

**Aufbau und Inbetriebnahme  
eines DAB-Multiplexers/-Modulators  
auf Basis der ODR-mmb-Tools  
und des DAB-Modulators EasyDab v2  
in einem Werkzeugkoffer  
(DAB-Toolbox-Sender)**

## EINLEITUNG

---

Lokalen und kleinen Rundfunkveranstaltern ist der Einstieg in digitales Radio erschwert. Der Betrieb von Soft- und Hardware für DAB/DAB+ ist oft mit hohen finanziellen Mitteln verbunden, die die kleineren Rundfunkveranstalter nicht zur Verfügung haben.

Um diesen Rundfunkveranstaltern den Einstieg in digitales Radio zu erleichtern, wurden in den Jahren 2013/14 und 2016 die ersten DAB-Small-scale-Sender auf Basis der Open-Source Software ODR-mmbTools im Auftrag der Landeszentrale für Medien und Kommunikation Rheinland-Pfalz in Kooperation mit der Technischen Universität Kaiserslautern [1] und der Hochschule Kaiserslautern [2] realisiert. Die Software ist eine Weiterentwicklung der ODR-Tools des Communication Research Centers (CRC) Kanadas durch die Organisation opendigitalradio.org. Der DAB-Multiplexer und die VHF-Sendeeinheit wurden hierbei in zwei transportablen Flightcases untergebracht.

Da der DAB-Sender in den Flightcases als doch ziemlich sperrig für Demonstrationszwecke, z. B. auf Vortragsveranstaltungen, erwiesen hat, wurde im Rahmen einer Praxisphase in Kooperation mit der Hochschule Kaiserslautern ein DAB-Sender in einem kleinen Aluminiumwerkzeugkoffer (Toolbox) realisiert. Der DAB-Sender besteht hierbei aus einem Raspberry Pi, der den DAB-Multiplex erzeugt, und dem Kleinstleistungssender EasyDAB v2, der als Modulator dient. Die Kommunikation der beiden Geräte erfolgt über Ethernet. Der Toolbox-Sender besitzt dafür einen eingebauten Netzwerkrouter.

Das Handbuch befasst sich mit dem Aufbau und der Realisierung des DAB-Toolbox-Senders. Außerdem wird die Einrichtung und die Bedienung des Senders beschrieben.

Das Handbuch basiert auf dem Abschlussbericht von Herrn Johannes Gehres, Studierender der Hochschule Kaiserslautern, Fachbereich Angewandte Ingenieurwissenschaft, Studiengang Elektrotechnik, der dieses Projekt in seiner Praxisphase vom bis in der LMK umgesetzt hatte. Seit der Erstellung des Abschlussberichtes wurden im Laufe der Nutzung des DAB-Toolbox-Senders an einzelnen Stellen Ergänzungen oder redaktionelle Änderungen mit zusätzlichen Erläuterungen der Abläufe gegeben, die durch Joachim Lehnert (LMK) vorgenommen wurden.

(Stand: 14.06.2018).

## INHALTSVERZEICHNIS

---

1	Konzept .....	4
1.1	Projektziel .....	4
1.2	Anforderungen .....	4
1.2.1	Anforderungen an die Hardware .....	4
1.2.2	Anforderungen an die Software .....	5
1.3	Umsetzung der Anforderungen .....	5
2	Komponenten .....	8
2.1	Hardwarekomponenten .....	8
2.1.1	Raspberry Pi 3 .....	8
2.1.2	EasyDab v2 .....	8
2.1.3	Netzwerk-Router .....	8
2.1.4	USV-Board .....	9
2.1.5	Audioverstärker .....	9
2.1.6	USB-Soundkarte .....	9
2.1.7	Touchdisplay .....	9
2.2	Softwarekomponenten .....	10
2.2.1	Betriebssystem .....	10
2.2.2	Systemkomponenten .....	10
2.2.3	Audiosoftware .....	14
2.2.4	DAB-Softwarekomponenten .....	15
2.2.5	ODR-mmbTools .....	17
2.3	Softwarekonzept .....	22
3	Aufbau des DAB-Toolbox-Senders .....	23
4	Bedienung .....	26
4.1	Benutzereinstellungen .....	26
4.2	Einrichtung des Mailkontos .....	26
4.2.1	Prozessüberwachung .....	27
4.3	Bedienung der S.USV-Software .....	27
4.4	Umschaltung zwischen internem LCD und externem HDMI-Anschlusses .....	28
4.5	Netzwerkeinstellungen .....	28
4.5.1	ERSTINSTALLATION DES ROUTERS .....	30
4.5.2	ROUTERKONFIGURATION IM BETRIEB .....	30
4.5.3	Internetverbindung über DHCP/LAN-Netzwerk .....	31
4.5.4	Internetverbindung über ein WLAN-Netzwerk .....	31
4.6	Audioquellen .....	33
4.6.1	Audio-Routing .....	33
4.6.2	Externer Audioeingangsstrom .....	33

4.6.3	MP3-Dateien/Playlists VLC-Player .....	34
4.6.4	Internet-Livestreams.....	37
4.7	ODR-mmbTools.....	38
4.7.1	ODR-AudioEnc.....	38
4.7.2	ODR-PadEnc .....	38
4.7.3	ODR-DabMux .....	39
4.7.4	ODR-DabMod .....	41
4.7.5	DABlin.....	43
4.8	Ausführung der ODR-mmbTools mittels Skript .....	44
4.8.1	Skripterstellung mittels dem Tools GUI4ODR .....	46
4.9	Ausführung der ODR-mmbTools mittels Supervisor.....	46
4.9.1	Webserver aktivieren.....	46
4.9.2	Konfigurationsdatei zum Starten des Prozesses.....	47
4.9.3	Email-Benachrichtigung aktivieren .....	47
4.9.4	Überwachung per Weboberfläche.....	48
4.9.5	DAB-Sender mit Supervisor mittels Skript starten und beenden .....	48
4.10	Bedienung des EasyDAB-Boards .....	50
5	Anlagen .....	52
5.1	Installationskript raspdab.sh .....	52
5.2	Parameter der S.USV Software .....	56
5.3	Verwendung des externen HDMI-Ausgang.....	57
5.3.1	display .....	57
5.3.2	Automatisches Starten auf den LCD-Display .....	60
5.4	JACK.....	61
5.5	ODR-PadEnc .....	61
5.6	ODR-AudioEnc.....	63
5.7	ODR-DabMux .....	65
5.8	ODR-DabMod .....	67
5.9	Konfigurationsdatei Multiplex .....	68
5.10	Konfigurationsdatei B100.ini.....	70
5.11	DABlin.....	71
6	Verzeichnisse .....	72
6.1	Abbildungsverzeichnis .....	72
6.2	Tabellenverzeichnis.....	72
6.3	Quellenverzeichnis.....	73

# 1 KONZEPT

---

## 1.1 PROJEKTZIEL

Ziel dieser Arbeit ist es, einen funktionsfähigen DAB-Sender in einen tragbaren Koffer (Werkzeugkoffer = Toolbox) einzubauen. Es wird die Erstellung des Konzepts beschrieben sowie der Aufbau des Toolbox-Senders und dessen Bedienung.

## 1.2 ANFORDERUNGEN

An den DAB-Toolbox-Sender wurden verschiedene Anforderungen gestellt, die sich in Anforderungen an die Hardware und Anforderungen an die Software gliedern. Die Anforderungen sollen nach Fertigstellung des Senders eingehalten werden.

### 1.2.1 Anforderungen an die Hardware

An den DAB-Toolbox-Sender wurden für die Hardware folgende Anforderungen gestellt:

- Bedienung per Touchdisplay.
- Der Sender soll in einem transportablen Koffer untergebracht werden.
- Der Sender soll Lautsprecher zum Abspielen von Audioströmen besitzen.
- Die Möglichkeit zur Verbindung von Bluetooth-Geräten z.B. Tastatur soll bestehen.
- Folgende Schnittstellen sollen zur Verfügung stehen:
  - Drei USB 2.0-Anschlüsse z.B. zur Verwendung von Maus, Tastatur und oder USB-Stick.
  - Zwei RJ45-Ethernetanschlüsse zur Verbindung mit dem Internet und für einen externen EDI- oder ZeroMQ-Stream.
  - Ein WLAN-Antennenanschluss zur alternativen Verbindung mit dem Internet.
  - Ein HDMI-Anschluss zum Anschluss eines Monitors oder Beamers.
  - Ein Stereoeingang in Form von zwei Cinchbuchsen soll die Bearbeitung von externen Audiosignalen ermöglichen
    - Ein GPS-Antennenanschluss zum möglichen Betreiben von Gleichwellennetzen
    - Ein BNC-Anschluss zum Anschließen einer Antenne oder eines HF-Verstärkers

## 1.2.2 Anforderungen an die Software

An die Software des DAB-Toolbox-Senders wurden die folgenden Anforderungen gestellt:

- Es sollen Internet-Livestreams, externe Audioeingangsströme und MP3-Dateien in ein DAB-gerechtes ETI-Signal codiert werden.
- Programmbegleitete Daten in Form von Sildeshows und Dynamic Label sollen erzeugt werden.
- Ein DAB-Multiplex soll in den Formen EDI und ETI-NI zur Verfügung gestellt werden.
- Der Multiplexer soll DAB-Basisbanddaten berechnen, sodass man ihn mit einem USRP betreiben kann.
- Zur Überwachung sollen Statusmeldungen per E-Mail versandt werden.
- Die richtige Funktionsweise des Multiplexes soll überprüft werden.

## 1.3 UMSETZUNG DER ANFORDERUNGEN

Die Umsetzung der Anforderungen werden anhand von verschiedenen Blockdiagrammen erklärt.

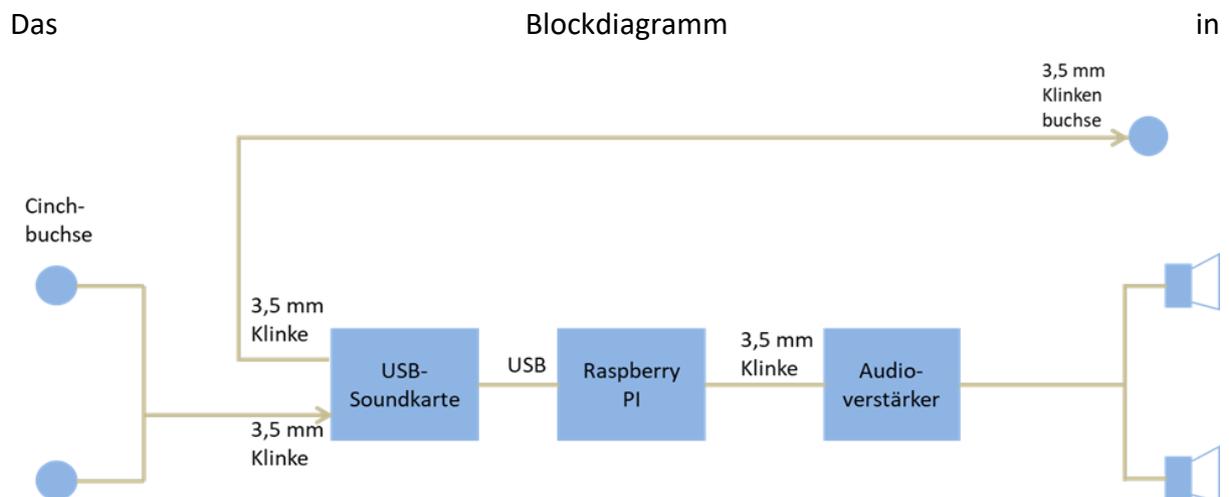


Abbildung 1 beschreibt die Verarbeitung der Audioeingänge. Das Stereosignal, eingespeist durch die beiden Cinchbuchsen, wird an eine USB-Soundkarte übermittelt. Die Soundkarte ist per USB mit dem Raspberry Pi verbunden. Die Audioausgabe erfolgt über den Audioverstärker mit zwei Lautsprechern oder über einen 3,5 mm Klinkenanschluss.

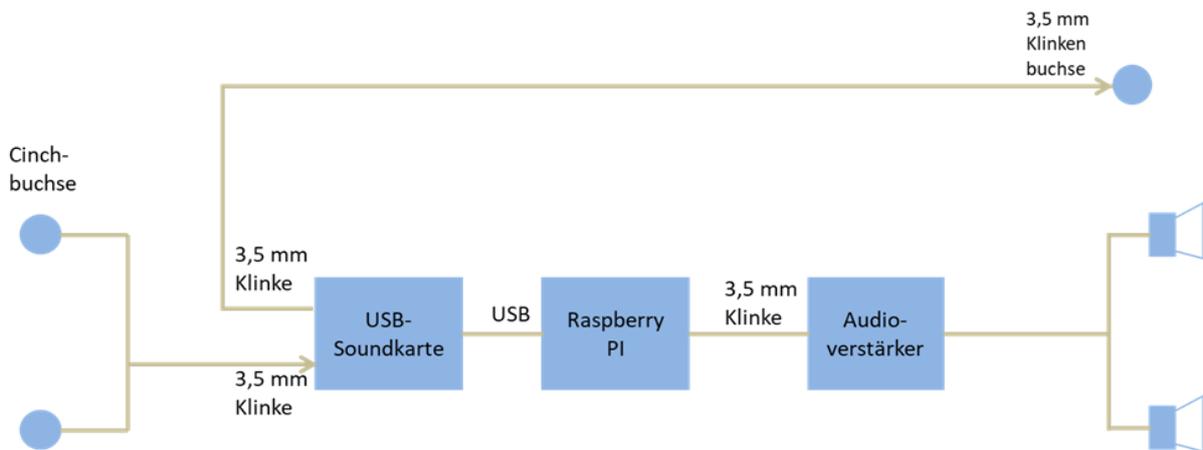


Abbildung 1: Blockdiagramm Audio

In Abbildung 2 wird die Umsetzung der Kommunikation des EasyDAB-Boards und des Raspberry Pi beschrieben sowie die Anbindung an das Internet. Die Kommunikation des Raspberry Pis mit dem EasyDAB-Board erfolgt über Ethernet, hierzu kommt ein Netzwerkrouter zum Einsatz. Über die beiden RJ45-Anschlüsse lassen sich einmal eine Anbindung an das Internet realisieren und zum anderen ein externes EDI-Signal empfangen oder versenden. Durch das Herausführen der SMA-Anschlüsse des EasyDAB-Boards kann eine BNC-Antenne und eine GPS-Antenne angeschlossen werden.

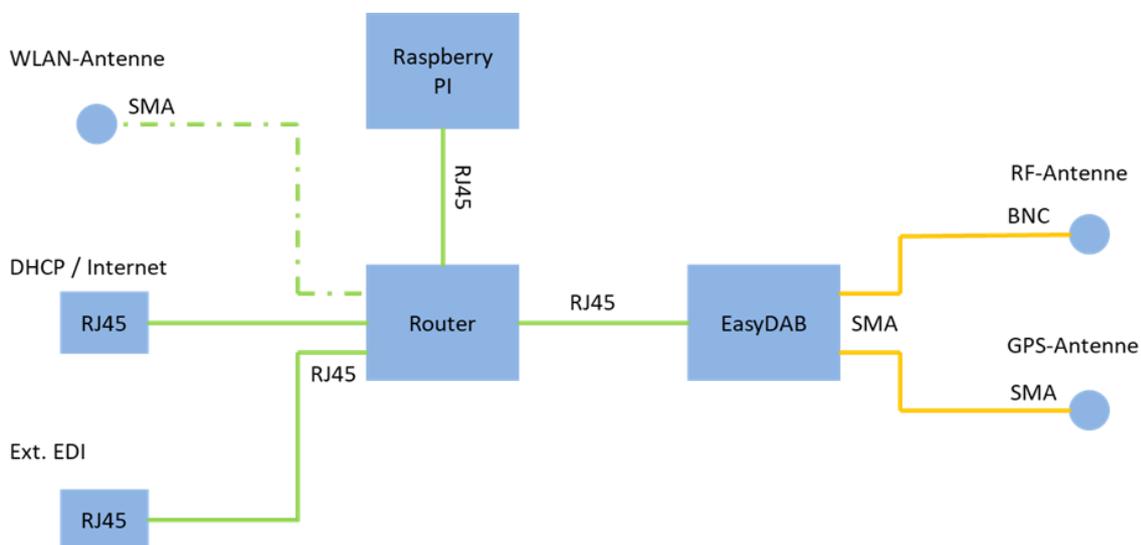


Abbildung 2: Blockdiagramm Netzwerk

Die Verarbeitung der Videoschnittstellen wird in Abbildung 3 dargestellt. Der Raspberry Pi kann mit einem HDMI-Gerät verbunden werden. Außerdem ist er über ein Flachbandkabel fest an einen Touchdisplay angeschlossen.

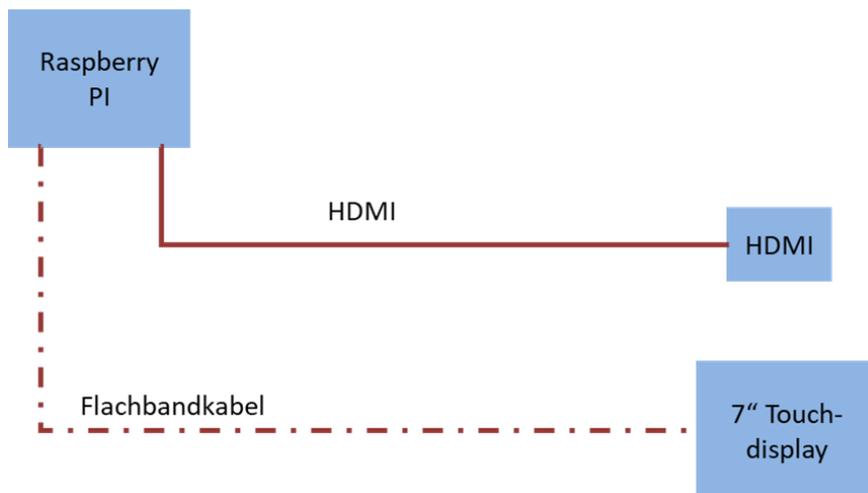


Abbildung 3: Blockdiagramm Video

## 2 KOMPONENTEN

---

Damit der DAB-Toolbox-Sender nach den zuvor definierten Anforderungen funktioniert, kommen verschiedene Komponenten zum Einsatz. Diese Komponenten lassen sich in Hardware- und Softwarekomponenten untergliedern.

### 2.1 HARDWAREKOMPONENTEN

Anhand der Anforderungen und der in Kapitel 1.3 beschriebenen Blockdiagramme wurden für die Hardware die folgenden Komponenten ausgewählt.

#### 2.1.1 Raspberry Pi 3

Der Raspberry Pi [3] ist ein Einplatinencomputer in der Größe einer Kreditkarte. Er besitzt einen ARM-Mikroprozessor mit 4 CPU-Kernen, die mit je 1,2 GHz takten. Außerdem besitzt er einen 1 GB großen Arbeitsspeicher, integriertes WLAN und ein Bluetooth-Interface. Das Bluetooth-Interface wird für die Anbindung einer kabellosen Tastatur verwendet. Der Raspberry Pi dient zur Verarbeitung der Eingangsdatenströme, indem er diese codiert und den Multiplex bildet. Darüber hinaus berechnet er die Basisbanddaten.

#### 2.1.2 EasyDab v2

Das EasyDab v2 Board [4] ist eine platzsparende DAB-HF-Sendeeinheit. Es besitzt einen Xilinx FPGA, der dazu dient, die Basisbanddaten anhand eines eingehenden Multiplexsignals zu berechnen. Außerdem können die HF-Sendefrequenz und der DAB-Mode eingestellt werden. Das Board hat einen integrierten GPS-Empfänger, der es ermöglicht, eine GPS-Synchronisation für Gleichwellennetze auszuführen.

#### 2.1.3 Netzwerk-Router

Der Netzwerk-Router „BR-6428nC“ der Firma Edimax besitzt 4 Fast-Ethernet-LAN-Ports, einen WAN-Port und ein WLAN-Modul. Der Router dient zur Kommunikation zwischen EasyDAB-Board und Raspberry Pi. Außerdem stellt er eine Internetverbindung für den Raspberry Pi zur Verfügung. Diese Internetverbindung kann über LAN oder WLAN betrieben werden. Durch einen freien LAN-Port ist es weiterhin möglich, ein externes EDI-Signal zu empfangen oder zu versenden.

