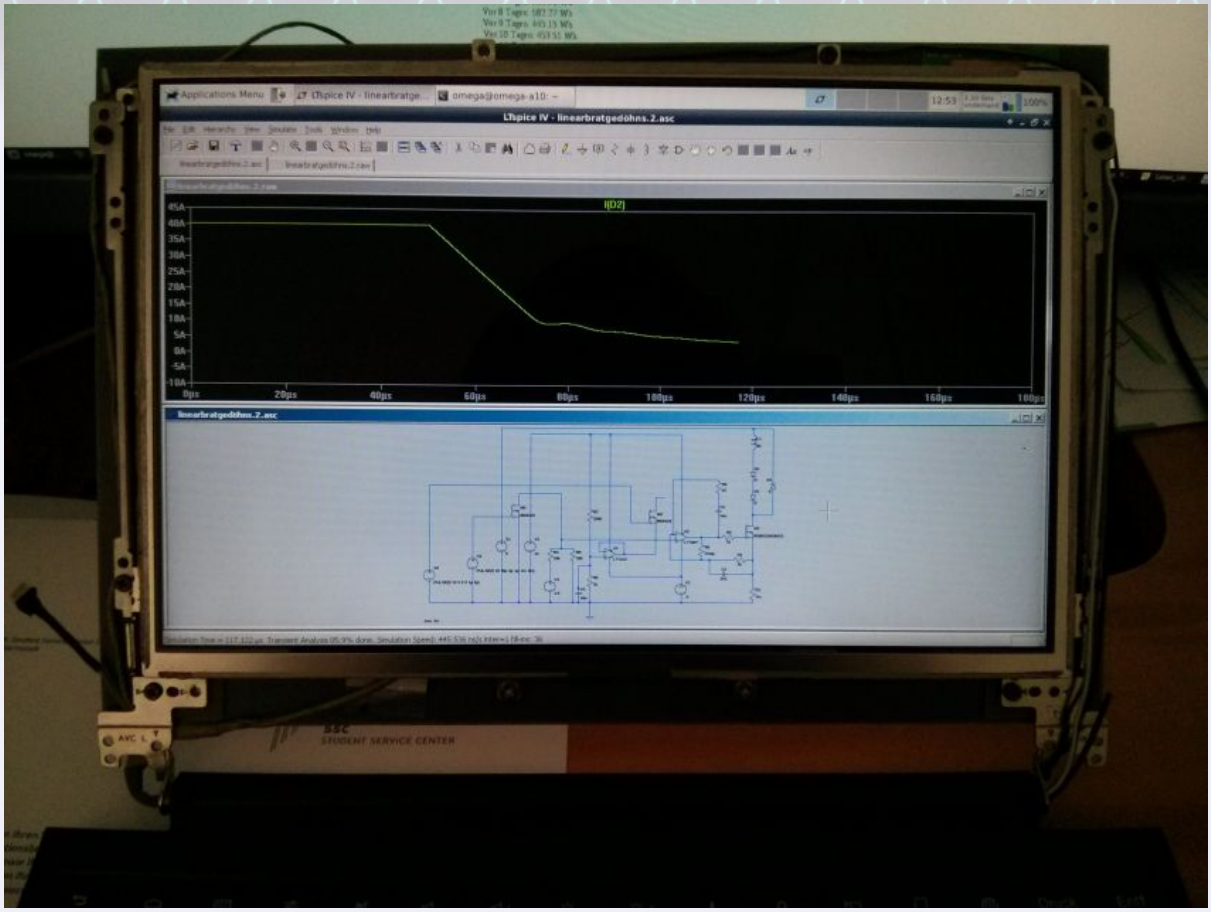
Home
Gefrickel
Audio
µC Schaltungen
Strom-versorgung
Lichttechnik
Modifikationen
NVIDIA-MXM 30-Sekunden Patch
HP-Notebook WLAN-MOD
Samsung CLP-310 Serie - Chip mod
Seagate Dockstar RTC-MOD
Seagate Dockstar Overclocking
Regelung für billige Heißluft-Station
W2022A Akkubetrieb
Debian Toradex Colibri T20
Lenovo A10 Linux
Reparaturen
Sonstiges
Über uns
Impressum

