Embedded Operating Systems A Practical Approach Undergraduate Topics In Computer Science

The aim of this book is to provide a practical introduction to the foundations of modern operating systems, with a particular focus on GNU/Linux and the Arm platform. The unique perspective of the authors is that they explain operating systems theory and concepts but also ground them in practical use through illustrative examples.

Embedded Systems Architecture is a practical and technical guide to understanding the components that make up an embedded system’s architecture. This book is perfect for those starting out as technical professionals such as engineers, programmers and designers of embedded systems; and also for students of computer science, computer engineering and electrical engineering. It gives a much-needed ‘big picture’ for recently graduated engineers grappling with understanding the design of real-world systems for the first time, and provides professionals with a systems-level picture of the key elements that can go into an embedded design, providing a firm foundation on which to build their skills. Real-world approach to the fundamentals, as well as the design and architecture process, makes this book a popular reference for the daunted or the inexperienced: if in doubt, the answer is in here! Fully updated with new coverage of FPGAs, testing, middleware and the latest programming techniques in C, plus complete source code and sample code, reference designs and tools online make this the complete package Visit the companion web site at http://booksite.elsevier.com/9780123821966/ for source code, design examples, data sheets and more A true introductory book, provides a comprehensive get up and running reference for those new to the field, and updating skills: assumes no prior knowledge beyond undergrad level electrical engineering Addresses the needs of practicing engineers, enabling it to get to the point more directly, and cover more ground. Covers hardware, software and middleware in a single volume Includes a library of design examples and design tools, plus a complete set of source code and embedded systems design tutorial materials from companion website

Build secure and auditable IoT applications by mastering the IoT stack through Tock Operating System. TockOS runs on low power devices that offer lower deployment and maintenance costs, making them an ideal part of every IoT system. Running secure and audited software on these devices results in the success of an entire system. This book shows IoT systems designers and integrators how to ensure that all the software components run with security guarantees. First, you'll explore the characteristics of TockOS and how to run it on ARM and RISC V platforms. You’ll also take a look at Rust and how to use it for building secure applications on top of TockOS. Next, you'll review the TockOS applications frameworks for C and Rust. And then move from using the simple TockOS APIs to the more complex Inter-Process Communication system that allows applications to expose and consume services. Further on, the book covers the internals of the TockOS kernel and presents the steps necessary to integrate new peripherals and hardware platforms. You'll write drivers, from simple to more complex, build a specific TockOS kernel image for your platform, and use communications busses to talk to peripherals and the cloud. By taking a practical approach, Getting Started with Secure Embedded Systems provides a complete view of the IoT stack based on the Tock Operating System. What You'll Learn Write applications and drivers for TockOS Customize the kernel for specific hardware platforms Run TockOS both on emulators and real hardware Set a solid base for building secure and auditable IoT applications Use TockOS to ensure the security of your microcontrollers and integrate it in your projects Who This Book Is For IoT system designers, developers and integrators who are familiar with operating systems concepts. The book can also be suitable for people with less experience, who want to gain an overview of the latest hardware and software technologies related to

building secure IoT systems.

Embedded Operating SystemsA Practical ApproachSpringer

This practical technical guide to embedded middleware implementation offers a coherent framework that guides readers through all the key concepts necessary to gain an understanding of this broad topic. Big picture theoretical discussion is integrated with down-to-earth advice on successful real-world use via step-by-step examples of each type of middleware implementation. Technically detailed case studies bring it all together, by providing insight into typical engineering situations readers are likely to encounter. Expert author Tammy Noergaard keeps explanations as simple and readable as possible, eschewing jargon and carefully defining acronyms. The start of each chapter includes a "setting the stage" section, so readers can take a step back and understand the context and applications of the information being provided. Core middleware, such as networking protocols, file systems, virtual machines, and databases; more complex middleware that builds upon generic pieces, such as MOM, ORB, and RPC; and integrated middleware software packages, such as embedded JVMs, .NET, and CORBA packages are all demystified. Embedded middleware theory and practice that will get your knowledge and skills up to speed Covers standards, networking, file systems, virtual machines, and more Get hands-on programming experience by starting with the downloadable open source code examples from book website

'... a very good balance between the theory and practice of real-time embedded system designs.' —Jun-ichiro itojun Hagino, Ph.D., Research Laboratory, Internet Initiative Japan Inc., IETF IPv6 Operations Working Group (v6ops) co-chair 'A cl

