Compilierung und Installation SVXLINK aus den Github-Sourcen

last update: 15.7.2024

WICHTIG - in eigener Sache (15.7.2024)

Alle meine bisherigen Beiträge in diesem Wiki in Sachen SVXLINK beziehen sich nur bzw. einschließlich bis zum Zeitpunkt des Erscheinens des letzten offiziellen Release SVXLINK 24.02 vom 25.2.2024.
In den darauf folgenden Entwicklerversionen, also Versionen nach dem 25.2.2024, wird es grundlegende Änderungen geben.
Diesen werde ich mich allerdings nicht mehr zuwenden.

73 Heiko, DL1BZ

0. Vorwort

Hier folgt eine Anleitung, wie man auf einem Standard-Linux-System SVXLINK aus den Github-Sourcen selber compilieren und installieren kann. Ich beziehe mich mit dieser Anleitung auf Linux/Debian 11 (Bullseye), eingeschlossen Raspian für den Raspberry Pi, andere Distributionen werden hier nicht berücksichtigt bzw. erfordern andere/abweichende Anpassungen.

Auch hier gilt, das ist nix für reine Anfänger, denen noch Grundlagenwissen in Sachen Linux fehlt. Grundlegende Kenntnisse im Umgang und der Bedienung eines Linux-System sind also zwingende Voraussetzung, um die folgenden Schritte umsetzen zu können.
Ein Einsatz „schwacher“ Raspberry Pi wie u.a. dem ZERO/ZERO-W ist für das hier dargestellte Vorgehen definitiv NICHT zu empfehlen, alles ab 2B oder höher ist aber völlig ok !

1. Vorbereitung

1.1 Gibts bereits ein fertiges Image ? Das wäre doch so schön einfach...oder doch nicht ?

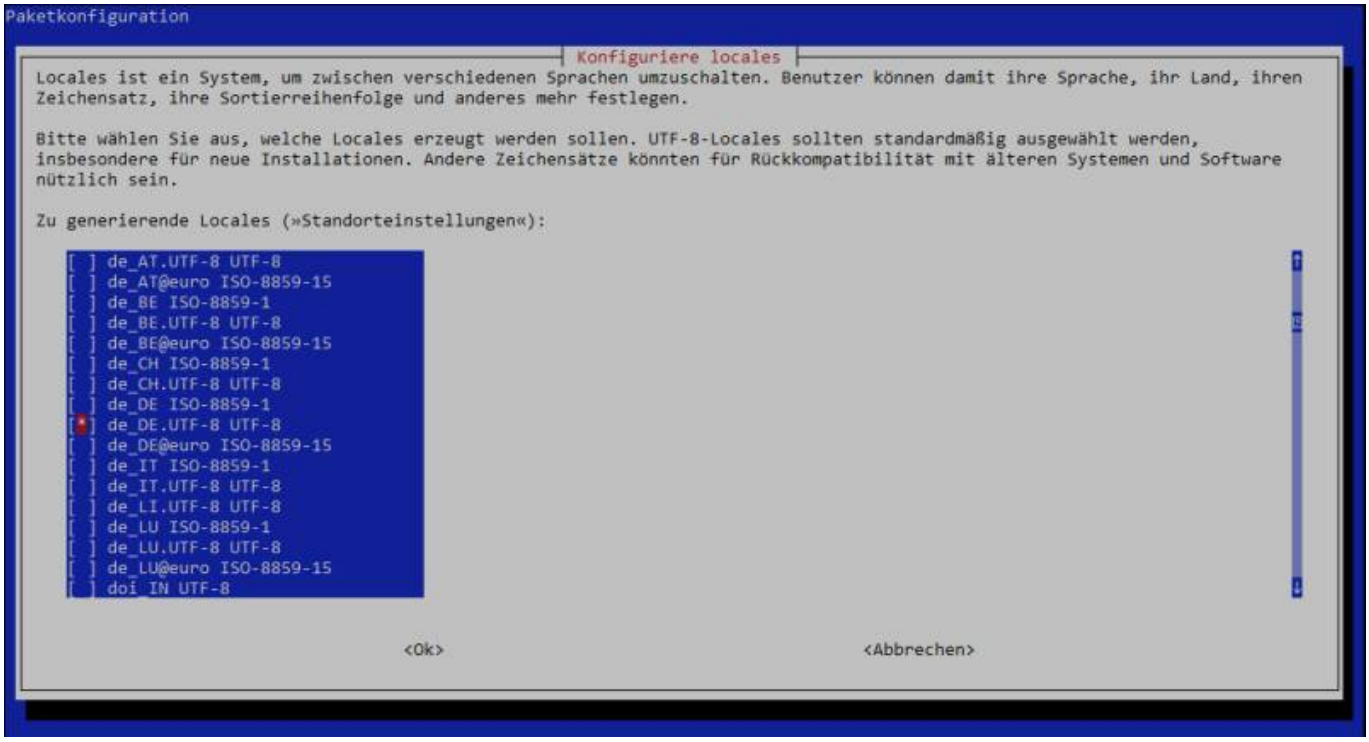
Diese Frage wird mir immer mal wieder gestellt, ich habe auch darüber nachgedacht. Aber die Antwort ist NEIN, gibt es nicht und wird es von mir auch nicht geben. Warum ? Um ein System mit SVXLINK zu betreiben, ist es zwingend erforderlich - so meine klare Ansicht - die Dinge zu verstehen, die man dafür benötigt. SVXLINK ist eine auf Software basierende Relaissteuerung für FM-Repeater und der Sysop muss in der Lage sein, dieses Werkzeug bedienen und konfigurieren zu lernen. Es ist und war niemals als Plug'n'Play-Lösung gedacht und kann es auch nicht sein, zu verschieden sind die Einsatzszenarien in Sachen Hardware & Co.

