63.UKW-Tagung 2018

# Linux ist sexy Freiheit! DAB/UKW/FM-Kanäle scannen mit dem Raspberry-Pi3

**Hayati Aygün (UKW/TV-Arbeitskreis)**
h_ayguen@web.de

## Einleitung

Unter Windows gibt es anwenderfreundliche eierlegende Wollmichsau-Programme.
Linux ist entwicklerfreundlich: So kann man Anwendungen aus vielen kleinen Werkzeugen zusammensetzen.
Am Beispiel eines Scanners für die FMLIST[1](https://fmscan.org/) wird die Entwickler-Freundlichkeit aufgezeigt.

## Linux/Unix Paradigma: „kleine (schöne) Programme verknüpfen“

Mit den GNU Werkzeugen werden mit einem Linux sehr viele einfache Programme mit ausgeliefert. Hinzu kommt, dass die Shell (Eingabeaufforderung) sehr mächtig ist. Mit einer „Pipe“ können die Ausgaben eines Programms direkt als Eingabe für ein weiteres Programm benutzt werden, zum Beispiel:

```
cat -n Datei | grep Suchmuster
```

Hier wird eine Ausgabe der Datei inklusive Zeilennummern mit anschließender Filterung auf einen Suchbegriff durchgeführt. Das lässt sich noch erweitern, z.B. um Sortierung.
Viele Werkzeuge/Programme sind auf dieses Paradigma hin ausgelegt. Zur Steuerung des jeweiligen Programms werden Kommandozeilen-Optionen verwendet. Am obigen Beispiel „-n“, der Dateiname und das Suchmuster.
Die bekanntesten Vertreter dieser Programmgattung sind „sed“ und „awk“:

- Sed ist ein „stream editor“, der nicht nur einfache Textersetzungen erledigt.
- Awk ist eine mächtige Skriptsprache zur Analyse und Manipulation von Textdateien.

## Die Shell als Klebstoff

Die Bash (Bourne-Again SHell) kann neben der manuellen Eingabe auch Skripte verarbeiten. Hier zeigt sich die Mächtigkeit der Shell: Sie ist eine komplexe Programmiersprache mit Variablen, Arrays, Bedingungen, Schleifen, Funktionen, Pipes und der Programmausführung als Basis.

## Die Sexiness: Freie Software

Für Entwickler aber auch Anwender machen die verschiedenen „freien“ Softwarelizenzen den Sex-Appeal aus. Zu nahezu allen Programmen eines Linux-Systems ist der Quellcode nebst Bauanleitung – in Form von Skripten, Makefiles oder ähnlich – erhältlich.
Man kann folglich Erweiterung in Auftrag geben oder ggf. - Programmierkenntnisse vorausgesetzt – selbst Modifikationen und Erweiterungen vornehmen.
Je nach konkreter Lizenz muss die angepasste/erweiterte Software selbst wiederum inklusive des Quellcodes weitergegeben werden, so z.B. bei der GNU GPL (GNU General Public License) oder GNU LGPL (GNU Lesser General Public License). Somit wird mit den GNU Lizenzen verhindert,

dass ein kommerzieller Anbieter die Software verwendet und verteilt – ohne dem neuen Nutzer dieselben Rechte und Möglichkeiten zu geben.
Andere MIT- oder BSD-artige Lizenzen sind diesbezüglich freizügiger. Hier eine vollständigere Liste von Lizenzen – als Hilfestellung für Entwickler:

```
https://choosealicense.com/appendix/
```

Dennoch können Selbständige und Firmen mit freier Software Geld verdienen. Denn über den Preis sagen o.g. Lizenzen nichts aus. So wird für einige Distributionen und insbesondere für Dienstleistungen wie Updates, Support und Service Geld verlangt.