Until the late 1980s, information processing was associated with large mainframe computers and huge tape drives. During the 1990s, this trend shifted toward information processing with personal computers, or PCs. The trend toward miniaturization continues and in the future the majority of information processing systems will be small mobile computers, many of which will be embedded into larger products and interfaced to the physical environment. Hence, these kinds of systems are called embedded systems. Embedded systems together with their physical environment are called cyber-physical systems. Examples include systems such as transportation and fabrication equipment. It is expected that the total market volume of embedded systems will be significantly larger than that of traditional information processing systems such as PCs and mainframes. Embedded systems share a number of common characteristics. For example, they must be dependable, efficient, meet real-time constraints and require customized user interfaces (instead of generic keyboard and mouse interfaces). Therefore, it makes sense to consider common principles of embedded system design. Embedded System Design starts with an introduction into the area and a survey of specification models and languages for embedded and cyber-physical systems. It provides a brief overview of hardware devices used for such systems and presents the essentials of system software for embedded systems, like real-time operating systems. The book also discusses evaluation and validation techniques for embedded systems. Furthermore, the book presents an overview of techniques for mapping applications to execution platforms. Due to the importance of resource efficiency, the book also contains a selected set of optimization techniques for embedded systems, including special compilation techniques. The book closes with a brief survey on testing. Embedded System Design can be used as a text book for courses on embedded systems and as a source which provides pointers to relevant material in the area for PhD students and teachers. It assumes a basic knowledge of information processing hardware and software. Courseware related to this book is available at http://ls12-www.cs.tu-dortmund.de/~marwedel.

For the Students of B.E. / B.Tech., M.E. / M.Tech. & BCA / MCA It is indeed a matter of great encouragement to write the Third Edition of this book on ';Operating Systems - A Practical Approach' which covers the syllabi of B.Tech./B.E. (CSE/IT), M.Tech./M.E. (CSE/IT),

BCA/MCA of many universities of India like Delhi University, GGSIPU Delhi, UPTU Lucknow, WBUT, RGPV, MDU, etc.

There's something really satisfying about turning theory into practice, bringing with it a great feeling of accomplishment. Moreover it usually deepens and solidifies your understanding of the theoretical aspects of the subject, while at the same time eliminating misconceptions and misunderstandings. So it's not surprising that the the fundamental philosophy of this book is that 'theory is best understood by putting it into practice'. Well, that's fine as it stands. Unfortunately the practice may a bit more challenging, especially in the field of real-time operating systems. First, you need a sensible, practical toolset on which to carry out the work. Second, for many self-learners, cost is an issue; the tools mustn't be expensive. Third, they mustn't be difficult to get, use and maintain. So what we have here is our approach to providing you with a low cost toolset for RTOS experimentation.The toolset used for this work consists of: A graphical tool for configuring microcontrollers (specifically STM32F variants) - STM32CubeMX software application.An Integrated Development Environment for the production of machine code.A very low cost single board computer with inbuilt programmer and debuggerAll software, which is free, can be run on Windows, OSX or Linux platforms. The Discovery kit is readily available from many electronic suppliers. The RTOS used for this work is FreeRTOS, which is integrated with the CubeMX tool.The author: Jim Cooling has had many years experience in the area of real-time embedded systems, including electronic, software and system design, project management, consultancy, education and course development. He has published extensively on the subject, his books covering many aspects of embedded-systems work such as real-time interfacing, programming, software design and software engineering. Currently he is a partner in Lindentree Associates (which he formed in 1998), providing consultancy and training for real-time embedded systems.See: www.lindentreeuk.co.uk

An introduction to embedding systems for C and C++++ programmers encompasses such topics as testing memory devices, writing and erasing Flash memory, verifying nonvolatile memory contents, and much more. Original. (Intermediate).

Front Cover; Dedication; Embedded Systems Security: Practical Methods for Safe and Secure Softwareand Systems Development; Copyright; Contents; Foreword; Preface; About this Book; Audience; Organization; Approach; Acknowledgements; Chapter 1 -- Introduction to Embedded Systems Security; 1.1What is Security?; 1.2What is an Embedded System?; 1.3Embedded Security Trends; 1.4Security Policies; 1.5Security Threats; 1.6Wrap-up; 1.7Key Points; 1.8 Bibliography and Notes; Chapter 2 -- Systems Software Considerations; 2.1The Role of the Operating System; 2.2Multiple Independent Levels of Security.

