

Raspberry Pi

Setup

Download Raspbian: <https://www.raspberrypi.org/downloads/raspbian/>

Write image with dd to the SD-Card:

```
umount /dev/mmcblk0p1  
sudo dd bs=10M if=2017-09-07-raspbian-stretch-lite.img of=/dev/mmcblk0
```

oder auch mit Fortschrittsanzeige:

```
sudo apt install pv  
sudo dd bs=10M if=RuneAudio_rpi2_rp3_0.4-beta_20160321_2GB.img | pv -s 2100M  
| sudo dd of=/dev/sdc
```

wobei pv die Imagegröße mitgegeben werden kann, und man somit eine Zeitschätzung erhält.

Auflisten der Installierten Pakete

Möchte man wissen, was alles installiert wurde nach der frischen Installation von Raspbian, kann man sich das ausgeben lassen:

```
cat /var/log/apt/history.log | egrep '^(Start-Date:|Commandline:)' | grep -v  
aptdaemon | egrep '^Commandline:' | egrep 'install'
```

Headless with UART

add this line to **config.txt**

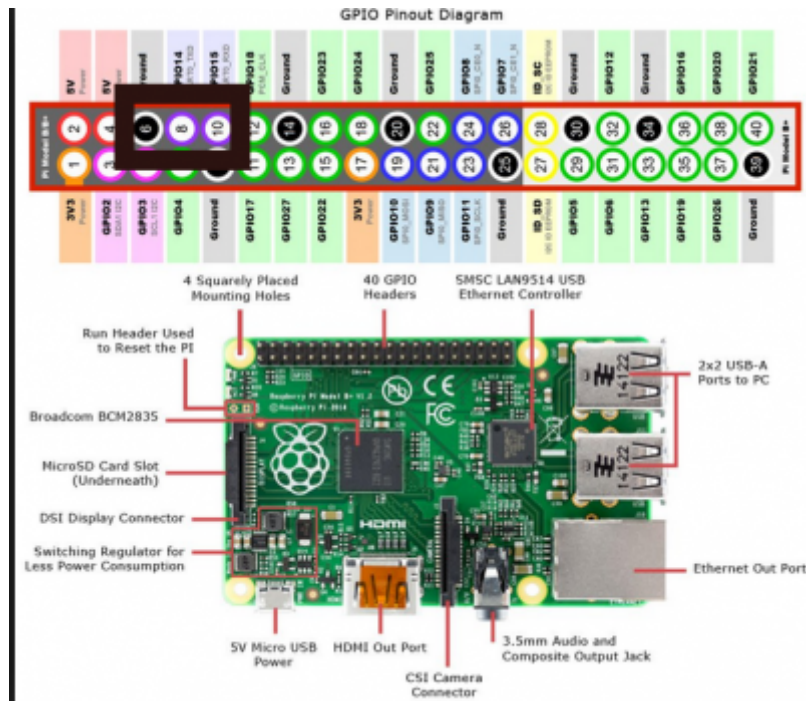
```
enable_uart=1
```

As terminal **minicom** is a very good choice, since it sends each character instantly, so you have the feeling as running linux on the terminal-machine.

```
sudo minicom -s
```

erlaubt das diekte verändern der systemweiten standard einstellungen.

Baudrate: 115200 8N1 Steckverbinder: wie MKZ Standard:



Headless with WiFi

First: create a file in boot with this content:

[wpa_supplicant.conf](#)

```
country=AT
update_config=1
ctrl_interface=/var/run/wpa_supplicant

network={
    ssid="Singervilla-iNet-2.4G"
    psk="myStrongPassword"
}
```

Second: create a file ssh in boot.

Then insert the SD card into the Raspi and start to scan for the Raspberry's IP-Address with

```
nmap -T5 -sn 10.0.1.0-255
```

or with

```
arp -a
```

Then connect to it with e.g.

```
ssh pi@10.0.1.169
```

Headless with Ethernet

In der Datei /etc/dhcpd.conf kann man eine statische IP festlegen:

```
# Example static IP configuration:
interface eth0
static ip_address=10.0.0.115/24
#static ip6_address=fd51:42f8:caae:d92e::ff/64
static routers=10.0.0.1
static domain_name_servers=10.0.0.100 8.8.8.8
```

SSH auf Raspbian Lite

for raspbian lite we have to make a file 'ssh' on the boot partition.

<https://raspberrypi.stackexchange.com/questions/40689/cannot-connect-to-raspbian-jessie-lite-but-to-raspbian-jessie#58455>

EDIT: scheint seit Beginn 2018 nicht mehr so zu funktionieren. Alternativ kann man sich per UART verbinden und mit raspi-config ssh einschalten.