Bei kommunal genutzter Software sollte man davon ausgehen können, dass die Kommunen immer auch der Eigentümer der erstellten Software inkl. des Quellcodes sein sollten. Mit entsprechend „freier" Software würden sich die Kommunen nicht in die Abhängigkeit einzelner Firmen begeben. Man könnte jederzeit eine Erweiterung/Anpassung in Auftrag geben. Bei „closed-source" Software kann der alleinige Anbieter die Preise diktieren – solange dieser unter den Kosten einer kompletten Neu-Beauftragung liegt.

In Analogie würde das bedeuten, dass man nach dem Kauf eines Automobils nur noch zu Werkstätten dieses Herstellers gehen kann. Man hätte weder das Recht noch die technische Möglichkeit eine andere (freie) Werkstatt aufzusuchen. Darüber hinaus kann man auch keinen Blick in den Motorraum werfen. Ein freier Markt wäre das nicht.

## Die Anwendung: Ausgangssituation und Idee

Der Fernempfang von Rundfunksendern ist nicht nur ein verbreitetes Hobby von SWLs, sondern hilft auch den Funkamateuren, besondere Ausbreitungsbedingungen zu erkennen.
Rundfunksender im VHF-Band II können oft gut über RDS identifiziert werden, beim Digitalradio im VHF-Band III helfen diverse IDs.

Stationäre und mobile Systeme (Raspberry-Pi nebst RTL-SDR und Antenne) scannen nach Sendern in diesen Bändern, dekodieren die digitalen Sender-Informationen und übermitteln die Ergebnisse an die FMLIST[1], der Senderdatenbank des UKW/TV-Arbeitskreis e.V. Nach halbautomatischem Abgleich der Daten kann die Datenbank aktualisiert bzw. erweitert werden.

Bei mobilem Einsatz wird das System um einen GPS-Empfänger erweitert, so dass auch die Erfassungskoordinaten zu den empfangenen Sendern vorliegen.

Bei stationärem Einsatz liefert der zeitliche Verlauf über Anzahl erfasster Sender bzw. die Signalstärke der erfassten Sender statistische Informationen zur Beurteilung der Ausbreitungsbedingungen. Das System soll automatisiert über gute DX-Bedingungen informieren.

Mit dem Scan der VHF Bänder II (UKW/FM-Rundfunk, 87,5 – 108,0 MHz) und III (DAB-Rundfunk, 174 – 230 MHz) können Rückschlüsse auf das 2m-Amateurfunk-Band (144 – 146 MHz) geschlossen werden.

## Hardware Voraussetzungen und Wahl

Mit Blick auf Preis/Leistung, fällt die Entscheidung auf den Raspberry-Pi 3B+ mit

- 4 CPU-Kernen
- 1 GB RAM
- 4 USB Steckplätzen

und

- integriertem WLAN.

Grundsätzlich kommt auch ein Raspberry-Pi 2 oder Alternativen wie Odroid C1+/C2 in Frage. Diese wurden nicht getestet.
Hinzu kommt ein RTL-SDR Empfänger-Dongle mit dem RTL2832U A/D-Wandler Chip nebst R820T- bzw. R820T2-Tuner. Die Wahl des Tuners ist wichtig. Mit einem E4000-Tuner wurden keine guten Erfahrungen gemacht: die Tuner-AGC fehlt bzw. erfüllt nicht seinen Zweck. Weitere wichtige Kriterien für die Auswahl eines RTL-Dongles sind:

- Aluminum Gehäuse, zur Reduzierung von Einstrahlungen
- Antennen-Stecker: MCX, IEC oder SMA
- Frequenz-Stabilität < 1 ppm

Weiterhin wird eine SD-Karte für Betriebssystem und Anwendung benötigt.
Um die SD-Karte zu schonen werden die Ergebnisse auf einen USB-Speicher-Stick gesichert.

