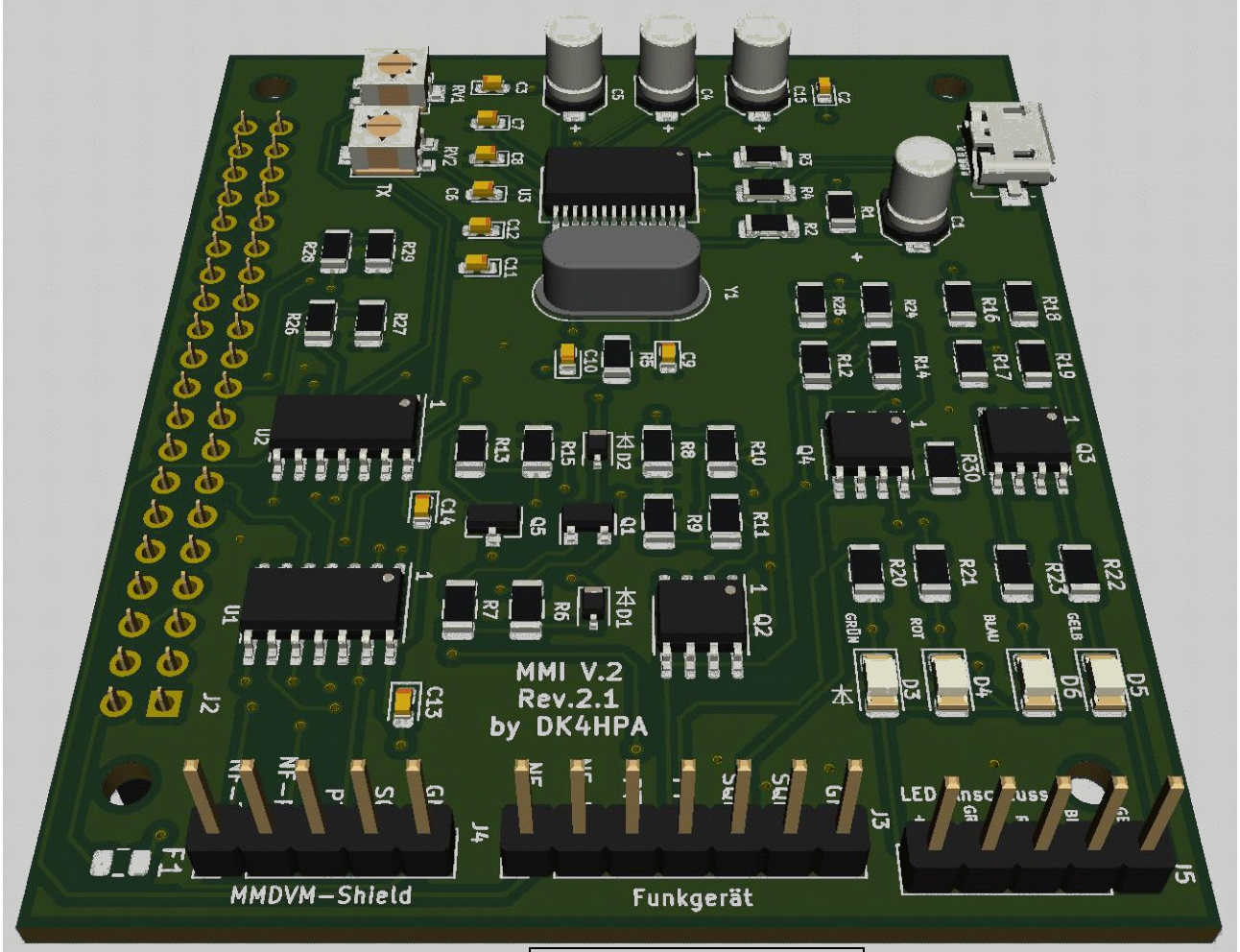
Multi-Mode-Interface

SMD-Version



MMDVM

D-STAR DMR System Fusion

SvxLink

Beschreibung der Hard- und Software

Hardware Vers.2 Image Vers. 1.0

Inhaltsverzeichnis

Seite

1 Welche Harware benötigen wir

Einen Raspberry Pi 2 oder 3.

Eine MMI-Platine für den Raspi (Mutlimode-Interface).

Die SvxLink Software für den analogen Teil.

Einen Arduino DUE mit MMDVM-Shield oder einen STM32 für den DV-Betrieb.

