

RASPBIAN – DOCUMENTATION – 1_2019 // Barbara Eder

Anmerkung: Diese Dokumentation ist work in progress, bei Nachfragen, Verbesserungsvorschlägen und/oder Unklarheiten Email an:

office@barbaraeder.org

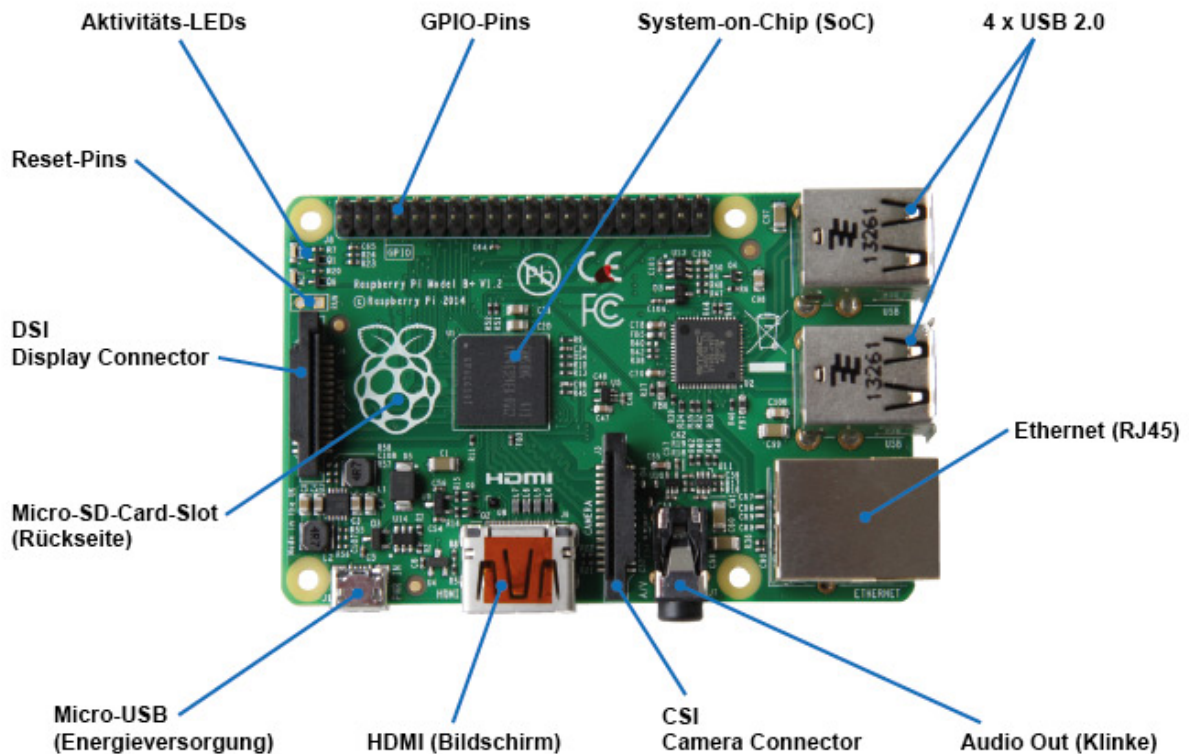
Der Name „candy“ bzw. „sugar“¹ steht für den jeweiligen Namen des Linux- bzw. Raspberry-User*s und muss in den Commands individuell angeglichen werden („name“).

INHALT

A. ALLGEMEIN – Was ist ein Raspberry Pi & was kann er?	2
1. Die microSD-Karte beschreiben	3
a. unter Linux.	3
b. unter Windows	6
c. unter MacOS	7
2. NETZWERKKONFIGURATION	8
3. FIRST BOOT	10
a. nativer Zugang zum Raspberry Pi	10
b. OLIMEX-Entwickler_innenkabel	10
Serielle Verbindung mit GNU screen (Linux)	11
Serielle Verbindung mit PuTTY (WINDOWS & MacOS)	12
4. Enter the SHELL! Server-Client-Verbindungen	13
a. unter Linux.	13
b. unter Windows & MacOS	13
B. ARBEITEN UNTER RASPBIAN	14
1. Make your prompt! Farbe geben & User* anlegen	14
2. SOUND ALSA-Soundsystem	18
3. AUDIO-AUFNAHME arecord	20
4. VIDEO fswebcam	22
5. BUS-ERWEITERUNGEN I2C & SPI	23
6. LIBRARIES	26

¹ „candy“ and „sugar“ are part of a *namespace*, that encourages *syntactic sugar* instead of *syntactic saccharin* or *syntactic syrup*. These names are chosen to make computing „sweeter“, that is: to make things easier & shorter.

A. ALLGEMEIN



Raspbian ist das Betriebssystem (OS) für den Raspberry Pi und basiert auf der Linux-Distribution Debian. Um es nutzen zu können, müssen wir es auf die *microSD*-Karte für den Raspberry Pi schreiben.

Für den First Boot des Raspberry Pi wird ein nativer Zugang (Maus, Tastatur, externer Bildschirm und/oder HDMI-Kabel) oder ein OLIMEX-Entwickler_innen-Kabel benötigt, mithilfe dessen die Daten der jeweiligen Pins ausgelesen werden können (Voraussetzung: Verbindung über Putty oder ein anderes *Terminal Emulation Programm*). Die dazugehörigen Arbeitsschritte werden in Punkt A angeführt, dieser endet mit der Konfiguration des Netzwerkzugangs für den Raspberry Pi.

Derzeit gibt es 4 Versionen von Raspbian:

- a. **stretch** – aktuelle Laufzeitversion: <https://www.raspberrypi.org/downloads/raspbian/>
- b. **jessie** – die Entwicklung wird derzeit nicht fortgeführt
- c. **wheezy** – die Entwicklung wird aktuell nicht unterstützt

d. noobs – OS für Anfänger_innen bei nativem Zugang, Nachteil: weniger Kontrolle über automatisch vorgenommene Einstellungen, das Beschreiben funktioniert „automagisch“:
<https://www.raspberrypi.org/downloads/noobs/>

Ebenso gibt es Lite-Versionen von Raspbian (maximal 2 GB) für weniger leistungsstarke *microSD*-Karten; sämtliche Raspbian-Distributionen finden sich hier:

<https://downloads.raspberrypi.org/raspbian/images/>

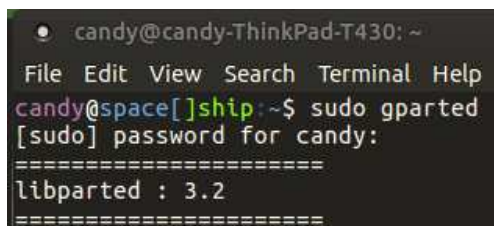
1. Die *microSD*-Karte beschreiben

In einem ersten Schritt formatieren und beschreiben wir die für den Slot am Raspberry vorgesehene *microSD*-Karte. Wir stecken sie (im Adapter) in den dafür vorgesehenen Eingang am PC und formatieren sie.

