Developing Embedded Linux Devices Using The Yocto Project

Right here, we have countless book **developing embedded linux devices using the yocto project** and collections to check out. We additionally come up with the money for variant types and after that type of the books to browse. The welcome book, fiction, history, novel, scientific research, as capably as various further sorts of books are readily to hand here.

As this developing embedded linux devices using the yocto project, it ends in the works physical one of the favored books developing embedded linux devices using the yocto project collections that we have. This is why you remain in the best website to see the incredible books to have.

Designing \u0026 manufacturing a custom embedded linux machine.

Phil Wise - Beyond Raspbian: Building Embedded Linux DevicesEmbedded Linux Device Tree and Platform Devices #04 *Scaling Embedded Linux Devices from Prototype to Production Embedded Linux with FPGA Device Drivers Basic #03* **Developing Embedded Linux Devices Using the Yocto Project and What's new in 1.1 - ELCE 2011** ~~Linux System Programming 6 Hours Course~~

Virtual Embedded Linux Development Computer~~Linux Training Course: Building Embedded Linux with the Yocto Project~~

How to Get Started Learning Embedded Systems~~How Do Linux Kernel Drivers Work? - Learning Resource~~ ~~Tutorial: Debugging Embedded Devices using GDB - Chris Simmonds, 2net Ltd~~ *Buildroot Tutorial- Linux Kernel on QEMU Virtual board - Booting Linux and Running Linux Application* ~~Preempt-RT Raspberry Pi Linux - Tiejun Chen, VMware~~ ~~Lecture 15: Booting Process~~ *Introduction to Realtime Linux* ~~What is a kernel - Gary explains~~ Linux Device Drivers Training 01, Simple Loadable Kernel Module ~~Technical Session 8.2 | Free DEMO Training on Linux BSP Kernel Porting on ARM BOARD~~ ~~Introduction to Linux~~ *Linux Device Tree*

Linux Device Drivers Training 06, Simple Character Driver~~Embedded Linux Introduction #01~~ ~~Arm Education Media – Embedded Linux - Online Course~~ Beaglebone: C/C++ Programming Introduction for ARM Embedded Linux Development using Eclipse CDT *How to Avoid Writing Device Drivers for Embedded Linux - Chris Simmonds, 2net*

New course : Linux device driver programming~~Debian C/C++ Cross Compilation for Embedded Linux using Eclipse (Luna), CDT, RSE \u0026 Remote Debug~~ ~~Embedded Linux \"from scratch\" in 45 minutes...on RISC V~~ ~~Embedded Linux Explained!~~ *Developing Embedded Linux Devices Using*

Developing Embedded Linux Systems. Jason Sando. Mar 31, 2019 · 11 min read. I ’ ve spent quite a bit of time in the last 10+ years shipping embedded Linux devices, and thought I ’ d do a write up ...

Developing Embedded Linux Systems ¦ by Jason Sando ¦ Medium

1. Go to http://yoctoproject.org, click “ documentation ” and consult the Quick Start guide 2. Set up your Linux system with the right packages (and firewall access, if needed) 3. Click “ Download ” and download the latest stable release (or check out “ bernard ” from the

git repo) 4.

Developing Embedded Linux Devices Using the Yocto Project™
It ' s not an embedded Linux distribution – it creates a custom one for you. YP lets you customize your embedded Linux OS. YP helps set up the embedded app developer. Both device and app development models supported. Getting started is easy. Make an impact – collaboration in its purest sense /30

Developing Embedded Linux Devices Using the Yocto Project™
The Eclipse-based TimeStorm IDE provides Windows 10 OS users with an already familiar development environment, making it easy to develop embedded Linux products within a Windows environment. This feature makes it an ideal solution for developers who want to migrate from microcontroller development to the development of microprocessor-based Linux devices.

Ready to tackle embedded Linux MPU development with ...
Developing Embedded Linux Device Drivers (LFD435) This instructor-led course is designed to show experienced programmers how to develop device drivers for embedded Linux systems, and give them a basic understanding and familiarity with the Linux kernel.

Developing Embedded Linux Device Drivers (LFD435) - Linux ...
Embedded Linux Development (LFD450) This instructor-led course will give you the step-by-step framework for developing an embedded Linux product. You ' ll learn the methods used to adapt the Linux kernel and user-space libraries and utilities to particular embedded environments, such as those in use in consumer electronics, military, medical, industrial, and auto industries.

