Audio und Linux - Von der Soundkarte auf den Streamingserver

Friederike Maier
Radiocamp Markelfingen 2011

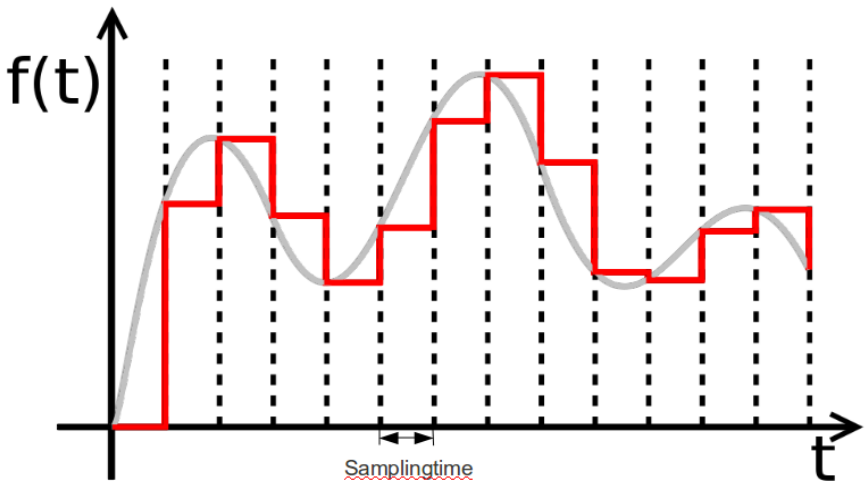
Inhalt

1 Digitalisierung von Audio

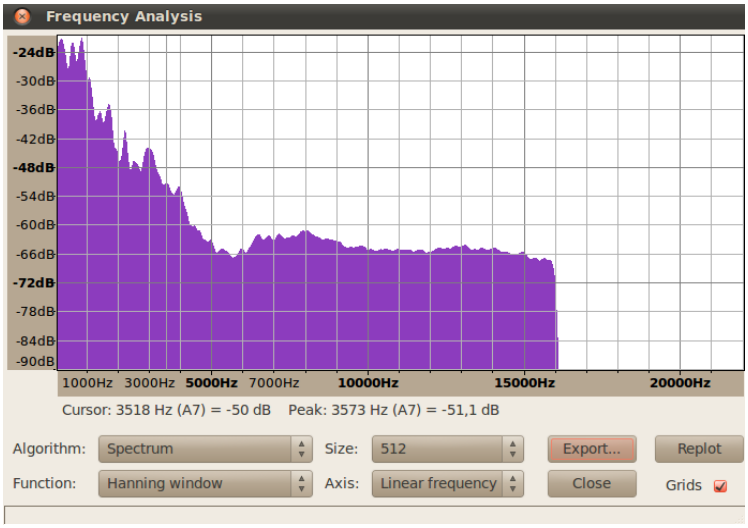
- Analoges Audiosignal soll in einen digitalen Datenstrom gewandelt werden (0010111000..)
- Audio = Luftdruckveränderungen → Mikrofon → ADC (Analog-Digital-Wandler, z.B. Soundkarte) → binäre Symbole
- Die Anzahl der Symbole/Sekunde = Samplerate
- Anzahl der Bits/Symbol = Wortbreite
- Samplerate*Wortbreite=Bitrate