a. unter Linux

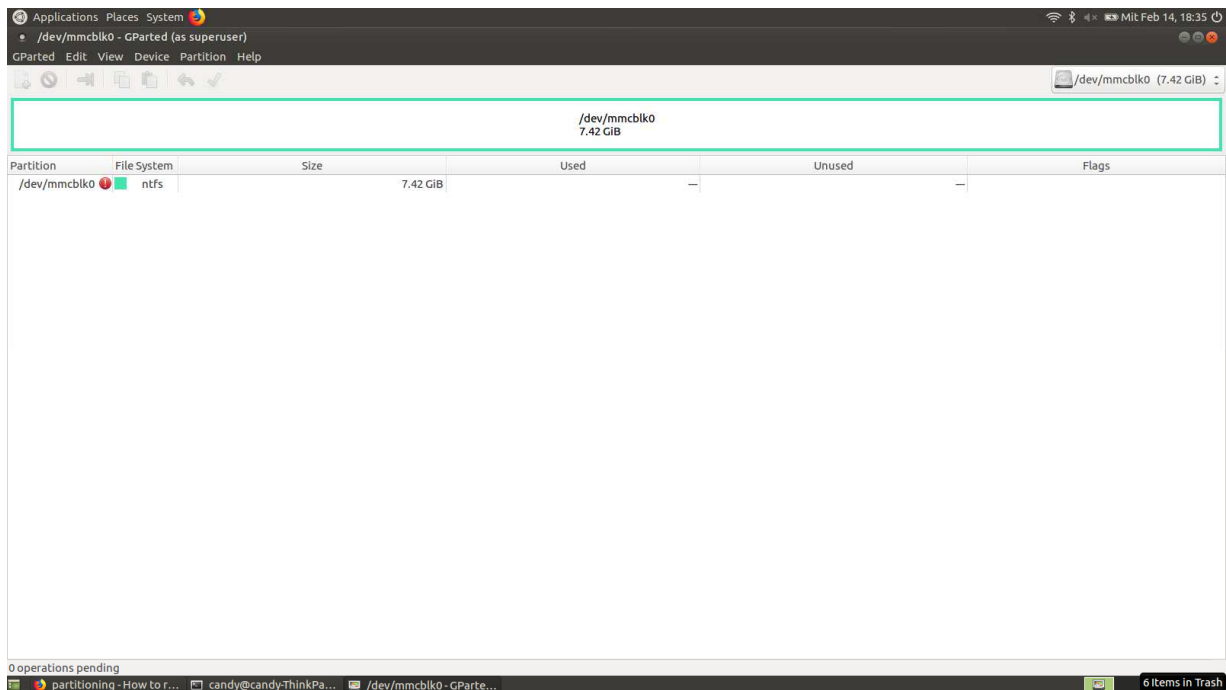
Linux bietet mehrere Möglichkeiten die *microSD*-Karte zu formatieren, dafür gibt es die Programme *GParted*, *Parted* und *fdisk*. Wir starten *GParted* im Terminal mit

```
sudo gparted
```



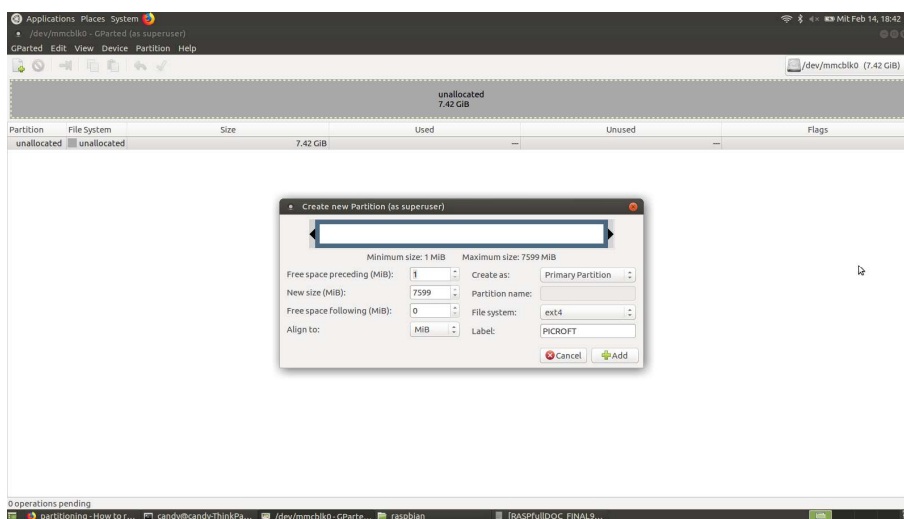
```
● candy@candy-ThinkPad-T430: ~
File Edit View Search Terminal Help
candy@space[ ]ship:~$ sudo gparted
[sudo] password for candy:
=====
libparted : 3.2
=====
```

Wir wählen im Fenster rechts oben das Laufwerk **/dev/mmcblk0** (Standard für *microSD*-Karten-Slot unter Linux).

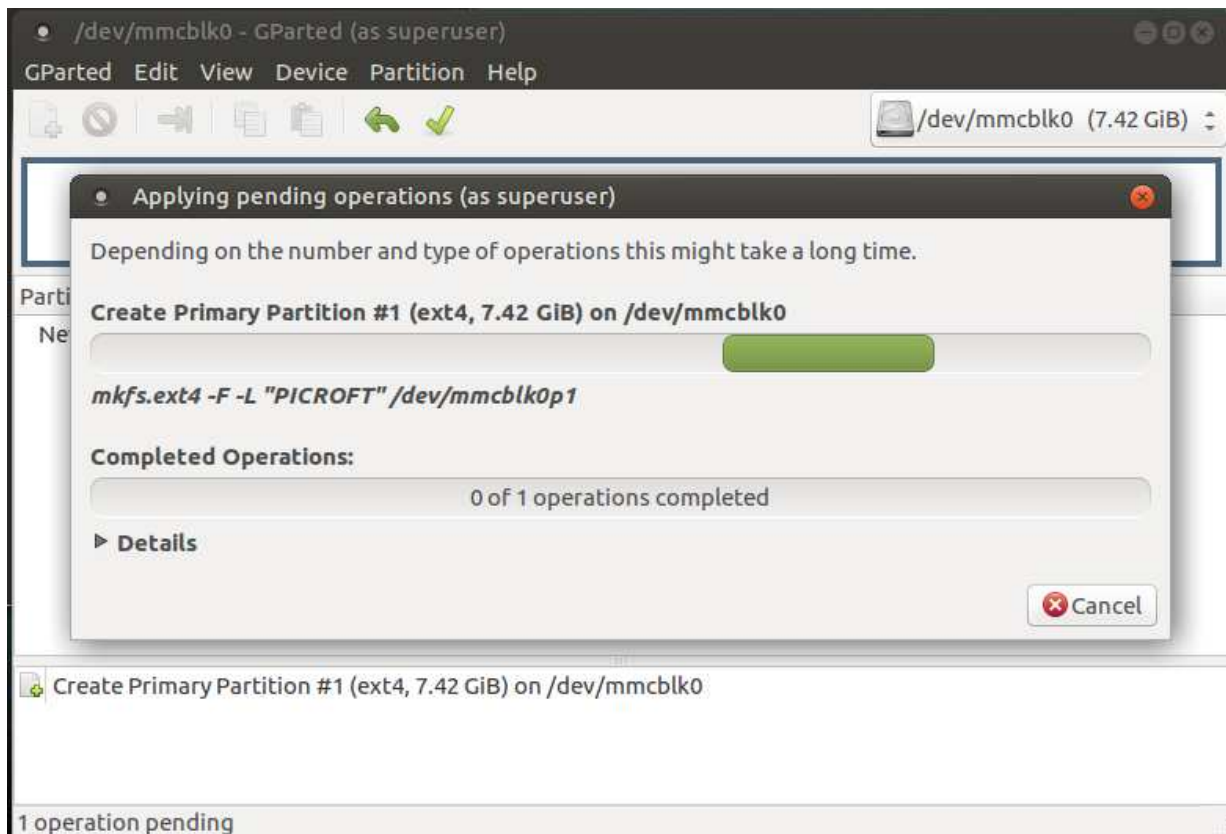


Mit Rechtsklick auf die nicht zugewiesene Partition, **unmount**, dann **delete**; es existiert nur noch eine einzige nicht zugewiesene Partition (**unallocated**).

