Inhaltsverzeichnis

Einrichten des VDR (2017) 3
Ubuntu Server 3
Installieren des VDR 3
DVB Karten 3
Technotrend USB DVB-S 3
DVBSky S952 V3 3
Kanalliste erzeugen 4
Konfiguration 5
Einrichten von WOL 5
vdradmin-am 6
vdr.service 7
Starten der installierten Minimal-Installation 7
Konfiguration VNSI Server 8
Debian - ACPI Wakeup mit sysvinit oder systemd 8
Kodi einrichten 11
Kodi Switch- Off einrichten 11
Koki Wake-on-Lan einrichten 13
Autostart 14
Fernbedienungs- Keymap 15
Weitere Einstellungen 16
Backup & Restore 16
Backup 16
Restore 17

Einrichten des VDR (2017)

Ubuntu Server

16.04 LTS Server installieren, sonst funktioniert die DVBSky Karte nicht. Unter 14.04 läßt sich der Kerneltreiber nicht compilieren, in 16.04 ist der Treiber schon enthalten.

Bei der Installation openSSH als einziges zusätzliches Packet auswählen

Installieren des VDR

(Quelle: http://wiki.ubuntuusers.de/VDR)

```
sudo apt-get install vdr vdradmin-am vdr-plugin-streamdev-server w-scan
build-essential  vdr-plugin-epgsearch vdr-plugin-vnsiserver util-linux pm-
utils libcgi-pm-perl
```

DVB Karten

Technotrend USB DVB-S

Wenn der Befehl „dmesg“ nach dem Einstecken der USB-Box etwas wie

```
"dvb-usb: found a 'Technotrend TT-connect S-2400' in *cold *state."
```

enthält, so muss noch die Firmware kopiert werden.

Die Firmware muss unter /lib/firmware/dvb-usb-tt-s2400-01.fw (auch bei > 64bit-Systemen!) abgelegt werden (dazu sind root-rechte nötig)

DVBSky S952 V3

https://linuxtv.org/wiki/index.php/DVBSky

Download der Karten Firmware

dvb-demod-m88rs6000.fw und kopieren nach `/lib/firmware` (siehe ansonsten auch linuxtv) (Lokale Firmware-Kopie)

Danach ein Reboot, und dann sollten eigentlich zwei Karten da sein

```
ls /dev/dvb*
```

Kanalliste erzeugen

Astra Scannen

```
sudo w_scan -fs -sS19E2 >> /etc/vdr/channels.conf.scan
```

dann die Liste mit einem Editor zusammenkürzen und sicherheitshalber auch als Kopie ablegen (channels.config.edited) und das Resultat als channels.conf speichern

Eine bereits zubereitete channels.conf kann man mit diesem Script und dem darin benutzten Python-Script aktualisieren, ohne wieder ganz von vorne anfangen zu müssen:

channelsUpdate.sh

```
systemctl stop vdr.service

cp channels.conf channels.conf.backup

rc=$?; if [[ $rc != 0 ]]; then echo "Error while copying
channels.conf.backup!" ; exit $rc; fi
echo "channels.conf copied to channels.conf.backup"

sudo w_scan -fs -sS19E2 >channels.conf.scan
rc=$?; if [[ $rc != 0 ]]; then echo "Error while scanning channels!" ;
exit $rc; fi

python3 updateChannels.py channels.conf.backup channels.conf.scan >
channels.conf
rc=$?; if [[ $rc != 0 ]]; then echo "Error while updating
channels.conf!" ; exit $rc; fi

echo "Do not forget to check if new channels are ok!"
echo "If not, restore from channels.conf.backup"
systemctl start vdr.service
```

updateChannels.py

```
#!/usr/bin/python3

import sys

oldChannels={}
oldChannelsData={}
index=0
#first read the existing channeldata into dictionary
for line in open(sys.argv[1]):
```

```
        channeldata =line.split(":",1)[0].strip()
        if len(channeldata)>0:
                oldChannels[channeldata]=index
                oldChannelsData[index]=line.strip()
                index+=1
#then read the new file
for line in open(sys.argv[2]):
        channeldata =line.split(":",1)[0].strip()
        if len(channeldata)>0 and channeldata in oldChannels: # when
the channel
                oldChannelsData[oldChannels[channeldata]]=line.strip()
# update the channeldata o

for counter in range(0,index): # dump the updated list
        print(oldChannelsData[counter])
```

Konfiguration

Verhindern, dass der VDR selber nach neuen Kanälen sucht, sowie automatisches Abschalten nach 10 Minuten

In `/var/lib/vdr/setup.conf`

```
UpdateChannels = 3
MinUserInactivity = 10
```

Einrichten von WOL

(Kopiert von http://askubuntu.com/a/764159)

In Ubuntu 16.04 set `WOL_DISABLE=N` in `/etc/default/tlp` to avoid getting WOL disabled by TLP power management.