#### 2.1.4 USV-Board

Das USV-Board „S.USV“ der Firma Olmatic ist eine Unterspannungsversorgung für den Toolbox-Sender. Es verfügt über einen Weitbereichsspannungsanschluss sowie über einen Anschluss für einen Akku. Das Board kann bis zu 3 A liefern, wodurch es ohne Probleme den Raspberry Pi mit Touchdisplay, das EasyDAB-Board sowie den Netzwerk-Router versorgen kann. Bei Ausfall der externen Spannungsversorgung schaltet das Board auf einen Akku-Betrieb um. Somit ist das System gegen kurzzeitige Spannungseinbrüche gesichert. Das S.USV-Board führt außerdem beim Unterschreiten der kritischen Kapazität des Akkus einen File-Safe-Shutdown durch, damit die SD-Karte vor Beschädigungen geschützt wird. Die Kapazität des LiPo-Akkus beträgt 3000 mAh, wodurch eine Spannungsüberbrückung von einigen Minuten erreichbar ist.

#### 2.1.5 Audioverstärker

Der Audioverstärker der Firma Vellemann ist ein Lötbausatz. Er besitzt zwei 5 W 4-32 Ohm Lautsprecheranschlüsse. Mit dem Audioverstärker lassen sich die zwei 2 W Lautsprecher des Toolbox-Senders betreiben. Er wird über Klinke mit dem Raspberry Pi verbunden. Somit lässt sich das DAB-Audiosignal überprüfen.

#### 2.1.6 USB-Soundkarte

Die Creative Soundblaster Play! 3 ist eine USB-DAC-Soundkarte. Sie besitzt zwei 3,5 mm Klinkenanschlüsse für den Anschluss von Audioausgabegeräten und für den Anschluss von Audioeingabegeräten. Die Aufnahmequalität des Audioeingangs beträgt 24 bit 48 kHz.

#### 2.1.7 Touchdisplay

Das 7 Zoll-Touchdisplay der Raspberry Pi Foundation ist ein kapazitiver Touchscreen mit einer Auflösung von 800x480 Pixel. Das Display kann mit einem Flachbandkabel direkt an den Displayanschluss des Raspberry Pi angeschlossen werden. Da der Raspberry Pi und das Touchdisplay vom selben Hersteller sind, ist keine zusätzliche Treiberinstallation notwendig.

## 2.2 SOFTWAREKOMPONENTEN

Die folgenden Softwarekomponenten wurden anhand der definierten Anforderungen ausgewählt.

### 2.2.1 Betriebssystem

Als Betriebssystem für den Raspberry Pi wurde die Linux-Distribution Raspbian Jessie auf der SD-Karte installiert. Raspbian basiert auf der Distribution Debian und wurde speziell für den Raspberry Pi entwickelt. Sie wird unter anderem verwendet, da die ODR-mmbTools unter Linux entwickelt worden sind. So ist ein stabiler Betrieb gewährleistet.

Die aktuelle Linux-Distribution Raspbian Stretch kommt nicht zur Anwendung, da mit dieser Distribution bei der Installation der ODR-mmb-Tools einige Fehler aufgetreten sind.

### 2.2.2 Systemkomponenten

#### 2.2.2.1 *LightDM*

Damit eine virtuelle Tastatur während des Login-Fensters verwendet werden kann, wird als Display-Manager GDM3 anstatt LightDM verwendet. Mit dem folgenden Befehl lässt sich GDM3 installieren:

```
sudo apt-get install gdm3
```

#### 2.2.2.2 *Twofing*

Twofing ist ein Programm, das einen Rechtsklick bei angeschlossenem LCD-Touchdisplay simuliert. Der Rechtsklick wird durch Berühren des Displays mit zwei Fingern ausgeführt. Um das Programm zu installieren und zu aktivieren, wird wie folgt vorgegangen:

Zuerst werden die folgenden Voraussetzungen installiert:

```
sudo apt-get install build-essential libx11-dev libxtst-dev libxi-  
dev x11proto-randr-dev libxrandr-dev  
sudo apt-get install xserver-xorg-input-evdev
```

Anschließend wird Twofing heruntergeladen und entpackt:

```
wget http://plippo.de/dwl/twofing/twofing-0.1.2.tar.gz  
tar -xvzf twofing-0.1.2.tar.gz
```

Nach dem Wechsel in den entpackten Ordner *twofing-0.1.2/* kann das Programm installiert werden:

```
make
sudo make install
```

Danach muss die X11-conf Datei bearbeitet werden. Mit folgendem Befehl öffnet man die Datei mit dem Editor nano:

```
sudo nano /usr/share/X11/xorg.conf.d/40-libinput.conf
```

Hier wird am Ende der Datei folgendes eingetragen:

```
Section "InputClass"
  Identifier "calibration"
  Driver "evdev"
  MatchProduct "FT5406 memory based driver"

  Option "EmulateThirdButton" "1"
  Option "EmulateThirdButtonTimeout" "750"
  Option "EmulateThirdButtonMoveThreshold" "30"
EndSection
```

Jetzt werden die folgenden neuen Rule-Dateien erstellt:

```
sudo nano /etc/udev/rules.d/70-touchscreen-raspberrypi.rules
```

Der Inhalt der Datei muss wie folgt lauten:

```
KERNEL=="event*",ATTRS{name}=="FT5406 memory based
  driver",SYMLINK+="twofingtouch",MODE="0440"
```

Die zweite Rule-Datei wird folgendermaßen erstellt:

```
sudo nano /etc/udev/rules.d/70-touchscreen-egalax.rules
```

Der Inhalt dieser Datei lautet:

```
KERNEL=="event*",ATTRS{name}=="FT5406 memory based
  driver",SYMLINK+="twofingtouch",RUN+="/bin/chmod a+r
  /dev/twofingtouch"
```

Damit Twofing bei jedem Neustart des Systems automatisch gestartet wird, wird in das Verzeichnis *~/config/autostart* gewechselt und die Datei *twofing.desktop* mit dem folgenden Inhalt erzeugt:

```
[Desktop Entry]
Type=Application
Name=Twofing
Exec=twofing
StartupNotify=false
```

Zum Schluss muss das Systems neu gestartet werden.

### 2.2.2.3 S.USV Daemon

Damit der Raspberry Pi und das S.USV-Board untereinander kommunizieren können, wird die S.USV Daemon Software benötigt. Mit der Software lassen sich Statusmeldungen des Unterspannungsversorgungsboards anzeigen. Außerdem ist es möglich, bestimmte Einstellungen, wie beispielsweise das Herunterfahren nach einer bestimmten Zeit im Akkubetrieb, festzulegen. Da das Board und der Raspberry Pi über die I<sup>2</sup>C-Schnittstelle kommunizieren, müssen zuerst die I<sup>2</sup>C-Tools installiert werden:

```
sudo apt-get install python-smbus
sudo apt-get install i2c-tools
sudo reboot
```

Nach dem Neustart kann die Funktionsweise der I<sup>2</sup>C-Tools mit dem folgenden Befehl überprüft werden:

```
Sudo i2cdetect -y 1
```

Falls das Board auf dem Raspberry Pi aufgesteckt ist, sollten folgende Adressen erscheinen: *0x0F* und *0x68*.

Anschließend kann die Clientsoftware von der Downloadseite des Herstellers heruntergeladen werden (<https://shop.olmatic.de/en/content/7-downloads>).

Das heruntergeladene Tar-Verzeichnis kann mit dem folgenden Befehl entpackt und danach installiert werden:

```
sudo tar -xvf susvd-en-x.x-all.tar
sudo dpkg -i susvd-en-x.x-all.deb
```

Die Software findet sich nach erfolgreichem Installationsvorgang im Verzeichnis */opt/susvd*. Nach dem Wechsel in dieses Verzeichnis kann der Daemon mit dem folgenden Befehl gestartet werden:

```
sudo ./susvd -start
```

Eine detaillierte Installationsanleitung findet sich unter [5].

#### 2.2.2.4 Verwendung des externen HDMI-Anschlusses

Für Demonstrationszwecke kann an dem Toolbox-Sender ein externes HDMI-Signal abgegriffen werden, mit dem sich zum Beispiel der Bildschirm des Raspberry Pi auf einen Beamer übertragen lässt. Da der Raspberry Pi keine zwei Videosignale gleichzeitig bereitstellt, kann entweder das Display oder der HDMI-Ausgang aktiv sein. Zwischen den beiden Videoausgängen kann mit dem Skript *display* (Anlage 5.3.1) umgeschaltet werden. Es wird mit dem Befehl `sudo bash display` ausgeführt. Das Skript veranlasst eine Umschreibung der Datei `/boot/config.txt` entweder mit der Datei `/boot/config.txt.lcd` (Anlage 5.3.1.1) für den LCD-Display oder `/boot/config.txt.hdmi` (Anlage 5.3.1.2) für den HDMI-Ausgang. Der Raspberry Pi startet hierbei neu. Alle nicht gespeicherten Einstellungen sollten vor Ausführen des Skriptes gespeichert werden, damit kein Datenverlust entsteht.

Der Raspberry Pi bootet nach jedem ordnungsgemäßen Herunterfahren oder Neustarten auf den LCD-Bildschirm. Dies verhindert, dass der Raspberry Pi in den HDMI-Ausgang bootet obwohl kein HDMI-Gerät angeschlossen ist. Dazu wurde das Skript *hdmi* in Anlage 5.3.2.1 im Verzeichnis `/home/rp3/` erstellt. Anschließend wurde das Skript mit dem folgenden Befehl ausführbar gemacht:

```
sudo chmod +x hdmi
```

Das Skript überschreibt die `/boot/config.txt` Datei mit der Datei `/boot/config.txt.lcd`.

Damit das Skript beim Herunterfahren ausgeführt wird, muss ein Service erstellt werden. Dazu erstellt man den Service (Anlage 5.3.2.2) in dem Unterverzeichnis `/etc/systemd/system` mit dem Namen *hdmi.service*. Mit den folgenden Befehlen lässt sich der Service installieren und aktivieren:

```
sudo systemctl enable hdmi
sudo systemctl restart hdmi
```

Damit der Service nicht nach jedem Neustarten erneut aktiviert werden muss, wird das Skript in Anlage 5.3.2.3 unter dem Namen *startHDMIService* im Verzeichnis `/etc/` erstellt. Anschließend wird das Skript ausführbar gemacht. Danach wird das Skript in der Datei `/etc/rc.local` ausgeführt, dazu muss folgendes in die Datei am Ende, jedoch vor `exit 0`, eingefügt werden:

```
...
/etc/startHDMIService
exit 0
```

Nach einem Neustart kann mit dem Befehl `sudo systemctl is-active hdmi` überprüft werden, ob der HDMI-Service aktiviert wurde.

Hinweis: Falls der Raspberry Pi nicht ordnungsgemäß heruntergefahren wurde und der letztbenutzte Videoausgang HDMI war, wird er auch im nächsten Bootvorgang in den HDMI-Ausgang booten.

### 2.2.3 Audiosoftware

Damit der Multiplexgenerator die Audiosignale verarbeiten kann, ist eine spezielle Software notwendig, die das Entgegennehmen, Weiterreichen und Bearbeiten der Signale ermöglicht.

#### 2.2.3.1 ALSA

Die Advanced Linux Sound Architecture ist eine freie Soundarchitektur für Linux und der grundlegende Baustein des Soundsystems. ALSA bietet unter anderem bestimmte Voreinstellungen, mit denen die meisten Soundkarten erkannt werden können.

#### 2.2.3.2 JACK

JACK ist ein Audioserver für Linux und andere unixoide Betriebssysteme. Als Kernfeatures bietet er niedrige Latenzzeiten und Synchronität aller mit ihm verbundenen Programme. Für die Echtzeitfähigkeit wird der Realtime-Modus des Betriebssystems vorausgesetzt. Verschiedene Programme lassen sich mittels JACK beliebig miteinander verbinden. Auch ist beispielsweise der Abgriff des Ausgangssignals einer Anwendung und das Benutzen als Eingangssignal in mehreren anderen Programmen möglich. Das Programm QjackCtl stellt eine grafische Benutzeroberfläche für JACK bereit, mit der Einstellungen oder auch Audioverbindung einfach geändert werden können. Die Installation von JACK mit der dazugehörigen Grafikoberfläche wird mit dem folgenden Befehl ausgeführt:

```
sudo apt-get install jackd qjackctl libjack-jackd2-dev jackd2
```

#### 2.2.3.3 VLC-Player

Der VLC Media Player ist ein Open-Source Mediaplayer. Er ist plattformunabhängig und bietet eine Vielzahl an Unterstützungen für Dateiformate an. Er wird verwendet, um MP3-Dateien oder Internetstreams, die MP3-codiert sind, zu decodieren und anschließend als zusätzliches Programm in den Multiplex aufnehmen zu können. Die Ausgabe des VLC kann direkt auf den JACK-Audioserver erfolgen, wozu das VLC-Jack-Plugin installiert werden muss. Der VLC-Player mit Plugin für den JACK-Server wird wie folgt installiert:

```
sudo apt-get install vlc  
sudo apt-get install vlc-plugin-jack
```

#### 2.2.3.4 *Ebumeter*

Das Paket „ebumeter“ (Bild 12) ermöglicht das Messen der Lautheit von Audiosignalen nach der EBU-Empfehlung R 128 [6]. Es bietet eine Momentan-, Kurz- und über die Gesamtzeit integrierte Lautheitsberechnung. Darüber hinaus ist die Bestimmung der Dynamik der Lautheit möglich, wodurch über die Notwendigkeit einer Komprimierung entschieden werden kann. Das Ebumeter wird wie folgt installiert:

```
sudo apt-get install ebumeter
```

### 2.2.4 DAB-Softwarekomponenten

In diesem Abschnitt werden die Softwarekomponenten beschrieben, die für die Verwendung der ODR-mmbTools erforderlich sind.

#### 2.2.4.1 *FIFO*

FIFO ist eine Named Pipe, die von zwei Prozessen gleichzeitig zum Schreiben und Lesen genutzt wird. Sowohl der Audioencoder und PAD-Encoder als auch der DAB-Multiplexer und DABlin kommunizieren über FIFO-Dateien. Das Programm muss nicht zusätzlich installiert werden, da es bereits unter Raspbian vorhanden ist.

#### 2.2.4.2 *Screen*

Screen ist ein Programm, das es ermöglicht, in einer Shell mehrere Programme gleichzeitig auszuführen, indem sie in verschiedenen virtuellen Shells, den sogenannten Screens, ausgeführt werden. Screen wird mit dem folgenden Befehl installiert:

```
sudo apt-get install screen
```

#### 2.2.4.3 *ZeroMQ*

Der Transport der Daten zwischen Multiplexer und Audioencoder sowie zwischen Multiplexer und EasyDAB-Board erfolgt mit ZeroMQ ([www.zeromq.org](http://www.zeromq.org)) statt per veraltetem „FIFO/named Pipe“-Prinzip. Diese API-Bibliothek implementiert eine hochleistungsfähige, asynchrone Datenübertragung, die TCP als Transportprotokoll nutzt. Auch verfügt der Dienst über eine Nachrichtenwarteschlange und richtet sich an parallele und verteilte Anwendungen, wie in diesem Fall. ZeroMQ wird mit folgendem Befehl installiert:

```
sudo apt-get -y install libzmq3-dev libzmq3
```

#### 2.2.4.4 SSMTP

SSMTP (simples SMTP) ist ein MTA (Mail Transfer Agent). Er wird verwendet, um Emails vom Raspberry Pi aus zu versenden. Das Programm wird verwendet, um Informationen über den Zustand des Systems zu versenden. SSMTP lässt sich mit folgendem Befehl installieren:

```
sudo apt-get install ssmtp mailutils
```

#### 2.2.4.5 Supervisor

Supervisor ist ein Monitoring-Programm. Es lassen sich mehrere Prozesse gleichzeitig starten und über eine Weboberfläche überwachen bzw. steuern. Supervisor bietet außerdem die Möglichkeit des Versendens von Status-Mails. Im Gegensatz zur Remoteüberwachung mit Screen muss nicht per SSH auf den Raspberry Pi zugegriffen werden, sondern man ruft in einem beliebigen Browser die IP-Adresse des Pi mit Angabe des Ports des supervisor-Programms auf. Nachfolgend wird der Befehl zur Installation von Supervisor gezeigt:

```
sudo apt-get install supervisor
```

#### 2.2.4.6 Superlance

Superlance wird benötigt, um Emails aus dem Programm Supervisor heraus zu versenden. Es stellt den Befehl *crashmail* zur Verfügung. Mit Angabe der Email-Adresse des Empfängers lassen sich dann Mails versenden, wenn ein Prozess in Supervisor ausgefallen ist. Superlance wird mit dem nachfolgenden Befehl installiert:

```
sudo pip install superlance
```

#### 2.2.4.7 UHD-Treiber

Der USRP Hardware Driver (UHD) ist ein Treiber, der von der Firma Ettus Research bereitgestellt wird. Er wird verwendet um mit dem ODR-Modulator ein Universal Software Radio Peripheral (USRP) zu betreiben. Um die UHD-Treiber zu installieren sollte zuerst sichergestellt sein, dass die folgenden Voraussetzungen installiert worden sind:

```
sudo apt-get install libboost-all-dev libusb-1.0-0-dev python-mako  
doxygen python-docutils cmake build-essential
```

Danach können die UHD-Treiber installiert werden. Dies geschieht mit den folgenden Befehlen:

```
git clone git://github.com/EttusResearch/uhd.git
cd <uhd-repo-path>/host
mkdir build
cd build
cmake ../
make
make test
sudo make install
sudo ldconfig
```

Anschließend müssen mit dem folgenden Befehl Firmware-Images installiert werden:

```
sudo /usr/local/lib/uhd/utils/uhd_images_downloader.py
```

Die korrekte Funktionsweise der UHD-Treiber kann nach Anschließen eines USRP-Gerätes mit dem Befehl `sudo uhd_find_devices` überprüft werden.

Falls der ODR-Modulator auf dem Raspberry Pi nicht zum Einsatz kommt, kann die Installation der UHD-Treiber auch ignoriert werden.

## 2.2.5 ODR-mmbTools

In diesem Abschnitt wird die Software beschrieben, die zum Erzeugen eines DAB-Signals erforderlich ist. Dabei werden die ODR-mmbTools verwendet. Sie sind eine Weiterentwicklung der Open-Source-Software CRC-DABMUX des CRC (Communications Research Centre Canada) durch Mathias Brändli. Außerdem wird die Installation der Software auf einem Raspberry Pi beschrieben.

### 2.2.5.1 Quellcodierung

Damit die Audiodaten im Multiplexgenerator verarbeitet werden können, sieht der DAB-Standard zwei verschiedene Verfahren zur Quellcodierungen der Audioströme vor.

Dies sind:

- MPEG 1 Layer 2 (MUSICAM) für DAB
- MPEG 4 Part 3 (AAC) für DAB+

Der ODR-Audioencoder ist ein DAB- und DAB+-Encoder. Der DAB-Encoder verwendet die toolame-Bibliothek, wohingegen der DAB+-Encoder die fdk-aac Bibliothek des Fraunhofer-Instituts für Integrierte Schaltungen (IIS) verwendet. In diesem Projekt wird hauptsächlich der

---

DAB+-Encoder verwendet. Der Audioencoder kann als Eingang eine Audiodatei und alle Eingänge, die libVLC und ALSA unterstützen, kodieren. Somit ist die Kodierung, eines Internetstreams oder eines Audioausgangs des JACK-Audioservers möglich. Der Audioencoder kodiert hierbei entweder einen ZeroMQ-Ausgang oder eine Datei. Beim DAB+-Encoder liegt die einstellbare Ausgangsbitrate zwischen 8 und 192 kbit/s. Der Audioencoder kann außerdem die im DAB-Standard festgelegten PAD-Bytes, die mit dem ODR-PAD-Encoder erzeugt wurden, hinzufügen.

