# Inhaltsverzeichnis

# Raspi im QEmu ausführen

(Dies ist eine ältere Version, es gibt jetzt auch eine Fortsetzung)

Eigentlich ist es ja einfach:

- Raspian- Image von https://www.raspberrypi.org/downloads/raspbian/ holen
- passenden Kernel von https://github.com/dhruvvyas90/qemu-rpi-kernel/
- Qemu installieren mit `sudo apt-get install qemu-system-arm`

So ist jedenfalls die Theorie. Damit's dann wirklich funktioniert, muß man noch ein wenig das Bootverhalten des Image frisieren:

Die Partition im „Inneren“ des Imagefiles als Loopback mounten (geht in der Art angeblich erst ab Ubuntu 16.04, man beachte den -P switch)

```
sudo losetup -f --show -P 2016-09-23-raspbian-jessie-lite_modified.img
```

der obige Befehl zeigt dann auch, welches Loop- Device erzeugt wurde:

```
/dev/loop0
```

Die zweite Partition des Images mounten:

```
sudo mkdir /mnt/rpi
sudo mount /dev/loop0p2 /mnt/rpi/
```

Ein paar Dateien editieren

```
# die Zeile in der Datei mit # auskommentieren
sudo vim /mnt/rpi/etc/ld.so.preload
# Alle Zeilen auskommentieren, die mit "/dev/mmcblk*" beginnen
sudo vim /mnt/rpi/etc/fstab
```

Alles wieder abmelden:

```
sudo umount /mnt/rpi
sudo losetup -d /dev/loop0
```

und jetzt kann's losgehen. In der Bash starten:

```
 qemu-system-arm -kernel kernel-qemu-4.4.13-jessie -cpu arm1176 -m 256 -M
versatilepb -no-reboot -serial stdio -append "root=/dev/sda2 panic=1
rootfstype=ext4 rw" -hda 2016-09-23-raspbian-jessie-lite_modified.img -net
user,hostfwd=tcp::10022-:22 -net nic
```

oder man bastelt sich eben schnell einen symbolischen Link auf den gerade aktuellen Kernel

```
ln -s kernel-qemu-4.4.13-jessie  actual-kernel
```

und überläßt das Parameter- Handling einem kleinen Script

startRaspiQemu.sh

```
#!/bin/bash
# Bash Menu Script Example

PORT=${2:-10022}
echo Port $PORT
PS3='Please enter your starting option: '
options=("with root bash" "normal" "Quit")
select opt in "${options[@]}"
    do
        case $opt in
            "with root bash")
                echo "you chose root bash"
                # start with root bash
                qemu-system-arm -kernel actual-kernel -cpu arm1176 -m
256 -M versatilepb -no-reboot -serial stdio -append "root=/dev/sda2
panic=1 rootfstype=ext4 rw init=/bin/bash" -hda $1 -net
user,hostfwd=tcp::$PORT-:22 -net nic
            ;;
            "normal")
                echo "you chose normal"
                # start normal
                qemu-system-arm -kernel actual-kernel -cpu arm1176 -m
256 -M versatilepb -no-reboot -serial stdio -append "root=/dev/sda2
panic=1 rootfstype=ext4 rw" -hda $1 -net user,hostfwd=tcp::$PORT-:22 -
net nic
            ;;
            "Quit")
                break
            ;;
            *) echo invalid option;;
        esac
    done
```

und wenn man dann auch noch wissen muss, wie man den verd###ten Mauszeiger wieder aus dem Fenster bekommt: Das ist linke Seite Strg+Alt..

Der Standard- User ist „pi“, das Passwort „raspberry“, und wer sich nach einer halben Stunde immer noch nicht einloggen konnte, dem sei verraten, das der Qemu eine amerikanische Tastatur fährt, das Passwort auf einer deutschen Tastatur also „raspberrz“ geschrieben wird. Nicht, das ich da natürlich nicht gleich drauf gekommen bin....

Mit dem `-net..` Parameter des Qemu wird übrigens Portforwarding eingerichtet, so dass man sich auch per

```
ssh pi@localhost -p10022
```

anmelden kann

## Das Image vergrößern

Im Original sind auf dem Image nur gute 200MB frei. So fügt man mal komfortable 2G hinzu:

- qemu-img.exe resize myImageFile.img +2G

in raspian

- `sudo fdisk -l` you have the list of normally 2 partitions, second beginning 122880
- `sudo fdisk /dev/sda`

in fdisk

- „d“ to delete a partition
- „2“ choose the second

the 2nd partition is now deleted

- „n“ to create partition

accept default primary, second partition but entry the good beginning address (122880)

- if doing mistake quit without saving change with „q“
- otherwise „w“ will write the new table to disk

in raspian

- `sudo reboot`

after Reboot

- `sudo resize2fs /dev/sda2`
- verify with df diskfree command

## Vom Qemu rüber auf den echten Raspi

Wenn man seine Anpassungen im Emulator soweit fertig hat, möchte man sie ja vielleicht auch gerne mal im echten Raspi laufen lassen. Dazu macht man folgendes:

1. eine Kopie des fertigen Image anlegen
2. die **Kopie** im QEmu starten
3. die oben beschriebenen Änderungen in /etc/fstab und /etc/ld.so.preload wieder rückgängig machen. **Achtung:** Die `preload` Datei als letztes, denn danach verweigert irgendwie die bash den Dienst.
4. QEmu beenden
5. die gerade geänderte Image- Kopie kann jetzt auf die Speicherkarte gebrannt und auf dem Raspi gebootet werden.

# SocketCan installieren

(schamlos abgekupfert von:
http://tiqni.com/raspberry-pi/raspbian-built-in-can-controller-for-raspberry-pi)

The latest Rapsbian versions have now a built-in support for Microchip CAN Controller MCP251x, this makes things more easy than before, we don’t need anymore to compile and install modules manually for every new version.

## Get latest Raspbian

Download latest Raspbian version here : https://www.raspberrypi.org/downloads/, and/or update your Raspberry Pi:

```
sudo su
apt-get update
apt-get upgrade
apt-get autoremove
apt-get install can-utils
reboot
```

## Enable SPI module

In Advanced Options of Raspi-config, enable SPI and load it by default at boot:

```
sudo su
mount /dev/sda1 /boot
raspi-config
```

## Configure SPI Module

Overlay SPI configuration at boot configuration by adding below parameters:

```
sudo nano /boot/config.txt
dtparam=spi=on

# its a 12Mhz Quarz, so it needs to be 12000000 instead of 16000000
dtoverlay=mcp2515-can0,oscillator=12000000,interrupt=22
dtoverlay=mcp2515-can1,oscillator=12000000,interrupt=25
dtoverlay=spi-bcm2835-overlay
dtoverlay=spi-dma-overlay
```

Fix Me! do we need this?

```
# modprobe spi-bcm2708
```

```
# modprobe can
# modprobe can-dev
# modprobe can-raw
# modprobe can-bcm
# modprobe mcp251x
# ip link set can0 up type can bitrate 500000
```

## Set-up CAN interface

You can then configure and use CAN interface as usual:

```
sudo ip link set can0 up type can bitrate 250000
```