# Linux Rapid Embedded Programming

## GNU/Linux Rapid Embedded Programming

An annotated guide to program and develop GNU/Linux Embedded systems quickly Key Features Rapidly design and build powerful prototypes for GNU/Linux Embedded systems Become familiar with the workings of GNU/Linux Embedded systems and how to manage its peripherals Write, monitor, and configure applications quickly and effectively, manage an external micro-controller, and use it as co-processor for real-time tasks Book DescriptionEmbedded computers have become very complex in the last few years and developers need to easily manage them by focusing on how to solve a problem without wasting time in finding supported peripherals or learning how to manage them. The main challenge with experienced embedded programmers and engineers is really how long it takes to turn an idea into reality, and we show you exactly how to do it. This book shows how to interact with external environments through specific peripherals used in the industry. We will use the latest Linux kernel release 4.4.x and Debian/Ubuntu distributions (with embedded distributions like OpenWrt and Yocto). The book will present popular boards in the industry that are user-friendly to base the rest of the projects on - BeagleBone Black, SAMA5D3 Xplained, Wandboard and system-on-chip manufacturers. Readers will be able to take their first steps in programming the embedded platforms, using C, Bash, and Python/PHP languages in order to get access to the external peripherals. More about using and programming device driver and accessing the peripherals will be covered to lay a strong foundation. The readers will learn how to read/write data from/to the external environment by using both C programs or a scripting language (Bash/PHP/Python) and how to configure a device driver for a specific hardware. After finishing this book, the readers will be able to gain a good knowledge level and understanding of writing, configuring, and managing drivers, controlling and monitoring applications with the help of efficient/quick programming and will be able to apply these skills into real-world projects. What you will learn Use embedded systems to implement your projects Access and manage peripherals for embedded systems Program embedded systems using languages such as C, Python, Bash, and PHP Use a complete distribution, such as Debian or Ubuntu, or an embedded one, such as OpenWrt or Yocto Harness device driver capabilities to optimize device communications Access data through several kinds of devices such as GPIO's, serial ports, PWM, ADC, Ethernet, WiFi, audio, video, I2C, SPI, One Wire, USB and CAN Who this book is for This book targets Embedded System developers and GNU/Linux programmers who would like to program Embedded Systems and perform Embedded development. The book focuses on quick and efficient prototype building. Some experience with hardware and Embedded Systems is assumed, as is having done some previous work on GNU/Linux systems. Knowledge of scripting on GNU/Linux is expected as well.

## Mastering Embedded Linux Programming

Build, customize, and deploy Linux-based embedded systems with confidence using Yocto, bootloaders, and build tools Key Features Master build systems, toolchains, and kernel integration for embedded Linux Set up custom Linux distros with Yocto and manage board-specific configurations Learn real-world debugging, memory handling, and system performance tuning Book DescriptionIf you’re looking for a book that will demystify embedded Linux, then you’ve come to the right place. Mastering Embedded Linux Programming is a fully comprehensive guide that can serve both as means to learn new things or as a handy reference. The first few chapters of this book will break 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 will show you 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 how to implement Linux on embedded devices, then this book is for you. It's also aimed at embedded systems engineers accustomed to programming for low-power microcontrollers, who can use this book to help make the leap to high-speed systems on chips that can run Linux. Anyone who develops hardware that needs to run Linux will find something useful in this book – but before you get started, you'll need a solid grasp on POSIX standard, C programming, and shell scripting.

## Building Embedded Linux Systems

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.

## Embedded Linux System Design and Development

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.

## Linux Device Driver Development Cookbook

Over 30 recipes to develop custom drivers for your embedded Linux applications Key Features Use kernel facilities to develop powerful drivers Learn core concepts for developing device drivers using a practical approach Program a custom character device to get access to kernel internals Book DescriptionLinux is a unified kernel that is widely used to develop embedded systems. As Linux has turned out to be one of the most popular operating systems worldwide, the interest in developing proprietary device drivers has also increased. Device drivers play a critical role in how the system performs and ensure that the device works in the manner intended. By exploring several examples on the development of character devices, the technique of managing a device tree, and how to use other kernel internals, such as interrupts, kernel timers, and wait queue, you'll be able to add proper management for custom peripherals to your embedded system. You'll begin by installing the Linux kernel and then configuring it. Once you have installed the system, you will learn to use different kernel features and character drivers. You will also cover interrupts in-depth and understand how you can manage them. Later, you will explore the kernel internals required for developing applications. As you approach the concluding chapters, you will learn to implement advanced character drivers and also discover how to write important Linux device drivers. By the end of this book, you will be equipped with the skills you need to write a custom character driver and kernel code according to your requirements.What you will learn Become familiar with the latest kernel releases (4.19/5.x) running on the ESPRESSOBin devkit, an ARM 64-bit machine Download, configure, modify, and build kernel sources Add and remove a device driver or a module from the kernel Understand how to implement character drivers to manage different kinds of computer peripherals Get well-versed with kernel helper functions and objects that can be used to build kernel applications Gain comprehensive insights into managing custom hardware with Linux from both the kernel and user space Who this book is for This book is for anyone who wants to develop their own Linux device drivers for embedded systems. Basic hands-on experience with the Linux operating system and embedded concepts is necessary.