1.1 Samplerate

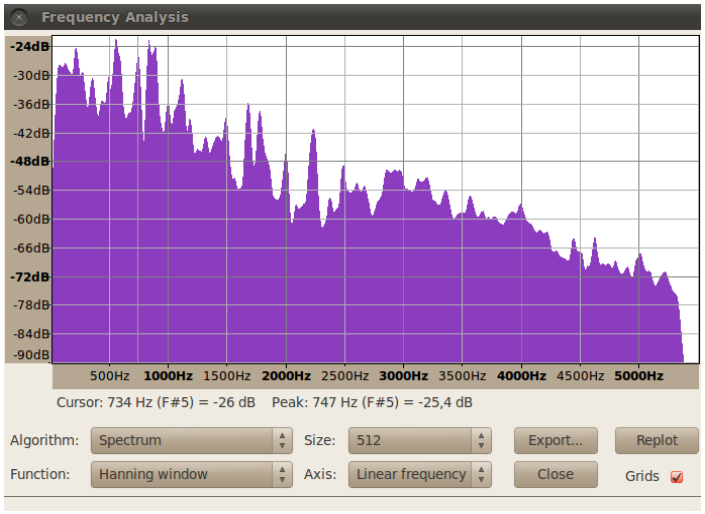
- Muss hoch genug sein um auch die hohen Frequenzen darstellen zu können → Nyquist: Die Samplingfrequenz muss mind. doppelt so groß, wie die höchste Audiofrequenz, die digitalisiert werden soll
- z.B. Samplerate: 44.1 kHz → alle 22.67 µs ein Abtastwert (hörbarer Frequenzbereich: 20 Hz – 20 kHz), passt
- Bei kleineren Samplingraten wird die Bandbreite des Audiosignals (hörbar) kleiner



© Wikipedia



Frequenzen bei einer Sampligrate von 44.1 kHz



Frequenzen bei einer Samplingrate von 11.025 kHz

1.2 Quantisierung

- Auflösung der Amplitudenwerte
- Typische Wortlängen:
- 16 bit = 2^16 = 65536 Werte
- 32 bit = 131072 Werte
- Hat Auswirkung auf das Rauschlevel, ab 24 bit überwiegt meist das Rauschlevel der Bauteile

2 Datenreduktion

- Ziel: Audiodateien/Stream kleiner bekommen
- z.B. zum Upload auf Webseite oder Internetradiostream
- Möglichst keine höbaren Änderungen der Audiodateien
- 2 Sorten von Kompression:
- Verlustfrei (z.B. Free Lossless Audio Codec (FLAC), Apple Lossless, Advanced Lossless (ATRAC))
- Verlustbehaftet (z.B. mp3, ogg, AAC...)

2.1 Datenreduktion verlustbehaftet

- Simple Verfahren (µLaw, Alaw) quantisieren die Amplituden neu

- Modernere Verfahren nutzen psychoakustische Modelle
- Das Signal wird in den Frequenzbereich transformiert
- Die Eigenschaften des Ohres werden nachgebildet und gezielt die Teile, die das 'normale' Ohr nicht wahrnimmt wegreduziert

2.2 Kompressionsartefakte

- Ausgedünntes Klangspektrum
- blubbern/gurgeln: besonders schlimm bei sehr zufälligen Signalen mit scharfem Anschlag, z.B. Applaus
- Stichwort: Generationsverlust (mehrfaches reencodieren) dabei werden in den allermeisten Fällen nicht immer dieselben Teile wieder wegkomprimiert und es klingt bei jedem mal schlechter → möglichst vermeiden!

2.3 MP3 (MPEG 1/2 Layer 3)

- Verschiedene Patente liegen auf Teilverfahren des MP3 codecs, lame >wird als best practice Programmierbeispiel als sourcecode verbreitet
- Verschiedene Einstellungen:
  - CBR: Constante Bitrate
  - ABR: Average Bitrate, Anteile mit komplexerem Audio bekommen mehr Bitrate, die mittlere Bitrate wird vorgegeben
  - VBR: Variable Bitrate, es werden verschiedene Qualitätsstufen vorgegeben, größe am ende nicht absehbar
- Lame:
  - Qualität (-q) Auswahl verschieden komplexer Algirithmen, bessere Qualität → mehr rechnen → langsamer
- Stereo: zwei getrennte Kanäle werden einzeln encodiert
- Joint Stereo: Nutzt die Korrelation zwischen R und L, mehr Bits für Qualität