Um den ODR-Audioencoder zu installieren, muss sichergestellt werden, dass die folgenden Voraussetzungen installiert wurden:

- Ein C++11 Compiler
- FDK-AAC Bibliothek
- ZeroMQ 4.04 oder höher
- JACK-Audioserver
- ALSA-Bibliotheken(libasound2)
- Libvlc- und vlc-Bibliotheken

Die FDK-AAC Bibliothek wird wie folgt installiert:

```
git clone https://github.com/Opendigitalradio/fdk-aac.git
pushd fdk-aac
./bootstrap
./configure
make
sudo make install
sudo ldconfig
popd
```

Die ALSA-, libvlc- und vlc- Bibliotheken werden wie folgt installiert:

```
sudo apt-get -y install libasound2 libasound2-dev libvlc-dev vlc-nox
```

Danach wird der Audioencoder installiert. Die Alsa-, Jack- und VLC-Eingänge können mit dem *-enable* Befehl in der Zeile mit *./configure* aktiviert werden:

```
git clone https://github.com/Opendigitalradio/ODR-AudioEnc.git
pushd ODR-AudioEnc
./bootstrap
./configure --enable-alsa --enable-jack --enable-vlc
make
sudo make install
popd
```

### 2.2.5.2 *Programmbegleitende Daten*

Der Audioencoder ermöglicht es, die zum Radioprogramm gehörenden Zusatzinformationen, wie Dynamic Label, Dynamic Label Plus und Slideshows, in Form von PAD (Programm Associated Data) zu übertragen. Damit der Audioencoder diese Daten verarbeiten kann, werden sie vom PAD-Encoder kodiert. Die Kommunikation zwischen Audioencoder und PAD-Encoder erfolgt über FIFO-Dateien. Neben Textdateien kann auch Bildmaterial übertragen werden.

Zur erfolgreichen Installation müssen die folgenden Voraussetzungen installiert werden:

- Ein C++ 11 Compiler
- ImageMagick MagickWand

Die ImageMagick-Bibliothek wird wie folgt installiert:

```
sudo apt-get install libmagickwand-dev
```

Der PAD-Encoder wird mit den folgenden Befehlen installiert:

```
git clone https://github.com/Opendigitalradio/ODR-PadEnc.git
pushd ODR-PadEnc
./bootstrap
./configure --enable-jack --enable-vlc
make
sudo make install
popd
```

### 2.2.5.3 *DAB-Multiplex*

Um einen DAB-Multiplex zu generieren, wird das Tool ODR-DabMux verwendet. Aus Audio- und Datenströmen wird ein nach ETSI EN 300 401 DAB-konformes ETI-Signal erzeugt. Der Multiplexer kann die Eingangsströme aus einer Datei, einer UDP-Verbindung oder einer ZeroMQ-Übertragungen einlesen. Der dann generierte Multiplex kann in Form eines ETI-Signals, als Datei, EDI (in UDP eingekapseltes ETI) oder ZeroMQ-Stream ausgegeben werden. Der Multiplexer ermöglicht es, Einstellungen für den Fehlerschutz vorzunehmen, Zeitstempel für ein Gleichwellennetz einzufügen, und er generiert die FIC-Daten (Fast Information Channel). Die Konfiguration des Multiplexer erfolgt über die Kommandozeile oder eine Konfigurationsdatei.

Zur Installation müssen die folgenden Voraussetzungen installiert sein:

- Ein C++11 Compiler
- Boost 1.48-Bibliothek oder höher
- ZeroMQ 4 oder höher
- cURL optional für EDI-Ausgang

Die Voraussetzungen werden mit folgendem Befehl installiert:

```
sudo apt-get install build-essential libzmq3-dev libzmq3 automake  
libtool libboost-all-dev libcurl4-openssl-dev
```

Die Installation des ODR-DabMux erfolgt dann folgendermaßen:

```
git clone https://github.com/Opendigitalradio/ODR-DabMux.git  
pushd ODR-DabMux  
./bootstrap.sh  
./configure --with-boost-libdir=/usr/lib/arm-linux-gnueabi/hf  
make  
sudo make install  
popd
```

#### 2.2.5.4 DAB-Modulator

Um I/Q-Basisbanddaten zu erzeugen, wird der ODR-DabMod verwendet. Er ist ein DAB-Modulator, der ein ETI-Signal über einen Stream oder eine Datei einliest. Die generierten I/Q-Samples können dann als Datei abgespeichert oder anhand der integrierten USRP-Hardware-Treiber (UHD) an ein USRP-Gerät gesendet werden. Zur weiteren Verarbeitung können die I/Q-Samples außerdem über ZeroMQ weiter übertragen werden. In der Konfigurationsdatei werden außerdem wichtige HF-Parameter, wie die Sendefrequenz, der Gain für die Ausgangsleistung des Senders und der DAB-Mode, eingestellt. Der Modulator unterstützt auch Zeitstempel zur Übertragung in Gleichwellennetzen.

Die folgenden Voraussetzungen müssen vor der Installation erfüllt sein:

- Ein C++11 Compiler
- Boost 1.54-Bibliothek oder höher
- FFTW 3.x-Bibliothek
- Optional UHD-Treiber für USRP
- Optional ZeroMQ

Die FFTW-Bibliothek wird wie folgt installiert:

```
sudo apt-get install libfftw3-dev
```

Die Installation der Boost-Library wird nicht aufgeführt, da diese bereits in Kapitel 2.2.5.3 beschrieben wurde.

Zur Installation des ODR-DabMod werden folgende Befehle ausgeführt:

```
git clone https://github.com/Opendigitalradio/ODR-DabMod.git
pushd ODR-DabMod
./bootstrap.sh
CFLAGS="-O3" CXXFLAGS="-O3" ./configure --disable-native --with-
    boost-libdir=/usr/lib/arm-linux-gnueabi/f
make
sudo make install
popd
```

### 2.2.5.5 DABlin

Das Programm DABlin ist ein DAB/DAB+-Player, der eine Liveübertragung oder eine gespeicherte Ensemble-Datei abspielen kann. Er wird verwendet, um das erzeugte DAB-Multiplex-Signal zu überprüfen. Zur Installation des DABlin-Players ist sicherzustellen, dass die folgenden Voraussetzungen installiert worden sind:

```
sudo apt-get install git gcc g++ cmake
sudo apt-get install libmpg123-dev libfaad-dev libsdl2-dev libgtkmm-
    3.0-dev
```

Anschließend kann der DABlin-Player wie folgt installiert werden:

```
git clone https://github.com/Opendigitalradio/dablin.git
cd dablin
mkdir build
cd build
cmake ..
make
sudo make install
cd
```

### 2.2.5.6 Skript zur vereinfachten Installation

Um die Installation der ODR-mmbTools zu vereinfachen, kann das in der Anlage 5.1 befindliche Skript verwendet werden. Es muss nur noch ausführbar gemacht werden und kann dann gestartet werden. Bei Verwendung des USRP muss das Skript abgeändert werden oder die UHD-Treiber und der ODR-DabMod werden nachträglich per Hand installiert. Das Skript installiert alle benötigten Voraussetzungen.

```
chmod +x raspdab.sh
./raspdab.sh
```

### 2.3 SOFTWAREKONZEPT

Die Verarbeitung zu einem standardkonformen DAB-Signal geschieht in verschiedenen Programmen, die jeweils einen Teil der Datenverarbeitung übernehmen. Sowohl die vom VLC-Player bereitgestellten MP3-Playlisten als auch die physischen Audioeingangskanäle stehen dem JACK-Audioserver als Eingänge bereit. Diese Daten werden vom JACK in den Audioencoder weitergeleitet. Der Audioencoder kann direkt Internetstreams abspielen und extrahiert die ICY-Infos in eine Datei, die dann vom PAD-Encoder so kodiert werden, dass der Audioencoder diese in den MPEG-Stream einfügen kann. Außerdem besteht die Möglichkeit, eine Slideshow mit dem PAD-Encoder zu verarbeiten, sodass diese durch den Audioencoder in den MPEG-Stream neben den ICY-Infos eingefügt werden kann. Der MPEG-Stream wird mittels ZeroMQ an den Multiplexer transportiert. Dieser erzeugt dann ein ETI-Signal. Das ETI-Signal kann über Ethernet an den EasyDAB weitergesendet werden, der dann die I/Q-Samples generiert oder es mittels ZMQ-Transport an den Modulator sendet. Außerdem kann das ETI-Signal in Form von EDI an einen externen Ethernetanschluss gesendet werden. Der Modulator generiert aus dem ETI-Signal dann die Basisband-I/Q-Samples und sendet diese im Nachgang z.B. per USB an einen USRP.

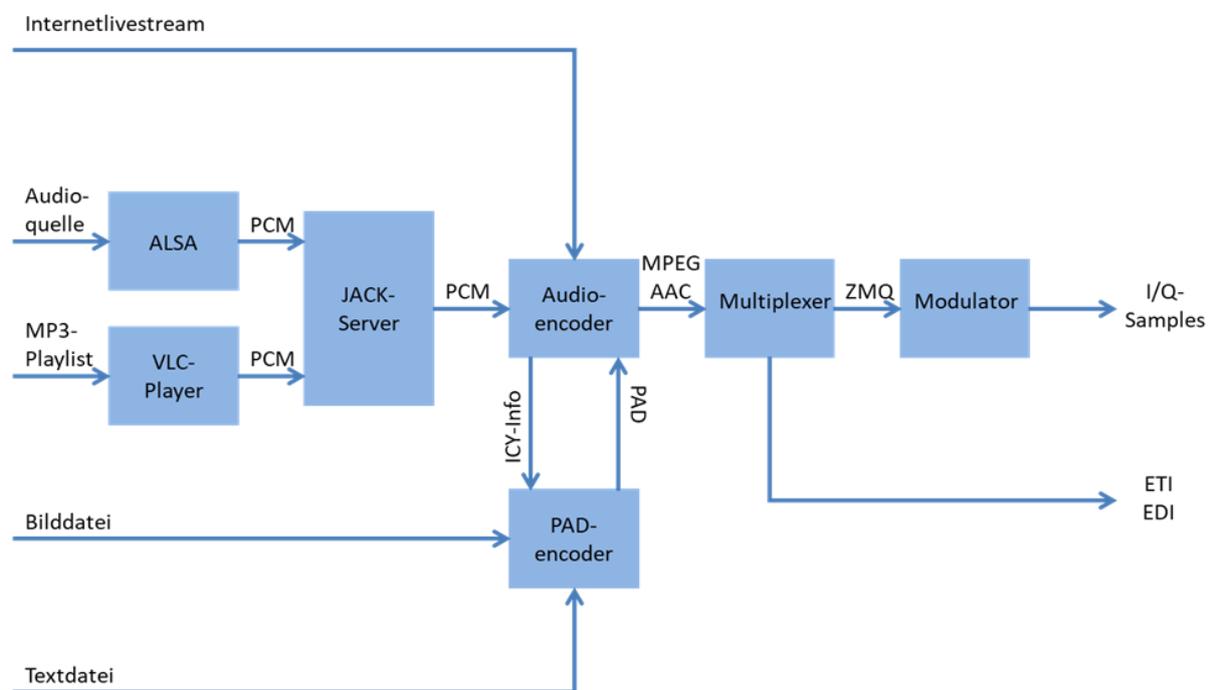


Abbildung 4: Softwarekonzept DAB-Toolbox-Sender

### 3 AUFBAU DES DAB-TOOLBOX-SENDERS

Der Aufbau des DAB-Toolbox-Senders erfolgte in einem Aluminiumkoffer der Größe 320x230x150 mm (BxTxH). Im Koffer befinden sich der Raspberry Pi mit aufgestecktem S.USV-Board, das EasyDAB-Board, der Audioverstärker sowie der Netzwerkrouter (Abbildung 5).

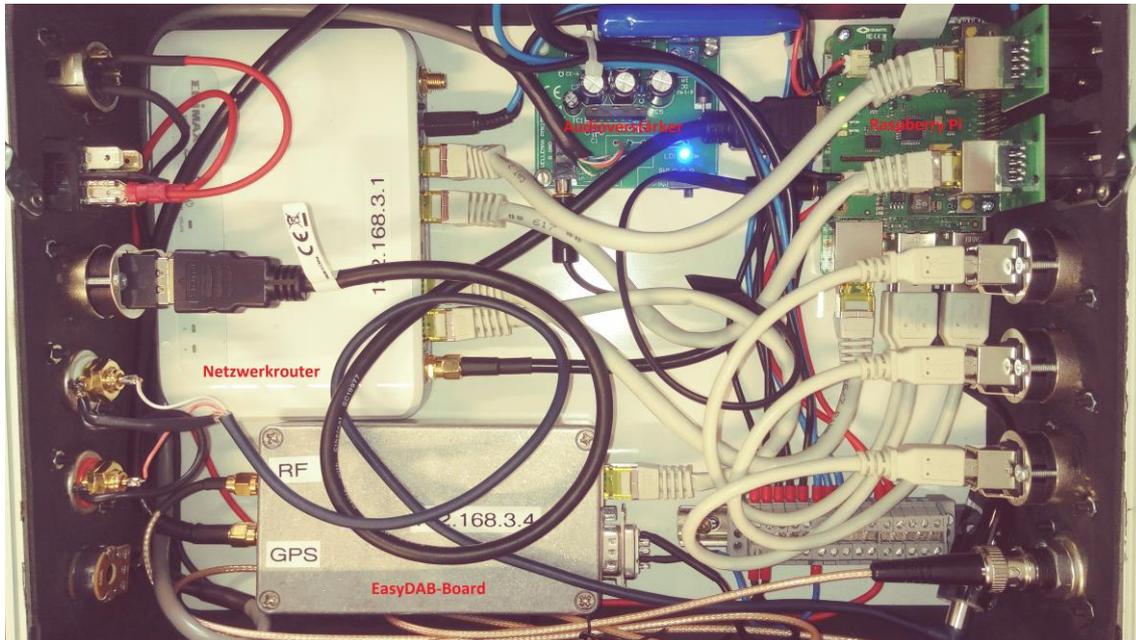


Abbildung 5: Innenaufbau DAB-Toolbox-Sender

Auf der rechten Seite (Abbildung 6) des Koffers befinden sich folgende Schnittstellen: 12V-Spannungseingang, Ein-/Ausshalter, ein HDMI-Anschluss für den Anschluss eines externen HDMI-Geräts, zwei Cinchbuchsen zur Einspeisung eines Stereoeingangssignals, ein Lautstärkereglер für die eingebauten Lautsprecher und ein 3,5 mm Klinkeausgang, mit dem sich beispielsweise ein Kopfhörer anschließen lässt. Der Klinkeausgang ist der Ausgang der USB-Soundkarte.

Zur Überwachung der 12 V Eingangsspannung befindet sich außerdem eine LED. Falls die Eingangsspannung ausfällt, erlischt auch die Kontroll-LED.



Abbildung 6: Rechte Seite Toolbox-Sender

Auf der linken Seite (Abbildung 7) befinden sich zwei RJ45-Ethernetanschlüsse, der Anschluss für das Internet über DHCP und der Anschluss für einen externen EDI-Strom. Der Anschluss für DHCP ist der WAN-Anschluss am Router. Außerdem befinden sich drei USB 2.0-Anschlüsse auf der linken Seite zum Anschließen von externen USB-Geräten sowie ein BNC-Anschluss zum Abgreifen der Sendeleistung bzw. für den Anschluss einer Antenne.



Abbildung 7: linke Seite Toolbox-Sender

Auf der oberen Seite (Abbildung 8) befinden sich zwei SMA-Anschlüsse jeweils für die WLAN-Antenne und GPS-Antenne.



Abbildung 8: Obere Seite Toolbox-Sender

Auf der Vorderseite (Abbildung 9) ist das 7 Zoll Touchdisplay installiert sowie die zwei Lautsprecher.

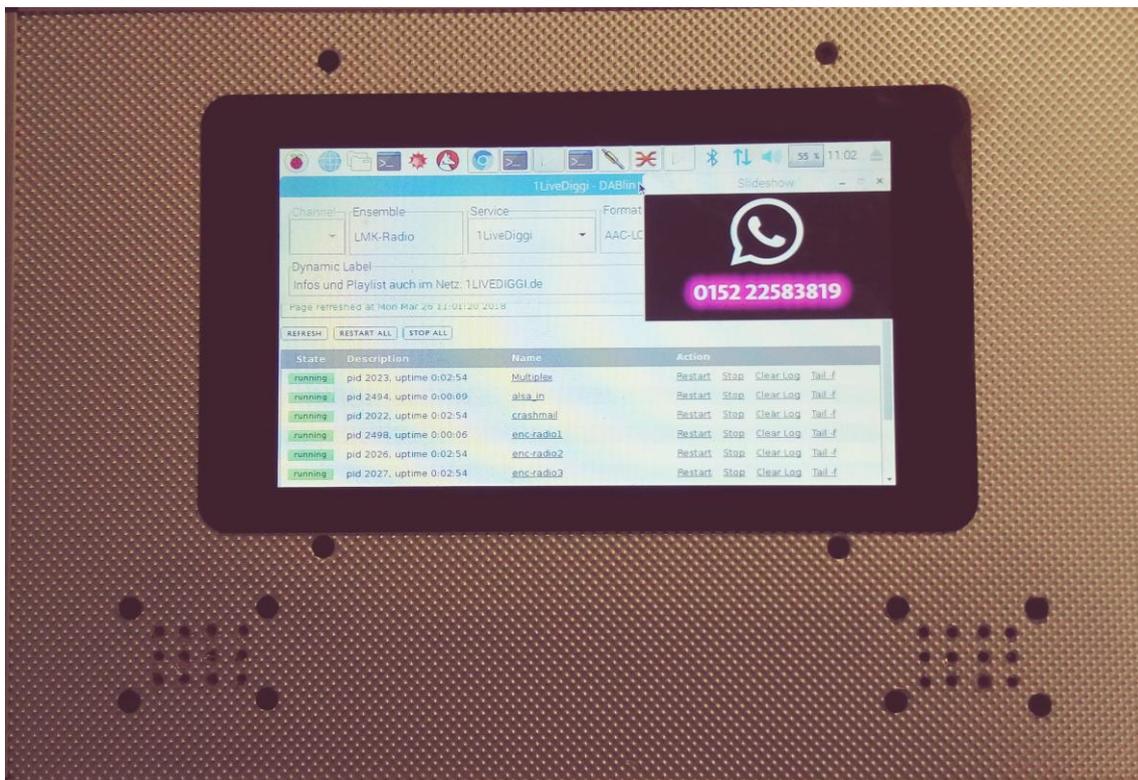


Abbildung 9: Vorderseite Toolbox-Sender

## 4 BEDIENUNG

In diesem Kapitel wird die wesentliche Bedienung und Einrichtung des Toolbox-Senders beschrieben.

### 4.1 BENUTZEREINSTELLUNGEN

Das Login zur Benutzung des Raspbian-Betriebssystems und der DAB-Softwarekomponenten lautet:

User	rp3
Passwort	rp3user

*Tabelle 1: Anmeldedaten Raspbian*

Außerdem ist der Standard-Benutzer von Raspbian Jessie „pi“ mit dem Passwort „raspberrry“ eingerichtet. Bei Eingabe des Passworts muss darauf geachtet werden, welche Ländereinstellung für die Tastatur gültig ist; ggf. muss die Taste „z“ zur Übergabe des Buchstaben „y“ gedrückt werden (eingegeben wird also als Passwort „raspberrrz“).

### 4.2 EINRICHTUNG DES MAILKONTOS

Damit Emails vom Raspberry Pi versendet werden können, müssen zuerst gewisse Einstellungen getroffen werden. Zuerst muss eine Email-Adresse angegeben werden, von der später die Emails gesendet werden. Dazu wird die Datei `/etc/ssmtp/ssmtp.conf` folgendermaßen bearbeitet:

```
root=rp3.taschensender.lmkrlp@gmail.com
mailhub=smtp.gmail.com:587
hostname=localhost
AuthUser=rp3.taschensender.lmkrlp
AuthPass=rp3user2018
FromLineOverride=YES
UseSTARTTLS=YES
```

Die Datei muss mit den eigenen Angaben abgeändert werden. Ein Sicherheitshinweis ist, dass das Passwort in Klartext in der Datei gespeichert wird. Also sollte dafür gesorgt werden, dass niemand anderes darauf zugreifen kann.

Nach der Einstellung des Email-Servers muss nun noch angegeben werden, welcher Benutzer Emails versenden darf. Dafür wird die Datei `/etc/ssmtp/revaliases` bearbeitet. Hier wird nach dem root-Eintrag folgendes angegeben:

```
Rp3: rp3.taschensender.lmkrlp gmail.com:smtp.gmail.com:587
```

Dabei müssen die Angaben wieder durch die Eigenen ersetzt werden.

Eine Email lässt sich dann mit dem folgenden Befehl versenden:

```
echo "Hello world email body" | mail -s "Test Subject"  
Empfänger@domain.com
```

#### 4.2.1 Prozessüberwachung

Um Emails zu versenden, wenn ein bestimmter Prozess beendet wurde, wird das folgende Skript verwendet:

```
#!/bin/bash  
while pgrep '$1' > /dev/null;  
do  
sleep 1;  
done  
echo "Prozess $1 wurde beendet" | mail -s "Prozess $1"  
Empfänger@domain.com
```

Das erstellte Skript wird unter dem Namen `mailTest.sh` abgespeichert. Es wird danach wie folgt aufgerufen:

```
bash mailTest.sh [prozessname]
```

Als Übergabeparameter erhält es den Prozessnamen des Prozesses, der überwacht werden soll. Der Prozessname kann z.B. mit dem `top`-Befehl herausgefunden werden.