## Real-Time Embedded Components and Systems with Linux and RTOS

This book is intended to provide a senior undergraduate or graduate student in electrical engineering or computer science with a balance of fundamental theory, review of industry practice, and hands-on experience to prepare for a career in the real-time embedded system industries. It is also intended to provide the practicing engineer with the necessary background to apply real-time theory to the design of embedded components and systems. Typical industries include aerospace, medical diagnostic and therapeutic systems, telecommunications, automotive, robotics, industrial process control, media systems, computer gaming, and electronic entertainment, as well as multimedia applications for general-purpose computing. This updated edition adds three new chapters focused on key technology advancements in embedded systems and with wider coverage of real-time architectures. The overall focus remains the RTOS (Real-Time Operating System), but use of Linux for soft real-time, hybrid FPGA (Field Programmable Gate Array) architectures and advancements in multi-core system-on-chip (SoC), as well as software strategies for asymmetric and symmetric multiprocessing (AMP and SMP) relevant to real-time embedded systems, have been added. Companion files are provided with numerous project videos, resources, applications, and figures from the book. Instructors' resources are available upon adoption. FEATURES: • Provides a comprehensive, up to date, and accessible presentation of embedded systems without sacrificing theoretical foundations • Features the RTOS (Real-Time Operating System), but use of Linux for soft real-time, hybrid FPGA architectures and advancements in multi-core system-on-chip is included • Discusses an overview of RTOS advancements, including AMP and SMP configurations, with a discussion of future directions for RTOS use in multi-core architectures, such as SoC • Detailed applications coverage including robotics, computer vision, and continuous media • Includes a companion disc (4GB) with numerous videos, resources, projects, examples,

and figures from the book • Provides several instructors' resources, including lecture notes, Microsoft PP slides, etc.

## Embedded Linux Primer

How can we build bridges from the digital world of the Internet to the analog world that surrounds us? By bringing accessibility to embedded components such as sensors and microcontrollers, JavaScript and Node.js might shape the world of physical computing as they did for web browsers. This practical guide shows hardware and software engineers, makers, and web developers how to talk in JavaScript with a variety of hardware platforms. Authors Patrick Mulder and Kelsey Breseman also delve into the basics of microcontrollers, single-board computers, and other hardware components. Use JavaScript to program microcontrollers with Arduino and Espruino Prototype IoT devices with the Tessel 2 development platform Learn about electronic input and output components, including sensors Connect microcontrollers to the Internet with the Particle Photon toolchain Run Node.js on single-board computers such as Raspberry Pi and Intel Edison Talk to embedded devices with Node.js libraries such as Johnny-Five, and remotely control the devices with Bluetooth Use MQTT as a message broker to connect devices across networks Explore ways to use robots as building blocks for shared experiences

## Node.js for Embedded Systems

Interested in developing embedded systems? Since they donâ??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 whoâ??s created embedded systems ranging from urban surveillance and DNA scanners to childrenâ??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. Itâ??s very well writtenâ??entertaining, evenâ??and filled with clear illustrations.\" â??Jack Ganssle, author and embedded system expert.

## Making Embedded Systems

1. What Makes an Embedded Application Tick? -- 2. Memory in Embedded Systems -- 3. Memory Architectures -- 4. How Software Influences Hardware Design -- 5. Migrating your Software to a New Processor Architecture -- 6. Embedded Software for Transportation Applications -- 7. How to Choose a CPU for Your SoC Design -- 8. An Introduction to USB Software -- 9. Towards USB 3.0.

## Embedded Software

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.

