1 Einleitung

Im Bereich eingebetteter Systeme ist Linux weit verbreitet und eine feste Größe. In Kombination mit der preiswerten Embedded-Plattform Raspberry Pi bildet es ein optimales Gespann, um Kenntnisse und Techniken, die für die Entwicklung eingebetteter Systeme notwendig sind, nachvollziehbar und praxisorientiert zu vermitteln.

Das als Einführung in das Thema gedachte Buch beschreibt den Aufbau, die Konzeption und die Realisierung eingebetteter Linux-Systeme auf Basis des Raspberry Pi.

Es demonstriert, wie als Teil einer Host-/Target-Entwicklung auf einem Linux-Hostsystem eine (Cross-)Toolchain installiert wird, um damit lauffähigen Code für die Zielplattform zu erzeugen. Es zeigt, aus welchen Komponenten die Systemsoftware besteht und wie sie für den spezifischen Einsatz konfektioniert und zu einem funktionstüchtigen Embedded System zusammengebaut wird. Dieser Vorgang wird in seinen Einzelschritten (from scratch) ebenso beschrieben wie die Automatisierung mithilfe des Systembuilders »Buildroot«. Das Buch führt außerdem in die softwaretechnische Ankopplung von Hardware ein, stellt dazu aktuelle Applikationsschnittstellen vor und demonstriert die Programmierung einfacher Linux-Treiber, beispielsweise für den Zugriff auf GPIOs. Tipps und Tricks, wie beispielsweise zur Erreichung kurzer Entwicklungszyklen durch ein Booten über ein Netzwerk, runden das Thema ebenso ab wie ein Abschnitt über die Sicherheit (Embedded Security) im Umfeld eingebetteter Systeme.

Ein beispielhaftes Projekt zum Abschluss zeigt die vorgestellten Techniken im Verbund.

Ziel des Buches ist es,

- eine praxisorientierte, kompakte Einführung in Embedded Linux zu geben und die Unterschiede zur Entwicklung bei einem Standardsystem aufzuzeigen,
- anhand eines von Grund auf selbst gebauten Linux-Systems den internen Aufbau und die Abläufe nachvollziehbar vorzustellen,
- exemplarisch eine einfache, auf Linux basierende Cross-Entwicklungsumgebung für eingebettete Systeme vorzustellen und damit

Bootloader, Kernel und Rootfilesystem für den Raspberry Pi zu erstellen,

- Grundkenntnisse für den Hardwarezugriff und die zugehörige Treibererstellung zu vermitteln,
- die Limitierungen und Besonderheiten bei der Erstellung von Applikationen für eingebettete Systeme vorzustellen,
- die Anforderungen an Security zu verdeutlichen und geeignete Techniken zur Realisierung mitzugeben.

Nicht thematisiert werden unter anderem die Portierung des Linux-Kernels auf eine neue Hardwareplattform, grafische Benutzerinterfaces, Realzeiterweiterungen wie der RT-PREEMPT Patch, Adeos/Xenomai oder Rtai und die Verwendung von integrierten Entwicklungsumgebungen (IDE).

Zielgruppen

Entwickler

Das Buch richtet sich damit an Entwickler, die in das Thema Embedded Linux neu einsteigen oder als Umsteiger bisher mit anderen eingebetteten Systemen Erfahrungen gesammelt haben. Nach der Lektüre kennen sie die Vorteile und die Eigenheiten eines Embedded Linux ebenso wie die Problembereiche. Neben dem Gesamtüberblick lernen sie das Treiberinterface und die wichtigsten Anwendungsschnittstellen kennen. Sie sind in der Lage, eine Entwicklungsumgebung aufzusetzen, die notwendigen Komponenten auszuwählen, zu konfigurieren und schließlich automatisiert zu einem funktionierenden Gesamtsystem zusammenzufügen.

Studenten der technischen Informatik

Studenten der technischen Informatik erarbeiten mit diesem Buch praxisorientiert das allgemeine Grundlagenwissen über eingebettete Systeme und deren Entwicklung. Sie lernen die Komponenten und die Zusammenhänge im Detail kennen. Nach der Lektüre können sie eingebettete Systeme mit Linux planen und softwareseitig realisieren. Über vollständige Anleitungen sammeln sie die ersten praktischen Erfahrungen, wobei der Einstieg in die praktische Umsetzung über die im Anhang zu findenden Crashkurse erleichtert wird.