Rechtsklick in **unallocated**, dann auf **New** im Auswahlmeneü klicken. Für die Formatierung werden uns verschiedene Filesysteme angeboten, unter Linux sind ext2, ext3, ext4 zulässig – wir entscheiden uns für eine dieser Möglichkeiten und tragen – je nach Bedarf – ein Label ein.



Nun wird die **Partitionierung** der *microSD*-Karte durchgeführt:



Die alte Partition wurde gelöscht, eine neue angelegt. Die *microSD*-Karte ist jetzt formatiert!
Wir rebooten mit

```
sudo reboot
```

Nach dem Neustart wird die *microSD*-Karte am Desktop angezeigt, wir können sie nun mit dem Image des Raspbian-Betriebssystems beschreiben!

Die Karte lokalisieren mit

```
df -h
```

Wir kopieren das Image der gewünschten Raspbian-Version – hier ist es [2017-07-05-raspbian-jessie.img](#), zum Download anderer Versionen siehe Seite 1 – mit root-Rechten auf die *microSD*-Karte – Vorsicht beim Verzeichnis, hier ist es `/dev/mmcblk0`, nicht aber eines der beiden partitionierten Verzeichnisse (`/dev/mmcblk0p1` oder `/dev/mmcblk0p0`)!

```
sudo -s
```

Der Befehl zum Beschreiben der *microSD*-Karte lautet wie folgt:

```
dd bs=4M if=2017-07-05-raspbian-jessie.img of=/dev/mmcblk0
```

```
root@space[~]ship ~/raspbian# ls
2017-07-05-raspbian-jessie-lite.img  OSMC_TGT_rbp2_20180207.img  RASPBIAN_dokus
2017-09-07-raspbian-stretch.img    PiCroft_v0.8b_Raspian_JessieLite_2017-01-26.img  RASPFULLDOC_FINAL9_2017.pdf
root@space[~]ship ~/raspbian# dd bs=4M if=PiCroft_v0.8b_Raspian_JessieLite_2017-01-26.img of=/dev/mmcblk0
954+0 records in
954+0 records out
4001366016 bytes (4,0 GB, 3,7 GiB) copied, 167,977 s, 23,8 MB/s
root@space[~]ship ~/raspbian#
```

dd - Befehl zum kopieren von Datensätzen zwischen devices

bs - bytes per second
hier: 4 MB

Debian-Image,
das auf SD-Karte
geschrieben werden soll

Angabe des Zielvolumens;
hier ist es der Pfad der
SD-Karte mmcblk0

Die *microSD*-Karte ist nun fertig für die Inbetriebnahme!

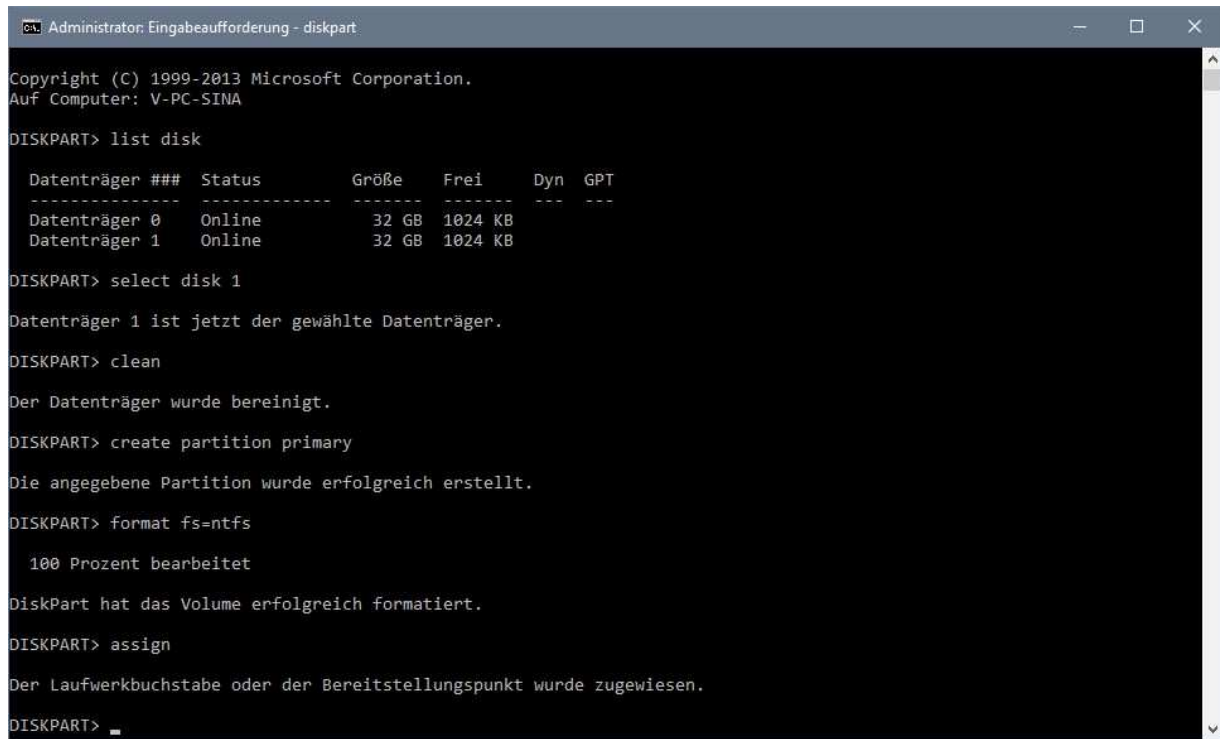
b. unter WINDOWS

...erfolgt die *microSD*-Karten-Partitionierung mit dem **Win32DiskImager**, das benötigte Dateisystem ist ntfs.

Dazu das cmd-Fenster öffnen und die Befehle im Screenshot ausführen (siehe unten)

Nähere Infos zum Beschreiben der *microSD*-Karte mit dem Programm **Etcher** findest du hier:
<https://etcher.io/>

<https://www.raspberrypi.org/documentation/installation/installing-images/windows.md>



```
Administrator: Eingabeaufforderung - diskpart
Copyright (C) 1999-2013 Microsoft Corporation.
Auf Computer: V-PC-SINA

DISKPART> list disk

Datenträger ###  Status           Größe   Frei     Dyn  GPT
-----
Datenträger 0   Online          32 GB   1024 KB
Datenträger 1   Online          32 GB   1024 KB

DISKPART> select disk 1
Datenträger 1 ist jetzt der gewählte Datenträger.

DISKPART> clean
Der Datenträger wurde bereinigt.

DISKPART> create partition primary
Die angegebene Partition wurde erfolgreich erstellt.

DISKPART> format fs=ntfs
100 Prozent bearbeitet
DiskPart hat das Volume erfolgreich formatiert.

DISKPART> assign
Der Laufwerkbuchstabe oder der Bereitstellungspunkt wurde zugewiesen.

DISKPART> _
```

c. unter MacOS

...ist das benötigte Dateisystem für die *microSD*-Karte FAT32; die *microSD*-Karte liegt im Verzeichnis

`/dev/disk1s2`

Beschrieben wird die *microSD*-Karte ebenso mit `dd`, der Befehl lautet:

```
sudo dd bs=1m if=2017-07-05-raspbian-jessie.img of=/dev/rdiskn conv=sync
```

Eine detaillierte Anleitung dazu findest du hier:

<https://www.raspberrypi.org/documentation/installation/installing-images/mac.md>

Zum Beschreiben der *microSD*-Karte kannst du übrigens auch als Mac-User_in **Etcher** nutzen – sofern du mit dem Terminal nicht allzu vertraut bist, ist dies die einfachere Lösung:

<https://etcher.io/>

2. NETZWERKKONFIGURATION

Für den First Boot des Rasperrys ist es nötig, im Vorfeld die Netzwerkeinstellungen zu konfigurieren, damit auf das jeweilige WLAN zugegriffen werden kann. Des Weiteren muss im Vorfeld eine Portweiterleitung am Router gemacht werden – dies ist ohne Informationen zum Netzwerk nicht möglich. Fragt also den/die Admina* eures Netzwerkes im Vorfeld!