### 4.3 BEDIENUNG DER S.USV-SOFTWARE

Um Statusmeldungen des S.USV-Boards anzuzeigen, kann in dem folgenden Verzeichnis `/opt/susvd` der folgende Befehl ausgeführt werden:

```
sudo ./susv -status
```

Anstelle des Parameters `-status` kann auch der Parameter `-h` übergeben werden. Damit werden alle möglichen Parameterrufe angezeigt. Eine Übersicht befindet sich in der Anlage 5.2. Eine detaillierte Bedienungsanleitung findet sich unter [5].

#### 4.4 UMSCHALTUNG ZWISCHEN INTERNEM LCD UND EXTERNEM HDMI-ANSCHLUSSES

Der Raspberry Pi kann keine zwei Videosignale gleichzeitig bereitstellen. Daher muss mit dem Skript *display* (Anlage 5.3.1) zwischen den beiden Videoausgängen (LCD-Display oder HDMI-Ausgang ) mit folgendem Befehl umgeschaltet werden:

```
sudo bash display
```

Der Raspberry Pi bootet nach jedem ordnungsgemäßen Herunterfahren oder Neustarten auf den LCD-Bildschirm. Dies verhindert, dass der Raspberry Pi in den HDMI-Ausgang bootet obwohl kein HDMI-Gerät angeschlossen ist.

Nach einem Neustart kann mit dem Befehl

```
sudo systemctl is-active hdmi
```

überprüft werden, ob der HDMI-Service aktiviert wurde.

Hinweis: Falls der Raspberry Pi nicht ordnungsgemäß heruntergefahren wurde und der letztbenutzte Videoausgang HDMI war, wird er auch im nächsten Bootvorgang in den HDMI-Ausgang booten.

#### 4.5 NETZWERKEINSTELLUNGEN

Die Kommunikation zwischen dem Raspberry Pi und dem EasyDAB-Board erfolgt mittels Internetprotokoll (IP). Dazu besitzt der Toolbox-Sender sein eigenes Netzwerk. Der Aufbau dieses Netzwerks ist in Abbildung 10 dargestellt. Zur Verbindung mit dem Internet gibt es zwei Möglichkeiten: über ein LAN-Kabel oder über ein WLAN-Netzwerk. Außerdem kann ein EDI-Strom extern über einen zusätzlichen Ethernetanschluss abgegriffen werden oder ein ZMQ/TCP-Strom eingespeist werden.

Die Zuordnung der IP-Adressen zu den Netzwerkgeräten, die über die Routereinstellung festgelegt werden, ist in Tabelle 2 angegeben.

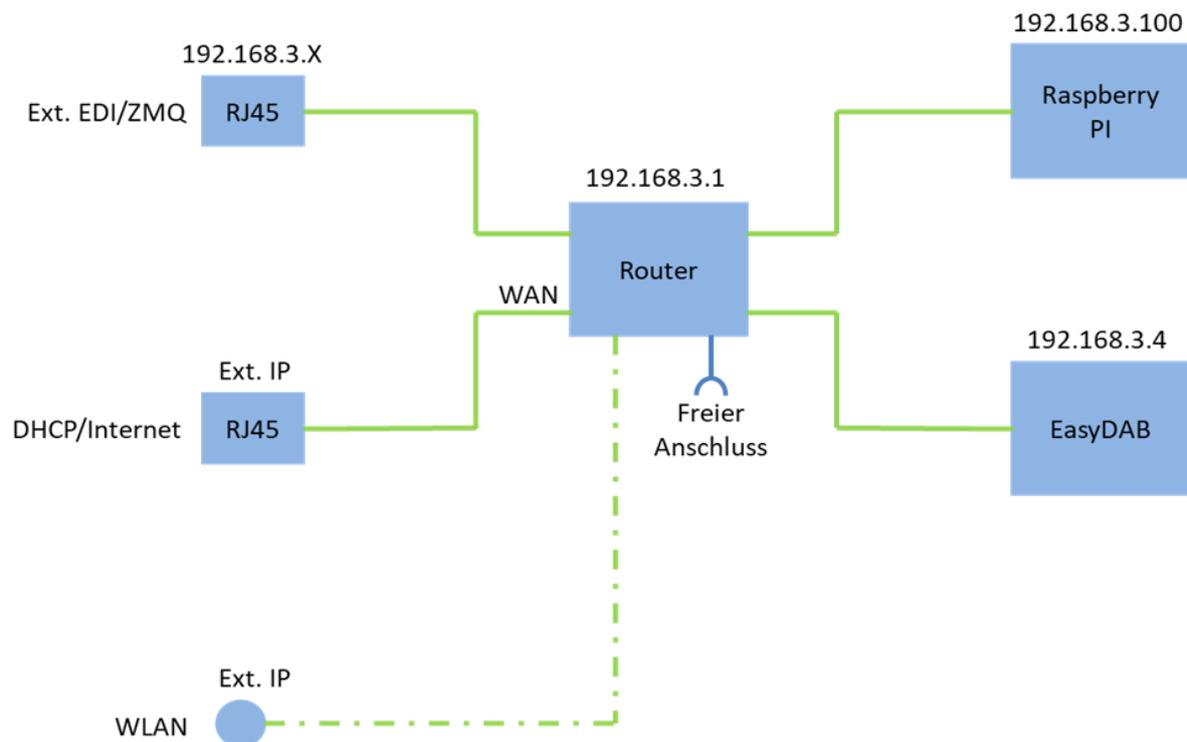


Abbildung 10: Netzwerkplan

Netzwerkgerät	IP-Adresse
EDIMAX-Router	192.168.3.1 (feste IP-Adresse)
Raspberry Pi	192.168.3.100 (Die Adresse wird über den DNS-Server des Routers vergeben)
EasyDABv2	192.168.3.4 (feste IP-Adresse)
Ext. EDI/ZMQ	192.168.3.x (die letzte Stelle der Adresse wird im ODR-DabMux vergeben)
DHCP/Internet	Ext. IP-Adresse
WLAN/Internet	Ext. IP-Adresse

Tabelle 2: Zuordnung der IP-Adressen zu den Geräten

#### 4.5.1 ERSTINSTALLATION DES ROUTERS

Zur Erstinstallation des Netzwerks über den Router oder beim Reset des Routers (Drücken der Reset-Taste auf der Rückseite des Routers für 10 Sekunden bis die Power-LED blinkt) muss der Router wie folgt (neu) konfiguriert werden:

Verbinden eines WLAN-fähigen PCs über das offene WLAN-Netz „edimax.setup“ und Aufruf der Seite <http://edimax.setup> in einem Browser. Folgen Sie dann den Anweisungen im Menu.

Als WLAN-Netzwerkname und Passwort wird anschließend eingerichtet:

WLAN-Netzwerkname (2,4 GHz)	ODR-Toolbox
Passwort	nichtwichtig

*Tabelle 3: Anmeldedaten Router-WLAN*

Der Router kann nach Neuansmeldung im WLAN-Netzwerk „ODR-Toolbox“ im Browser nach Aufruf der Seite <http://edimax.setup> über seine Bedienoberfläche konfiguriert werden. Die Anmeldedaten lauten:

User	Admin
Passwort	1234

*Tabelle 4: Anmeldedaten zur Router-Konfigurierung*

Die standardmäßige IP-Adresse des Routers ist 192.168.2.1. Diese muss geändert werden, da der EasyDAB v2.0 die feste IP-Adresse 192.168.3.4 hat.

Als neue IP-Adresse wird 192.168.3.1 vergeben (Menupunkt LAN -> LAN IP -> IP-Adresse -> 192.168.3.1). Die Änderung muss gespeichert und der Router neu gestartet werden.

Die Browser-Anmeldung muss dann erneut über die IP-Adresse 192.168.3.1 erfolgen.

#### 4.5.2 ROUTERKONFIGURATION IM BETRIEB

Der Router wird über seine Bedienoberfläche konfiguriert. Um auf die Weboberfläche zuzugreifen, wird im Browser die IP-Adresse 192.168.3.1 aufgerufen und die o.a. Anmeldedaten zur Administration eingegeben.

### 4.5.3 Internetverbindung über DHCP/LAN-Netzwerk

Um eine Internetverbindung über ein LAN-Netzwerk zu erzeugen, muss in die Bedienoberfläche des EDIMAX-Router gewechselt werden. Dazu wird in einem Browser die folgende IP-Adresse 192.168.3.1 geöffnet. Danach kann unter dem Reiter Internet/WAN-Setup (1), wie in **Fehler! Verweisquelle konnte nicht gefunden werden.** aufgeführt, der Connection-Mode (2) Dynamic IP ausgewählt werden. Falls die IP nicht über DHCP (3) bezogen werden soll, sondern eine statische IP-Adresse vergeben werden soll, muss der Punkt *Use the following IP adress* (4) angewählt werden. Anschließend müssen die Einstellungen gespeichert (5) und der Router neugestartet (6) werden.

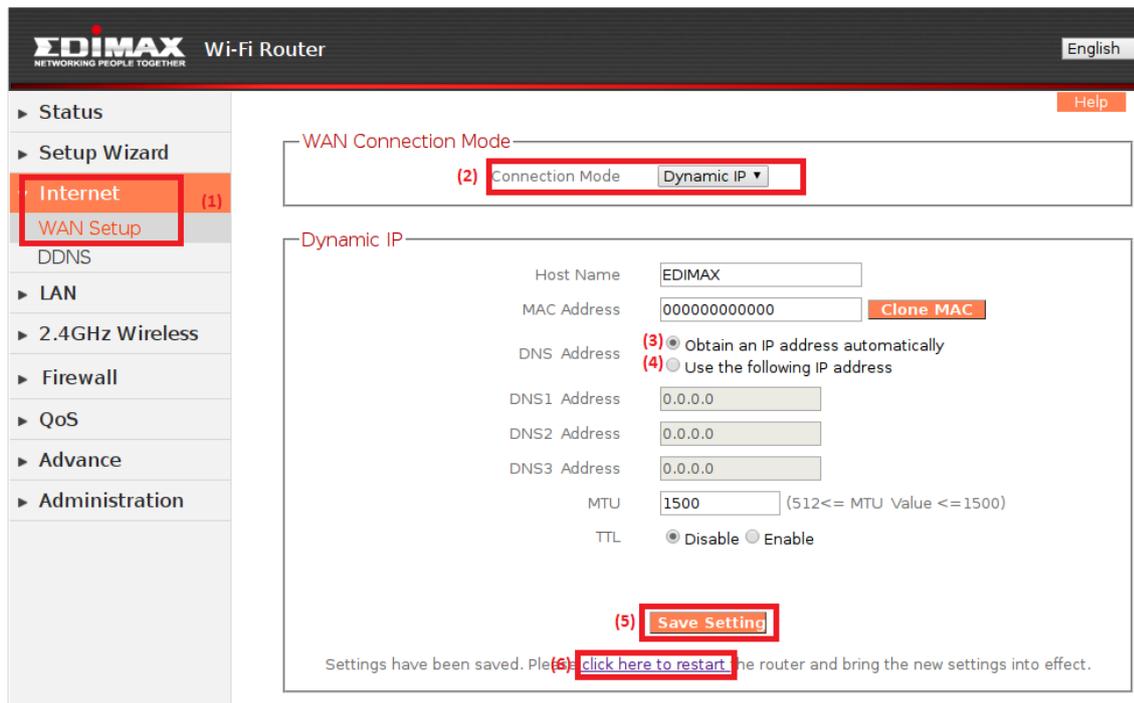


Abbildung 11: : DHCP/LAN-Einstellung EDIMAX

Falls vorher eine Verbindung über ein WLAN-Netzwerk vorhanden war, muss unter Connection Mode WSIP der Punkt für Disable ausgewählt werden. Siehe hierzu das Kapitel 4.5.4.

### 4.5.4 Internetverbindung über ein WLAN-Netzwerk

Soll eine Internetverbindung über ein bestehendes WLAN-Netzwerk einrichtet werden, muss im Browser zuerst wieder die Weboberfläche des EDIMAX-Routers aufgerufen werden. Dazu wird in einem Browser die IP-Adresse 192.168.3.1. aufgerufen. Anschließend wird der Reiter Internet/WAN Setup (1) aufgerufen (siehe Abbildung 12). Unter dem Modus Connection Mode (2) wird nun WSIP ausgewählt. Danach muss WSIP noch aktiviert werden, indem der

Punkt *enable* ausgewählt wird (3). Unter Punkt (4) kann manuell die ESSID des Netzwerkes eintragen werden. Alternativ kann der Button Select Site Survey (siehe Abbildung 13) geöffnet werden. Dort werden alle vorhanden WLAN-Netzwerke angezeigt und anschließend kann das gewünschte Netzwerk ausgewählt werden. Das Passwort für das WLAN-Netzwerk kann unter Punkt (5) eingetragen werden.

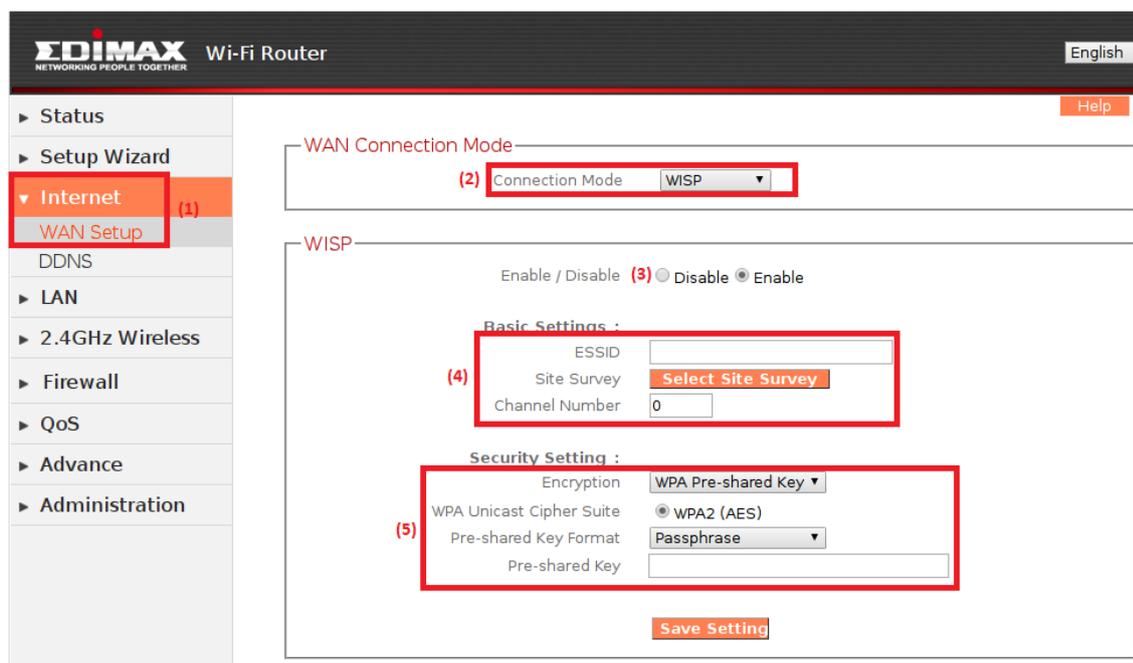


Abbildung 12 WSIP-Einstellung EDIMAX-Router

## Wireless Site Survey

Wireless site survey is a tool to scan for available wireless networks. In client mode, you can manually connect to any available access point or IBSS.

Select	ESSID	BSSID	Band	Channel	Type	Encryption	Signal
<input checked="" type="radio"/>	Raum 221b	d4:21:22:c8:68:db	(B+G+N)	1	AP	WPA2-PSK	76
<input type="radio"/>	Telekom_FON	d4:21:22:c8:68:dc	(B+G+N)	1	AP	no	76
<input type="radio"/>	TP-LINK_AP_B6F4	18:a6:f7:6b:b6:f4	(B+G+N)	8	AP	WPA-PSK/WPA2-PSK	60
<input type="radio"/>	media-gast	20:b3:99:5d:f5:49	(B+G)	13	AP	no	28
<input type="radio"/>	media-cert	20:b3:99:5d:f5:4a	(B+G)	13	AP	WPA2	28
<input type="radio"/>	FRITZBox LMK	44:4e:6d:04:84:c6	(B+G+N)	1	AP	WPA2-PSK	28
<input type="radio"/>	media	20:b3:99:5d:f5:48	(B+G+N)	13	AP	WPA2-PSK	20
<input type="radio"/>	FASBKVLU02	24:65:11:8a:ea:43	(B+G+N)	6	AP	WPA-PSK/WPA2-PSK	20
<input type="radio"/>	Raum118	00:1b:2f:d8:a3:0b	(B+G+N)	5	AP	WPA2-PSK	12
<input type="radio"/>	Vodafone Hotspot	86:5c:44:c7:6e:21	(B+G+N)	6	AP	no	4
<input type="radio"/>	media	20:b3:99:5e:04:58	(B+G+N)	5	AP	WPA2-PSK	4

Abbildung 13 Select Site Survey

Die Einstellungen müssen im Anschluss wieder gespeichert und der Router neugestartet werden.

## 4.6 AUDIOQUELLEN

Der Toolbox-Sender erlaubt es, verschiedene Audioquellen zu kodieren und diese in den Multiplex mit aufzunehmen. Hierbei kann eine externe Audioquelle, ein Internetlivestream oder eine MP3-Playlist verwendet werden. In diesem Kapitel wird erläutert, wie diese Audioquellen zur Verwendung eingestellt werden.

### 4.6.1 Audio-Routing

Bevor die Audiosignale auf den Audioencoder geleitet werden, muss der Jack-Audioserver gestartet werden. Mit dem Befehl `qjackctl -s` startet der Server. Durch den Button „Connect“ lassen sich dann die verfügbaren Ausgänge anzeigen. Mittels Drag and Drop oder mit dem Befehl „jack\_connect“ lassen sich nun die Ausgänge beliebig umleiten.

Weitere Befehlsinformationen zum JACK befinden sich in der Anlage 5.4.

### 4.6.2 Externer Audioeingangstrom

Der Toolbox-Sender verfügt über einen Stereoeingang mit zwei Cinch-Eingangsbuchsen. Damit lassen sich externe Audioeingangsströme aufnehmen, die direkt in den DAB-Multiplex eingespeist werden können. Dazu muss zuerst folgender Befehl ausgeführt werden, damit die Eingänge im Jack-Server bereitstehen:

```
alsa_in -d hw:1
```

hw:1 ist die Adresse der Soundkarte, die von ALSA vergeben wurde. Nun stehen die Eingänge der Soundkarte im Jack-Server unter dem Namen `alsa_in` bereit. Damit der Audiostrom auf den Audioencoder geleitet wird, muss dieser zuerst so gestartet werden, dass auch er im Jack-Server bereitsteht. Ein Beispielauf Ruf sieht folgendermaßen aus:

```
odr-audioenc -j odr -R -c 2 -r 48000 -o tcp://localhost:9001
```

Genauere Erläuterungen zum Audioencoder können im Kapitel 4.7.1 gefunden werden.

Zum Schluss müssen im Jack-Server noch die beiden erzeugten Inputs und Outputs miteinander verbunden werden.

---

### 4.6.3 MP3-Dateien/Playlists VLC-Player

Damit MP3-Dateien oder MP3-Playlists abgespielt werden können, kommt der VLC-Player zum Einsatz. Er besitzt eine grafische Oberfläche, mit dessen Hilfe sich der Player bedienen lässt. Der Player kann entweder durch die Kommandozeile mittels *vlc* gestartet werden oder durch das Startmenü unter *Sound & Video*. Nach dem Start des VLC besteht nun die Möglichkeit im Menü (Abbildung 14) unter *Media* durch den Eintrag *Open File...* eine Datei abzuspielen oder unter *Open Multiple Files...* mehrere Dateien abzuspielen.

