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.

Headless with UART

add this line to **config.txt**

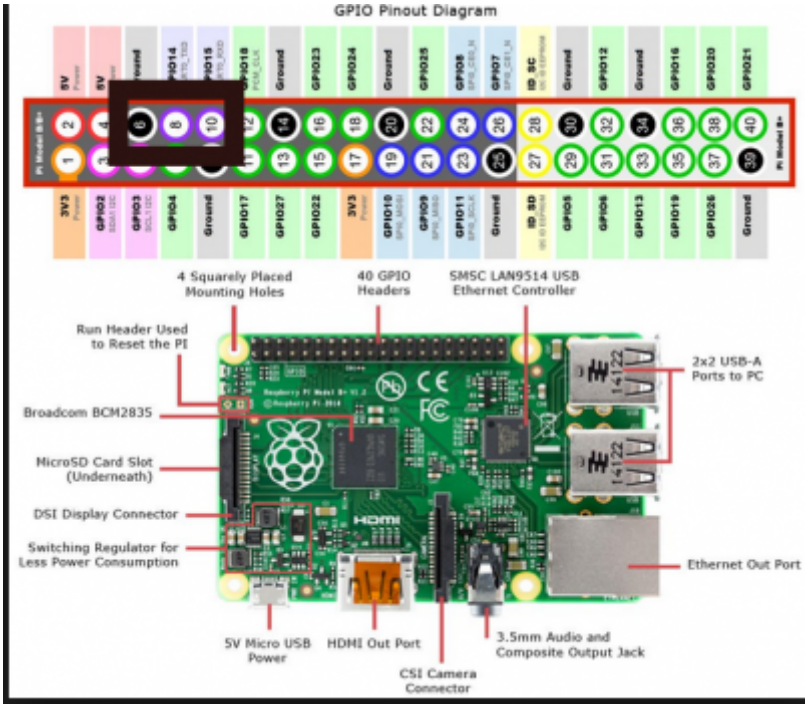
```
enable_uart=1
```

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

https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md

Headless with Ethernet

In der Datei /etc/dhcpcd.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

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
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
```

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
```

english, software, raspberry, c++, linux, technical