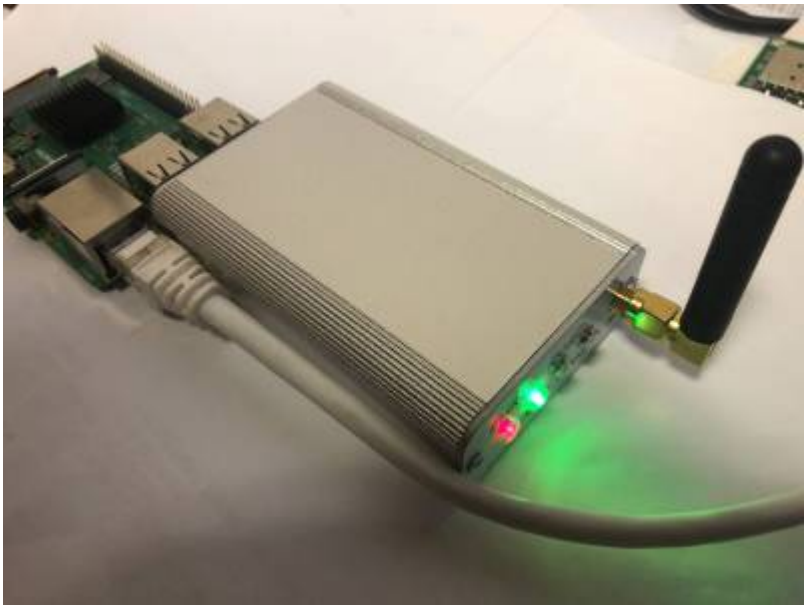
Hinweis:
Ich bin und werde kein Supporter für diesen Hotspot ! Ich gebe nur Hinweise für dessen Inbetriebnahme, da ich ihn selber auch besitze und einsetze.
Der Betrieb ist nur lizenzierten Funkamateuren gestattet, eine anderweitige Verwendung z.B. für PMR446 oder Freenet ist aufgrund der geltenden Gesetze und Vorschriften nicht zulässig !

# FM/analog Hotspot SHARI für SVXLINK & Raspberry Pi mit SA818

letzte Aktualisierung: 27.1.2023

Bei Aliexpress gibt es hier ein FM/analog-Hotspot-USB-Modul für ca. 60..70€ zum Anstecken an einen Raspberry Pi >2B oder auch anderen Kleincomputern mit Linux. Dieses Modul besteht aus folgenden Komponenten:



- integrierte Soundkarte CM108B inkl. integrierter GPIO-Steuerung für SQL-Detection und PTT-Steuerung mittels HID_RAW_DEVICES
- FM-TRX-Modul SA818 (0,5W/1W) UHF/VHF, je nach Ausführung, üblicherweise meist UHF/70cm
- USB-to-Serial Converter CH340 zur Programmierung des SA818

**Hinweis: Es gibt inzwischen wohl eine andere/geänderte Version, die nicht den SA818, sondern einen SR110U als FM-TRX-Modul verwendet. Weitere Infos sind hier zu finden.**

Dieses Modul basiert auf einem aus den USA stammenden Hamradio-Projekt, genannt SHARI (SA818 Ham Allstar Radio Interface).
Von einem Klone würde ich nicht unbedingt sprechen, verwendet werden letztlich bekannte Komponenten wie der CM108B-USB-Audiochip, das FM-TRX-Modul von NiceRF SA818S (eine andere Ausführung verwendet das FM-TRX-Modul SR110U von Sunrisedigit) und einem USB-UART-Chip CH34x zur Programmierung der FM-TRX-Module. Alle Beschaltungen kann man den Datenblättern der eingesetzen Komponeten entnehmen - es wird also alles Standard-Hardware verwendet. Ich hatte selbst schon einen Eigenbau-Hotspot in Benutzung, der exakt die gleichen Komponenten verwendete,

nur nicht so schön kompakt konstruiert war 😉

⚠ Es ist USB-Bus-powered, wird also via USB mit Spannung versorgt und benötigt demnach keine eigene Stromversorgung. Allerdings sind hier USB-Ports erforderlich, die bis zu 1A Strom liefern können, was beim Raspberry Pi durch eine kleine Modifikation der **`/boot/config.txt`** möglich ist:

```
max_usb_current=1
```

Danach kann man die USB-Ports des Raspberry Pi mit bis zu 1,2A belasten.