2.4 OGG Vorbis

- OGG: Container, kann alles mögliche rein (Audio, Video, mp3, etc.), free, open standard

- OGG Vorbis: wurde Xiph.Org Foundation als patentfreie Alternative zum weit verbreiteten MP3-Format entwickelt

- Verbreitung: Mittelmäßig, wird von recht vielen Endgeräten nicht unterstützt

2.5 AAC (Advanced Audio Codec)

- Weiterentwicklung von mp3, ebenfalls Patente drauf, die Verbreitung von in AAC encodierten Inhalten ist jedoch umsonst

- Löst einige Probleme von mp3 (z.B. Pre Echo)

- Verschiedene Profile (z.B. HE: für niedrige Datenraten, Spektralband-Replikation)

3 Linux Audio Treiber

Linux ALSA Soundkartenunterstützung:

http://www.alsa-project.org/main/index.php/Matrix:Main

Zitat Wikipedia: 'In a typical installation scenario under Linux, the user configures ALSA to use a virtual device provided by PulseAudio. Thus, applications using ALSA will output sound to PulseAudio, which then uses ALSA itself to access the real sound card.' ;-)

- **Hardware Treiber**: ALSA , OSS
    - sprechen direkt mit der Soundkarte
- **Soundserver**: Pulseaudio, Jack, (Portaudio), ALSA
    - Häkeln die einzelnen Soundstreams zusammen
    - Ermöglichen das mixen und weiterleiten an die Treiber, wenn die Soundkarte dies nicht auf Hardwareebene ermöglicht (bzw. Die Linuxtreiber, ausnahmen z.B. Maudio etc.)
    - Softwaremixing

3.1 OSS (Open Sound System)

- Recht alt

- Unterstützt kein Multiplexing

- Wenn gar nichts anderes läuft kann mans mal damit probieren

3.2 ALSA

- Advanced Linux Sound Architecture
- Verwendung als Hardwaretreiber
- Meist Multiplexfähig
- Soundserver:
- Ist sowieso schon auf dem System aktiv
- Kürzester Weg des Sounds
- Benötigt sehr wenig Rechenzeit
- Bei Problemen erst mal alles auf ALSA umstellen:
- 'gstreamer-properties' in console

3.3 Pulseaudio

- Ist bei den meisten Ubuntus als Soundserver voreingestellt
- Einfache grafische Oberfläche
- Netzwerkfähig

3.4 JACK

- Sehr niedrige Latenzzeiten
- Sehr umfassendes Routing möglich
- Komplizierter im Gebrauch
- Benötigt vergleichsweise wenig Rechenleistung
- Zur Vereinfachung der Konfiguration gibts verschiedene tools, z.B. Qjackctl, jackEQ

3.5 Routing

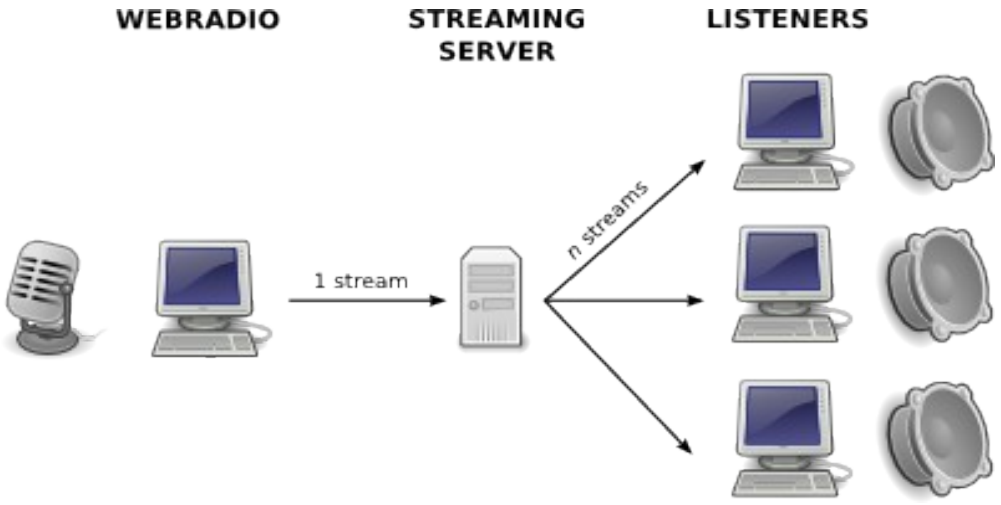
Weg, den die Ton-Signale nehmen, wenn sie durch den Computer geleitet werden