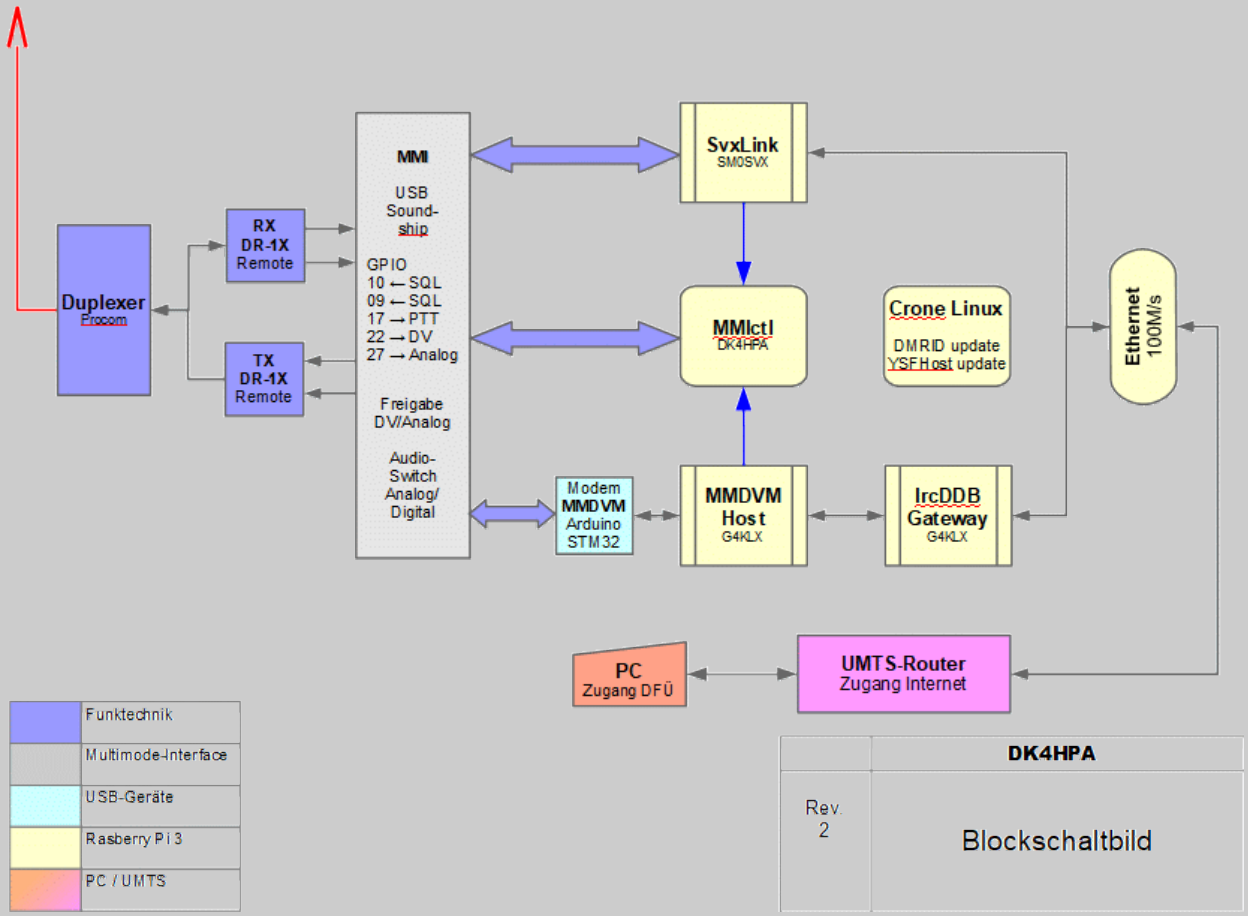
Software für den Arduino oder STM32.

MMDVMHost Software für den Raspi.

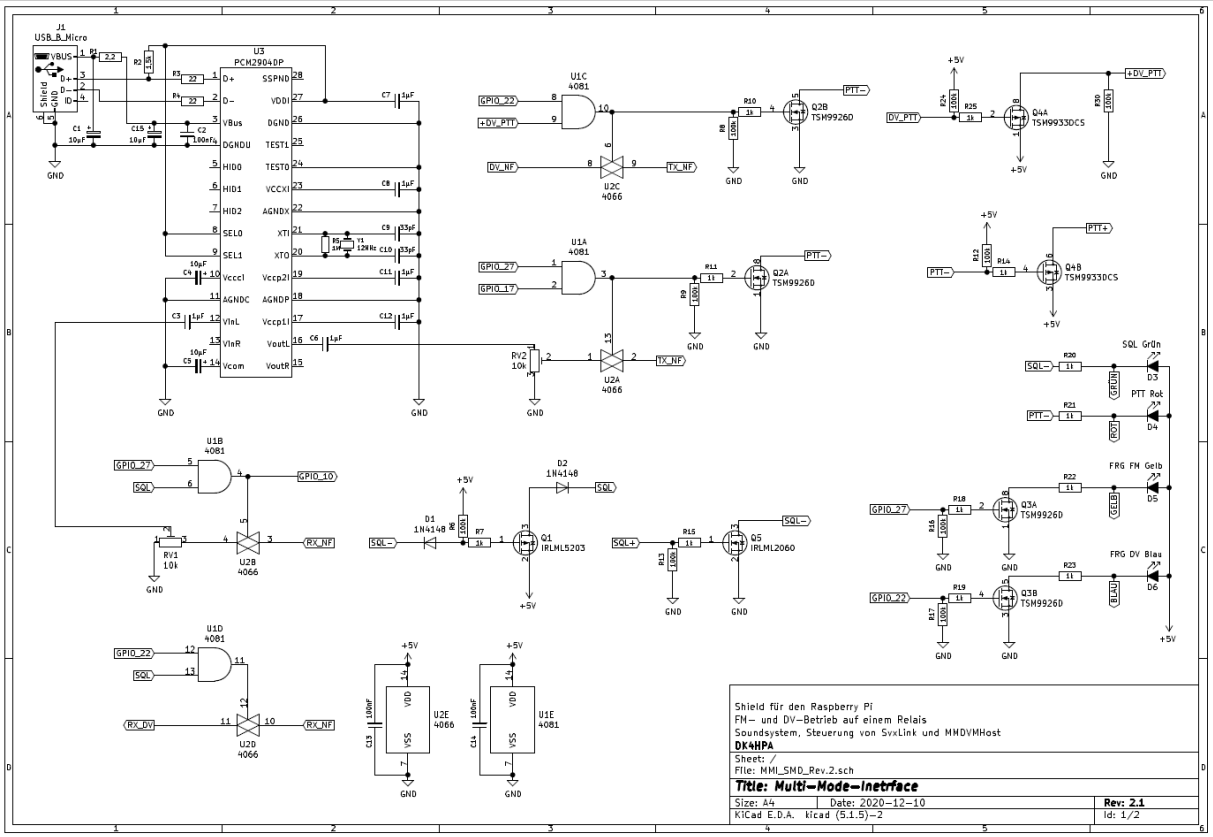
Linux-Editor mc (Midnight-Commander) oder nano.

