Karlsruher Institut für Technologie
Communications Engineering Lab
Univ.-Prof. i.R. Dr.rer.nat. Friedrich K. Jondral

Write my thesis

Example Template

Moritz Luca Schmid

Hauptreferent	:	Dr.-Ing. Holger Jäkel
Betreuer	:	M.Sc. Felix Wunsch

Beginn	:	12.06.2017
Abgabe	:	12.12.2017

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig und unter Beachtung der Satzung des Karlsruher Instituts für Technologie (KIT) zur Sicherung guter wissenschaftlicher Praxis in der aktuellen Fassung angefertigt habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und wörtlich oder inhaltlich übernommene Stellen als solche kenntlich gemacht.

Karlsruhe, den 31.11.2015

Moritz Luca Schmid

Abstract

This is my abstract. CEL thesis rules require it to be about 3-5 pages. It is a summary of what you do in your thesis. Use around 5 pictures and outline whatever you did. And now a few lines of information.

This is a to-do example

Inhaltsverzeichnis

1. Einführung

Digital Audio Broadcasting (DAB) ist ein Übertragungsstandard zur Verbreitung von digitalem Radio. Das System wurde in den 90er Jahren im Zuge des Eureka 147 DAB Projekts entwickelt und kommt seit 1997 in vielen Teilen Europas und Asiens zum Einsatz. Mit dem DAB System soll langfristig die analoge Am/Fm Übertragung ersetzt werden.
Die digitale Datenübertragung bietet dabei eine verbesserte Audioqualität, da Übertragungsfehler durch Codierung empfängerseitig korrigiert werden können und somit nicht zu einem verrauschten Audiosignal wie bei FM führen. Eine Audiokompression verringert zusätzlich die Datenrate, was sich in einer gesteigerten Frequenzeffizienz pro Audiokanal widerspiegelt. Die digitale Übertragung bietet außerdem viele neue Möglichkeiten der medialen Unterstützung in Form von Service Informationen wie Albumcovers, ausführliche Stauinformationen oder Wetterkarten.
DAB Sender sind in sog. Ensembles strukturiert. Ein DAB Ensemble enthält einen ganzen Multiplex an Audio- sowie Datenkanälen. Sender dieses Ensembles greifen auf eine Auswahl dieser Kanäle zu. Dieser dynamische Multiplex erlaubt eine flexible und individuelle Programmgestaltung.
DAB stellt vier verschiedene Übertragungsmodi zur Verfügung, die für verschiedene Übertragungsszenarien und Frequenzbereiche ausgelegt sind. Der Fokus liegt dabei auf dem mobilen Empfang bei terrestrischer Übertragung, es existieren aber auch jeweils ein Modus für die Satellitenübertragung sowie die niederfrequente Übetragung per Kabel. Eine Besonderheit bei DAB stellt der Einsatz von sog. Single Frequency Networks (SFNs) dar, bei denen eine Vielzahl von örtlich getrennten Sendestationen auf der gleichen Frequenz ausstrahlen wodurch ein enormer Gewinn an Frequenzeffizienz erzielt werden kann. Der Einsatz von SFNs wird bei DAB durch eine hohe Robustheit des Systems gegenüber Mehrwegeempfang ermöglicht [**?**].
Ziel dieser Arbeit ist die Implementation und Evaluation eines Senders und Empfängers für DAB sowie DAB+. Die Implementation soll dabei in Form eines Software Radios erfolgen, einem System bei dem ein Großteil der Signalverarbeitung auf Softwareebene durchgeführt wird. Dazu wird die Open-Source Software GNU Radio verwendet. Im Weiteren soll eine grafische Transceiver Applikation realisiert werden, die auf Basis des implementierten Senders und Empfängers eine benutzerfreundliche Oberfläche für das Senden und den Empfang von DAB sowie DAB+ Signalen darstellt.

schon OFDM und andere techniken Erwähnen, wie ANdr Mueller?

2. DAB Standard

In diesem Kapitel werden die verwendeten Strukturen und Verfahren vorgestellt, die nötig sind um ein funktionierendes DAB Übertragungssystem aufzubauen. Eine vollständige und detailliertere Beschreibung über den kompletten DAB Standard liefert das vom Europäischen Institut für Telekommunikationsnormen (ETSI) veröffentlichte Dokument [**?**]. Die in Abschn. 1 angesprochenen Übertragungsmodi basieren alle auf dem selben Prinzip und sind nur quantitativ auf ihre jeweiligen Einsatzbereiche angepasst. Diese Arbeit bezieht sich bei quantitativen Betrachtungen immer auf den Übertragungsmodus I, der bei der terrestrischen Übertragung verwendet wird.

2.1. Aufbau eines DAB Ensembles

Der Aufbau eines DAB Services gestaltet sich wesentlich komplexer, aber auch flexibler als die feste Struktur einer FM Übertragung, bei der jeder Radiosender genau einen Audiokanal überträgt und ein eigenes Frequenzband belegt. Ein DAB Ensemble beinhaltet in der Regel mehrere Radiosender, die wiederum aus vielen Audio- und Datenkanälen bestehen können. Es erfolgt also eine Trennung von Radiosender und Audio-/Datenkanal. Ein Beispiel für ein DAB Ensemble zeigt Abb. **??**.

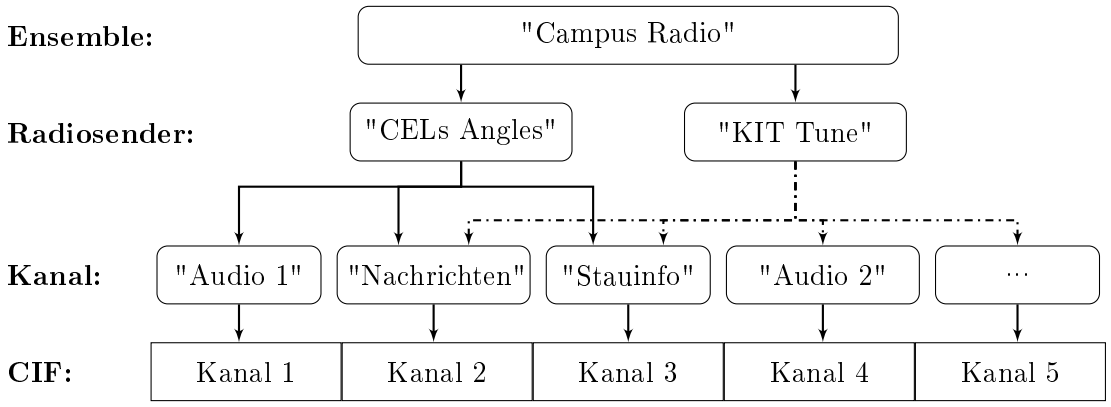


Abbildung 2.1.: Beispielhafte Struktur eines DAB Ensembles

Diese Trennung erlaubt es besipielsweise, dass mehrer Radiosender den selben Datenkanal, der zum Beispiel Verkehrsinformationen transportiert, gleichzeitig zu nutzt. Dies spart auf der auf der einen Seite eine redundante Datenübertragung und ermöglicht es den Radiosendern zudem, ihr Programm bzw. Angebot breiter zu fächern bzw. flexibel anzupassen. Alle Kanäle der verschiedenen Radiosender werden im Main Service Multiplexer zu einem Common Interleaved Frame (CIF) gebündelt und übertragen. Ohne Kenntnis über den

Aufbau und die Struktur des Mulitplex, ist ein CIF am Empfänger nutzlos, da keine Information über die Lage einzelner Audio Streams im Multiplex bekannt ist. Die Übertragung dieser Multiplex Configuration Information (MCI) ist Aufgabe des Fast Information Channel (FIC).

Struktur mit Ensemble Services als Grafik

2.2. Fast Information Channel (FIC)

Der FIC spielt bei der Übertragung eines DAB Ensembles eine sehr wichtige Rolle, da er sowohl die MCI als auch Service Information (SI), wie zum Beispiel den Namen eines Radiosenders, überträgt. Die Informationen werden Paketweise in sogenannten Fast Information Blocks (FIBs) übertragen. Die Bedeutung der MCI wurde schon erläutert und ist naheliegend, aber auch die SI ist von großer Bedeutung, da ein Nutzer das Radioprogramm nicht ohne dessen Namen auswählen kann.
Die Übertragung des FIC erfordert daher eine hohe Robustheit was durch eine gute Kanalcodierung sichergestellt wird.

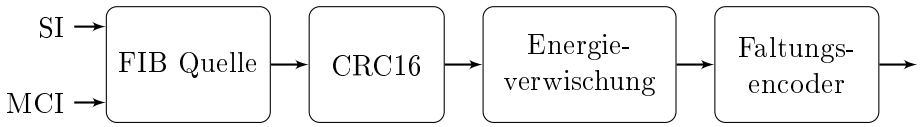


Abbildung 2.2.: Quelle und Kanalcodierung des FIC

2.2.1. CRC

Jedem FIB der Länge 30 bytes (bestehend aus Fast Information Groupt (FIG)) wird ein 16 bit CRC Wort angehängt. Das CRC Wort wird über die 30 Nutzbytes mittels des Generatorpolynoms

$$G(x) = x^{16} + x^{12} + x^5 + 1 \tag{2.1}$$

berechnet. Das CRC Wort bietet keine Möglichkeit zur Fehlerkorrektur sondern dient lediglich der Fehlerdetektion, wofür es nach [?] optimiert ist.

2.2.2. Energieverwischung

Im nächsten Schritt wird sichergestellt, dass die Binärquelle ideale Eigenschaften erfüllt. Dazu müssen die zu übertragenden Bits gleichverteilt sein um die Entropie der Quelle zu maximieren und zyklischen Wiederholungen von Bitsequenzen durch etwaige Wiederholungen von FIGs müssen vermieden werden. Der Bitstream wird dazu über eine Modulo-2 Operation mit einer Pseudo-Random Binary Sequency (PRBS) gescrambled. Die PRBS erfüllt die gewünschten Eigenschaften und ist durch das Polynom

$$P(x) = X^9 + X^5 + 1 \tag{2.2}$$

im Empfänger exakt reproduzierbar.

2.2.3. Faltungscodierung

Eine Kanalcodierung ermöglicht die Erkennung und Korrektur von Bitfehlern auf Kosten einer geringeren Nutzdatenrate. Das DAB System verwendet eine Faltungscodierung mit der Coderate $R = 1/4$ und einer Einflusslänge von 7 Bits. Für jedes Eingangsbit produziert der Encoder also ein Codewort der Länge 4 bit, das über die Polynome

$$\begin{aligned} g_1(x) &= 1 + x^2 + x^3 + x^5 + x^6 \\ g_2(x) &= 1 + x + x^2 + x^3 + x^6 \\ g_3(x) &= 1 + x + x^4 + x^6 \\ g_4(x) &= 1 + x^2 + x^3 + x^5 + x^6 \end{aligned} \tag{2.3}$$

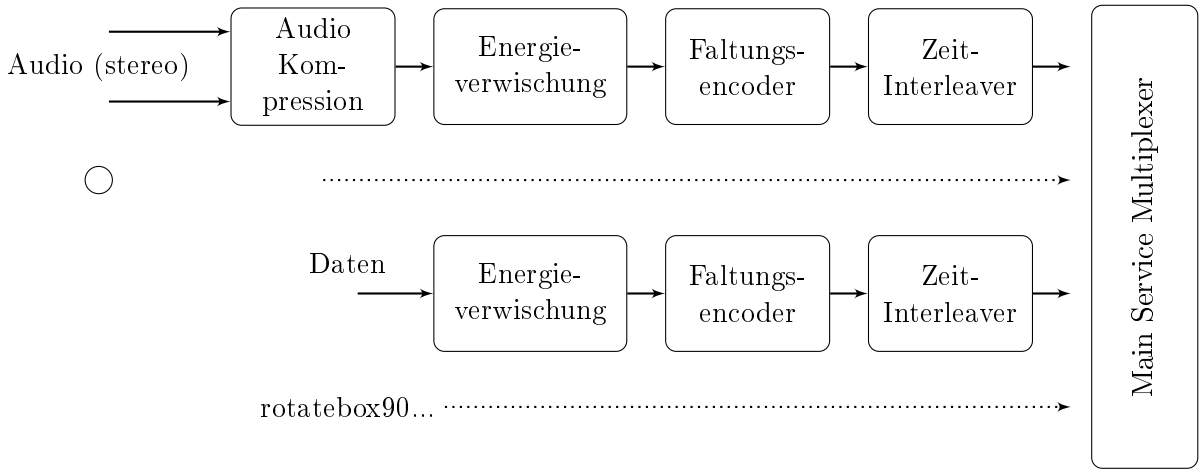
berechnet werden.
Um die Coderate R zu erhöhen wird die punktierte Faltungscodierung verwendet, bei der aus einer Gruppen von Codebits ein Teil der Bits wieder gestrichen (punktiert) wird. Dadurch ergeben sich eine Vielzahl an möglichen Coderaten

reinbringen, dass Block-weise? Lese Friedrichs

$$R_{punktiert} = \frac{8}{8 + PI} \text{mit} PI \in [1; 24] \tag{2.4}$$

wobei $R = 1/4 \leq R_{punktiert} \leq 1$ gilt.
Der FIC verwendet eine konstante Coderate von $R_{punktiert,FIC} = 1/3$.

