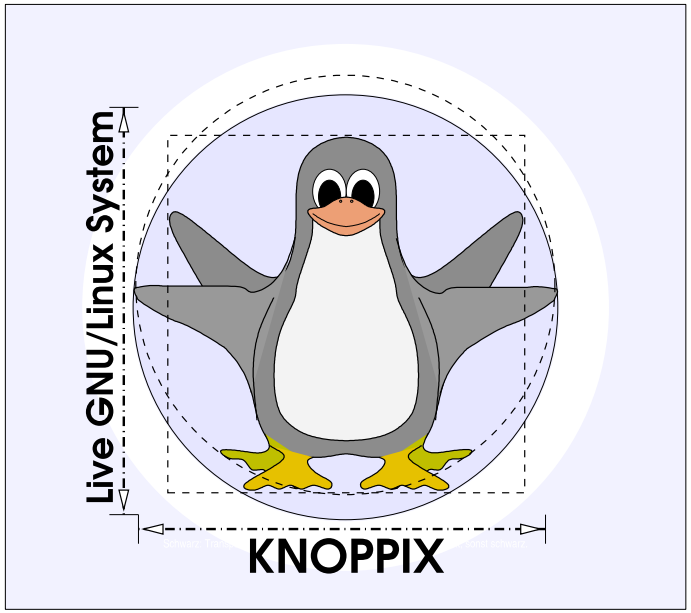
Knoppix
auf dem
Raspberry Pi



Prof. Dipl.-Ing. Klaus Knopper
<knoppix@knopper.net>

Einführung

- **KNOPPIX**: „Linux Live-System“, startet mit automatischer Hardwareerkennung von DVD/CD, USB-Stick, SD-Karte, PXE-Netzwerkboot

- **Raspberry Pi** ist nicht erst seit der „Power Edition“ 2 ein beliebter Mini-Computer für Steuerungszwecke und zum „normalen Arbeiten“, mit geringem Stromverbrauch (<= 3W) bei akzeptabler Leistung durch Multimedia-optimierten Chipsatz.

Idee:

→ KNOPPIX in einer ARM-Architektur-Version für alle Smartphones, Smartwatches, VR-Brillen, Tablets, Mini-Boards usw...

Problem: Es geht nicht! ;-)