Hier die komplette Einkaufsliste:

| | |
|---|---|
| Raspberry Pi 3 Modell B+ Starterkit inkl.<br>• Netzteil<br>• Kühlkörper<br>• Gehäuse<br>• 16 GB MicroSDHC Speicherkarte inkl. USB-Adapter für PC/Notebook - falls kein Lesegerät vorhanden<br>• ggf. HDMI Kabel<br>• ggf. Netzwerkkabel | Mit einem Starterkit hat man schon mal vieles beisammen – insbesondere für den stationären Betrieb.<br><br>Das HDMI Kabel – nebst Bild-schirm und weiterem VGA-Adapter – wird lediglich zur initialen Erst-Einrichtung benötigt. Später kann der Kontakt über kabelgebundenes oder drahtloses Netzwerk aufgenommen werden, z.B. ssh/PuTTY und scp/WinSCP.<br><br>Für den Notfall sollte natürlich ein passendes Kabel in der Nähe liegen. |
| Optional: Gehäuse für Raspberry Pi 3 **B+**<br>• Aussparung für Kühlkörper auf der Unterseite<br>• Aussparung an der Stiftleiste, z.B. für den Anschluss eine Piepsers<br>• Lüfter auf Oberseite<br>• insgesamt 3 Kühlkörper | Gehäuse mit Lüfter zur besseren Kühlung. Der Raspberry kann unter Last ziemlich heiß werden. Bis 80°C Kerntemperatur wurden ohne Lüfter registriert. |
| USB Speicherstick, z.B. 16 GB oder mehr | Daten/Ergebnisse werden auf den USB-Stick gespeichert um die SD-Karte zu schonen bzw. Deren Lebensdauer zu erhöhen.<br>Die SD-Karte ist typischerweise die Schwach-stelle eines Raspberry-Pi. |
| RTL-SDR mit RTL2832 Chip<br>und R820T bzw. R820T2 Tuner.<br><br>Stecker: MCX / IEC / SMA ?<br><br>Vorschlag:<br>kurze 10 – 15 cm USB-Verlängerung | Günstiger Empfänger.<br>Je nach Gehäuse wird der Empfänger ziemlich heiß!<br>Ein frequenz-stabilisierender TCXO auf unter 1 ppm ist interessant, aber nicht notwendig.<br>Eine einmalige Kalibrierung ist in jedem Fall angebracht. |

| | |
|---|---|
| Antenne für VHF Band II und III<br>Ggf. Passender Adapter auf den Stecker des RTL-SDR Empfängers | |
| Optional:<br>GPS Maus des Typs “VK-162 G-Mouse”<br><br>alternativ: “Navilock NL-442U”<br><br>ggf. weiteres USB Verlängerungskabel | Bei der mobilen Erfassung sind die Koordinaten der Erfassungsortes sehr hilfreich.<br><br>Mit diesen GPS Empfängern wurde getestet. Andere Modelle, die von gpsd unterstützt werden, könnten auch funktionieren.<br>Aber: Kein Gewähr! |
| Optional: Piezo Summer zur Benachrichtigung.<br>https://www.amazon.de/dp/B00W8YEG8S/<br><br>Dazu noch kurze Drahtbrücken (weiblich-weiblich) zum Anschluss über die Stiftleiste des Raspberry Pi | Dieses Modell, mit dem + Aufdruck, sieht nach dem getesteten funktionierenden Modell aus.<br>Ggf. wäre eine Anpassung des Programms notwendig.<br><br>Im Moment wird am Ende eines kompletten (UKW+DAB) Scans – nach der Sicherung der Ergebnisse – eine Piepsfolge abgespielt.<br><br>Der Summer wird “irgendwie” außen am Gehäuse befestigt. |
| Optional: Drahtbrücke (weiblich – männlich) zusätzlich zu obigem weiblich – männlich. | Sicheres Herunterfahren des Raspberry Pi nach Kurzschluss der Drahtbrücken. |
| Optional für mobilen Betrieb:<br>Kfz-Adapter für 5V USB Anschluß – oder gleich mehrere, z.B. für den Becherhalter | Unbedingt auf ausreichende Stromstärke achten. |

## Software-Komponenten zum Scan von UKW/FM

Um an Senderinformationen zu kommen muss der RDS[2] (Radio Data System) - Anteil aus dem FM demoduliertem MPX-Signal eines Senders dekodiert werden. Zum Dekodieren kann das MIT-lizensierte ***redsea***[3] verwendet werden. Autorin ist Oona Räisänen, OH2EIQ.