2.3. MSC



logical Frames, Audio und Data, Packed und Stream

2.3.1. Audio Kompression

Die subjektiv empfundene Qualität der übertragenen Audio Streams soll der einer Audio CD gleichen. Audio CDs verwenden 16 PCM pro Monokanal bei einer Samplerate von $44{,}1kHz$. Dies entpricht einer gesamten Bitrate von $1{,}41Mbit/s$ für eine einzelne Audioübertragung

mit Stereokanälen. Nimmt man nun noch eine Faltungscodierung mit exemplarischer Coderate von $R = 1/2$ in die Rechnung auf, übersteigt schon diese einzelne Audioübertragung die Gesamtkapazität des Main Service Channel (MSC) von $2{,}304 Mbit/s$ (siehe 2.3.5). Um daher eine Vielzahl von Audiokanälen im MSC unterbringen zu können, ist die Notwendigkeit einer Audiokompression ersichtlich, deren Kompressionsrate etwa einer Größenordnung entsprechen muss. Die verwendete Audiokompression ist MPEG 1/2 Audio Layer II (MPEG 2) [?] für den DAB Standard und MPEG4 HE-AACv2 für DAB+.

schon erklärt was DAB+ ist?! (bzw DAB+ erst nach erklärung von MPEG 2 bringen?

MPEG 2 ist ein generischer Audiocodec der verschiedene Effekte zur verlustbehafteten Kompression verwendet. Zum einen wird die Redundanz des Audiosignals reduziert, indem durch statistische Korrelation eine Vorhersage über das Zeitsignal getroffen werden kann [?]. Zudem verwendet MPEG 2 ein psychoakustisches Modell des menschlichen Ohres. Damit ist es möglich die spektrale Hörschwelle an einem bestimmten Zeitpunkt zu bestimmen. Diese Information kann genutzt werden, um das in 32 Frequenzbänder zerteilte Signal in Abhängigkeit der temporären Empfindlichkeit des Ohres zu Quantisieren und somit das Quantisierungsrauschen in jedem Frequenzband knapp unter der Hörschwelle zu halten. Abb. 2.3 zeigt dieses Prinzip als Blockschaltbild.

Nochmal blöcke oder iwas erklären? Masking reinbringen??

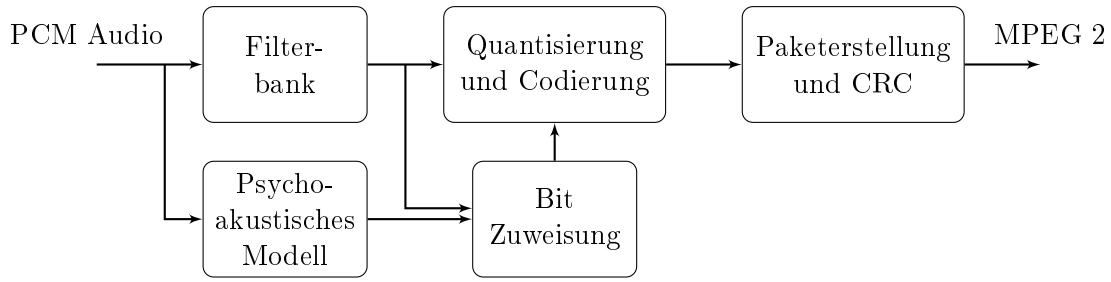


Abbildung 2.3.: Blockdiagramm der MPEG 2 Kompression

Mit MPEG 2 lässt sich die erwähnte CD Qualität subjektiv mit einer Bitrate von etwa 192 kbit/s erreichen. Eine noch stärkere Komprimierung bietet MPEG 4 HE-AACv2 (MPEG 4), welches nach subjektiven Hörtests dieselbe Qualität bei einer geringeren Daterate von 96 - 128 kbit/s erreich [?]. MPEG 4 stellt eine eine Weiterentwicklung von MPEG 2 dar und basiert auf dem selben Prinzip. Um die Datenrate noch weiter zu reduzieren werden Verfahren wie Spektralbandrepilaktion und Kanalkopplung (Joint-Stereo) genutzt. Der MPEG 4 Codec wurde zusammen mit einer zusätzlichen Reed-Solomon Fehlerkorrektur in den DAB Standard integriert und wird in Deutschland seit 2011 als "DABplusäusgestrahlt.

Spektralbandrepilaktion und Kanlkopplung erklären? Mehr zu RS?

2.3.2. Energieverwischung

Eine Energieverwischung wird wie in Abschn. 2.2.2 auch im MSC durchgeführt.

2.3.3. Faltungscodierung

Die in Abschn. 2.2.3 beschriebene Faltungscodierung mit anschließender Punktierung kann im MSC genutzt werden, um die Coderate jedes einzelnen Kanals individuell einzustellen. Dabei stehen sog. Protection Level zur Verfügung die durch eine Kombination verschiedener Punktierungsmuster Coderaten von $R = 1/4$ bis $4/5$ erzeugen.

Neben der Equal Error Protection (EEP) gibt es zudem die Möglichkeit einzelne Bit-Blöcke innerhalb eines Frames stärker zu codieren als andere. Die Unequal Error Protection (UEP) ist zum Beispiel für MPEG Audio Frames sehr sinnvoll um die für die subjektive Audioqualität (wenigen) wichtigen Bits mit einer sehr hohen Redundanz zu versehen ohne dabei die Gesamtcoderate deutlich zu erhöhen. Ein Bitfehler im Header kann beispielsweise zu einer Fehlskalierung eines ganzen Frequenzbandes führen was sich beim Hörer als unangenehmes Pfeifen (sog. 'birdies') auswirkt, wohingegen ein einzelner Bitfehler des Zeitsignals unbemerkt bleibt.

2.3.4. Zeit-Interleaving

Die Faltungscodierung kommt schnell an ihre Grenzen, wenn anstelle von einzelnen Bitfehlern ganze Bündelfehler auftreten. Solche zeitlich begrenzten Empfangsausfälle entstehen zum Beispiel bei mobilen Empfängern in Bewegung durch Fadingeffekte. Im MSC befindet sich mit dem Zeit-Interleaving deshalb ein zusätzliches Element in der Kanalcodierungskette, das die nicht-korrigierbaren Bündelfehler über die Zeit verteilt wodurch viele korrigierbare Einzelbitfehler entstehen. Der Zeit-Interleaver verzögert die Bits eines Frames nach den Regeln eines Interleaving Vektors um 0 bis 15 Logische Frames, was einer maximalen Verzögerung von $T_{delay,max} = 15 \cdot 24ms = 360ms$ nach sich zieht. $T_{delay,max}$ entspricht also der maximalen Länge von Bündelfehlern die vom Zeit-Interleaver verteilt werden können. Gleichzeitig ist $T_{delay,max}$ aber auch die Verzögerung der Gesamtübertragung, da ein Frame erst decodiert werden kann, wenn alle Elemente (die maximal eine Verzögerung von $T_{delay,max}$ haben) empfangen wurden. Diese konstante Verzögerung wird im MSC für die verbesserte Fehlerkorrektur in Kauf genommen. Im FIC ist solch eine Verzögerung nicht möglich, da eventuell zeitkritische Daten wie zum Beispiel die Umkonfiguration des DAB Multiplexes übertragen werden, weshalb beim FIC kein Zeit-Interleaving angewendet wird.

wirklich, oder weniger?

2.3.5. Multiplexing

Im Multiplexer werden alle Kanäle des MSC gebündelt. Dabei beinhaltet das resultierende CIF jeweils ein logisches Frame aus jedem Kanal, also Daten von $24ms$ jedes Streams. Das CIF wird in sog. Capacity Unit (CU) von jeweils $64bit$ partitioniert und besitzt eine Gesamtkapazität von $864\text{CU}s$, also $55296bit$. Über die CUs erfolgt auch die Adressierung der einzelnen Kanäle, was den wichtigsten Teil der in 2.2 beschriebenen MCI ausmacht.
Die Kapazität des CIFs ist groß genug um eine Vielzahl von Audio- und Datenkanälen zu transportieren. Ein typischer [1] $16bit$ PCM Stereo Audio Stream mit der Abtastrate $32kHz$ hat nach der Komprimierung eine Bitrate von $2 \cdot 64kbit/s$. Bei einem Protection Level von A3 ($\hat{=}$ Coderate $R = 1/2$) können also beispielsweise 10 Audiostreams parallel übertragen werden. Wird das CIF nicht komplett mit Kanälen gefüllt, werden die unbesetzten CUs mit einer PRBS (siehe 2.2.2) aufgefüllt um ein ungleiche Energieverteilung im CIF zu verhindern.

Auf DAB Ensemble beziehen, das weiter oben erwähnt werden muss

2.4. OFDM Modulation

anderer Titel, sowas wie Sendesignal

[1] Bezogen auf ein in Karlsruhe (Deutschland) empfangenes DAB+ Ensembles des Südwestrundfunks.

2.4.1. Partitionierung des OFDM Frames

Die Daten aus FIC und MSC werden vor der OFDM Modulation jeweils zu einem Sendeframe zusammengefasst. Ein Frame enthält dabei 4 CIFs und deren zugehörigen FIBs, was einer Streamdauer von $4 \cdot 24ms = 96ms$ entspricht. Die Strukturierung des Sendeframes erfolgt in sogenannten OFDM Symbolen, die Pakete von jeweils 1536 QPSK Symbolen darstellen. $K = 1536$ entspricht dabei der Anzahl der genutzten OFDM Unterträger (siehe Abschn. 2.4.5). Zusätzlich zu den Datensymbolen beinhaltet jedes Frame noch das NULL Symbol und das Phasenreferenzsymbol für Synchronisationszwecke.

Grafik mit Frameaufteilung

NULL Symbol

Das erste OFDM Symbol jedes Sendeframes m ist das Nullsymbol $z_{0,k}$, für das

$$S(t) = 0 \text{ für } t \in [0, T_{NULL}] \tag{2.5}$$

gilt. Diese Energielücke von bekannter Dauer $T_{NULL} = 1{,}297ms$ stellt eine Möglichkeit für eine sehr einfache und robuste grobe Zeitsynchronisation auf die Sendeframes dar.

gescheiten zeitparameter t, zumindest durchgängig gleich

Phasenreferenzsymbol

Das Phasenreferenzsymbol $z_{1,k}$ bildet das zweite Symbol jedes Frames. Es enthält eine vordefinierte Sequenz an komplexen Constant Amplitude Zero Auto-Correlation (CAZAC) Symbolen, deren Amplitude stets konstant, und deren Autokorrelation stets null ist. Das Phasenreferenzsymbol dient zum einen, namensgebend, als Bezugsphase für differentielle Modulation der nachfolgenden D-QPSK Datensymbole. Zum anderen eignet es sich aufgrund seiner CAZAC Eigenschaften und der Tatsache, dass das Phasenreferenzsymbol auch im Empfänger bekannt ist, ideal für eine feine Zeitsynchronisation sowie für eine feine und grobe[2] Frequenzsynchronisation über Korrelation.

Ansatz von dab buch mit matched filter hier oder in der implementierung??

2.4.2. QPSK Modulation

Die Bits des FIC und MSC werden mit QPSK moduliert. Dabei werden $2K = 3072$ Bits ($p_{l,n}$ für $n \in [0; 2K)$) zu $K = 1536$ QPSK Symbolen (q_l,n für $n \in [0; K)$) moduliert, wobei die erste Hälfte der Bits die Realteile $n \in [0; K)$ und die zweite Hälfte die Imaginärteile $n \in [K; 2K)$ bilden, sodass sich die modulierten Symbole zu

$$q_{l,n} = \frac{1}{\sqrt{2}} \left[(1 - 2p_{l,n}) + j\,(1 - 2p_{l,n+K}) \right] \tag{2.6}$$

ergeben.

evt. Bild mit Contellationsdiagramm

2.4.3. Frequenzinterleaving

Die modulierten OFDM Unterträger werden vor der OFDM Modulation durch eine pseudozufällige Folge untereinander vertauscht. Bei langsamer Bewegung oder Stillstand führt ein Mehrwegeempfang zu langsamem selektivem Fading. Daraus resultierende Bündelfehler

[2] Nach [?] kann durch die CAZAC mittels einer AFC (Automatic Frequency Conrol) ein Frequenzoffset von $\pm 32kHz$ ($\hat{=} \pm 32\, Untertrger$) eindeutig erkannt werden.

auf einzelnen Unterträgern übersteigen die Einflussdauer des Zeitinterleavers. Durch das Vertauschen der Unterträger werden die Effekte eines solchen frequenzselektives Fadings minimiert.

samples gehen durch freq-interleaving von q zu y über

2.4.4. Differetielle Modulation

Die QPSK modulierten und frequenzvertauschten Unterträger werden in einem letzten Schritt vor der OFDM Modulation differentiell codiert. Die Phase jedes Sendesymbols ergibts sich aus der Summe Phase des Symbols zur Phase dessen Vorgängers.

$$\begin{aligned} z_{l,k} =& z_{l-1,k} \cdot y_{l,k} = e^{j\varphi_z} \cdot e^{j\varphi_y} \\ & \text{mit } \varphi_z \in \{n \cdot \frac{\pi}{4} : l \text{ gerade}\} \cup \{\frac{\pi}{4} + n \cdot \frac{\pi}{2} : l \text{ ungerade}\}, n \in \mathbb{N} \end{aligned} \tag{2.7}$$

Die zu übertragende Information wandert also von der eigentlichen Phase des Symbols in die relative Änderung der Phase des Vorgängersymbols zum aktuellen Symbol. Der Vorgänger des ersten Datensymbols $z_{1,k}$ eines jeden Frames ist das Phasenreferenzsymbol $z_{0,k}$ aus 2.4.1. Die differentielle Modulation hat den großen Vorteil, dass der Empfänger den Übertragungskanal nicht kennen bzw. nicht schätzen muss. Beispielsweise habt sich eine konstante Phasendrehung im Kanal durch die Differenzbildung im Empfänger wieder auf.