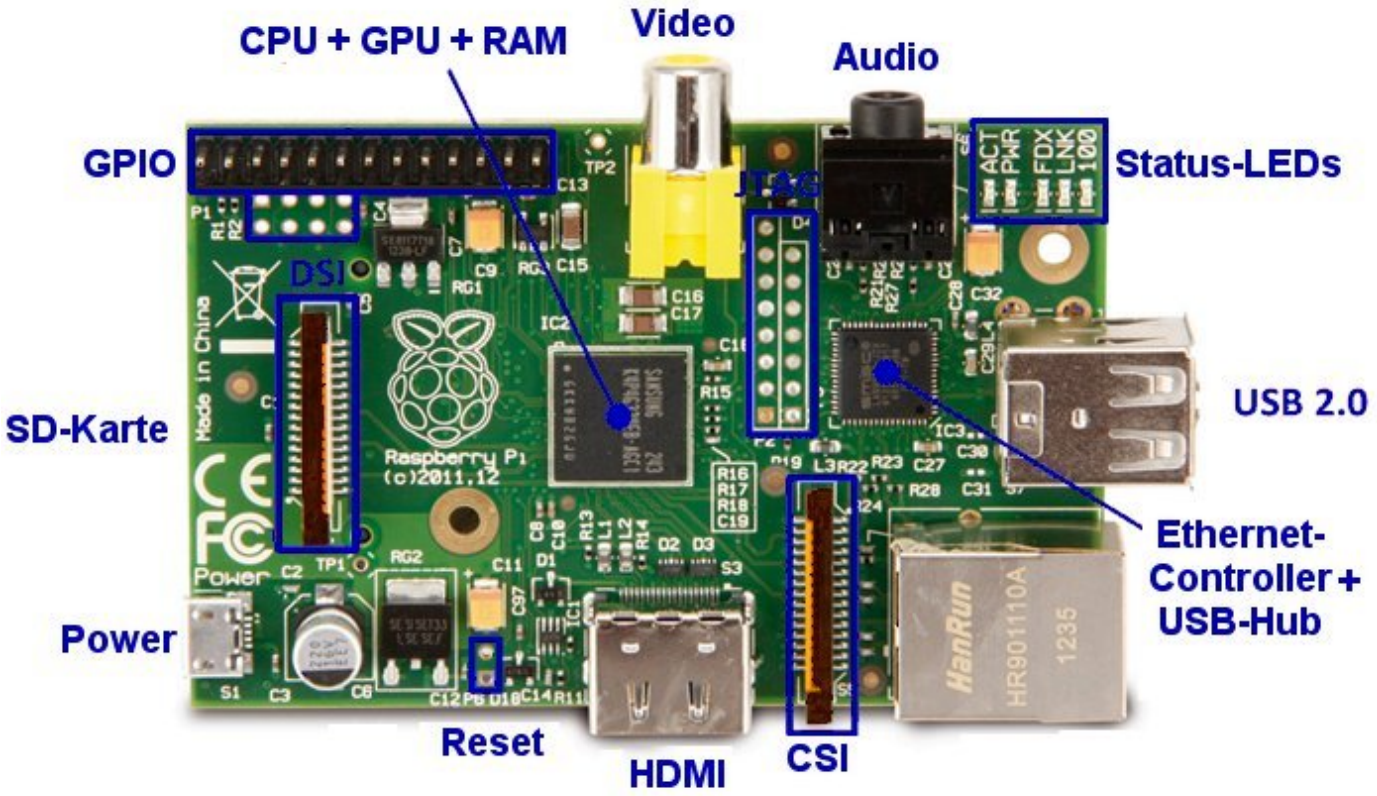
- Die ARM-Architektur unterstützt keine Hardware-Erkennung (standardmäßig kein BIOS, kein Plug&Play, kein PCI-Bus, unbekannte Adressen für sämtliche Peripherie)
- Jedes ARM-Board ist unterschiedlich (vergl. Cyanogenmod für `n+1` Geräte-Klassen mit `m+1` Revisionen)
- Für jede Hardware muss ein eigener Kernel + Module compiliert werden!

Oder doch?

- Beschränkt man sich auf ein bestimmtes Board, oder zumindest eine Klasse von kompatiblen Boards und CPUs, die mit dem gleichen Kernel lauffähig sind, so bleibt die Basiskonfiguration gleich, und eine Hardware-Erkennung kann zumindest auf der Ebene ***USB*** und ***onboard-Chipsätze mit bekannten Adressen*** stattfinden.
- „Knoppifizierung“ im Sinne von Optimierung auf schnelles Booten und resourcenoptimiertes Arbeiten auch auf „langsamer“ Hardware ist möglich.
- Gemeinsamkeiten: RasPi startet standardmäßig ohnehin von SD-Karte!

