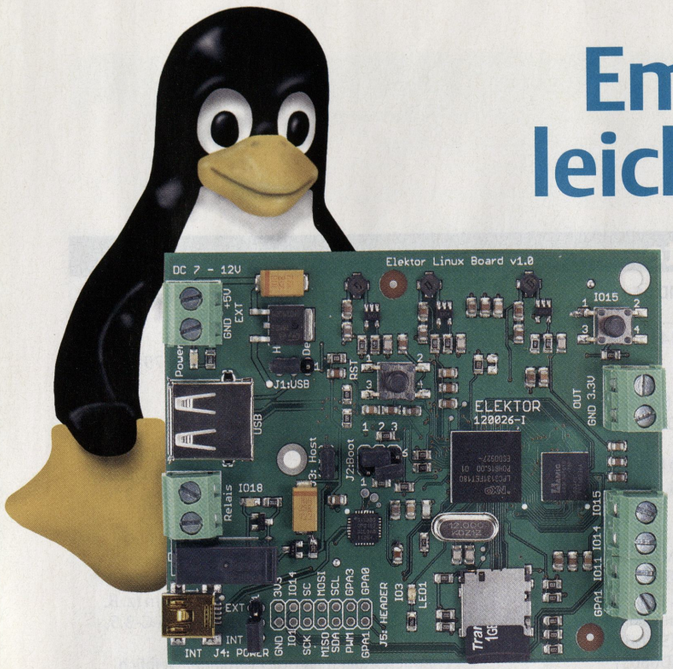


# Embedded Linux leicht gemacht (2)

## Die Hardware

Die Wahl des passenden Mikrocontrollers und der übrigen ICs ist fester Bestandteil zu Beginn eines jeden Projektes. In diesem Teil unseres Kurses schauen wir uns das Schaltbild unseres Boards und die wichtigsten Bauteile genauer an. Dann booten wir zum ersten Mal und sammeln eigene Erfahrungen. Wir werden sehen, wie einfach Linux sein kann!

Von Benedikt Sauter [1]

Im ersten Teil der Artikelserie haben wir einen Blick auf die Geschichte von GNU/Linux und die wichtigsten Komponenten des freien Betriebssystems geworfen. Nun wollen wir uns detailliert der Hardware widmen: dem Elektor-Linux-Board.

### Hauptkomponenten

Die Schaltung ist bewusst einfach gehalten und nur mit den notwendigsten Bauteilen ausgestattet. Es soll jeder Leser die Möglichkeit haben, das komplette System zu verstehen: Vom Anlegen der Betriebsspannung bis zum Starten komplexerer Anwendungen.

Um ein GNU/Linux-Betriebssystem betreiben zu können, benötigt man die klassischen Komponenten einer Computerarchitektur: Den Prozessor, RAM (Arbeitsspeicher) und ROM (nichtflüchtiger Speicher) (**Bild 1**). Wie spielt dies alles auf unserem Board zusammen? Nach dem Start muss ein Mini-Bootloader (das Äquivalent zum BIOS im PC) gestartet werden, der den Kernel vom ROM in das RAM kopiert. Anschließend wird der Kernel gestartet, der wiederum später Zugriff auf das ROM hat, um nach und nach die gewünschten Anwendungen in das RAM zu kopieren. Schauen wir uns nun die Komponenten im Detail an.