Es gibt zwei Möglichkeiten eine Netzwerkverbindung für den Raspberry Pi einzurichten – über eine dynamische oder über eine statische IP-Adresse. Der Weg zur Einrichtung letzterer wird hier beschrieben – die statische Adresse ist für unsere Zwecke besser geeignet, da sie konstant bleibt und nicht jedes Mal vom Router neu vergeben wird.

Wir sehen uns dazu nochmals die – nunmehr formatierte – *microSD*-Karte an und gehen wie folgt vor:

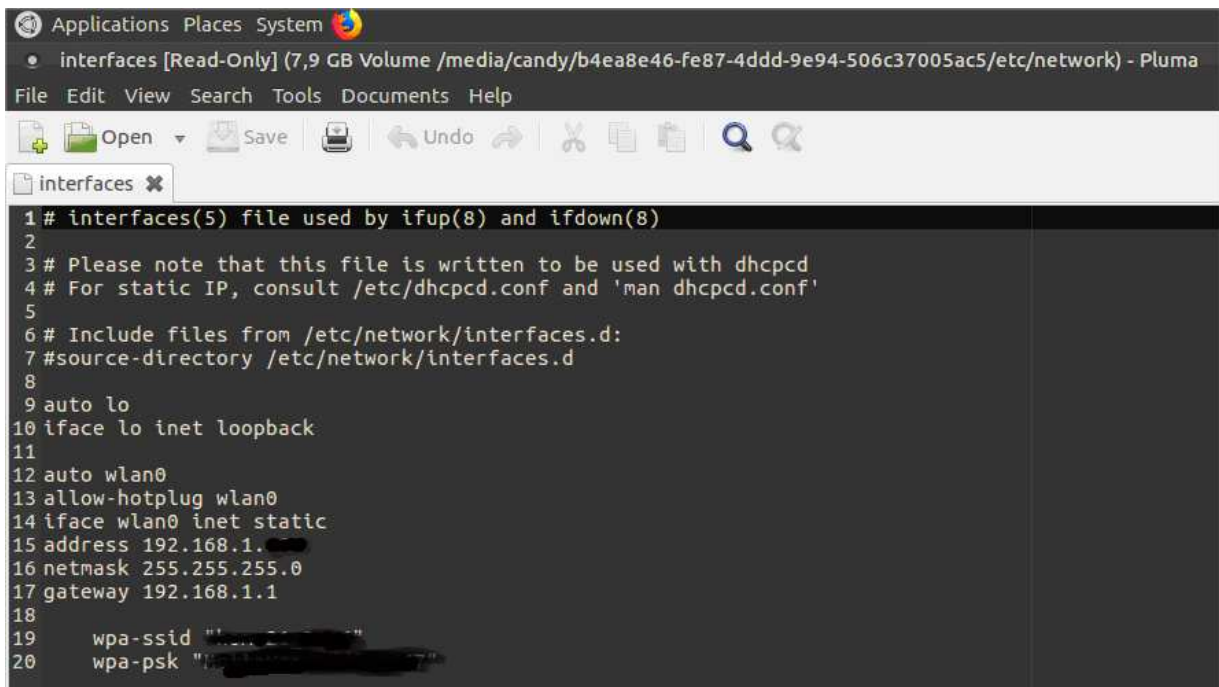
Die Datei interfaces suchen – sie befindet sich im Verzeichnis unter

`/etc/network/interfaces`

interfaces ist das WIFI-Konfigurationsfile in Linux-Betriebssystemen. Wir öffnen die Datei – hier mit dem Editor nano –

`sudo nano interfaces`

- und konfigurieren es wie folgt:



```
1 # interfaces(5) file used by ifup(8) and ifdown(8)
2
3 # Please note that this file is written to be used with dhcpcd
4 # For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'
5
6 # Include files from /etc/network/interfaces.d:
7 #source-directory /etc/network/interfaces.d
8
9 auto lo
10 iface lo inet loopback
11
12 auto wlan0
13 allow-hotplug wlan0
14 iface wlan0 inet static
15 address 192.168.1.100
16 netmask 255.255.255.0
17 gateway 192.168.1.1
18
19     wpa-ssid "XXXXXXXXXX"
20     wpa-psk "XXXXXXXXXX"
```


VORSICHT! Hier werden selbstverständlich die für das jeweilige Netzwerk relevanten Daten eingegeben:

```
wpa-ssid "name meines netzwerks"  
wpa-psk "mein passwort"
```

Die interne Adresse des Raspberry Pi ist in diesem Fall 192.168.1.10, sie liegt innerhalb des zulässigen Adressraums xxx.xxx.x.10 bis xxx.xxx.x.100.

Speichern mit STRG + O und schließen mit STRG + X, dann

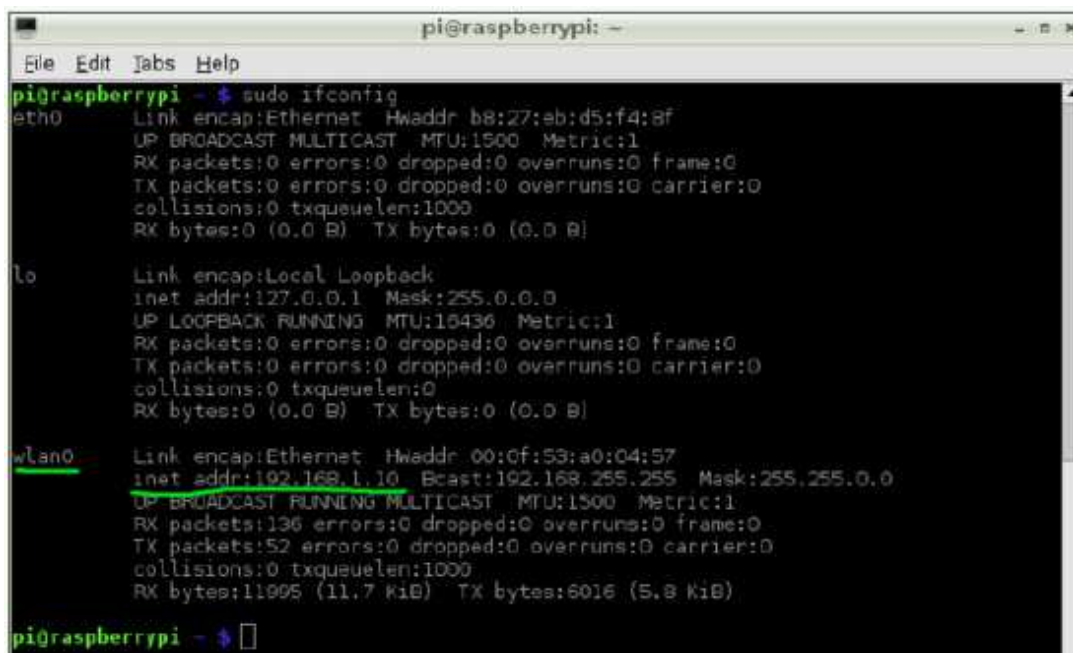
```
sudo reboot
```

Nach dem Neustart findet der Raspberry die ihm zugewiesene Netzwerk-Adresse von selbst, wir können die dazugehörigen Einstellungen aufrufen mit

```
sudo ifconfig
```

