Embedded Linux leicht gemacht (3)

Softwareentwicklung

Von Benedikt Sauter [1]

Erst die richtige Software erweckt einen Mikrocontroller zum Leben. Zusätzlich zur klassischen Firmware muss man bei Embedded GNU/Linux auch die Betriebssystemkomponenten generieren. In diesem Artikel zeigen wir, wie das funktioniert. Und ein erstes kleines C-Programm schreiben wir auch!

Ein Embedded-Linux-System entwickelt man am besten mit einem Linux-System, üblicherweise auf einem PC. Als Basis verwenden wir ein „Ubuntu"-Linux in der Version 12.04 [2]. Was benötigt man, um das System zu installieren? Eigentlich nicht viel – freien Platz auf der Festplatte und idealerweise einen Netzwerkanschluss.

Zuerst muss man sich das Abbild der Installations-CD aus dem Internet herunterladen [2]. Für unsere Arbeiten bieten sich die 32-bit- oder 64-bit-Desktop-Versionen an, im Zweifel wählt man die 32-bit-Version.

Nachdem man das CD-Abbild heruntergeladen hat, muss man es nur noch mit einem Brennprogramm auf eine CD brennen. Wichtig ist, dass man es als „Image" und nicht als Datei auf eine CD brennt. Ist die CD gebrannt, kann man diese in den PC stecken und dort von CD booten (muss man entsprechend im BIOS konfigurieren). Ubuntu meldet sich nun mit dem Startbildschirm in **Bild 1**. Um mit dem System vernünftig arbeiten zu können, sollten wir es auf Festplatte installieren, also wählen wir den zweiten Menüpunkt (**Bild 2**). Dann wird man Schritt für Schritt durch den Installationsvorgang geführt. Zuerst wählt man die Sprache (**Bild 3**). Third-Party-Software (**Bild 4**) müssen wir nicht installieren. Im folgenden Fenster in **Bild 5** kann man festlegen, ob Linux über die ganze Festplatte verfügen darf (etwa bei einem Computer, auf den noch kein Betriebssystem aufgespielt wurde). Falls auf dem Computer bereits ein anderes Betriebssystem installiert ist, muss man entweder über eine zweite Festplatte verfügen oder man teilt die Platte in mehrere Partitionen ein. In **Bild 6** wird das Laufwerk oder die Partition ausgewählt, dann geht es mit den Fenstern in **Bild 7** und **8** weiter. Was es mit der Passwortvergabe (**Bild 9**) auf sich hat, erklären wir später. Wenn alles klappt, müsste man **Bild 10** zu Gesicht bekommen.

Eine andere (ziemlich bequeme) Möglichkeit ist, das Betriebssystem in einer virtuellen Maschine laufen zu lassen. Speziell für Elektor-Leser hat der Autor ein fertiges Image eines Linux-Computers für Entwicklungszwecke generiert, das man von der Elektor-Website downloaden kann [3]. Als Virtualisierungsprogramm kommt „VirtualBox" zum Einsatz, das kostenlos unter [4] erhältlich ist. Nachdem Virtualbox installiert ist, kann man das Image einfach über Hauptmenü > Datei > „Appliance import" importieren. Ist die Maschine importiert, kann man diese direkt starten.

Toolchain auf CD

Nach dem Start des Betriebssystems geht es im nächsten Schritt darum, die Toolchain zu installieren. Wer sich für das VirtualBox-Image entschieden hat, braucht die Schritte in den nächsten zwei Abschnitten nicht auszuführen, die Toolchain wartet dann bereits fertig installiert auf ihren Einsatz.

Bild 1. Mit der Pfeiltaste eins nach unten.

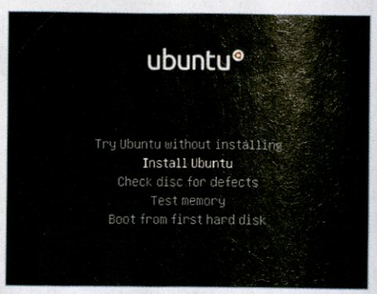

Bild 2. „Install Ubuntu" wählen und Enter eingeben.

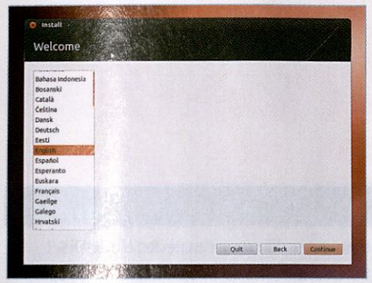

Bild 3. Sprache wählen.