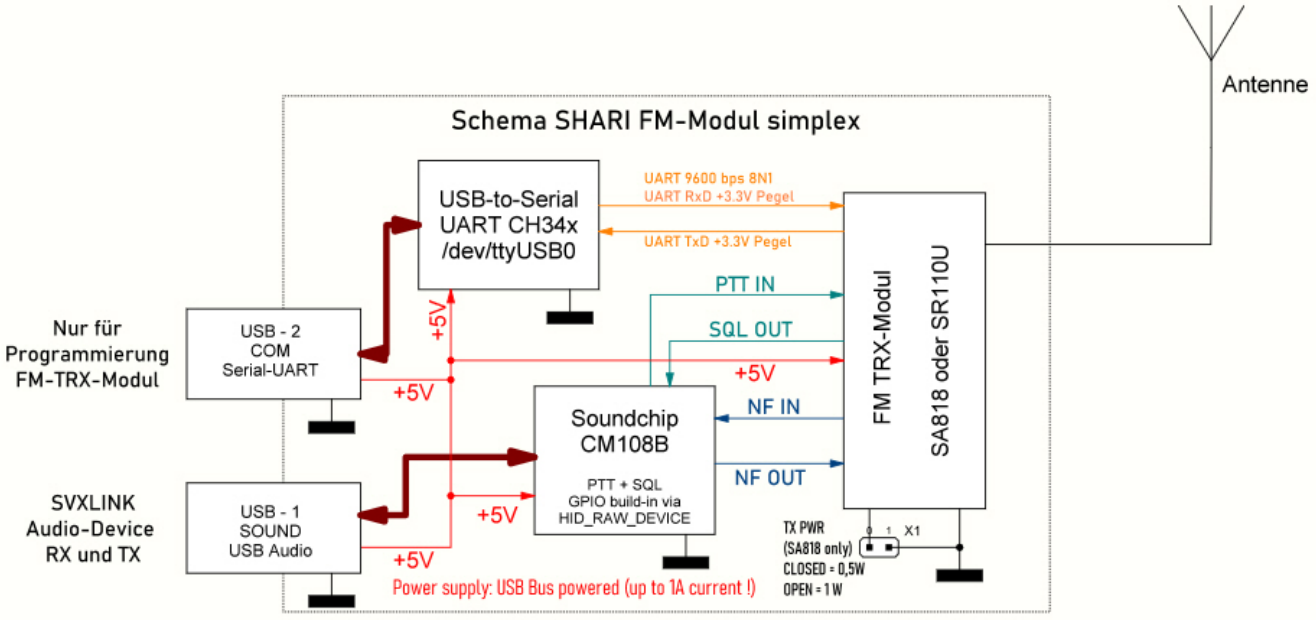
⚠ NUR beim SA818: Die Modulplatine besitzt einen nicht bestückten Jumperblock. Ergänzt man den und brückt diesen Jumper, verringert sich die Sendeleistung von 1W auf dann nur 0.5W. Leider lässt sich das SA818 nicht per Software auf kleinere Leistung umschalten, das ist nur durch diese zusätzliche Brücke möglich.

Das Modul besitzt 2 USB-Anschlüsse, die wie folgt beschalten sind (Sicht von hinten):
USB links: USB-to-Serial Converter CH340 zur Programmierung des SA818
USB rechts: integrierte Soundkarte CM108B

⚠ Hinweise: Nach erfolgter Programmierung (!) des SA818 ist nur der rechte USB-Port (Soundkarte) erforderlich, der linke USB-Port muss nicht zwingend angeschlossen werden.