Embedded Linux Development (LFD450) - Linux Foundation ...
Presentation entitled " Developing Embedded Linux Devices Using the Yocto Project and What ' s new in 1.1 " by David Stewart, Intel, at Embedded Linux Conference Europe 2011. Abstract: The Yocto Project is a joint project to unify the world ' s efforts around embedded Linux and to make Linux the best choice for embedded designs. The Yocto Project is an open source starting point for embedded Linux development which contains tools, templates, methods and actual working code to get started ...

Developing Embedded Linux Devices Using the Yocto Project ...
Hands-on/Lecture. Download the Complete Course Syllabus. Whether you are developing Linux device drivers for unsupported peripherals or writing a board support package (BSP) to port the operating system to custom embedded hardware, there's a steep learning curve. Through a mix of lectures and hands-on programming exercises on real hardware, this course will help you quickly move on to developing your own Linux driver code.

Embedded Linux Customization and Driver Development

Linux continues to be the leading choice for embedded device operating systems but the decision to choose Linux for use in a medical device setting includes the additional considerations of patient...

Using Linux in Medical Devices - embedded-computing.com
A proof of concept using AndroidXML and TotalCross provides an easier way of creating UIs for Raspberry Pi and other devices. Creating a great user experience (UX) for your applications is a tough job, especially if you are developing embedded applications.

A new way to build cross-platform UIs for Linux ARM devices
Key Features Learn to develop customized Linux device drivers Learn the core concepts of device drivers such as memory management, kernel caching, advanced IRQ management, and so on. Practical experience on the embedded side of LinuxBook Description Linux kernel is a complex, portable, modular and widely used piece of software, running on around 80% of servers and embedded systems in more than ...

Device Drivers Development For Embedded Linux - Copperhill
We can apply the same concept when developing an embedded Linux device! In the end, there is no such thing as a 100% secure system. An attacker needs only one flaw to compromise the device. It ' s just a matter of how hard and difficult we want this process to be. So we should design with security in mind, being aware of the trade-offs.

Introduction to Embedded Linux Security - part 2 - # ...
Introduction Embedded devices are running complex resource-intensive applications on edge. A preferred way to do so is to containerize them and then deploy on the remote IoT edge devices. This helps with better orchestration and resource planning of the applications. Docker is an open platform for developing, shipping, and running applications.

Deploy Docker Containers to Embedded Linux Devices ¦ Aikaan
Building an embedded medical device using the Texas Instruments Zoom™ OMAP35x Development Kit from Logic PD with LinuxLink This exciting hands-on webinar series will not only introduce you to fast Linux product development with Timesys tools, but it also will demonstrate how open source technology can be harnessed to build an embedded medical device using one of the powerful OMAP-3530 processors from TI.

Embedded Linux Webinars ¦ Timesys Embedded Linux
The host development system is a standard PC running Linux. We use the target as an example of a modern embedded system which can control and interact with many available interfaces including USB. Lab sessions follow a logical sequence, and result in a Linux-powered web-controlled rocket launcher. Introduction.

Developing for Embedded Linux ¦ Feabhas

Presentation entitled “ Developing Embedded Linux Devices Using the Yocto Project and What ’ s new in 1.1 ” by David Stewart, Intel, at Embedded Linux Conference Europe 2011. Abstract: The Yocto Project is a joint project to unify the world ’ s efforts around embedded Linux and to make Linux the best choice for embedded designs.

ppc News - CNX Software - Embedded Systems News
For StrongARM-based Linux devices, a kernel module that uses USB calls sa1100_usb_open () to initialize kernel code that manages the chip's onboard USB device controller peripheral. The module then invokes sa1100_usb_get_descriptor_ptr () and sa1100_usb_set_string_descriptor () to set the USB descriptors given to a USB host during enumeration.

Linux-based USB Devices - Embedded.com
Drew Moseley - Drew is currently part of the Mender.io open source project to deploy OTA software updates to embedded Linux devices. He has worked on embedded projects such as RAID storage controllers, Direct and Network attached storage devices and graphical pagers. He has spent the last 7 years working in Operating System Professional Services helping customers develop production embedded Linux systems.