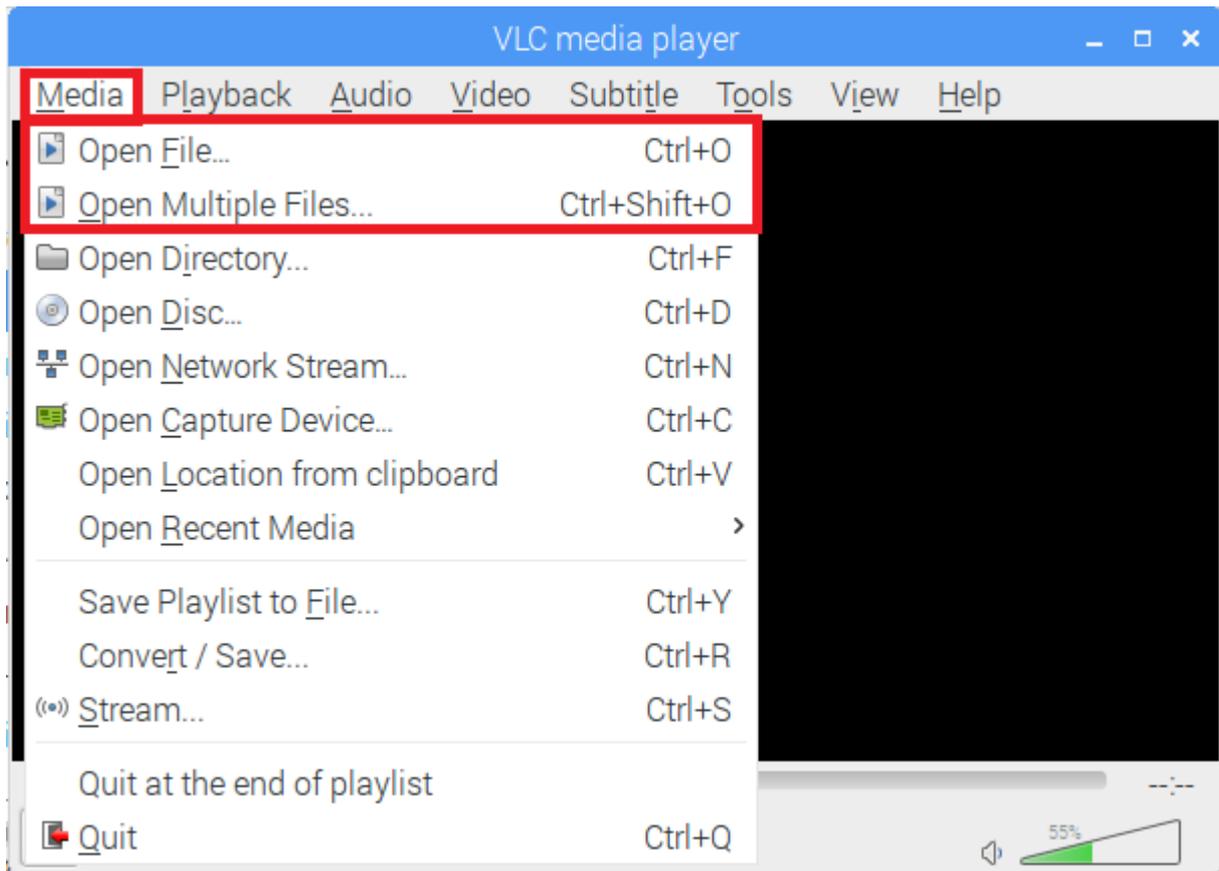


Abbildung 14: Öffnen von Dateien

Nun kann im neu geöffneten Fenster (Abbildung 15) zu den abzuspielenden Dateien mit *Add...* navigiert werden. Nach dem Hinzufügen kann mit dem Button *Play* die Dateien abgespielt werden.

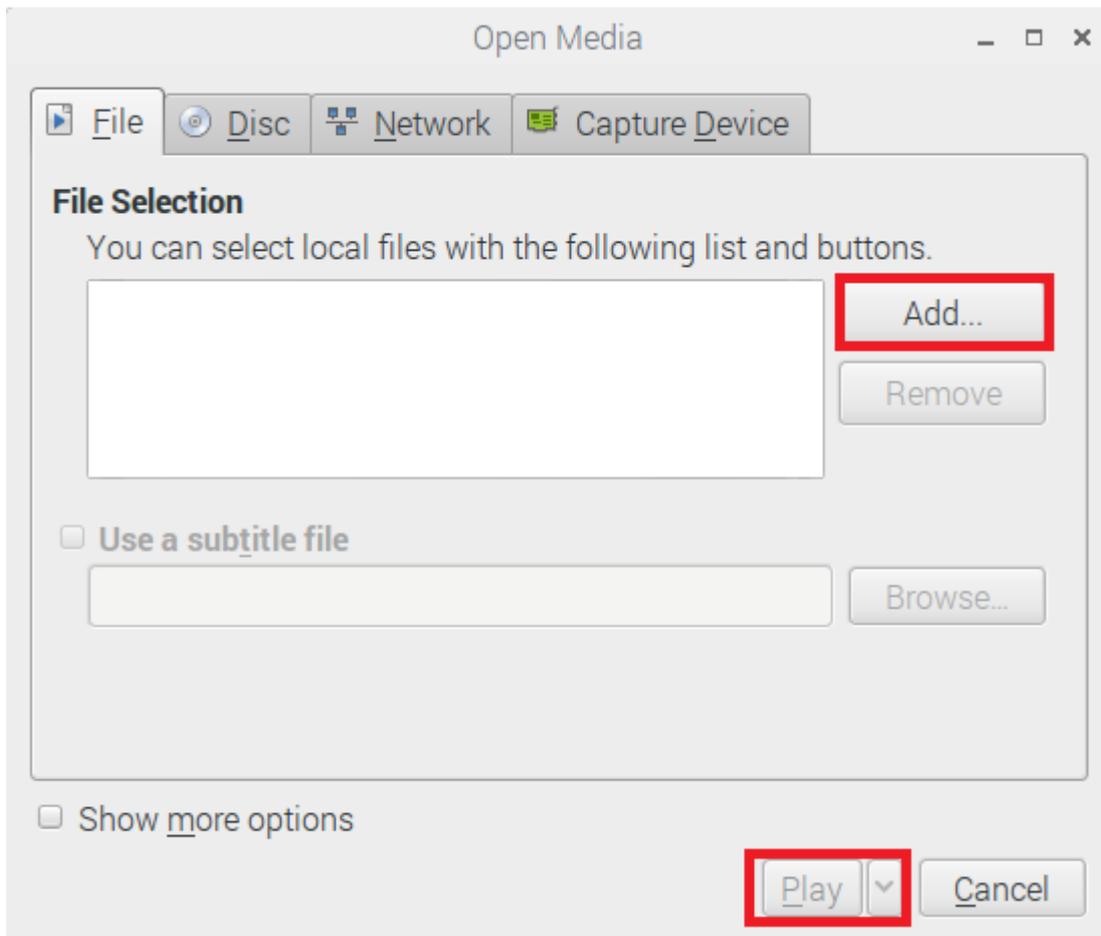


Abbildung 15: Dateien hinzufügen

Damit der VLC-Player im JACK-Server bereitsteht, muss das Menü Tools/Preferences geöffnet werden. Im neuen Fenster (Abbildung 16) unter (1) Audio wird das (2) Output Module gewählt. Im Anschluss müssen die Änderungen abgespeichert werden.

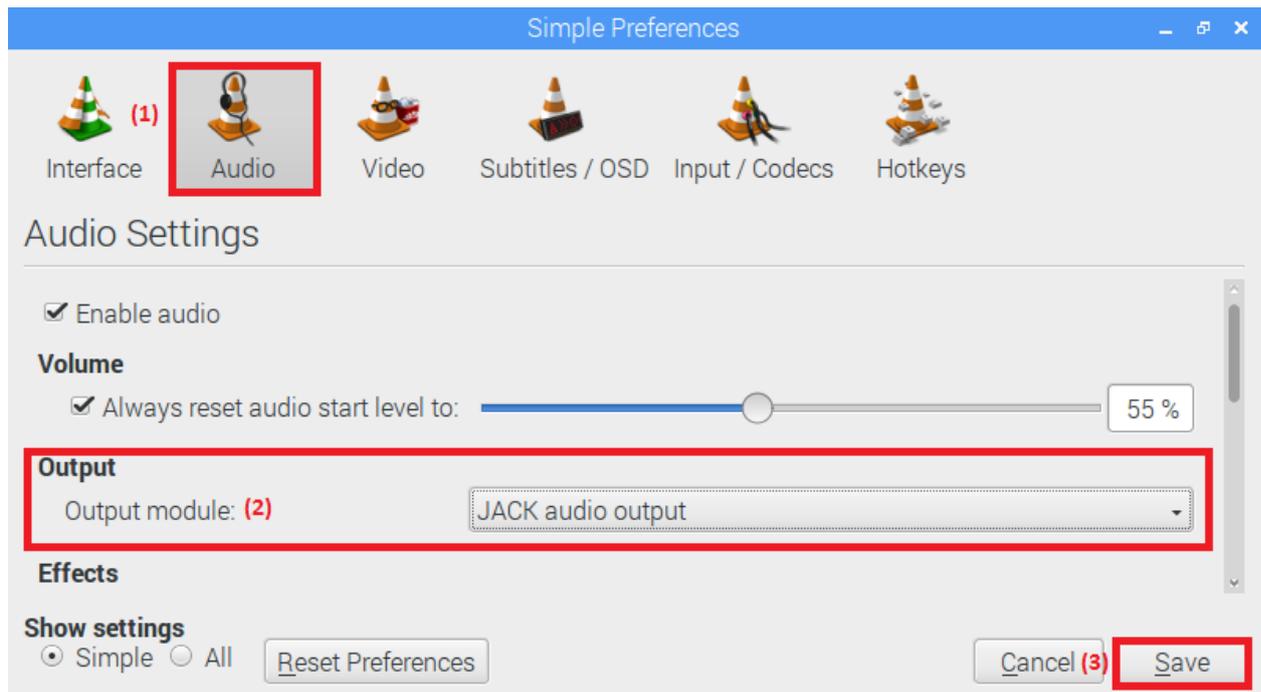


Abbildung 16: Output Module wählen

Da der VLC-Player nach dem Abspielen eines Titels die erstellte Verbindung im JACK-Server löscht, kann im Player angegeben werden, mit welchem JACK-Port sich der Player automatisch verbinden soll. Um dies einzustellen, wird im selbigen Fenster wie in Abbildung 16 der Punkt *Show settings/All* angewählt. Das Fenster verändert seine Darstellung wie in Abbildung 17. Zuerst wird unter *Audio/Output modules* (1) JACK ausgewählt. Unter Punkt (2) *Connect to clients matching* wird der Name des JACK-Ports angegeben, der zuvor im Audioencoder angegeben wurde. Ein Beispielaufwurf des Audioencoders sieht folgendermaßen aus:

```
odr-audioenc -j odr -D -b 96 -c 2 -r 48000 -P ./PAD/pad-  
SC_Radio_1.fifo -p 58 -o tcp://localhost:9001
```

Der Parameter *-j* gibt hierbei den Namen des JACK-Ports an. Genauere Erläuterungen zum Audioencoder finden sich im Kapitel 4.7.1.

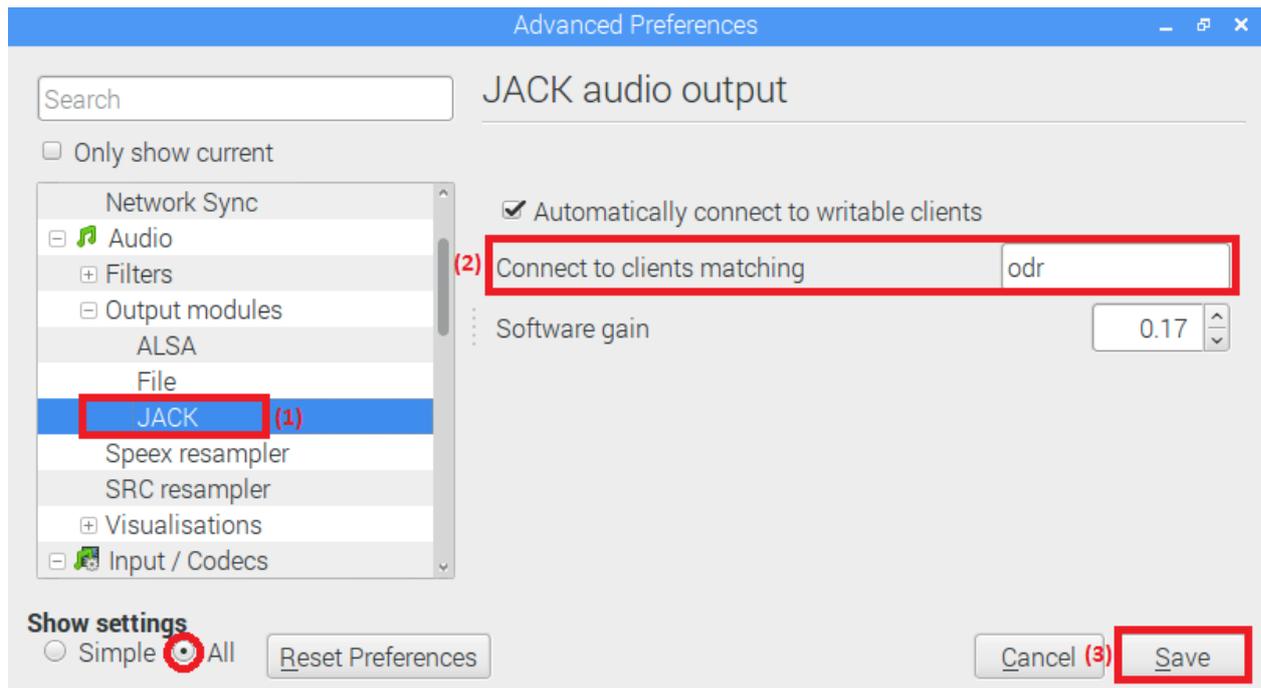


Abbildung 17: Automatisches Verbinden mit JACK-Port

#### 4.6.4 Internet-Livestreams

Der ODR-Audioencoder kann Internet-Livestreams direkt über die integrierte libVLC-Bibliothek abspielen. Der Ausgang kann dann mittels TCP weiter übertragen werden. Über den Audioencoder können auch die ICY-Infos aus dem Livestream in eine Textdatei geschrieben werden, die dann mit dem PAD-Encoder so kodiert werden können, dass sie der Audioencoder in den Audiostrom einfügen kann. Außerdem kann die Bitrate und die Samplerate eingestellt werden. Ein Beispielaufwurf des ODR-Audioencoder zum Abspielen eines Livestreams sieht folgendermaßen aus:

```
odr-audioenc -v http://streaming.brf.be/brf2-high.mp3 -R -b 128 -c 2  
-r 48000 -P ./PAD/pad-SC Radio_1.fifo -p 58 -o  
tcp://localhost:9001 --write-icy-text=./PAD/radio1_dyn-dl.txt
```

## 4.7 ODR-MMBTOOLS

In diesem Kapitel wird die Bedienung der ODR-mmbTools beschrieben. Um ein DAB-Signal zu erzeugen, müssen die Tools PAD-Encoder, Audioencoder und der DAB-Multiplexer gestartet werden. Falls eine I/Q-Basisbandmodulation gewünscht ist, muss zusätzlich der DAB-Modulator gestartet werden.

### 4.7.1 ODR-AudioEnc

Da der Multiplexer nur DAB-konforme Audiostreams entgegennimmt, wird vom Audioencoder der PCM-Stream in einen MP2- oder AAC-LC-Stream codiert. Mit dem Audioencoder können nicht nur Internetstreams entgegengenommen werden, sondern er kann auch als Eingang im Jack-Audioserver dienen. Dazu wird der Audioencoder mit dem Parameter *-j odr* ausgeführt, womit er anschließend im Jack-Server unter dem Namen *odr* zur Verfügung steht.

Weitere Hilfestellungen befinden sich in der Anlage 5.6.

### 4.7.2 ODR-PadEnc

Um programmbegleitende Daten zu übertragen, müssen diese vorher korrekt kodiert werden. Diese Kodierung übernimmt der ODR-PadEncoder. Dabei kann er sowohl eine Slideshow als auch Dynamic-Labels verarbeiten. Damit der Audioencoder später auf diese Daten zugreifen kann, werden sie in einer FIFO-Datei gespeichert. Ein Beispielaufruf, der eine Slideshow und ein Label generiert und diese anschließend in der Datei *pad-SC\_Radio\_1.fifo* abspeichert, sieht wie folgt aus:

```
odr-padenc -t ./ dyn-dl.txt -c 15 -d ./ mot_1 -p 58 -o ./ pad-  
SC_Radio_1.fifo
```

Mit *-t* wird der Pfad zur Datei des Dynamic Label-Textes angegeben, *-c* setzt die Zeichenkodierung, *-d* gibt das Unterverzeichnis der Bilddateien für die Slideshow an, *-p* gibt die Länge der PAD an und mit *-o* wird der Output gesetzt. Die empfohlene Bildgröße beträgt 320x240 Pixel, größer Bilddateien werden komprimiert.

Der Parameter *-t* kann mehrmals angegeben werden. Dadurch lassen sich mehrere Dynamiclabels einfügen.

Weitere Befehlsinformationen können der Anlage 5.5 entnommen werden.

### 4.7.3 ODR-DabMux

Der ODR-DabMux fasst mehrere verschiedene Audio- und Datendienste in einen Multiplex zusammen. Der Aufbau des Multiplex wird dabei in einer Konfigurationsdatei definiert. Die Struktur der Datei lehnt sich dabei an die Struktur des DAB-Multiplex an. Ein Multiplex, auch Ensemble genannt, umfasst mehrere Services. Die Services sind die mit einem DAB-Radioempfänger auswählbaren und anhörbaren Programme. Dabei besteht ein Service aus mindestens einer Komponente, die die Services mit den jeweiligen Audiodaten oder Zusatzdaten in den Subchannels des Main Service Channels verbindet. Eine Komponente kann dabei mehreren Services zugeordnet werden. So muss z.B. ein Verkehrsinformationsdienst nicht mehrfach übertragen werden. Der Fast Information Channel (FIC) überträgt die Steuerungs- und Decodierungsdaten.

Eine Beispielkonfigurationsdatei für den ODR-DabMux sieht wie in Anlage 5.9 aus. Der Name der erstellten Datei sollte *dab.mux* lauten. Die Konfiguration erzeugt 3 Radioprogramme, die per ZeroMQ an das EasyDAB-Board übermittelt werden. Die Datei besteht dabei aus sieben Abschnitten, wobei ein Abschnitt mit geschweiften Klammern eingegrenzt wird und deren Reihenfolge irrelevant ist.

- general
- remotecontrol
- ensemble
- services
- subchannels
- components
- outputs

#### 4.7.3.1 *general*

In diesem Abschnitt werden die grundsätzlichen Parameter des Multiplex definiert. Dies sind zum einen die DAB-bezogenen Parameter, wie der DAB-Mode (1-4), die Anzahl der zu erzeugenden DAB-Frames (0 für Dauerbetrieb), der Zeitstempel für SFN und das Setzen des SCCA-Feld für diverse ETI-Analysatoren. Zum anderen werden auch softwarespezifische Parameter definiert, die beispielsweise das Debuggen erleichtern. Diese sind das Loggen im Systemlog oder das Bereitstellen eines Statistikservers.

#### 4.7.3.2 *remotecontrol*

Der Abschnitt *remotecontrol* ist optional und kann auch weggelassen werden. Er dient dazu eine Telnetport für einen Fernzugriff freizuschalten, mit dem sich einige Parameter verändern lassen.

#### 4.7.3.3 *ensemble*

Im Abschnitt *ensemble* sind die Parameter für den Multiplex definiert. Diese Parameter sind die Ensemble ID, die von der Bundesnetzagentur vergeben wird, der Extended Country Code (ECC), die lokale Zeitzone, die internationale Tabelle für programmspezifische Parameter und das Label sowie das Short-Label. Beim Short-Label gilt es zu beachten, dass es aus Buchstaben des Labels bestehen muss.

#### 4.7.3.4 *Services*

Dieser Teil definiert die DAB-Programme in einem Multiplex. Dabei wird jeder Service in einem eigenen Unterabschnitt definiert. Ein Unterabschnitt beginnt mit einer individuellen, möglichst sinnvollen Benennung. Diese Kennung wird nur innerhalb des ODR-DabMux verwendet und beeinflusst nicht das Label. Ein Unterabschnitt beginnt und endet jeweils mit einer geschweiften Klammer. Für einen Service muss ein Label definiert werden. Ergänzend kann auch ein Shortlabel, ein Programmtype und eine Programmsprache gesetzt werden.