Weiterhin gibt es eine LED, die mit COS bezeichnet wird. Diese ist direkt an einen der GPIO des CM108B-Soundchips verschalten und lässt sich - leider - *derzeit mit SVXLINK nicht nutzen*, dafür wären Quellcodeänderungen im SVXLINK erforderlich. Bei der in den USA üblichen Alternative zum in Europe verbreiteten SVXLINK, genannt AllStarLink, kann man die COS-LED wohl ansteuern. Denn SHARI, also dieses Modul, wurde ja mal für den AllStarLink entwickelt.

## Schemadarstellung des SHARI-FM-Moduls



Wenn das FM-TRX-Modul bereits programmiert ist, benötigt man zum Betrieb mit SVXLINK nur den USB-SOUND-Anschluß. Der USB-COM ist nur zur Programmierung notwendig und kann im Normalbetrieb frei bleiben bzw. muss nicht zwingend angeschlossen werden. Der Einsatz einer USB-Verlängerung ST-BU zwischen Computer und Modul ist möglich, diese sollte aber geschirmt und nicht zu lang sein (ca. 0.5..1m). Zur Sicherheit die USB-Verlängerung mit Klappferriten gegen HF abblocken, am besten auf beiden Seiten (ST und BU).

## Vorbemerkungen - bitte genau lesen

Diese Hotspot-Lösungen oder besser als FM-Simplex-Microrepeater bezeichnet, sind keine Plug'n'Play-Lösungen. Viele denken das vielleicht, das ist aber zu kurz gedacht, weil es so nicht zutrifft. Es sind immer, auch bei diesen Lösungen, Einstellungen und Abgleicharbeiten erforderlich. Das gilt besonders für den Abgleich der korrekten Audio-Pegel des RX und TX.
Meine Hilfestellungen und Anleitungen sind für Anwender gedacht, die in der Lage sein müssen, mit einem Raspberry Pi und dessen Betriebssystem Raspian/Linux grundsätzlich umgehen zu können. Der Anwender muss in der Lage sein, Pakete zu installieren, Dateien zu editieren, per SSH auf den Pi zugreifen zu können - kurz gesagt, ein Basiswissen mit dem Umgang von Linux besitzen. Was es hier nicht geben wird, ist ein Grundkurs im Umgang mit Linux, das setze ich voraus, um meine Hilfestellungen korrekt anwenden zu können. Das schließt ebenfalls Kentnisse der Funktionsweise SVXLINK ein, man sollte wenigstens in der Lage sein, dessen Konfigurations-Dateien zu kennen und entsprechend bearbeiten und anpassen zu können.

Ich empfehle dringend, die leistungsstärkeren Raspberry Pi wie 2B/3B/3B+/4B oder auch P400 zu verwenden, die wenigstens über 1GB RAM verfügen. Ob ein Pi ZERO hier auch mitspielt, habe ich nicht getestet und werde es auch nicht, vielleicht der derzeit leider nicht erhältliche Pi ZERO 2 könnte ggf. noch in Frage kommen.

Zur Inbetriebnahme nutzt bitte nicht reichweitenstarke Talkgruppen wie die 777 oder ähnliches. Zum Testen ist am FM-Funknetz die TG20 vorgesehen, macht also erste Tests bitte dort und stört bitte nicht andere Talkgruppen wie die TG777 mit Euren ersten Tests. Das ist im Sinne aller und stellt ein Mindestmaß an Qualität des Gesamtsystems sicher. Beachtet das bitte, danke. Es ist auch immer eine Frage von Hamspirit dem QSO-Partner gegenüber, gut eingestellte und ordentlich kalibrierte Technik zu verwenden. Jeder Sysop eines öffentlichen Relais muss das sicherstellen aufgrund der Lizenzauflagen seitens der Behörde für das Relais, also sollte das auch den Betreibern von Hotspots ebenfalls Ansporn genug zu sein.