***redsea*** wiederum verwendet insbesondere die Bibliothek ***liquid-dsp*** des Autors Joseph D. Gaeddert, welches unter X11/MIT Lizenz steht. Die Bibliothek stellt diverse Signalverarbeitungsfunktionen, wie z.B. eine AGC (Automatic Gain Control) oder Filterfunktionen bereit, um nur wenige zu benennen.

Zuvor muss der Sender mit dem RTL-SDR-Dongle empfangen werden. Dies könnte mit dem Programm ***rtl_fm*** der ***librtlsdr***[4]-Bibliothek erledigt werden. Die Bibliothek unterliegt der GNU GPL Lizenz und hat verschiedene Autoren. Dieses ***rtl_fm*** Programm erledigt dann auch die FM Demodulation gleich mit. Dies erscheint auf den ersten Blick praktisch. Auf den zweiten Blick merkt man, das der Raspberry Pi 3 sich langweilt, da nur wenige Cores (CPU-Kerne) der ARM belastet werden. Die anderen Kerne müssen warten. Der Scan bzw. Test aller 206 FM-Kanäle im 100 kHz Raster zwischen 87,5 bis 108,0 MHz benötigt bei 3 Sekunden je Frequenz mindestens 618 Sekunden, also über 10 Minuten.

Mit dem Programm ***rtl_sdr*** aus derselben Bibliothek können ca. 2 MHz Bandbreite, mit mehreren darin enthaltenen Sendern, empfangen und als I/Q-Rohdatenstrom aufgezeichnet werden. Die 4 Kerne der Raspberry-CPU werden deutlich besser ausgelastet, wenn parallel zur Aufzeichnung die anderen Kerne die Aufzeichnung des vorherigen 2 MHz Bandes verarbeitet. Parallele Ausführung stellt für Linux kein Probleme dar – solange genügend Rechenkapazitäten vorhanden sind. In unserem Fall ist es wichtig, dass die Aufzeichnung keine Aussetzer hat. Dies kann mit der Festlegung von Prozess-Prioritäten mit dem Programm ***nice*** erfolgen. Im einfachsten Fall wird die Priorität der Verarbeitung heruntergestuft.
Zur Verarbeitung fehlt bei Nutzung von ***rtl_sdr*** die FM-Demodulation sowie ein vorangestellter Frequenz-Mischer, der mittels komplexer NCO (Numerically Controlled Oscillator), die jeweilige Frequenz auswählt und anschließend das Signal mit einem Tiefpass-Filter für die Demodulation aufbereitet und dezimiert. Zusammenfassend sind 3 Schritte zu bewältigen:

- jeweilige Frequenz mischen
- Tiefpass-Filter + Dezimation der Samplerate
- FM Demodulation

Für alle obigen Schritte und einige mehr kann das Programm ***csdr***[5] von Andras Retzler, HA7ILM, verwendet werden. Aus der gleichen Feder stammt das Programm OpenWebRX. ***csdr*** unterliegt weitestgehend der BSD-Lizenz. Optional kann die Funktionalität von ***csdr*** mit GPL-lizenzierten Anteilen erweitert werden.