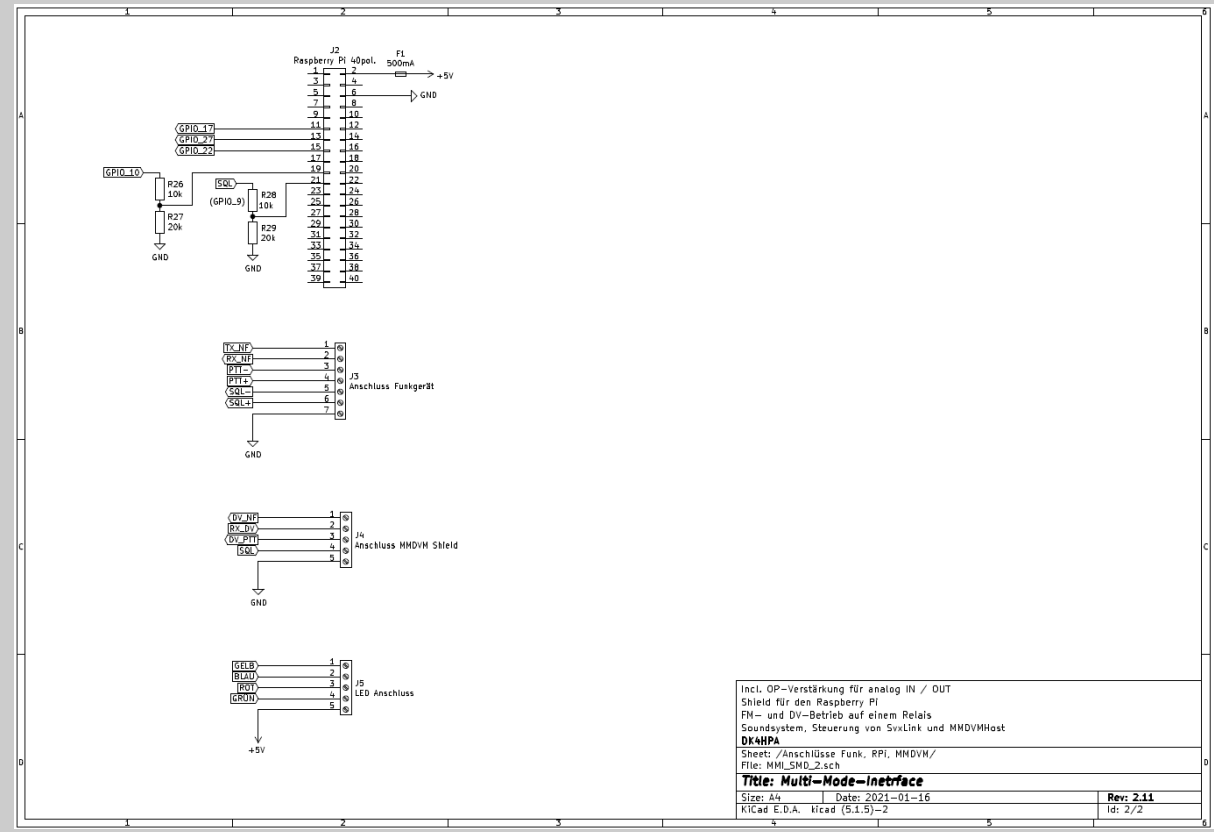
Ausserdem müssen ein paar Dateien angelegt werden um eine effektive Steuerung zu schaffen. In der SvxLink-Software müssen zwei TCL-Scripte bearbeitet werden. Natürlich müssen auch die INI-Dateien angepasst werden. Hier muss darauf geachtet werden, wo das MMDVMHost Log abgelegt wird.