Kaufe ich mir einen neuen Kurzwellen-TRX muss ich auch lernen, wie ich diesen richtig bediene, das Setup richtig einstelle, damit ich die Ergebnisse erziele, wofür ich mir diesen TRX gekauft habe. Dieser Lernprozess *muss* auch bei Einsatz von SVXLINK erfolgen. Man kann kein Relais betreiben, dessen Funktionsweise der Sysop nicht im Griff hat. Da gibts für mich kein Wenn und Aber. Autofahren kann ich auch nur, wenn ich es erlernt habe und am Straßenverkehr kann ich erst teilnehmen, wenn ich meine Kenntnisse in einer Fahrprüfung nachgewiesen habe. Im übertragenen Sinne sind wir hier mit SVXLINK in der gleichen Ausgangsposition. Deswegen meine folgende Anleitung, wie man so ein System installiert und betriebsfertig bekommen kann.

1.2 Es geht los...

Ich empfehle für den Raspberry Pi das Raspian lite, was *ohne grafische Oberfläche (X11)* ausgestattet ist. Die X11-Umgebung benötigt ziemlich viel Resourcen und steigert erheblich den RAM-Verbrauch, außerdem wird meist Pulseaudio mitinstalliert, was später Konflikte mit SVXLINK verursachen könnte. Wir benötigen für SVXLINK keine grafische Systemumgebung !
Weiterhin gehe ich davon aus, das Ihr bereits den Zugriff auf den Pi per SSH aktiviert habt, um die folgenden Arbeiten remote ausführen zu können. Alternativ kann man es natürlich auch per Konsole, also Tastatur und angeschlossenem HDMI-Display, durchführen. Stellt bitte die *korrekte Zeitzone (Europe/Berlin)* ein und stellt die Systemumgebung auf *de_DE.UTF-8 UTF-8* unter Verwendung des mitgelieferten Tools *raspi-config* im Menüpunkt 5 Localisation Options » L1 Locale. Auch der SSH-Zugang, falls noch nicht konfiguriert, lässt sich dort aktivieren: 3 Interface Options » I2 SSH.

```
$ sudo raspi-config
```



Zuerst bringen wir mal unser System auf aktuellen Stand:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

In einigen Fällen kann noch das erforderlich sein, muss aber nicht. Wir definieren das mal als *OPTIONAL*:

```
$ sudo apt-get dist-upgrade
```