Raspberry Pi Modell B

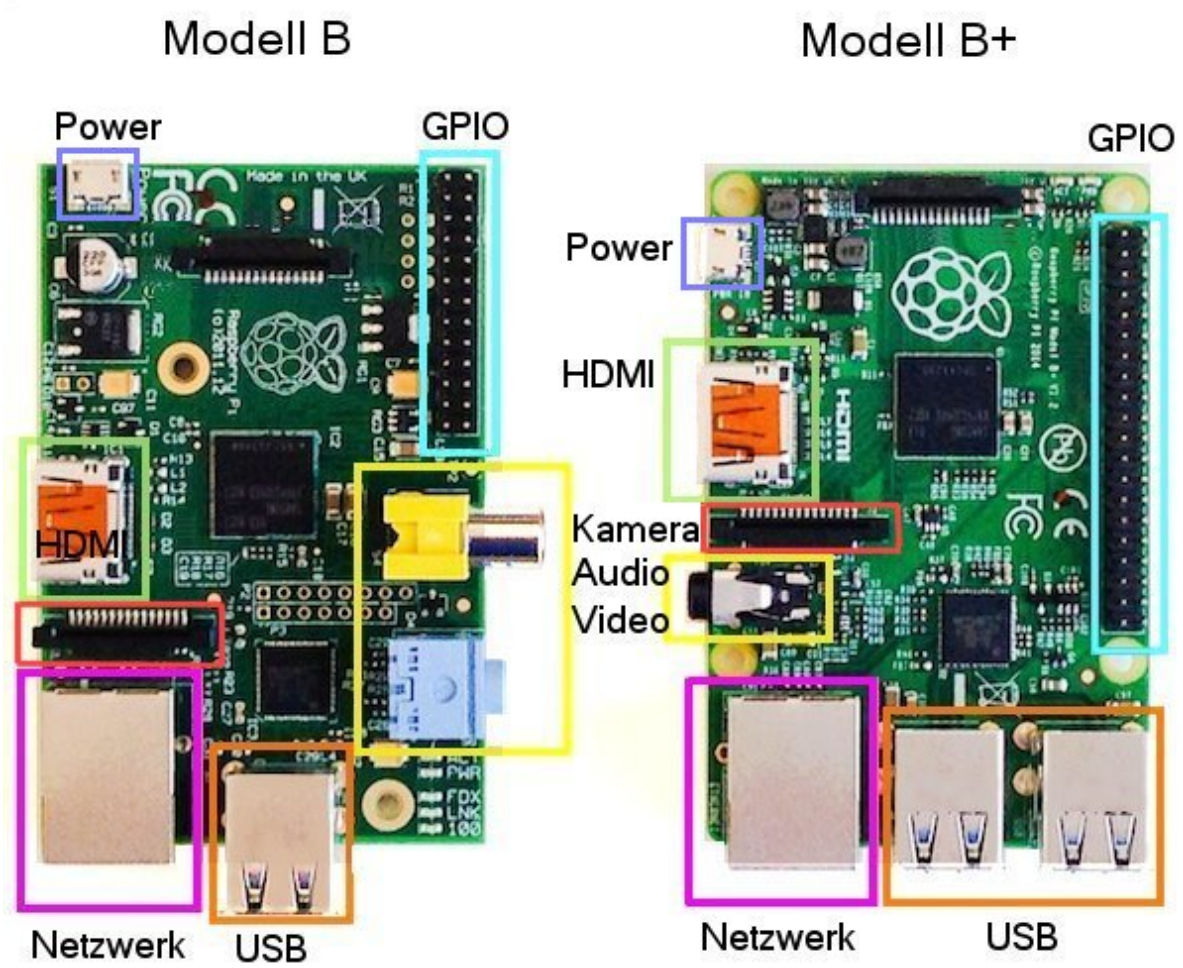
- BCM2835 SoC
- 512 MB RAM
- ARM11 (armv6) Prozessor
- VideoCore-IV-GPU
- 2x USB-2.0
- HDMI-Ausgang (Video + Audio)
- Composite-Video
- Audio-Ausgang (3,5mm Klinke)
- D-/MMC-Karten-Slot
- Ethernet-Port (10/100 MBit)
- 21 GPIO-Pins (z.B. UART, SPI und I2C)
- DSI (Display Serial Interface) über 15-poligen Flat Flex Connector
- Kamera-Interface, CSI (Camera Serial Interface) über 15-poligen Flat Flex Connector
- Altern. Stromversorgung per µ-USB



Quelle: netzmafia.de

Unterschiede Modell B und B+

+ zusätzliche Pins an der GPIO-Steckerleiste (I²C Bus)
| Audio und Composite-Video in einem Stecker
+ 2 zusätzliche USB 2.0 Ports
| Geringerer Stromverbrauch
+ stabiler beim Umstecken
| µ-SD Kartenslot
| Qualität der PWN Audio-Ausgabe durch geänderten Stromlaufplan besser