From the Foreword: "...the presentation of real-time scheduling is probably the best in terms of clarity I have ever read in the professional literature. Easy to understand, which is important for busy professionals keen to acquire (or refresh) new knowledge without being bogged down in a convoluted narrative and an excessive detail overload. The authors managed to largely avoid theoretical-only presentation of the subject, which frequently affects books on operating systems. ... an indispensable [resource] to gain a thorough understanding of the real-time systems from the operating systems perspective, and to stay up to date with the recent trends and actual developments of the open-source real-time operating systems." —Richard Zurawski, ISA Group, San Francisco, California, USA Real-time embedded systems are integral to the global technological and social space, but references still rarely offer professionals the sufficient mix of theory and practical examples required to meet intensive economic, safety, and other demands on system development. Similarly, instructors have lacked a resource to help students fully understand the field. The information was out there, though often at the abstract level, fragmented and scattered throughout literature from different engineering disciplines and computing sciences. Accounting for readers’ varying practical needs and

experience levels, Real Time Embedded Systems: Open-Source Operating Systems Perspective offers a holistic overview from the operating-systems perspective. It provides a long-awaited reference on real-time operating systems and their almost boundless application potential in the embedded system domain. Balancing the already abundant coverage of operating systems with the largely ignored real-time aspects, or "physicality," the authors analyze several realistic case studies to introduce vital theoretical material. They also discuss popular open-source operating systems—Linux and FreRTOS, in particular—to help embedded-system designers identify the benefits and weaknesses in deciding whether or not to adopt more traditional, less powerful, techniques for a project.

This book covers the basic concepts and principles of operating systems, showing how to apply them to the design and implementation of complete operating systems for embedded and real-time systems. It includes all the foundational and background information on ARM architecture, ARM instructions and programming, toolchain for developing programs, virtual machines for software implementation and testing, program execution image, function call conventions, run-time stack usage and link C programs with assembly code. It describes the design and implementation of a complete OS for embedded systems in incremental steps, explaining the design principles and implementation techniques. For Symmetric Multiprocessing (SMP) embedded systems, the author examines the ARM MPcore processors, which include the SCU and GIC for interrupts routing and interprocessor communication and synchronization by Software Generated Interrupts (SGIs).Throughout the book, complete working sample systems demonstrate the design principles and implementation techniques. The content is suitable for advanced-level and graduate students working in software engineering, programming, and systems theory.

Authored by two of the leading authorities in the field, this guide offers readers the knowledge and skills needed to achieve proficiency with embedded software.

Software Engineering for Embedded Systems: Methods, Practical Techniques, and Applications, Second Edition provides the techniques and technologies in software engineering to optimally design and implement an embedded system. Written by experts with a solution focus, this encyclopedic reference gives an indispensable aid on how to tackle the day-to-day problems encountered when using software engineering methods to develop embedded systems. New sections cover peripheral programming, Internet of things, security and cryptography, networking and packet processing, and hands on labs. Users will learn about the principles of good architecture for an embedded system, design practices, details on principles, and much more. Provides a roadmap of key problems/issues and references to their solution in the text Reviews core methods and how to apply them Contains examples that demonstrate timeless implementation details Users case studies to show how key ideas can be implemented, the rationale for choices made, and design guidelines and trade-offs

Intelligent readers who want to build their own embedded computer systems-- installed in everything from cell phones to cars to handheld organizers to refrigerators-- will find this book to be the most in-depth, practical, and up-to-date guide on the market. Designing Embedded Hardware carefully steers between the practical and philosophical aspects, so developers can both create their own devices and gadgets and customize and extend off-the-shelf systems. There are hundreds of books to choose from if you need to learn programming, but only a few are available if you want to learn to create hardware. Designing Embedded Hardware provides software and hardware engineers with no prior experience in embedded systems with the necessary conceptual and design building blocks to understand the architectures of embedded systems. Written to provide the depth of coverage and real-world examples developers need, Designing Embedded Hardware also provides a road-map to the pitfalls and traps to avoid in designing embedded systems. Designing Embedded Hardware covers such essential topics as: The principles of developing computer hardware Core hardware designs Assembly

language concepts Parallel I/O Analog-digital conversion Timers (internal and external) UART Serial Peripheral Interface Inter-Integrated Circuit Bus Controller Area Network (CAN) Data Converter Interface (DCI) Low-power operation This invaluable and eminently useful book gives you the practical tools and skills to develop, build, and program your own application-specific computers.