Die Label-Vorschrift gilt wie im Abschnitt 4.7.3.3 auch hier.

#### 4.7.3.5 *Subchannels*

Im Abschnitt *subchannels* werden die Subchannel des DAB-Multiplex in jeweils einem eigenen Unterabschnitt, beginnend mit einer individuellen Kennung, definiert. Zunächst muss der Datentyp des Subchannels definiert werden. Für Audiodaten wird Audio (MP2) oder Dabplus (AAC) gesetzt. Es lassen sich außerdem noch weitere Typen, wie z.B. DMB, Data, Packet etc. angeben. Als Eingangsdatei wird entweder der Pfad zu einer Datei oder der ZeroMQ-Stream angegeben. Die Bitrate des Subchannels muss so eingestellt werden, dass sie mit der Bitrate des Audioencoders übereinstimmt. Die Höhe des Fehlerschutzes, auch Protection Level genannt, und das Protection Profil (UEP für DAB, EEP\_A oder EEP\_B für DAB+) werden für jeden Subchannel einzeln eingestellt. Optional lässt sich noch eine ID für einen Subchannel angeben.

#### 4.7.3.6 Components

Im Abschnitt components werden die Subchannel mit den Services durch die Service Components verknüpft. Auch hier beschreibt ein Unterabschnitt eine Service Component. Zwingend erforderlich ist die Übergabe eines Service und des zu verbindenden Subchannels mittels ihrer Kennungen.

Optional können auch Label vergeben werden.

#### 4.7.3.7 Outputs

In diesem Abschnitt lässt sich die Ausgabe des ETI-Datenstroms festlegen. Die Ausgangsdaten lassen sich über eine Datei ausgeben. Dazu muss der Dateipfad und das Format (raw, framed, streamed) angegeben werden. Außerdem lassen sich die Daten auch über ein ZMQ+TCP-Stream ausgeben; hierbei muss der Port angegeben werden. Weiterhin ist es auch möglich, die Daten über EDI auszugeben.

Weitere Informationen können unter *example.mux* und *advanced.mux* auf <https://github.com/OpenDigitalradio/ODR-DabMux/tree/master/doc> gefunden werden.

### 4.7.4 ODR-DabMod

Soll die Modulation der I/Q-Samples auf dem Raspberry Pi berechnet werden, so muss der ODR-DABMod installiert und gestartet werden. Der Modulator muss mit einer Konfigurationsdatei, die z.B. *B100.ini* heißt, gestartet werden. Ein Aufruf sieht wie folgt aus:

```
odr-dabmod -C B100.ini
```

Der Aufbau der Konfigurationsdatei befindet sich in der Anlage 5.10. Im Folgenden wird der Aufbau dieser Datei erläutert. Kommentare werden durch ein Semikolon gekennzeichnet.

#### 4.7.4.1 Remotecontrol

In diesem Abschnitt wird die Fernwartung aktiviert sowie der Telnet Port angegeben.

#### 4.7.4.2 Log

Der Log-Abschnitt dient dazu eine Protokollierung zu aktivieren. Diese kann im Systemlog oder in einer Datei gespeichert werden. Die Angabe des Pfades ist hierbei zu berücksichtigen.

#### 4.7.4.3 *Input*

In diesem Abschnitt wird der Eingangsstrom des Modulators eingestellt. Das ETI-Signal kann hierbei entweder als Datei oder ZeroMQ-Stream eingespeist werden. Bei Verwendung des ZeroMQ-Streams muss zusätzlich der eingestellte Port des Multiplexers angegeben werden. Außerdem lässt sich die maximale Anzahl zwischengespeicherter Frames angeben.

#### 4.7.4.4 *Modulator*

Der Abschnitt `modulator` beschreibt die Modulation der OFDM-Symbole. Der Parameter `gainmode` sollte laut [7] den festen Wert 2 haben. Es lässt sich nachträglich der DAB-Mode des Multiplexs einstellen. Außerdem kann eine Digitalverstärkung und die Ausgangs-Samplerate (2,048 MHz beim B100 und 5 MHz beim N200) gesetzt werden.

#### 4.7.4.5 *Firfilter*

Ein FIR-Filter kann aktiviert und die Koeffizienten für das Filter anhand einer Textdatei übergeben werden.

#### 4.7.4.6 *Output*

In diesem Abschnitt kann der Ausgang des Modulators definiert werden. Hierbei kann entweder zwischen einer Datei, einem ZeroMQ-Stream oder dem USRP (UHD-Output) gewählt werden.

#### 4.7.4.7 *fileoutput*

Wenn die Option `File` gewählt ist, wird der Ausgangsdatenstrom in die angegebene Datei geschrieben.

#### 4.7.4.8 *uhdoutput*

Die Parameter für den USRP werden in diesem Abschnitt angegeben. Der `Type` ist für den USRP B100 „`b100`“ und für den USRP N200 „`usrp2`“. Anschließend wird die Sendefrequenz oder der DAB-Block definiert und eine USRP-interne HF-Verstärkung angegeben.

Die Netzwerkkonfiguration USRP N-Serien müssen mit dem des Servers übereinstimmen.

#### 4.7.4.9 zmqoutput

Der Abschnitt zmqoutput gibt den „zu hörenden Port“ an, sowie den Typ des Sockets an.

Weitere Informationen für die Konfigurationsdatei des Modulators finden sich unter <https://github.com/OpenDigitalradio/ODR-DabMod/blob/master/doc/example.ini>

#### 4.7.5 DABlin

Der DABlin ist ein DAB-Player er besitzt eine grafische Oberfläche kann aber auch über die Kommandozeile bedient werden. Die grafische Oberfläche lässt sich mit dem folgenden Befehl starten.

```
dablin_gtk ./LMK.eti
```

*LMK.eti* ist die FIFO-Datei die der Multiplexer erzeugt. Über den Menüpunkt *Service* lassen sich die verschiedenen Services die vorher in der Multiplexkonfigurationsdatei angelegt worden sind auswählen.

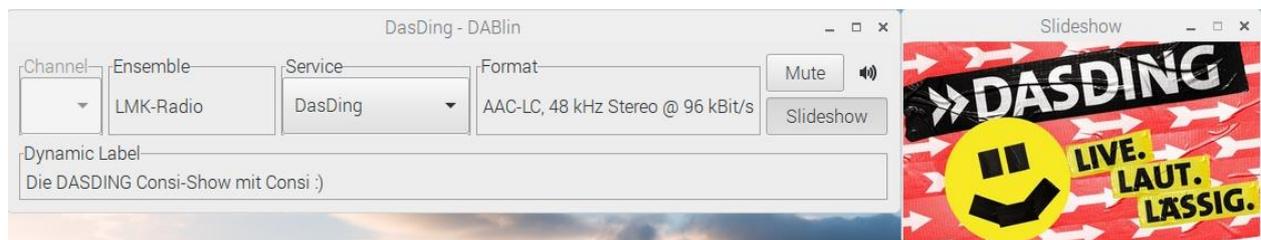


Abbildung 18: DABlin

Weitere Hilfestellungen zum Kommandozeilenprogramm des DABlin befinden sich in der Anlage 5.11.

## 4.8 AUSFÜHRUNG DER ODR-MMBTOOLS MITTELS SKRIPT

Damit die Programme nicht einzeln mittels Konsole aufgerufen werden müssen, kann ein Skript verwendet werden, das die Programme in screens startet. Es wird dazu ein Texteditor gestartet und eine Datei mit dem Namen *dab.sh* angelegt.

Nachfolgend wird ein Skript-Beispiel, das drei Radioprogramme startet, aufgeführt. Zwei Radioprogramme stehen unter dem JACK-Server zur Verfügung und das dritte spielt einen Internetlivestream ab. Das Skript kann mit dem Befehl *bash dab.sh -start* gestartet und mit dem Befehl *bash dab.sh -stop* gestoppt werden.

```
1 #!/bin/bash
2 case "$1" in
3 -start)
4   echo "DAB Prozess will be started"
5   killall screen
6   killall -9 screen
7   find . -maxdepth 1 -type p -delete
8   mkfifo ./PAD/pad-SC_Radio_1.fifo
9   mkfifo ./PAD/pad-SC_Radio_2.fifo
10  mkfifo ./PAD/pad-SC_Radio_3.fifo
11  mkfifo ./LMK.eti
12
13  #DABlin start
14  screen -dm -S dablin dablin_gtk ./LMK.eti
15  echo "Dablin started"
16  sleep 2
17
18  #Radio 1 Encoder start
19  screen -dm -S padenc-SC_Radio_1 odr-padenc -t ./PAD/radio1_dyn-
20  dl.txt -t ./PAD/Radio1_StaticDL.txt -c 15 -d ./PAD/mot_1 -s 10 -
21  p 58 -o ./PAD/pad-SC_Radio_1.fifo
22  echo "PADEncoder1 started"
23  screen -dm -S audioenc-SC_Radio_1 odr-audioenc -j soundcard -R -
24  b 96 -c 2 -r 48000 -P ./PAD/pad-SC_Radio_1.fifo -p 58 -o
25  tcp://localhost:9001 -D
26  echo "AudioEncoder1 started"
27
28  #Radio 2 Encoder start
29  screen -dm -S padenc-SC_Radio_2 odr-padenc -t ./PAD/radio2_dyn-
30  dl.txt -c 15 -d ./PAD/mot_2 -s 10 -p 58 -o ./PAD/pad-
31  SC_Radio_2.fifo
32  echo "PADEncoder2 started"
33  screen -dm -S audioenc-SC_Radio_2 odr-audioenc -j odr -R -b 96 -
34  c 2 -r 48000 -P ./PAD/pad-SC_Radio_2.fifo -p 58 -o
35  tcp://localhost:9002
36  echo "AudioEncoder2 started"
37
38  #Radio 3 Encoder start
39  screen -dm -S padenc-SC_Radio_3 odr-padenc -t ./PAD/radio3_dyn-
40  dl.txt -c 15 -d ./PAD/mot_3 -s 10 -p 58 -o ./PAD/pad-
41  SC_Radio_3.fifo
```

```
32 echo "PADEncoder3 started"
33 screen -dm -S audioenc-SC_Radio_3 odr-audioenc -v http://swr-
dasding-
live.cast.addradio.de/swr/dasding/live/mp3/128/stream.mp3 -R -b
96 -c 2 -r 48000 -P ./PAD/pad-SC_Radio_3.fifo -p 58 -o
tcp://localhost:9003 --write-icy-text=./PAD/radio3_dyn-dl.txt
34 echo "AudioEncoder3 started"
35 sleep 2
36 #Dab Multiplex start
37 screen -dm -S dabmux odr-dabmux -e dab.mux
38 echo "DABMux started"
39
40 #Process Monitoring
41 screen -dm -S mailTestDABmux bash mailTest.sh odr-dabmux
42 echo "Processmonitoring for DABmux started"
43 screen -dm -S mailTestDABmux bash mailTest.sh odr-padenc
44 echo "Processmonitoring for odr-padEncoder started"
45 screen -dm -S mailTestDABmux bash mailTest.sh odr-audioenc
46 echo "Processmonitoring for odr-audioEncoder started"
47 ;;
48 -stop)
49 echo "DAB-Process will be terminated"
50 killall -15 screen
51 rm ./PAD/pad-SC_Radio_1.fifo
52 rm ./PAD/pad-SC_Radio_2.fifo
53 rm ./PAD/pad-SC_Radio_3.fifo
54 rm ./LMK.eti
55 echo "DAB-Process terminated and all fifo-Files deleted"
56 ;;
57 -h)
58 echo " ~This Script starts or stops a DAB-Multiplex-Process~ "
59 echo " ~It will start the odr-padenc, odr-audioenc and odr-
dabmux~"
60 echo " ~Also a Script for monitoring the Processes starts~ "
61 echo " ~This script will send an email if a process interupts~ "
62 echo " Options: "
63 echo " -start      :Starts the Script "
64 echo " -stop       :Stops the Script and kills all active
Processes"
65 ;;
66 *)
67 echo "Wrong Parameters!!!"
68 echo "See -h for Help"
69 ;;
70 esac
```

Kommentare werden mit einem # markiert, die bei der Ausführung des Skripts ignoriert werden. In Zeile 5-6 werden ältere offene Screens beendet.

Da die Kommunikation zwischen PAD-Encoder und Audio-Encoder mittels FIFO-Dateien stattfindet, werden diese in den Zeilen 8-10 erzeugt.

In den Zeilen 18-34 werden jeweils der Audio-Encoder und der PAD-Encoder für die drei Radioprogramme gestartet.

In den Zeilen 40-46 werden die Skripte zur Prozessüberwachung aufgerufen, genauere Erläuterungen dazu befinden sich in Kapitel 4.2.1.

Die Zeilen 49-55 werden aufgerufen, wenn der *stop* Parameter dem Skript übergeben wurde. Es werden dann alle aktiven Screens gestoppt und alle erzeugten FIFO-Dateien gelöscht.

#### 4.8.1 Skripterzeugung mittels dem Tools GUI4ODR

Mit dem Java-Programm GUI4ODR lässt sich ein Skript sehr komfortabel und schnell erzeugen. Das Programm ist nicht auf dem Raspberry Pi lauffähig, da JavaFX nicht unterstützt wird. Es lassen sich aber auf einem PC, auf dem das Programm ausführbar ist, Skripte erstellen. Der Projektordner, der durch GUI4ODR angelegt wurde, kann daraufhin auf den Raspberry Pi übertragen werden. Das im Projektordner befindliche Skript *dab.sh* lässt sich dann mit dem Befehl *bash dab.sh* ausführen. Außerdem kann es wie das Skript in Kapitel 4.8 modifiziert werden, damit es ebenfalls mit dem *-start* und *-stop* Parameter gestartet und gestoppt werden kann. Eine detaillierte Bedienungsanleitung findet sich unter [8].

### 4.9 AUSFÜHRUNG DER ODR-MMBTOOLS MITTELS SUPERVISOR

Die ODR-Tools lassen sich auch mit der Anwendung Supervisor starten. Zuerst müssen dafür jedoch noch einige Konfigurationsdateien bearbeitet werden.

#### 4.9.1 Webserver aktivieren

Um per Weboberfläche auf Supervisor zuzugreifen können, muss zunächst die Datei */etc/supervisor/supervisord.conf* bearbeitet werden. Dazu wird am Ende der Datei folgendes eingefügt:

```
[inet_http_server]
port = 9100
username = user ; Auth username
password = pass ; Auth password
```

Dabei gibt *port* den Port an, auf dem der Server nach eingehenden Anfragen hören soll. Darüberhinaus lassen sich optional noch Benutzername und Passwort zur Authentifikation der Weboberfläche angeben.

---

## 4.9.2 Konfigurationsdatei zum Starten des Prozesses

Damit Supervisor weiß, welche Prozesse gestartet werden, muss eine Konfigurationsdatei wie folgt erstellt werden:

```
sudo nano supervisorLMKConfig.conf
```

Der Aufbau der Datei, um z.B. den ODR-DabMux zu starten, sieht wie folgt aus:

```
[program:Multiplex]
command=odr-dabmux dab.mux
directory=/home/rp3/ODR-Testconfigs/Test-Soundcard-ETI_SuperVisorD
autostart=true
autorestart=false
stderr_logfile=/var/log/supervisor/mux.log
stdout_logfile=/var/log/supervisor/mux.log
```

Mit *command* wird der Befehl definiert, der ausgeführt werden soll. *Directory* gibt an, in welchem Verzeichnis der Befehl ausgeführt wird. Mit *autorestart* lässt sich ein Neustarten des Befehls ausführen, falls er beispielsweise unerwartet beendet wurde. Am Ende lassen sich noch log-Files erzeugen, in denen die Standardausgabe und Fehlerausgabe gespeichert wird.

Um andere Befehle auszuführen, wird genauso vorgegangen. Sie können dann auch in die Konfigurationsdatei geschrieben werden.

Damit Supervisor nun auf die Konfigurationsdatei zugreifen kann, wird eine Verknüpfung erstellt. Anschließend muss Supervisor die Konfigurationsdatei neu einlesen und ein Update durchführen.

```
sudo ln -s /dir/of/configfile/supervisorLMKConfig.conf
    /etc/supervisor/conf.d/supervisorLMKConfig.conf
sudo supervisorctl reread
sudo supervisorctl update
```

## 4.9.3 Email-Benachrichtigung aktivieren

Um eine Email zu erhalten, falls ein Prozess unerwartet beendet wurde, wird der folgende Abschnitt noch in die Konfigurationsdatei hinzugefügt. Es gilt zu beachten, dass zuvor der sSMTP konfiguriert sein muss.

```
[eventlistener:crashmail]
command=/usr/local/bin/crashmail -a -m Empfänger@domain.com
events=PROCESS_STATE
```

#### 4.9.4 Überwachung per Weboberfläche

Die ausgeführten Prozesse lassen sich mit Supervisor bequem über eine Weboberfläche steuern und überwachen. Die Weboberfläche lässt sich über jeden beliebigen Browser öffnen, indem man die `http://localhost:9100` aufruft. Falls man per Fernzugriff zugreifen möchte, wird die externe IP-Adresse des Toolbox-Senders angegeben. Zuvor muss jedoch im Router Port-Forwarding auf den Port 9100 aktiviert werden. Die Portangabe variiert je nach Einstellung der Datei `supervisord.conf` (Abschnitt 4.9.1). Außerdem ist eine Authentifizierung erforderlich, falls ein Username und Passwort in der Konfigurationsdatei angegeben worden sind.

Auf der Oberfläche lässt sich anschließend z.B. mit dem Button `Tail -f` die Standardausgabe eines Prozesses anzeigen.

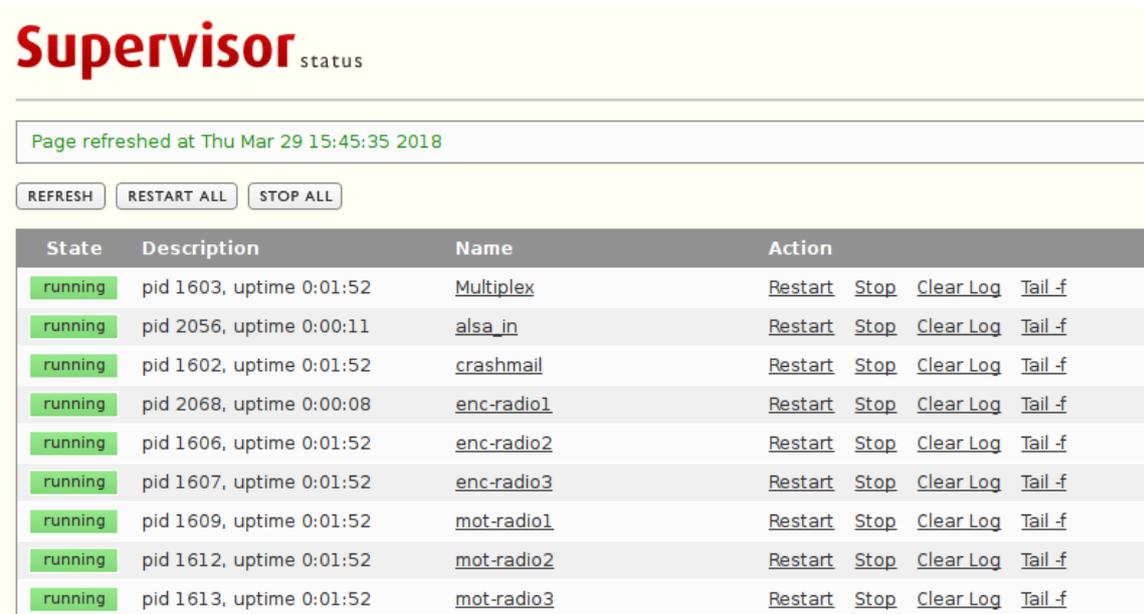


Abbildung 19: Weboberfläche Supervisor

#### 4.9.5 DAB-Sender mit Supervisor mittels Skript starten und beenden

Damit die Befehle nicht einzeln aufgerufen werden müssen, kann der DAB-Sender auch mit einem Skript gestartet werden. Ein Skript, das 3 Radioprogramme bereitstellt und die mittels Supervisor überwacht werden können, sieht wie folgt aus:

```

1 #!/bin/bash
2 # This Script will start DAB-Multiplex, Audioencoder and Pad-
  Encoder
3 # it runs with supervisord, monitoring is possible over
  webinterface
4 # from supervisor
5 case "$1" in
```

```
6 -start)
7   echo "DAB Prozess will be started"
8   find . -maxdepth 1 -type p -delete
9   killall -9 screen
10  #create Fifo-Files for Communication between padenc and audioenc
11  mkfifo ./PAD/pad-SC_Radio_1.fifo
12  mkfifo ./PAD/pad-SC_Radio_2.fifo
13  mkfifo ./PAD/pad-SC_Radio_3.fifo
14  mkfifo ./LMK.eti
15  #Delete Old Supervisorconfigfiles
16  sudo rm -r /etc/supervisor/conf.d/*
17
18
19  #DABlin start
20  screen -dm -S dablin dablin_gtk ./LMK.eti
21  echo "Dablin started"
22  sleep 2
23
24  # Load new Config into Supervisor conf.d Folder
25  sudo ln -s /home/rp3/ODR-Testconfigs/Test-Soundcard-
    ETI SuperVisorD/supervisorConfig/enc-radiol.conf
    /etc/supervisor/conf.d/enc-radiol.conf
26  sudo supervisorctl reread
27  sudo supervisorctl update
28  sudo service supervisor reload
29  echo "Supervisor ready"
30 ;;
31 -stop)
32  echo "DAB-Process will be terminated"
33  #Delete Fifo-Files and Supervisor config files
34  rm ./PAD/pad-SC_Radio_1.fifo
35  rm ./PAD/pad-SC_Radio_2.fifo
36  rm ./PAD/pad-SC_Radio_3.fifo
37  rm ./LMK.eti
38  sudo rm -r /etc/supervisor/conf.d/*
39  sudo supervisorctl reread
40  sudo supervisorctl update
41  #stop dablin_gtk
42  killall -9 screen
43  echo "DAB-Process terminated and all fifo-Files deleted"
44 ;;
45 -h)
46  echo " ~This Script starts or stops a DAB-Multiplex-Process~ "
47  echo " ~It will start the odr-padenc, odr-audioenc and odr-
    dabmux~"
48  echo " ~Also a Script for monitoring the Processes starts~ "
49  echo " ~This script will send an email if a process interupts~ "
50  echo " Options: "
51  echo " -start           :Starts the Script "
52  echo " -stop            :Stops the Script and kills all active
    Processes"
53 ;;
54 *)
55  echo "Wrong Parameters!!!"
56  echo "See -h for Help"
```

```
58 ;;  
59 esac
```

Das Skript lässt sich mit dem Befehl `bash dab.sh` starten. Als Parameter muss entweder `-start` zum Starten des DAB-Prozesses oder `-stop` zum Stoppen des DAB-Prozesses angegeben werden.