Add `NETDOWN=no` in `/etc/default/halt` to prevent powering off the network card during shutdown

Enable Wake on LAN in `/etc/network/interfaces` when static network configuration is used:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
# The loopback network interface

auto lo
iface lo inet loopback
# The primary network interface

  auto eth0
```

```
  iface eth0 inet static
  address 192.168.1.6
  netmask 255.255.255.0
  gateway 192.168.1.1
  dns-nameservers 192.168.1.1
  up ethtool -s eth0 wol g
```

Enable wake on lan in BIOS, enter the BIOS setup and look for something called „Wake up on PCI event“, „Wake up on LAN“ or similar. Change it so that it is enabled. Save your settings and reboot.

https://help.ubuntu.com/community/WakeOnLan

Warning some motherboards / network controllers don't support WOL from the cold boot (S5 state, where the power to the system is physically turned off and back on again). In that case, at least one power cycle (power up, shutdown) has to be performed. To mitigate to the problem, the BIOS can be configured to power up when AC is restored and schedule a shutdown inside Ubuntu afterwards. Refer to the motherboard's manual for further details.

vdradmin-am

Zuerst einmal sind in 16.04 wohl die Packet-Abhängigkeiten nicht richtig gesetzt, und so fehlt dem vdradmin die Bibliothek `CGI.pm` aus dem Paket `libcgi-pm-perl` , was mit der altbekannten Technik installiert wird:

```
sudo apt-get install libcgi-pm-perl
```

Dann setzt man in `/etc/default/vdradmin-am`

```
ENABLED="1"
```

Benutzt man das Webinterface vdradmin-am, so muss die Datei `/var/lib/vdradmin-am/vdradmind.conf` angepasst werden:

```
PASSWORD = passwort
USERNAME = benutzer
```

Hier werden Benutzer und Passwort für die Web-Oberfläche festgelegt. Man sollte sie auf jeden Fall ändern.

Die Weboberfläche erreicht man dann im Browser via:

```
http://localhost:8001/
```

Irgendwie (mit user root beim ersten Ausprobieren?) werden manchmal temporäre Verzeichnisse des vdradmind mit dem Owner root angelegt und können dann vom regulären Daemon im Normalbetrieb nicht mehr genutzt werden, was dann die absolut kryptische Fehlermeldung

```
Not a SCALAR reference at /usr/share/perl/5.22/Compress/Zlib.pm
```

im Browser erzeugt.

Doch in einem Forum fand sich die rettende Lösung:

```
cd /var/cache/vdradmin-am
chown -R vdradmin-am:vdradmin-am usr
```

vdr.service

Einstellungen in `/etc/vdr/conf.d/00-vdr.conf`

```
  --video=/media/video
  # bei dieser Zeile die Auskommentierung **entfernen**
  --shutdown=/usr/lib/vdr/vdr-shutdown.wrapper
  # Lirc auskommentieren
  ##--lirc
```

Hier noch ein Hinweis [1)] für alle, die z.B. Hörspiele aufnehmen wollen, wo dann der Dateiname der Aufnahme 40 Zeichen überschreiten kann: Dies ist in der Standardkonfiguration von linVDR nicht möglich, weil dort sicherheitshalber das alte Windows- Dateisystem eingestellt ist, welches nicht mehr als 40 Zeichen kann.

Auf Linux und andereren modernen Betriebssystem kann man diese 40-Zeichen- Beschränkung abschalten, indem man in der `/etc/vdr/conf.d/00-vdr.conf` die Option

```
--vfat
```

auskommentiert wird. Natürlich nur, wenn man kein FAT32 für das Aufnahmeverzeichnis verwendet. Das dürfte aber bei den wenigsten der Fall sein.

Starten der installierten Minimal-Installation