## Programming Embedded Systems

Linux offers many advantages as an operating system for embedded designs - it's small, portable, scalable,

vendor-independent, and based on the open source model. Most Linux books concentrate on desktop and server applications but this text restores the focus to embedded systems.

## Linux for Embedded and Real-time Applications

Create the perfectly customized system by unleashing the power of Android OS on your embedded device About This Book Understand the system architecture and how the source code is organized Explore the power of Android and customize the build system Build a fully customized Android version as per your requirements Who This Book Is For If you are a Java programmer who wants to customize, build, and deploy your own Android version using embedded programming, then this book is for you. What You Will Learn Master Android architecture and system design Obtain source code and understand the modular organization Customize and build your first system image for the Android emulator Level up and build your own Android system for a real-world device Use Android as a home automation and entertainment system Tailor your system with optimizations and add-ons Reach for the stars: look at the Internet of Things, entertainment, and domotics In Detail Take a deep dive into the Android build system and its customization with Learning Embedded Android Programming, written to help you master the steep learning curve of working with embedded Android. Start by exploring the basics of Android OS, discover Google's “repo” system, and discover how to retrieve AOSP source code. You'll then find out to set up the build environment and the first AOSP system. Next, learn how to customize the boot sequence with a new animation, and use an Android “kitchen” to “cook” your custom ROM. By the end of the book, you'll be able to build customized Android open source projects by developing your own set of features. Style and approach This step-by-step guide is packed with various real-world examples to help you create a fully customized Android system with the most useful features available.

## Learning Embedded Android N Programming

This is the eBook version of the printed book. If the print book includes a CD-ROM, this content is not included within the eBook version. Advanced Linux Programming is divided into two parts. The first covers generic UNIX system services, but with a particular eye towards Linux specific information. This portion of the book will be of use even to advanced programmers who have worked with other Linux systems since it will cover Linux specific details and differences. For programmers without UNIX experience, it will be even more valuable. The second section covers material that is entirely Linux specific. These are truly advanced topics, and are the techniques that the gurus use to build great applications. While this book will focus mostly on the Application Programming Interface (API) provided by the Linux kernel and the C library, a preliminary introduction to the development tools available will allow all who purchase the book to make immediate use of Linux.

## Advanced Linux Programming

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.

## Introduction to Embedded Systems, Second Edition

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.

## Linux: Embedded Development

Two leading Linux developers show how to choose the best tools for your specific needs and integrate them into a complete development environment that maximizes your effectiveness in any project, no matter how large or complex. Includes research, requirements, coding, debugging, deployment, maintenance and beyond, choosing and implementing editors, compilers, assemblers, debuggers, version control systems, utilities, using Linux Standard Base to deliver applications that run reliably on a wide range of Linux systems, comparing Java development options for Linux platforms, using Linux in cross-platform and embedded development environments.

## Embedded Linux Systems with the Yocto Project

Build safety-critical and memory-safe stand-alone and networked embedded systems Key FeaturesKnow how C++ works and compares to other languages used for embedded developmentCreate advanced GUIs for embedded devices to design an attractive and functional UIIntegrate proven strategies into your design for optimum hardware performanceBook Description C++ is a great choice for embedded development, most notably, because it does not add any bloat, extends maintainability, and offers many advantages over different programming languages. Hands-On Embedded Programming with C++17 will show you how C++ can be used to build robust and concurrent systems that leverage the available hardware resources. Starting with a primer on embedded programming and the latest features of C++17, the book takes you through various facets of good programming. You'll learn how to use the concurrency, memory management, and functional programming features of C++ to build embedded systems. You will understand how to integrate your systems with external peripherals and efficient ways of working with drivers. This book will also guide you in testing and optimizing code for better performance and implementing useful design patterns. As an additional benefit, you will see how to work with Qt, the popular GUI library used for building embedded systems. By the end of the book, you will have gained the confidence to use C++ for embedded programming. What you will learnChoose the correct type of embedded platform to use for a projectDevelop drivers for OS-based embedded systemsUse concurrency and memory management with various microcontroller units (MCUs)Debug and test cross-platform code with LinuxImplement an infotainment system using a Linux-based single board computerExtend an existing embedded system with a Qt-based GUICommunicate with the FPGA side of a hybrid FPGA/SoC systemWho this book is for If you want to start developing effective embedded programs in C++, then this book is for you. Good knowledge of C++ language constructs is required to understand the topics covered in the book. No knowledge of embedded systems is assumed.