2.4.5. OFDM

Das DAB System verwendet ein Frequenzmultiplexverfahren (FDM) zur Übertragung. Dabei werden die D-QPSK Symbole $z_{l,k}$ nicht seriell mit einer Symbolrate von $1{,}2 MSamples/s$ übertragen, sondern die zu übertragenden Symbole werden auf $N = 2048$ Unterträgern verteilt. Die gesamte Bandbreite $B = 2{,}048 MHz$ wird also unter den einzelnen Unterträgern k gleichmäßig aufgeteilt, sodass sich ein Unterträgerabstand von $\Delta f = \frac{B}{N} = 1 kHz$ ergibt. Diese schmalbandigen und niederratigen Unterträger haben in der Summe die gleiche Symbolrate wie eine serielle Übertragung, sie bieten aber vorteilhafte Übertragungseigenschaften:

- Der Übertragungskanal kann pro Unterträger als flach angenommen werden.
- Intersymbol Interferenz (ISI) wird wegen der deutlich größeren Symboldauer im Vergleich zur Seriellübertragung stark reduziert.

Das resultierende Basisbandsignal $S(t)$ ergibt sich dann als Summe der einzelnen Unterträgern, die mit dem Abstand Δf über dem Basisband verteilt sind.

$$S(t) = \frac{1}{K} \sum_{k=-K/2}^{K/2} z_{l,k}\, g_{k,l}(t - lT_S)\, e^{j2\pi k \Delta f t} \tag{2.8}$$

n=0 nicht belegt wegen DC Anteil!!

Die Symbole werden dabei mit $g_{l,k}(t)$ pulsgeformt. Damit sich die Unterträger nicht gegenseitig beeinflussen und Inter-Träger-Interferenzen (ICI) die Performanz des Systems beschneidet, werden die Unterträger orthogonal zueinander gewählt (orthogonales FDM

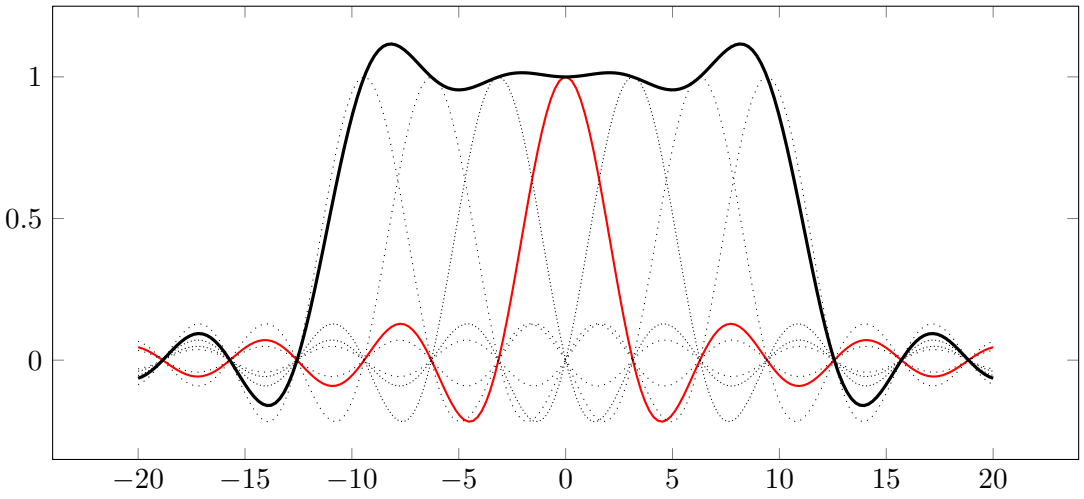
(OFDM)). Das kann durch eine rechteckige Pulsformung

$$g_{k,l}(t) = Rect(t/T_S) = \begin{cases} 1, \text{für } 0 \leq t < T_S \\ 0, \text{sonst} \end{cases}$$

$$\circ\!\!-\!\!\bullet$$

$$G_{k,l}(f) = T_S si\left(\pi f T_S\right) e^{-j\pi f T_S} \tag{2.9}$$

erreicht werden. Wie in Abb. 2.4.5 zu sehen ist, haben die si-Funktionen, die sich nach 2.9 aus den Rechteckfunktionen im Frequenzbereich ergeben, jeweils Nullstellen bei ganzzahligen Vielfachen von Δf, also genau an den Maxima aller anderen Unterträger. Diese Orthogonalität vermeidet ICI, unter der Annahme dass die Unterträger an ihrer exakten Mittenfrequenz abgetastet werden.



Von den $N = 2048$ möglichen Unterträgern sind im DAB System nur $K = 1536$ belegt. Zum einen wurde der zentrale Unterträger ($k = 0 \Leftrightarrow k \cdot \Delta f = 0kHz$) nicht belegt um einen Gleichanteil im Basisbandsignal zu vermeiden. Die restlichen unbelegten Unterträger liegen am rechten und linken Rand des Spektrums und bilden damit ein Schutzintervall gegenüber benachbarten Übertragungen. Das gesamte Sendesignal ergibt sich letztendlich nach dem Hochmischen auf die Trägerfrequenz f_c zu dem Bandpasssignal

$$S_{BP}(t) = \mathrm{Re}\left\{ e^{j2\pi f_c t} \sum_{m=-\infty}^{\infty} \sum_{l=0}^{L} \sum_{k=-\frac{K}{2}}^{\frac{K}{2}} z_{m,l,k}\, g_{k,l}(t - mT_F - T_{NULL} - (l-1)T_S)\, e^{j2\pi k \Delta f t} \right\} \tag{2.10}$$

evt noch Guard intervall in Formle aufnehmen (t zu (t-guard time))

Wie in [?] gezeigt wird, kann eine Inverse Diskrete Fourier Transformation (IDFT) verwendet werden um das OFDM-Sendesignals zu erzeugen.

3. Implementierung eines DAB/DAB+ Transceivers

3.1. GNU Radio

GNU Radio ist eine kostenlose und weitverbreitete Open-Source Software zur Programmierung und Ansteuerung von Software Defined Radio (SDR). Software Radio bezeichnet dabei ein Funksystem, welches die Signalverarbeitung anstatt durch Hardware (integrierte Schaltkreise) auf Softwareebene durchführt. Software hat den Vorteil, dass sie variabel und schnell austauschbar ist. Dadurch ist es möglich mit derselben Hardware viele verschiedene Funksysteme zu realisieren.
GNU Radio verwendet dabei das Konzept von funktionalen Blöcken, die verbunden werden und damit eine Signalverarbeitungskette, einen sog. Flowgraph, bilden. Die einzelnen Blöcke führen dabei jeweils eine spezifische Signalverarbeitungsoperation möglichst recheneffizient durch. Dabei stellt ein GNU Radio Block in den meisten Fällen eine eigene C++ Klasse dar. Die Ablaufsteuerung, also die Instanzierung und Anordnung der Blöcke in einem Gesamtsystem, geschieht in der Skriptsprache Python. Dazu werden die verwendeten C++ Klassen mit dem Tool Simplified Wrapper and Interface Generator (SWIG) nach Python übersetzt.
Neben der Möglichkeit, die GNU Radio Blöcke selbst zu entwerfen, gibt es schon eine umfangreiche Bibliothek an Signalverarbeitungsblöcken, die von grundlegenden Rechenoperationen bis hin zu komplexen Implementierungen wie ganzen Radarsystemen reichen. Die graphische Oberfläche GNU Radio Companion (GRC) ermöglicht es einen Flowgraph zu erstellen, ohne dabei Code schreiben zu müssen. Per Drag and Drop werden dabei die GNU Radio Blöcke angeordnet und mit Pfeilen verbunden. Der GRC ist ein nützliches Tool für schnelles Prototyping und für die Visualisierung eines Flowgraphs.
Die in diesem Kapitel vorgestellte Realisierung eines DAB/DAB+ Transceivers wurde mit GNU Radio implementiert. Das dabei enstandene Out-Of-Tree Modul enthält alle geschriebenen GNU Radio Blöcke und Flowgraphs und ist unter GNU General Public License (GPL3) lizenziert und frei zugänglich. Die dabei verwendete Hardware ist unabhängig von der Implentierung und kann von kostengünstigen RTL-SDRs bishin zu gut verarbeiteten Universal Software Radio Peripheral (USRP) reichen.
In der Evaluation des Systems in Abschn. 5 wurden GNU Radio Flowgraphs als Basis für Simulationen genutzt.

3.2. Implementierung des Senders

Abb. 3.1 zeigt einen funktionsfähigen DAB Sender im GRC. Der Sender transportiert zwei Audio Kanäle im MSC, wobei der obere Kanal DAB+ (MPEG4 und Reed Solomon Encoder) und der untere Kanal DAB (MPEG 2 Encoder) transportiert. Jeder Block in der grafischen Ansicht entspricht einer eigenständigen Klasse, wobei die Kanalcodierer des FIC und MSC sowie der OFDM Modulator hierarchische Blöcke sind, also mehrere Blöcke in sich zusammenfassen. Die Struktur der Blöcke als einzelne Funktionseinheiten ist von der Aufteilung dem DAB Standard nachempfunden. Bei der Implementierung wurde darauf geachtet, möglichst viel Funktionalität in eine Klasse zu integrieren ohne dabei an Flexibilität der Blöcke einzusparen.

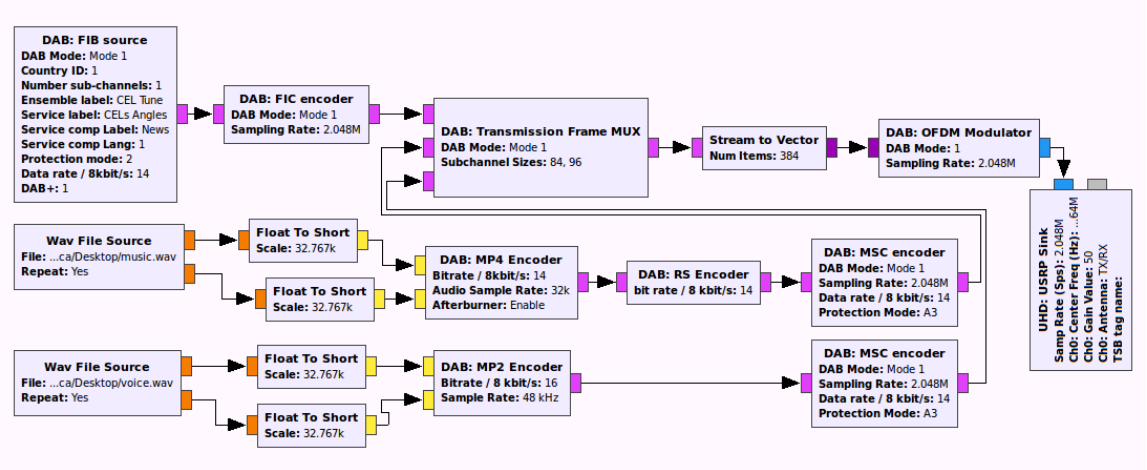


Abbildung 3.1.: DAB/DAB+ Sender im GRC

3.2.1. FIB Quelle

Die FIB Quelle produziert standardkonforme FIBs für den FIC. Die MCI und SI werden der Klasse dabei als Argumente übergeben. Die folgenden Informationen werden gesendet:

- Ensemble Infos: Country ID, Ensemble ID, CIF Zähler
- Service Organisation: Country ID, Anzahl der Services, Servicebeschreibung
- Service Component Beschreibung: sub-channel ID, Audio Typ, primary/secondary
- Sub-channel Organistaion: sub-channel ID, Start und Länge (in CUs), Protection Mode
- Labels für das Ensemble und jeden Service und deren Sprache (SI)

Diese Informationen umfassen bei weitem nicht das ganze Spektrum der möglichen Informationen. Sie sind aber ausreichend um dem Empfänger die nötigen MCI für die Decodierung und dem Nutzer die Informationen zur Identifikation des Programmes zur Verfügung zu stellen. Weil die MCI eine weit höhere Wichtigkeit als die SI einnimmt, wird sie mit jedem CIF geschickt. Die Größe der gesamten MCI hängt von der Anzahl der verwendeten Services

und Service Komponenten ab. Bei einer Begrenzung auf maximal 7 Services mit je einem sub-channel kann die Information in den ersten zwei FIBs gespeichert werden. Im dritten FIB wird dann die SI gesendet. Da ein Label mit 16 Buchstaben ein FIB schon nahezu füllt, werden die einzelnen Labels nacheinander jeweils im dritten FIB verschickt. Dies ist nicht problematisch, da die Labels im Regelfall über die Zeit konstant bleiben und durch eine CIF Rate von $\frac{1}{24ms} > 41\,\text{CIFs}/s$ trotzdem jedes Label mehrmals pro Sekunde gesendet wird.

evt grafik wie in blog-post week 02

3.2.2. FIC Encoder

Der FIC Encoder ist in einem hierarchischen Block (Abb. 3.2) implementiert. Er enthält die gesamte Kanalcodierungskette aus Kap. 2.2. Während für die CRC Berechnung und die Faltungscodierung eine jeweilige Klasse implementiert wurde, besteht die Energieverwischung, wie in Gl. 2.2 beschrieben, lediglich aus einer XOR Verknüpfung des Bitstreams mit der PRBS Folge.