```
systemctl start vdr.service
```

Der Zugriff auf den Server wird durch die streamdevhosts.conf konfiguriert:

```
/etc/vdr/plugins/streamdevhosts.conf
```

```
127.0.0.1             # always accept localhost
192.168.1.0/24        # any host on the local net
```

Den Stream-Server erreicht man mit dem Player der Wahl folgendermaßen:

```
# mit mplayer Programm 1 ansehen:
mplayer http://VDR_SERVER_IP:3000/1
# mit vlc Programm 2 ansehen:
vlc http://VDR_SERVER_IP:3000/2
```

Konfiguration VNSI Server

Auf dem Server muss das Netzwerk zum Verbindungsaufbau in `/etc/vdr/plugins/vnsiserver/allowed_hosts.conf` autorisiert werden, sonst ist ein Verbindungsaufbau nicht möglich.

```
#
# allowed_hosts.conf  This file describes a number of host addresses that
#                     are allowed to connect to the streamdev server running
#                     with the Video Disk Recorder (VDR) on this system.
# Syntax:
#
# IP-Address[/Netmask]
#

127.0.0.1             # always accept localhost
192.168.1.0/24        # any host on the local net
#204.152.189.113      # a specific host
#0.0.0.0/0            # any host on any net (USE THIS WITH CARE!)
```

Debian - ACPI Wakeup mit sysvinit oder systemd

kopiert von: http://www.vdr-wiki.de/wiki/index.php/Debian_-_ACPI_Wakeup_mit_sysvinit_oder_systemd

Voraussetzungen

Bei Debian basierten VDR-Installationen kann das Shutdownscript von VDR nicht direkt auf die RTC zugreifen. Dies lässt sich durch ein zusätzliches Shutdownscript in den Start- Stopmechanismen von Debian sysvinit (bis wheezy) oder systemd (ab jessie) bewerkstelligen. Für die nachfolgenden scripte wird ein ACPI-fähiges Mainboard und rtcwake (in util-linux enthalten) benötigt. Durch die Verwendung von rtcwake zum Setzen der Aufwachzeit in die RTC muss man (bis auf spezielle Fälle) nicht den Pfad zum Device beachten und braucht sich nicht um die verwendete Systemzeit (UTC, Lokale Zeit) kümmern. Mehr dazu hier im VDR-Wiki ACPI_Wakeup.

VDR Shutdown Script

shutdownhook in `/usr/share/vdr/shutdown-hooks/S95.acpi-shutdown.sh` erstellen, sicherheitshalber das eXecute Flag nicht vergessen (`chmod a+x S95.acpi-shutdown.sh`)

Schreibt die Aufwachzeit in die Datei /var/cache/vdr/acpiwakeup.time

S95.acpi-shutdown.sh

```
#!/bin/sh
# $1 : Next timer seconds from 1970 from 1970/01/01, UTC
# $2 : Next timer seconds from now
```

```
# $3 : Next timer title
# $4 : Shutdown forced

# read from acpi-rtcwakeup conf file
. /etc/vdr/vdr-acpi-acpirtcshutdown.conf
WAKEUP_FILE="/var/cache/vdr/acpiwakeup.time"
if [ ! -f "$WAKEUP_FILE" ]
then
  touch $WAKEUP_FILE
fi

# Defaults:
[ -z "$ACPI_REGULAR_TIME" ]  && export ACPI_REGULAR_TIME="00:00"
[ -z "$ACPI_START_AHEAD" ]   && export ACPI_START_AHEAD="300"

    datefixplus=$(date +%s -d 'tomorrow '"$ACPI_REGULAR_TIME"' +1 day')
    if [ $1 -eq 0 ] || [ $datefixplus -lt $1 ]; then
      # wakeup time if next timer ahead > 24h to regular wakeup time
      date +%s -d 'tomorrow '"$ACPI_REGULAR_TIME"'' > $WAKEUP_FILE
      else
      # wakeup time for next timer
      echo $(($1 - $ACPI_START_AHEAD)) > $WAKEUP_FILE
    fi
exit
```

conf Datei anlegen in `/etc/vdr/vdr-acpi-acpirtcshutdown.conf`

vdr-acpi-rtcwakeup.sh