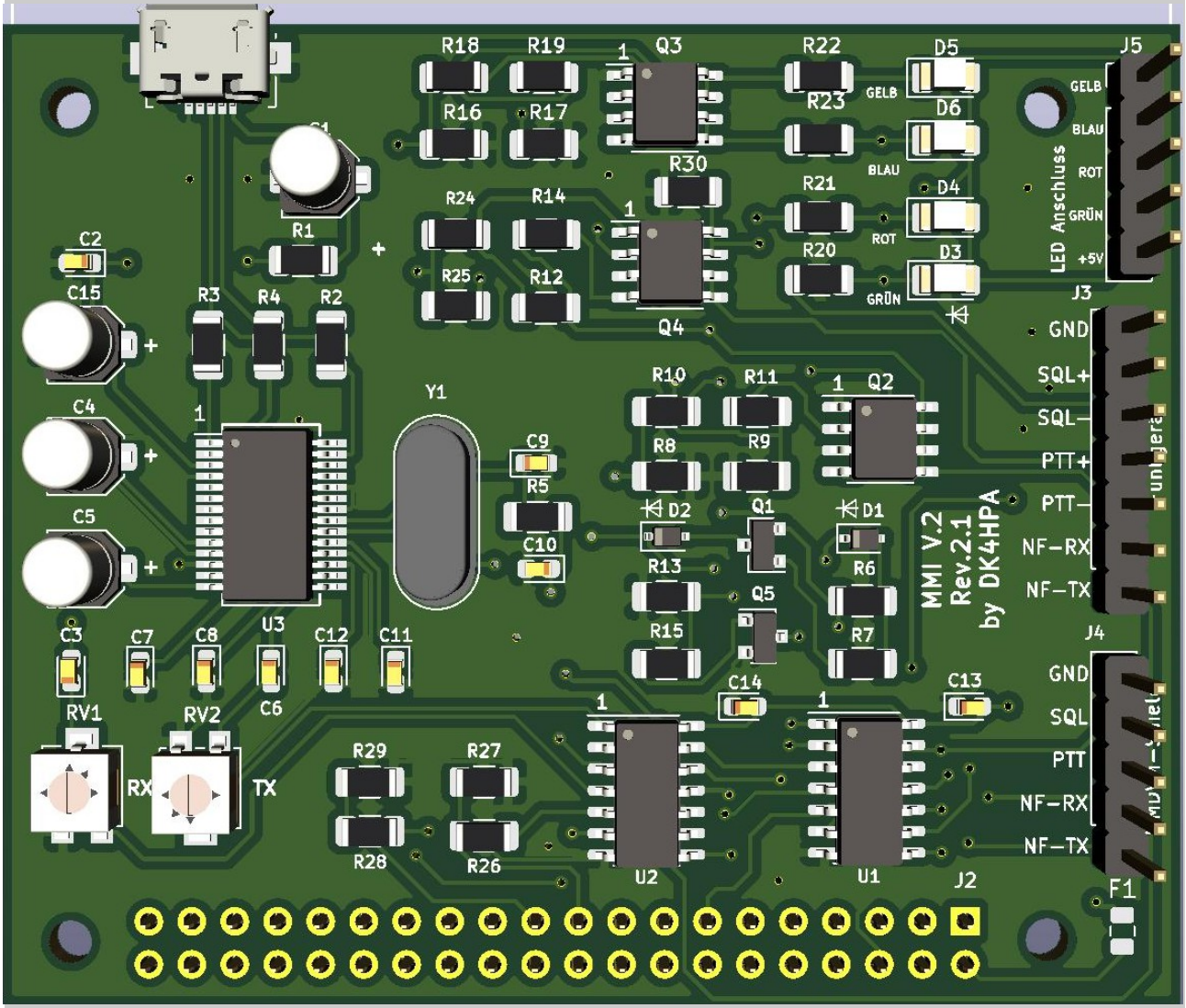
Noch ein kleiner Hinweis, das Ganze Projekt ist nur als Vorschlag zu verstehen. Eine kommerzielle Nutzung der Schaltung bzw. Veröffentlichung ist ohne meine Zustimmung jedoch ausdrücklich untersagt.