oder mit

```
ip a
```



```
pi@raspberrypi ~  
File Edit Tabs Help  
pi@raspberrypi ~$ sudo ifconfig  
eth0      Link encap:Ethernet  HWaddr b8:27:eb:d5:f4:8f  
UP BROADCAST MULTICAST  MTU:1500  Metric:1  
RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:1000  
RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)  
  
lo        Link encap:Local Loopback  
inet addr:127.0.0.1  Mask:255.0.0.0  
UP LOOPBACK RUNNING  MTU:16436  Metric:1  
RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:0  
RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)  
  
wlan0     Link encap:Ethernet  HWaddr 00:0f:53:a0:04:57  
inet addr:192.168.1.10  Bcast:192.168.255.255  Mask:255.255.0.0  
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
RX packets:136 errors:0 dropped:0 overruns:0 frame:0  
TX packets:52 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:1000  
RX bytes:11995 (11.7 KiB)  TX bytes:6016 (5.8 KiB)  
  
pi@raspberrypi ~$
```

Die IP-Adresse wird im Eintrag wlan0 angezeigt, in diesem Fall ist es 192.168.1.10

3. FIRST BOOT

Für den First Boot unseres Rasperrys gibt es zwei Möglichkeiten:

a. nativer Zugang zum Raspberry Pi (externe Maus, Tastatur und Bildschirm mit HDMI-Kabel)

Dazu wird die *microSD*-Karte in den Slot am Raspberry gesteckt und die externen Komponenten angeschlossen – alles weitere funktioniert nun „automagisch“! Die Initialisierung der Starterdatei *systemd* kann am externen Bildschirm bzw. am über HDMI angeschlossenen Gerät beobachtet werden, danach wirst du zum login aufgefordert. Es ist allerdings auch möglich, dass der Bildschirm nicht erkannt wird, dann wird ein X11-Client² nötig.

Für die WLAN-Verbindung muss eventuell zuvor eine Portweiterleitung am Router gemacht werden!

b. OLIMEX-Entwickler_innenkabel

Sofern keinerlei externe Komponenten vorhanden sind, kann der Raspberry auch auf folgendem Weg hochgefahren werden:

Dafür muss UART – eine spezielle Datenleitung des Rasperrys namens *Universal Asynchronous Receiver/Transmitter* – aktiviert werden.

Wir sehen uns dazu nochmals die formatierte *microSD*-Karte am Home-Computer an. Diese wird am Desktop in 2 Teilen – BOOT und ROOT – angezeigt. Wir öffnen den BOOT-Teil. Darin befindet sich die gesuchte Datei `config.txt`. Wir öffnen sie mit

```
sudo nano config.txt
```

und fügen am Ende folgende Textzeile hinzu:

```
enable_uart=1
```

Die *microSD*-Karte wird in den dafür vorgesehenen Slot am Raspberry Pi gesteckt und der USB-Anschluss des Olimex-Kabels an den jeweiligen Eingang am Home-Computer – es sieht so aus:

² X – also known as WINDOWS-server-client-architecture from a LINUX-user’s perspective – “was developed in the year of 1987 by the MIT lab and since then it has been deployed on every UNIX like system known to man. Currently it is in its 11th iteration hence, it is also know as ‘X11’ or simply as ‘X’. If there is GUI, there is X server”, quoted: <https://medium.com/mindorks/x-server-client-what-the-hell-305bd0dc857f> [last call: 21.10.2019]



Dann schließen wir die 3 kleinen Anschlüsse des OLIMEX-Kabels an die Pins des Raspberry Pi an, diese werden wie folgt verbunden:

pin 6: Ground (blau)

pin 8: Rx (grün)

pin 10: Tx (rot)

Nun geht es darum, mithilfe des OLIMEX-Entwickler_innenkabels eine Verbindung mit dem Raspberry Pi herzustellen. Dafür benötigen wir ein *Terminal Emulation Program* – unter Linux ist das etwa GNU screen (a), unter WINDOWS & MacOS der Server-Client PuTTY (b) – for details see

http://elinux.org/Rpi_Serial_Connection

a. Serielle Verbindung mit GNU screen (Linux)

Im ersten Schritt müssen wir in den tty-Dateien nach dem Port-Namen des USB-Kabels suchen, dieser ist meistens `/dev/ttyUSB0` oder `/dev/ttyACM0`. Wir überprüfen dies mit

```
ls -l /dev/ttyUSB0
```

Um herauszufinden, ob dieser Anschluss Teil der Gruppe *dialout* ist, geben wir den Command

```
id
```

ein und fügen diesen nach Bedarf hinzu mit

```
sudo usermod -a -G dialout <candy>
```

Wir rufen nun das *terminal emulation program* GNU screen auf mit

```
screen ttyUSB0 -a -G dialout 115200
```

Am Bildschirm erscheint das Konsolenfenster. Wir drücken ENTER und werden nach den Standard-Login-Daten des Raspberry Pi gefragt. Login mit

Benutzername: pi
Passwort: raspberry

b. Serielle Verbindung mit PuTTY (WINDOWS & MacOS)

Unter WINDOWS & MacOS erfolgt der Zugang über PuTTY. COM-Port im Gerätemanager herausuchen – meistens ist dies COM7 – dann PuTTY öffnen und folgende Einstellungen vornehmen:

Verbindung: serial
Port: COM7
Speed: 115200

dann Verbindung starten! Idealerweise erscheint am Bildschirm das Konsolenfenster, wir loggen uns mit Benutzernamen und Passwort ein. Wir können nun damit beginnen, unseren Raspberry einzurichten mit

```
sudo raspi-config
```

Im Menü können wir verschiedene Einstellungen vornehmen, z.B.

-> Zeitzone, Sprache, Keyboard
-> Interface Options: SSH muss als Standard-Verbindung ausgewählt werden!

Menü dann beenden und

```
sudo reboot
```

Nach dem Neustart erfolgt erneut die Anmeldung über das Login-Prompt mit den Standard-Daten. Die Desktopoberfläche erscheint und wir müssen Raspbian abschließend nur mehr aktualisieren mit

```
sudo apt-get update  
sudo apt-get dist-upgrade
```

4. Enter the SHELL! | Server-Client-Verbindungen

Zum Thema Shell-Navigation gibt es eine eigene Einführung – an dieser Stelle ist es lediglich relevant wie wir die Verbindung zur Konsole unter LINUX und WINDOWS / MacOS herstellen. Dafür ist eine Client-Server-Verbindung nötig, die gemeinsame Sprache von Server und Client wird durch das jeweilige Protokoll festgelegt, es definiert die Art der User*-Authentifizierung, die der Verschlüsselung und stellt die Verbindung zum Terminal her.

a. unter LINUX:

Protokoll/Sprache: ssh-protocol (SSH - Secure Shell Host Protocol)

ssh ist Standard in allen Linux-Betriebssystemen, im Fall des Falles nachinstallieren mit

```
sudo apt-get install openssh-client
```

Wir öffnen einen Terminal am Home-Computer und geben den ssh-Command mitsamt der Netzwerk-Adresse des Raspberry Pi ein

```
ssh pi@192.168.1.10
```

Es erfolgt die Aufforderung zum Login, danach befinden wir uns im Betriebssystem des Raspberry Pi; die Navigation kann ausschließlich über das Terminal erfolgen – einen Client mit grafischer Oberfläche könnt ihr dazustallieren.