## Erster Schritt: Programmierung des SA818-FM-TRX

Die Programmierung des SA818 basiert auf dessen Datenblatt. Bei angestecktem Modul erzeugt der USB-to-Serial Converter CH340 eine Schnittstelle **`/dev/ttyUSB0`** , über die wir die Programmierung des SA818-TRX-Moduls ausführen werden.
Folgende Parameter werden gesetzt: **RX/TX Frequenz 430.025MHz, CTCSS 77Hz RX und TX, SQL-Level 3, Lautstärke NF-OUT 7, flataudio, alle Filter AUS**
Wir erzeugen die CTCSS-Auswertung im Modul, was per SQL-Signalleitung an den GPIO des CM108B-Soundchips zur Auswertung übergeben wird. Ebenso erzeugen wir CTCSS im Sendefall, was auch durch das Modul selbst und nicht per Software aus dem SVXLINK erzeugt wird. Sende- und Empfangsfrequenz sind gleich, denn es handelt sich ja hier um ein FM-Simplex-System.
Jetzt brauchen wir erstmal Python3:

```
$ sudo apt-get install python3-dev python3-pip
$ sudo pip3 install pyserial
```

Ich habe dafür ein kleines Python3-Script `sa818-running.py` geschrieben, was diese Programmierung umsetzt:

```
#!/usr/bin/env python3

import serial

serport = '/dev/ttyUSB0'

baud = '9600'

channelspace = '1'      # 0=12.5kHz, 1=25kHz

rxfreq = '430.0250'     # TX frequency
txfreq = rxfreq         # Same as rx freq - we work simplex

squelch = '3'           # 0-8 (0 = open)

txcxcss = '0004'        # CTCSS 77Hz
rxcxcss = '0004'        # CTCSS 77Hz
# txcxcss = rxcxcss

# txcxcss = '023N'        # CTCSS / CDCSS TX
# rxcxcss = '023N'        # CTCSS / CDCSS RX

flataudio = '1'           # switch to discriminator output and input if
value = 1
bypass_lowpass = '1'      # bypass lowpass-filter if value = 1
bypass_highpass = '1'     # bypass highpass-filter if value = 1

volume = '7'              # betweeen 0..8

ser = serial.Serial(serport, baud, timeout=2)
print('Opening port: ' + ser.name)

print ('\r\nConnecting...')
ser.write(b'AT+DMOCONNECT\r\n')

output = ser.readline()
print ('reply: ' + output.decode("utf-8"))

print ('\r\nConfiguring radio...')
config = 'AT+DMOSETGROUP={},{},{},{},{},{}\r\n'.format(channelspace, txfreq,
rxfreq, txcxcss, squelch, rxcxcss)
print (config)
ser.write(config.encode())
output = ser.readline()
print ('reply: ' + output.decode("utf-8"))
```

```
print ('\r\nSet filter...')
config = 'AT+SETFILTER={},{},{}\r\n'.format(flataudio, bypass_highpass,
bypass_lowpass)
print(config)
ser.write(config.encode())
output = ser.readline()
print ('reply: ' + output.decode("utf-8"))

print ('\r\nSetting volume...')
config = 'AT+DMOSETVOLUME={}\r\n'.format(volume)
print(config)
ser.write(config.encode())
output = ser.readline()
print ('reply: ' + output.decode("utf-8"))

print ('\r\nSetting emission tail tone...')
ser.write(b'AT+SETTAIL=0\r\n')
output = ser.readline()
print ('reply: ' + output.decode("utf-8"))

print ('\r\nGetting Module Version...')
ser.write(b'AT+VERSION\r\n')
output = ser.readline()
print ('reply: ' + output.decode("utf-8"))

print ('\r\nGetting Settings...')
ser.write(b'AT+DMOREADGROUP\r\n')
output = ser.readline()
print ('reply: ' + output.decode("utf-8"))
```

## Vorbereitung Linux / Raspian

Getestet habe ich das alles unter Raspian/Debian11(Bullseye) lite, sollte aber auf älteren Systemen wie Buster auch gehen.
Der Soundchip CM108B wird vom Linux nativ erkannt, es sind also keine Zusatzarbeiten wie bei anderen HAT-Lösungen erforderlich, gleiches gilt für den USB-to-Serial Converter CH340/CH341.