Lenovo A10 Netbook mit Linux



Das A10 ist ein recht günstiges (vor allem wenn man es defekt kauft), kleines, leichtes Netbook mit langer Akkulaufzeit.
Allerdings ist das wirklich Interessante an dem Gerät, dass es eine ARM Cortex A9 CPU mit 4 Kernen hat. Ein ARM statt x86 als Notebook wollte ich sowieso mal testen, dazu kommt noch die Aussage, dass es nicht möglich sei ein Linux darauf zu installieren. Also perfektes Gerät für mich.

Die Hardware

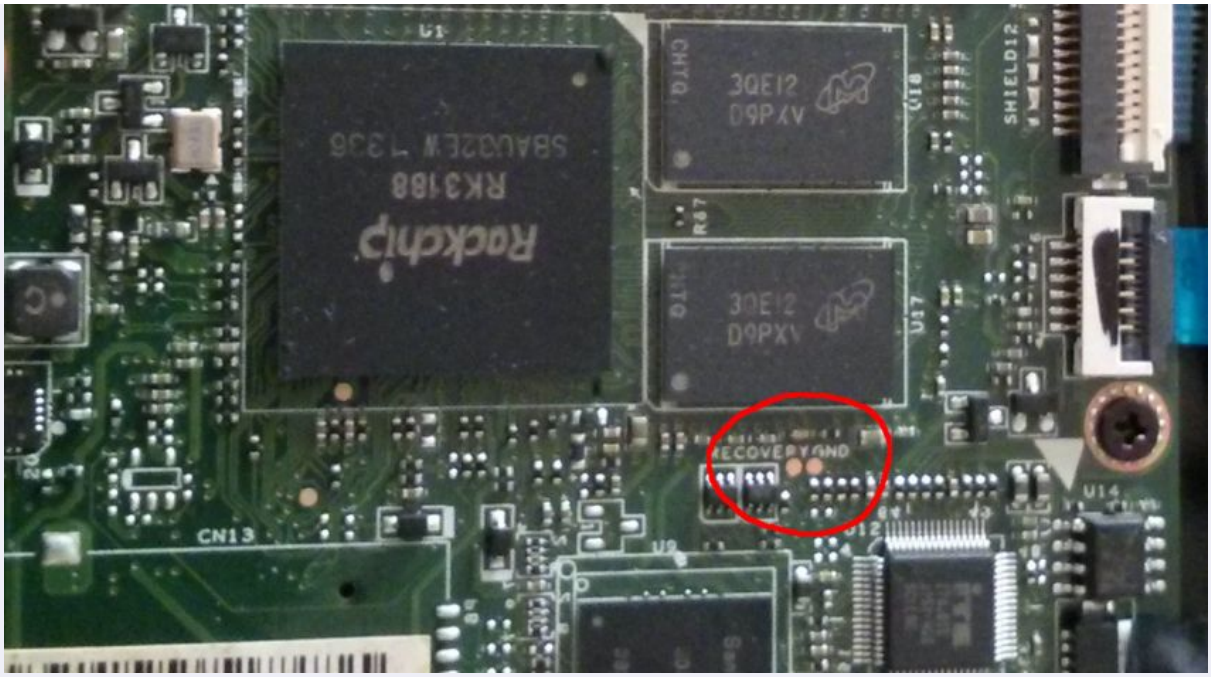
Folgende Ausstattung hat das Teil:

- 1,6 GHz Cortex A9 Quad Core CPU
- 1024 MB DDR3 RAM
- 16 GB eMMC NAND Flash
- 10 Zoll 1366 x 768 Pixel Display mit Touchscreen
- micro SDHC Steckplatz
- 2x USB Host
- 1x USB OTG
- HDMI Ausgang

Zugriff auf den Bootloader

Erfreulicherweise gibt es für die Rockchip CPUs einen Bootloader im ROM, der ein Firmwareupdate über USB ermöglicht und auf diesem Weg Vollzugriff auf den Flash ermöglicht.