1.1 Linux/Raspian anpassen

SWAP deaktivieren:
 NUR wenn RAM >512MB (Raspberry Pi 2B/3B/3B+/4):

```
$ sudo swapoff -a
$ sudo service dphys-swapfile stop
$ sudo free
$ sudo systemctl disable dphys-swapfile
```

RAM-Disk anlegen für Logging SVXLINK:

```
$ sudo mkdir /var/log/dv
$ sudo chmod 777 /var/log/dv
```

/etc/fstab ergänzen:

```
tmpfs   /var/log/dv             tmpfs
nodev,noatime,nosuid,mode=0777,size=128m        0       0
```

Das erzeugt beim Systemstart eine RAM-Disk mit 128MB, die unter /var/log/dv eingehangen wird, wo wir später das Log vom SVXLINK ablegen werden, um die Schreibzugriffe auf die SDCard zu deren Schutz zu minimieren.

RAM-Disk mounten:

```
$ sudo mount -a
```

SYSTEMd Journaling anpassen in /etc/systemd/journald.conf :

```
#Storage=auto
Storage=volatile
...
```

Unnötige Dinge entfernen:

```
$ sudo systemctl disable ModemManager.service
$ sudo systemctl stop ModemManager.service
$ sudo apt-get purge modemmanager
$ sudo systemctl stop hciuart
$ sudo systemctl disable hciuart
```

Um Konflikte mit dem on-board-Audio, dem HDMI-Audio, Blutooth und USB zu vermeiden, passen wir das Verhalten des Kernels und dessen zu ladenen Modulen etwas an. Das ist kein Muss, aber zu empfehlen.
Es können aber, je nach eingesetzter Hardware (z.B. dem ELENATA-Board, F8ASB µSVXCard, SVXHat), *weitere Anpassungen oder Änderungen erforderlich sein*. Das ist der jeweiligen Doku der einzusetzenden Hardware zu entnehmen.
Die folgenden Änderungen sind aber immer zu emfehlen.

Anpassen der /boot/config.txt :

```
# Enable audio (loads snd_bcm2835)
# on-board Soundkarte deaktivieren
#dtparam=audio=on
# HDMI-Audio deaktivieren
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. Stromentnahme USB 1.2A aktivieren
max_usb_current=1
# GPIO-UART aktivieren falls erforderlich
enable_uart=1
# Blutooth deaktivieren falls vorhanden
dtoverlay=disable-bt
```

Neustart Raspberry Pi:

```
$ sudo reboot
```

1.2 SVXLINK Abhängigkeiten und nützliche Tools installieren

Dann installieren wir alle notwendigen Tools und Bibliotheken, die zwingend zur Compilierung des SVXLINK benötigt werden. Das muss auf dem jeweiligen Linux nur EINMAL erledigt werden:

```
$ sudo apt-get install build-essential git
$ sudo apt-get install libasound2-dev g++ gcc make cmake groff gzip doxygen
tar graphviz
$ sudo apt-get install libsigc++-dev libsigc++-2.0-dev
$ sudo apt-get install libspeex-dev libspeexdsp-dev libopus-dev libogg-dev
$ sudo apt-get install libpopt-dev
$ sudo apt-get install libasound2-dev libgcrypt20-dev libgsm1-dev
$ sudo apt-get install librtlsdr-dev libjsoncpp-dev
$ sudo apt-get install tcl-dev
$ sudo apt-get install libcurl4-openssl-dev

  sudo apt install g++ make libsigc++-2.0-dev
    sudo apt install libgsm1-dev libpopt-dev tcl-dev
    sudo apt install libgcrypt-dev libspeex-dev
    sudo apt install i2c-tools libi2c-dev
    sudo apt install libasound2-dev alsa-utils
    sudo apt install git cmake
    sudo apt install libjsoncpp-dev libopus-dev
    sudo apt install gpiod libgpiod-dev librtlsdr-dev alsa-utils vorbis-
tools curl libcurl4-openssl-dev rtl-sdr libjsoncpp-dev
```

```
$ sudo apt-get install gpiod libgpiod-dev sigc++
```