### Der Prozessor

Beginnen wir beim Prozessor. Er befindet sich in der Mitte des Boards und ist ein 12 x 12 mm großer BGA-Baustein. Ursprünglich ist der Linux-Kernel für einen x86er entwickelt worden, der allerdings alles andere als ein typischer Embedded-Prozessor ist. Dank der offengelegten Arbeit der Programmierer und einer durchdachten Softwarestruktur konnte der Kernel aber nach und nach auf die verschiedensten Prozessoren portiert werden. Wesentlich für eine Portierung von Linux auf eine neue Architektur ist die Verfügbarkeit der Toolchain (GCC-Toolchain) [2], die bereits kurz im ersten Artikel dieser Reihe beschrieben wurde. Der Prozessor muss einen Hardware-Timer und eine 32-bit-Architektur besitzen. Eine MMU (Memory-Management-Unit) ist nicht unbedingt notwendig, doch erst diese ermöglicht es einem Betriebssystem, Anwendungen unabhängig voneinander stabil laufen zu lassen. Jeder Anwendung wird dabei ein eigener virtueller Speicherbereich zugeteilt, auf andere Bereiche hat sie keinen Zugriff. Bis vor kurzem gab es ein Projekt uclinux.org [3], das einen Patch (siehe Textkasten) für den Kernel bereithielt, um diesen ohne MMU zu betreiben. In der Zwischenzeit ist dieser Patch aber fester Bestandteil der Hauptlinie (Mainline) des Kernels. Wir werden aber die MMU unseres Prozessors LPC3131 nutzen und müssen daher nicht auf diesen Patch zurückgreifen.

Unser Prozessor ist ein ARM9 LPC3131 [4] der Firma NXP (ein ARM926 mit einer Befehlssatzversion armv5). Der Prozessor arbeitet mit 180 MHz und verfügt zudem über 192 KB internes SRAM. Über einen internen DRAM-Controller können externe Arbeitsspeicher-Bausteine angesteuert werden. Außerdem sind alle wichtigen Mikrocontrollerschnittstellen







wie I2C, SPI und UART integriert. Mit 21 GPIO-Signalen, vier A/D-Eingängen, einer LCD-Schnittstelle, einer I2S-Audioschnittstelle und dem Highspeed-USB-Interface (480 Mbit/s) lassen sich bereits viele Features von Linux nutzen. Da das Elektor-Linux-Board nur 2-seitig ohne weitere Lagen geroutet ist, konnten natürlich nicht alle Signale nach außen geführt werden; es sind aber alle Wichtigen dabei.

Der verwendete Prozessor ist für den Einstieg in die Linux-Welt perfekt, da er sich bei Ausstattung und Peripherie auf das Wesentliche beschränkt. Er fällt in die Gruppe der „Low-Cost"- und „Low-Power"-Prozessoren. Klassische ARM9-Kerne bieten zum Teil viel mehr Features an, sind aber auch wesentlich komplizierter zu bedienen. Ist das Grundverständnis jedoch einmal vorhanden, kann man einfach auf aufwändigere Prozessoren umschwenken.

### Der Arbeitsspeicher

Der Prozessor selbst bietet intern nur ein 192 KB großes SRAM. Daher benötigt man einen externen Arbeitsspeicher, in dem später der Kernel, die anderen Komponenten des Betriebssystems und die Anwendungsprogramme ausgeführt werden können. Das interne SRAM wird nur zum Ausführen des Bootloader-Programms genutzt. Sobald der Bootloader den externen Speicher fertig konfiguriert hat (Einstellung des dort angesiedelten Controllers für das Timing, für das Aktualisieren der Speicherzellen u.a.), wird das interne SRAM nur noch verwendet, wenn es um Geschwindigkeit geht (Cache für das Betriebssystem oder Anwendungen). Als externer Speicher wird ein dynamischer Speicher (SDRAM) verwendet. Die Chips bieten für wenig Geld eine hohe Speicherkapazität. Um ein passendes IC aussuchen zu können, muss man lediglich einen Blick in das Datenblatt des LPC3131 [4] werfen. Diesem kann man entnehmen, dass ein 16 bit breiter externer Datenbus mit einem maximalen Adressraum von 128 MB unterstützt wird. Als besonderes Feature bietet der LPC3131 eine Unterstützung für sogenannte Low-Power-SDRAM-Bausteine an. Diese sind hinsichtlich ihres Stromverbrauchs nochmals optimiert. Auf dem Elektor-Linux-Board verwenden wir einen