Solange das Android läuft, kann man über ADB den Bootloader starten. Aber da am Kernel gebastelt wird, wird der Zeitpunkt kommen, das gar nichts mehr bootet.
Aber man kann den Bootloader nach Öffnen des Gerätes durch verbinden der beiden im Bild gezeigten Pads immer erreichen, egal was mit dem Flash macht.



Wie kommt da Linux drauf?

Zuerst sollte man ein Backup vom Flash machen, falls man doch wieder ein Android haben möchte. Dazu sollte man genug im Internet finden, sodass ich das hier nicht beschreibe.

Man kann vorher schon einen USB Stick mit einem rootfs seiner lieblings Distribution anfertigen. Hier wird Debian beschreiben, aber z.B. Ubuntu wird quasi genauso gehen.

Erstellen eines rootfs USB stick

Wer ein Debian basiertes System hat, kann sich einfach mit apt-get debootstrap installieren. Danach einen beliebigen USB Stick partitionieren und danach mit ext3 oder ext4 formatieren.
Alle gezeigen Befehle werden als root ausgeführt.

USB Stick mounten (sdX entsprechend ersetzen):

```
mount /dev/sdX1 /mnt
```

Debootstrap ausführen:

```
debootstrap --foreign --arch=armhf --include=module-init-tools,
udev,netbase,ifupdown,iproute,openssh-server,dhcpcd,iputils-ping,wget,
net-tools,ntpdate,uboot-mkimage,uboot-envtools,vim,usbutils,build-essential,
hdparm,wireless-tools,wpasupplicant --variant=minbase wheezy /mnt
http://ftp.de.debian.org/debian
```

Kernel

Nachdem das fertig ist, muss ein angepasster Kernel auf das Netbook, damit das Gerät von USB Linux bootet statt Android.

Man kann sich gerne den Kernel selbst bauen, alles Nötige dafür, inkl. WLAN Treiber habe ich hier hinterlegt. Man muss sich noch den NAND Treiber aus dem Android extrahieren und patchen, damit der für diese Kernelversion passt.

Der einfachste Weg ist es, einen fertigen Kernel inkl. Initrd zu nehmen. In der Initrd ist ein WLAN Triber, NAND Treiber sowie fbcon enthalten, sodass man gleich auf eine Konsole kommt.
Fbcon musste etwas angepasst werden, da es ursprünglich nur in einem Segfault landete, da auf einen Null Pointer zugegriffen wurde.
Übrigens kann man mit 2 Stück FT232 o.ä. und

```
console=ttyUSB0
```

auch an eine Konsole kommen, sobald der USB Treiber geladen wurde.

Einen Kernel für den ersten Boot von USB gibt es hier: boot-sda1-init-sh.img
Das ist ein Android bootimage bestehend aus Kernel und Initrd.

Flashen

0. Als erstes das Flashtool von hier runterladen und kompilieren.

1. In den Bootloader gehen durch brücken der Pads und Reset per Taster auslösen

2. Flashen:

```
./rkflashtool-5.1-src/rkflashtool w 0x008000 0x00008000 < boot-sda1-init-sh.img
```

3. Wenn das durchlief reset ausführen, ohne gebrückte Pads. Das Netbook sollte jetzt in ein unfertig installiertes Debian booten.

4. Debootstrap fertig stellen:

```
cd /debootstrap
mount -o remount,rw /
./debootstrap --second-stage
passwd
```

5. Warten bis das fertig ist

6. Diesen Kernel flashen: boot-sda1-normal.img

```
./rkflashtool-5.1-src/rkflashtool w 0x008000 0x00008000 < boot-sda1-normal.img
```

7. Nach einem Reset bootet das Gerät in ein soweit vollständiges Debian von USB Stick. An dieser Stelle kann man mit wpa_supplicant das WLAN einrichten, wenn man will auch Xorg und einen Window Manager. Wenn man Debian auf dem internen NAND haben möchte muss man das debootstrap Spiel nochmal wiederholen.