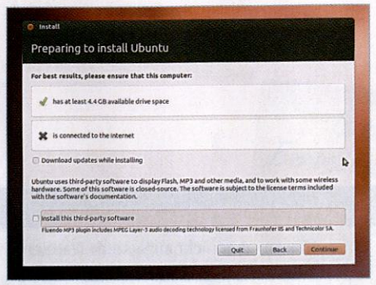

Bild 4. Vorbereitung der Installation.

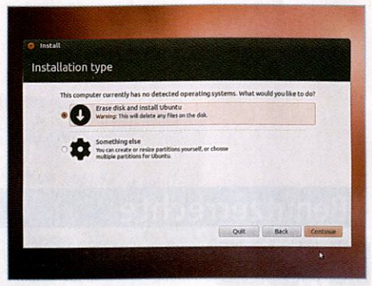

Bild 5. Installations-Ziel auswählen.

Der schnellste Weg unter Linux ist natürlich die Konsole. Am einfachsten öffnet man sich eine frische Konsole auf dem PC mit der Tastenkombination STRG-ALT-T.
In der Konsole wechselt man dann in den Ordner „/tmp":

```
cd /tmp
```

Dann folgt der Download der ARM-Toolchain-CD direkt von der Konsole aus mit dem Tool wget:

```
wget ftp://ftp.denx.de/pub/eldk/5.0/iso/armv5te-qte-5.0.iso
```

Nachdem das CD-Image heruntergeladen ist, kann man dieses unter Linux direkt öffnen, ohne sich erst eine echte CD brennen zu müssen. Das CD-Abbild muss man aber zuerst „mounten", was man mit „einhängen eines Datenträgers" in das Betriebssystem übersetzen könnte.
Vorher wechselt man in den Ordner /media.

```
cd /media
```

Prinzipiell könnte man die CD in jeden beliebigen Punkt im Verzeichnisbaum einhängen, jedoch gibt es unter Linux gewisse Standards; und so kann sich jeder immer schnell in einem neuen System zurechtfinden. Wie ein Standard-Verzeichnisbaum aufgebaut ist, wird später noch beschrieben.
Wir wollen nun einen leeren Ordner „eldk-iso" anlegen, in den wir das CD-Abbild in das Dateisystem einhängen werden. Als Windows-User darf man sich dies nicht so vorstellen, dass das CD-Image in diesen Ordner tatsächlich kopiert wird, vielmehr wird der Inhalt der CD über dieses Verzeichnis zugänglich gemacht. Denn in Linux wird ja alles durch Verzeich-

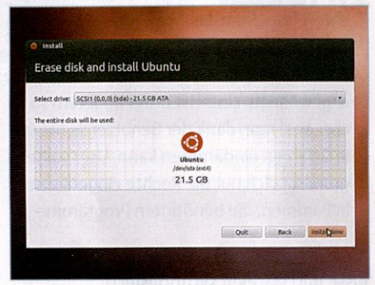

Bild 6. Festplatte auswählen.

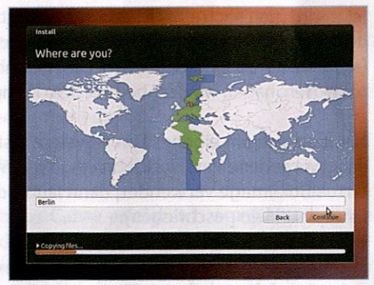

Bild 7. Zeitzone wählen.

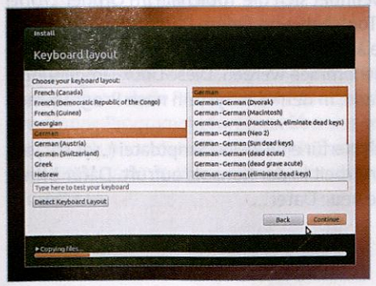

Bild 8. Tastatur-Layout einstellen.

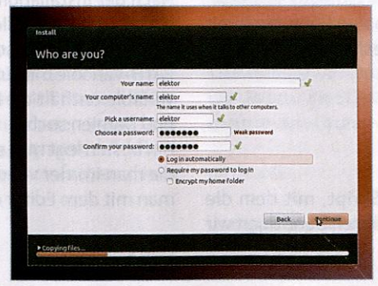

Bild 9. Benutzernamen und Passwort vergeben.

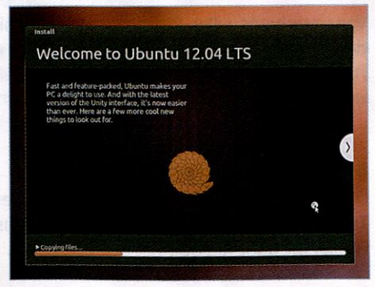

Bild 10. Installation starten.