Zuerst legen wir unter `/etc/udev/rules.d` eine neue Datei `90-cm108.rules` an und ergänzen diese mit folgendem Inhalt:

```
# block pulseaudio using the soundcard for SVXLINK
ATTRS{idVendor}=="0d8c", ATTRS{idProduct}=="013c", ENV{PULSE_IGNORE}="1"
ATTRS{idVendor}=="0d8c", ATTRS{idProduct}=="000c", ENV{PULSE_IGNORE}="1"
ATTRS{idVendor}=="0d8c", ATTRS{idProduct}=="0012", ENV{PULSE_IGNORE}="1"
# create a symlink /dev/hidrawX to /dev/cm108gpio
SUBSYSTEM=="hidraw", ATTRS{idVendor}=="0d8c", ATTRS{idProduct}=="013c",
SYMLINK+="cm108gpio", MODE="0666"
# 0d8c:000c
SUBSYSTEM=="hidraw", ATTRS{idVendor}=="0d8c", ATTRS{idProduct}=="000c",
SYMLINK+="cm108gpio", MODE="0666"
```

```
# 0d8c:0012 CM108B SHARI
SUBSYSTEM=="hidraw", ATTRS{idVendor}=="0d8c", ATTRS{idProduct}=="0012",
SYMLINK+="cm108gpio", MODE="0666"
```

Das erzeugt einen logischen Link von `/dev/hidrawX` zu `/dev/cm108gpio` , welchen wir als Deviceangabe in der `/etc/svxlink/svxlink.conf` später nutzen werden.

Weiterhin ändern wir folgende Zeile der `/boot/config.txt` :

```
dtoverlay=vc4-kms-v3d
```

in

```
dtoverlay=vc4-kms-v3d,audio=off
```

Das deaktiviert den HDMI-Audio (aber nicht das HDMI Video!), damit wir keine Probleme mit SVXLINK bekommen. Ich hatte mit einer Soundkarte und SVXLINK Probleme, wenn der HDMI-Audio aktiviert war. Nach dem Deaktivieren des HDMI-Audio war dieses Problem nicht mehr aufgetreten. Das ist also mehr als Vorsorgemaßnahme zu verstehen.
Jetzt starten wir mal den ganzen Pi einfach neu:

```
$ sudo reboot
```