```
# ACPI shutdown enabled
ACPI_ENABLED=yes

# How many seconds wakeup the machine before timer starts
ACPI_START_AHEAD=300

# Start machine at regular time next day if next timer is more than 24
hours ahead
ACPI_REGULAR_TIME=03:15 #HH:MM
```

(Shutdownscript Debian mit sysvinit entfernt, hat Ubuntu 16.04 nicht mehr)

Shutdownscript Debian mit systemd

Für systemd (ab Debian Jessie) muss eine Datei wie folgt in `/lib/systemd/system/acpi-rtcwakeup.service` angelegt werden. (Ob dies der korrekte Ort für ein eigenes Script ist, wäre zu diskutieren.)

acpi-rtcwakeup.service

```
[Unit]
Description=write alarm time to rtc
Before=ntp.service vdr.service
ConditionPathExists=/sys/class/rtc

[Service]
Type=oneshot
RemainAfterExit=true
ExecStart=/bin/true
ExecStop=/bin/bash -c 'TIME_TO_SET=$(</var/cache/vdr/acpiwakeup.time) ;
/usr/sbin/rtcwake -m no -a -t $TIME_TO_SET'

[Install]
RequiredBy=vdr.service
```

Zu achten ist auf den Punkt „`Before=ntp.service`“. Hier wurde der Service ntp gewählt. Dieser muss natürlich vorhanden sein. Ansonsten sollte man sich eine andere Referenz suchen vor - und damit vor allem nach welcher der Shutdownbefehl ausgeführt wird. Beim Shutdown gilt die umgekehrte Reihenfolge wie beim Start.

Userrechte sollten bei root liegen

Dann den Service aktivieren mit:

```
systemctl enable /lib/systemd/system/acpi-rtcwakeup.service
```

Den Service starten mit:

```
systemctl start acpi-rtcwakeup.service
```

Den Status des Service kann man hiermit überprüfen:

```
systemctl status acpi-rtcwakeup.service
```

Test

Nun kann man einen ersten Test durchführen

Eine Zeit in `/var/cache/vdr/acpiwakeup.time` in der Zukunft schreiben:

```
 date +%s -d 'tomorrow 15:30' > /var/cache/vdr/acpiwakeup.time
```

Den Service stoppen (Nicht vergessen danach wieder zu starten):

```
systemctl stop acpi-rtcwakeup.service
```

```
root@server:# cat /proc/driver/rtc
rtc_time    : 20:17:28
rtc_date    : 2015-03-04
alrm_time   : 14:30:00
alrm_date   : 2015-03-05
alarm_IRQ   : yes
alrm_pending    : no
```

(Die Stunde zurück liegt daran dass die Systemzeit in UTC läuft)

Kodi einrichten

SD-Image von openElec downloaden und installieren

SD-Karte neu mounten

```
mount -o remount,rw /flash
nano /flash/config.txt
```

und in der `config.txt` den Mpeg Licence key eintragen, z.B.:

```
decode_MPG2=0xf98c61ea
decode_MPG2=0xfbbf0968
```

die Karte wieder schreibschützen

```
mount -o remount,ro /flash
reboot
```

dann Raspi booten, die Installation durchlaufen und das vdr plugin installieren. Dabei die MAC Adresse des VDRs nicht vergessen 🙂

Kodi Switch- Off einrichten

Mit einem Server im Netz gibt es ein Problem: Kodi auf einem Raspi schläft nie, also fragt er permanent die EPG Daten ab. Dadurch bleibt der VDR Server immer an und läuft dann 24/7. Das ist natürlich nicht gewollt, aber da Kodi trotz allem Suchens (http://forum.kodi.tv/showthread.php?tid=226774&page=2) scheinbar keine Möglichkeit bietet, trotz CEC- Anbindung auf einen ausgeschalteten TV zu reagieren, half am Ende nur der Umstand, dass der TV im eingeschalteten Zustand quasi eine IP- Adresse hat, mit der man dann bei abgeschaltetem TV per Holzhammermethode den Kodi- Prozess anhalten und starten kann und somit den VDR- Server zum Einschlafen bekommt.

Dazu braucht man ein Überwachungsscript in `/storage/`, welches den Chromecast anpingt und entsprechend Kodi an- und ausschaltet:

pingsleepcontrol.sh