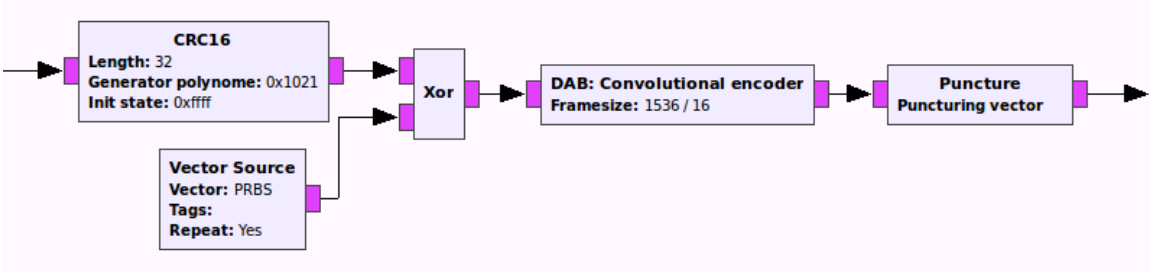


Abbildung 3.2.: Kanalcodierung im hierarchischen Block des FIC Encoders

3.2.3. Audio Quellen und Encoder

Die Audioeingangsdaten der Audioencoder sind 16 bit PCM Samples. Um diese für den Stream zur Verfügung zu stellen gibt es verschiedene Möglichkeiten. Entweder werden die Audiosamples über eine Binärdatei als Rohdatenquelle zur Verfügung gestellt. Eine andere Möglichkeit bietet der "WAV File Source"Block von GNU Radio, der aus dem populären "WAVEÄudioformat die Rohdaten extrahiert und somit auch eine PCM Quelle darstellt. Eine letzte Möglichkeit ist die direkte Generation von PCM Samples über die lokale Soundkarte. Alle drei Möglichkeiten resultieren letztendlich in einem Stream mit 16 bit Gleitkommawerten, dessen Samplingrate der Audiorate entspricht. Weil die implementierten Audiocodecs auf Integerebene arbeiten, wird der Stream vor der Encodierung auf den neuen Wertebereich abgebildet.

MPEG 1/2 Audio Layer II Encoder (DAB)

Der MPEG 2 Encoder Block verwendet einen Patch der frei verfügbare mp2 Encoder Bibliothek TooLAME. Der Patch stammt vom ODR mmbTools Projekt und ist speziell auf DAB angepasst. Der Code wurde in einen GNU Radio Block gepackt um ihn in einen GNU Radio Flowgraph einzubauen. Die Funktionalität des Encoder Blocks kann mit einem Loopback-Test verifiziert werden, indem der encodierte Stream mit einem Audio-Player erfolgreich decodiert und abgespielt wird.

MPEG 4 HE-AACv2 (DAB+)

Der MPEG 4 Encoder basiert auf einem Patch der FDK-AAC Codec Bibliothek für Android der Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.. Die Bibliothek untersützt eine Reihe von AOTs (Audio Optimization Tools) welche die Audiokompression nach unterschiedlichen Paramtern optimieren. Es stehen die folgenden AOTs zur Verfügung:

- AAC-LC: Geringe Komplexität (1,5 bits/sample) für minimales Codierungsdelay
- HE-AAC Dualband SBR: Hohe Effizienz (0,625 bits/sample) durch Spektralbandreplikation.
- HE-AAC v2 PS: Zusätzlich mit paramterisiertem Stereo (0,5 bits/sample)

Ein Patch des ÖDR-AudioEnc"Moduls wurde für die Implementierung des GNU Radio Blocks genutzt.
DAB+ enthält neben der neuen Audiokompression eine zusätzliche Fehlerkorrektur durch einen Reed-Solomon Code. Um den MPEG 4 Encoder Block nicht zu sehr auf DAB+ zu spezialisieren, wurde der Reed-Solomon Encoder in einem separaten Block Implementiert.

3.2.4. MSC Encoder

Der MSC Encoder entpricht von Aufbau und Struktur dem FIC Encoder Block aus 3.2.2. Der Block enthält eine zusätzliche Klasse für das Zeitinterleaving und besitzt keinen CRC Generator.

3.2.5. Multiplexer

Der Multiplexer Block hat prinzipiell die Aufgabe eines Parallel-Seriell Wandlers. Er besitzt 1+ # sub-channels Eingänge und setzt diese zu einem Sendesignale nach der Struktur aus 2.4.1 zusammen. Die Position der jeweiligen Audiostreams im CIF werden dabei über die selbe MCI bestimmt, die in den FIBs des FIC desselben Sendeframes stehen. Die Synchronisationssymbole $z_{0,k}$und$z_{1,k}$ sind an dieser Stelle noch nicht vorhanden und werden dem Sendeframe im OFDM Modulator hinzugefügt.

3.2.6. OFDM Modulator

Der OFDM Modulator Block ist ein hierarchischer Block mit vielen Unterklassen. Jede Klasse führt dabei einen der in 2.4 beschriebenen Schritte zur Modulation durch.

QPSK Mapping

Der QPSK Mapper bildet 1 Byte auf 4 komplexe QPSK Symbole ab. Da wie in 2.4.2 beschrieben zuerst die Realteile und anschließend die Imaginärteile eines kompletten OFDM Symbols übertragen werden, kann der QPSK Mapper nicht einfach im Streammodus arbeiten, also jedes Byte direkt auf 4 QPSK Symbole abbilden. Ein Gedächtnis der Klasse kostet Rechenzeit, weil die Eingangsdaten zusätzlich einmal ins Gedächtnis kopiert werden müssen. Um dies zu vermeiden arbeitet der Mapper auf Vektorbasis der Länge 1536/4 Bits am Eingang, bzw. 1536 Bits am Ausgang. Dies umfasst genau der Länge eines OFDM Symbols.

Einfügen des Phasenreferenzsymbols

Das Phasenreferenzsymbol wird schon als QPSK Symbol (um $\pi/4$ gedreht) generiert. Daher wird es erst nach der QPSK Modulation der Datensymbole an das Sendeframe angehängt. Das Phasenreferenzsymbol ist bei jedem Sendeframe gleich und kann deshalb im Konstruktur einmalig generiert werden.

Differentielle Modulation

Die Differentielle Modulation stellt eine einfache komplexe Multiplikation eines jeden Symbols mit dessen Vorgänger dar (siehe 2.4.4. Auch hier wird auf Vektorbasis gearbeitet.

Frequenzinterleaving

Inverse Schnelle Fourier-Transformation (IFFT)

Die OFDM Operation kann nach 2.4.5 als IDFT durchgeführt werden. Ein Teil der $N = 2048$ Unterträger sind nicht belegt. Diese müssen vor der Transformation als komplexe Nullsymbole eingefügt werden. Wegen $2048 = 2^{11}$ kann die IDFT als eine deutlich schnellere IFFT realisert werden. Ein IFFT Block ist im GNU Radio Modul gr-fftvorhanden.

cite properly

Einfügen des Cyclic Prefixes

Nach der IFFT liegt das Sendesignal nun im Zeitbereich vor. Das Einfügen des Cyclic Prefixes entspricht dem Kopieren vom Ende jedes Symbols an den Anfang. Das Einfügen eines Cyclic Prefixes ist eine weitverbreitete Technik bei OFDM. Daher existiert in GNU Radio schon ein Block mit dieser Funktionalität, der nur noch mit der passenden Symbollänge $T_S = 2048$ Samples und der Länge des Cyclic Prefixes $T_G = 504$ Samples initialisiert werden muss.

Einfügen des Nullsymbols

Als letzter Schritt in der Sendekette erfolgt das Einfügen des Nullsymbols. Dabei werden $T_{NULL} = 2656$ Samples mit dem Wert $0 + 0j$ zwischen zwei Sendeframes geschrieben.

Das Basisbandsignal ist nun bereit um hochgemischt und anschließend gesendet zu werden. Beide Schritte sind in dem GNU Radio Block USRP Sink vereint. Alternativ lässt sich das Basisbandsignal natürlich auch als I/Q Daten in einer Binärdatei speichern.

3.3. Implementierung des Empfängers

Bei der Übertragung des Sendesignals vom Sender zum Empfänger treten Störungen und Effekte auf, die das Empfangssignal vom Sendesignal abweichen lassen. Vor der Demodulation und Auswertung der Daten wird ist die erste Stufe des Empfängers eine Synchronisation. Die Synchronisation hat zum Ziel, das Empfangssignal so zu korrigieren sodass es dem Sendesignal wieder möglichst nahe kommt. Die anschließende Kanaldecodierung und Auswertung der Daten im FIC und MSC findet in den entsprechenden hierarchischen GNU Radio Blöcken statt. Abb. 3.3 zeigt den Aufbau eines DAB+ Empfängers im GRC.

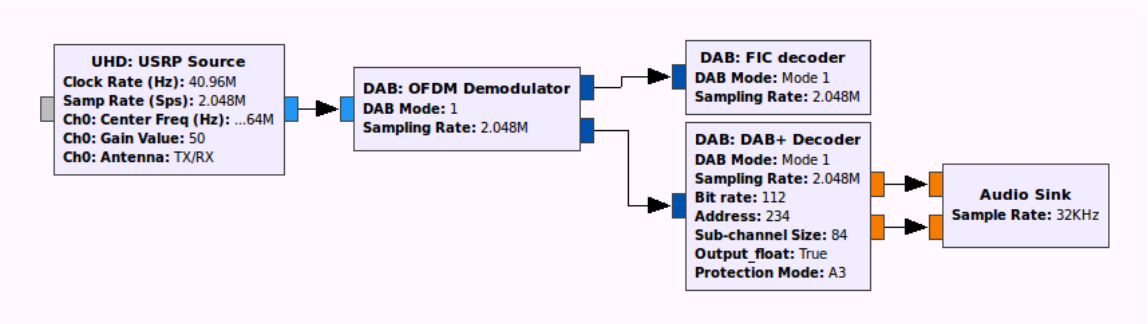


Abbildung 3.3.: DAB+ Empfänger im GRC

Das Empfangssignal $r(t)$ wird in folgender Beziehung zum Sendesignal $s(t)$ angenommen:

$$r(t) = a(t,f) \cdot s(t - \tau_{off}) \cdot e^{j(2\pi f_{off} t + \varphi_{off})} + n(t) \tag{3.1}$$

Das Signal wird um einen Zeitoffset τ_{off} verzögert und hat wegen Doppler-Verschiebungen einen Frequenzoffset f_{off}. Ein konstanter Phasenfoffset φ_{off} der durch Reflektionen der Welle entsteht, wirkt sich in diesem Fall nicht auf das demodulierte QPSK Signal aus, da sich wegen der differentiellen Modulation eine konstante Phase bei der Differenzbildung heraushebt. Die Amplitude des Signals wird durch frequenzselektives Fading über der Zeit und der Frequenz unterschiedlich stark gedämpft. Wegen der Schmalbandigkeit der Unterträger bei OFDM kann der Kanal trotzdem in jedem Unterträger als flach angenommen werden. Die Dämpfung des Signals ist für die Entscheidungsgrenzen bei PSK zudem nicht ausschlaggebend. Letztendlich wird das Signal noch durch Additives Weißes Gaußsches Rauschen (AWGN) überlagert. In der folgenden Signalverarbeitungskette werden die genannten, relevanten Effekte schrittweise gemessen und korrigiert um eine möglichst optimale Synchronisations zu erreichen.

Es ist also keine Kanalschätzung nötig

3.3.1. Zeit Synchronisation

Eine geeignete Zeitsynchonisation muss sowohl eine grobe Synchronisation des OFDM Frames als auch eine feine Synchonisation der einzelnen OFDM Symbole sicherstellen.

Der Beginn eines jeden Frames k wird durch das Nullsymbol $z_{0,k}$ markiert. Wegen $S(t) = 0$ für $t \in [0,T_{NULL}]$ stellt es eine zuverlässige Möglichkeit dar den Anfang eines Frames über eine Energiemessung zu detektieren.

Formel Energie aber Intervall kürzer als T NULL

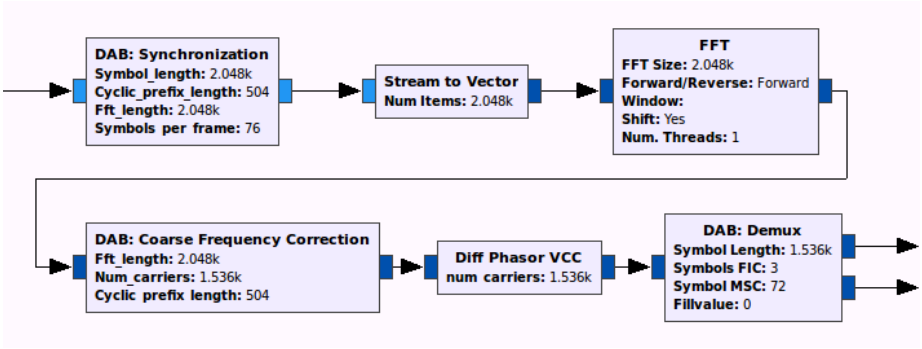


Abbildung 3.4.: Aufbau der kompletten Synchronisationskette im GRC

$$E[i] = \sum_{j=i}^{T_G} |x[j]|^2 \tag{3.2}$$

Nachdem der Anfang eines Frames detektiert wurde, muss im Folgenden der Beginn jedes OFDM Symbols $z_{l,k}$ ($l \in [1,76]$) festgelegt werden. Diese feine Zeitsynchronisation erfordert keine sehr hohe Genauigkeit, da jedem Symbol ein Cyclic Prefix der Dauer $T_G = 246\mu s$ vorgeschoben ist, dessen Inhalt dem Ende des eigentlichen Symbols entspricht. Dadurch ergibt sich ein Zeitbereich von $T_D \in [0,T_G]$ in dem der Symbolanfang fehlerfrei gesetzt werden kann, also das gesetzte Symbolfenster nicht in das vorhergehende oder nachfolgende Symbol hereinragt.