Hobbyisten

Hobbyisten finden ein Mitmach-Buch vor, das ihnen hilft, die Möglichkeiten des Raspberry Pi auszuschöpfen. Die auf dem Webserver zur Verfügung gestellten Komponenten erleichtern dabei den Aufbau und die Fehlersuche und helfen bei der Überbrückung fachlicher Lücken.

Notwendige Voraussetzungen

Für die Lektüre des Buches sind Grundkenntnisse in den folgenden Bereichen sehr nützlich:

- Linux-Systemadministration
- C-Programmierung
- Rechnerhardware
- Ankopplung und Ansteuerung von Hardware
- Raspberry Pi

Begriffe wie Shell, Editor, Compiler, Flash, Interrupt, Ethernet, tcp/ip, DHCP oder Hexziffern sollten Ihnen nicht ganz fremd sein. Sie finden die benötigten Grundkenntnisse übrigens auch in vielen Einsteigerbüchern zum Raspberry Pi.

Die Inhalte werden praxisorientiert vorgestellt, sodass diese mit einem Linux-Rechner als Host und einem Raspberry Pi als Target vom Leser nachgebaut werden können. Das Buch ist als Einführung (appetizer) gedacht und verweist zur vertiefenden Auseinandersetzung mit dem spannenden Thema auf weiterführende Quellen (Literatur).

Scope

Die Entwicklung eingebetteter Systeme findet typischerweise als sogenannte Host-/Target-Entwicklung statt. Als Hostsystem dient für dieses Buch ein Ubuntu 12.04 LTS in der 64-Bit-Variante, das auf einem Kernel in der Version 3.2.0 aufbaut (Abb. 1-1). Als Target wird ein Raspberry Pi Typ B mit 512 MByte Hauptspeicher und einer mindestens 2 GByte großen Flash-Karte eingesetzt. Hier kommt beim selbst gebauten System ein Kernel der Version 3.6.11 und der zur Zeit der Bucherstellung aktuellen Version 3.10.9 zum Einsatz. Ab und zu werden Anleihen auf ein vorgefertigtes System für den Raspberry Pi gemacht, auf Raspbian. Dieses wird in Version `2013-07-26-wheezy-raspbian` verwendet.

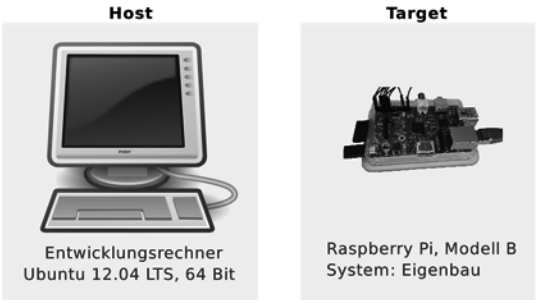

Abb. 1-1
Host-/Target-Entwicklung

Für das selbst gebaute System wird die sogenannten Busybox — ein Multicall-Binary — in der Version 1.21.1 eingesetzt. Der Systembuilder `buildroot` wird in der Version 2013.05 verwendet. Neuere Versionen der

Werkzeuge dürften typischerweise ebenfalls funktionieren. Als Emulator wird Qemu in der Version 1.0 benutzt.

Aufbau des Buches

Kapitel 2 und 3: Grundlagen

Die Einführung in das Thema Embedded Linux in Kapitel 2 beschäftigt sich mit dem grundlegenden Basiswissen. Dazu gehören die Architektur, der Entwicklungsprozess, aber auch Linux und der Raspberry Pi.

Darauf aufbauend wird im dritten Kapitel gezeigt, wie ein Kernel, ein Rootfilesystem und ein Bootloader von Grund auf (from Scratch) manuell generiert und zu einem eingebetteten System zusammengesetzt werden.

Kapitel 4: Systembuilder

Je mehr Funktionalität benötigt wird, desto komplexer wird der manuelle Aufbau eines Embedded System. Systemgeneratoren wie beispielsweise `buildroot`, der im vierten Kapitel vorgestellt wird, automatisieren und erleichtern diesen Vorgang.