- Soundquelle -> ALSA -> PulseAudio -> ALSA-Treiber -> Hardware
- Soundquelle -> PulseAudio -> ALSA-Treiber -> Hardware
- Soundquelle -> ALSA -> ALSA-Treiber -> Hardware
- Soundquelle -> jack -> ALSA-Treiber -> Hardware

3.6 Portaudio

- Crossplattform Audiolibary
- Hauptanliegen des Projekts ist ein Plattformunabhängige Schnittstelle zu schaffen, sodass Programme (z.B. Audacity) auf verschiedenen Systemen laufen können

4 Internetstream



Stream- generator (Ices, Liquidsoap)

Streaming-server (Icecast, Shoutcast)

Mediaplayer etc.

- Streamgenerator → Server, wird ständig gesendet
- Jeder Streamgenerator erzeugt einen Mountpoint auf dem Server (z.B. http://radioflora.de:8000/test.mp3)

- Die HörerInnen verbinden zu dem Mountpoint und für jedeN wird ein Stream aufgemacht
- Bandbreite des Servers steigt mit jeder HörerIn

4.1 Liquidsoap

- Skriptsprache zum generieren von Audiostreams
- Angepasst für Radiostreams
- Sehr viele Möglichkeiten, z.B.
  - Sendungen/Jingles zu bestimmten Zeiten
  - Random, 1/5 Jingles, 4/5 Musik
  - Umschalten, wenn z.B. Livesendung verfügbar ist
  - Verschiedene Outputs
  - Hörerwünsche via Telnet
  - Deamonbetrieb auf Server
  - .....siehe http://savonet.sourceforge.net/index.html sehr gute Dokumentation und Beispiele was andere Radios machen
- Configuring Liquidsoap (Sehr gutes Getting Started: http://savonet.sourceforge.net/doc-svn/quick_start.html)
  - Source: z.B. Playlist, muss 'unfehlbar' (unfaillible) sein, sonst beschwert es sich, weil der Eingangsstream plötzlich weg sein kann, Mksave operator (spielt Stille, wenn kein Audio verfügbar), oder besser Fallbacks einrichten