In Zeile 17 werden alte Konfigurationsdateien aus dem Ordner `/etc/supervisor/conf.d` gelöscht. Damit wird vermieden, dass Supervisor auf nicht aktuelle Konfigurationen zurückgreift.

In Zeile 25 wird dann eine Verknüpfung auf die aktuelle Konfigurationsdatei erstellt. Daraufhin wird der Supervisor in den Zeilen 26-29 neu gestartet, sodass er die aktuelle Konfiguration ausführt.

Bei Übergabe des `stop` Parameters werden die Zeilen 32-44 ausgeführt. Hier werden die erzeugten FIFO-Dateien und die Verknüpfung zur Konfiguration des Supervisors gelöscht. Außerdem wird der Supervisor neu gestartet.

#### 4.10 BEDIENUNG DES EASYDAB-BOARDS

Das EasyDAB-Board kann über eine Weboberfläche eingestellt werden. Dazu wird im Browser die IP-Adresse des Boards 192.168.3.4 geöffnet. Anschließend gibt man die folgenden Anmeldedaten an:

User	admin
passwort	admin

Tabelle 5: Anmeldedaten EasyDAB

Es öffnet sich das in Abbildung 20 das dargestellte Fenster. Unter Punkt (1) lässt sich die IP-Adresse des EasyDAB-Boards angeben. Unter *Gateway* und *Netmask* lässt sich jeweils die IP-Adresse des Routers und die Netzmaske des Netzwerkes einstellen. Um mit dem Raspberry Pi zu kommunizieren, wird unter Punkt (2) bei *Remote-IP* die IP-Adresse des Raspberry Pi eingestellt und unter *Remote-Port* der Port, der in der Multiplexkonfigurationsdatei für ZMQ angegeben wurde.

Unter dem Menüpunkt *RF-Frontend* lassen sich die HF-Parameter einstellen. Hierbei kann die Leistung des Signals mit den Punkten (3) und (5) eingestellt werden und die Frequenz des verwendeten DAB-Kanals mit dem Punkt (4).

Die Anmeldedaten für den Zugriff auf die Weboberfläche lassen sich in Punkt (6) einstellen.

Unter den Punkten (7) und (8) lassen sich verschiedene Aktionen durchführen. So kann z.B. unter Punkt (7) die Konfiguration mit *apply config* gespeichert werden. Das Speichern der Konfiguration sollte nach jeder Änderung durchgeführt werden.

Unter Punkt (8) lässt sich der ETI-Socket und das RF-Frontend aus- und anschalten.

**DAB Modulator control panel  
easydabv2.6144-28.11.2017**

**Network:**

IP-address: 192.168.3.4 (1)

Netmask: 255.255.255.0

Gateway: 192.168.3.1

MAC-address: 00:08:dc:18:01:91

Connection mode:   
 TCP-Client   
 TCP-Server   
 Off   
 status: ● Connecting

Remote IP: 192.168.3.100 (2)

Remote port: 18081

Enable ZeroMQ Client Handshake

No Delayed TCP ACK

**RF Frontend:**

Amplitude: 107% (abs:137) (3)

Frequency: 223936000 Hz (4)

DAC Current: 97% (abs:124) (5)

Source Frequency: 30.72 MHz   
FPLL: 30.72x20=614.4MHz   
 SR: 614.4/100=6.144MHz

PLL Charge-pump: 387 uA

FREQ/PPS Correction: -1638.4 Hz

Status: ● RF-frontend active

Abbildung 20: Weboberfläche EasyDAB

**Processing:**

Controls:

Enable external SPI RAM buffers

Enable Single Frequency Network

Enable TX only if oscillator is GPS-synced

Enable internal GPSDO

ETI-GPS Delay: 1250 ms+0 us

Enable TII

TII Main Id: 0

TII Sub Id: 0

Info:

Buffer fullness: 0 %

Flags:

- GPS time is not known yet
- GPS reference isn't ready
- SPI SRAM buffers detected
- Broadcast OFF

**Authentication:**

Login: admin (6)

Password: admin

**Actions:**

Config:    (7)

Actions:   (8)

System:

Status: received

Abbildung 21: Weboberfläche EasyDAB

## 5 ANLAGEN

### 5.1 INSTALLATIONSSKRIPT RASPDAB.SH

```
#!/bin/bash
#
# Installer script for
#
# * ODR-mmbTools:
#   * ODR-DabMux
#   * auxiliary scripts
#   * the FDK-AAC library with DAB+ patch
#   * ODR-AudioEnc
#   * ODR-PadEnc
#
# and all required dependencies for a
# Raspbian stable system.
#
# Requires: sudo

RED="\e[91m"
GREEN="\e[92m"
NORMAL="\e[0m"

DISTRO="unknown"

if [ $(lsb_release -d | grep -c wheezy) -eq 1 ] ; then
    echo -e $RED
    echo "Warning, debian wheezy is not supported anymore"
    echo -e $NORMAL
    exit 1
elif [ $(lsb_release -d | grep -c jessie) -eq 1 ] ; then
    DISTRO="jessie"
fi

echo
echo "This is the mmbTools installer script for raspbian/debian"
echo "======"
echo
echo "It will install ODR-DabMux, ODR-AudioEnc, ODR-PadEnc"
echo "and all prerequisites to your machine."
echo $DISTRO

if [ "$DISTRO" == "unknown" ] ; then
    echo -e $RED
    echo "You seem to be running something else than"
    echo "debian jessie. This script doesn't"
    echo "support your distribution."
    echo -e $NORMAL
    exit 1
fi
```

```
echo -e $RED
echo "This program will use sudo to install components on your"
echo "system. Please read the script before you execute it, to"
echo "understand what changes it will do to your system !"
echo
echo "There is no undo functionality here !"
echo -e $NORMAL

if [ "$UID" == "0" ]
then
    echo -e $RED
    echo "Do not run this script as root !"
    echo -e $NORMAL
    echo "Install sudo, and run this script as a normal user."
    exit 1
fi

which sudo
if [ "$?" == "0" ]
then
    echo "Press Ctrl-C to abort installation"
    echo "or Enter to proceed"

    read
else
    echo -e $RED
    echo -e "Please install sudo first $NORMAL using"
    echo " apt-get -y install sudo"
    exit 1
fi

# Fail on error
set -e

if [ -d dab ]
then
    echo -e $RED
    echo "ERROR: The dab directory already exists."
    echo -e $NORMAL
    echo "This script assumes a fresh initialisation,"
    echo "if you have already run it and wish to update"
    echo "the existing installation, please do it manually"
    echo "or erase the dab folder first."
    exit 1
fi

echo -e "$GREEN Updating debian package repositories $NORMAL"
sudo apt-get -y update

echo -e "$GREEN Installing essential prerequisites $NORMAL"
# some essential and less essential prerequisites
sudo apt-get -y install build-essential git wget \
sox alsa-tools alsa-utils \
```

```
automake libtool mpg123 \  
libasound2 libasound2-dev \  
libjack-jackd2-dev jackd2 \  
ncdu vim ntp links cpufrequtils \  
libfftw3-dev \  
libcurl4-openssl-dev \  
libmagickwand-dev \  
libvlc-dev vlc-nox \  
libfaad2 libfaad-dev \  
python-mako python-requests  
  
# this will install boost, cmake and a lot more  
sudo apt-get -y build-dep uhd  
  
# stuff to install from source  
mkdir dab || exit  
cd dab || exit  
  
#COMMENTED OUT FOR RASPDAB  
#echo -e "$GREEN Compiling UHD $NORMAL"  
#git clone http://github.com/EttusResearch/uhd.git  
#pushd uhd  
#cd host  
#mkdir build  
#cd build  
#cmake ../  
#make  
#sudo make install  
#popd  
  
#echo -e "$GREEN Downloading UHD device images $NORMAL"  
#sudo /usr/local/lib/uhd/uhd_utils/downloader.py  
  
sudo apt-get -y install libzmq3-dev libzmq3  
  
#echo -e "$GREEN Installing KA9Q libfec $NORMAL"  
#git clone https://github.com/Opendigitalradio/ka9q-fec.git  
#pushd ka9q-fec  
#./bootstrap  
#./configure  
#make  
#sudo make install  
#popd  
  
echo  
echo -e "$GREEN PREREQUISITES INSTALLED $NORMAL"  
### END OF PREREQUISITES  
  
echo -e "$GREEN Fetching mmbtools-aux $NORMAL"  
git clone https://github.com/mpbraendli/mmbtools-aux.git  
  
echo -e "$GREEN Fetching etisnoop $NORMAL"  
git clone https://github.com/Opendigitalradio/etisnoop.git  
pushd etisnoop
```

```
./bootstrap.sh
./configure
make
sudo make install
popd

echo -e "$GREEN Compiling ODR-DabMux $NORMAL"
git clone https://github.com/Opendigitalradio/ODR-DabMux.git
pushd ODR-DabMux
./bootstrap.sh
./configure --enable-input-zeromq --enable-output-zeromq --with-
    boost-libdir=/usr/lib/arm-linux-gnueabi/f
make
sudo make install
popd

#COMMENTED OUT FOR RASPDAB
#echo -e "$GREEN Compiling ODR-DabMod $NORMAL"
#git clone https://github.com/Opendigitalradio/ODR-DabMod.git
#pushd ODR-DabMod
#./bootstrap.sh
#./configure --enable-zeromq --enable-output-uhd --disable-native
#make
#sudo make install
#popd

echo -e "$GREEN Compiling fdk-aac library $NORMAL"
git clone https://github.com/Opendigitalradio/fdk-aac.git
pushd fdk-aac
./bootstrap
./configure
make
sudo make install
popd

echo -e "$GREEN Updating ld cache $NORMAL"
# update ld cache
sudo ldconfig

echo -e "$GREEN Compiling ODR-AudioEnc $NORMAL"
git clone https://github.com/Opendigitalradio/ODR-AudioEnc.git
pushd ODR-AudioEnc
./bootstrap
./configure --enable-alsa --enable-jack --enable-vlc
make
sudo make install
popd

echo -e "$GREEN Compiling ODR-PadEnc $NORMAL"
git clone https://github.com/Opendigitalradio/ODR-PadEnc.git
pushd ODR-PadEnc
./bootstrap
./configure --enable-jack --enable-vlc
```

```
make
sudo make install
popd

echo -e "$GREEN Done installing all tools $NORMAL"
echo -e "All the tools have been dowloaded to the dab/ folder,"
echo -e "compiled and installed to /usr/local"
echo
echo -e "The stable versions have been compiled, i.e. the latest"
echo -e "'master' branch from the git repositories"
echo
echo -e "If you know there is a new release, and you want to
update,"
echo -e "you have to go to the folder containing the tool, pull"
echo -e "the latest changes from the repository and recompile"
echo -e "it manually."
echo
echo -e "To pull the latest changes for ODR-DabMux, use:"
echo -e " cd ~/dab/ODR-DabMux"
echo -e " git pull"
echo -e " ./bootstrap.sh"
echo -e " ./configure --enable-input-zeromq --enable-output-zeromq"
echo -e " make"
echo -e " sudo make install"
echo
echo -e "This example should give you the idea. For the options"
echo -e "for compiling the other tools, please see in the debian.sh"
echo -e "script what options are used. Please also read the README"
echo -e "and INSTALL files in the repositories."
```

## 5.2 PARAMETER DER S.USV SOFTWARE

```
Usage: ./susv -chrgon | -chrgoff | -vin [0] | -pwrext [0] | -pwrbat
[0] | -capbat [0] | -status | -timer | -auto | -sleep | -mail |
-boot | -setboot | -shutdown | -chrgpwr | -chgadd | -flash
Options:
  -chrgon           Switch charging circuit on
  -chrgoff          Switch charging circuit off
  -vin              Read input voltage
  -pwrext           Read power consumption
  -pwrbat           Read battery consumption
  -capbat           Read battery capacity
  -status           Read S.USV status
  -timer            Set shutdown timer
  -auto             Set autostart of S.USV Daemon [0|1]
  -sleep            Set S.USV Daemon sleep variable [sec]
  -mail             Set S.USV Mail notification [0|1]
  -boot             Set S.USV timed boot [0|1]
  -setboot          Set S.USV boot time [HH:mm:ss]
  -shutdown         Set S.USV timed shutdown [0|1]
```

```
-setshutdown      Set S.USV shutdown time [HH:mm:ss]
-chrgpwr          Set charging current [mA]
-chgadd           Set I2C-Adress [eg. 0x0F]
-flash           Upgrade Firmware
```

Optional Parameter:

```
0                Clear Output
```

## 5.3 VERWENDUNG DES EXTERNEN HDMI-AUSGANG

### 5.3.1 display

```
#!/bin/bash
mount /dev/mmcblk0p1
echo "Select 1 or 2 to choose your display type. Type 3 to Skip"
OPTIONS="hdmi lcd"
select opt in $OPTIONS; do
    if [ "$opt" = "hdmi" ]; then
        systemctl stop hdmi
        cp /boot/config.txt.hdmi /boot/config.txt
        reboot
        break
    elif [ "$opt" = "lcd" ]; then
        systemctl stop hdmi
        cp /boot/config.txt.lcd /boot/config.txt
        reboot
        break
    else
        echo "Thanks"
        break
    fi
done
```

#### 5.3.1.1 config.txt.lcd

```
# For more options and information see
# http://rpf.io/configtxt
# Some settings may impact device functionality. See link above for
  details

# uncomment if you get no picture on HDMI for a default "safe" mode
#hdmi_safe=1

# uncomment this if your display has a black border of unused pixels
  visible
# and your display can output without overscan
#disable_overscan=1
```

```
# uncomment the following to adjust overscan. Use positive numbers
  if console
# goes off screen, and negative if there is too much border
#overscan_left=16
#overscan_right=16
#overscan_top=16
#overscan_bottom=16

# uncomment to force a console size. By default it will be display's
  size minus
# overscan.
#framebuffer_width=1280
#framebuffer_height=720

# uncomment if hdmi display is not detected and composite is being
  output
#hdmi_force_hotplug=1

# uncomment to force a specific HDMI mode (this will force VGA)
#hdmi_group=1
#hdmi_mode=1

# uncomment to force a HDMI mode rather than DVI. This can make
  audio work in
# DMT (computer monitor) modes
#hdmi_drive=2

# uncomment to increase signal to HDMI, if you have interference,
  blanking, or
# no display
#config_hdmi_boost=4

# uncomment for composite PAL
#sdtv_mode=2

#uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=800

# Uncomment some or all of these to enable the optional hardware
  interfaces
#dtparam=i2c_arm=on
#dtparam=i2s=on
#dtparam=spi=on

# Uncomment this to enable the lirc-rpi module
#dtoverlay=lirc-rpi

# Additional overlays and parameters are documented
  /boot/overlays/README

dtparam=i2c1=on
dtparam=i2c_arm=on
```

```
# Enable audio (loads snd_bcm2835)
dtparam=audio=on

#Uncomment for HDMI as Video output device
#ignore_lcd=1
lcd_rotate=2
```

### 5.3.1.2 *config.txt.hdmi*

```
# For more options and information see
# http://rpf.io/configtxt
# Some settings may impact device functionality. See link above for
  details

# uncomment if you get no picture on HDMI for a default "safe" mode
#hdmi_safe=1

# uncomment this if your display has a black border of unused pixels
  visible
# and your display can output without overscan
#disable_overscan=1

# uncomment the following to adjust overscan. Use positive numbers
  if console
# goes off screen, and negative if there is too much border
#overscan_left=16
#overscan_right=16
#overscan_top=16
#overscan_bottom=16

# uncomment to force a console size. By default it will be display's
  size minus
# overscan.
#framebuffer_width=1280
#framebuffer_height=720

# uncomment if hdmi display is not detected and composite is being
  output
#hdmi_force_hotplug=1

# uncomment to force a specific HDMI mode (this will force VGA)
#hdmi_group=1
#hdmi_mode=1

# uncomment to force a HDMI mode rather than DVI. This can make
  audio work in
# DMT (computer monitor) modes
#hdmi_drive=2

# uncomment to increase signal to HDMI, if you have interference,
  blanking, or
# no display
#config_hdmi_boost=4
```

```
# uncomment for composite PAL
#sdtv mode=2

#uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=800

# Uncomment some or all of these to enable the optional hardware
  interfaces
#dtparam=i2c_arm=on
#dtparam=i2s=on
#dtparam=spi=on

# Uncomment this to enable the lirc-rpi module
#dtoverlay=lirc-rpi

# Additional overlays and parameters are documented
  /boot/overlays/README

dtparam=i2c1=on
dtparam=i2c_arm=on

# Enable audio (loads snd_bcm2835)
dtparam=audio=on

#Uncomment for HDMI as Video output device
ignore_lcd=1
```

## 5.3.2 Automatisches Starten auf den LCD-Display

### 5.3.2.1 *hdmi*

```
#!/bin/sh
cp /boot/config.txt.lcd /boot/config.txt
```

### 5.3.2.2 *hdmi.service*

```
[Unit]
Description=Veranlasst das bei Neustart/Herunterfahren immer in LCD
  bootet
Conflicts=reboot.target
After=network.target

[Service]
Type=oneshot
RemainAfterExit=true
ExecStart=/bin/true
ExecStop=/home/rp3/hdmi
```

```
[Install]
WantedBy=poweroff.target
```

### 5.3.2.3 startHDMIservice

```
#!/bin/sh
systemctl restart hdmi
```

## 5.4 JACK

Wichtige Parametereinstellung für den JACK-Audioserver:

```
Usage: qjackctl [options] [command-and-args]
QjackCtl - JACK Audio Connection Kit - Qt GUI Interface

Options:
  -s, --start
        Start JACK audio server immediately

  -p, --preset=[label]
        Set default settings preset name

  -a, --active-patchbay=[path]
        Set active patchbay definition file

  -n, --server-name=[label]
        Set default JACK audio server name

  -h, --help
        Show help about command line options

  -v, --version
        Show version information
```