Kapitel 5: Anwendungsentwicklung

Ein eingebettetes System erhält seine eigentliche Funktionalität erst durch eine Applikation. Bedingt durch die notwendige Cross-Entwicklung, die limitierten Ressourcen, die häufig vorkommenden Anforderungen an das Realzeitverhalten und die Interaktion mit Hardware, gibt es einige Einschränkungen und Besonderheiten bei der Applikationserstellung, die in Kapitel 5 zusammen mit relevanten Schnittstellen vorgestellt werden.

Kapitel 6: Gerätetreiber

Für die Ankopplung von proprietärer Hardware, zum effizienten und schnellen Einlesen von Sensoren und der Ausgabe von Signalen (Aktoren) werden eigene Gerätetreiber benötigt. Die dafür erforderlichen Grundlagen der Kernel- und Treiberprogrammierung beschreibt Kapitel 6.

Kapitel 7: Sicherheit

Unverzichtbar für jedes vernetzte, technische Gerät, ob als Produkt oder im privaten Bereich eingesetzt, sind grundlegende Mechanismen aus dem Bereich IT-Security. Hierzu gehört beispielsweise die Härtung des Systems durch eine Firewall, das Rechtemanagement oder der Entwurf sicherer Applikationen. Der »Embedded Security« ist Kapitel 7 gewidmet.

Kapitel 8: Ein komplettes Projekt

In Kapitel 8 wird als abgeschlossenes Projekt gezeigt, wie mithilfe der im Buch gewonnenen Erkenntnisse eine simple Messagebox aufgebaut werden kann. Diese zeigt beliebige Nachrichten an, die Sie per Webinterface an das Embedded Linux übermitteln.

Weitere Informationen zu den Themenbereichen Embedded Systems und Embedded Linux

- Das von mir mitverfasste Buch *Linux-Treiber entwickeln* [QuKu2011b] beschreibt systematisch die Gerätetreiber- und Kernelprogrammierung. Im Kontext der eingebetteten Systeme sind, mit

vielen Codebeispielen untermauert, detailliert die wesentlichen kernelspezifischen Aspekte nachzulesen. Das Buch stellt damit eine Abrundung des Themas nach »unten« dar.

- Während das Treiberbuch vor allem die kernelspezifischen Aspekte erörtert, behandelt das ebenfalls von mir mitverfasste Buch *Moderne Realzeitsysteme kompakt — Eine Einführung mit Embedded Linux* [QuMä2012] anwendungs- und architekturspezifische Aspekte. Das Buch stellt damit eine Abrundung des Themas nach »oben« dar.
- Auf der Webseite von Free-Electrons [http://free-electrons.com] finden Sie qualitativ hochwertige Unterlagen und Tutorials rund um das Thema Embedded Linux. Die Macher der Seite bieten auch Schulungen an.
- Thomas Eißenlöffel beschreibt in seinem Buch *Embedded-Software entwickeln* [Eißenlöffel2012] die Grundlagen der Programmierung eingebetteter Systeme mit dem Schwerpunkt auf der Anwendungsentwicklung von Deeply Embedded Systems.
- Die Webseite [elinux.org] widmet sich dem Thema *Embedded Linux*. Hier finden sich neben allgemeinen Informationen auch sehr wertvolle, spezifische Angaben beispielsweise zum Raspberry Pi.

Weitere Informationen zum Raspberry Pi

- The MagPi. Das monatliche Magazin behandelt Themen rund um den Raspberry Pi. Die einzelnen Ausgaben sind kostenlos und lassen sich als PDF von der Webseite herunterladen. Die Artikel sind in englischer Sprache verfasst. Die verständlich geschriebenen Artikel eignen sich sehr gut für Anfänger. Weitere Informationen unter [http://www.themagpi.com/].
- Maik Schmidt zeigt in seinem Buch *Raspberry Pi* [Schmidt2014] alles, was zum Umgang mit der Himbeere notwendig ist. Dazu gehört die Installation eines Systems auf der SD-Karte, die Konfiguration von Standardapplikationen und der einfache Anschluss von Hardware. Im Anhang finden Sie eine Einführung in Linux.
- Zur Hardware des Raspberry Pi gibt es diverse Datenblätter, beispielsweise auch das Datenblatt *BCM2835 ARM Peripherals*, das die Register für den Peripheriezugriff beschreibt [bcm2835].
- Homepage der Standarddistribution für den Raspberry Pi: [http://www.raspberrypi.org]. Download eines Systemimages unter [http://www.raspberrypi.org/downloads].