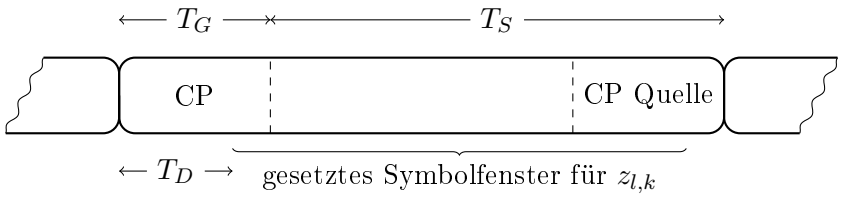


Abbildung 3.5.: OFDM Symbol und Cyclic Prefix

Um ISI zu minimieren ist der theoretisch optimale Abtastzeitpunkt genau am Ende des Cyclic Prefix zu wählen. Eine komplette ISI Unterdrückung ist wegen $T_G < \tau_{max}$ mit einer maximalen Echo Verzögerung von $\tau_{max} = 300\mu s$ nicht möglich. Jedoch wird die ISI zu einem Minimum reduziert, wenn ein maximal später Startpunkt gewählt wird, da dadurch die Interferenzzeit $\tau_{max} - T_D$ (bei $\tau_{max} < T_G + T_S$) minimiert wird.

cite

Um den exakten Anfang eines Symbols zu bestimmen, wird die zyklische Wiederholung des Symbolanteils im Cyclic Prefix genutzt. Durch eine Korrelation des abgetasteten Empfangssignal $r[i]$ mit einer um T_S verzögerten Version desselben Signals $r[i+T_S]$ über das Intervall $[i,i+T_G]$ kann der Anfang des Cyclic Prefix über einen Peak der Korrelation identifiziert werden.

Bild mit Symbol und evt Frame, wo Zeitreferenz markiert ist, auf dem dann die Formeln/Delays basieren können

klären, wann T_G Zeit und wann samples angibt, bzw ande-

$$y[i] = \sum_{j=i}^{T_G} r[j]\overline{r[j+T_S]}, \;\; y[i] \in \mathbb{C} \tag{3.3}$$

Unter Berücksichtigung von AWGN ist r[i]=x[i]+n[i] mit $N[i] \sim \mathcal{N}(\mu, \sigma^2)$ und es resultiert für die Korrelation

$$\begin{aligned} y[i] &= \sum_{j=i}^{T_G} (r[j]+n[j])\overline{(r[j+T_S]+n[j])} \\ &= \sum_{j=i}^{T_G} x[j]\overline{r[j+T_S]} + \sum_{j=i}^{T_G} r[j]\overline{n[j+T_S]} + \sum_{j=i}^{T_G} n[j]\overline{x[j+T_S]} + \sum_{j=i}^{T_G} n[j]\overline{n[j+T_S]} \\ &= PT_G + 2(P\sigma^2) + \sigma^4 \approx PT_G + 2(P\sigma^2) \end{aligned} \tag{3.4}$$

und damit ein SNR des Korrelationssignals von

$$SNR = \frac{P_s}{P_n} = \frac{PT_G}{2P\sigma^2} = \frac{T_G}{2\sigma^2} \tag{3.5}$$

Durch eine Mittelung über $T_G \cdot Samplerate = 504$ Samples ist das SNR somit ausreichend groß um eine entprechend genaue Peakdetektion durchführen zu können.
Die Korrelation ist dabei auf die Energieanteile des Cyclic Prefixes und dessen Quelle nach T_S normiert, sodass das Ergebnis unabhängig von der Empfangsleisung bleibt, die durch Empfangsqualität und verwendeter Hardware stark variieren kann.

$$y_{norm}[i] = \frac{\sum_{j=i}^{T_G} r[j]\overline{r[j+T_S]}}{\sqrt{\sum_{j=i}^{T_G} |r[j]|^2 \sum_{j=i}^{T_G} |r[j+T_S]|^2}} = \frac{\sum_{j=i}^{T_G} r[j]\overline{r[j+T_S]}}{\sqrt{E[i]E[i+T_S]}} \tag{3.6}$$

In Abbildung 3.6 ist zu erkennen, dass die Flanken der Korrelation linear ansteigen. Die Breite einer Flanke entspricht T_G, also gerade dem Entscheidungsbereich für T_D. Durch die Linearität der Flanke und der Normierung, kann der relative Abtastzeitpunkt innerhalb des Cyclic Prefix über einen Schwellwert eingestellt werden. Der tatsächliche Abtastzeitpunkt wird anschließend mit einer Verzögerung von T_G gestetzt. In der Implementierung von Abbildung 3.6 wurde ein Schwellwert von 0,85 eingestellt, der sich als guter Kompromiss zwischen ISI Unterdrückung und einem Sicherheitsabstand zu $T_D > T_G$ herausgestellt hat. Man beachte, dass $T_D > T_G$ dazu führt, dass Samples vom nachfolgenden Symbol im gesetzen Symbolfenster liegen was zum Empfang von falscher Information führt. Dieser Fall entspricht einer fehlerhaften Zeitsynchronisation.

Implementierung

Um eine mehrfache Berechnung von Gleichung 3.2 für die Energiemessung (Gl. 3.2) sowie die normierte Korrelation (Gl. 3.6) zu vermeiden, wurde die feine und grobe Zeitsynchronisation in einer gemeinsamen Klasse implementiert.
Abbildung 3.7 zeigt den Programmablaufplan der Zeitsynchronisation. Er ist aufgeteilt in die beiden Zweige SSuche Frame Startünd "Kontrolle". Der Zweig SSuche Frame Start"kommt bei einem der folgenden Fälle zum Einsatz:

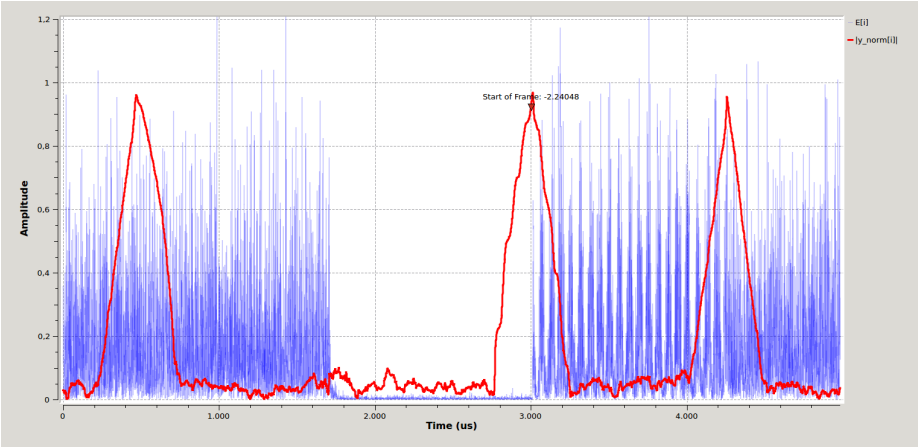


Abbildung 3.6.: Korrelation des Cyclic Prefix

- Die Synchronisation wird initial gestartet.
- Das Programm ist nicht mehr in Synchronisation.
- Das Ende eines Frames wurde erreicht.

Alle diese Situationen haben gemeinsam, dass im Folgenden nach dem Peak des Symbols z_1,k (erstes Symobl nach dem Nullsymbol) gesucht wird. Es wird iterativ für jedes Sample des Zeitsignals $x[i]$ eine Korrelation $y[i]$ nach Gl. 3.6 durchgeführt. Dadurch können (analog zu Abb. 3.6) Korrelationspeaks detektiert werden die jeweils dem Start eines Symbols entsprechen. Durch die Energiemessung das Symbols $z_{1,k}$ von den restlichen Symbolen $z_{l,k}$, mit $l \in [2{,}76]$ unterschieden werden.
Die Komplexität der Korrelationsoperation kann drastisch reduziert werden, indem die Summe, die laut Gl. 3.3 in jedem Schritt $T_G = 504$ Additionen durchführt, durch eine gleitende Summe ersetzt wird.

$$y[i+1] = y[i] - r[i]\overline{r[i+T_S]} + r[i+T_G]\overline{r[i+T_G+T_S]} \tag{3.7}$$

Dabei ist zu beachten, dass natürlich $y[i=0]$ komplett berechnet werden muss. Um einen das Ergebnis verfälschenden Drift von $y[i]$ zu vermeiden, wird die Summe außerdem alle $i = 100000$ Samples neu berechnet.
Wenn der Anfang von $z_{1,k}$ gefunden und festgelegt wurde, stehen auch die Anfänge aller anderen Symbole des Frames fest, da sie mit dem festen und bekannten Abstand von $T_G + T_S$ direkt aufeinanderfolgen. Der Zweig "Kontrolleßpringt daher nur noch vom Start eines Symbols $x[i]$ zum Nächsten $x[i+T_G+T_S]$ und berechnet an dieser Stelle einmalig die Korrelation $y[i+T_G+T_S]$. Liegt $y[i+T_G+T_S]$ über einem Schwellwert, wurde das Sample als Anfang des nächsten Symbols bestätigt und die Kontrollschleife iteriert zum nächsten Symbol. Falls an einem erwarteten Symbolanfang der Schwellwert der Korrelation unterschritten wird, wechselt das Programm in den Zweig SSuche Frame Start".
Da die Symboldauer genau bekannt ist, mag eine Kontrolle von jedem einzelnen Symbol zunächst überflüssig erscheinen. Der Aufwand wird jedoch gerechtfertigt um Störeffekte wie einen Clockdrift rechtzeitig erkennen und korrigieren zu können bevor dieser zu einem

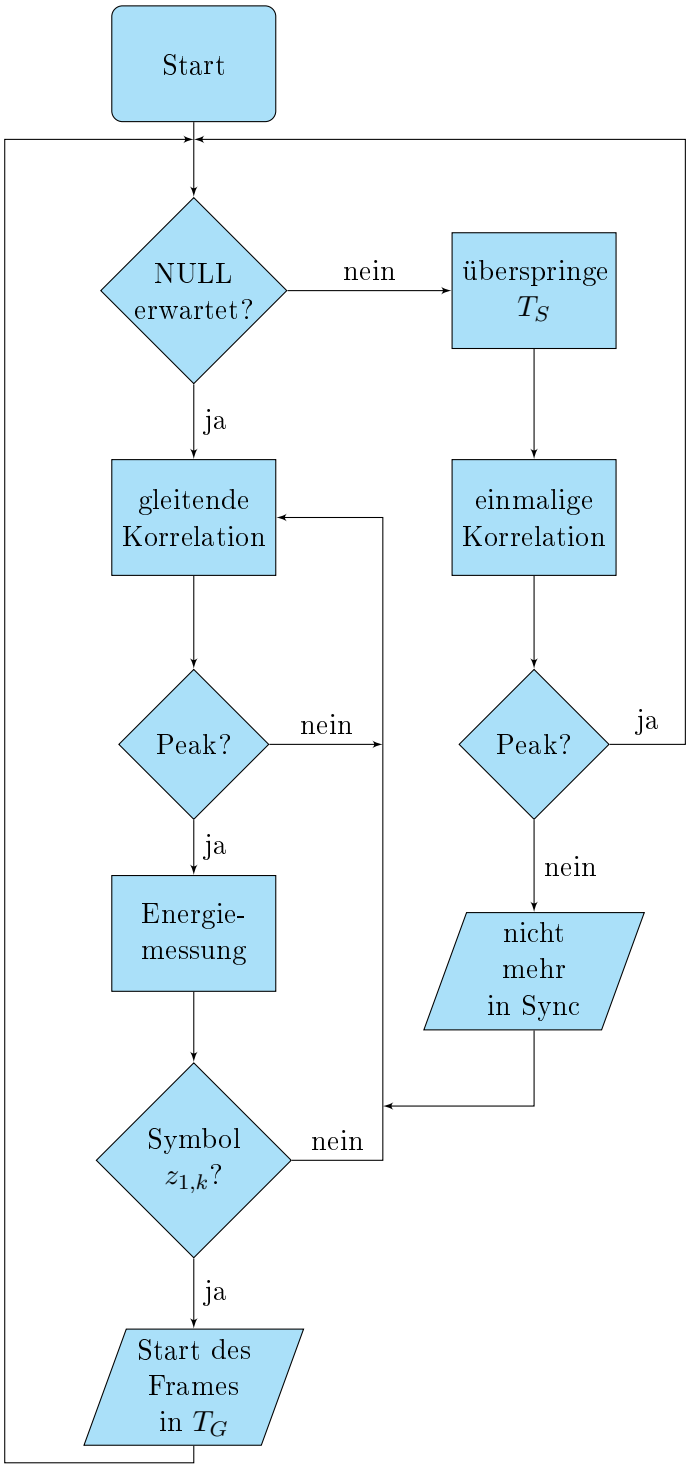


Abbildung 3.7.: Programmablaufplan der Zeitsynchronisation

Verlust der Synchronisation und damit zu Datenverlusten führen kann. Ein beispielhafter Clockdrift von 50 ppm kann im rauschfreien Fall zu einem maximalen Phasenfehler von

$$\begin{aligned}\Delta\varphi_{max} = 2\pi f_{max}\Delta t &= 2\pi\frac{B}{2}Clockdrift(T_G + T_S) \\ &= 2\pi\frac{1536Hz}{2}50ppm1246\mu s \\ &= 0{,}3rad = 17{,}2^\circ\end{aligned} \tag{3.8}$$

führen. Rechnung 3.8 zeigt, dass eine feine Zeitsynchronisation zu jedem Framebeginn zu wenig ist, da sich der Phasenfehler bei 76 Symbolen pro Frame aufsummiert und damit die Entscheidungsgrenzen einer QPSK Demodulation überschreitet. Damit ist der Zweig "Kontrolle"gerechtfertigt.