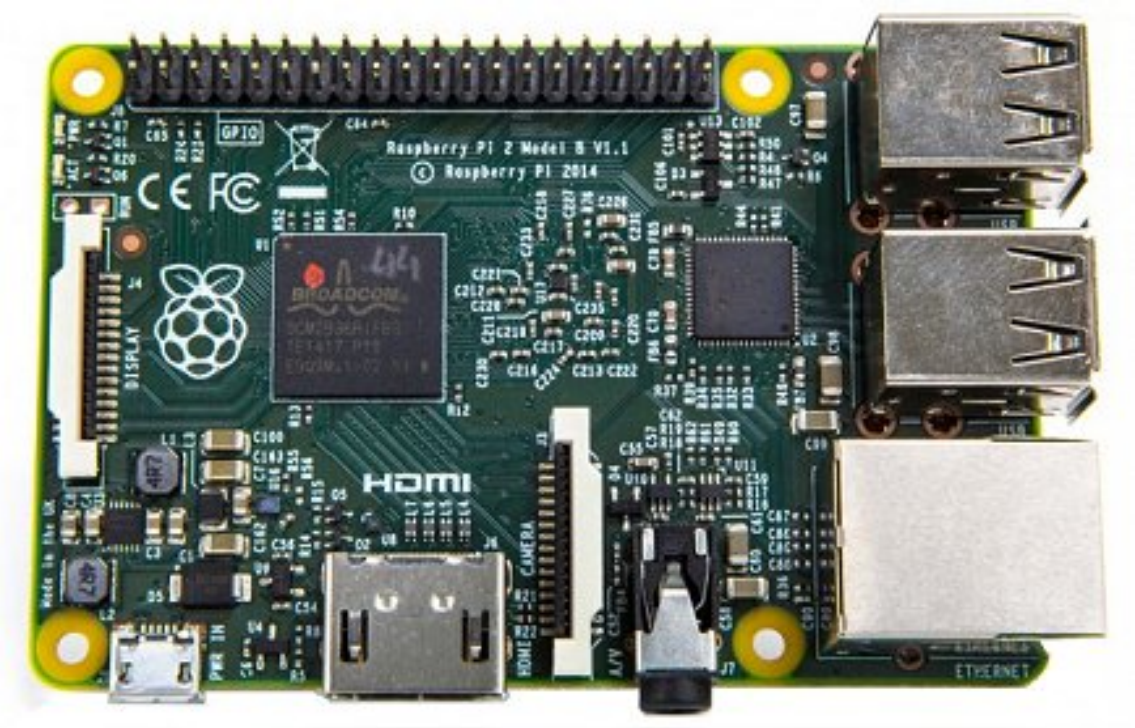


Quelle: netzmafia.de

Raspberry Pi 2

- BCM2836 SoC
- **1GB RAM**
- **ARM (4 Cortex A7 Kerne) Prozessor**
- VideoCore-IV-GPU
- 2x USB-2.0
- HDMI-Ausgang (Video + Audio)
- Composite-Video
- Audio-Ausgang (3,5mm Klinke)
- D-/MMC-Karten-Slot (Micro SD)
- Ethernet-Port (10/100 MBit)
- 21 GPIO-Pins (z.B. UART, SPI und I2C)
- DSI (Display Serial Interface) über 15-poligen Flat Flex Connector
- Kamera-Interface, CSI (Camera Serial Interface) über 15-poligen Flat Flex Connector
- Altern. Stromversorgung per µ-USB

Quelle: netzmafia.de



Optisch unterscheidbar durch einen Chip auf der Unterseite, der bei den anderen fehlt.

Betriebssysteme (Images)

- http://www.raspberrypi.org/downloads/
- **NOOBS** – Installations-Frontend für die anderen Distributionen
- **Raspbian** – Debian für Raspberry Pi (generische Distribution mit Softwarepaketen und graphischer Oberfläche)
- **OpenELEC** – Multimedia mit XBMC
- **RaspBMC** – Multimedia mit XBMC
- **Ubuntu** – „Mate"-Desktop ab Modell 2

Raspbian als Basis für KnopPIx?