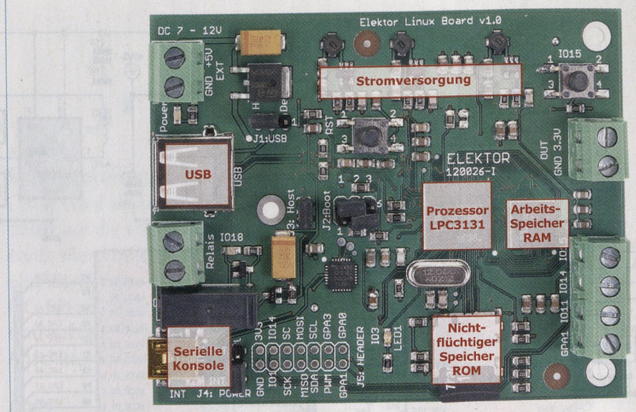


Bild 1. Die wichtigsten Komponenten des Elektor-Linux-Boards.

8 MB großen Baustein. Alternativ kann dieser später durch einen 32 MB großen Chip ersetzt werden. Da Arbeitsspeicher in der Halbleiterwelt unter dem Standard JEDEC [5] normiert ist, kann man bei vielen Herstellern einen passenden Speicher finden. In der aktuellen Schaltung haben wir den Baustein A43E26161 [6] von AMIC verwendet. Der Low-Power-SDRAM-Baustein benötigt eine Spannung von 1,8 V.

### Die Festplatte

Die „Festplatte" unseres Boards ist eine gewöhnliche microSD-Karte (**Bild 2**), wie man sie heutzutage in fast jeder Digitalkamera und vielen Handys findet. Im Wesentlichen ist diese ein NAND-Speicher mit einem kleinen Controller. Einen solchen Speicher spricht man im Vergleich zu NOR-Speicher immer block- bzw. sektorweise an. Unsere Standard-Karte hat eine Kapazität von 1 GB, jederzeit kann eine größere gewählt werden. Zur guten Verfügbarkeit einer solchen microSD-Karte kommt noch der wesentliche Vorteil hinzu, dass man dank der Karte keinen speziellen Programmer für das Board braucht. Alle Programme und Dateien kann man einfach am PC mit Hilfe eines einfachen Kartenlesers auf die Karte kopieren.

### Die Schaltung

Nun schauen wir uns einmal den Schaltplan (**Bild 3**) an. Los geht es bei der Stromversorgung. Gespeist wird das Board zum Beispiel per USB-Kabel über den Anschluss X2. Gleichzeitig lässt sich dann direkt die serielle Konsole des Linux-Systems via USB ansprechen; CP2102 (IC7) dient dabei als USB/UART-Konverter.

Wenn das System ohne Konsole auskommt, bzw. die UART-Schnittstelle für eine Anwendung reserviert ist, kann man eine externe Spannungsquelle an Anschluss X6 hängen. Hierüber lässt sich das Board auch mit höheren Spannungen versorgen, da sich vor den eigentlichen Spannungsreglern für die Prozessorspannungen noch ein MC7805ABDT (IC8) Linearregler befindet. Wir empfehlen,

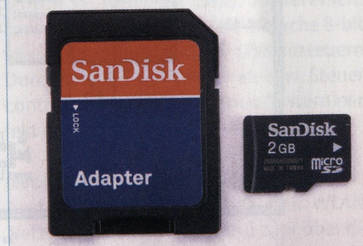


Bild 2. Eine microSD-Karte und ein SD-Karten-Adapter.

hier mit 7..12 V Gleichstrom zu arbeiten. Ob die externe oder die USB-Versorgung verwendet werden soll, stellt man über den Jumper J4 ein.

Von der internen Spannung +5 V geht es dann direkt auf die getakteten Regler IC1 bis IC3 (sie erlauben eine maximale Eingangsspannung von knapp 6 V). Über die Widerstände R6, R7 und R4 wird die entsprechende Ausgangsspannung an den Reglern eingestellt. Es werden benötigt: 3,3 V für