Build a strong foundation in designing and implementing real-time systems with the help of practical examples Key Features Get up and running with the fundamentals of RTOS and apply them on STM32 Enhance your programming skills to design and build real-world embedded systems Get to grips with advanced techniques for implementing embedded systems Book Description A real-time operating system (RTOS) is used to develop systems that respond to events within strict timelines. Real-time embedded systems have applications in various industries, from automotive and aerospace through to laboratory test equipment and consumer electronics. These systems provide consistent and reliable timing and are designed to run without intervention for years. This microcontrollers book starts by introducing you to the concept of RTOS and compares some other alternative methods for achieving real-time performance. Once you've understood the fundamentals, such as tasks, queues, mutexes, and semaphores, you'll learn what to look for when selecting a microcontroller and development environment. By working through examples that use an STM32F7 Nucleo board, the STM32CubeIDE, and SEGGER debug tools, including SEGGER J-Link, Ozone, and SystemView, you'll gain an understanding of preemptive scheduling policies and task communication. The book will then help you develop highly efficient low-level drivers and analyze their real-time performance and CPU utilization. Finally, you'll cover tips for troubleshooting and be able to take your new-found skills to the next level. By the end of this book, you'll have built on your embedded system skills and will be able to create real-time systems using microcontrollers and FreeRTOS. What you will learn Understand when to use an RTOS for a project Explore RTOS concepts such as tasks, mutexes, semaphores, and queues Discover different microcontroller units (MCUs) and choose the best one for your project Evaluate and select the best IDE and middleware stack for your project Use professional-grade tools for analyzing and debugging your application Get FreeRTOS-based applications up and running on an STM32 board Who this book is for This book is for embedded engineers, students, or anyone interested in learning the complete RTOS feature set with embedded devices. A basic understanding of the C programming language and embedded systems or microcontrollers will be helpful.

The textbook is used at the Faculty of Electrical Engineering of the University of Ljubljana. It introduces the students of Electronics into the operating systems and real-time concepts having the embedded systems perspective in mind. In the opening chapters, the textbook presents the basic properties of operating systems and computer networks with the Internet Protocol. Linux is used as an example platform. It continues with embedded system peculiarities using the PYHTEC phyCORE-i.MX27 development kit as a platform. Programing of peripheral devices and graphical applications is described. The characteristics of real-time systems follow. The real-time application structure is given. The principles of the inter-process communication, addressing resource sharing problem with synchronization and deadlock situations are presented.

As the embedded world expands, developers must have a strong grasp of many complex topics in order to make faster, more efficient and more powerful microprocessors to meet the public’s growing demand. Embedded Software: The Works covers all the key subjects embedded engineers need to understand in order to succeed, including Design and Development, Programming, Languages including C/C++, and UML, Real Time Operating Systems Considerations, Networking, and much more. New material on Linux, Android, and multi-core gives engineers the up-to-date practical know-how they need in order to succeed.

Colin Walls draws upon his experience and insights from working in the industry, and covers the complete cycle of embedded software development: its design, development, management, debugging procedures, licensing, and reuse. For those new to the field, or for experienced engineers looking to expand their skills, Walls provides the reader with detailed tips and techniques, and rigorous explanations of technologies. Key features include: New chapters on Linux, Android, and multi-core - the cutting edge of embedded software development! Introductory roadmap guides readers through the book, providing a route through the separate chapters and showing how they are linked About the Author Colin Walls has over twenty-five years experience in the electronics industry, largely dedicated to embedded software. A frequent presenter at conferences and seminars and author of numerous technical articles and two books on embedded software, he is a member of the marketing team of the Mentor Graphics Embedded Software Division. He writes a regular blog on the Mentor website (blogs.mentor.com/colinwalls). New chapters on Linux, Android, and multi-core - the cutting edge of embedded software development! Introductory roadmap guides readers through the book, providing a route through the separate chapters and showing how they are linked

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.