```
#!/bin/bash

while :; do
        systemctl is-active kodi > /dev/null
        isRunning=$?
        # isrunning is 0, when kodi is running
        # echo "isrunning:" $isRunning " tvIsOn:" $tvIsOn

        # seaching for chromecast devices
        #avahi-browse -t -p -r _googlecast._tcp | grep -i chromecast |
grep -q -i <yourChromeDeviceName>
        # in case of a fixed TV device IP adress
        if [ "$isRunning" -eq 0 ] ; then
                # the name of the TV needs to be configured in the home
router
                # otherways use a fixed
                #i=360
                i=5
                while [ $i -gt 0 ]
                do
                        # the bash has no build in command to decrease
the loop counter, so we have to do a workaround
                        i=$(python -c "import sys;
print(int(sys.argv[1])-1)" $i )
                        ping -c 1 192.168.1.59 > /dev/null
                        tvIsOn=$?
                        if [ "$tvIsOn" -eq 0 ] ; then
                                break
                        else
                                sleep 5
                        fi
                        echo "Loop: $i"
                done
                echo "Left loop..."
        else
                TvCecStatus=$(echo pow 0 | cec-client -s -d 1 | grep
'power status' | awk '{split($0,a,":");print a[2]}' | sed 's/ //')
                if [ "$TvCecStatus" == "on" ]; then
                        tvIsOn=0
                else
                        tvIsOn=1
                fi
        fi
        if [ "$tvIsOn" -ne 0 -a "$isRunning" -eq 0 ] ; then
                date +"kodi stops at %c" >> /tmp/kodi.log
                systemctl stop kodi
        elif [ "$tvIsOn" -eq 0 -a "$isRunning" -ne 0 ] ; then
                date +"kodi starts at %c" >> /tmp/kodi.log
                systemctl start kodi
```

```
        fi
        sleep 10
done

exit 0
```

Koki Wake-on-Lan einrichten

So richtig faul wird's dann, wenn man den Fernseher schon mal auf dem Weg dahin per Wake-on-LAN einschalten kann.

Dazu horcht das Python Script auf das Magic Packet auf Port 9

wol_listen.py

```
import socket
import pprint
import struct
import sys

# Kommando. um den Fernseher anzubekommen:
# echo 'on 0' | cec-client -s

from uuid import getnode as get_mac

# seems to be the easiest way to get the "outer" IP address. All other
tricks failed as the host name is hardcoded against 127.0.0.1 in
/etc/hosts on Kodi :-(

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.connect(("8.8.8.8", 80))
UDP_IP=s.getsockname()[0]
s.close()
UDP_PORT = 9

sock = socket.socket(socket.AF_INET, # Internet
                     socket.SOCK_DGRAM) # UDP
sock.bind((UDP_IP, UDP_PORT))
mac = get_mac()
mac=''.join(("%012X" % mac)[i:i+2] for i in range(0, 12, 2))

# create magic packet
magic_packet = ''.join(['FF' * 6, mac * 16])
send_data = ''
for i in range(0, len(magic_packet), 2):
        send_data = ''.join([send_data, struct.pack('B',
int(magic_packet[i: i + 2], 16))])
```

```
result=0
while result==0:
        data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
        #print ("received message:")
        #pprint.pprint(data)
        #print("--------")
        #pprint.pprint(send_data)
        if data==send_data:
                print("Magic Paket identified")
                result=1
sys.exit(result)
```

und wenn ein Magic Packet eintrifft,schaltet das Script dann Fernseher per CEC und Kodi selber wieder ein:

wol_listen.sh

```
#!/bin/bash

while :; do
        python wol_listen.py
        magicPacket=$?
        if [ "$magicPacket" -eq 1 ] ; then
                systemctl is-active kodi > /dev/null
                isRunning=$?
                # isrunning is 0, when kodi is running
                if [ "$isRunning" -ne 0 ] ; then
                        echo 'TV & Kodi on'
                        echo 'on 0' | cec-client -s
                        systemctl start kodi
                fi
        fi
        sleep 10
done
exit 0
```

Autostart

Kodi WoL und -Off werden dann per `/storage/.config/autostart.sh` gleich beim Booten gestartet

autostart.sh