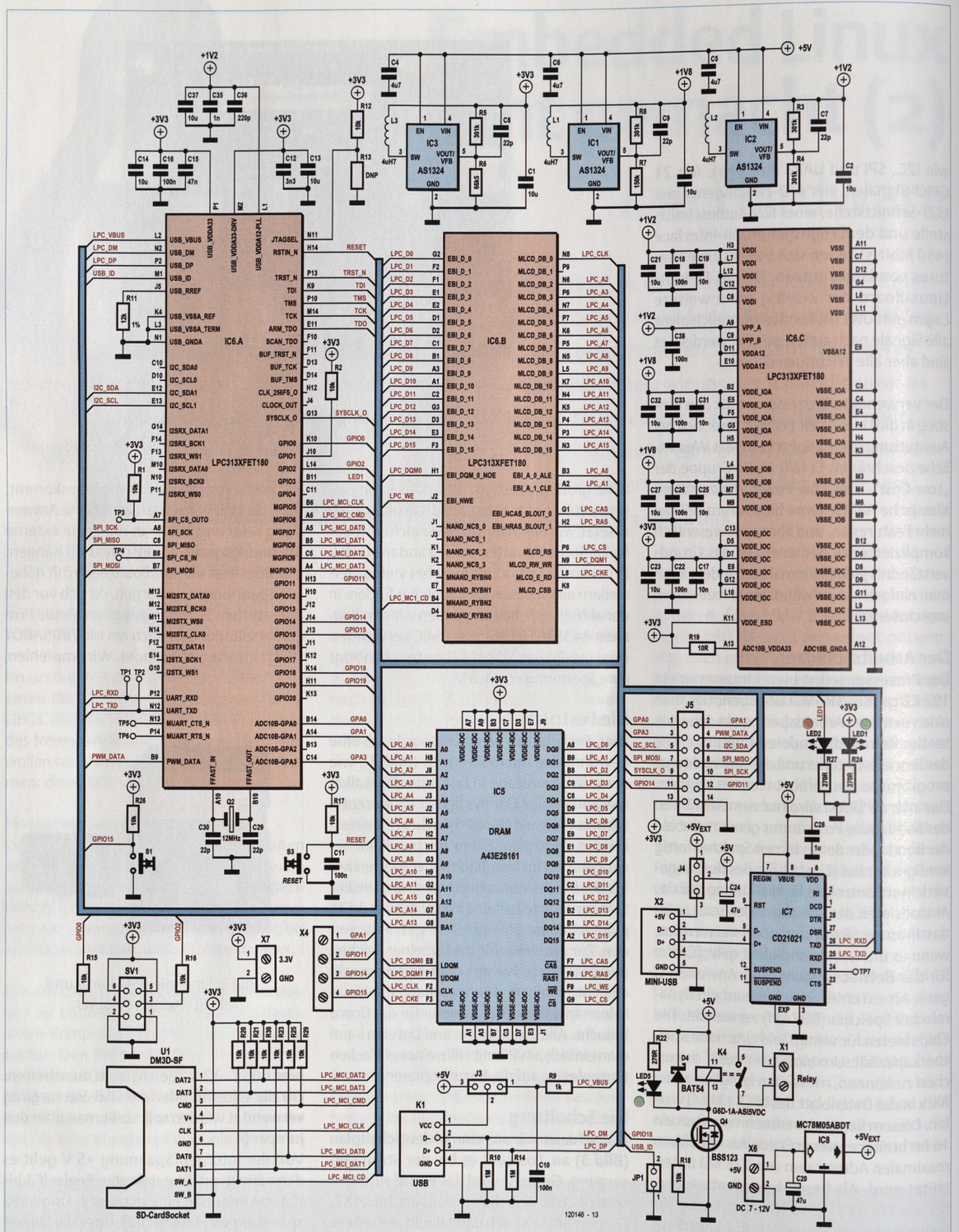


Bild 3. Schaltplan des Elektor-Linux-Boards.





den LPC3131, 1,8 V für den Arbeitsspeicher und nochmals 1,2 V für den ARM9-Kern des Prozessors. Nach Anlegen der Betriebsspannung werden die „Enable"-Signale (jeweils Pin 1) der Regler direkt aktiviert und die Erzeugung der Spannungen 1,2 V bis 3,3 V startet unverzüglich. Aufwendigere ARM-Prozessoren benötigen hier oft ein stufenweises Einschalten der Spannungen.