B. ARBEITEN UNTER RASPBIAN

1. Make your prompt! | FARBE geben & User* anlegen

pi ist der Standard-User unter Raspbian. Mit vorangestelltem `sudo` kann pi Commands mit root-Rechten ausführen. **root** ist der_ die mächtigste User*in unter Linux und standardmäßig vorhanden, diese_r verfügt über sämtliche Berechtigungen im System; für alle anderen User_innen werden diese gesondert vergeben (die Berechtigungen wurden zu anderem Zeitpunkt erklärt und finden an dieser Stelle deshalb keine gesonderte Erwähnung).

Bevor du **root** wirst, beachte bitte folgendes:

```
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

root's password:
```

Vom pi-User* um root-User* wechseln mit

```
sudo su
```

(beenden mit `exit`)

Es ist ebenso möglich, weitere User* im system anzulegen, z.B. den User* *candy*:

```
sudo adduser candy
```

Anschließend wirst du nach dem Passwort für den_ die neue User*in gefragt, wenn du keines angeben möchtest, lass das Feld frei!

Folgende Commands ermöglichen den Wechsel zwischen den unterschiedlichen Usern*:

```
su - pi
```

für den Wechsel zum User pi. Damit wechselst du auch in das jeweilige Verzeichnis des Users; wenn du in derselben GUI bleiben möchtest, reicht folgender command aus:

```
su pi
```

```
candy@space[ ]ship:/dev$
```

Möchtet ihr euer Prompt – das ist die Zeile mit der Eingabeaufforderung (siehe oben) – kolorieren? Dies funktioniert über ein Shell-Script, in dem ihr bestimmte Änderungen vornehmen könnt – anbei die Anleitung:

Das Shell-Script `bashrc` befindet sich im Home-Verzeichnis der jeweiligen User*in. Wir erstellen dort eine Kopie des Originals mit

```
cp .bashrc .bashrcORG
```

Sofern etwas „schief“ geht, kann man mit dem folgenden Command zum farblosem prompt zurückkehren:

```
mv .bashrcORG .bashrc
```

Konfigurations-Datei öffnen mit

```
sudo nano .bashrc
```

und darin auf die Zeile „set a fancy prompt“ achten. Der darunter stehende Text muss wie folgt formatiert werden:

```
# set a fancy prompt (non-color, unless we know we "want" color)
case "$TERM" in
    xterm-color|*-256color) color_prompt=yes;;
esac
```

Ebenso hier die Zeile „color_prompt“ auf „yes“ setzen:

```
# uncomment for a colored prompt, if the terminal has the capability; turned
# off by default to not distract the user: the focus in a terminal window
# should be on the output of commands, not on the prompt
```

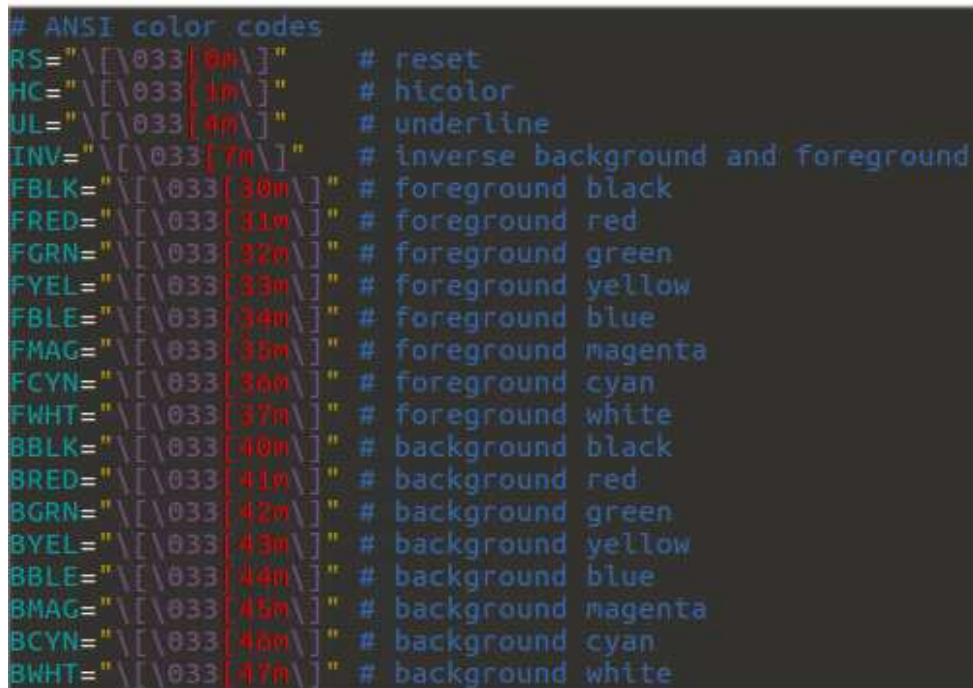
```
force_color_prompt=yes
```

```
if [ -n "$force_color_prompt" ]; then
    if [ -x /usr/bin/tput ] && tput setaf 1 >&/dev/null; then
        # We have color support; assume it's compliant with Ecma-48
        # (ISO/IEC-6429). (Lack of such support is extremely rare, and such
        # a case would tend to support setf rather than setaf.)
        color_prompt=yes
    else
        color_prompt=no
    fi
fi
```

Sofern die Farb-Tabelle nicht vorhanden ist, muss diese nach dem letzten der beiden „fi“ – letzteres markiert das Ende der Anweisung im Shell-Script – importiert werden. Dafür den folgenden Text ans Ende des Shellscripts .bashrc kopieren:

```
#ANSI color codes
RS="\[\033[0m]" # reset
HC="\[\033[1m]" # hicolor
UL="\[\033[4m]" # underline
INV="\[\033[7m]" # inverse background and foreground
FBLK="\[\033[30m]" # foreground black
FRED="\[\033[31m]" # foreground red
FGRN="\[\033[32m]" # foreground green
FYEL="\[\033[33m]" # foreground yellow
FBLE="\[\033[34m]" # foreground blue
FMAG="\[\033[35m]" # foreground magenta
```

Nach dem Einfügen der Farbtabelle sollte sich folgender Abschnitt im Shellsript .bashrc befinden:



```
# ANSI color codes
RS="\[\033[0m]" # reset
HC="\[\033[1m]" # hicolor
UL="\[\033[4m]" # underline
INV="\[\033[7m]" # inverse background and foreground
FBLK="\[\033[30m]" # foreground black
FRED="\[\033[31m]" # foreground red
FGRN="\[\033[32m]" # foreground green
FYEL="\[\033[33m]" # foreground yellow
FBLE="\[\033[34m]" # foreground blue
FMAG="\[\033[35m]" # foreground magenta
FCYN="\[\033[36m]" # foreground cyan
FWHT="\[\033[37m]" # foreground white
BBLK="\[\033[40m]" # background black
BRED="\[\033[41m]" # background red
BGRN="\[\033[42m]" # background green
BYEL="\[\033[43m]" # background yellow
BBLE="\[\033[44m]" # background blue
BMAG="\[\033[45m]" # background magenta
BCYN="\[\033[46m]" # background cyan
BWHT="\[\033[47m]" # background white
```

Nach Lust & Laune könnt ihr mit diesen Farbcodes nun euer Prompt einfärben – und zwar in der folgenden Zeile:

```
PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
```


Unter Raspbian ist übrigens keine Kolorierung des Prompts im root-Modus vorgesehen, folgende Zeile kann dennoch in das Shell-Script eingefügt werden, um das Prompt des root-Users rot einzufärben:

```
echo "PS1='${debian_chroot:+($debian_chroot)}\[\033[01;31m\]\u@\h\[\033[00m\] \[\033[01;34m\]\w \$\[\033[00m\] "' >> /etc/profile
```

Der Code für mein Prompt sieht übrigens so aus – die Option für ein farbloses Prompt bleibt nach der else-Option erhalten:

```
if [ "$color_prompt" = yes ]; then
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u\[\033[01;37m\]@\[\033[01;34m\]space\[\033[01;35m\]]ship\[\033[01;37m\]:\[\033[01;00m\]\w\[\033[00m\]\$ '
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
```

Copy this and insert it into the shell-script to have the same Prompt!