Benutzerrechte

Unter Linux gibt es immer den sogenannten Benutzer „root“. Dies ist der User mit den meisten Rechten, alle anderen Benutzer haben gewöhnliche Zugangsberechtigungen. Möchte man jedoch auf System-Dateien, Geräte o.ä. zugreifen, braucht man eben diese „Root-Rechte“. Jetzt könnte man der Einfachheit halber immer als Benutzer „root“ arbeiten, doch das sollte man sich nie angewöhnen. Linux ist unter anderem so sicher, weil man dank der Benutzerrechte viele Gefahren eindämmen kann. User sollten grundsätzlich nur die Rechte eingeräumt bekommen, die benötigten Programme auszuführen, aber nicht an System-Dateien oder andere wichtige Daten herankommen.

Als Linux-Programmierer kann man den Anwendern immer gezielt die richtigen Rechte einräumen, so dass es keinen Zwang gibt, komplett als „root“ zu arbeiten. Auf unserem Linux Board werden auch wir schon bald einen Benutzer anlegen, mit dem wir dann arbeiten bzw. unsere Programme ausführen lassen werden.

Benötigt man aber nur kurzfristig „Root-Rechte“, etwa zum Anlegen eines Verzeichnisses in einem System-Ordner, dann kann man unter den meisten neuen Linux-Distributionen (wie auch bei unserer Ubuntu-Version) einfach das Kommando sudo vor den eigentlichen Befehl schreiben. Dies bedeutet, dass der nächste Befehl einmalig als Root ausgeführt wird.

Hat man sich ein eigenes Linux-System installiert, muss man danach das dabei vergebene Passwort eingeben. Wenn man das Elektor-VirtualBox-Image verwendet, dann lautet das Passwort „elektor“ (komplett klein geschrieben).

nisse und Dateien symbolisiert!
Hier ist der Befehl zum Anlegen des Ordners:

```
sudo mkdir eldk-iso
```

Jetzt wird man nach einem Passwort gefragt, weil wir zum Anlegen des Verzeichnisses kurzzeitig mit Root-Rechten arbeiten müssen (siehe Kasten „Benutzerrechte“). Dass wir mit diesen Rechten arbeiten wollen, haben wir der Shell mit dem Kommando sudo mitgeteilt.

Hat man sich ein eigenes Linux-System installiert, muss man das dabei vergebene Passwort jetzt eingeben. Wenn man das Elektor-VirtualBox-Image verwendet, dann lautet das Passwort „elektor“ (komplett klein geschrieben).

Jetzt kann das CD-Abbild in das Dateisystem eingehängt werden, so dass wir auf den Inhalt der CD zugreifen können:

```
sudo mount -o loop /tmp/armv5te-qte-5.0.iso /media/
eldk-iso
```

Von unserer aktuellen Position im Verzeichnisbaum können wir mit der folgenden Eingabe sozusagen in die CD wechseln:

```
cd eldk-iso
```

Installation der Toolchain

Direkt im Ordner befindet sich ein kleines Skript, mit dem die Toolchain installiert werden kann. Zum Ausführen benötigen wir wieder die Root-Rechte:

```
sudo ./install.sh -s -i qte armv5te
```

Als Ausgabe erscheint so etwas:

```
*** Installing ./targets/armv5te/eldk-eglibc-i686-
arm-toolchain-qte-5.0.tar.bz2
```

Ist die Installationsroutine beendet, kann das Verzeichnis verlassen werden:

```
cd ..
```

Dieser Schritt war wichtig, damit wir im nächsten Schritt die CD wieder aushängen können. Niemals darf man sich in dem Ordner befinden, der wieder ausgehängt wird, sonst gibt es eine Fehlermeldung und das Betriebssystem verweigert diesen Befehl.

```
sudo umount /media/eldk-iso
```

Das für die Toolchain-CD erstellte Verzeichnis kann jetzt auch wieder entfernt werden.

```
sudo rmdir eldk-iso/
```

Nach der Installation befindet sich die Toolchain im Ordner „/opt/eldk-5.0/“. Damit die Toolchain von jedem Punkt im Verzeichnisbaum über die Konsole aufgerufen werden kann, muss diese in die PATH-Variable mit aufgenommen werden. Diese Linux-Umgebungsvariable enthält die Pfade, in dem das System nach Programmen und Dateien sucht.
Am besten legt man sich hierfür eine kleine Skriptdatei („set.sh“) an, die man immer vor der Arbeit in der Konsole aufruft. Dafür öffnet man mit dem Editor die neue Datei ...

```
gedit set.sh
```