Jetzt ist das Board mit Strom versorgt. Als Nächstes schauen wir uns den LPC3131 im Schaltbild an. Er findet sich unter der Nummer IC6 mit den Blöcken IC6.A bis IC6.C wieder. Das Schaltbild wird übersichtlicher, wenn man große Bausteine nach logischen Gruppen aufteilt. In Block IC6.A sind alle wesentlichen Schnittstellen sowie die Ein- und Ausgänge des Prozessors zusammengefasst. In Block IC6.B ist der Daten- und Adressbus gruppiert, und schließlich findet man noch in Block IC6.C alle Stromversorgungsanschlüsse des Chips.
Ganz links unten im Schaltplan ist der SD-Karten-Halter U1 zu sehen. In diesem steckt typischerweise die SD-Karte, von der die Firmware bzw. das Linux-System geladen wird. Der LPC3131 kann jedoch von verschiedenen Speichern bzw. Schnittstellen aus booten. Die für unser Elektor-Linux-Board möglichen Varianten sind über die Steckbrücke SV1 einstellbar (siehe **Bild 4**). So lässt sich ein Boot via SD-Karte oder über die oben erwähnte USB/UART-Schnittstelle X2 durchführen. Eine weitere Möglichkeit ist das Booten über den zweiten USB-Anschluss K1 (siehe unten), wobei dann ein DFU-Programmer (Software-Standard für Bootloader) genutzt wird.

Für einfache Experimente oder eine Statusanzeige findet man noch die LED2 auf dem Board. Die LED1 leuchtet dagegen immer, sobald die Betriebsspannung anliegt. Mit Hilfe des Tasters S1 kann man an GPIO15 später Eingaben abfragen. Für eine spätere Steuerung externer Elektronik ist noch ein Relais (X1) vorgesehen, dazu aber an entsprechender Stelle in der Artikelreihe mehr.

Die gesamte Stromaufnahme des Boards liegt momentan bei ca. 85 bis 100 mA, was einer Leistungsaufnahme von etwa einem halben Watt entspricht.

## Schnittstellen

Linux wird dann spannend, wenn man direkt auf die vielen verschiedenen Mikrocontrollerschnittstellen wie digitale Ein- und Ausgänge, analoge Eingänge, PWM, I2C, SPI zugreifen kann. Die Controllerpins sind auf den 14-poligen Stecker J5 bzw. den Anschluss X4 herausgeführt.

### GPIO

Die 3,3-V-kompatiblen Ein- und Ausgänge GPIO11, GPIO14 und GPIO15 sind an den Schraubklemmen X4 verfügbar. Parallel kann man GPIO14 und GPIO11 auch über den 14-poligen Stecker J5 nutzen.

### A/D-Kanäle

Für einfache analoge Messungen können drei von maximal vier verfügbaren Kanälen genutzt werden. Als Referenzspannung werden aktuell 3,3 V verwendet. GPA1, GPA0 und GPA3 findet man an J5, GPA1 zusätzlich an einer Schraubklemme.

### I2C

Der LPC3131 kann I2C-Master und -Slave sein. Primär im Fokus steht jedoch die Masterfunktionalität. Mit dieser können wir externe Bausteine (wie z.B. einen PCA9555 I/O-Expander) einfach ansteuern. Die Signale SDA und SCL sind an J5 verfügbar.

### SPI

Genauso wie I2C-Bausteine kann man SPI-Peripherie ansteuern. Die Signale MOSI, MISO, SCK findet man ebenso an J5. Auf den Chip-Select-Pin OUT0 und (falls der LPC3131 als SPI-Slave betrieben wird) auf CS_IN hat man über die Testpunkte TP3 und TP4 Zugriff.

### PWM

Geht es darum, einen Servo anzusteuern oder eine analoge Spannung zu erzeugen, dann ist ein PWM-Ausgang perfekt. Der LPC3131 bietet einen Hardware-PWM-Ausgang. Der Anschluss findet sich ebenfalls an J5.