2 Das Multi-Mode Interface







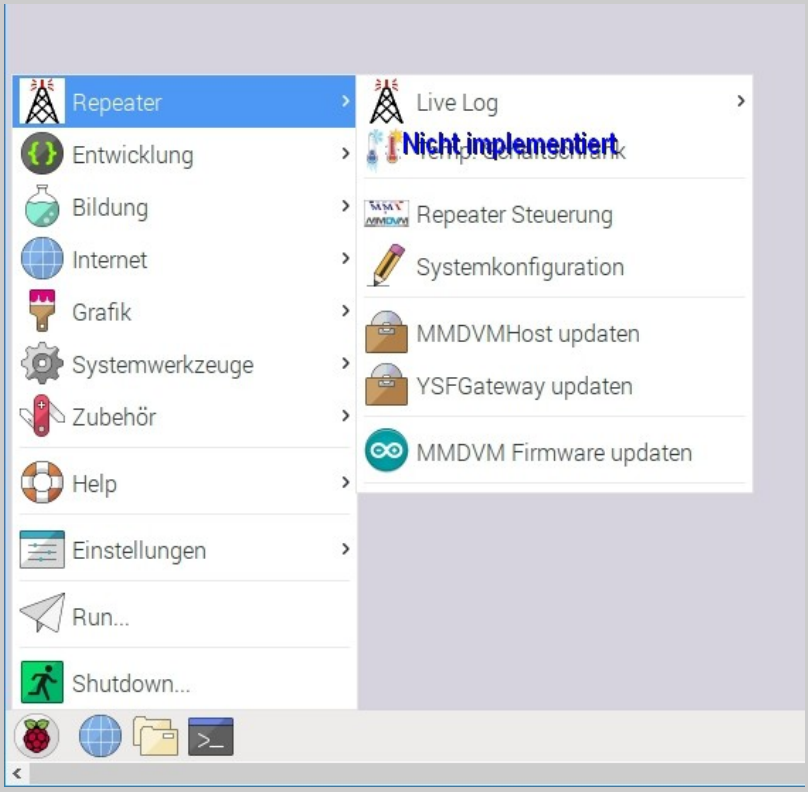
Der Soundship U3 stellt den AD/DA Wandler für die SvxLink Software via USB-Anschluss zur Verfügung. Somit besteht weiterhin die Möglichkeit den I2C-Bus für andere Zwecke (STM32 Modem) zu verwenden.
Der Audioswitch U2 schaltet die NF für RX und TX entsprechend der Freigaben zum MMDVM Modem oder zum Soundship durch. Somit werden bei einer Anforderung durch den Repeater-User die NF Wege entkoppelt.
Signalisiert werden SQL (grün), PTT (rot), Freigabe Analog (gelb) und Freigabe DV (blau).
Der Anschluss des MMDVM Modem erfolgt an J4 mit den NF-, PTT- und SQL-Signal.
Der Anschluss für das Funkgerät J3 bietet, ausser den NF-RX und NF-TX, die Auswahl der Polarität von den Signalen SQL und PTT. Da es bei den verwendeten Funkgeräten hier doch Unterschiede gibt, wurde dem hier Rechnung getragen.
Die Signale der LED's können via J5 incl. +5V nach aussen verlegt werden. Um den Raspi nicht zu überlasten ist der Vorwiderstand so dimensoniert das 5mA über die LED's fließen.

3 Aufgabe des MMIctl Programms

Das in Python geschriebene und kompelierte Programm hat die Ausgabe bei einem SQL-Signal zu ermitteln welche Modulation der Benutzer vom Relais anfordert. Wird hier eine DV-Anforderung festgestellt, so wird die analoge Freigabe entzogen und die NF-Signale nur für den MMDVM Modemzweig durch gereicht. Bei einem analogen Signal wird diese Aufgabe durch die entsprechenden modifizierten TCL-Dateien übernommen. Zustätzlich überwacht das MMIctl die laufenden Programme SvxLink und MMDVMHost (Watchdogfunktion).
Der Zeitpunk ab wann der Repeater wieder in den IDL Mode zurück fällt wir in der MMIctl.ini eingestellt. Zusätzlich kann die Anzahl von Logzeilen zum feststellen ob ein DV-Betrieb erfolgt angepasst werden. Mit dem Aufruf Systemkonfiguration, siehe weiter unten, werden alle nötigen ini Dateien bereit gestellt.

Ein kleines "Helferlein" unterstürtzt den Sysop im kleinen Umfang.
Das repeaterctl steht auf der grafischen Oberfläche zur Verfügung.
Zusätzlich ist im Menüe eine Möglichkeit für diverse Aufgaben vorhanden.
So z.B. Aufruf von Online Logs, bearbeiten der ini Dateien und Update Scripte. Bei den Update Scripten ist vorsicht geboten, da es eventuell nötig macht einige weitere Programme einem Update zu unterziehen.

Ein Quelltext für MMIctl und repeaterctl kann vom Verfasser angefordert werden.



4 Anpassungen der SvxLink TCL Scipte

Die modifizierten Dateien liegen im Verzeichnis
/usr/share/svxlink/event.d/local
Damit wird sicher gesellt das bei Programmupdates keine Änderungen in den modifizierten TCL Dateien vorgenommen wird.