... und gibt den folgenden Inhalt ein:

```
#!/bin/bash
```



```
P1=/opt/eldk-5.0/armv5te/sysroots/i686-oesdk-linux/
usr/bin/armv5te-linux-gnueabi/
P2=/opt/eldk-5.0/armv5te/sysroots/i686-oesdk-linux/
bin/armv5te-linux-gnueabi/

export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabi-
export PATH=$P1:$P2:$PATH
```

Der letzte Befehl erweitert die PATH-Variable um die oben definierten Pfade.
Anschließend speichert man die Datei am besten direkt im Startverzeichnis des Benutzers ab (auch „Homeverzeichnis“ genannt, dort sollte man sich automatisch befinden). Die Datei muss man zu Beginn der Arbeit mit der Konsole „sourcen“, das heißt von der Shell (dem Linux-Kommandozeilen-Interpreter) einlesen lassen. Dies macht man zum Beispiel mit folgendem Befehl:

```
. ./set.sh
```

Hierbei genau auf die Schreibweise achten: Punkt Leerstelle Punkt Schrägstrich ... !

Möchte man nicht jedes Mal nach dem Aufruf einer neuen Konsole . ./set.sh eingeben, kann man den Inhalt dieser Datei auch in der Datei „.bashrc“ unterbringen, diese wird bei jedem Öffnen der Konsole automatisch ausgeführt. Die „.bashrc“ liegt im Homeverzeichnis, in das man immer mit cd gelangt. Dann öffnet man die Datei mit:

```
gedit .bashrc
```

Der Compiler in Aktion

Die Toolchain-Programme zur Erstellung des Linux-Kernels und des Bootladers beginnen mit „armv5te-“. Der Compiler GCC trägt den Namen „armv5te-gcc“. Mit:

```
armv5te-gcc --version
```

kann man sich die Versionsnummer des Compilers ausgeben lassen (vor „version“ stehen zwei einzelne Striche!). Hiermit kann man auch prüfen, ob die PATH-Variable nun die passenden Pfade enthält. Um eigene Programme für Linux übersetzen zu können, muss man die Toolchain verwenden, die mit „arm-linux-gnueabi-“ (nicht armv5te-!) startet. Wie wäre es jetzt mit einem einfachen „Hello World“?

Datei anlegen:

```
gedit hello.c
```

Inhalt der Datei eintippen:

Gratis-Webinar!
„Embedded Linux made easy“
Mit dem Autor und Entwickler
Benedikt Sauter!
Am Donnerstag, 20. September 2012,
16.00 Uhr, veranstaltet Elektor in
Zusammenarbeit mit element14 (Farnell) ein
Gratis-Webinar zum Elektor-Linux-Board.
Gleich anmelden unter:
www.elektor.de/webinar

```
#include <stdio.h>

int main(void)
{

  printf(„Hello World!\r\n“);
  return 0;

}
```

Auf Speichern und Beenden klicken. Jetzt ist man wieder in der Konsole.

Nun übersetzen wir „hello.c“ auf dem PC:

```
arm-linux-gnueabi-gcc -o hello hello.c
```

Um zu überprüfen, ob das Ganze erfolgreich war, kann man die Datei „hello“ nun auf die SD-Karte kopieren. Dafür steckt man die SD-Karte des Elektor-Linux-Boards (aus dem ausgeschalteten Board ziehen) in einen PC-Kartenleser. Nach dem Anstecken des Lesers inklusive der Karte muss man einen kurzen Moment warten. Normalerweise geht jetzt automatisch ein Fenster in Ubuntu auf. Dieses kann man getrost schließen, da wir die Datei per Konsole auf die SD-Karte kopieren wollen.

Ein großes Betriebssystem wie Ubuntu mountet den externen Datenträger automatisch ein. Wir müssen jetzt nur herausfinden, wo Ubuntu die SD-Karte eingehängt hat. Am besten wechselt man dafür in den Ordner /media

```
cd /media
```

und gibt dort einmal

```
ls
```

(= „list“) ein. Sollten sich dort mehrere Ordner befinden, kann man einfach mal mit

Erste Hilfe bei „zerschossener“ SD-Karte

Wenn das System nach dem Booten von einer SD-Karte nicht korrekt mit `halt` oder `poweroff` heruntergefahren wird, besteht die Gefahr, dass die Dateisystemtabelle zerstört wird.