### UART

Eine einfache Kommunikation, vor allem zwischen Mikrocontrollern, kann man sehr einfach mit dem UART-Protokoll realisieren. Leider hat der Prozessor nur einen UART-Ausgang, welcher in der Standardkonfiguration für die Root-Konsole verwendet wird.

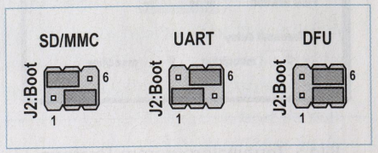


Bild 4. Bootloader-Jumper.

Arbeitet man später aber mit einem Netzwerk-Anschluss (siehe unten), dann kann man die Root-Konsole auch über diesen Kanal verwenden, und entsprechend das UART-Interface für eigene Anwendungen nutzen. RX und TX sind hierfür über die Testpunkte TP1 und TP2 zugänglich.

### USB

An den USB-Anschluss K1 lassen sich viele Erweiterungen an das Elektor-Linux-Board anschließen. Einerseits gibt es für Linux schon lange sehr viele verschiedene Programme für USB-Geräte (Sound, Video, UTMS, WLAN, LAN, etc.). Andererseits kann man mittlerweile viele einfache 8-bit-Mikrocontroller schön per USB ansteuern. Durch solche Co-Prozessoren bzw. Steuercontroller lässt sich ein Linux-System perfekt ergänzen.

### Netzwerk

Auch ein Anschluss an ein LAN- bzw. WLAN-Netzwerk ist realisierbar, und zwar über die USB-Buchse K1. Die Schnittstelle kann wahlweise im Host- oder Device-Modus betrieben werden (USB-OTG). In ersterem Fall lassen sich zum Beispiel LAN- oder WLAN-Adapter in USB-Stick-Form anschließen. Im Device-Mode fungiert das Linux-Board als USB-Gerät (z.B. als virtuelle USB-Netzwerkkarte). Wie das alles funktioniert, werden wir natürlich noch zeigen.

## Der erste Boot

Bevor wir das Board zum ersten Mal gemeinsam booten lassen, schauen wir uns einmal an, was während des Bootprozesses passiert. Um mit dem Board arbeiten zu können, verbindet man sich per USB mit einem Computer. Dort benötigt man ein RS232-

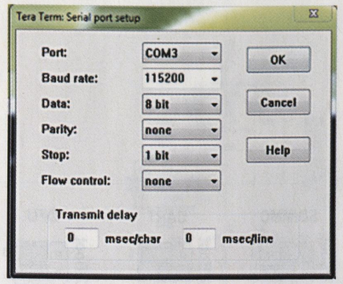


Bild 5. Terminalprogramm Teraterm.

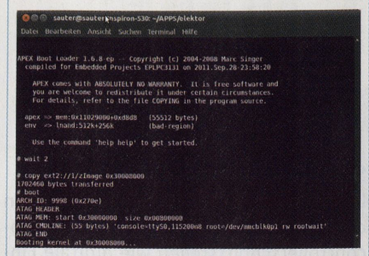

Bild 7. Bootloader APEX.

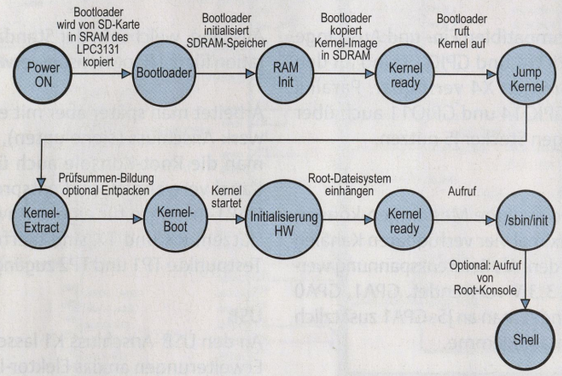


Bild 6. Bootablauf.

Terminalprogramm. Unter Windows kann man dafür TeraTerm oder Hyperterminal verwenden. Hat man einen Linux-Rechner zur Hand, kann man picocom, microcom oder ein ähnliches Programm benutzen.

