abhängig war inkompatibel wurden.

Nach einiger Recherche wie man ein funktionierendes Gesamtpaket erhalten könnte sind wir auf das Programm PyBombs gestoßen. Dieses macht genau das, was wir gesucht haben, es installiert ein Programm und alle Abhängigkeiten die dieses zur Nutzung benötigt. Im Gegensatz zu apt-get, wie man es von von Ubuntu und sonstigen Linux Distributionen kennt bezieht sich PyBOMBS nicht nur auf die Standard Repositorys und die dort hinterlegte Softwareversionen sondern installiert genau die Versionen die benötigt werden. Hierbei kann sich PyBOMBS auf eigens hinterlegte Repositorys, sogenannte Recipes beziehen von wo es die Software aus den Quelldateien kompiliert. So schön es klingt führte es zu den selben Problemen wie im ersten Versuch nur,

dass dieses mal andere Programme inkompatibel wurden.

An diesem Punkt haben wir beschlossen, dass eine zeitnahe Umsetzung mit gr-gsm wohl aussichtslos erscheint. Gleichzeitig standen wir mit Piotr Krysik in Kontakt um einen Weg zu finden wie es doch zu lösen sein könnte. Seither ist Herr Krysik dabei seine Anpassungen in einem Raspberry Emulator zu überprüfen. Wir entschieden uns dennoch Airprobe zu versuchen, da es nicht absehbar war wann wir mit einer erfolgversprechenden Antwort hätten rechnen können. Dafür haben wir uns an einer jahrealten Anleitung orientiert und sind dabei auf Kali Linux gestoßen, dieses war glücklicherweise auch für Raspberrys verfügbar. Nachdem wir das installiert hatten, war, wer hätte es gedacht GnuRadio mittlerweile zu Neu um mit Airprobe arbeiten zu können. Allerdings war die stabile Version des standard Repsotitory glücklicherweise 3.7.10. Somit war das installierte GnuRadio mit dem neusten gr-gsm kompatibel und so konnten wir alle Abhängigkeiten installieren, die Kalibrierung durchführen und gr-gsm installieren. Zu der Zeit haben wir schon einen Monat Arbeit in das Projekt stecken und die Grenzen unserer Frustrationstoleranz immer ein Stückchen weiter schieben müssen.

Im Endeffekt ist die Anleitung wie sie in Kapitel 2 dokumentiert wird, unter Kali Linux voll durchführbar. Es wäre allerdings wünschenswert den GSM Scanner unter Raspbian zum laufen zu bringen da es sich hierbei um ein besser auf den Raspberry zugeschnittenes Betriebssystem handelt. Damit diese Portierung auf Raspbian möglich ist, müssen die vorher beschriebenen Probleme mit den Abhängigkeiten gelöst werden.

Zusammenfassung

In dieser Projektarbeit ist es uns gelungen ein Handheld zum Auslesen des BCCH zu konzipieren, diesen aufzubauen und in Betrieb zu nehmen. Als Herzstück dient uns ein von Kali Linux betriebener Raspberry Pi 3, das da-zugehörige Display mit Toucheingabe und ein DVB-T-Stick. Letzterer wurde nach dem Prinzip des Software Defined Radio verwendet, welches besagt, dass möglichst viel der Signalverarbeitung in der Software durchgeführt wird. So können die analogen Komponenten der Kommunikationssysteme einfach gehalten werden.

Mit den Gnuradio Blöcken "gr-gsm" von Piotr Krysik ist es uns gelungen den Informationsstring des BCCH auszulesen und mit einer eigens geschriebenen GUI darzustellen. Die Darstellung der GUI und der Suchvorgang der /acGSM Basisstationen laufen parallel auf zwei Threads um eine laufende Aktualisierung zu ermöglichen.

Mit dem fertigen Handheld ist eine mobile Nutzung von bis zu 5 Stunden möglich. So kann auch die Netzabdeckung der entferntesten Orte analysiert werden.