Dann erscheinen typischerweise die „EXT2-fs errors“:

```
Dateisystem „EXT2-fs (mmcblk0p1): error: ext2_lookup:
deleted inode referenced: 694962“:
```

```
e2fsck 1.42 (29-Nov-2011)
/dev/sdh1 was not cleanly unmounted, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sdh1: 66/24576 files (0.0% non-contiguous), 8294/979
```

Glücklicherweise kann man das Dateisystem mit einem Linux-PC über die Konsole wiederherstellen.

Zuerst steckt man die SD-Karte in den Kartenleser und prüft mittels `dmesg`, welcher Gerätenamen vergeben wurde. Man erhält zum Beispiel folgende Ausgabe:

```
[ 1549.424156] sd 7:0:0:2: [sdh] Assuming drive cache: write through [ 1549.425624] sdh: sdh1 sdh2 [
1549.427527] sd 7:0:0:2: [sdh] Assuming drive cache: write through [ 1549.427533] sd 7:0:0:2: [sdh] Attached SCSI
removable disk [ 1549.730223] EXT2-fs (sdh1): warning: mounting unchecked fs, running e2fsck is recommended
```

Die erste Partition ist die, um die es geht, in diesem Fall lautet der Name „sdh1“. Hier befindet sich ein ext2-formatiertes Dateisystem, welches man reparieren kann.

Als erstes trennen wir die Verbindung in das Dateisystem:

```
umount /dev/sdh1
```

Jetzt kann man das Dateisystem mit dem Tool „e2fsck” wiederherstellen (mit „fsck.ext2“ funktioniert es auch) ...

```
sudo e2fsck /dev/sdh1
```

... und man bekommt eine Ausgabe, wie sie im Screenshot oben rechts zu sehen ist (gezeigt ist die Ausgabe in englischer Sprache). Abwärts kommt es vor, dass das Programm fragt, ob bestimmte Aktionen durchgeführt werden sollen. In diesem Fall immer alles mit „y“ beantworten („j“ bei deutscher Sprache).

Jetzt sollte wieder alles fehlerfrei arbeiten!

```
cd <Verzeichnis-Name>
```

in die jeweiligen Ordner schauen (danach jeweils mit `cd ..` wieder eine Ebene nach oben wechseln). Das Verzeichnis, in dem man Zugriff auf den Inhalt der SD-Karte hat (eingehängt im Ordner /media), erkennt man an einer typischen langen Nummer. In dem Ordner liegt das Dateisystem des Linux-Systems, es enthält zum Beispiel die Dateien mit den Namen „zImage“ oder „swapfile1“.
Noch ein Tipp: Wenn man nicht mehr weiß, in welchem Ordner man sich gerade befindet, kann man einfach `cd` eingeben, dann landet man immer wieder im Startverzeichnis des eigenen Benutzer-Ordners. Oder man gibt `pwd` ein, dann wird direkt der Pfad ausgegeben.

Hallo Welt!

Um „Hello World“ testen zu können, kann man den SD-Karten-Ordner aufsuchen und die Datei einfach direkt hineinkopieren:

```
cp ~/hello ./
```

Anschließend muss man das Verzeichnis manuell aushängen, damit das Betriebssystem alle noch nicht auf die Speicherkarte geschriebenen Blöcke endgültig überträgt.

```
sudo umount /media/<Verzeichnis-Name>
```

„<Verzeichnis-Name>“ ist jeweils der Name des Ordners, den das Betriebssystem für die SD-Karte vergeben hat.
Jetzt kann man die SD-Karte wieder in das Board stecken und von dieser booten. Wir verbinden uns anschließend von einem PC-Terminalprogramm aus mit dem Board, so wie im letzten Teil beschrieben [5].
Mit Hilfe des Terminalprogramms wechseln wir auf dem Board in das Hauptverzeichnis des Dateisystems:

```
cd /
```

und rufen unser Programm auf:


```
./hello
```

Als Ausgabe sollte nun „Hello World!“ im Terminalprogramm erscheinen (siehe **Bild 11**).

Wenn von der SD-Karte gebootet wird, ist es wichtig, dass man das System nach getaner Arbeit auch korrekt herunterfährt. Hierfür ist der Befehl

```
halt
```

zuständig. Dann muss man warten, bis tatsächlich `System halted` ausgegeben wurde, sonst besteht die Gefahr, dass die Dateisystemtabelle der SD-Karte zerstört wird. Wenn dies passiert ist, erscheinen typischerweise die „EXT2-fs errors“. Mit einem Linux-PC (so wie wir ihn hier installiert haben) kann man das Dateisystem aber wiederherstellen, siehe Kasten.