This easy-to- follow textbook/reference guides the reader through the creation of a fully functional embedded operating system, from its source code, in order to develop a deeper understanding of each component and how they work together. The text describes in detail the procedure for building the bootloader, kernel, filesystem, shared libraries, start-up scripts, configuration files and system utilities, to produce a GNU/Linux operating system. This fully updated second edition also includes new material on virtual machine technologies such as VirtualBox, Vagrant and the Linux container system Docker. Topics and features: presents an overview of the GNU/Linux system, introducing the components of the system, and covering aspects of process management, input/output and environment; discusses containers and the underlying kernel technology upon which they are based; provides a detailed examination of the GNU/Linux filesystem; explains how to build an embedded system under a virtual machine,

and how to build an embedded system to run natively on an actual processor;introduces the concept of the compiler toolchain, and reviews the platforms BeagleBone and Raspberry Pi; describes how to build firmware images for devices running the Openwrt operating system. The hands-on nature and clearly structured approach of this textbook will appeal strongly to practically minded undergraduate and graduate level students, as well as to industry professionals involved in this area.

This Expert Guide gives you the techniques and technologies in software engineering to optimally design and implement your embedded system. Written by experts with a solutions focus, this encyclopedic reference gives you an indispensable aid to tackling the day-to-day problems when using software engineering methods to develop your embedded systems. With this book you will learn: The principles of good architecture for an embedded system Design practices to help make your embedded project successful Details on principles that are often a part of embedded systems, including digital signal processing, safety-critical principles, and development processes Techniques for setting up a performance engineering strategy for your embedded system software How to develop user interfaces for embedded systems Strategies for testing and deploying your embedded system, and ensuring quality development processes Practical techniques for optimizing embedded software for performance, memory, and power Advanced guidelines for developing multicore software for embedded systems How to develop embedded software for networking, storage, and automotive segments How to manage the embedded development process Includes contributions from: Frank Schirrmeister, Shelly Gretlein, Bruce Douglass, Erich Styger, Gary Stringham, Jean Labrosse, Jim Trudeau, Mike Brogioli, Mark Pitchford, Catalin Dan Udma, Markus Levy, Pete Wilson, Whit Waldo, Inga Harris, Xinxin Yang, Srinivasa Addepalli, Andrew McKay, Mark Kraeling and Robert Oshana. Road map of key problems/issues and references to their solution in the text Review of core methods in the context of how to apply them Examples demonstrating timeless implementation details Short and to- the- point case studies show how key ideas can be implemented, the rationale for choices made, and design guidelines and trade-offs

This book outlines the major elements that go into embedded operating systems and board support packages (BSPs), and gives readers a strong foundation on which to apply the technology they must master in the future. It describes types of embedded operating systems and demonstrates how they can be implemented within a particular embedded device. Current exciting technology trends within the embedded systems arena are discussed, including the latest examples of real world technologies: references to real-world hardware and software protocols, as well as references to technical specifications, are used. With this book you will learn: what an embedded operating system is and what a board support package is to understand the main components that make up embedded operating systems, and their internal design how to determine the most suitable embedded operating system for a particular device how to select the best embedded operating system for a particular product, between different real world embedded operating systems available on the market what the main components that make up a board support package (BSP) are, and the internal design of BSPs how to port an embedded operating system from one platform to another, and the importance of a board support package in solving the challenges of porting embedded middleware and application software from one embedded platform design, development and debugging of an embedded operating system and an underlying board support package how to predict the security, robustness and predictability of an embedded system that utilizes a particular embedded operating system design Outlines the entire process of designing embedded operating systems and their underlying board support packages References to real-world hardware and software demonstrate and explain practical ‘know how’, relative to the underlying concepts throughout the entire book Contains numerous real world examples (based on industry software), source code, problems and solutions, references, projects,

glossary, and web links with software

The proliferation of multicore processors in the embedded market for Internet-of-Things (IoT) and Cyber-Physical Systems (CPS) makes developing real-time embedded applications increasingly difficult. What is the underlying theory that makes multicore real-time possible? How does theory influence application design? When is a real-time operating system (RTOS) useful? What RTOS features do applications need? How does a mature RTOS help manage the complexity of multicore hardware? Real-Time Systems Development with RTEMS and Multicore Processors answers these questions and more with exemplar Real-Time Executive for Multiprocessor Systems (RTEMS) RTOS to provide concrete advice and examples for constructing useful, feature-rich applications. RTEMS is free, open-source software that supports multi-processor systems for over a dozen CPU architectures and over 150 specific system boards in applications spanning the range of IoT and CPS domains such as satellites, particle accelerators, robots, racing motorcycles, building controls, medical devices, and more. The focus of this book is on enabling real-time embedded software engineering while providing sufficient theoretical foundations and hardware background to understand the rationale for key decisions in RTOS and application design and implementation. The topics covered in this book include: Cross-compilation for embedded systems development Concurrent programming models used in real-time embedded software Real-time scheduling theory and algorithms used in wide practice Usage and comparison of two application programmer interfaces (APIs) in real-time embedded software: POSIX and the RTEMS Classic APIs Design and implementation in RTEMS of commonly found RTOS features for schedulers, task management, time-keeping, inter-task synchronization, inter-task communication, and networking The challenges introduced by multicore hardware, advances in multicore real-time theory, and software engineering multicore real-time systems with RTEMS All the authors of this book are experts in the academic field of real-time embedded systems. Two of the authors are primary open-source maintainers of the RTEMS software project.