Choosing the right model for maintaining and enhancing ...
There are a wide variety of distribution and build systems you can use to develop your embedded Linux system. Many desktop distributions can be pared down for use in limited resource environment and systems such as Ubuntu have varieties specifically targeted at IoT devices. The Raspberry Pi platform uses a customized Debian image as its primary target OS image.

Linux® is being adopted by an increasing number of embedded systems developers, who have been won over by its sophisticated scheduling and networking, its cost-free license, its open development model, and the support offered by rich and powerful programming tools. While there is a great deal of hype surrounding the use of Linux in embedded systems, there is not a lot of practical information. Building Embedded Linux Systems is the first in-depth, hard-core guide to putting together an embedded system based on the Linux kernel. This indispensable book features arcane and previously undocumented procedures for: Building your own GNU development toolchain Using an efficient embedded development framework Selecting, configuring, building, and installing a target-specific kernel Creating a complete target root filesystem Setting up, manipulating, and using solid-state storage devices Installing and configuring a bootloader for the target Cross-compiling a slew of utilities and packages Debugging your embedded system using a plethora of tools and techniques Details are provided for various target architectures and hardware configurations, including a thorough review of Linux's support for embedded hardware. All explanations rely on the use of open source and free software packages. By presenting how to build the operating system components from pristine sources and how to find more documentation or help, this book greatly simplifies the task of keeping complete control over one's embedded operating system, whether it be for technical or sound financial reasons.Author Karim Yaghmour, a

well-known designer and speaker who is responsible for the Linux Trace Toolkit, starts by discussing the strengths and weaknesses of Linux as an embedded operating system. Licensing issues are included, followed by a discussion of the basics of building embedded Linux systems. The configuration, setup, and use of over forty different open source and free software packages commonly used in embedded Linux systems are also covered. uClibc, BusyBox, U-Boot, OpenSSH, thttpd, tftp, strace, and gdb are among the packages discussed.

Leverage the power of Linux to develop captivating and powerful embedded Linux projects About This Book Explore the best practices for all embedded product development stages Learn about the compelling features offered by the Yocto Project, such as customization, virtualization, and many more Minimize project costs by using open source tools and programs Who This Book Is For If you are a developer who wants to build embedded systems using Linux, this book is for you. It is the ideal guide for you if you want to become proficient and broaden your knowledge. A basic understanding of C programming and experience with systems programming is needed. Experienced embedded Yocto developers will find new insight into working methodologies and ARM specific development competence. What You Will Learn Use the Yocto Project in the embedded Linux development process Get familiar with and customize the bootloader for a board Discover more about real-time layer, security, virtualization, CGL, and LSB See development workflows for the U-Boot and the Linux kernel, including debugging and optimization Understand the open source licensing requirements and how to comply with them when cohabiting with proprietary programs Optimize your production systems by reducing the size of both the Linux kernel and root filesystems Understand device trees and make changes to accommodate new hardware on your device Design and write multi-threaded applications using POSIX threads Measure real-time latencies and tune the Linux kernel to minimize them In Detail Embedded Linux is a complete Linux distribution employed to operate embedded devices such as smartphones, tablets, PDAs, set-top boxes, and many more. An example of an embedded Linux distribution is Android, developed by Google. This learning path starts with the module Learning Embedded Linux Using the Yocto Project. It introduces embedded Linux software and hardware architecture and presents information about the bootloader. You will go through Linux kernel features and source code and get an overview of the Yocto Project components available. The next module Embedded Linux Projects Using Yocto Project Cookbook takes you through the installation of a professional embedded Yocto setup, then advises you on best practices. Finally, it explains how to quickly get hands-on with the Freescale ARM ecosystem and community layer using the affordable and open source Wandboard embedded board. Moving ahead, the final module Mastering Embedded Linux Programming takes you through the product cycle and gives you an in-depth description of the components and options that are available at each stage. You will see how functions are split between processes and the usage of POSIX threads. By the end of this learning path, your capabilities will be enhanced to create robust and versatile embedded projects. This Learning Path combines some of the best that Packt has to offer in one complete, curated package. It includes content from the following Packt products: Learning Embedded Linux Using the Yocto Project by Alexandru Vaduva Embedded Linux Projects Using Yocto Project Cookbook by Alex Gonzalez Mastering Embedded Linux Programming by Chris Simmonds Style and approach This comprehensive, step-by-step, pragmatic guide enables you to build custom versions of Linux for new embedded systems with examples that are immediately applicable to your embedded developments. Practical examples provide an easy-to-follow way to learn Yocto project development using the best practices and working methodologies. Coupled with hints and best practices, this will help you understand embedded Linux better.