Bootloader und Kernel

Jetzt können wir uns an die Übersetzung der Betriebssystemkomponenten Bootloader und Kernel machen.

Die notwendigen Quelltexte (aktuell 290 MB) findet man auf der Elektor-Website [3]. Am einfachsten lädt man die Files mit einem Browser herunter, unter Linux liegen diese anschließend meist im Ordner „Downloads“.

Nach dem Download der Datei sollte man in den Ordner „Downloads“ wechseln und dort die Datei „120026-11.zip“ finden. Die Datei kann einfach in das eigene Benutzer-Verzeichnis verschoben werden:

```
mv ~/Downloads/120026-11.zip ~/
```

Um die Datei zu entpacken, wechselt man mit `cd` in das eigene Benutzer-Verzeichnis und gibt ein:

```
unzip 120026-11.zip
```

Die nun folgende Ausgabe auf der Konsole ist in **Bild 12** zu sehen.

Bootloader erstellen

Der Bootloader wird nach dem Einschalten des Prozessors von der SD-Karte in das interne SRAM des LPC3131 kopiert (wenn die Jumper entsprechend gesetzt sind, siehe [5]). Nachfolgend übersetzen wir diesen Bootloader selbst und übertragen ihn auf die SD-Karte, so dass von dort gebootet werden kann. Bevor es losgeht, müssen noch zwei Pakete in das Ubuntu-Linux installiert werden:

```
sudo apt-get install patch libncurses5-dev
```

Jetzt wechselt man in den Ordner für die Quelltexte ...

```
root@gnublin:~# cd /
root@gnublin:/# ./hello
Hello World!
root@gnublin:/#
```

Bild 11. „Hello World“ auf dem Board.

```
cd  ElektorLinuxBoardDownload_20120509
```

... und entpackt das tar-Archiv, in dem sich der Bootloader befindet:

```
tar xvzf bootloader.tar.gz
```

Anschließend wechselt man in den neuen Ordner „bootloader“ ...

```
cd bootloader
```

... und entpackt den eigentlichen Quelltext:

```
tar xvzf apex-1.6.8.tar.gz
```

Jetzt kommt ein wesentlicher Schritt für das spätere Weitergeben von Änderungen. Wir erzeugen uns sozusagen eine „Arbeitskopie“ vom Original-Quelltext und werden alle unsere Änderungen dort machen. So könnten wir später auch einfach einen Patch erzeugen und veröffentlichen.

```
mv apex-1.6.8 work_1.6.8

cd work_1.6.8
```

In unseren Quelltextbaum müssen wir anschließend die Änderungen einspielen, die für das Elektor-Linux-Board nötig sind:

```
patch -p1 < ../apex-1.6.8_lpc313x.patch
patch -p1 < ../gnublin-apex-1.6.8.patch
```

Das Übersetzen des Bootloaders wird durch sogenannte Konfigurationsdateien gesteuert. Wir müssen die Vorlage dieser Datei in den Ordner kopieren, und zwar mit dem neuen Namen „.config“ (auch hier ist der Punkt von dem Wort „config“ sehr wichtig, er kennzeichnet die Datei als „versteckte“ Datei im Betriebssystem).

```
cp ../gnublin-apex-1.6.8.config .config
```

Für diejenigen, die die Toolchain selbst installiert haben:
Falls man die Umgebungsvariablen noch nicht automatisch durch die „.bashrc“ gesetzt hat, muss man nun die selbst erstellte Skript-Datei vor dem Übersetzen mit `. ~/set.sh` aufrufen.

Nun startet man den Übersetzungsvorgang:

```
make apex.bin
```

Jetzt würde normalerweise die Stelle kommen, an der man die Firmware (Bootloader) mit einem Programmer in den Flashspeicher des Mikrocontrollers überträgt. In unserem Fall können wir die Firmware ganz einfach mit Standard-Linux-Programmen auf die SD-Karte schreiben.
Die SD-Karte wird dazu wieder in einen Kartenleser am PC gesteckt. Mit dem Befehl

```
dmesg
```

kann man sich einmal die Meldungen des Kernels ansehen. Auf dem PC des Autors sieht dies beispielsweise so aus wie in **Bild 13**. Man sieht dort, dass die SD-Karte als „/dev/sdh" erkannt wurde und zwei Partitionen „sdh1" und „sdh2" enthält. Am besten hängt man die Geräte sicherheitshalber nochmal aus, da wir die Firmware nicht über das Dateisystem, sondern direkt auf den entsprechenden Block der SD-Karte schreiben werden:

```
sudo umount /dev/sdh1
sudo umount /dev/sdh2
```