4.1 Logic.tcl

```
#
# Executed when a short identification should be sent
#   hour    - The hour on which this identification occur
#   minute  - The hour on which this identification occur
#
#    .....D-Star Modifizierung.....

proc send_short_ident {{hour -1} {minute -1}} {
  global mycall;
  variable CFG_TYPE;

# Abfrage ob DV-Betrieb via Netz vorhanden?

    set datum [clock format [clock seconds] -format "-%Y-%m-%d.log"];
    set zeit [clock seconds];
    set a [file mtime "/media/DVrepeater/MMDVM$datum"];
    set b 0;
    set b [expr {$zeit-$a}];
    if {$b < "60"} {
      puts "DV-Betrieb / keine Bake";
      return;
    }
    set no [ exec cat /sys/class/gpio/gpio27/value];
    if { $no == 1 } { exec sudo echo "0" > /sys/class/gpio/gpio22/value &; }
```

```
#
# Executed when a long identification (e.g. hourly) should be sent
#   hour    - The hour on which this identification occur
#   minute  - The hour on which this identification occur
#
proc send_long_ident {hour minute} {
  global mycall;
  global loaded_modules;
  global active_module;
  variable CFG_TYPE;
  variable zufall;

# Abfrage ob DV-Betrieb via Netz vorhanden?

    set datum [clock format [clock seconds] -format "-%Y-%m-%d.log"];
    set zeit [clock seconds];
    set a [file mtime "/media/DVrepeater/MMDVM$datum"];
    set b 0;
    set b [expr {$zeit-$a}];
    if {$b < "59"} {
      puts "DV-Betrieb / keine Bake";
      return;
    }
    set no [ exec cat /sys/class/gpio/gpio27/value];
    if { $no == 1 } { exec sudo echo "0" > /sys/class/gpio/gpio22/value &; }
```

```
#
# Executed each time the transmitter is turned on or off
#   is_on - Set to 1 if the transmitter is on or 0 if it's off
#
proc transmit {is_on} {
#  puts "Turning the transmitter $is_on";
  global dvmute
  variable prev_ident;
  variable need_ident;
  if {$is_on && ([clock seconds] - $prev_ident > 5)} {
    set need_ident 1;
  }
# Nach Bake DV-Freigabe geben
  if { $is_on == 0 && $dvmute == 0 } { exec sudo echo "1" > /sys/class/gpio/gpio22/value &; }
}
```

```
#
# Executed each time the squelch is opened or closed
#   rx_id   - The ID of the RX that the squelch opened/closed on
#   is_open - Set to 1 if the squelch is open or 0 if it's closed
#
#   -----D-Star Modifikation-----
#
proc squelch_open {rx_id is_open} {
 variable sql_rx_id;
 global sql_time_div;
 global dvdowntime;
 variable akttime;

 set akttime [clock seconds];

 puts "The squelch is $is_open on RX $rx_id";
 if {$is_open } {
     set sql_time_div $akttime;
 }
 if {$is_open == 0 && ([clock seconds] - $sql_time_div > 30)} {
   set dvdowntime [expr [clock second]+60];
 }
  set sql_rx_id $rx_id;
}
```

```
#
# Executed once every whole minute. Don't put any code here directly
# Create a new function and add it to the timer tick subscriber list
# by using the function addTimerTickSubscriber.
#
# -------D-Star-Modifikation-------
#
proc every_minute {} {
 variable timer_tick_subscribers;
 global dvmute;
 global dvdowntime;
 global tx_on;
 variable akttime;

 set akttime [clock seconds];

 if {$dvmute == 1 } {
   if {$tx_on == 1 } {
     set dvdowntime [expr [clock second]+60];
   } else {
      if {[clock second] > $dvdowntime } {
       set dvmute 0;
       exec sudo echo "1" > /sys/class/gpio/gpio22/value &;
       puts "Freigabe MMDVMHost Time $akttime OffTime $dvdowntime";
      }
     }
 }

 foreach subscriber $timer_tick_subscribers {
    $subscriber;
 }
}
```

4.2 Repeaterlogic

```
#
# Executed when the SvxLink software is started
#
# -------D-Star Modifikation--------
#
proc startup {} {
  global logic_name;
  #
  # D-Star_Modifikation
  #
  global dvdowntime;
  global dvmute;
  global tx_on;
  global counter;
  set dvdowntime 0;
  set dvmute 0;
  set tx_on 0;
  set counter 0;
#  puts "DVmute-$dvmute - DVtime-$dvdowntime - TX-$tx_on"
  # Ende Modifikation
  #
  append func $logic_name "::checkPeriodicIdentify";
  Logic::addTimerTickSubscriber $func;
  Logic::startup;
}
```

```
#
# Executed each time the transmitter is turned on or off
#
# D-Star-Modifikation
#
proc transmit {is_on} {

# DV-Betrieb Abfrage auf FM Freigabe

  set no [exec cat /sys/class/gpio/gpio27/value];
  if { $no == 0 } { puts "Keine Freigabe ^ DV-Betriieb"; return; }

  global tx_on;
  set tx_on $is_on;

  Logic::transmit $is_on;
}
```

```
#
# Executed when the repeater is activated
#   reason  - The reason why the repeater was activated
#               SQL_CLOSE               - Open on squelch, close flank
#               SQL_OPEN      - Open on squelch, open flank
#               CTCSS_CLOSE     - Open on CTCSS, squelch close flank
#               CTCSS_OPEN      - Open on CTCSS, squelch open flank
#               TONE            - Open on tone burst (always on squelch close)
#               DTMF            - Open on DTMF digit (always on squelch close)
#               MODULE          - Open on module activation
#               AUDIO           - Open on incoming audio (module or logic linking)
#               SQL_RPT_REOPEN  - Reopen on squelch after repeater down
#
#               -----D-Star-Modifikation------
#
proc repeater_up {reason} {
 global mycall;
 global active_module;
 variable repeater_is_up;
 variable uptime;
 global dvmute;
 global downtime;

 set repeater_is_up 1;
 set uptime [clock seconds];
 set downtime $uptime;

 if { $dvmute == 0 } {
   set dvmute 1;
   set downtime [expr $uptime+60];
    exec sudo echo "0" > /sys/class/gpio/gpio22/value &;
    puts "Freigabe Stop MMDVMHost $uptime $downtime";
 }
```

5. Start der GPIO's

Da bei jedem Neusstart des Raspberry Pi die Einstellungen für die GPIO's verloren gehen, ist es nötig ein kleines Script ausführen zu lassen.