3.3.2. Frequenz Synchronisation

Die Messung und Korrektur eines Frequenzoffsets f_{off} ist der nächste Schritt der Synchronisationskette. Sie spielt in OFDM Systemen eine besonders wichtige Rolle, da ein Frequenzoffset die Orthogonalität zwischen den Unterträgern zerstört und somit zu ICI führt.

Feine Frequenz-Schätzung

Für die Messung des Frequenzoffsets kann wieder auf die Korrelaton aus Gl. 3.3 zurückgegriffen werden. Ein Frequenzoffset lässt sich hier als konstante Änderung der Phase über der Zeit T_S messen, da die Phase von Cyclic Prefix und dessen Wiederholung ohne Frequenzoffset gleich sind.

$$f_{off}[i] = \frac{arg(y[i])}{T_S} \tag{3.9}$$

Ein zusätzliches AWGN ändert die Phase jedes einzelnen Samples. Wegen der Mittelwertfreiheit von AWGN kann die Varianz des gemessenen Frequenzoffsets durch eine Mittelung reduziert werden. Durch das Korrelationsintervall von $T_G = 504$ Samples wird solche eine Mittelung durchgeführt was die relativ kleine Varianz der Frequenzoffsetmessung in Abb. 5 bestätigt.

Rechnung zu Varianz bei Mittelung (Gleichverteilung mit unabh. ZVs als Ausgangspunk

Durch die aktive Mittelung über N Symbole wird eine weitere Senkung der Varianz um Faktor N erreicht. Pro Symbol wird genau eine Korrelation berechnet, also ergibt sich ein Mittelungsfenster der Dauer $N(T_G + T_S)$. Ein zu langes Mittelungsfenster führt zu einer Trägheit der Frequenzmessung und damit auch zu einer Zeitverzögerung in der Frequenzkorrektur, was bei schnellen Frequenzänderungen zu Synchronistaionsverlusten führen kann. Vor allem bei mobilen DAB Empfängern wird dieser Fall aufgrund der Dopplerverschiebung relevant. Deshalb wird für die obere Grenze der Mittelung die Bedingung gestellt, dass bei einer maximalen, konstanten Beschleunigung a der durch die Mittelung verursachte Messfehler f_M unter der 3σ Grenze der Messabweichung liegt.

Wie korrigieren reinbringen

$$f_M \overset{!}{<} \frac{3\sigma(SNR)}{\sqrt{N}} \tag{3.10}$$

Mit $a = 50m/s^2$ und einer Trägerfrequenz von $f_T = 200MHz$ ist

$$\frac{df}{dt} = f_T\frac{a}{c} = 33{,}3Hz/s \tag{3.11}$$

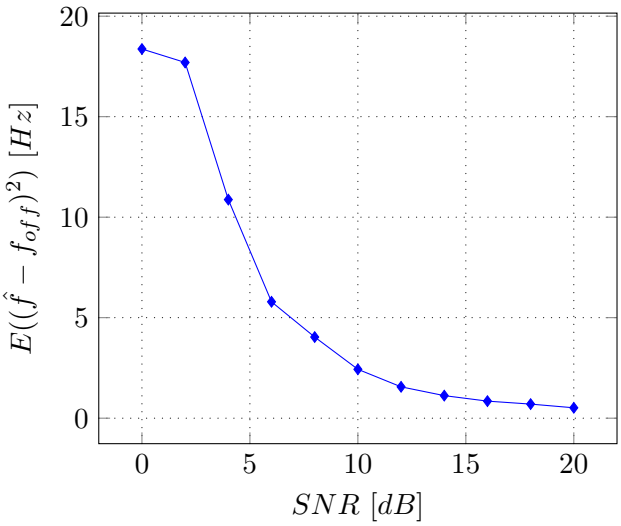


Abbildung 3.8.: Varianz der feinen Frequenzoffsetmessung

und damit

$$\begin{aligned} f_M &= \frac{1}{N}\sum_{i=0}^{N-1} i\,\frac{df}{dt}(T_G+T_S) \\ &= \frac{1}{N}\frac{df}{dt}(T_G+T_S)\left(\frac{N^2+N}{2}-N\right) \overset{N\text{groß}}{\approx} \frac{df}{dt}(T_G+T_S)N \end{aligned} \tag{3.12}$$

Aus 3.10 und 3.12 ergibt sich eine rauschabhängige Obergrenze für das Mittelungsintervall.

$$N \overset{!}{<} \left(\frac{df}{dt}\frac{T_G+T_S}{3\sigma(SNR)}\right)^{-2/3} \overset{SNR=10dB}{\approx} 116 \tag{3.13}$$

Grobe Frequenz-Schätzung

Wegen $arg(y[i]) \in (-\pi,\pi]rad$ ist der gemessene Frequenzoffset aus Gl. 3.9 nur in $f_{off} \in (-500,500]Hz$ eindeutig. Dieser Eindeutigkeitsbereich entpricht genau der Breite eines OFDM Unterträgers von $1kHz$. Nach der feinen Frequenzkorrektur liegen also die Unterträger wieder auf dem Frequenzraster und es tritt durch die Orthogonalität keine ICI auf. Jedoch kann das Signal um das ganze Vielfache des Unterträgerabstandes verschoben sein, wodurch Symbole im Empfänger durch eine falsche Zuweisung der FFT Bins fehlinterpretiert würden und die komplette Information des Frames verloren ginge. Da sowohl die Erkenung als auch die Korrektur einer Unterträgerverschiebung im Frequenzbereich wesentlich einfacher und anschaulicher ist, wird die grobe Frequenz-Korrektur nach der FFT Operation (Abschn. 3.3.3) durchgeführt. Von den 2048 Unterträgern die aus der FFT resultieren, sind jeweils die ersten $K = \frac{1536}{2}$ rechts und links des zentralen DC-Trägers belegt. Die restlichen, unbelegten Träger enthalten am Empfänger neben möglichst geringer ICI lediglich AWGN. Durch eine Korrelation des FFT Vektors mit dem Vektor der bekannten Trägerbelegung [1,1,...,1,0,1,...,1,1] der Länge $1536+1$ kann der Trägeroffset n ermittelt werden.

arctan bei Eindeutigkeit reinbringen?

grobe fe nach FFT reinbringen

FFT Kapitel vorziehen?

evt Grafik mit beschriftung auf der die Beschreibung aufbauen kann, T_u als FFT

$$X[n] = \underset{i}{\operatorname{argmax}} \sum_{n=i}^{i+K} |X[n]|^2 \quad \text{mit} \quad i \in [0, L_{FFT} - K) \tag{3.14}$$

3.3.3. FFT

Die modulierten Symbole werden durch das OFDM System parallel auf den $K = 1536$ Unterträgern moduliert und sind innerhalb eines Sendesymbols T_S durch ihre Phase und Frequenz vollständig beschrieben [?]. Um aus dem empfangenen Zeitsignal nach der Zeit- und Frequenzsynchronisation nun wieder die D-QPSK Symbole zu erhalten, wechselt man mittels einer Diskrete Fourier-Transformation (DFT) wieder in den Frequenzbereich, was in Gl. 2.8 gezeigt wurde. Wegen $L_{DFT} = \frac{2{,}048 Msamples/s}{T_S} = 2048 = 2^{11}$ kann die DFT durch eine Schnelle Fourier-Transformation (FFT) der Länge 2048 effizient implementiert werden. Der FFT Algorithmus ist ist in GNU Radio [?] bereits implementiert und kann verwendet werden.

zitiere FFT aus SUS Buch

3.3.4. Frequenzspreizung

Die im Sender vorgenommene Frequenzspreizung wird rückgängig gemacht, indem die Unterträger wieder in ihre ursprüngliche Reihenfolge gebracht werden. Diese Operation entpricht, genau wie im Sender, einem Vertauschen der Unterträger nach einer festgelegten Vertauschungsregel. Deshalb kann hier die gleiche Klasse wie im Sender genutzt werden, die mit dem invertierten Vertauschungsvektor initialisiert wird.

Unbedingt Unterschiedliche Symbole für Zeit und Samples einführen!!!

3.3.5. Demodulation

Die differentielle Demodulation erfolgt durch eine komplexe Multiplikation des D-QPSK Symbols mit dessen komplex konjugierten Vorgängersymbol. An dieser Stelle würde entsprechend der Umkehrung des Senders nun die QPSK Demodulation erfolgen, indem eine Bitentscheidung der komplexen Eingangswerte zum Beispiel nach dem Maximum Likelihood (ML)-Kriterium erfolgt. In dieser Implementierung wird aber hier noch keine sog. Harddecision durchgeführt, sondern die verrauschten QPSK Symbole werden in ihren Real- und Imaginärteil zerlegt und als Gleitkommawerte, sog. Softbits, weitertransportiert. Eine Bitentscheidung wird in der Faltungsdecodierung über den Viterbialgorithmus getroffen.

3.3.6. Kanaldecodierung

Die Kanaldecodierung durchläft die Encodierungskette in umgekehrter Reihenfolge und stellt so schrittweise den gesendeten Bitstream wieder her.

Zeitinterleaving

Das Zeitinterleaving hat im Sender die Bits nach festen Regeln mit einer Verzögerung versehen. Der Bitstream kann also wiederhergestellt werden, indem dieselbe Verzögerung angewendet wird um mit dem Schreiben auf das jeweilige Bit zu warten. Um ein komplettes Symbol schreiben zu können benötigt der Interleaverblock also neben dem aktuellen Symbol, auch alle 15 nachfolgenden Symbole, oder anderst gesagt benötigt der Block ein Gedächtnis von 15 Symbolen. Die in 2.3.4 angesprochene Verzögerung äußert sich in diesem Fall darin,

dass die ersten 15 produzierten Symbole des Blocks kein ausreichend großes Gedächtnis haben und die entstandenen Symbole daher Fehlerbehaftet sind. Wegen der Fehlerkorrekturfähgkeit des im Folgenden beschriebenen Faltungsdecoders ist es aber praktisch sogar möglich, die tatsächliche Verzögerung um einige Symbole zu reduzieren, da die letzten verzögerten Bits vom Decoder schon a priori bestimmt werden können.

Faltungsdecodierung und Viterbi-Algorithmus

Der Faltungsdecoder nutzt die im Encoder eingefügte Redundanz um aus der empfangenen Empfangsbitfolge die wahrscheinlichste Sendefolge zu bestimmen. Das Maximum A-Posteriori (MAP) Kriterium ist wegen der Gleichverteilung der Sendesymbole gleich dem ML Kriterium. Um die Komplexität des Entscheiders von exponentieller Ordnung auf lineare Ordnung zu reduzieren wird der Viterbi-Algorithmus zur Entscheidungsfindung herangezogen [**?**]. Eine effiziente Implementierung eines Softbit QPSK Viterbi Decodierers ist im GNU Radio Modul gr-trellis vorhanden.

Energieverwischung

Die angewendete Energieverwischung kann rückgängig gemacht werden, indem die selbe Operation wie im Empfänger angewendet wird. Die identischen Bits $y[i]$ der PRBS löschen sich zu jedem Sample $i \in \mathbb{N}$ durch die Exklusiv-Oder Verküpfung gegenseitg aus, sodass wieder das Sendebit $x[i]$ entsteht.

$$x[i] \oplus y[i] \oplus y[i] = x[i], \quad x[i],y[i] \in \{0,1\} \tag{3.15}$$

Die Bits haben nun die Kanaldecodierungskette durchlaufen und entsprechen im besten Fall den FIBs im FIC bzw. den komprimierten Audiostreams im MSC.

3.3.7. FIC Senke

Bevor die Informationen aus den FIBs verarbeitet werden, wird das CRC Wort geprüft. Dazu wird aus den Datenbits wie in 2.2.1 das CRC Wort berechnet und dieses mit dem gesendeten CRC Wort verglichen. Stimmen die beiden überein, enthält der FIB mit hoher Wahrscheinlichkeit keine Bitfehler. Korrekt übertragene FIBs können dann im Folgenden gelesen und ausgewertet werden.