EDIT2: geht doch. mit kopieren der datei config.txt und umbenennen auf ssh ¹⁾

Add a Service Module

To execute a programm on each boot automatically, here is a good tutorial:

<http://www.diegoacuna.me/how-to-run-a-script-as-a-service-in-raspberry-pi-raspbian-jessie/>

```
cd /lib/systemd/system/
sudo nano hello.service
```

[hello.service](#)

```
[Unit]
Description=Hello World
After=multi-user.target

[Service]
Type=simple
WorkingDirectory=/home/karl/hello-service
ExecStart=/usr/bin/python /home/pi/hello_world.py
Restart=on-abort

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

Start the Service

```
sudo chmod 644 /lib/systemd/system/hello.service
chmod +x /home/pi/hello_world.py
sudo systemctl daemon-reload
sudo systemctl enable hello.service
sudo systemctl start hello.service
```

Inspect the Logs

```
journalctl -u hello
```

Image Erstellen

Ein Image wird ganz einfach per dd erstellt. Vorher sollte es aber z.B. mit **gparted** auf die notwendige Größe verkleinert werden. Dann Image erstellen und mit truncate wieder verkleinern:

```
truncate --size=2782920704 image.img
```

Dabei wird die Größe mittels

```
fdisk -l image.img
```

berechnet: 512 Bytes/Sector * (5435391+1) Sectors = 2782920704 Bytes

Verkleinern

Wurde das Image bereits erstellt und hat z.B. 16GB, dann kann es recht einfach so verkleinert werden:

```
sudo modprobe loop
sudo losetup -f
sudo losetup /dev/loop0 myimage.img
sudo partprobe /dev/loop0
sudo gparted /dev/loop0
sudo losetup -d /dev/loop0
```

Quelle: <https://softwarebakery.com/shrinking-images-on-linux>

PIGPIO

A very great library for hardware access of the IO-pins:

<http://abyz.me.uk/rpi/pigpio/index.html>

Hints

- You have to run the compiled program as super user (sudo ./myprogram)
- Not all pins can be used for IO.

C/C++ Programming

Architecture Specific Code

The architecture can be detected with this symbol:

```
#ifdef __arm__  
    // do some raspberry specific stuff  
#endif
```

POSIX on Linux

Using the POSIX library helps a lot on various tasks timing, interprocess communication, shared memory, threads and processes and many more.

For simple projects these can be very handy:

- `sleep()` and `usleep()`
- `poll()` from `<poll.h>` is great for reading/writing (device-) files with timeout.

For an overview please have a look at Wikipedia: [C POSIX Library](#)

A comprehensive guide (700 pages) can be downloaded here: [Posix.4 Programmers Guide](#)

Simple Makefile with PIGPIO

This makefile compiles all C++ files within one folder. It does a complete recompile on each run.

Makefile

```
all:  
    g++ -o myprogram -Wall -pthread -lpigpio -lrt -I. *.cpp  
  
clean:  
    rm myprogram
```

Realtime Kernel

- <https://www.get-edi.io/Real-Time-Linux-on-the-Raspberry-Pi/>
- <https://github.com/lueschem/edi-pi#creating-a-cross-development-lxd-container>
- <https://www.get-edi.io/>
- <https://www.raspberrypi.org/documentation/linux/kernel/building.md>
- <https://medium.com/@patdhlk/realtime-linux-e97628b51d5d>
- <https://lemariva.com/blog/2019/09/raspberrypi-pi-4b-preempt-rt-kernel-419y-performance-test>

Auf dem Raspberry Pi, mit Raspberry OS vom 11. Jan. 2021 ²⁾:

```
sudo apt update
sudo apt upgrade
sudo apt install git bc bison flex libssl-dev make
git clone --depth=1 --branch rpi-4.19.y-rt
https://github.com/raspberrypi/linux
```

Configure Kernel ³⁾:

```
sudo apt install libncurses5-dev
cd linux
make menuconfig
```

Hierbei wurde jedoch weiters nichts konfiguriert.

```
make bcm2711_defconfig
make -j4 zImage
make -j4 modules
make -j4 dtbs
```

Dauert etwa 1h15min.

```
sudo make -j4 modules_install
sudo make -j4 dtbs_install
sudo cp arch/arm/boot/dts/*.dtb /boot/
sudo cp arch/arm/boot/dts/overlays/*.dtb* /boot/overlays/
sudo cp arch/arm/boot/dts/overlays/README /boot/overlays/
sudo cp arch/arm/boot/zImage /boot/$KERNEL-rt.img
```

[english](#), [software](#), [raspberrypi](#), [c++](#), [linux](#), [technical](#)

¹⁾

vielleicht gabs oben ein Problem mit den Dateiberechtigungen...

²⁾

<https://www.raspberrypi.org/documentation/linux/kernel/building.md>

³⁾

<https://www.raspberrypi.org/documentation/linux/kernel/configuring.md>

From:

<http://www.zeilhofer.co.at/wiki/> - **Verschiedenste Artikel von Karl Zeilhofer**

Permanent link:

<http://www.zeilhofer.co.at/wiki/doku.php?id=raspberry>

Last update: **2022/03/15 09:04**