Weitere Emfehlungen zur Installation (optional):

```
$ sudo apt-get install raspberrypi-kernel-headers
$ sudo apt-get install dnsutils
$ sudo apt-get install mc
```

Wenn es bisher noch keinen User `svxlink` gab, müssen wir diesen anlegen und einige Gruppenzuweisungen machen. Gibt den User `svxlink` schon, dann die erste Befehlszeile weglassen (useradd), aber trotzdem die drei folgenden (usermod) ausführen. Auch das ist auf dem jeweiligen System nur EINMAL zu erledigen:

```
$ sudo useradd svxlink
$ sudo usermod -a -G gpio svxlink
$ sudo usermod -a -G audio svxlink
$ sudo usermod -a -G plugdev svxlink
```

Um zu prüfen, ob der User `svxlink` schon existiert und wenn ja, ob die Gruppen schon zugewiesen

wurden, macht man folgendes:

```
$ id svxlink
```

und prüft die Ausgabe des Befehls. Da kann man alles überprüfen.

2. Beziehen der Sourcen von Github und Compilierung

Zuerst prüfen wir mal, ob es das Verzeichnis `/usr/local/src` gibt. Sollte das nicht der Fall sein, legen wir es an, ansonsten überspringen wir diesen Schritt:

```
$ sudo mkdir /usr/local/src
```

Jetzt geht es weiter:

```
$ sudo chmod 777 /usr/local/src
$ cd /usr/local/src
$ sudo git clone https://github.com/sm0svx/svxlink.git
$ cd svxlink/src/
$ sudo mkdir build
$ cd build
$ sudo cmake -DCMAKE_INSTALL_PREFIX=/usr -DSYSCONF_INSTALL_DIR=/etc -
DLOCAL_STATE_DIR=/var -DUSE_QT=NO -DWITH_SYSTEMD=yes ..
$ sudo make
$ sudo make doc
$ sudo make install
$ sudo ldconfig
```

Sollte unter `/usr/local/src` schon ein Verzeichnis `svxlink` vorhanden sein, löschen wir es am besten vor dem Befehl `git clone` :

```
$ cd /usr/local/src
$ ls -l
drwxr-xr-x  9 root root 4096  9. Jan 18:52 svxlink
$ sudo rm -fr svxlink
```

Das installiert SVXLINK ins Linux, Konfiguration ist unter `/etc/svxlink` zu finden, Zusatzprogramme wie die TCL-Dateien unter `/usr/share/svxlink` . Definitionen wie die Pfadangabe für das Logfile sind in der Datei `/etc/default/svxlink` zu finden und können dort ggf. angepasst werden.

3. SVXLINK konfigurieren

Bevor wir SVXLINK überhaupt starten können, muss jetzt die Konfiguration der `/etc/svxlink/svxlink.conf` an Euer System angepasst werden. Hier **bitte die Dokumentation des SVXLINK studieren** und die korrekten Einstellungen vornehmen, je nachdem was eingerichtet werden soll (Repeater oder FM-simplex-System, ECHOLINK etc.). Eine Beispiel-Konfiguration kann es eigentlich nicht geben, zu verschieden dürfte die Hardware sein, speziell was Soundinterfaces und GPIO-Steuerungen für PTT und/oder SQL angeht. **Das müsst ihr dann den entsprechenden Hardware-Dokumentationen entnehmen und dementsprechend im Setup umsetzen** .

SVXLINK ist keine Plug'n'Play-Anwendung, sich damit ausgiebig zu beschäftigen ist ein Muss, das kann Euch niemand abnehmen, wenn ihr es alles selbst machen wollt !