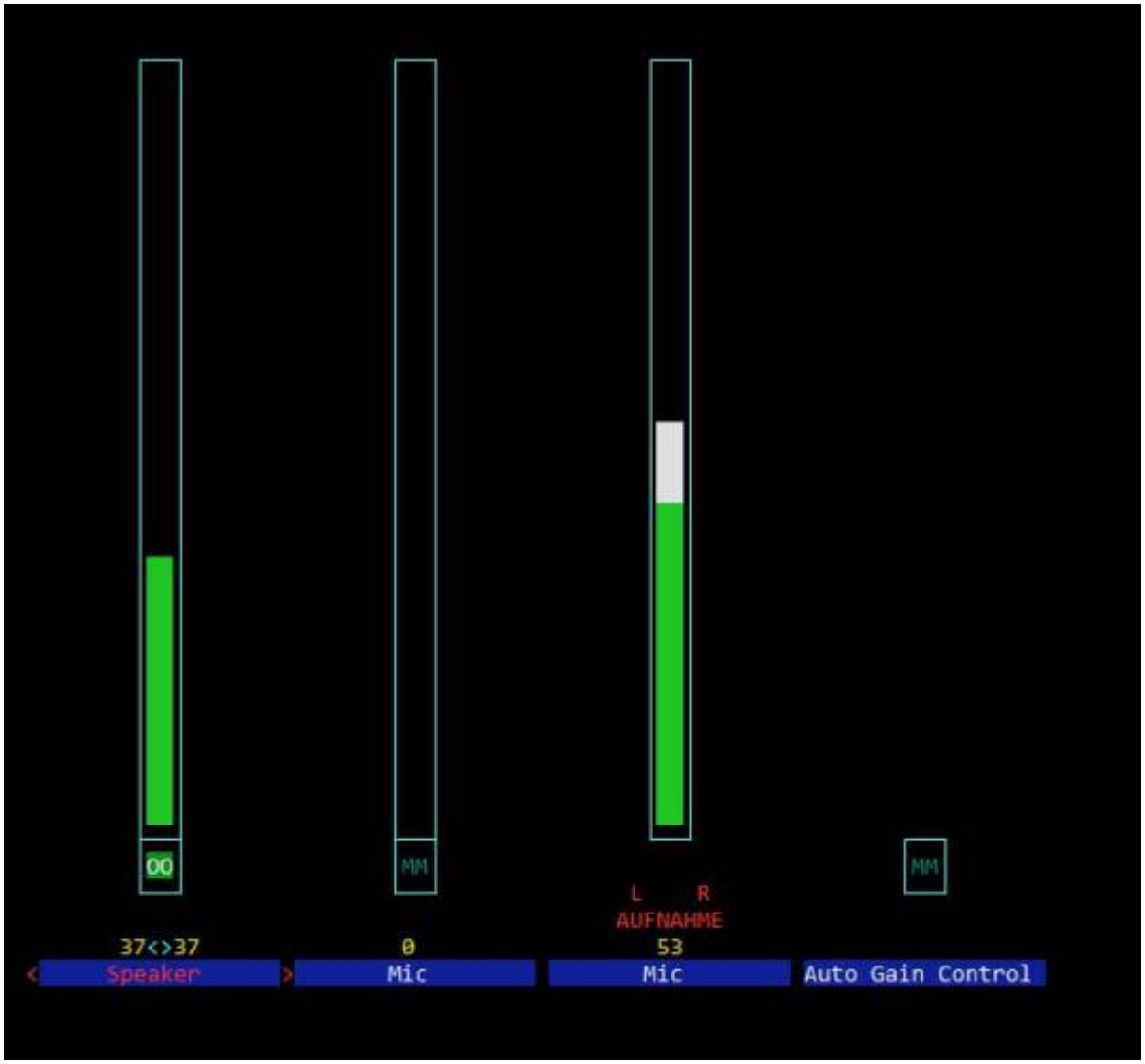
## Einstellungen der Soundkarte für SVXLINK

Wenn das SA818-Modul so programmiert wurde wie angegeben, müssen wir die Audiopegel der Soundkarte anpassen:

```
$ sudo alsamixer -c 1
```

Meist wird der Soundchip CM108B als Card 1 (und nicht 0) im Linux gelistet, deswegen auch der Parameter -c 1, der dem Alsamixer übergeben werden muss.

Jetzt einmal F5 drücken und den Mixer (nur bei einem SA818 - für andere FM-Module gelten andere Mixer-Einstellungen !) so einstellen:




Bitte einmal mit den Cursortasten auf *Auto Gain Control* gehen und die Taste m für Mute drücken, dort muss dann *MM* angezeigt werden. Das "Muten" (= Stummschalten bzw. Deaktivieren) der *Auto Gain Control* gilt generell für diesen Soundchip CM108B im Zusammenspiel mit SVXLINK, und zwar unabhängig vom verwendeten FM-TRX-Modul. Anderenfalls führt das sonst zu nicht kontrollierbaren Effekten beim Verarbeiten der Audiopegel des RX wie Verzerrungen oder Übersteuerungen.