8. Formatieren des Flashs:

```
mkfs.ext4 /dev/mtdblock3
```

9. Mounten und debootstap:

```
mount /dev/mtdblock3 /mnt

debootstrap --foreign --arch=armhf --include=module-init-tools,
udev,netbase,ifupdown,iproute,openssh-server,dhcpcd,iputils-ping,wget,
net-tools,ntpdate,uboot-mkimage,uboot-envtools,vim,usbutils,build-essential,
hdparm,wireless-tools,wpasupplicant --variant=minbase wheezy /mnt
http://ftp.de.debian.org/debian
```

10. System runterfahren und im Bootloader diesen Kernel flashen: flashen: boot-mtd-init.sh.img

```
./rkflashtool-5.1-src/rkflashtool w 0x008000 0x00008000 < boot-mtd-init-sh.img
```

11. Debootstrap fertig stellen:

```
cd /debootstrap
mount -o remount,rw /
./debootstrap --second-stage
passwd
```

12. Diesen Kernel flashen: boot-mtd-normal.img

```
./rkflashtool-5.1-src/rkflashtool w 0x008000 0x00008000 < boot-mtd-normal.img
```

13. Jetzt bootet das Netbook vom internen Flash in ein Debian. Man kann sich jetzt Xorg und einen beliebigen Windows Manager installieren und das Gerät wie ein normales Notebook nutzen. Wer x86 Anwendungen ausführen will oder gar mit Wine Windows emulieren möchte muss noch etwas weiter frickeln.

Was funktioniert bisher

- Grafik mit Frambuffer Treiber problemlos
- WLAN problemlos
- USB Host problemlos, OTG ungetestet
- Webcam problemlos
- Touchscreen unbekannt, ich habe keinen
- Tastatur und Touchpad gehen, Tastaturtreiber ist angepasst um Funktionstasten zu haben
- Soundausgabe geht, Mikrophon ungetestet

Was funktioniert noch nicht

- Standbymodus

Sonstiges

Die Beschreibung hier klingt relativ einfach und kurz, aber der Weg dahin war lang. Was führt der Bootloader eigentlich aus? Wie muss das Image dafür aussehen und wo muss es hin. Funktioniert überhaupt der gebaute Kernel?! Wieso funktioniert fbcon nicht, und wie bekomme ich jetzt eine Konsole um weiter zu machen, bzw. rauszubekommen, warum der Kernel nicht bootet.
Es war nicht einfach, aber es macht Spaß!
Da mein Gerät etwas halb ist (es war dafür günstig) kann ich den Touchscreen nicht testen, wenn jemand helfen möchte, gerne!
Der Kernel, sowie alle Tools zum bauen des Kernels und Bootimages liegen hier: https://github.com/steffen-g/lenovo-a10
Falls es irgendwo in der Anleitung Fehler gibt würde ich mich über eine Nachricht freuen, dann behebe ich das natürlich.
Feedback und Fragen dazu sind erlaubt und erwünscht.

Nachtrag 11.06.2015

Der Mali binary Treiber zusammen mit dem sunxi fbturbo scheint auch zu funktionieren. Welche Vorteile das bringt ist mir noch unklar.
Man kann mit qemu übrigens sehr gut x86 Anwendungen benutzen. Eagle auf ARM geht so recht problemlos.

Nachtrag 27.07.2015

Henning hat mir noch etwas sehr Interessantes rausgefunden:

"Ich habe herausgefunden, dass man für die Recovery gar nicht das Gehäuse zu öffnen und die Pins zu brücken braucht - wenn man beim Einschalten die Lupe-Taste und "R" gedrückt hält, landet man auch dort. Man erkennt Erfolg daran, dass sich der Lenovo mit USB-ID 2207:310b meldet."

Um als normaler Nutzer Zugriff aufs Netzwerk zu bekommen muss man die Gruppe inet mit der ID 3003 hinzufügen: Mit beliebigem Editor die Datei /etc/group öffnen und inet:x:3003:root hinzufügen. Und mit useradd username inet den username hinzufügen.

Vielen Dank dafür.

Außerdem habe ich noch den Kernelsource auf Github aktualisiert, sodass auch das neuste udev und damit Debian 8 funktioniert. Die vorhandenen kompilierten Kernel Images werde ich demnächst erstellen und hinzufügen.