3.1 Ansteuerung der GPIO-PINs - wichtige Hinweise

Viele Anleitungen bzgl. SVXLINK und GPIO verweisen auf ein Paket namens Wiring-Pi. Dieses Verfahren ist aber bereits seit längerer Zeit deprecated (dt. „abgekündigt“) und wird über kurz oder lang von den Kernel-Entwicklern entfernt werden. Wenn das passiert, funktionieren dann die Initialisierungs-Scripte nicht mehr. Es ist auch bereits nicht mehr Grund-Bestandteil neuerer Raspian-Versionen und muss bereits manuell nachinstalliert werden.
Das neuere, empfohlene Verfahren nennt sich GPIOD, was SVXLINK schon länger unterstützt. Ich kann nur dringend emfehlen, dieses neuere Verfahren anzuwenden und das alte, Wiring-Pi außen vor zu lassen. *Noch* aber kann man Wiring-Pi allerdings benutzen.
Für GPIOD sind folgende Anpassungen in der /etc/svxlink/svxlink.conf vorzunehmen:

```
[Rx1]
...
SQL_DET=GPIOD
SQL_GPIOD_CHIP=gpiochip0
# active-low Logik wird mit einem Ausrufezeichen verwendet es tauscht
praktisch die Auswertelogik auf GPIO-PIN 23 welche per default high-active
ist
SQL_GPIOD_LINE=!23
...

[Tx1]
...
PTT_TYPE=GPIOD
PTT_GPIOD_CHIP=gpiochip0
# PTT auf GPIO-PIN 13
PTT_GPIOD_LINE=13
...
```

Weitere Einträge bzw. Scripte zur Initialisierung der GPIO sind nicht mehr erforderlich, es reicht die Anpassung der /etc/svxlink/svxlink.conf !

4. SVXLINK für automatischen Start vorbereiten

4.1 SVXLINK Log-Pfad anpassen

Wir haben ja unter /var/log/dv eine RAM-Disk angelegt. Die verwenden wir jetzt, um die SVXLINK-Logdatei abzulegen.
Dazu ist in der /etc/default/svxlink folgende Änderung zu machen:

```
...
# Where to place the log file
# LOGFILE=/var/log/svxlink
# wir speichern das Log in unsere RAM-Disk
LOGFILE=/var/log/dv/svxlink
...
```

4.2 SVXLINK mit Zeitverzögerung starten

Ich empfehle, um Probleme mit gewissen Abläufen beim Systemstart zu vermeiden, den Autostart des SVXLINK, der standardmäßig mit dem systemd erfolgt, etwas zu verzögern. Ein guter Wert wären ca. 30sec.
Dazu bedienen wir uns ebenfalls bei den Möglichkeiten des systemd.

```
$ sudo nano /lib/systemd/system/svxlink.timer
```

und fügen in diese svxlink.timer folgendes ein:

```
[Unit]
Description=Startup delay SVXLINK

[Timer]
OnBootSec=30
Unit=svxlink.service

[Install]
WantedBy=multi-user.target
```

und speichern diese. Anschließend aktivieren wir diese, damit sie beim Systemstart aktiviert wird:

```
$ sudo systemctl enable svxlink.timer
```

Eine Aktivierung der eigentlichen `svxlink.service` darf bei Einsatz einer zugehörigen `svxlink.timer` nicht erfolgen ! Es wird nur die `svxlink.timer` aktiviert.
Wer mehr darüber wissen will, muss sich mit den Möglichkeiten des systemd beschäftigen und dessen Dokumentation lesen.
Um zu prüfen, ob SVXLINK korrekt gestartet wurde, sehen wir uns einfach mal dessen Log an:

```
$ sudo tail -n 500 -f /var/log/dv/svxlink
```

Das zeigt immer die letzten 500 Zeilen der Logdatei vom SVXLINK an, raus kommt man da immer mit STRG-C.