- Basiert auf Debian GNU/Linux:
  - Größte komplett freie, nicht an eine Firma gebundene Community-unterstützte Distribution
  - Basis für viele andere Distributionen (u.a. Ubuntu und Knoppix)
- Leicht erweiterbar
- Nachteil: Außer Architektur-Anpassung wenig „Embedded"-spezifische Anpassungen, um nicht zu stark vom Debian-Mainstream abzuweichen, standardmäßig ist viele Firmware für verschiedene USB-Geräte nicht installiert.

Installation / Konfiguration Raspbian

Grundsätzliches Vorgehen:

- Image auf SD-Karte entpacken (z.B. `dd if=raspbian.img of=/dev/sdb`, mehrere Partitionen werden dabei angelegt)
- Booten (Resizing auf physikalische Größe und Konfiguration per (Text-) GUI `raspi-config`
- Neue Pakete installieren (Plugins bei XBMC oder Pakete bei Raspbian)
- *Knoppix-spezifische Optimierungen durchführen, Bootsystem anpassen*

Bootoptionen Raspian

- Die Bootdateien liegen bei Raspian auf der ersten Partition (Kernel, Initialsystem)

- Konfigurationsdatei: `config.txt`

- Z.B. HDMI-Auflösung (Framebuffer) auf 800x600 fest einstellen.

Zugang zum Raspberry Pi

- „Traditionell“: Monitor (HDMI, Composite, Adapter), USB-Keyboard+Maus

- Zeitgemäß:
  - Login per SSH verschlüsselt via LAN oder WLAN
  - Login per Remote-Desktop (VNC, rdesktop)

  Problem: Noch keine IP-Adresse konfiguriert oder bekannt!

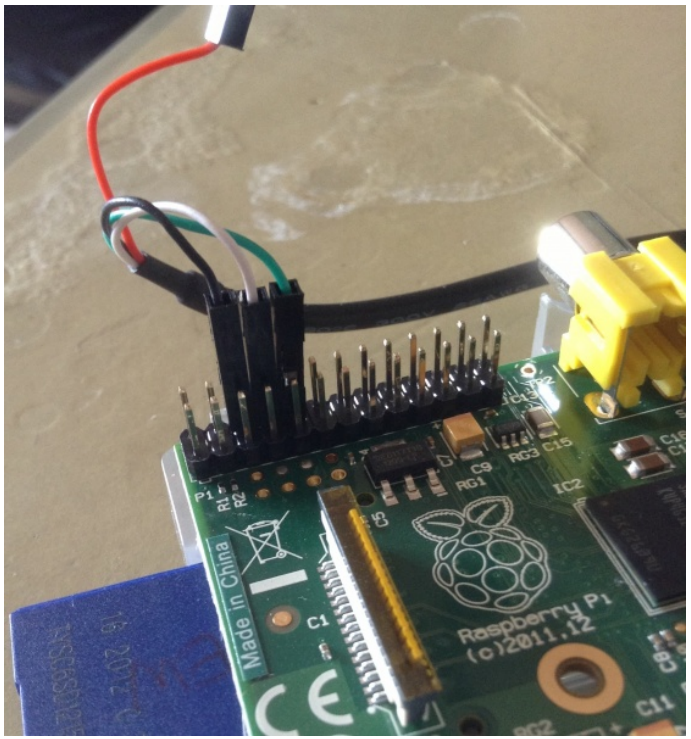
- Professionell (Embedded Programmierung): Serielle Konsole über GPIO / (D)UART (direktes „Andocken“ ans System)

Serielle Konsole (Hardware)

- **USB → GPIO/UART-Kabel** (Pinbelegung beachten!)

GROUND = Schwarz = Pin 6
TxD = Weiß = Pin 8
RxD = Grün = Pin 10

5V = Rot = Pin 2 *kann* zur Kopplung der Stromversorgung angeschlossen werden, sollte aber weggelassen werden, falls nicht erforderlich!