Die Parallelisierung und Stapelverarbeitung übernimmt GNU ***parallel***[6]. Der Name ist Programm. Man gibt an, bis wie viele Jobs (wir wählen 3) parallel ausgeführt werden sollen. Zusätzlich benötigt parallel die auszuführenden Programme inkl. Parametern. Übersteigt die Anzahl auszuführender Programme die Job-Anzahl, so werden diese nacheinander ausgeführt, so dass ständig 3 der angegebenen Programme gleichzeitig ausgeführt werden. Ole Tange aus Frederiksberg, Dänemark, ist der Entwickler. Die Lizenz ist etwas ungewöhnlich – aber dennoch GNU GPL: Der Autor möchte, dass auf seinen wissenschaftlichen Artikel verwiesen wird. Alternativ soll die Weiterentwicklung mit 10.000 EUR finanziert werden. Hier die Referenz:

```
O. Tange (2011): GNU Parallel - The Command-Line Power Tool,

;login: The USENIX Magazine, February 2011:42-47.
```

Ungewöhnlich für ein GPL lizenziertes Programm ist, dass das Programm mit jedem Start darauf aufmerksam macht. Ein ähnliches Verhalten ist man eher von Shareware Programmen unter Windows gewöhnt. Man kann der Bedingung einwilligen oder das Programm mit der Option *–no-notice* aufrufen, sodass die Meldung unterdrückt wird. Man ist natürlich auch frei, andere Alternativen zu nutzen.

Die Parallelisierung hat allerdings nicht nur Vorteile: Wenn man die Temperatur im Kern des Raspberry-Pi beobachtet, so steigt diese – durch Nutzung aller Kerne – bedrohlich an. Es wurden regelmäßig Temperaturen bis 80°C beobachtet.

Die Rechenlast – und in Folge auch die Temperatur – wird durch die Voruntersuchung des Spektrums reduziert. Für die spektrale Voruntersuchung reichen 200 – 500 ms Signal aus, um Frequenzen mit zu niedrigem Signal-Rausch-Abstand auszusortieren. Die weitere Verarbeitung würde aktuell jeweils 4 Sekunden Signal – je Frequenz – verarbeiten.

Um die Scan-Dauer noch weiter zu reduzieren, sollte man die Ergebnisse der RDS-Dekodierung per Programm beobachten und die Ausführung der kompletten Pipe, durch Beendigung des Beobachtungsprogrammes, früher abbrechen. Im Moment findet die Dekodierung immer über die komplette Aufzeichnung statt – abgesehen von der Voruntersuchung.

## Software-Komponenten zum Scan von DAB (Digital Audio Broadcasting)

Ein DAB-Kanal belegt ca. 1,6 MHz Bandbreite. Limitiert durch die Empfangsbandbreite eine RTL-SDR-Dongles von knapp 2 MHz können nicht mehrere Kanäle gleichzeitig empfangen und parallel verarbeitet werden. Test aller 38 DAB-Kanäle im VHF Band III benötigt bei mind. 3 Sekunden je Frequenz insgesamt 114 Sekunden, also knapp 2 Minuten.

Für Linux existieren verschiedene DAB Programme. Am vielversprechendsten ist Qt-DAB[7] des niederländischen Autors Jan van Katwijk. Neben der GUI-Variante hat er auch einige andere Varianten erstellt. Diese benötigen keine grafische Bedienoberfläche und sind somit auch für die Kommandozeile des Raspberry-Pi geeignet. Der Nachteil hierbei: Obwohl die GUI eine Scan Funktion anbietet, bieten die Kommandozeilen-Varianten der ***dab-cmdline***[8] diese Funktion nicht.

Durch die GPL-Lizenz war es möglich, eines der Kommandozeilen-Programme zu erweitern. Example-10 der dab-cmdline kann per Option in den Scan-Modus versetzt werden: Nach Ausgabe der Programminformationen wird das Programm beendet. Der eigentliche Scan erfolgt durch sequentiellen Aufruf der dab-cmdline über ein bash/Shell-Skript für jede Frequenz einzeln.