Aufgabe dieser GPIO's:

- GPIO 10 → SQL Signal für SvxLink
- GPIO 15 → PTT Signal von SvxLink
- GPIO 22 → Freigabe Signal DV von SvxLink
- GPIO 27 → Freigabe Signal Analog von MMIctl

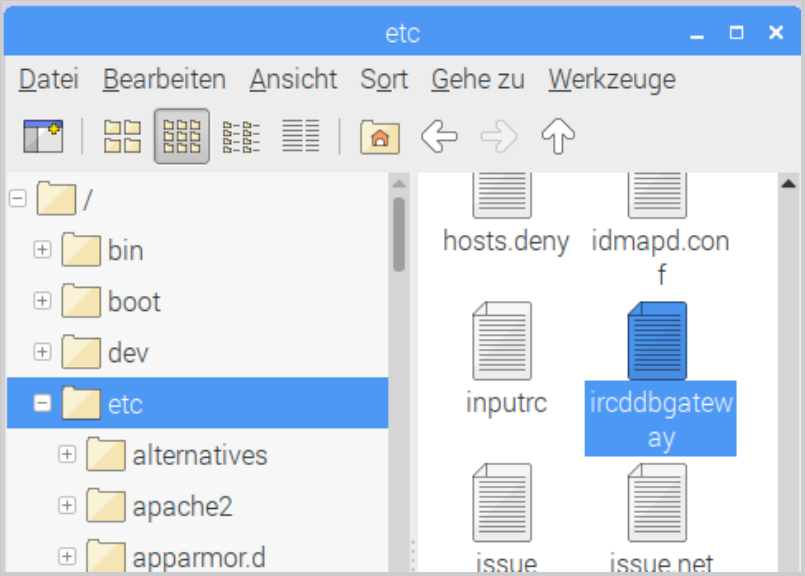
- GPIO 09 → SQL Signal für MMIctl (Wird mit dem MMIctl initialisiert)

```
#!/bin/bash

# GPIO anlegen für SvxLink

echo "10" > /sys/class/gpio/export
sudo chmod o+rw /sys/class/gpio/gpio10/direction
sudo chmod o+rw /sys/class/gpio/gpio10/active_low
sudo chmod o+rw /sys/class/gpio/gpio10/value
echo "in" > /sys/class/gpio/gpio10/direction
echo "0" /sys/class/gpio/gpio10/active_high

echo "17" > /sys/class/gpio/export
sudo chmod o+rw /sys/class/gpio/gpio17/direction
sudo chmod o+rw /sys/class/gpio/gpio17/value
echo "out" > /sys/class/gpio/gpio17/direction

echo "22" > /sys/class/gpio/export
sudo chmod o+rw /sys/class/gpio/gpio22/direction
sudo chmod o+rw /sys/class/gpio/gpio22/value
echo "out" > /sys/class/gpio/gpio22/direction
echo "1" > /sys/class/gpio/gpio22/value

echo "27" > /sys/class/gpio/export
sudo chmod o+rw /sys/class/gpio/gpio27/direction
sudo chmod o+rw /sys/class/gpio/gpio27/value
echo "out" > /sys/class/gpio/gpio27/direction
echo "1" > /sys/class/gpio/gpio27/value

sleep 30
```

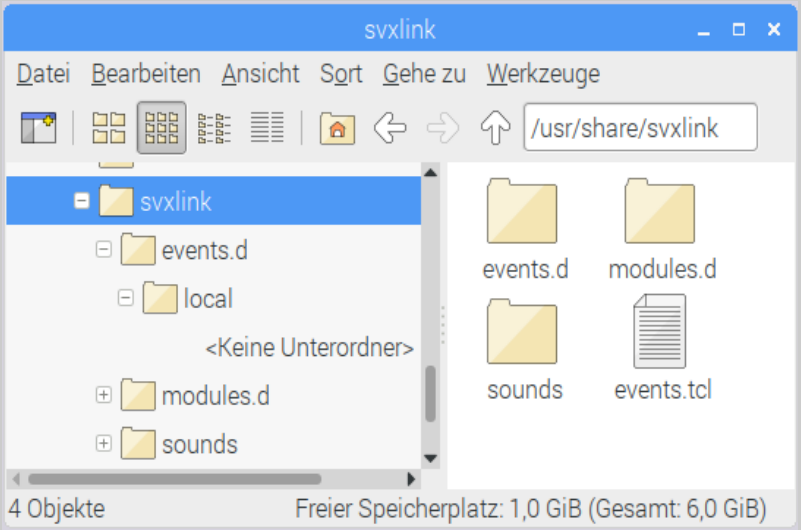
6 Die Verzeichnisstruktur

Da es sich für die Steuerung des Repeater um optionale Programme handelt, sind die Komponenten bis auf ein paar Ausnahmen in den opt Pfaden zu finden.

Hier vorrangestellt die Ausnahmen:

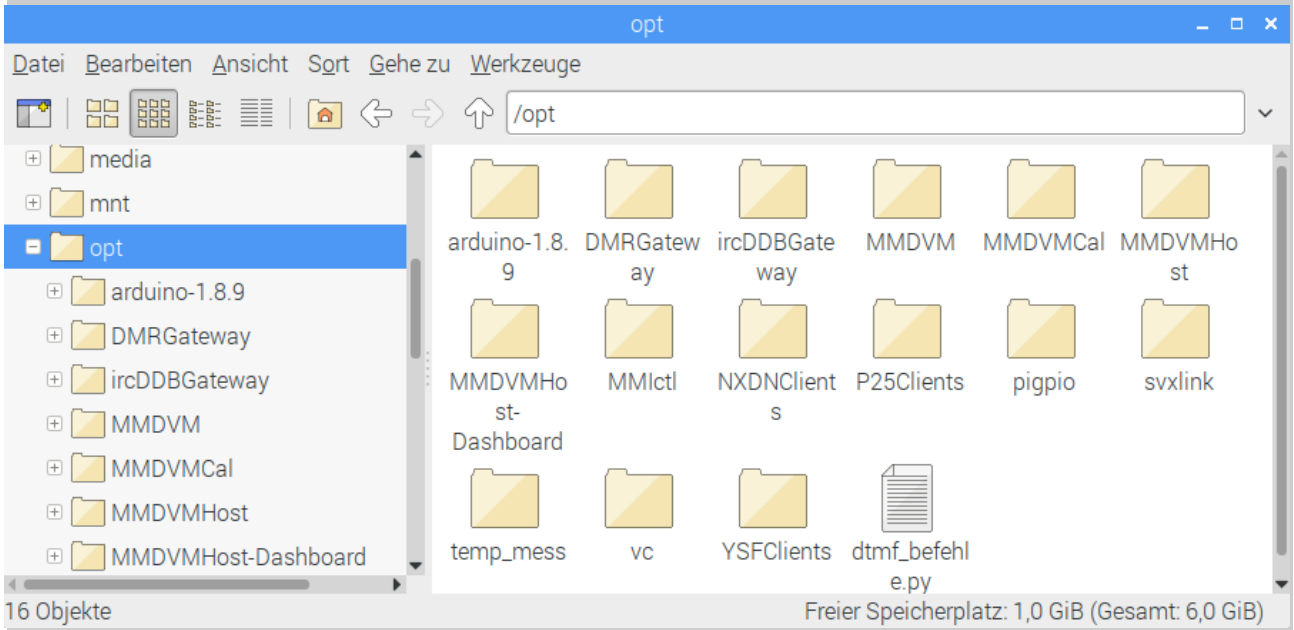


Hier ist die Initialisierungs Datei für das ircDDBGateway zu finden.

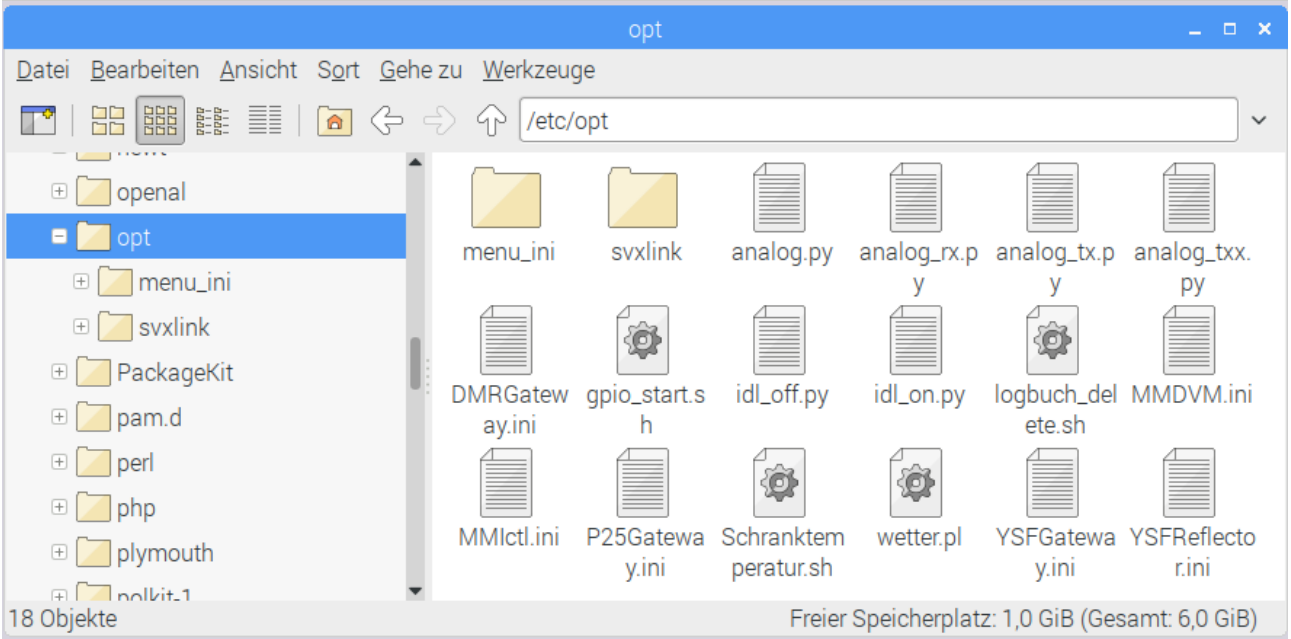


Hier sind alle wichtigen Files für SvxLink untergebracht. Im /local Pfad verbergen sich die modifizierten tcl Scripte.

Hier nun die optionalen Pfade:



Hier sind die mit git clone geladenen Systeme untergebracht. Bei einem Update werden die Inhalte überschrieben und müssen im Anschluss kompeliert werden mit den Befehlen make u.s.w. Dies habe ich in den Updatescripten im Menü Repeater hinterlegt.



Hier sind alle *.ini Files zu finden incl. der ausführbaren Shell-Scripte.