Mit folgender Zeile (muss man selbst noch anpassen!) kann man die „apex.bin"-Datei auf die SD-Karte schreiben, so dass sie der LPC3131 beim Booten findet:

```
sudo dd if=src/arch-arm/rom/apex.bin of=/dev/sdh2 bs=512
```

Mit der Zeile kopieren wir den Bootloader an den Anfang der zweiten Partition, mit einer Blockgröße von 512 Byte. Unter einem Betriebssystem wie Linux können wir nicht sagen, wann die geänderten Blöcke tatsächlich in den NAND-Speicher der SD-Karte geschrieben werden. Mit dem Befehl

```
sync
```

kann man das Betriebssystem aber zwingen, alle noch nicht geschriebenen Blöcke jetzt definitiv zu schreiben. Anschließend befindet sich der neue Bootloader sicher auf der SD-Karte. Gerne kann man nun einmal ein Test-Boot durchführen.

Kernel übersetzen

Auf ähnliche Weise wie der Bootloader lässt sich auch der Kernel übersetzen. Zuerst kommt der Wechsel in das Hauptverzeichnis des eigenen Benutzers mit `cd`, dann wechselt man in den Ordner für die Quelltexte...

```
cd  ElektorLinuxBoardDownload_20120509
```

... und entpackt den Quelltext des Kernels:

```
tar xvzf linux-2.6.33-lpc313x-gnublin-032012.tar.gz
```

Man wechselt in den Ordner des Kernels ...

```
cd linux-2.6.33-lpc3131x
```

... und startet einen Übersetzungslauf, um einen bootbaren Kernel zu erzeugen:

```
make zImage
```

Zusätzlich kann man nachladbare Treiber übersetzen. Dies macht man mit einem eigenen Aufruf:

```
make modules
```

Nach dem Übersetzen muss man den Kernel und die Module auf die SD-Karte kopieren. In unserem Fall liegt der Kernel „zImage" schon dort, doch sollte man den Kernel zu Übungszwecken einmal austauschen. Die Datei „arch/arm/boot/zImage" im Kernel-Quelltextbaum muss dazu direkt auf die erste Partition der SD-Karte geschrieben werden. Anschließend darf man das Aushängen der Partition nicht vergessen.

Schaffen Sie den Austausch des Kernels schon selbst?

Ein erster Überblick

Nun sind wir in der glücklichen Lage, den Bootloader und Kernel selbst übersetzen zu können. Das ermöglicht uns nun, auch Änderungen am Kernel vornehmen zu können. Natürlich sollte man sich hierzu zumindest grob auskennen. Einen ersten Blick in die Konfiguration des Linux-Kernels kann man mit

```
make menuconfig
```

wagen, den Befehl muss man im Ordner „linux-2.6.33-lpc3131x" aufrufen. Das Ergebnis sieht man in **Bild 14**. Möchte man z.B. ein bestimmtes USB-Gerät verwenden, muss man hier gezielt den Treiber aktivieren.
In der blauen Oberfläche kann man mit den Pfeil-Tasten navigieren. Mit der Enter-Taste kann man Menüs öffnen und wieder schließen. Wer hier auf Entdeckungstour geht, findet eventuell auch schon den einen oder anderen Treiber eines Gerätes, das er kennt. Im nächsten Artikel werden wir hier noch einmal näher darauf eingehen.

Boot-Image wiederherstellen

Da wir jetzt beginnen, unter der Motorhaube des Elektor-Linux-Boards zu arbeiten, empfiehlt es sich, vorher eine 1:1-Kopie der SD-Karte anzufertigen.

Nach dem Einlegen der Karte in den Kartenleser und dem Anstecken des Lesers am PC prüft man mit `dmesg`, was für ein Device-Name vom Betriebssystem vergeben wurde.
Sicherheitshalber sollte man nochmals die Partitionen aushängen:


```
umount /dev/sd<Buchstabe>1
umount /dev/sd<Buchstabe>2
```

(statt <Buchstabe> muss hier der vom Betriebssystem vergebene Buchstabe stehen, z.B. /dev/sdb1 und /dev/sdb2 oder /dev/sdh1 und /dev/sdh2).

