

Projektarbeit im SoSe2017

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

von
Dennis Dette und Christian Kobiela



!!Datum!!

Inhaltsverzeichnis

Abkürzungsverzeichnis	iii
Abbildungsverzeichnis	iv
Tabellenverzeichnis	v
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	5
3 Bedienung des Handhelds	7
3.1 Stromversorgung	8
3.2 Bedienung über die Desktopoberfläche	8
3.3 Erweitern und Verändern	9

4 Setup des Raspberry PI	12
4.1 Das How To und interdependencies	12
5 Einige Features	14
5.1 Ein Unterabschnitt	14
6 Zusammenfassung	16
7 Ausblick	17

Abkürzungsverzeichnis

UML	Unified Modelling Language
GSM	Global System for Mobile Communications
BTS	Base transceiver station
BCCH	Broadcast control channel

Abbildungsverzeichnis

3.1	Beschreibung der Anschlüsse	7
3.2	Einstellung der automatischen Helligkeitsregelung	8
3.3	Ansicht des Dekstops	9
5.1	Beispielbild	14
5.2	Inkskape Bild mit Text	15

Tabellenverzeichnis

5.1 Beispieldatenebene	14
----------------------------------	----

Einleitung

Die Aufgabe bestand darin ein Handheld zu konzipieren mit welchem es möglich ist die umgebenden Global System for Mobile Communications (GSM) Base transceiver station (BTS) zu scannen. Durch das Auslesen 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 ein wenig mobiler zu gestalten und auch auf den neuesten Stand zu bringen.

Die Verwendung eines DVB-T Sticks sowie der Software Gnuradio wurde vorgeschrieben

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).