## Notwendige Anpassungen der svxlink.conf

Voraussetzung ist, ein korrekt installiertes SVXLINK zu haben und die Schritte so umgesetzt wie eben beschrieben zu haben.
Wichtig bei der Programmierung der SA818 sind unbedingt diese Parameter, *die nicht geändert werden sollten*:

```
# flataudio with all filters OFF, nur wenn SA818-Modul
AT+SETFILTER=1,1,1
# optimal audio volume output for AUDIO-IN soundcard
AT+DMOSETVOLUME=7
# Modul rgr-beep aus, nur wenn SA818-Modul
```

```
AT+SETTAIL=0
```

Dinge wie Frequenz, SQL-Level, CTCSS können natürlich je nach Belieben verändert/angepasst werden. Dazu verweise ich erneut auf das Datenblatt des SA818, wo alles drinsteht, was man dafür benötigt. Zum Programmieren des SA818 kann man mein Beispiel-Python3-Script verwenden oder auch andere Tools Eurer Wahl, die eine Programmierung des SA818 ermöglichen.

Hier die Anpassungen für dieses Modul in der `/etc/svxlink/svxlink.conf` :

 Hinweis: Es werden *nur die Optionen* dargestellt, die *angepasst* werden müssen.

```
[GLOBAL]
LOGICS=SimplexLogic,ReflectorLogic
CARD_SAMPLE_RATE=48000
CARD_CHANNELS=1

[SimplexLogic]
TYPE=Simplex
RX=Rx1
TX=Tx1
# CTCSS AUS das 818 erzeugt das selbst
# TX_CTCSS=ALWAYS
OPEN_ON_SQL=500
MUTE_RX_ON_TX=1
MUTE_TX_ON_RX=1

[Rx1]
AUDIO_DEV=alsa:plughw:1
AUDIO_CHANNEL=0
SQL_DET=HIDRAW
SQL_START_DELAY=50
# stoppt unkontrolliertes Auftasten des Senders
SQL_DELAY=200
SQL_HANGTIME=0
# vermeidet eine Rauschsperrenschleppe aufgrund zu langsamen Schliessen der
SQL
# Audio wird um den angegebenen Zeitfaktor in ms abgeschnitten
SQL_TAIL_ELIM=100
# HID-DEVICE als Link wie in /etc/udev/rules.d/90-cm108.rules definiert
HID_DEVICE=/dev/cm108gpio
HID_SQL_PIN=VOL_DN
SIGLEV_DET=NONE
# wir betreiben die SA818 als flataudio
DEEMPHASIS=1
PREAMP=0
PEAK_METER=1

[Tx1]
AUDIO_DEV=alsa:plughw:1
AUDIO_CHANNEL=0
```

```
PTT_TYPE=Hidraw
# HID-DEVICE als Link wie in /etc/udev/rules.d/90-cm108.rules definiert
HID_DEVICE=/dev/cm108gpio
HID_PTT_PIN=GPIO3
# HS: bei den 818 braucht der TX etwas laenger
TX_DELAY=800
# wir betreiben die SA818 als flataudio
PREEMPHASIS=1
MASTER_GAIN=0.0
```

Dananch starten wir einfach mal SVXLINK neu:

```
$ sudo systemctl restart svxlink.service
```

Damit sind wir mit der notwendigen Konfiguration durch und das Hotspotmodul sollte laufen.