Harness the power of Linux to create versatile and robust embedded solutions Key Features: Learn how to develop and configure robust embedded Linux devices Explore the new features of Linux 5.4 and the Yocto Project 3.1 (Dunfell) Discover different ways to debug and profile your code in both user space and the Linux kernel Book Description: Embedded Linux runs many of the devices we use every day. From smart TVs and Wi-Fi routers to test equipment and industrial controllers, all of them have Linux at their heart. The Linux OS is one of the foundational technologies comprising the core of the Internet of Things (IoT). This book starts by breaking down the fundamental elements that underpin all embedded Linux projects: the toolchain, the bootloader, the kernel, and the root filesystem. After that, you will learn how to create each of these elements from scratch and automate the process using Buildroot and the Yocto Project. As you progress, the book explains how to implement an effective storage strategy for flash memory chips and install updates to a device remotely once it's deployed. You'll also learn about the key aspects of writing code for embedded Linux, such as how to access hardware from apps, the implications of writing multi-threaded code, and techniques to manage memory in an efficient way. The final chapters demonstrate how to debug your code, whether it resides in apps or in the Linux kernel itself. You'll also cover the different tracers and profilers that are available for Linux so that you can quickly pinpoint any performance bottlenecks in your system. By the end of this Linux book, you'll be able to create efficient and secure embedded devices using Linux. What You Will Learn: Use Buildroot and the Yocto Project to create embedded Linux systems Troubleshoot BitBake build failures and streamline your Yocto development workflow Update IoT devices securely in the field using Mender or balena Prototype peripheral additions by reading schematics, modifying device trees, soldering breakout boards, and probing pins with a logic analyzer Interact with hardware without having to write kernel device drivers Divide your system up into services supervised by BusyBox runit Debug devices remotely using GDB and measure the performance of systems using tools such as perf, ftrace, eBPF, and Callgrind Who this book is for: If you're a systems software engineer or system administrator who wants to learn Linux implementation on embedded devices, then this book is for you. Embedded systems engineers accustomed to programming for low-power microcontrollers can use this book to help make the leap to high-speed systems on chips that can run Linux. Anyone responsible for developing new hardware that needs to run Linux will also find this book useful. Basic working knowledge of the POSIX standard, C programming, and shell scripting is assumed.

Learn to develop customized device drivers for your embedded Linux system About This Book Learn to develop customized Linux device drivers Learn the core concepts of device drivers such as memory management, kernel caching, advanced IRQ management, and so on. Practical experience on the embedded side of Linux Who This Book Is For This book will help anyone who wants to get started with developing their own Linux device drivers for embedded systems. Embedded Linux users will benefit highly from this book. This book covers all about device driver development, from char drivers to network device drivers to memory management. What You Will Learn Use kernel facilities to develop powerful drivers Develop drivers for widely used I2C and SPI devices and use the regmap API Write and support devicetree from within your drivers Program advanced drivers for network and frame buffer devices Delve into the Linux irqdomain API and write interrupt controller drivers Enhance your skills with regulator and PWM frameworks Develop measurement system drivers with IIO framework Get the best from memory management and the DMA subsystem Access and manage GPIO subsystems and develop GPIO

controller drivers In Detail Linux kernel is a complex, portable, modular and widely used piece of software, running on around 80% of servers and embedded systems in more than half of devices throughout the World. Device drivers play a critical role in how well a Linux system performs. As Linux has turned out to be one of the most popular operating systems used, the interest in developing proprietary device drivers is also increasing steadily. This book will initially help you understand the basics of drivers as well as prepare for the long journey through the Linux Kernel. This book then covers drivers development based on various Linux subsystems such as memory management, PWM, RTC, IIO, IRQ management, and so on. The book also offers a practical approach on direct memory access and network device drivers. By the end of this book, you will be comfortable with the concept of device driver development and will be in a position to write any device driver from scratch using the latest kernel version (v4.13 at the time of writing this book). Style and approach A set of engaging examples to develop Linux device drivers