IMPORTANT: This is a rebadged version of Real-time Operating Systems, Book 1, The Theory which (so far) has received eleven 5-star, one 4-star and one 3-star reviews.This book deals with the fundamentals of operating systems for use in real-time embedded systems. It is aimed at those who wish to develop RTOS-based designs, using either commercial or free products. It does not set out to give you a knowledge to design an RTOS; leave that to the specialists. The target readership includes:- Students.- Engineers, scientists and mathematicians moving into software systems.- Professional and experienced software engineers entering the embedded field.- Programmers having little or no formal education in the underlying principles of software-based real-time systems.The material covers the key 'nuts and bolts' of RTOS structures and usage (as you would expect, of course). In many cases it shows how these are handled by practical real-time operating systems. It also places great emphasises on ways to structure the application software so that it can be effectively implemented using an RTOS. After studying this even the absolute beginner will see that it isn't particularly difficult to implement RTOS-based designs and should be confident to take on such work.

By using this innovative text, students will obtain an understanding of how contemporary operating systems and middleware work, and why they work that way. Now in its 2nd edition, this textbook has been updated on a new development board from STMicroelectronics - the Arm Cortex-M0+ based Nucleo-F091RC. Designed to be used in a one- or two-semester introductory course on embedded systems.

This book deals with the fundamentals of operating systems for use in real-time

embedded systems. It is aimed at those who wish to develop RTOS-based designs, using either commercial or free products. It does not set out to give you a knowledge to design an RTOS; leave that to the specialists. The target readership includes:- Students.- Engineers, scientists and mathematicians moving into software systems.- Professional and experienced software engineers entering the embedded field.- Programmers having little or no formal education in the underlying principles of software-based real-time systems.The material covers the key 'nuts and bolts' of RTOS structures and usage (as you would expect, of course). In many cases it shows how these are handled by practical real-time operating systems. It also places great emphasises on ways to structure the application software so that it can be effectively implemented using an RTOS. After studying this even the absolute beginner will see that it isn't particularly difficult to implement RTOS-based designs and should be confident to take on such work.

An introduction to the engineering principles of embedded systems, with a focus on modeling, design, and analysis of cyber-physical systems. The most visible use of computers and software is processing information for human consumption. The vast majority of computers in use, however, are much less visible. They run the engine, brakes, seatbelts, airbag, and audio system in your car. They digitally encode your voice and construct a radio signal to send it from your cell phone to a base station. They command robots on a factory floor, power generation in a power plant, processes in a chemical plant, and traffic lights in a city. These less visible computers are called embedded systems, and the software they run is called embedded software. The principal challenges in designing and analyzing embedded systems stem from their interaction with physical processes. This book takes a cyber-physical approach to embedded systems, introducing the engineering concepts underlying embedded systems as a technology and as a subject of study. The focus is on modeling, design, and analysis of cyber-physical systems, which integrate computation, networking, and physical processes. The second edition offers two new chapters, several new exercises, and other improvements. The book can be used as a textbook at the advanced undergraduate or introductory graduate level and as a professional reference for practicing engineers and computer scientists. Readers should have some familiarity with machine structures, computer programming, basic discrete mathematics and algorithms, and signals and systems.