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 libpcslite-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/potrkrysik/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
```

Bedienung des Handhelds

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

Um einen 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. In diesem Fall finden wir die Ladebuchse rechts über die das Tablet sowohl geladen als auch statio-när verwendet werden kann, HDMI, AUX und einen Mikro USB Anschluss 2 zur Direktversorgung, Überbrückung des Akkus, links. Der ON/OFF Schal-ter befindet sich auf der Unterseite rechts. Dieser ist leider nicht all zu gut zugänglich, weshalb es empfohlen wird den Schalter mit einem spitzen Ge-genstand zu betätigen (Bspw. bietet sich hier die Antenne an).

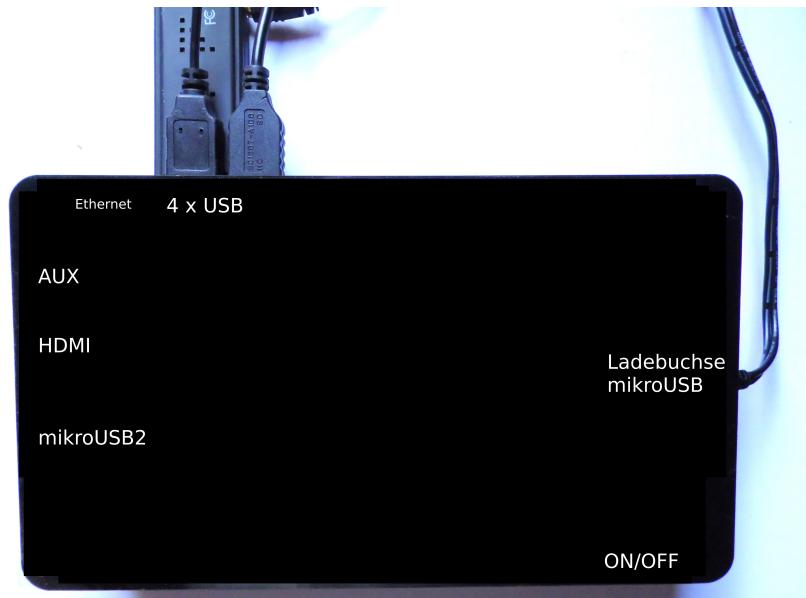


Abbildung 3.1: Beschreibung der Anschlüsse

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 angeschlüssenes 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 GSM Suchlauf durchzuführen. Die Displaybeleuchtung dunkelt sich nach 10 Sekunden ab um diesen Fall der möglichen Unterversorgung auszuschließen. Dies kann in den Einstellungen geändert werden.



Abbildung 3.2: Einstellung der automatischen Helligkeitsregelung

Bedienung über die Desktopoberfläche

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

Das Touchscreen ist so eingestellt, dass man nur einmal klicken muss um Programme auszuführen was eine Bedienung mit dem Finger erleichtern

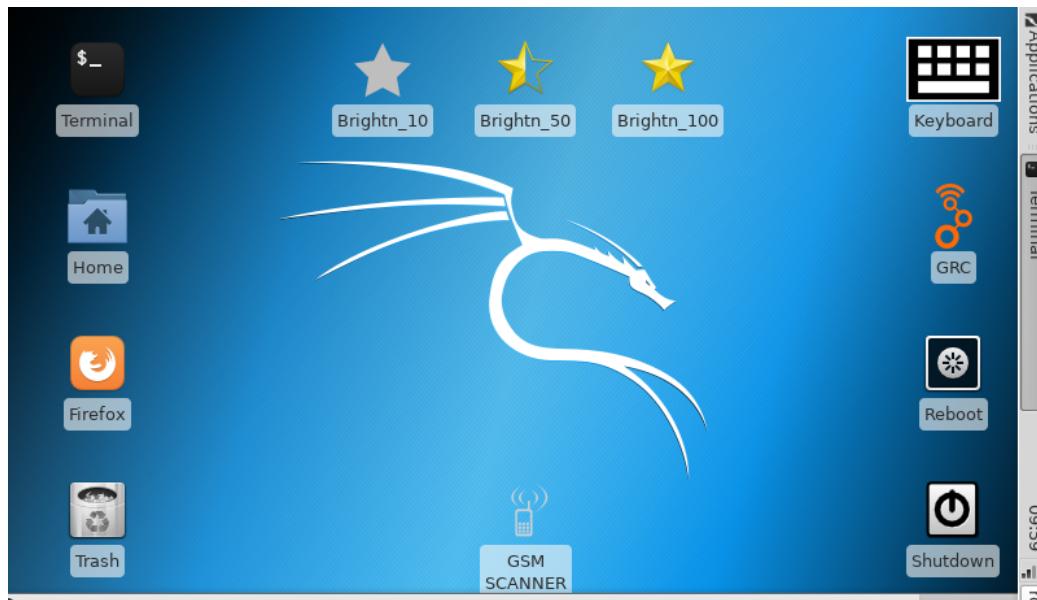


Abbildung 3.3: Ansicht des Dekstop

sollte. 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

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

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.

Der GSM Scanner hat ebenfalls ein Desktop Shortcut welches mit einem Klick die Suche nach GSM Basisstationen ermöglicht. Um den Hintergrund zu verstehen erläutere ich hier auf die möglichen Einstellmöglichkeiten mit denen ein Scan gestartet werden kann.

Erweitern und Verändern

Nach dem Anschalten am ON/OFF Schalter auf der Untersei-

te des Handhelds fährt dieses hoch ohne, dass eine Anmeldung erforderlich ist. Sollte dies geändert werden, so müssen 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

```

```

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

```

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 berechnet durch eine Kalibrierung des DVB-T Sticks und ist immer mit anzugeben.

Das zugehörige Desktop Shortcut, wie auch die Shortcuts zur

Helligkeitsregulierung, führen Shell Skripte aus welche unter

```
1 /root/Documents
```

hinterlegt sind. Hier kann man eingreifen falls etwas geändert werden soll. Am besten öffnet man diese über das Terminal mit "nano" da sonst kein Textverarbeitungsprogramm installiert ist.

Alle von uns geschriebenen Komponenten sind 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.

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

GrGsm wird hauptsächlich von Piotr Krysik entwickelt und wird von diesem auch gepflegt. Es gibt auch einen vorgeschlagenen Weg wie man GrGsm auf einem Raspberry Pi 3 zu installieren hat. Dieser Weg hat sich allerdings als Sackgasse erwiesen. Grund dafür war, dass in Zwischenzeit GnuRadio weiterentwickelt aber nicht für Raspbian Jessie und dadurch inkompatibel zu GrGsm wurde.

GrGsm benötigte damals eine Version von GnuRadio > 3.7.9. Für Raspbian war allerdings nur die Version 3.7.6 verfügbar. Der Versuch auf unstable Versionen 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, 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

!!!!!!HIER PYBOMBS ERKLÄREN!!!!!!