3.3.8. Audio Decoder

Die Audiodecoder haben die Aufgabe aus dem Bitstream wieder ein PCM Stream zu generieren, der dem ursprünglichen PCM Stream vor der Kompression möglichst nahe[1] Für die Implementation des MPEG-1/2 Audio Layer II Decodes für DAB wurde die Bibliothek kjmp2 genutzt. Wie schon bei den Encodern wurden auch die Decoder jeweils in einen GNU Radio Block implementiert. Der Decoder für DAB+ Audio Frames wurde in 3 separaten Klassen implementiert. Der vorgeschaltete Reed-Solomon Decoder wurde auf Basis der Bibliothek für Vorwärtsfehlerkorrektur libfec implementiert. Für den eigentlichen MPEG 4 HE-AACv2 wurde eine Minimal-Implementation des Standards ETSI TS 102 563 von dem Open-Source Projekt qt-dab genutzt.

In Quellen tun

[1] Im Sinne einer subjektiven Wahrnehmung des menschlichen Gehörs.

4. Implementierung einer grafischen Transceiver Applikation

Ein in GNU Radio implementierte DAB/DAB+ Sender und Empfänger erfüllen funktionstechnisch alle geforderten Anforderungen. Jedoch ist die Benutzerfreundlichkeit sehr eingeschränkt. Beispielsweise muss die Speicheradresse eines sub-channels manuell aus dem Terminal abgelesen werden um sie im MSC Decoder anzugeben. Um die Benutzterfreundlichkeit zu erhöhen und den DAB/DAB+ Transceiver damit zu einer vollwertigen und ansprechenden Applikation werden zu lassen, wird eine Python Applikation mit grafischem Front-End implementiert.

4.0.1. Struktur

Abb. **??** zeigt das Klassendiagramm der Applikation.

Python main

Die Python Klasse "DABstepßtellt das Zentrum der Applikation aus Nutzersicht dar. Sie enthält alle Methoden die den Nutzer mit der App interagieren lassen. Dazu gehören zum einen Setter Methoden die beispielsweise den DAB Empfänger starten oder die Lautstärke regeln und zum anderen Getter Methoden die hauptsächlich für die angezeigten Informationen wie zum Beispiel eine Programmliste zuständig sind.

Graphisches Frontend

Die graphischen Informationen zum Design der Applikation liegen in einer separaten Klasse, dem sog. Frontend. Es beinhaltet alle Tabs, Widgets, Beschriftungen, usw. und deren Anordnung im Anzeigefenster der Applikation. Für die Implementierung der Graphische Benutzeroberfläche (GUI) wird die Python Bibliothek PyQt4 verwendet[1] Ein Großteil des Inhaltes dieser Klasse wurde mit dem Programm QtDesigner4 erzeugt.

4.0.2. GNU Radio flowgraphs

Den funktionellen Kern der Applikation bilden die beiden GNU Radio flowgraphs für den Transmitter und den Receiver. Sie enthalten die Definition, Instanzierung und Verbindung der einzelnen C++ Funktionalblöcke und bilden einen vollwertigen Transmitter bzw. Receiver. Im groben Aufbau entsprechen sie den in den vorherigne Kapiteln vorgestellten GNU Radio flographs aus Abb. 3.3 bzw. Abb. 3.1. Abhängig von den Argumenten der Klasse, werden die Flowgraphs aber individuell instanziert bzw. initialisiert. Zum Beispiel kann je nach Konfiguration des Empfängers eine Binärdatenquelle oder eine Hardwarequelle als

[1] Die aktuelle Nachfolgerversion PyQt5 verwendet Python3 was in GNU Radio noch nicht komplett unterstützt ist.

IQ Datenquelle dienen. Im Empfänger können ganze Zweige hinzukommen, wenn mehrere Audiostreams parallel gesendet werden. Die Flowgraphs laufen in einem separaten Thread. Dies verhindert, dass die komplette Applikation eingefroren ist, wenn der Flowgraph zum Beispiel überlastet ist.

GNU Radio Blocks

Die grundlegendste Klasse bilden die C++ Funktionalblöcke. Dazu gehören die in Abschn. 3.2 und 3.3 beschriebenen Blöcke des in diesem Projekt implementierten Moduls gr-dab, sowie einige grundlegenden Blöcke aus anderen GNU Radio Modulen. Manche Blöcke sind aus Gründen der Übersicht zu hierarchischen Python Blöcken zusammengefasst.

beschreibe schwerpunkte der blöcke und vor allem ,dass in DAB-Step möglichst wenig verarbeitung/logik mehr passieren soll

4.0.3. Datenfluss

Ein Großteil des Datenflusses geschieht in den C++ Blöcken, die große Mengen an komplexen IQ Samples oder Bits mit hohen Raten verarbeiten müssen und diese über Ringbuffer an den jeweils nächsten Block weitergeben. Zusätzlich müssen aber auch Daten, wie zum Beispiel die MCI und SI, von den Datenverarbeitungsblöcken zur Benutzeroberfläche transportierte werden um dort in einer strukturierten und für Menschen gut lesbaren Art dargestellt zu werden. Bei diesem Datenstrom fallen geringe Datenmengen an und der Fokus liegt auf einer guten Strukturierung und Charakterisierung der Daten. Eine geeignete Datenstruktur muss daher folgenden Anforderungen erfüllen:

- Unterstützung verschiedener Datentypen (string für Sendernamen, integer für Adressierungen, boolean für Bitschalter)
- Flexible Struktur in:
  - Anzahl der Attribute (das Kanalinfo FIB enthält beispielsweise 5 Attribute, während das Sendenamen FIB nur 2 Attribute beinhaltet)
  - Anzahl der Elemente (a priori ist nicht bekannt, wie viele Sender im empfangenen Ensemble enthalten sind, bzw. wie viel zusätzliche SI ausgestrahlt wird)
- Gute Lesbarkeit für Menschen

Das Format JavaScript Object Notation (JSON) erfüllt diese Eigenschaften hinreichend.

JSON

Die grundlegenden Datenstrukturen des JSON Formats sind:

$$\begin{aligned} object &= \Big\{ string : value, string : value, ... \Big\} \\ array &= \Big[value, value, ... \Big] \\ \text{mit } value &\in \{string, number, object, array, true, false, null\} \end{aligned} \tag{4.1}$$

Eine für die beschriebenen Anforderungen vorteilhafte Kombination von JSON Strukturen ist in Gl. 4.2 beispielhaft für ein FIB mit Kanalinformationen dargestellt.

$$FIB = \Bigg[\Big\{ "ID" : 4, "address" : 54, "size" : 84, "protection" : 2 \Big\} \Big\{ "ID" : 3, "address" : 234, "size" : 96, "protection" : 3 \Big\} \Big\{ "ID" : 2, "address" : 54, "size" : 54, "protection" : 1 \Big\} \Bigg] \tag{4.2}$$

Das Array liefert die flexible Anzahl an Elementen und das Objekt bietet eine variable Anzahl an Attributen. Die Daten sind gut lesbar und werden genau in dieser Form als String in den jeweiligen Blöcken (eine Vielzahl davon in der FIC Senke) generiert. Diese können dann von der Receiver Klasse über eine get-Funktion abgefrage werden und an die Klasse DABstep weitergeleitet werden, wo sie an entsprechender Stelle angezeigt werden. Das Lesen der Daten in DABstep gestaltet sich besonders intuitiv, da ein JSON Array als Python Array und ein JSON Objekt als Python Dictionary interpretiert werden kann.

4.0.4. Graphischer Aufbau

Neben der Funktionalität steht bei der graphischen Applikation DABstep auch die Benutzerfreundlichkeit im Mittelpunkt. Die Applikation soll auch von Laien intuitiv bedienbar sein und übersichtlich bleiben. Das Programm ist in Empfänger und Sender aufgeteilt die jeweils über zwei Tabs erreichbar sind. Abb. 4.2 zeigt die Empfangsseite. Die räumliche Aufteilung entspricht in etwa dem Ablauf einer Empfangssituation:

1. Initiale Einstellungen: Wahl der Quelle und des DAB Modus.
2. Senderliste: Anzeige und Auswahlfeld der DAB Sender.
3. Infos: Anzeige von zusätzlichen Informationen über das Empfangene DAB Ensemble.
4. Audio: Kontrolle über die abgespielte Musik.

Während die Empfangsseite hauptsächlich Informationen anzeigt, liegt der Schwerpunkt bei der Senderseite auf dem Schreiben von Informationen für das zu Sendende DAB Ensemble. Um die Bedienungsfreundlichkeit zu bewahren wurde die Aufteilung im Sender von der funktionalen Aufteilung dem Empfänger nachemfunden. Abb. 4.3 zeigt die GUI im Sendemodus.
Durch erhöhen der Anzahl der Kanäle wird die linke Spalte automatisch um eine Komponente ergänzt.

Entwicklermodus

Ein zusätzliches Feature stellt der Entwickler-Modus dar, der in der Empfängerseite über das "+ "Symbol in der oberen rechten Ecke erreichbar ist. Dieser erweitert die GUI um einige technische Anzeigen (siehe Abb. **??**.
Eine Anzeige für die Fehlerrate, getrennt für FIC und MSC, stellt das Ergebnis des CRC checks der FIBs bzw. des Firecode Checks für die Audio Frames dar. Die rechte Spalte

ergänzt die GUI um ein Konstellationsdiagramm, eine FFT Anzeige und einen Wasserfall Plot. Die drei Widgets wurden dabei aus dem GNU Radio Modul gr-qtgui übernommen, in den Flowgraph eingebaut und beim Öffnen des Entwicklermodus an die GUI als Objekte übergeben.