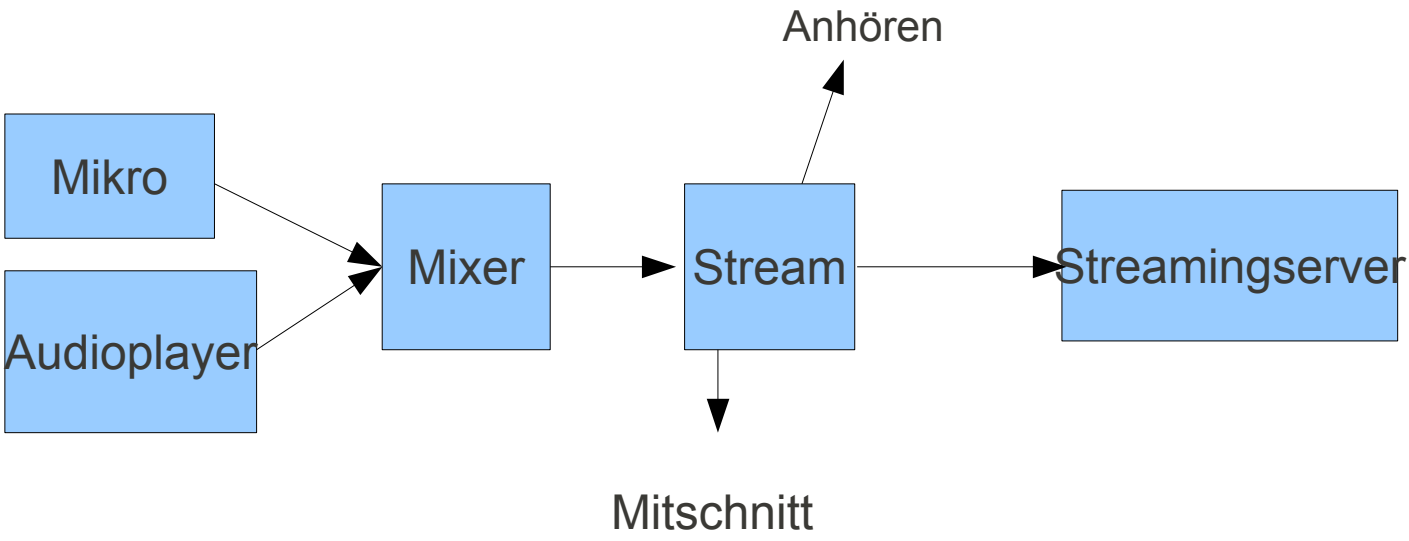
Beispiel: Simple Playlist abspielen und streamen speichern in Textdatei unter mp3stream.liq ausführbar machen mit chmod 0755 mp3stream.liq, starten in der Konsole mit ./mp3stream.liq

```
#!/usr/bin/liquidsoap -v

set("log.file",false)

set("log.stdout",true)

play = mksafe(playlist("playlist2.pls"))

out(play) # Direkt am Rechner abspielen
```

```
output.icecast.lame(host="radioflora.de",port=8000,password="XXXXX",mount="
test.mp3",play) # Streamen zum Server
```

5 Setup für ein Ein-Rechner-Internetradio

- Mikro am Eingang + Audioplayer auf Rechner, umschaltbar
- Alles auf einen Stream → Liquidsoap
- gleichzeitig mittschneiden und anhören



Player mit Jackunterstützung z.B.Aqualung oder Audacious, Mixer: JackEQ, Infos zur Installation von Jack: http://wiki.ubuntuusers.de/jack

unten gibt's auch Infos zu Programmen, die Jack unterstützen und wie man Programme, dies eigentlich nicht unterstützen zum laufen bekommt.

Jack Tutorial: http://www.linuxmintusers.de/index.php?action=wiki;page=JACK-Tutorial

5.1 Jack Konfiguration

Einrichten von Jack zum Beginn:

- Unter Konfiguration/Setup
  - Echtzeit (Wenn ein Realtime Kernel installiert ist, ansonsten nicht)
  - Gegebenenfalls die richtige Soundkarte einstellen
  - Evtl. 16 Bit Audio (wenn du eine 16 bit Soundkarte hast)