Unter einem Standard-Ubuntu-System kann man sich auf der Linux-PC-Konsole mit

```
sudo apt-get install picocom
```

und

```
picocom -b 115200 /dev/ttyUSB0
```

mit dem Elektor-Linux-Board verbinden.

Unter Windows geht dies natürlich eben-

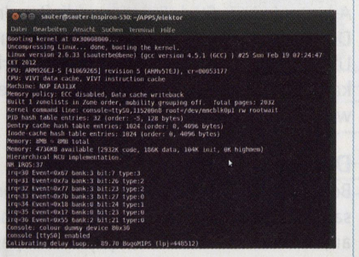

Bild 8. Linux-Kernel-Bootmeldungen.

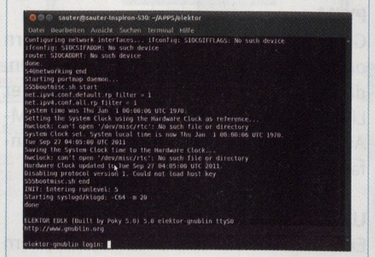

Bild 9. Login-Prompt.

falls. Den VCP-Treiber für den Seriell/USB-Konverterchip muss man manuell installieren [7]. Mit TeraTerm [8] oder dem Hyperterminal kann man sich dann mit der neuen COM-Schnittstelle verbinden. Die Einstellungen findet man im Screenshot (**Bild 5**). Nachdem man sich mit der Schnittstelle verbunden hat, sieht man die Ausgaben des Bootloaders und des Kernels im Terminalprogramm auf dem PC.

Sollte die Ausgabe des Bootens schon vorbei sein, kann man mit einem Tastendruck auf Reset (RST) den Bootvorgang jederzeit neu auslösen. In **Bild 6** ist der Bootablauf grafisch dargestellt:

- Ein Power-On- und Reset-Interrupt wird ausgelöst.
- Der interne Bootloader im LPC3131 versucht (abhängig vom Bootloader-Jumper) die Bootloader-Firmware in das interne SRAM zu kopieren (in unserem Fall von der SD-Karte).
- Die Firmware wird in das interne SRAM kopiert und dort gestartet.
- Die Firmware (Bootloder APEX) initialisiert das externe SDRAM und kopiert den Kernel von der SD-Karte in das SDRAM (**Bild 7**).
- Die Firmware ruft den Kernel auf.
- Der Kernel entpackt sich selbst und startet anschließend automatisch (**Bild 8**).
- Der Kernel initialisiert die Hardware (UART für die Konsole u.a.).
- Der Kernel hängt während des Bootvorgangs das Root-Dateisystem ein (liegt auf der SD-Karte).
- Der Kernel startet den ersten Prozess/sbin/init.
- Der Kernel startet auf der UART-Schnittstelle die Konsole.

Nachdem jetzt der Login-Prompt (**Bild 9**) auf eine Eingabe wartet, kann man sich als Benutzer „root“ am System anmelden: Dazu gibt man einfach root ein und drückt anschließend auf die Enter-Taste.

Erste Erfahrungen kann man nun mit den Befehlen machen, die wir in der **Tabelle 1** zusammengestellt haben. Zum Beispiel sollte man einmal eine Test-Datei anlegen





## Begriffe aus der Source-Code-Welt

### Patch

Ein Patch ist eine Datei, die Änderungen des ursprünglichen Quelltexts mit einer Hilfssoftware einspielen kann. Baut man eine eigene Erweiterung für ein Programm, lassen sich außerdem mittels dieser Hilfssoftware die Unterschiede zum Original-Quelltext ermitteln und eine Patch-Datei generieren.

Mit den Patch-Dateien können alle Mitglieder eines Entwicklungsteams ihren Quelltext auf den gleichen Stand bringen. Die Patch-Dateien sind das Rückgrat der Open-Source-Entwicklung, denn alle großen Projekte werden von vielen verschiedenen Entwicklern bearbeitet.