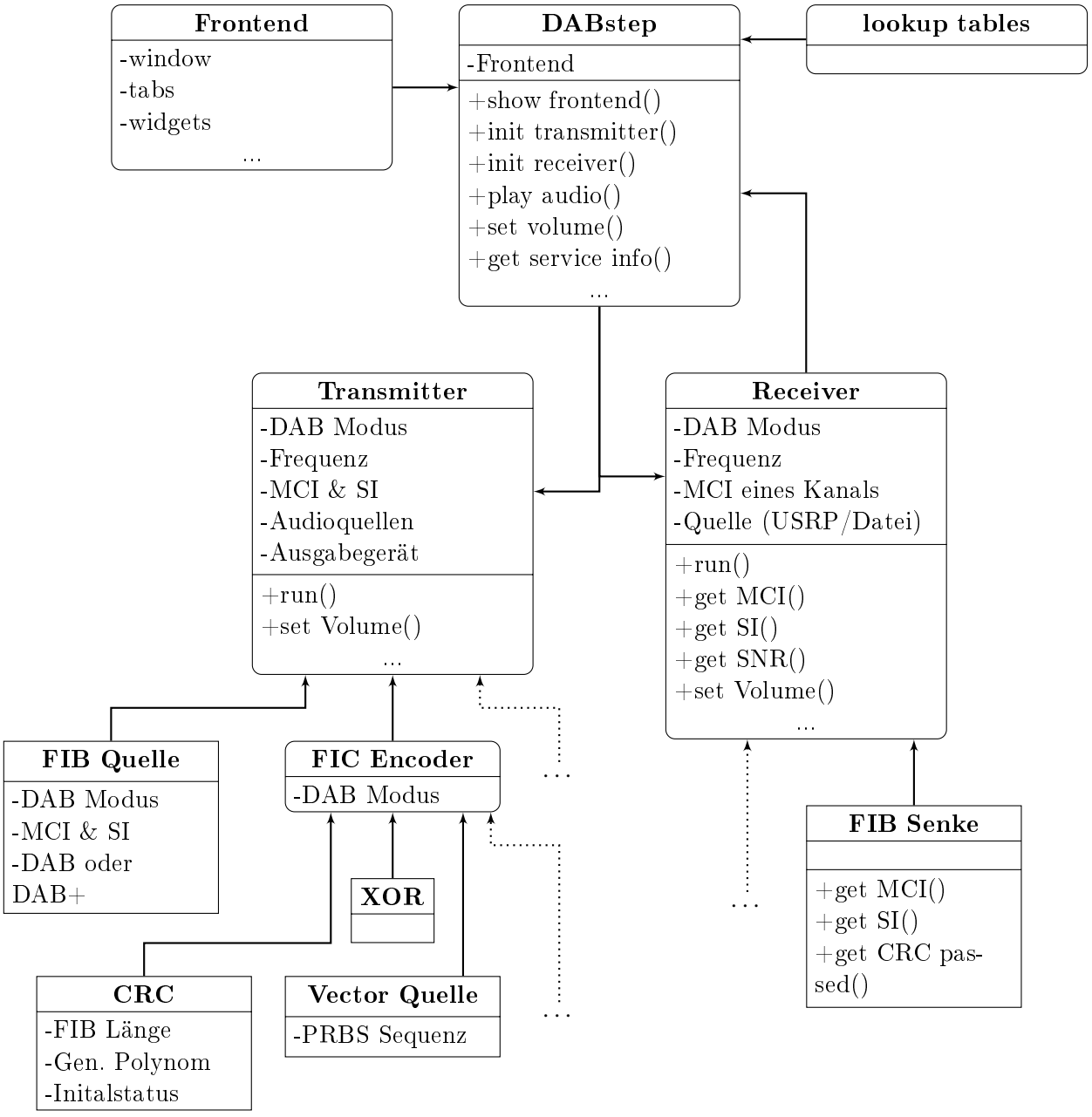


Abbildung 4.1.: Klassendiagramm der DAB/DAB+ Transceiver Applikation

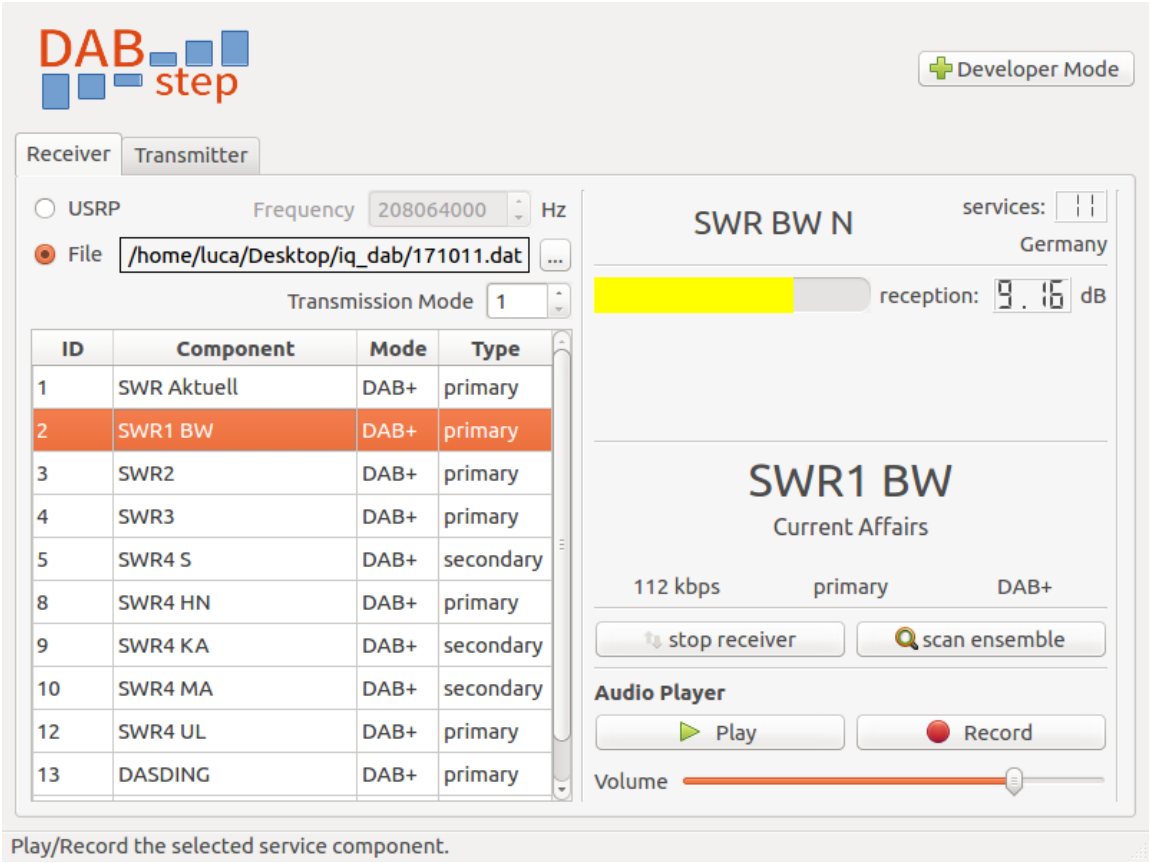


Abbildung 4.2.: DABstep im Empfangsmodus

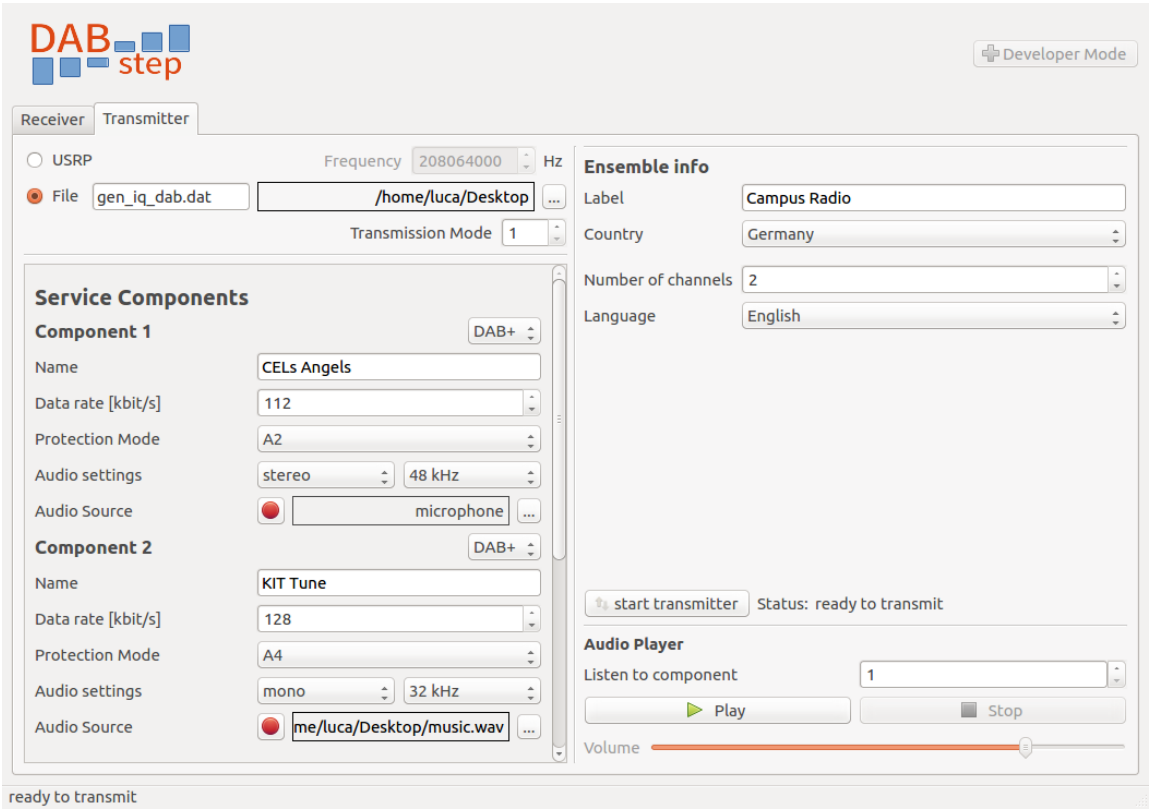
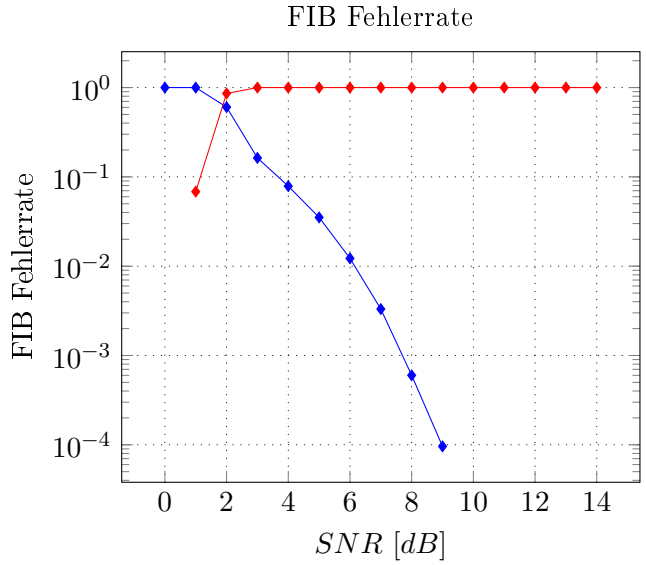
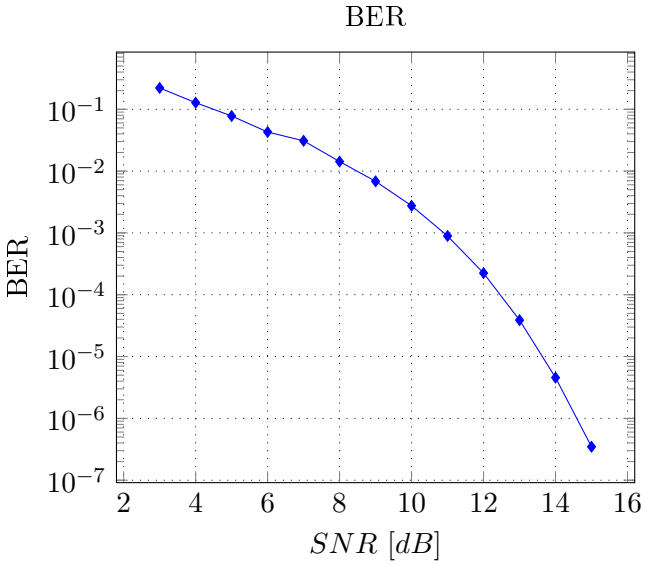


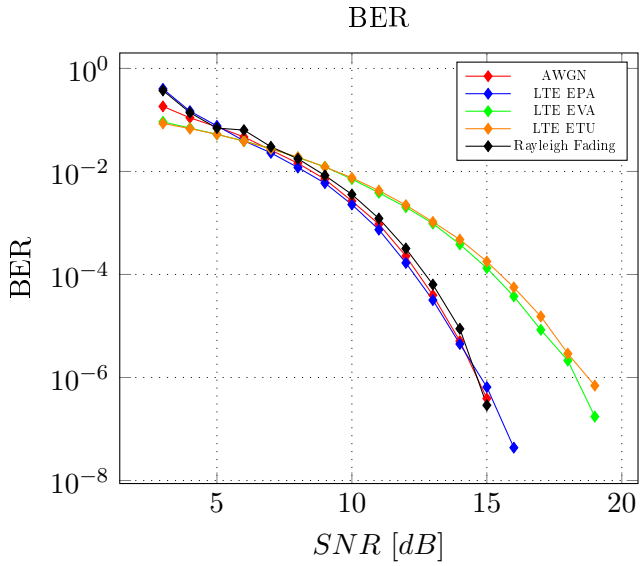
Abbildung 4.3.: DABstep im Sendemodus

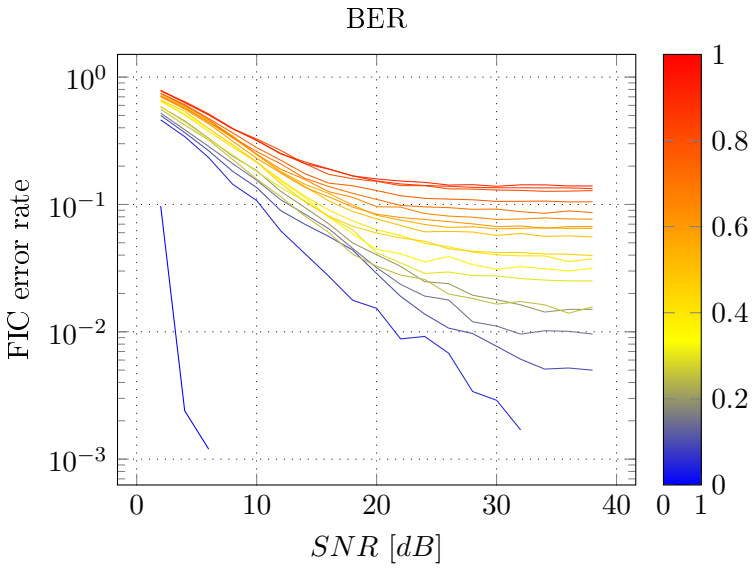
Abbildung 4.4.: DABstep im Entwicklermodus

5. Evaluation des Systems






BER
AWGN
LTE EPA
LTE EVA
LTE ETU
Rayleigh Fading
BER
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
5
10
15
20
$SNR\ [dB]$


BER
FIC error rate
10^{0}
10^{-1}
10^{-2}
10^{-3}
0
10
20
30
40
$SNR\ [dB]$
1
0.8
0.6
0.4
0.2
0
0 1

6. Fazit

So you made it! This is the last part of your thesis. Tell everyone what happened. You did something... and you could show that ... followed.
In the end make a personal statement. Why would one consider this thesis to be useful?

A. Abkürzungsverzeichnis

AWGN Additives Weißes Gaußsches Rauschen

DFT Diskrete Fourier-Transformation

FFT Schnelle Fourier-Transformation

IFFT Inverse Schnelle Fourier-Transformation

GRC GNU Radio Companion

IDFT Inverse Diskrete Fourier Transformation

MAP Maximum A-Posteriori

ML Maximum Likelihood

SDR Software-Defined Radio

ISI Intersymbol Interferenz

ICI Inter-Träger-Interferenzen

SDR Software Defined Radio

DAB Digital Audio Broadcasting

GRC GNU Radio Companion

DAB Digital Audio Broadcasting

FIB Fast Information Block

FIG Fast Information Groupt

MSC Main Service Channel

CRC Zyklische Redundanzprüfung

CIF Common Interleaved Frame

CU Capacity Unit

FIC Fast Information Channel

MCI Multiplex Configuration Information

OFDM Orthogonales Frequenzmultiplexverfahren

FDM Frequenzmultiplexverfahren

PRBS Pseudo-Random Binary Sequency

SI Service Information

MPEG 2 MPEG 1/2 Audio Layer II

MPEG 4 MPEG 4 HE-AACv2

EEP Equal Error Protection

UEP Unequal Error Protection

CAZAC Constant Amplitude Zero Auto-Correlation

ETSI Europäischen Institut für Telekommunikationsnormen

GUI Graphische Benutzeroberfläche

JSON JavaScript Object Notation

SWIG Simplified Wrapper and Interface Generator

GPL3 GNU General Public License

USRP Universal Software Radio Peripheral