- Für diejenigen, die sich mehr mit Hardware beschäftigen, ist das Werkzeug Fritzing interessant. Damit lassen sich sehr intuitiv Schaltpläne erstellen. Fritzing unterstützt den Raspberry Pi. Weiterführende Informationen unter [http://fritzing.org].

Verzeichnisbaum

Im Rahmen des Buches werden verschiedene eingebettete Systeme aufgebaut und im Emulator Qemu oder auf dem Raspberry Pi getestet. Die notwendigen Softwarekomponenten sind dabei folgendermaßen organisiert (Abb. 1-2):

Abb. 1-2
Ordnerstruktur zur Datenablage

Im Heimatverzeichnis wird das Verzeichnis `embedded/` angelegt. Darunter gibt es die Verzeichnisse `qemu/`, `raspi/`, `application/` und `driver/`. In `qemu/` wiederum findet sich ein Verzeichnis für den Kernel (`linux/`) und ein Verzeichnis für das Userland (`userland/`).

Das Verzeichnis `raspi/` ist etwas komplexer strukturiert. Hier werden wir zwei unterschiedliche Systeme konstruieren. Ein System — ähnlich dem für den Emulator — ist komplett von Grund auf in allen Komponenten selbst entwickelt. Die Komponenten hierfür finden Sie in den Verzeichnissen `linux/` und `userland/`. Das zweite System wird mithilfe des Systembuilders buildroot aufgesetzt. Alle Komponenten sind im zugehörigen Verzeichnis `buildroot-2013.05/` zu finden. Außerdem werden wir für den Raspberry Pi den Bootloader »Das U-Boot« generieren (Verzeichnis `bootloader/`). Im Ordner `firmware` finden sich die Dateien für den Original-Bootloader des Raspberry Pi. Außerdem gibt es mit

scripts/ noch ein Verzeichnis, in dem die eigenen Skripte für die Generierung der Komponenten abgelegt werden.

Der Code der im Rahmen des Buches vorgestellten Applikationen findet sich unter application/, Code für drei einfache Gerätetreiber unter driver/.

Eine genauere Aufschlüsselung der nachfolgenden Verzeichnisse erfolgt in den entsprechenden Kapiteln.

In den Beispielen hat der Entwicklungsrechner den Namen »felicia«, der für die Entwicklung eingesetzte Login lautet »quade«. Der Standardprompt (Eingabezeile) im Terminal des Entwicklungsrechners sieht damit in den Beispielen folgendermaßen aus:

```
quade@felicia:~>
```

Der Prompt auf dem Raspberry Pi besteht aus einem Doppelkreuz (»#«):

```
#
```

Für Ihre Umgebung müssen Sie den Rechnernamen »felicia« und den Loginnamen »quade« durch Ihre eigenen austauschen.

Normalerweise sind Kommandos zeilenorientiert. Passt ein Kommando einmal nicht in eine einzelne Zeile, darf es auch auf mehrere Zeilen verteilt werden. Dazu ist am Ende einer Zeile, zu der es eine Folgezeile gibt, ein umgekehrter Schrägstrich »\« zu setzen:

```
quade@felicia:~/embedded/raspi> mv \
   /media/boot/kernel.img /media/boot/kernel.img.org
```

Onlinematerial

Die Webseite zum Buch

Entwicklungen im Umfeld eingebetteter Systeme sind komplex und deshalb gibt es ganz unterschiedliche Ursachen für auftretende Fehler. Da fast immer mehrere Komponenten beteiligt sind, finden Sie auf der Webseite zum Buch ([https://ezs.kr.hsnr.de/EmbeddedBuch/]) wesentliche Entwicklungsergebnisse der einzelnen Kapitel wie Konfigurationsdateien, Skripte aber auch Imagedateien. Mithilfe dieser Dateien können Sie den Fehler eingrenzen und typischerweise auf die schadhafte Komponente reduzieren.