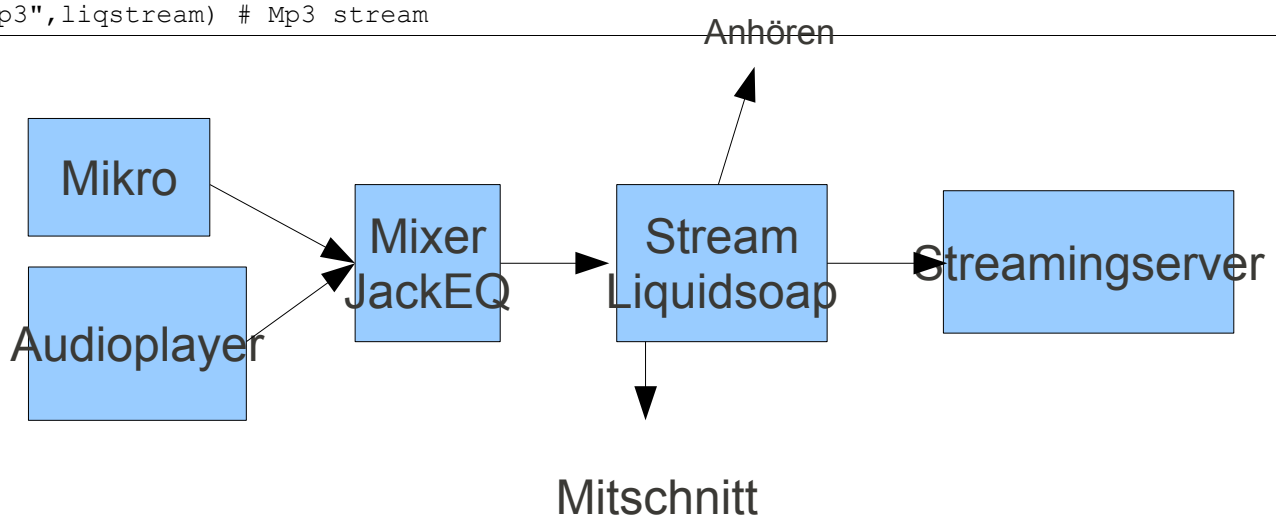
- Samplerate (Manche Soundkarten arbeiten gerne mit 48 kHz)
- Soft Mode: Jack stoppt nicht, wenn er xruns bekommt

- Latenz: gibt an, wieviel Verzögerung zwischen dem Audio Input (das spielen eines Tons auf dem MIDI Keyboard beispielsweise) und dem Hören des Output besteht
  - Frames/Periode → Latenz rechts unten
  - Balance zwischen xruns und Verzögerung

5.2 Liquidsoap mit Jack

Beispielkonfiguration:

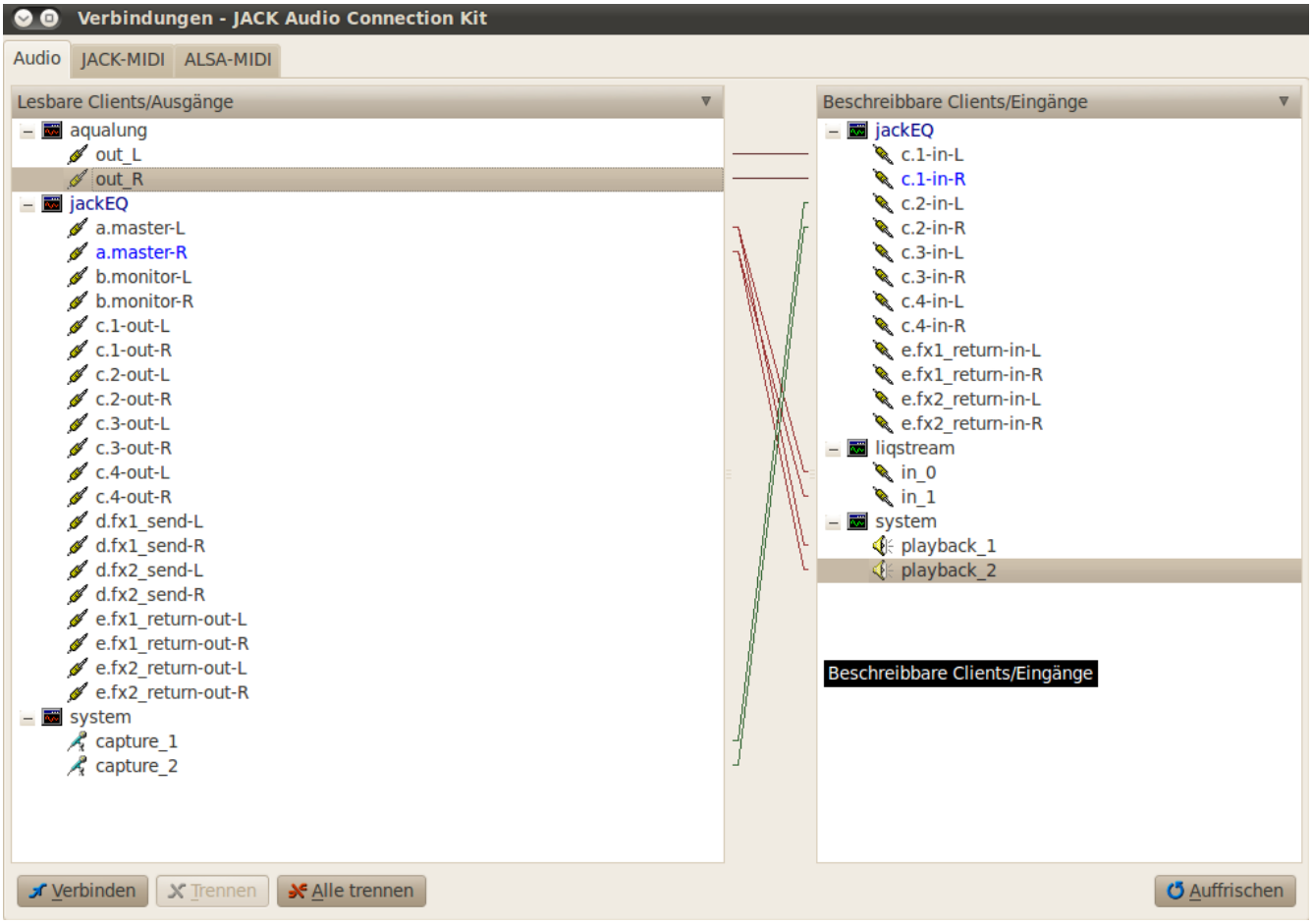
```
#!/usr/bin/liquidsoap

### global settings

set("frame.channels",2)

set("log.file",true)

set("log.file.path","/home/zound/radio/livestreaming/bin/liq/livestream.log")

set("log.stdout",true)

### jack input with the same number of channels than the global setting above

liqstream=mksafe(input.jack(id="liqstream"))

### output liqstream

output.file.vorbis("mitschnitt.ogg",liqstream) # ogg Mitschnitt

output.icecast.lame(host="radioflora.de",port=8000,password="XXXXXXXX",mount="test.
mp3",liqstream) # Mp3 stream
```



5.3 Jack Verbindungen

Im Tab Verbindungen:

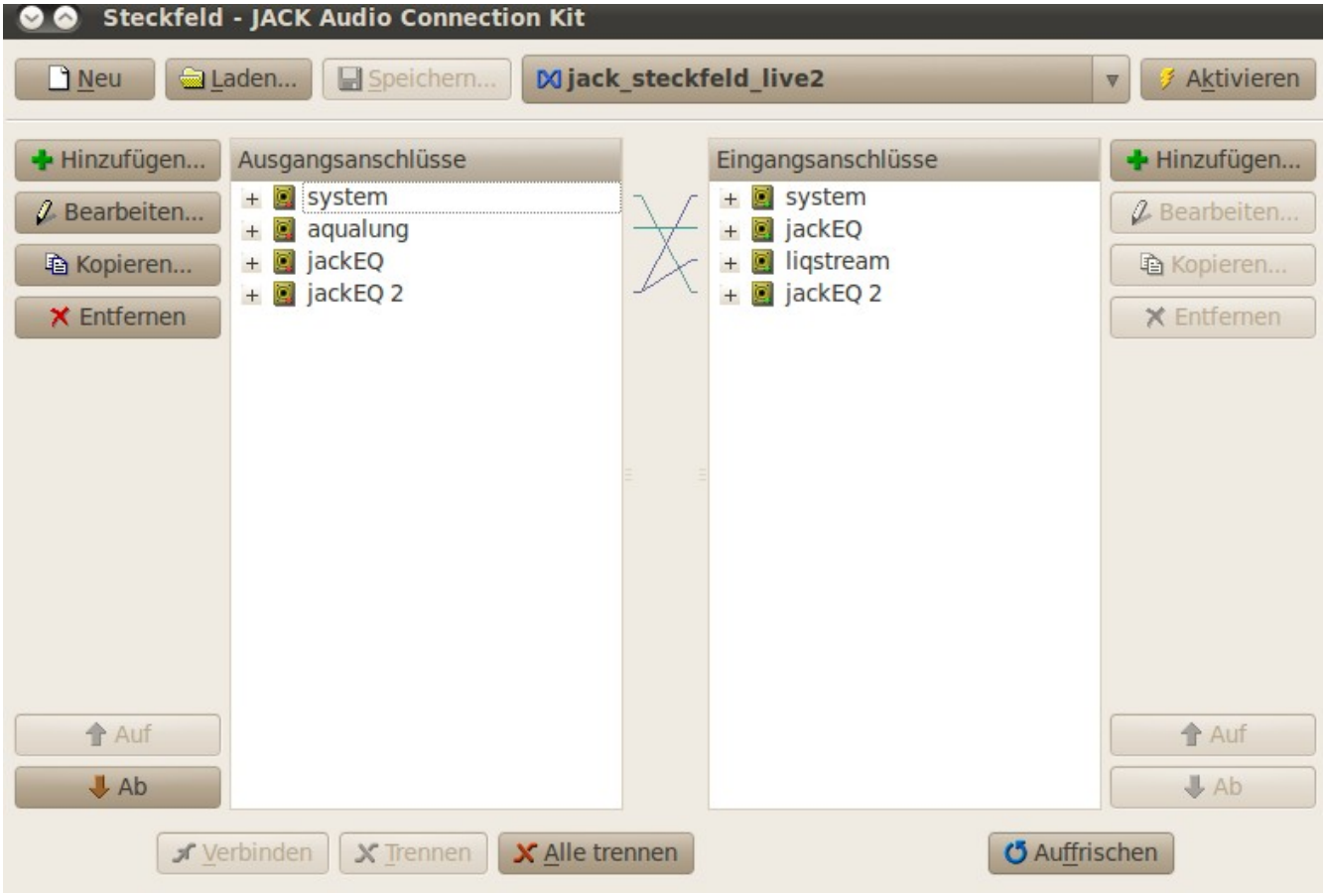
Es werden Aqualung mit Eingang 1 (c.1-in-L/R) und System Capture, also das Mikrofon mit Eingang 2 verbunden, sodass mit JackEQ zwischen Musik und Moderation umgeschaltet werden kann. Der Ausgang von JackEQ wird mit Liquidsoap verbunden und mit dem System, playback, also dem Kopfhörerausgang zum mithören.



5.4 Abspeichern der Verbindungen

Tab Steckfeld:

- Neues Steckfeld; Auf Frage, ob das die aktuellen Verbindungen als Snapshot verwendet werden sollen mit ja antworten; abspeichern


Steckfeld - JACK Audio Connection Kit
Neu
Laden...
Speichern...
jack_steckfeld_live2
Aktivieren
Hinzufügen...
Bearbeiten...
Kopieren...
Entfernen
Ausgangsanschlüsse
system
aqualung
jackEQ
jackEQ 2
Eingangsanschlüsse
system
jackEQ
liqstream
jackEQ 2
Hinzufügen...
Bearbeiten...
Kopieren...
Entfernen
Auf
Ab
Auf
Ab
Verbinden
Trennen
Alle trennen
Auffrischen

FERTIG :-)