Serielle Konsole (Software)

- Unter Raspbian ist die serielle Konsole standardmäßig per `/etc/inittab`-Eintrag aktiv:

```
#Spawn a getty on Raspberry Pi serial line
T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

- Auf der anderen Seite des Kabels muss ein Terminalprogramm gestartet werden, z.B. **minicom** oder **screen**

  Einstellung: Device `/dev/ttyUSB0` Speed `115200`

```
screen /dev/ttyUSB0 115200
```

- Einloggen mit Login: `pi` Passwort: `raspberry`

Kommandozeile vs. Graphische Oberfläche

- `ssh raspi-ip-adresse` startet zunächst nach erfolgreichem Login eine Shell für den Benutzer „pi“.

- SSH unter Linux erlaubt auch den Direktstart graphischer Programme, die auf die eigene Desktop-Oberfläche „getunnelt“ werden.

- Desktop-Projektion bzw. Starten eines virtuellen Desktop per `tightvncserver` ist möglich (VNC). Für RDP-Clients kann entsprechend ein RDP-Server installiert werden:
  `sudo apt-get install xrdp`

Software-Auswahl und Konfiguration anpassen

- Raspbian: Menügeführte Konfiguration mit `sudo raspi-config`
- Softwarepakete aus Debian/Raspian-Repository nachinstallieren:

Kommando	Wirkung
`apt-get update`	SW-Datenbank aktualisieren
`apt-get upgrade`	(VORSICHT!) Komplettes System aktualisieren
`apt-cache search Stichwort`	Software suchen
`apt-cache show paketname`	Details anzeigen
`apt-get install paketname`	Softwarepaket installieren oder aktualisieren

Knoppix-Optimierungen

- Hardware-Erkennung: Fällt aus, aber bei Knoppix sind üblicherweise alle für den Startvorgang notwendigen Komponenten im statischen Teil des Kernels.
- Ramdisk (minirt.gz bei Knoppix) zum Finden des Boot-Mediums: unnötig.
- **RAM-Kompression** (v.a. bei Modellen vor 2 mit nur 512MB Ram sinnvoll)
- **Filesystem-Tuning** v.a. wg. langsamen Schreibvorgängen
- **Bootsystem** (Schnelles Starten, „Umgehen" des Standard-Bootvorgangs)

RAM-Kompression

- Mit 512 MB RAM ist ein „Desktop-Betrieb“ mit LibreOffice und Firefox gerade noch möglich, aer absturzgefährdet durch „overcommit“ von Speicher, der tatsächlich nicht zur Verfügung steht.
- Swap auf Flash-Medium oder USB2-Disk ist viel zu langsam!
- Lösung: „Swap in RAM, komprimiert“ → zram!

```
/sbin/modprobe zram
echo "400000000" > /sys/block/zram0/disksize
mkswap /dev/zram0
/sbin/swapon -p 0 /dev/zram0
```

- Nun kann in 400MB des Hauptspeichers komprimiert ausgelagert werden (virtuell 512+400MB Hauptspeicher).

Filesystem-Tuning (1)

Oft wird (z.B. in Foren) empfohlen, bei Flash-Medien generell ein Flash-Filesystem zu verwenden, bei dem softwareseitig ein „wear leveling" stattfindet, d.h. jedes Segment wird gleich oft geschrieben, und für defekte Segmente werden Ersatzblöcke reserviert.

Das ist bei USB-Sticks und SD-Karten kontraproduktiv, da diese einen (Hardware-)Controller besitzen, der sich um die Segmentverwaltung bereits kümmert.
→ Doppeltes wear-leveling und defect block management führt zu höherer Schreibbelastung und mehr Fehlern!

Filesystem-Tuning (2)

→ Ein gängiges Linux-Dateisystem mit schneller Binärsuche (`btree+`) wie `ext4` oder `reiserfs` genügt, aber intensives Journaling belastet wiederum bei langsamen Schreibvorgängen.

Hauptproblem: ***Schreiben auf SD-Karte ist langsam*** und blockiert auch die Lesevorgänge! → *„Einfrieren für einige Sekunden"*-Effekt

Teillösung: Vorzugsweise SD-Karten verwenden, die eine **hohe Schreib**geschwindigkeit besitzen. Die **Lese**geschwindigkeit ist hingegen (relativ) egal.

Filesystem-Tuning (3)

- Problem: Da immer „segmentweise“ geschrieben wird, ist es günstiger und insgesamt schneller, Daten „am Stück“ zu schreiben als in kleineren und häufigeren Schreibvorgängen.
- Problem: Linux schreibt standardmäßig die gepufferten Daten im Hauptspeicher alle 5 Sekunden auf den Datenträger zurück. Wenn 5 Sekunden nicht genügen, um den Schreibvorgang abzuschließen, kann sich dies auch „hochschaukeln“, denn nach jedem Schreibvorgang entsteht zusätzliche Latenz durch eine Controller-interne Verifikationsphase.

Filesystem-Tuning (4)

- Lösungsansatz: Weniger schreiben
  - Logging reduzieren oder ganz abschalten
  - Verzeichnisse wie `/tmp` und `/run` als `tmpfs` (ramdisk)
- Lösungsansatz: Anzahl der zeitaufwändigen „kleinen“ Schreibvorgänge reduzieren, um die Latenzen zu eliminieren: Tunen des I/O-Schedulers mit größeren Zeitabständen zwischen dem Zurückschreiben