An diesem Punkt haben wir beschlossen, dass eine zeitnahe Umsetzung mit GrGsm wohl aussichtslos erscheint. Indes standen wir mit Piotr Krysik in Kontakt um einen Weg zu finden wie es doch zu lösen sein könnte. Zu dieser Zeit war dieser dabei seine Anpassungen in einem Raspberry Emulator

zur überprüfen. Wir entschieden uns doch Airprobe zu verwenden. 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 glücklicherweise 3.7.10, somit mit dem neusten GrGsm kompatibel und so konnten wir alle Abhängigkeiten installieren, die Kali-brierung durchführen und GrGsm 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.

Einige Features

Hier werden einige Features gezeigt.

Ein Unterabschnitt

Die Tabelle 5.1 ist ein Beispiel für eine wissenschaftliche Tabelle (ohne vertikale Trennlinien).

Tabelle 5.1: Beispieltabelle

Apfel	Birne	Katze
200 g	180 g	3.4 kg

So zitiert man [?]. Man kann auch mehrere Quellen auf einmal referieren [?, ?, ?, ?]

In Abbildung 5.1 ist eine Kurve dargestellt.

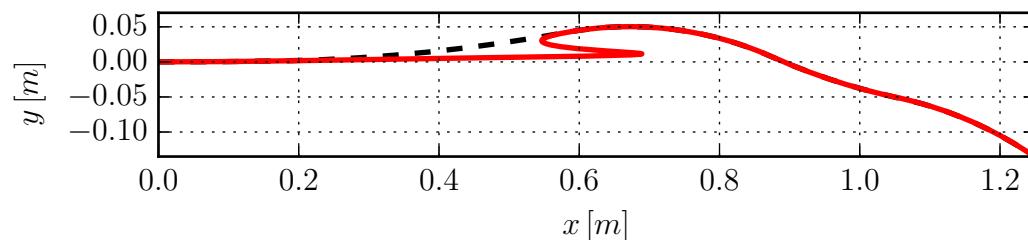


Abbildung 5.1: Beispielbild

Inkscape erlaubt es eps-Files und den Text getrennt zu exportieren, auf diese

Weise kann man die Schrift nachträglich anpassen. Ein Beispiel dazu ist in Abbildung 5.2 dargestellt.

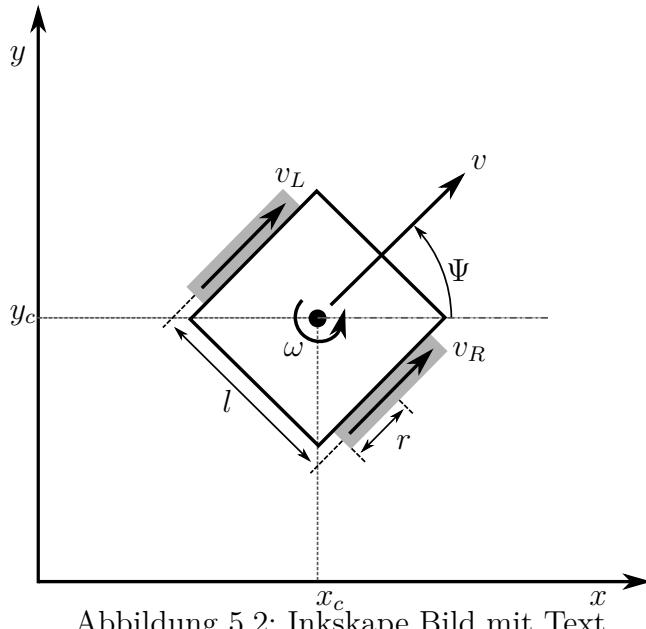


Abbildung 5.2: Inkskape Bild mit Text

Außerdem gibt es eine Umgebung zum Schreiben von Pseudo-Code. Ein Beispiel dazu ist in Algorithmus 5.1 dargestellt.

Algorithm 5.1 Beispielalgorithmus

```

1: procedure ADDTWO NUMBERS( $x, y$ )
2:    $sum \leftarrow x + y$ 
3:   if  $sum = 42$  then
4:     FIXEVERYTHING

```

Das Paket acronym handelt Abkürzungen automatisch. Bei der ersten Verwendung sieht das so aus: Unified Modelling Language (UML). Wenn die Abkürzung nochmal verwendet wird, steht da nur noch UML.

Zusammenfassung

Ausblick