5. weitere Arbeiten

SVXLINK enthält per Default keine Sound-/Sprachdateien für die Sprachausgabe. Es gibt dt. Sprachpacks (u.a. hier zu finden), die aber teilweise mit lizenzpflichtigen Text-to-Speech-Applikationen erstellt wurden und damit nicht einfach frei zum Download bereitgestellt werden können bzw. deren Lizenzbedingungen zu beachten sind. Man könnte selber solche Sprachdateien erstellen mit Open-Source-Tools wie picotts oder man muss eben ggf. eine entsprechende Lizenz erwerben, um schon vorhandene Sprachpacks verwenden zu können. Hier müsst ihr leider selber auf die Suche gehen, wenn es benötigt wird.
Das Format dieser Sprachdateien ist .wav/11kHz/mono/16bit und müssen in einer Art Struktur nach `/usr/share/svxlink/sounds/de_DE/` kopiert werden, wobei `de_DE` der eingestellten Sprache in der `svxlink.conf` entspricht, denn SVXLINK ist multilingual:

```
.
├── Airport
├── AnnounceLogic
├── Core
├── Default
├── DtmfRepeater
├── EchoLink
├── Help
├── MetarInfo
├── Own
├── Parrot
├── PhoneLogic
├── PropagationMonitor
├── SelCallEnc
├── SipLogic
├── TclVoiceMail
├── TrafficInfo
└── WeatherInfo
```

Weiter gehe ich aber auf die Sprachunterstützung nicht ein.

Weiterhin gibt es unter `/usr/share/svxlink/events.d` sog. TCL-Scripts, mit deren Hilfe man Ereignis-basierend das SVXLINK anpassen kann. Ein Ereignis wäre z.B. SQL_OPEN, also wenn die Rauschsperre öffnet. Das es sich bei TCL („Tool command language“) um eine Programmier- bzw. Scriptsprache handelt, muss man also lernen, wie diese Sprache angewendet wird, um eigene Anpassungen vornehmen zu können. Das kann an dieser Stelle hier leider nicht erfolgen, da es grundlegende Programmierkenntnisse und -fähigkeiten voraussetzt.

6. Update SVXLINK aus den Github-Sourcen

In gewissen Abständen aktualisiert der Programmautor vom SVXLINK, Tobias/SM0SVX, den Programmcode. Wie wir ein Update machen können, beschreibe ich hier. Dazu bedienen wir uns dem Befehl `git pull` , der unser lokales Quellcode-Verzeichnis auf den Stand von Github bringt. Voraussetzung ist also, das unter `/usr/local/src` ein Verzeichnis `svxlink` existiert, was früher einmal mittels `git clone` erzeugt wurde.

```
$ sudo systemctl stop svxlink.service svxlink.timer
$ cd /usr/local/src/svxlink
$ sudo git pull
$ cd src
$ sudo rm -fr build
$ sudo mkdir build
$ cd build
$ sudo cmake -DCMAKE_INSTALL_PREFIX=/usr -DSYSCONF_INSTALL_DIR=/etc -
DLOCAL_STATE_DIR=/var -DUSE_QT=NO -DWITH_SYSTEMD=yes ..
$ sudo make
$ sudo make doc
$ sudo make install
$ sudo ldconfig
$ sudo systemctl daemon-reload
$ sudo systemctl start svxlink.service
```

 HINWEIS: Wurde innerhalb von `/usr/local/src/svxlink` irgendeine Datei verändert, schlägt `git pull` fehl. Das sog. lokale Repository weicht dann von dem auf Github ab und kann aufgrund lokaler Änderungen nicht mehr so einfach geupdated werden. Der Abgleichmechanismus ist damit praktisch beschädigt. In diesem Fall löschen wir also das `/usr/local/src/svxlink` einfach komplett:

```
$ sudo rm -fr /usr/local/src/svxlink
```

und beginnen wieder bei Pkt. 2 !

 Bei einem erneuten `sudo make install` bleiben die Daten unter `/etc/svxlink/` erhalten, die `svxlink.conf` und andere Daten werden also nicht überschrieben. Anders sieht das bei den TCL-Scripts unter `/usr/share/svxlink/events.d` aus, wenn man dort in den Original-Dateien Veränderungen vorgenommen hat, werden diese überschrieben. Um das zu vermeiden, ist unter `/usr/share/svxlink/events.d` ein weiteres Verzeichnis `local` anzulegen und die geänderten TCL-Scripte dort hinein zu kopieren. Dieses Verzeichnis wird nicht überschrieben.

---

73 Heiko, DL1BZ

zurück zur Startseite