## 5.5 ODR-PADENC

Wichtige Parametereinstellung für den ODR-PadEncoder:

```
ODR-PadEnc v2.3.0 - DAB PAD encoder for MOT Slideshow and DLS

By CSP Innovazione nelle ICT s.c.a r.l. (http://rd.csp.it/) and
Opendigitalradio.org

Reads image data from the specified directory, DLS text from a file,
and outputs PAD data to the given FIFO.
https://opendigitalradio.org

Usage: odr-padenc [OPTIONS...]
```

```
-d, --dir=DIRNAME      Directory to read images from.
-e, --erase            Erase slides from DIRNAME once they have
                      been encoded.
-s, --sleep=DUR       Wait DUR seconds between each slide
                      Default: 10
-o, --output=FILENAME FIFO to write PAD data into.
                      Default: /tmp/pad.fifo
-t, --dls=FILENAME    FIFO or file to read DLS text from.
                      If specified more than once, use next
                      file after slide switch (for uniform PAD encoder, -l is used
                      instead).
-p, --pad=LENGTH      Set the PAD length in bytes.
                      Possible values: 6 (short X-PAD), 8 to
                      196 (variable size X-PAD)
                      Default: 58
-c, --charset=ID      ID of the character set encoding used for
                      DLS text input.
                      ID = 0: Complete EBU Latin based
                      repertoire
                      ID = 6: ISO/IEC 10646 using UCS-2 BE
                      ID = 15: ISO/IEC 10646 using UTF-8
                      Default: 15
-r, --remove-dls      Always insert a DLS Remove Label command
                      when replacing a DLS text.
-C, --raw-dls         Do not convert DLS texts to Complete EBU
                      Latin based repertoire
                      character set encoding.
-m, --max-slide-size=SIZE Recompress slide if above the specified
                      maximum size in bytes.
                      Default: 51200 (Simple Profile)
-R, --raw-slides      Do not process slides. Integrity checks
                      and resizing
                      slides is skipped. Use this if you know
                      what you are doing !
-v, --verbose         It is useful only when -d is used
                      Print more information to the console

Parameters for uniform PAD encoder only:
-f, --frame-dur=DUR   Enable the uniform PAD encoder and set
                      the duration of one frame/AU in milliseconds.
-l, --label=DUR       Wait DUR seconds between each label (if
                      more than one file used)
                      Default: 12
-L, --label-ins=DUR   Insert label every DUR milliseconds
                      Default: 1200
-i, --init-burst=COUNT Sets a PAD burst amount to initially fill
                      the output FIFO
                      Default: 12
```

## 5.6 ODR-AUDIOENC

Wichtige Parametereinstellung für den ODR-AudioEncoder:

```
Welcome to ODR-AudioEnc v2.3.0, compiled at Apr  9 2018, 07:11:19
  http://opendigitalradio.org

ODR-AudioEnc v2.3.0 is an audio encoder for both DAB and DAB+.
The encoder can read from JACK, ALSA or
a file source and encode to a ZeroMQ output for ODR-DabMux.
It can also use libvlc as an input.

The -D option enables sound card clock drift compensation.
A consumer sound card has a clock that is always a bit imprecise,
  and
would drift off slowly. ODR-DabMux cannot handle such drift
because it would have to throw away or insert complete encoded audio
  frames,
which would create audible artifacts. This drift compensation can
make sure that the encoding rate is correct by inserting or deleting
audio samples. It can be used for both ALSA and VLC inputs and
  requires
a system clock synchronised using NTP.

When this option is enabled, you will see U and O printed in the
console. These correspond to audio underruns and overruns caused
by sound card clock drift. When sparse, they should not create
  audible
artifacts.

This encoder is able to insert PAD (DLS and MOT Slideshow)
generated by ODR-PadEnc.

Usage:
odr-audioenc [INPUT SELECTION] [OPTION...]
  For the alsa input:
    -d, --device=alsa_device          Set ALSA input device.
  For the file input:
    -i, --input=FILENAME              Input filename (use -i -
for stdin).
    -f, --format={ wav, raw }        Set input file format
(default: wav).
    --fifo-silence                    Input file is fifo and
encoder generates silence when fifo is empty. Ignore EOF.
  For the JACK input:
    -j, --jack=name                  Enable JACK input, and
define our name
  For the VLC input:
    -v, --vlc-uri=uri                Enable VLC input and use
the URI given as source
    -C, --vlc-cache=ms               Specify VLC network cache
length.
    -g, --vlc-gain=db                Enable VLC audio
compressor, with given compressor-makeup value.
```

```
to correct the gain for streams that are much too loud.
-V Increase the VLC verbosity
by one (can be given multiple times)
-L OPTION Give an additional options
to VLC (can be given multiple times)
-w, --write-icy-text=filename Write the ICY Text into
the file, so that ODR-PadEnc can read it.
-W, --write-icy-text-dl-plus When writing the ICY Text
into the file, add DL Plus information.
Drift compensation
-D, --drift-comp Enable ALSA/VLC sound card
drift compensation.
Encoder parameters:
-b, --bitrate={ 8, 16, ..., 192 } Output bitrate in kbps.
Must be a multiple of 8.
-c, --channels={ 1, 2 } Nbof input channels
(default: 2).
-r, --rate={ 24000, 32000, 48000 } Input sample rate
(default: 48000).
DAB specific options
-a, --dab Encode in DAB and not in
DAB+.
--dabmode=MODE Channel mode: s/d/j/m
(default: j if stereo, m
if mono).
--dabpsy=PSY Psychoacoustic model
0/1/2/3 (default: 1).
DAB+ specific options
-A, --no-afterburner Disable AAC encoder
quality increaser.
--aaclc Force the usage of AAC-LC
(no SBR, no PS)
--sbr Force the usage of SBR
(HE-AAC)
--ps Force the usage of SBR and
PS (HE-AACv2)
-B, --bandwidth=VALUE Set the AAC encoder
bandwidth to VALUE [Hz].
--decode=FILE Decode the AAC back to a
wav file (loopback test).
Output and PAD parameters:
-o, --output=URI Output ZMQ uri. (e.g.
'tcp://localhost:9000')
-or- Output file uri. (e.g.
'file.dabp')
-or- a single dash '-' to
denote stdout
If more than one ZMQ
output is given, the socket
```

```
listed endpoints.
-k, --secret-key=FILE           Enable ZMQ encryption with
the given secret key.
-p, --pad=BYTES                 Enable PAD insertion and
set PAD size in bytes.
-P, --pad-fifo=FILENAME        Set PAD data input fifo
name                            (default:
/tmp/pad.fifo).
-l, --level                     Show peak audio level
indication.
-s, --silence=TIMEOUT          Abort encoding after
TIMEOUT seconds of silence.
```

Only the tcp:// zeromq transport has been tested until now,  
but epgm://, pgm:// and ipc:// are also accepted

## 5.7 ODR-DABMux

Wichtige Parametereinstellung für den ODR-DabMultiplexer:

```
Welcome to ODR-DabMux v2.1.1, compiled at Apr  9 2018, 07:39:20

Copyright (C) 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010,
2011, 2012
Her Majesty the Queen in Right of Canada
(Communications Research Centre Canada)

Copyright (C) 2018 Matthias P. Braendli
LICENCE: GPLv3+

http://opendigitalradio.org

Input URLs supported:
prbs udp file zmq
Inputs format supported:
raw mpeg packet epm
Output URLs supported:
file fifo udp tcp simul

NAME
odr-dabmux - A software DAB multiplexer

SYNOPSIS
This software requires a configuration file:
odr-dabmux configuration.mux
See doc/example.config for an example format for the
configuration file

DESCRIPTION
odr-dabmux is a software multiplexer that generates an ETI stream
from
```

audio and data streams. Because of its software based architecture, many typical DAB services can be generated and multiplexed on a single PC platform with live or pre-recorded sources.

A DAB multiplex configuration is composed of one ensemble. An ensemble is the entity that receivers tune to and process. An ensemble contains several services. A service is the listener-selectable output. Each service contains one mandatory service component which is called primary component. An audio primary component define a program service while a data primary component define a data service. Service can contain additional components which are called secondary components. Maximum total number of components is 12 for program services and 11 for data services. A service component is a link to one subchannel (of Fast Information Data Channel). A subchannel is the physical space used within the common interleaved frame.

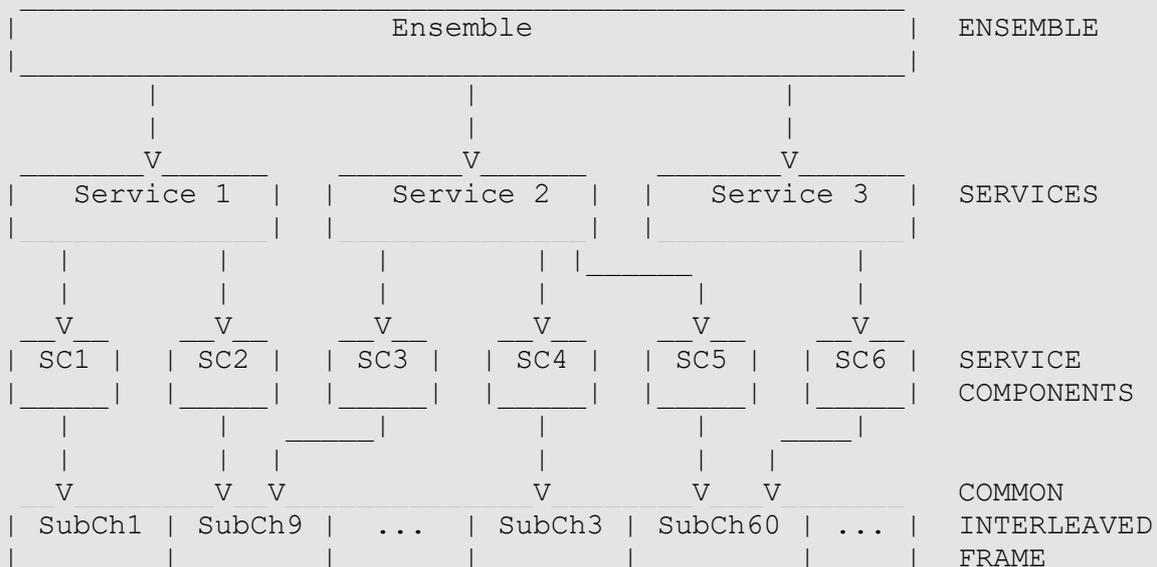


Figure 1: An example of a DAB multiplex configuration  
 ERROR Multiplex initialisation aborted: Nothing to do  
 exiting...  
 ALERT ...aborting

## 5.8 ODR-DABMOD

Wichtige Parametereinstellung für den ODR-DabModulator:

```
ODR-DabMod version v2.0.0, compiled at Apr 12 2018, 08:22:48
Compiled with features: zeromq output_uhd
ODR-DabMod version v2.0.0, compiled at Apr 12 2018, 08:22:48
Compiled with features: zeromq output_uhd
Usage with configuration file:
    odr-dabmod config_file.ini

Usage with command line options:
    odr-dabmod input (-f filename -F format | -u uhddevice -F
    frequency)
    [-G txgain] [-o offset] [-T filter_taps_file] [-a gain] [-c
    clockrate]
    [-g gainMode] [-h] [-l] [-m dabMode] [-r samplingRate]

Where:
input:          ETI input filename (default: stdin), or
                tcp://source:port for ETI-over-TCP input, or
                zmq+tcp://source:port for ZMQ input.
                udp://:port for EDI input.
-f name:       Use file output with given filename. (use /dev/stdout
                for standard output)
-F format:     Set the output format (see doc/example.ini for
                formats) for the file output.
-u device:     Use UHD output with given device string. (use for
                default device)
-F frequency: Set the transmit frequency when using UHD output.
                (mandatory option when using UHD)
-G txgain:     Set the transmit gain for the UHD driver (default: 0)
-o:           (UHD only) Set the timestamp offset added to the
                timestamp in the ETI. The offset is a double.
                Specifying this option has two implications: It
                enables synchronous transmission,
                requiring an external REFCLK and PPS signal and
                frames that do not contain a valid timestamp
                get muted.

-T taps_file:  Enable filtering before the output, using the
                specified file containing the filter taps.
                Use 'default' as taps_file to use the internal taps.
-a gain:      Apply digital amplitude gain.
-c rate:      Set the DAC clock rate and enable Cic Equalisation.
-g gainmode:  Set computation gain mode: fix, max or var
-h:          Print this help.
-l:          Loop file when reach end of file.
-m mode:      Set DAB mode: (0: auto, 1-4: force).
-r rate:      Set output sampling rate (default: 2048000).
```

## 5.9 KONFIGURATIONSDATEI MULTIPLEX

```
general {
    dabmode 1
    nbframes 0
    syslog false
    writescca false
    tist false
    managementport 1337
}

remotecontrol {
    telnetport 0
}

ensemble {
    id 0xD101
    ecc 0xE0
    international-table 1
    local-time-offset auto
    label "LMK-Radio"
    shortlabel "LMK"
}

services {
    SV_Radio_1 {
        id 0xD102
        label "Radio Soundcard"
        shortlabel "RaSound"
        language 0x08
        pty 10
    }

    SV_Radio_2 {
        id 0xD103
        label "Radio VLC"
        shortlabel "RaVLC"
        language 0x08
        pty 10
    }
    SV_Radio_3 {
        id 0xD104
        label "DasDing"
        shortlabel "DD"
        language 0x08
        pty 10
    }
}

subchannels {
    SC_Radio_1 {
        type dabplus
        inputfile "tcp://*:9001"
        bitrate 96
    }
}
```

```
    protection-profile EEP_A
    protection 1
    zmq-buffer 96
    zmq-prebuffering 48
  }

  SC_Radio_2 {
    type dabplus
    inputfile "tcp://*:9002"
    bitrate 96
    protection-profile EEP_A
    protection 1
    zmq-buffer 96
    zmq-prebuffering 48
  }

  SC_Radio_3 {
    type dabplus
    inputfile "tcp://*:9003"
    bitrate 96
    protection-profile EEP_A
    protection 1
    zmq-buffer 96
    zmq-prebuffering 48
  }
}

components {
  CP_Radio_1 {
    service SV_Radio_1
    subchannel SC_Radio_1
    type 5
    id 0x001
    figtype 0x2
  }

  CP_Radio_2 {
    service SV_Radio_2
    subchannel SC_Radio_2
    type 5
    id 0x002
    figtype 0x2
  }

  CP_Radio_3 {
    service SV_Radio_3
    subchannel SC_Radio_3
    type 5
    id 0x003
    figtype 0x2
  }
}

outputs {
  zeromq {
```

```
    endpoint "tcp://*:18081"  
    allowmetadata false  
  }  
  
  ETI_DabLin "fifo://./LMK.eti?type=raw"  
  
  throttle "simul://"  
}
```

## 5.10 KONFIGUATIONSDATEI B100.INI

```
[remotecontrol]  
telnet=0  
telnetport=0  
zmqctrl=0  
zmqctrlendpoint=tcp://127.0.0.1:0  
  
[log]  
syslog=0  
filelog=0  
filename=/dev/stderr  
  
[input]  
transport=zeromq  
source=tcp://localhost:9100  
max_frames_queued=100  
  
[modulator]  
gainmode=2  
digital_gain=0.8  
rate=2048000  
dac_clk_rate=0  
  
[firfilter]  
enabled=0  
  
[output]  
output=uhd  
  
[uhdoutput]  
master_clock_rate=32768000  
device=  
type=b100  
txgain=2.0  
frequency=223936000  
refclk_source=internal  
pps_source=none  
behaviour_refclk_lock_lost=ignore  
offset=0.0
```

```
[delaymanagement]
synchronous=0
mutenotimestamps=0
offset=0.0
```

## 5.11 DABLIN

```
DABlin v1.7.0 - capital DAB experience
Plays a DAB/DAB+ audio service from a frame-aligned ETI-NI stream.
https://github.com/Opendigitalradio/dablin

Usage: dablin [OPTIONS] [file]
-h                Show this help
-d <binary>      Use DAB live source (using the mentioned binary)
-c <ch>          Channel to be played (requires DAB live source)
-s <sid>         ID of the service to be played
-x <scids>       ID of the service component to be played (requires
                 service ID)
-r <subchid>     ID of the sub-channel (DAB) to be played
-R <subchid>     ID of the sub-channel (DAB+) to be played
-g <gain>        USB stick gain to pass to DAB live source (auto gain
                 is default)
-p              Output PCM to stdout instead of using SDL
file            Input file to be played (stdin, if not specified)
```

---

## 6 VERZEICHNISSE

---

### 6.1 ABBILDUNGSVERZEICHNIS

Abbildung 1: Blockdiagramm Audio .....	6
Abbildung 2: Blockdiagramm Netzwerk.....	6
Abbildung 3: Blockdiagramm Video.....	7
Abbildung 4: Softwarekonzept DAB-Toolbox-Sender .....	22
Abbildung 5: Innenaufbau DAB-Toolbox-Sender .....	23
Abbildung 6: Rechte Seite Toolbox-Sender .....	24
Abbildung 7: linke Seite Toolbox-Sender .....	24
Abbildung 8: Obere Seite Toolbox-Sender.....	25
Abbildung 9: Vorderseite Toolbox-Sender .....	25
Abbildung 10: Netzwerkplan.....	29
Abbildung 11: : DHCP/LAN-Einstellung EDIMAX.....	31
Abbildung 12 WSIP-Einstellung EDIMAX-Router .....	32
Abbildung 13 Select Site Survey.....	32
Abbildung 14: Öffnen von Dateien .....	34
Abbildung 15: Dateien hinzufügen .....	35
Abbildung 16: Output Module wählen .....	36
Abbildung 17: Automatisches Verbinden mit JACK-Port .....	37
Abbildung 18: DABlin .....	43
Abbildung 19: Weboberfläche Supervisor .....	48
Abbildung 20: Weboberfläche EasyDAB .....	51
Abbildung 21: Weboberfläche EasyDAB .....	51

### 6.2 TABELLENVERZEICHNIS

Tabelle 1: Anmeldedaten Raspbian .....	26
Tabelle 2: Zuordnung der IP-Adressen zu den Geräten.....	29
Tabelle 3: Anmeldedaten Router-WLAN.....	30
Tabelle 4: Anmeldedaten zur Router-Konfigurierung.....	30
Tabelle 5: Anmeldedaten EasyDAB.....	50

### 6.3 QUELLENVERZEICHNIS

- [1] Michael Kröger. (2014, Aug.) drm-radio-kl.eu. [Online]. [http://www.drm-radio-kl.eu/berichte\\_vortraege/koffersender\\_2013-15/Abschlussbericht\\_V3.0\\_final.pdf](http://www.drm-radio-kl.eu/berichte_vortraege/koffersender_2013-15/Abschlussbericht_V3.0_final.pdf)
- [2] Friedrichsen Immanuel. (2018, Apr.) DRM Radio KL. [Online]. [http://www.drm-radio-kl.eu/berichte\\_vortraege/koffersender\\_2016/Praxisbericht\\_Aufbau-DAB-Koffersender-V2.pdf](http://www.drm-radio-kl.eu/berichte_vortraege/koffersender_2016/Praxisbericht_Aufbau-DAB-Koffersender-V2.pdf)
- [3] (2018, Mar.) RaspberryPi.org. [Online]. <https://www.raspberrypi.org/documentation/>
- [4] Sergiy. (2018, Mar.) Tipok.org. [Online]. <http://tipok.org.ua/node/46>
- [5] Olmatic. (2018, Mar.) [Online]. [http://seprotronic.com/susv/files/pdf/SUSV\\_Manual\\_Rev2\\_0\\_EN.pdf](http://seprotronic.com/susv/files/pdf/SUSV_Manual_Rev2_0_EN.pdf)
- [6] Messrs G. Spikofski (IRT) & F. Camerer (ORF). (2011, September) Lautheitsaussteuerung, Normalisierung. [Online]. <https://tech.ebu.ch/docs/r/r128-2014.pdf>
- [7] ETSI EN 300 798. (2018, Apr.) ETSI. [Online]. [http://www.etsi.org/deliver/etsi\\_en/300700\\_300799/300798/01.01.01\\_60/en\\_300798v010101p.pdf](http://www.etsi.org/deliver/etsi_en/300700_300799/300798/01.01.01_60/en_300798v010101p.pdf)
- [8] Immanuel Friedrichsen. (2018, Feb.) drm-radio-kl.eu. [Online]. [http://drm-radio-kl.eu/berichte\\_vortraege/koffersender\\_2016/GUI4ODR\\_Manual-DE-14042017.pdf](http://drm-radio-kl.eu/berichte_vortraege/koffersender_2016/GUI4ODR_Manual-DE-14042017.pdf)