Based upon the authors' experience in designing and deploying an embedded Linux system with a variety of applications, Embedded Linux System Design and Development contains a full embedded Linux system development roadmap for systems architects and software programmers. Explaining the issues that arise out of the use of Linux in embedded systems, the book facilitates movement to embedded Linux from traditional real-time operating systems, and describes the system design model containing embedded Linux. This book delivers practical solutions for writing, debugging, and profiling applications and drivers in embedded Linux, and for understanding Linux BSP architecture. It enables you to understand: various drivers such as serial, I2C and USB gadgets; uClinux architecture and its programming model; and the embedded Linux graphics subsystem. The text also promotes learning of methods to reduce system boot time, optimize memory and storage, and find memory leaks and corruption in applications. This volume benefits IT managers in planning to choose an embedded Linux distribution and in creating a roadmap for OS transition. It also describes the application of the Linux licensing model in commercial products.

Get up to speed with the most important concepts in driver development and focus on common embedded system requirements such as memory management, interrupt management, and locking mechanisms Key Features Write feature-rich and customized Linux device drivers for any character, SPI, and I2C device Develop a deep understanding of locking primitives, IRQ management, memory management, DMA, and so on Gain practical experience in the embedded side of Linux using GPIO, IIO, and input subsystems Book Description Linux is by far the most-used kernel on embedded systems. Thanks to its subsystems, the Linux kernel supports almost all of the application fields in the industrial world. This updated second edition of Linux Device Driver Development is a comprehensive introduction to the Linux kernel world and the different subsystems that it is made of, and will be useful for embedded developers from any discipline. You'll learn how to configure, tailor, and build the Linux kernel. Filled with real-world examples, the book covers each of the most-used subsystems in the embedded domains such as GPIO, direct memory access, interrupt management, and I2C/SPI device drivers. This book will show you how Linux abstracts each device from a hardware point of view and how a device is bound to its driver(s). You'll also see how interrupts are propagated in the system as the book covers the interrupt processing mechanisms in-depth and describes every kernel structure and API involved. This new edition also addresses how not to write device drivers using user space libraries for GPIO clients, I2C, and SPI drivers. By the end of this Linux book, you'll be able to write device drivers for most of the embedded devices out there. What you will learn Download,

configure, build, and tailor the Linux kernel Describe the hardware using a device tree Write feature-rich platform drivers and leverage I2C and SPI buses Get the most out of the new concurrency managed workqueue infrastructure Understand the Linux kernel timekeeping mechanism and use time-related APIs Use the regmap framework to factor the code and make it generic Offload CPU for memory copies using DMA Interact with the real world using GPIO, IIO, and input subsystems Who this book is for This Linux OS book is for embedded system and embedded Linux enthusiasts/developers who want to get started with Linux kernel development and leverage its subsystems. Electronic hackers and hobbyists interested in Linux kernel development as well as anyone looking to interact with the platform using GPIO, IIO, and input subsystems will also find this book useful.

Master the art of developing customized device drivers for your embedded Linux systems Key Features Stay up to date with the Linux PCI, ASoC, and V4L2 subsystems and write device drivers for them Get to grips with the Linux kernel power management infrastructure Adopt a practical approach to customizing your Linux environment using best practices Book Description Linux is one of the fastest-growing operating systems around the world, and in the last few years, the Linux kernel has evolved significantly to support a wide variety of embedded devices with its improved subsystems and a range of new features. With this book, you'll find out how you can enhance your skills to write custom device drivers for your Linux operating system. Mastering Linux Device Driver Development provides complete coverage of kernel topics, including video and audio frameworks, that usually go unaddressed. You'll work with some of the most complex and impactful Linux kernel frameworks, such as PCI, ALSA for SoC, and Video4Linux2, and discover expert tips and best practices along the way. In addition to this, you'll understand how to make the most of frameworks such as NVMEM and Watchdog. Once you've got to grips with Linux kernel helpers, you'll advance to working with special device types such as Multi-Function Devices (MFD) followed by video and audio device drivers. By the end of this book, you'll be able to write feature-rich device drivers and integrate them with some of the most complex Linux kernel frameworks, including V4L2 and ALSA for SoC. What you will learn Explore and adopt Linux kernel helpers for locking, work deferral, and interrupt management Understand the Regmap subsystem to manage memory accesses and work with the IRQ subsystem Get to grips with the PCI subsystem and write reliable drivers for PCI devices Write full multimedia device drivers using ALSA SoC and the V4L2 framework Build power-aware device drivers using the kernel power management framework Find out how to get the most out of miscellaneous kernel subsystems such as NVMEM and Watchdog Who this book is for This book is for embedded developers, Linux system engineers, and system programmers who want to explore Linux kernel frameworks and subsystems. C programming skills and a basic understanding of driver development are necessary to get started with this book.