Jetzt kann man sich die Karte 1:1 in eine Datei „dumpen":

```
sudo dd if=/dev/sd<Buchstabe> of=Image_SD_Karte_
Sicherheitskopie.img
```

Jetzt dauert es einige Zeit. Schließlich findet man mit folgendem Befehl eine Datei im aktuellen Ordner, die so groß ist wie die Karte:

```
ls -lh
```

Jetzt benötigt man eine gleich große SD-Karte, die man in den Kartenleser steckt. Mittels `dmesg` muss man wieder den Gerätenamen ermitteln.
Mit folgendem Befehl schreibt man das Karten-Image auf die neue Karte:

```
sudo dd if=Image_SD_Karte_Sicherheitskopie.img of=/dev/
sd<Buchstabe>
```

Jetzt dauert es nochmal etwas länger als vorhin. Nach einer Erfolgsmeldung sollte man sicherheitshalber `sync` aufrufen, so dass wirklich alle Blöcke auf die SD-Karte geschrieben werden. Daraufhin kann man die Karte zum Testen in das Elektor-Linux-Board stecken. Ein vorheriges `umount` ist hier übrigens unnötig, da das Dateisystem ja nicht eingehängt war (wir haben direkt die Gerätedatei beschrieben).

So einfach kann man sich Sicherheitskopien erstellen, doch leider versagt die Methode bei unterschiedlichen Kartengrößen. Und manchmal möchte man sich auch eine Karte erstellen, die andere Partitionen und ein anderes Dateisystem als die Originalkarte besitzt. Für diese Fälle gibt es einen grafischen Installer, der im nächsten Artikel kurz vorgestellt wird.

Ausblick

Mit diesem Teil der Serie sind wir auf dem Weg zur eigenen Embedded-GNU/Linux-Anwendung ein gutes Stück vorangekommen. Die Entwicklungsumgebung steht, der Bootloader und Kernel wurden übersetzt und ein kleines Programm haben wir auch selbst geschrieben und laufen lassen.

In der nächsten Runde werden wir uns ein wenig mit der Struktur des Quelltextes von Linux beschäftigen, so dass wir zumindest in der Lage sind, einen eigenen Treiber für ein beliebiges Gerät zu integrieren. Außerdem werden wir zeigen, wie einfach mit Skriptsprachen programmiert werden kann!

(120180)

```
Archive:  ElektorLinuxBoardDownload_20120509.zip
   creating: ElektorLinuxBoardDownload_20120509/
 extracting: ElektorLinuxBoardDownload_20120509/bootloader.tar.gz
  inflating: ElektorLinuxBoardDownload_20120509/gnublin-installer-1.3-beta-bin.tar.gz
  inflating: ElektorLinuxBoardDownload_20120509/gnublin-installer-1.3-beta-i386.deb
  inflating: ElektorLinuxBoardDownload_20120509/linux-2.6.33-lpc313x-gnublin-032012.tar.gz
  inflating: ElektorLinuxBoardDownload_20120509/rootfs.tar.gz
```

Bild 12. Ausgabe beim Entpacken des Software-Downloads.

```
[ 1148.953837] sd 5:0:0:2: [sdh] Assuming drive cache: write through
[ 1148.955001]  sdh: sdh1 sdh2
[ 1148.958583] sd 5:0:0:3: [sdi] Attached SCSI removable disk
[ 1148.959202] sd 5:0:0:2: [sdh] Assuming drive cache: write through
```

Bild 13. An der Kernel-Meldung sieht man, dass die SD-Karte als „/dev/**sdh**" erkannt wurde.

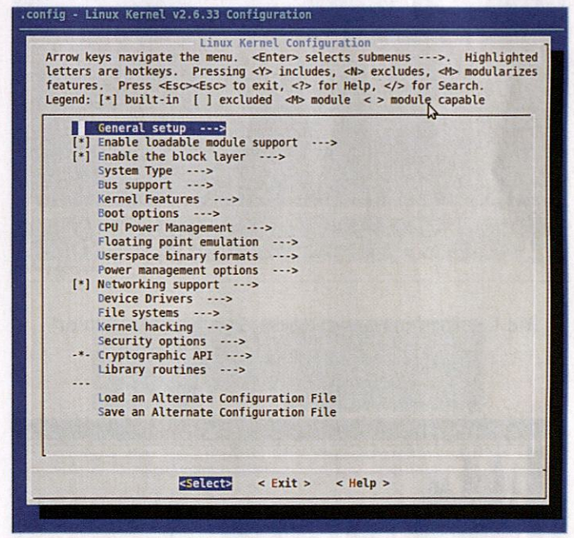

Bild 14. Konfiguration des Linux-Kernels.

Weblinks

[1] sauter@embedded-projects.net
[2] www.ubuntu.com
[3] www.elektor.de/120180
[4] www.virtualbox.org
[5] www.elektor.de/120146