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.
Interested in developing embedded systems? Since they don't tolerate inefficiency, these systems require a disciplined approach to programming. This easy-to-read guide helps you cultivate a host of good development practices, based on classic software design patterns and new patterns unique to embedded programming. Learn how to build system architecture for processors, not operating systems, and discover specific techniques for dealing with hardware difficulties and manufacturing requirements. Written by an expert who's created embedded systems ranging from urban surveillance and DNA scanners to children's toys, this book is ideal for intermediate and experienced programmers, no matter what platform you use. Optimize your system to reduce cost and increase performance Develop an architecture that makes your software robust in resource-constrained environments Explore sensors, motors, and other I/O devices Do more with less: reduce RAM consumption, code space, processor cycles, and power consumption Learn how to update embedded code directly in the processor Discover how to implement complex mathematics on small processors Understand what interviewers look for when you apply for an embedded systems job "Making Embedded Systems is the book for a C programmer who wants to enter the fun (and lucrative) world of embedded systems. It's very well written—entertaining, even—and filled with clear illustrations." —Jack Ganssle, author and embedded system expert.
Offering comprehensive coverage of the convergence of real-time embedded systems scheduling, resource access control, software design and development, and high-level system modeling, analysis and verification Following an introductory overview, Dr. Wang delves into the specifics of hardware components, including processors, memory, I/O devices and architectures, communication structures, peripherals, and characteristics of real-time operating systems. Later chapters are dedicated to real-time task scheduling algorithms and resource access control policies, as well as priority-inversion control and deadlock avoidance. Concurrent system programming and POSIX programming for real-time systems are covered, as are finite state machines and Time Petri nets. Of special interest to software engineers will be the chapter devoted to model checking, in which the author discusses temporal logic and the NuSMV model checking tool, as well as a chapter treating real-time software design with UML. The final portion of the book explores practical issues of software reliability, aging, rejuvenation, security, safety, and power management. In addition, the book: Explains real-time embedded software modeling and design with finite state machines, Petri nets, and UML, and real-time constraints verification with the model checking tool, NuSMV Features real-world examples in finite state machines, model checking, real-time system design with UML, and more Covers embedded computer programing, designing for reliability, and designing for safety Explains how to make engineering trade-offs of power use and performance Investigates practical issues concerning software reliability, aging, rejuvenation, security, and power management Real-Time Embedded Systems is a valuable resource for those responsible for real-time and embedded software design, development, and management. It is also an excellent textbook for graduate courses in computer engineering, computer science, information technology, and software engineering on embedded and real-time software systems,

and for undergraduate computer and software engineering courses.

Embedded Systems: ARM Programming and Optimization combines an exploration of the ARM architecture with an examination of the facilities offered by the Linux operating system to explain how various features of program design can influence processor performance. It demonstrates methods by which a programmer can optimize program code in a way that does not impact its behavior but improves its performance. Several applications, including image transformations, fractal generation, image convolution, and computer vision tasks, are used to describe and demonstrate these methods. From this, the reader will gain insight into computer architecture and application design, as well as gain practical knowledge in the area of embedded software design for modern embedded systems. Covers three ARM instruction set architectures, the ARMv6 and ARMv7-A, as well as three ARM cores, the ARM11 on the Raspberry Pi, Cortex-A9 on the Xilinx Zynq 7020, and Cortex-A15 on the NVIDIA Tegra K1 Describes how to fully leverage the facilities offered by the Linux operating system, including the Linux GCC compiler toolchain and debug tools, performance monitoring support, OpenMP multicore runtime environment, video frame buffer, and video capture capabilities Designed to accompany and work with most of the low cost Linux/ARM embedded development boards currently available

MicroC/OS II Second Edition describes the design and implementation of the MicroC/OS-II real-time operating system (RTOS). In addition to its value as a reference to the kernel, it is an extremely detailed and highly readable design study particularly useful to the embedded systems student. While documenting the design and implementation of the ker

Embedded Software Development: The Open-Source Approach delivers a practical introduction to embedded software development, with a focus on open-source components. This programmer-centric book is written in a way that enables even novice practitioners to grasp the development process as a whole. Incorporating real code fragments and explicit, real-world open-source operating system references (in particular, FreeRTOS) throughout, the text: Defines the role and purpose of embedded systems, describing their internal structure and interfacing with software development tools Examines the inner workings of the GNU compiler collection (GCC)-based software development system or, in other words, toolchain Presents software execution models that can be adopted profitably to model and express concurrency Addresses the basic nomenclature, models, and concepts related to task-based scheduling algorithms Shows how an open-source protocol stack can be integrated in an embedded system and interfaced with other software components Analyzes the main components of the FreeRTOS Application Programming Interface (API), detailing the implementation of key operating system concepts Discusses advanced topics such as formal verification, model checking, runtime checks, memory corruption, security, and dependability Embedded Software Development: The Open-Source Approach

capitalizes on the authors' extensive research on real-time operating systems and communications used in embedded applications, often carried out in strict cooperation with industry. Thus, the book serves as a springboard for further research.