### Hauptlinie / Mainline

Die Entwicklung von großen Open-Source-Projekten ist unterschiedlich organisiert. So gibt es meist einen Maintainer. Dieser pflegt den Quelltext, speist die Patch-Dateien anderer Entwickler ein und erzeugt regelmäßig neue Programm-Versionen für die Anwender der Software. Beim Linux-Kernel gibt es eine sogenannte Hauptlinie, die von Linus Torvalds und seinem Team gepflegt wird. Da es aber eine Fülle an Patch-Lieferanten für Erweiterungen gibt, dauert es oft einige Zeit, bis neue Features in die Hauptlinie aufgenommen sind. Vor diesem Zeitpunkt kann man die Patch-Dateien direkt von den Entwicklern (über deren Homepage) herunterziehen und selbst seinen Kernel patchen. Ziel ist es aber, alles in die Mainline aufzunehmen.

### Maintainer

Der Maintainer ist meist eine Person (oft der Erfinder des Projekts). Er pflegt die Quelltexte als oberste Instanz, spielt Patch-Dateien ein und gibt regelmäßig eine neue Version des Quellcodes frei. Bei Linux gibt es für die verschiedenen Subsysteme wie Netzwerk, Treiber, Dateisysteme u.a. eigene Maintainer, die sich um die Quelltexte kümmern und so die Stabilität und Verfügbarkeit sichern.

und diese editieren. Auch ein Verzeichnis ist schnell erzeugt. Wer ein wenig übt, kommt später gut mit dem Dateisystem zurecht. Als kleines Highlight nehmen wir uns vor, die rote LED auf dem Board ein- und wieder auszuschalten. Doch wie geht das?

Eine Grundidee von Unix lautet: Alles ist eine Datei! Jedes Gerät ist im Dateisystem als Datei symbolisiert, von der man lesen und auf die man schreiben kann. So auch unsere LED!

Zuerst muss man folgendes eingeben:

```
cd /sys/class/gpio

echo 3 > export

cd gpio3

echo out > direction
```

Dies konfiguriert den Pin für die Ausgabe. Die Eingabe von

```
echo 1 > value
```

schaltet die LED ein. Und mit

```
echo 0 > value
```

kann man sie wieder ausschalten. So einfach kann Linux sein!

Bevor Sie mir der Arbeit stoppen, müssen Sie das Board (wie ein PC) vor dem Ausschalten herunterfahren. Dies macht man wie in der Tabelle 1 beschrieben mit halt. Sobald System halted. auf der Konsole erscheint, kann das Board abgesteckt werden.

## Ausblick

Im nächsten Artikel werden wir das erste Mal Quelltexte herunterladen und kleine Programme selbst schreiben. Ein etwas längerer Abschnitt unseres Kurses wird die Installation der Entwicklungsumgebung werden. Doch auch hier machen wir es unseren Lesern einfach, indem wir ein fertiges Image zum Download anbieten, das man in einer virtuellen Maschine verwenden kann.

(120146)

## Weblinks

[1] sauter@embedded-projects.net
[2] http://gcc.gnu.org
[3] www.uclinux.org
[4] http://ics.nxp.com/products/lpc3000/datasheet/lpc3130.lpc3131.pdf
[5] www.jedec.org
[6] www.amictechnology.com/pdf/A43E26161.pdf
[7] www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx
[8] http://ttssh2.sourceforge.jp/

**Tabelle 1: Wichtige Linux-Befehle.**

| Befehl | Beschreibung |
|---|---|
| ps ax | Anzeigen aller Prozesse |
| free | Speicherauslastung |
| date | Aktuelle Uhrzeit ausgeben |
| touch test.xt | Anlegen einer leeren Datei test.txt |
| rm test.txt | Löschen der Datei test.txt |
| nano test.txt | Datei test.txt im Editor öffnen (STRG-o Datei schreiben, STRG-x Editor beenden) |
| df | Partitionen anzeigen |
| mkdir test | Anlegen des Verzeichnisses „test“ |
| cd test | Wechseln in das soeben anlegte Verzeichnis „test“ |
| cd .. | Eine Verzeichnis-Ebene nach oben wechseln |
| rmdir test | Verzeichnis „test“ löschen |
| cat /proc/cpuinfo | Ausgabe der Datei /proc/cpuinfo |
| halt | Linux-System herunterfahren |