```
(
/storage/pingsleepcontrol.sh
) &
```

```
nohup storage/wol_listen.sh &
```

Nicht vergessen, die beiden neuen Dateien per chmod a+x auch ausführbar zu machen.

Dann kann man noch eben den Newsfeed in /storage/.kodi/userdata/RssFeeds.xml etwas allgemeiner gestalten

RssFeeds.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rssfeeds>
  <!-- RSS feeds. To have multiple feeds, just add a feed to the set.
You can also have multiple sets.  !-->
  <!-- To use different sets in your skin, each must be called from
skin with a unique id.              !-->
  <set id="1">
    <feed updateinterval="30">http://www.tagesschau.de/xml/rss2</feed>
  </set>
  <set id="2">
    <feed updateinterval="30">http://feeds.openelec.tv/news</feed>
    <feed updateinterval="30">http://feeds.xbmc.org/xbmc</feed>
    <feed
updateinterval="30">http://feeds.xbmc.org/latest_xbmc_addons</feed>
    <feed
updateinterval="30">http://feeds.xbmc.org/updated_xbmc_addons</feed>
  </set>
</rssfeeds>
```

Fernbedienungs- Keymap

Dann kann man noch, wenn notwendig, die Tastatur seiner Fernbedienung anpassen.

Dazu zieht man sich erstmal eine Debug- Keymap, die jede gedrückte Taste der Fernbedienung auf dem Fernseher als Notification anzeigen kann:

```
cd /storage/.kodi/userdata/keymaps/
curl
https://gist.githubusercontent.com/stevenocchipinti/42f2eca2a9f04ed9e52f/raw
/remote.xml > remote.xml
sudo reboot
```

und schreibt danach die remote.xml nach Bedarf um. Die dafür eventuell notwendige Section-ID läßt sich hier finden: http://kodi.wiki/view/keymap#Remotes

remote.xml

```
<keymap>
```

```
    <tvrecordings>
        <remote>
            <yellow>contextmenu</yellow>
        </remote>
    </tvrecordings>
</keymap>
```

oder falls man's noch auf der Platte hat, die Backup-Zip Datei der obigen Settings per

```
scp sk_OpenELec_Kodi_settings.zip root@192.168.1.127:/storage
```

auf den Raspi übertragen und einfach dort nur auspacken

Weitere Einstellungen

- unter System/add-ons/OpenElec-Repository/pvr-plugins den VNSI- client installieren
- den „Expert“-Modus bei den Settings aktivieren
- „Aeon Nox“ als Skin installieren
- unter TV- Settings TV aktivieren
- Die EPG- Vorlaufzeit auf 31 Tage setzen
- bei EPG „Aktualisierung bei Wiedergabe verhindern“ deaktivieren, sonst bleibt das EPG leer!
- unter „Aufnahme“ die Vor- und nachlaufzeit auf 5minuten
- unter Region/Länder die Zeitzone auf Deutschland
- unter „Darstellung“ TV als Startseite
- überflüssige Seiten ausblenden (Programme, Wetter usw.)
- Mediaserver bei den Video-Quellen hinzufügen
- und nicht vergessen, die MPEG- Lizenz einzuspielen

Damit kann der neue Kodi dann endlich mal in den Produktiveinsatz übergehen...

Backup & Restore

Solange sich die Version, sprich das schreibgeschützte Image von Kodi/OpenElec nicht ändert, scheint es zu genügen, die gesamte `/storage` Partition zu archivieren bzw. wieder herzustellen, was man auch sehr einfach machen kann, wenn die SD-Karte im Desktop- Rechner steckt

Backup

```
cd /media/steffen/ebbb55fb-ef6e-4d19-8474-71f61e77d066/
tar -czvf ./Desktop/OpenElec_Kodi_Backup_Wohnzimmer_storage_20181003.tar.gz
.
```

Restore

Neues Kodi- Image einmal booten lassen, damit der automatische Partitions- Größenanpassung laufen kann, den Rechner wieder runterfahren und die Karte wieder in den Desktop - PC. Dann

```
cd /media/steffen/ebbb55fb-ef6e-4d19-8474-71f61e77d066/
sudo tar -xzvf
./Desktop/OpenElec_Kodi_Backup_Wohnzimmer_storage_20181003.tar.gz
```

[1)]

vielen Dank an Sigi Gassner für diesen Tipp