Embedded Microcomputer Systems: Real Time Interfacing provides an in-depth discussion of the design of real-time embedded systems using 9S12 microcontrollers. This book covers the hardware aspects of interfacing, advanced software topics (including interrupts), and a systems approach to typical embedded applications. This text stands out from other microcomputer systems books because of its balanced, in-depth treatment of both hardware and software issues important in real time embedded systems design. It features a wealth of detailed case studies that demonstrate basic concepts in the context of actual working examples of systems. It also features a unique simulation software package on the bound-in CD-ROM (called Test Execute and Simulate, or TExaS, for short) that provides a self-contained software environment for designing, writing, implementing, and testing both the hardware and software components of embedded systems. Important Notice: Media content referenced within the product description or the product text may not be available in the ebook version.

This book integrates new ideas and topics from real time systems, embedded systems, and software engineering to give a complete picture of the whole process of developing software for real-time embedded applications. You will not only gain a thorough understanding of concepts related to microprocessors, interrupts, and system boot process, appreciating the importance of real-time modeling and scheduling, but you will also learn software engineering practices such as model documentation, model analysis, design patterns, and standard conformance. This book is split into four parts to help you learn the key concept of embedded systems; Part one introduces the development process, and includes two chapters on microprocessors and interrupts---fundamental topics for software engineers; Part two is dedicated to modeling techniques for real-time systems; Part three looks at the design of software architectures and Part four covers software implementations, with a focus on POSIX-compliant operating systems. With this book you will learn: The pros and cons of different architectures for embedded systems POSIX real-time extensions, and how to develop POSIX-compliant real time applications How to use real-time UML to document system designs with timing constraints The challenges and concepts related to cross-development Multitasking design and inter-task communication techniques (shared memory objects, message queues, pipes, signals) How to use kernel objects (e.g. Semaphores, Mutex, Condition variables) to address resource sharing issues in RTOS applications The philosophy underpinning the notion of "resource manager" and how to implement a virtual file system using a resource manager The key principles of real-time scheduling and several key algorithms Coverage of the latest UML standard (UML 2.4) Over 20 design patterns which represent the best practices for reuse in a wide range of real-time

embedded systems Example codes which have been tested in QNX---a real-time operating system widely adopted in industry

The ultimate resource for making embedded systems reliable, safe, and secure Embedded Systems Security provides: A broad understanding of security principles, concerns, and technologies Proven techniques for the efficient development of safe and secure embedded software A study of the system architectures, operating systems and hypervisors, networking, storage, and cryptographic issues that must be considered when designing secure embedded systems Nuggets of practical advice and numerous case studies throughout Written by leading authorities in the field with 65 years of embedded security experience: one of the original developers of the world's only Common Criteria EAL 6+ security certified software product and a lead designer of NSA certified cryptographic systems. This book is indispensable for embedded systems and security professionals, new and experienced. An important contribution to the understanding of the security of embedded systems. The Kleidermachers are experts in their field. As the Internet of things becomes reality, this book helps business and technology management as well as engineers understand the importance of "security from scratch." This book, with its examples and key points, can help bring more secure, robust systems to the market. Dr. Joerg Borchert, Vice President, Chip Card & Security, Infineon Technologies North America Corp.; President and Chairman, Trusted Computing Group Embedded Systems Security provides real-world examples of risk and exploitation; most importantly the book offers clear insight into methods used to counter vulnerabilities to build true, native security into technology. Adriel Desautels, President and CTO, Netragard, LLC. Security of embedded systems is more important than ever. The growth in networking is just one reason. However, many embedded systems developers have insufficient knowledge of how to achieve security in their systems. David Kleidermacher, a world-renowned expert in this field, shares in this book his knowledge and long experience with other engineers. A very important book at the right time. Prof. Dr.-Ing. Matthias Sturm, Leipzig University of Applied Sciences; Chairman, Embedded World Conference steering board Gain an understanding of the operating systems, microprocessors, and network security critical issues that must be considered when designing secure embedded systems Contains nuggets of practical and simple advice on critical issues highlighted throughout the text Short and to –the-point real case studies included to demonstrate embedded systems security in practice

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.
Software -- Operating Systems.