```
echo 30000 > /proc/sys/vm/dirty_writeback_centisecs
```

→ Schreibt nur einmal alle 5 Minuten

Bootsystem (1)

Debian führt für jeden installierten Dienst ein Skript beim Start aus (dies wird nach Einführung von systemd parallelisiert, was es teilweise schlimmer macht, je nach Latenz des Datenträgers).

Bis zum Erreichen des interaktiven Desktops werden bis zu 1000 Programme gestartet und (die meisten) wieder beendet!

Bootsystem (2)

Knoppix-Ansatz: Umgehen des Debian-Bootsystems.

1) Abschalten des automatischen Starts der init-Skripte:

/etc/inittab:

```
# l0:0:wait:/etc/init.d/rc 0
# l1:1:wait:/etc/init.d/rc 1
# l2:2:wait:/etc/init.d/rc 2
# l3:3:wait:/etc/init.d/rc 3
# l4:4:wait:/etc/init.d/rc 4
# l5:5:wait:/etc/init.d/rc 5
# l6:6:wait:/etc/init.d/rc 6
...
```

Bootsystem (3)

Knoppix-Ansatz: Umgehen des Debian-Bootsystems.

2) Systemstart/Reboot/Shutdown als monolithische Skripte:

/etc/inittab (vollständig):

```
si::sysinit:/etc/init.d/knoppix-autoconfig
l0:0:wait:/etc/init.d/knoppix-halt
l6:6:wait:/etc/init.d/knoppix-reboot
x5:5:respawn:/etc/init.d/knoppix-startx
```

Bootsystem (4)

Knoppix-Ansatz: Umgehen des Debian-Bootsystems.

3) `/etc/init.d/knoppix-autoconfig`:

- Reduzierter Systemstart mit nur den wichtigsten Diensten:
  - `udev`
  - `(k)dbus`
  - `network-manager`
  - `cups`?

Wenn möglich als Hintergrunddienste.

Bootsystem (5)

Knoppix-Ansatz: Umgehen des Debian-Bootsystems.

4) **/etc/init.d/knoppix-startx**:

```
su -l -c "export STARTUP=startlxde; \
 exec /usr/bin/startx -- \
  -noreset -nolisten tcp vt05" "knoppix"
```

→ Der Benutzer „knoppix“ wird auf Konsole 5 (graphisch) eingeloggt und erhält eine LXDE-Session.

TODO

- RasPi-Kernel auch für Modell 2 mit `zram`-Option bauen und mit allen für den Start des Systems notwendigen Modulen im statischen Kernel-Teil integrieren (nicht als separate Module)
- Reduktion der Systemdienste auf das notwendige Maß (wie bei Knoppix Intel/AMD)
- Ggf. Kompression des Basissystems (reduziert Lesezugriffe auf 1/3)
- Zeitversetzte Parallelisierung
- Compiz!
- Design Goal: Bis zum Desktop in <= 15 Sekunden
- Wettbewerb im Open Source Forum CeBIT 2016

Fragen?

http://knopper.net/tuebix/