Am Ende überprüfen und aufrufen mit

```
sudo su –
```

2. SOUND | ALSA-Soundsystem

USB-Boxen über 3,5mm Audio-Klinke anschließen & Buchse aktivieren (Klinke) mit

```
sudo raspi-config
```

→ Advanced Options

→ Force 3,5 mm jack

```
sudo amixer cset numid=3 1
```

ODER für HDMI (für uns hier **nicht** relevant!) mit

```
sudo amixer cset numid=3 1
```

0 = auto

1 = Klinke analog

2 = HDMI

ALSA ist das Linux-Soundsystem, es stellt Kernelmodule für den Sound-Adapter bereit; dieser wird beim Booten normalerweise erkannt, dennoch kann die gesonderte Installation des *ALSA driver configuration file* und der *Utilities for configuring and using* nötig sein mit

```
sudo apt-get install alsa-base
```

```
sudo apt-get install alsa-utils
```

Test der **Lautsprecher** (Ansage):

```
speaker-test -c2 -t wav
```

(beenden mit STRG-C)

Test mit **Sinuston**:

```
speaker-test -t sine -f 440 -c 2 -s 1
```

```
speaker-test -t sine -f 440 -c 2 -s 2
```

(beenden mit STRG-C)

Test über die Schnittstelle "front" (in der Regel die Standardlautsprecher) mit **Rosa Rauschen** über zwei Kanäle:

```
speaker-test -c 2 -D front
```

(beenden mit STRG-C)

Man kann die Soundkarte auch direkt ansprechen (Devicenummer unter `plughw:x,x` angeben)

```
speaker-test -c 2 -D plughw:1,0
```

(beenden mit STRG-C)

Wenn die Soundausgabe nicht funktioniert und man ermitteln will, wo der Fehler liegt, empfiehlt sich

```
speaker-test 1.0.25
```

Das Programm **aplay** erkennt den Status von ALSA und kann Audio-Dateien abspielen. Das Default-Format ist wav, für die anderen von aplay akzeptierten Formate muss mit dem Parameter `-t` der Dateityp eingestellt werden (voc, wav, raw oder au). Mit der Option `-v` können Informationen über die Arbeit des Programms ausgegeben werden

```
aplay -v -D front test.wav
```

Das Programm kann Sounddateien auch direkt über die Soundkarte ausgeben:

```
aplay -D front test.wav
```

amixer ist ein Tool zur Soundverwaltung im ALSA-Paket. Der höchste Wert liegt bei 400, der niedrigste bei -10239. Da wir die Lautstärke über die am Raspi angeschlossenen USB-Boxen regeln wollen, wird die Maximallautstärke eingestellt mit

```
amixer cset numid=1 400
```

Werden Kopfhörer verwendet, muss die Lautstärke reduziert werden. Bei negativen Werten Werte – z. B. -100 – ist ein Trick notwendig, damit das Programm diese nicht als unbekannte Parameter zurückweist. Mit "--" wird dem Programm mitgeteilt, dass keine Parameter mehr folgen:

```
amixer cset numid=1 -- -1000
```

Die aktuellen Werte können abgerufen werden mit

```
amixer contents
```

3. AUDIO-AUFNAHME | arecord

USB-Mikrofon über USB am Raspberry anschließen, dann mit **arecord** testen. Die Syntax für die Audio-Aufzeichnung lautet

```
arecord -f [Format] -D [Hardware] -d [Dauer]
```

Vorsicht: Files sollten in Kleinschreibung mit „_“ anstelle eines Abstands und möglichst kurz benannt werden, das erleichtert die Auffindbarkeit!

Beispiel für eine 20 Sekunden lange Aufnahme mit dem USB-Mikrofon in CD-Qualität:

```
arecord -f cd -D plughw:1,0 -d 10 test.wav
```

Beispiel für eine 20 Sekunden lange Aufnahme mit dem USB-Mikrofon in Stereo-Qualität:

```
arecord -f cd -t wav -D plughw:1,0 test.wav
```

Standard-Audioquelle in Mono, 8000 Samples pro Sekunde, 8 Bit pro Aufnahme, startet alle 60 Sekunden eine neue Datei:

```
arecord -t wav --max-file-time=60 -D plughw:1,0 test.wav
```

Aufnahme in Stereo mit CD-Qualität, jede Viertelstunde wird eine neue Datei erzeugt:

```
arecord -f cd -t wav --max-file-time 900 -D plughw:1,0 test.wav
```

(end with STRG-C)

Setzt man den Parameter bei -d auf Null (-d 0), wird so lange aufgenommen, bis das Programm mit STRG-C beendet wird.

Um mitzuhören, was über das Mikrofon aufgenommen wird, kann der **Sound von arecord per pipe an aplay übergeben** werden. Es tritt dabei aber eine leichte Verzögerung zwischen Aufnahme und Wiedergabe auf:

```
arecord -t wav -D plughw:1,0 | aplay -D plughw:1,0
```

oder

```
arecord -t wav -D sysdefault:CARD=1 | aplay -D sysdefault:CARD=1
```

Nur der User* root und der die User* in der Gruppe audio können direkt auf die Soundkarte zugreifen, deshalb den User* candy in die Gruppe audio eintragen mit

```
sudo gpasswd -a sugar audio
```

Wiedergabe der Aufnahme mit **aplay**

```
aplay test.wav
```

Wiedergabe der Aufnahme mit **omxplayer**

```
omxplayer -p -o hdmi test.wav
```

ODER

```
omxplayer -p -o hdmi /home/sugar/sounds/test.wav
```

(beenden with q)

Die Aufnahme mit **omxplayer** funktioniert auch mit dem command

```
arecord test.wav -D sysdefault:CARD=1
```

Zum Schneiden von Tonaufnahmen kann **audacity** verwendet werden

```
sudo apt-get install audacity
```

Für die Wiedergabe von mp3-Files empfiehlt sich immer noch der altbewährte VLC-Player

```
sudo apt-get install vlc vlc-plugin-pulse mozilla-plugin-vlc  
sudo apt-get update
```

Die Default-Wiedergabe und VLC für audio-Files lautet

```
cvlc test.mp3
```

Ein anderer, mp3-kompatibler Linux-Player ist mp123

```
sudo apt-get install mp123
```

File abspielen mit

```
mp123 test.mp3
```

4. VIDEO | fswebcam

USB-Webcam über USB an den Raspbian anschließen und Ausgang des Geräts ermitteln mit

```
lsusb
```