Zur Beschleunigung des DAB-Scans wurde eine schnelle Vorauswahl entwickelt:
DAB verwendet eine OFDM Modulation, welches für jeden Symboltakt das Ende des korrespondierenden Zeitsignals jeden Symbols zusätzlich an den Anfang stellt. Dieses sog. „Guard-Intervall“ ist eigentlich für die Entzerrung gedacht. In diesem Zeitintervall soll die Impulsantwort des Kanals abklingen. Zusätzlich zur Entzerrung lässt sich das vorangestellte Duplikat per Autokorrelation zur schnellen Vorauswahl verwenden. Die Autokorrelation prüft die Selbstähnlichkeit des Signals in gewissen Zeitabständen. Für DAB Mode I, inzwischen der einzige genutzte Mode, beträgt der Zeitabstand zwischen Duplikat (Guard Intervall) und Original exakt eine Millisekunde.
Die Trennschärfe der Autokorrelation ist nicht perfekt. Um nicht fälschlicherweise DAB-Sender abzuweisen, müssen mit einer niedrigen Akzeptanz-Schwelle (Schwellwert für den Autokorrelations-Koeffizienten) einige Falsch-Positiv Frequenzen zugelassen werden. Dennoch führt diese Vorauswahl insgesamt zu einem schnelleren Scan.

## Ausblick

Um das System abzurunden, benötigt es noch einiger Arbeiten.

Redsea liefert seine Ergebnisse in JSON:

```
{"bler":2,"di":
{"dynamic_pty":true},"group":"0A","is_music":true,"pi":"0xD30A","prog_
type":"Pop music","ps":"antenne1","ta":false,"tp":true}
```

Diese JSON-Ergebnisse müssen noch ausgewertet und zusammengefasst werden.

Die dab-cmdline liefert die Ergebnisse ähnlich unstrukturiert:

```
ensemblenameHandler: 'DR Deutschland  ' ensemble (Id 10BC) is
recognized
```

```
checked program 'DRadio DokDeb   ' with SId D240

        audioData:

                protLevel       =2: 'EEP 3-A'

                codeRate        =2: '1/2'

                bitRate =48

                ASCTy   =63: 'DAB+'

                country ECC E0, Id D: 'Germany'

                programType     =3: 'Information'
```

Nach Zusammenfassung der FM/DAB-Ergebnisse muss auch der Upload zur FMLIST entwickelt werden. Dab-cmdline liefert – im Gegensatz zu Qt-DAB – bislang nicht den TII (siehe Vortrag von Ulrich Onken, DK2GO). Der Kontakt zum Entwickler steht, sodass die Chancen gut sind, den TII auch in die dab-cmdline aufzunehmen.

Für den Funkamateur ist die Auswertung/Bestimmung von Überreichweiten und Alarmierung hierüber sicherlich interessanter. Die Bestimmung von Überreichweiten sollte, über die Beobachtung von Signal-Rausch-Abständen über die Zeit gut, möglich sein.

Linux und Open-Source haben, mit der Menge an vorhandenem Quellcode sowie den Möglichkeiten zur Automation, dieses Projekt erst realisierbar gemacht. Ohne die vorhandenen Programme, auf die man aufsetzen konnte, ist fraglich, ob man das Unterfangen überhaupt erst in Angriff genommen hätte.

Weitere Informationen zum Projekt werden auf der Open-Source-Plattform github bereitgestellt:

`https://github.com/hayguen/fmlist_scan`

## Referenzen


[1] https://www.fmlist.org/
Die Senderdatenbank des UKW/TV-Arbeitskreises e.V.

[2] https://de.wikipedia.org/wiki/Radio_Data_System

[3] https://github.com/windytan/redsea
RDS Dekoder Software

[4] https://github.com/librtlsdr/librtlsdr
Software zum Betrieb von Geräten mit RTL2832U-Chipsatz als SDR: rtl_sdr, rtl_fm, ..

[5] https://github.com/simonyiszk/csdr
DSP Bibliothek und Kommandozeilen-Tool

[6] https://www.gnu.org/software/parallel/
Tool zur parallelen Ausführung

[7] https://github.com/JvanKatwijk/qt-dab
DAB-Dekoder mit GUI für verschiedene Empfänger

[8] https://github.com/JvanKatwijk/dab-cmdline
DAB-Dekoder für die Kommandozeile