## OPTIONAL: Anpassungen für den Einsatz eines zusätzlichen MMDVM_HAT bzw. MMDVM_DUAL_HAT auf dem eingesetzten Raspberry Pi

Da wir ja den GPIO-Connector des Raspberry Pi frei haben, spricht natürlich nichts dagegen, ein MMDVM-Hotspot-Modem gleich mit auf diesem Pi zu betreiben. Folgende Anpassungen an der `/boot/config.txt` wären umzusetzen:

```
# Enable audio (loads snd_bcm2835)
#dtparam=audio=on
dtoverlay=vc4-kms-v3d,audio=off

[cm4]
# Enable host mode on the 2711 built-in XHCI USB controller.
# This line should be removed if the legacy DWC2 controller is required
# (e.g. for USB device mode) or if USB support is not required.
otg_mode=1

[pi4]
# Run as fast as firmware / board allows
arm_boost=1

[all]
max_usb_current=1
enable_uart=1
dtoverlay=disable-bt
```

Dann stoppen wir am besten noch den Blutooth-Service, um Konflikte mit der internen UART-Schnittstelle zu vermeiden, denn die wird vom MMDVM-Modem benötigt:

```
$ sudo systemctl stop hciuart
$ sudo systemctl disable hciuart
```

Nun kann man die nötigen Tools wie MMDVMHost, MMDVMCal etc. compilieren und installieren und

man hat sozusagen einen FM/analog-MMDVM-Hotspot in Einem und spart sich damit einen weiteren Raspberry Pi. Natürlich sollte man die Freqenzabstände beider TRX-Module analog und digital möglichst weit voneinander trennen.

## Verbesserung des Verhaltens der Rauschsperre des FM-TRX-Moduls

Bei den SA818 wie auch bei den SR110U schliesst die Rauschsperre (SQL oder CTCSS) relativ langsam, was zur Folge hat, das die Gegenstelle nach dem Loslassen der eigenen PTT noch für einen kurzen Moment ein Rauschen übertragen bekommt. Um diesen Effekt zu verrringern bzw. zu vermeiden, sind in der `svxlink.conf` folgende Anpassungen zu machen:

```
[Rx1]
SQL_START_DELAY=50
# HS: stoppt gelegentliches unkontrolliertes Auftasten des Senders, wobei
Ursache leider unklar
SQL_DELAY=200
# sofort SVXLINK das Schliessen der SQL melden, Wert bedeutet 0ms
SQL_HANGTIME=0
# die letzten 100ms Audio abschneiden evtl. mit Werten zwischen 100 bis 200
experimentieren
SQL_TAIL_ELIM=100
```

## Zusatzhinweise und Fazit

Von allen Hotspotlösungen (DJSpot, SPspot), die ich kenne, habe ich mit diesem Modul die besten Ergebnisse erzielt, zumindest was die Version mit dem SA818 betrifft. Das betrifft besonders auch die Audioqualität, was mich ein wenig postiv überrascht hat. Zu der neueren Version mit dem SR110U anstelle des SA818 kann ich derzeit noch kein Fazit abliefern.
Durch das Aluminiumcase ist auch wenigstens ein bisschen EMV-Behandlung vorhanden, wie es sich für sendefähige HF-Baugruppen eigentlich gehört.

Sollte es zu Brummproblemen kommen, ist in 95% aller Fälle das Problem einer fehlenden Masseverbindung in der Spannungsversorgung des Pi die Ursache, denn dann kommt es zu Netzbrummen in Kombination mit der Soundkarte. Leider zeigen auch die Original-Netzteile vom Raspberry Pi diese Probleme, wie ich selber schon feststellen durfte. HF-Einstreuung ist eher seltener die Ursache. Hier also mal einfach mit verschiedenen Netzteilen experimentieren und wenn man sich nicht sicher ist, einfach mal einen Akku-Powerpack anstatt eines Netzteils verwenden. Denkt aber daran, wir benötigen ca. 5,1V und mind. 2,5A für den Raspberry Pi. Ist das Brummen dann weg, liegt es eindeutig an der Spannungsversorgung bzw. deren fehlender oder unzureichender Masseverbindung. Warum haben wohl unsere KW-TRX alle eine Erdungsanschluß ???

---

73 Heiko, DL1BZ
Januar 2023