Installation des Programms **fswebcam**

```
sudo apt-get install fswebcam
```

Achtung: fswebcam funktioniert nicht gleichzeitig mit motion!
Wenn du motion aktiviert hast, dann deaktiviere es mit

```
sudo service motion stop
```

Bildordner am pi einrichten und ins Verzeichnis wechseln, z.B. so :

```
cd /home/sugar/schreibtisch  
mkdir pi_pics  
cd /home/sugar/schreibtisch/pi_pics
```

Command zum Erstellen eines Bildes von der USB-Webcam mit der Auflösung 640x480

```
fswebcam --no-banner -r 640x480 image.jpg
```

kleinere Auflösung (hier: 358x288)

```
fswebcam --no-banner -r 358x288 image1.jpg
```

Empfehlenswerter Picture Viewer unter Raspbian Jessie:

```
sudo apt-get install ristretto
```

Bild öffnen mit

```
ristretto image.jpg
```

5. BUS-ERWEITERUNGEN | I2C & SPI

Der GPIO-Header des Raspberry Pi 3 verzeichnet insgesamt 40 Anschlüsse, die auch GPIO-Pins (*General Purpose Input Output Pins*) genannt werden. 2 davon sind I2C-Anschlüsse, 5 SPI-Devices. I2C (*Inter-Integrated Circuit*) und SPI (*Serial peripheral Interface*) sind Module zur Steuerung von ICs (=integrated circuits, dt. Schaltkreise) am Breadboard (z. B: Mikrokontroller, EEPROMs oder Digital-Analog-Wandler). Um sie nutzen zu können, müssen wir die dazugehörigen Pins aktivieren. Dies ermöglicht die Kommunikation der ICs am Breadboard mit den GPIOs am Raspberry Pi.

I2C & SPI sind Teil der Bussysteme des Raspberry Pi , kurz: **BUS** (*Binary Unit System*). Davon gibt es 3 weitere:

UART = *Universal Asynchronous Receiver/Transmitter* (serielle Schnittstelle)

I2S = *Integrated Interchip Sound* (für digitale Audio-Dateien)

1 Wire – für einfache Sensoren, zB. für die Temperaturmessung

durch Bussysteme können größere Datenmenge übertragen werden als über die normalen GPIOs. Abhängig vom verwendeten System werden die Daten seriell (hintereinander) oder parallel (gleichzeitig) übertragen.

In einem ersten Schritt müssen die GPIO-Pins aktiviert werden. Dafür benötigen wir das *Python Development Toolkit* (python-dev), dann die *Rpi.GPIO* (python-rpi)

```
sudo apt-get install python-dev  
sudo apt-get install python-rpi.
```

I2C – *Inter-Integrated Circuit*

Zur Installation von I2C benötigen wir die Python-Library für Bussysteme

```
sudo apt-get install python-smbus  
sudo apt-get install i1c-tools
```

Die Kernel-Unterstützung vom I2C wird konfiguriert unter

```
sudo raspi-config
```

im Menü:

- interfacing options
- i2c
- ARM i2c Interface is enabled
- save

Reboot mit

```
sudo reboot
```

Nach dem Reboot Installation überprüfen mit

```
lsmod | grep i2c_
```

Ausgabe:

```
i2c_bcm2835      7167 0
i2c_dev         6913 0
```

Es müssen 2 Devices angezeigt werden, i2c_bcm2835 ist eines davon!

Breadboard verbinden, IC anstecken und I2C aufrufen mit

```
sudo i2cdetect -y 1
```

Beispiel für Ausgabe:

```
sugar@raspberrypi:~$ sudo i2cdetect -y 1
[sudo] password for sugar:
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50: 50  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```


SPI – Serial peripheral Interface

Die Kernel-Unterstützung von SPI wird konfiguriert unter

```
sudo raspi-config
```

Menü

```
→interfacing options  
→ Spi  
→ Spi Interface is enabled  
→ save
```

Reboot mit

```
sudo halt
```

Nach dem Reboot Installation überprüfen mit

```
lsmod | grep spi_
```

SPI ist nun aktiviert, dies wird überprüft mit

```
ls -l /dev/spidev*
```

Auch in diesem Fall werden 2 Devices angezeigt, jeweils eines für einen der beiden SPI-Busse!

5. LIBRARIES

Empfehlungen:

- a. Wiring Pi Library (C-Library für Raspbian)**
- b. Botbook (Python-Library zur Sensorsteuerung)**
- c. Squid (Python-Library für RGB-LED-Farbmischungen)**

a. Wiringpi

Ordner für die Library anlegen mit

```
mkdir wiringpi
```

Der Import der Wiringpi-Library erfolgt über git-core mit

```
sudo apt-get install git-core
git clone git://git.drogon.net/wiringPi
```

Zur Installation der Library reicht die Ausführung des Scripts build. Dafür ins Verzeichnis gehen

```
cd /home/pi/my_c/wiringPi
```

und im Terminal Programm starten

```
./build
```

anschließend die Installation überprüfen mit

```
gpio -v
gpio readall
```

Die Files der WiringPI-Library werden mithilfe von Make-Files kompiliert, Beispiel:

```
make
make piglow
./piglow
```

```
sugar@raspberrypi:~$ gpio readall
```

Pi 3											
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	
		3.3v			1	2		5v			
2	8	SDA.1	ALTO	1	3	4		5v			
3	9	SCL.1	ALTO	1	5	6		0v			
4	7	GPI0. 7	IN	0	7	8	0	IN	TxD	15	14
		0v			9	10	1	IN	RxD	16	15
17	0	GPI0. 0	IN	0	11	12	0	ALTO	GPI0. 1	1	18
27	2	GPI0. 2	IN	0	13	14	0		0v		
22	3	GPI0. 3	IN	0	15	16	0	IN	GPI0. 4	4	23
		3.3v			17	18	0	IN	GPI0. 5	5	24
10	12	MOSI	ALTO	0	19	20		0v			
9	13	MISO	ALTO	0	21	22	0	IN	GPI0. 6	6	25
11	14	SCLK	ALTO	0	23	24	1	OUT	CE0	10	8
		0v			25	26	1	OUT	CE1	11	7
0	30	SDA.0	IN	1	27	28	1	IN	SCL.0	31	1
5	21	GPI0.21	IN	1	29	30		0v			
6	22	GPI0.22	IN	1	31	32	0	IN	GPI0.26	26	12
13	23	GPI0.23	IN	0	33	34		0v			
19	24	GPI0.24	ALTO	0	35	36	0	IN	GPI0.27	27	16
26	25	GPI0.25	IN	0	37	38	0	ALTO	GPI0.28	28	20
		0v			39	40	0	ALTO	GPI0.29	29	21

```
-----Pi 3-----
```

BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM
-----	-----	------	------	---	----------	---	------	------	-----	-----

BOARD – **GPIO.BOARD** ist ident mit der physikalischen Nummerierung am Raspberry Pi, lineare Nummerierung von links nach rechts -> relevant für den Import der jeweiligen Library im Programmcode

BCM – **GPIO.BCM** bezieht sich auf die GPIO-Angaben nach dem "Broadcom SOC channel", ident mit der äußeren, nicht-linearen Nummerierung in der Grafik auf Seite 28 (die allgemeinen Pins ohne besondere Funktion sind grün markiert: GPIO04, GPIO17, GPIO18, GPIO27, GPIO22, GPIO23, GPIO24, GPIO25, GPIO5, GPIO6, GPIO12, GPIO13, GPIO19, GPIO16, GPIO26, GPIO26, GPIO20, GPIO21).

