

# Setup

Hans Buchmann FHNW/IME

18. September 2014

# Terminologie

Host der Entwicklungsrechner Notebook

Target **RaspberryPi**

# Ziel

Verbindung: Host-Target

## Host GNU/Linux

- ▶ als virtuelle Maschine
- ▶ native
- ▶ Distribution: Ubuntu

## Target

- ▶ [www.raspberrypi.org/downloads](http://www.raspberrypi.org/downloads)
  - ▶ eines auswählen
  - ▶ Vorschlag: *Arch Linux*

# Image

## Für RaspberryPi

- ▶ Bootcode
- ▶ Kernel
- ▶ UNIX

# Die Files

**Original** [http://downloads.raspberrypi.org/arch\\_latest](http://downloads.raspberrypi.org/arch_latest)

**Modifikation** <http://sourceforge.net/projects/fhnw-tinl/files>

# Inhalt

## Partitionen

### ► `fdisk -l image`

Disk ArchLinuxARM-2014.06-rpi.img: 1.8 GiB, 1960837120 bytes, 3829760 sectors

Units: sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disklabel type: dos

Disk identifier: 0x417ee54b

Device	Boot	Start	End	Sectors	Size	Id	Type
ArchLinuxARM-2014.06-rpi.img1		2048	186367	184320	90M	c	W95 FAT32 (LB)
ArchLinuxARM-2014.06-rpi.img2		186368	3667967	3481600	1.7G	5	Extended
ArchLinuxARM-2014.06-rpi.img5		188416	3667967	3479552	1.7G	83	Linux

## Partitionen vom Original

ArchLinuxARM-2014.06-rpi.img1 Boot Kernel

ArchLinuxARM-2014.06-rpi.img2 Extended

ArchLinuxARM-2014.06-rpi.img5 UNIX

Remark: Extended braucht es nicht

## Transfer → **RaspberryPi**

### Verschiedene Verfahren

**Original** direkt auf die SD-Card

- ▶ Einstellung von `ssh` auf dem **RaspberryPi**

**sd-card.gz** direkt auf die SD-Card

**Zusammenbau** aus `boot.tar.gz` und `target-root.tar.gz`

- ▶ als File auf dem Host
- ▶ direkt auf der SD-Card



# Original, sd-card.gz

## Der Befehl dd

- ▶ `dd if=image of=device`
  - ▶ *image* von `raspberry.org` oder `sd-card.gz`
  - ▶ typisches *device* `/dev/mmcblki`  $i = 0, 1..$

# Zusammenbau Host

alles ist ein File

1. erzeuge 'leeren' File *sd-card* 1GiB:

- ▶ `dd if=/dev/zero of=sd-card bs=bs count=count`
  - ▶ mit  $bs \times count = 1GiB$

2. formatiere *sd-card* wie wenn es eine 'richtige' SD-Card wäre:

- ▶ `fdisk sd-card`

*Offset in sector*

sd-card1	2048	133119	131072	64M	c W95 FAT32 (LBA)
sd-card2	133120	2097151	1964032	959M	83 Linux

3. fasse `sd-card{1|2}` als *device* auf:

- ▶ `losetup -o offset /dev/loopi sd-card`
  - ▶ mit *offset* in Bytes
  - ▶  $i = 0, 1$

4. erzeuge *Filesysteme* auf `/dev/loopi`

- ▶ `mkfs.fs /dev/loopi`
  - ▶ mit  $fs=vfat$  für `sd-card1` und  $fs=ext4$  für `sd-card2`

# Zusammenbau Host

## Fortsetzung

5. montiere *devices* /dev/loop*i*
  - ▶ `mount /dev/loopi mount-point`
6. kopiere Files
  - ▶ `tar -xzf part.tar.gz mount-point`
    - ▶ mit `part=boot|target-root`
7. autfräumen
  - ▶ `umount /dev/loopi`
  - ▶ `losetup -D`
8. kopieren auf SD-Card
  - ▶ `dd if=sd-card of=/dev/mmcblki`

## Zusammenbau SD-Card

### 1. formatiere SD-Card 2 Partitionen:

- Partition 1 Kleine 64MiB für Boot/Kernel
- Partition 2 Grosse > 1GiB für UNIX

### 2. erzeuge Filesystem

- Partition 1 vfat
- Partition 2 ext4

### 3. Kopiere

- Partition 1 boot.tar.bz2
- Partition 2 target-root.tar.bz2

# Der RaspberryPi Default

HDMI, USB

- ▶ Speisung per USB vom Host
- ▶ Bildschirm per HDMI
- ▶ Tastatur per USB

## Vorteil

- ▶ der Standard

## Nachteil

- ▶ braucht Bildschirm/Tastatur
  - ▶ nicht so viele vorhanden

## Via Netzwerk/Ethernet

### ssh

- ▶ Speisung via USB vom Host
- ▶ Verbindung zum Target via ssh

### Vorteil

- ▶ braucht keinen Bildschirm/Tastatur

### Nachteil

- ▶ etwas komplexere Konfiguration
  - ▶ vor allem im Schulnetz

Remark: Ist aber unser Ziel

# Die Tools für die Verbindung

- ▶ Wireshark für die Netzüberwachung
- ▶ Nmap für Portscans
- ▶ ifconfig für die Netzchnittstelle
- ▶ Ein DHCP Server z.B. dnsmasq

→ net-setup.txt

## Lokales Netzwerk

Host → Target

Host DHCP Server

Target Mit dem Host verbunden



## Via Schulnetz

- ▶ einfach einstecken

# Konfiguration SSH

## Server

- ▶ `ssh-keygen -t type -f /etc/ssh/ssh_host_type_key`
  - ▶ mit `type=rsa|dsas`

Remark: Auf dem *Target*

RS232

RS232→USB

TODO: Adapter