Up-to-the-Minute, Complete Guidance for Developing Embedded Solutions with Linux Linux has emerged as today ' s #1 operating system for embedded products. Christopher Hallinan ' s Embedded Linux Primer has proven itself as the definitive real-world guide to building efficient, high-value, embedded systems with Linux. Now, Hallinan has thoroughly updated this highly praised book for the newest Linux kernels, capabilities, tools, and hardware support, including advanced multicore processors. Drawing on more than a decade of embedded Linux experience, Hallinan helps you rapidly climb the learning curve, whether you ' re moving from legacy environments or you ' re new to embedded programming. Hallinan addresses today ' s most important development challenges and demonstrates how to solve the problems you ' re most likely to encounter. You ' ll learn how to build a modern, efficient embedded Linux development environment, and

then utilize it as productively as possible. Hallinan offers up-to-date guidance on everything from kernel configuration and initialization to bootloaders, device drivers to file systems, and BusyBox utilities to real-time configuration and system analysis. This edition adds entirely new chapters on UDEV, USB, and open source build systems. Tour the typical embedded system and development environment and understand its concepts and components. Understand the Linux kernel and userspace initialization processes. Preview bootloaders, with specific emphasis on U-Boot. Configure the Memory Technology Devices (MTD) subsystem to interface with flash (and other) memory devices. Make the most of BusyBox and latest open source development tools. Learn from expanded and updated coverage of kernel debugging. Build and analyze real-time systems with Linux. Learn to configure device files and driver loading with UDEV. Walk through detailed coverage of the USB subsystem. Introduces the latest open source embedded Linux build systems. Reference appendices include U-Boot and BusyBox commands.

Master the techniques needed to build great, efficient embedded devices on Linux About This Book Discover how to build and configure reliable embedded Linux devices This book has been updated to include Linux 4.9 and Yocto Project 2.2 (Morty) This comprehensive guide covers the remote update of devices in the field and power management Who This Book Is For If you are an engineer who wishes to understand and use Linux in embedded devices, this book is for you. It is also for Linux developers and system programmers who are familiar with embedded systems and want to learn and program the best in class devices. It is appropriate for students studying embedded techniques, for developers implementing embedded Linux devices, and engineers supporting existing Linux devices. What You Will Learn Evaluate the Board Support Packages offered by most manufacturers of a system on chip or embedded module Use Buildroot and the Yocto Project to create embedded Linux systems quickly and efficiently Update IoT devices in the field without compromising security Reduce the power budget of devices to make batteries last longer Interact with the hardware without having to write kernel device drivers Debug devices remotely using GDB, and see how to measure the performance of the systems using powerful tools such as perk, ftrace, and valgrind Find out how to configure Linux as a real-time operating system In Detail Embedded Linux runs many of the devices we use every day, from smart TVs to WiFi routers, test equipment to industrial controllers - all of them have Linux at their heart. Linux is a core technology in the implementation of the inter-connected world of the Internet of Things. The comprehensive guide shows you the technologies and techniques required to build Linux into embedded systems. You will begin by learning about the fundamental elements that underpin all embedded Linux projects: the toolchain, the bootloader, the kernel, and the root filesystem. You'll see how to create each of these elements from scratch, and how to automate the process using Buildroot and the Yocto Project. Moving on, you'll find out how to implement an effective storage strategy for flash memory chips, and how to install updates to the device remotely once it is deployed. You'll also get to know the key aspects of writing code for embedded Linux, such as how to access hardware from applications, the implications of writing multi-threaded code, and techniques to manage memory in an efficient way. The final chapters show you how to debug your code, both in applications and in the Linux kernel, and how to profile the system so that you can look out for performance bottlenecks. By the end of the book, you will have a complete overview of the steps required to create a successful embedded Linux system. Style and approach This book is an easy-to-follow and pragmatic guide with in-depth analysis of the implementation of embedded devices. It follows the life cycle of a project from inception through to completion, at each stage giving both the theory that underlies the topic and practical step-by-step walkthroughs of an example implementation.