## The Linux Development Platform

For the first time in a single reference, this book provides the beginner with a coherent and logical introduction to the hardware and software of the PIC32, bringing together key material from the PIC32 Reference Manual, Data Sheets, XC32 C Compiler User's Guide, Assembler and Linker Guide, MIPS32 CPU manuals, and Harmony documentation. This book also trains you to use the Microchip documentation, allowing better life-long learning of the PIC32. The philosophy is to get you started quickly, but to emphasize fundamentals and to eliminate \"magic steps\" that prevent a deep understanding of how the software you write connects to the hardware. Applications focus on mechatronics: microcontroller-controlled electromechanical systems incorporating sensors and actuators. To support a learn-by-doing approach, you can follow the examples throughout the book using the sample code and your PIC32 development board. The exercises at the end of each chapter help you put your new skills to practice. Coverage includes: A practical introduction to the C programming language Getting up and running quickly with the PIC32 An exploration of the hardware architecture of the PIC32 and differences among PIC32 families Fundamentals of embedded computing with the PIC32, including the build process, time- and memory-efficient programming, and interrupts A peripheral reference, with extensive sample code covering digital input and output, counter/timers, PWM, analog input, input capture, watchdog timer, and communication by the parallel master port, SPI, I2C, CAN, USB, and UART An introduction to the Microchip Harmony programming framework Essential topics in mechatronics, including interfacing sensors to the PIC32, digital signal processing, theory of operation and control of brushed DC motors, motor sizing and gearing, and other actuators such as stepper motors, RC servos, and brushless DC motors For more information on the book, and to download free sample code, please visit http://www.nu32.org Extensive, freely downloadable sample code for the NU32 development board incorporating the PIC32MX795F512H microcontroller Free online instructional videos to support many of the chapters

## Hands-On Embedded Programming with C++17

Freely available source code, with contributions from thousands of programmers around the world: this is the spirit of the software revolution known as Open Source. Open Source has grabbed the computer industry's attention. Netscape has opened the source code to Mozilla; IBM supports Apache; major database vendors haved ported their products to Linux. As enterprises realize the power of the open-source development model, Open Source is becoming a viable mainstream alternative to commercial software.Now in Open Sources, leaders of Open Source come together for the first time to discuss the new vision of the software industry they have created. The essays in this volume offer insight into how the Open Source movement works, why it succeeds, and where it is going.For programmers who have labored on open-source projects, Open Sources is the new gospel: a powerful vision from the movement's spiritual leaders. For businesses integrating open-source software into their enterprise, Open Sources reveals the mysteries of how open development builds better software, and how businesses can leverage freely available software for a competitive business advantage.The contributors here have been the leaders in the open-source arena: Brian Behlendorf (Apache) Kirk McKusick (Berkeley Unix) Tim O'Reilly (Publisher, O'Reilly & Associates) Bruce Perens (Debian Project, Open Source Initiative) Tom Paquin and Jim Hamerly (mozilla.org, Netscape) Eric Raymond (Open Source Initiative) Richard Stallman (GNU, Free Software Foundation, Emacs) Michael Tiemann (Cygnus Solutions) Linus Torvalds (Linux) Paul Vixie (Bind) Larry Wall (Perl) This book explains why the majority of the Internet's servers use open- source technologies for everything from the operating system to Web serving and email. Key technology products developed with open-source software have overtaken and surpassed the commercial efforts of billion dollar companies like Microsoft and IBM to dominate software markets. Learn the inside story of what led Netscape to decide to release its source code using the open-source mode. Learn how Cygnus Solutions builds the world's best compilers by sharing the source code. Learn why venture capitalists are eagerly watching Red Hat Software, a company that gives its key product -- Linux -- away.For the first time in print, this book presents the story of the open- source phenomenon told by the people who created this movement.Open Sources will bring you into the world of free software and show you the revolution.

## Embedded Computing and Mechatronics with the PIC32 Microcontroller

Explore the complete process of developing systems based on field-programmable gate arrays (FPGAs), including the design of electronic circuits and the construction and debugging of prototype embedded devices Key Features Learn the basics of embedded systems and real-time operating systems Understand how FPGAs implement processing algorithms in hardware Design, construct, and debug custom digital systems from scratch using KiCad Book DescriptionModern digital devices used in homes, cars, and wearables contain highly sophisticated computing capabilities composed of embedded systems that generate, receive, and process digital data streams at rates up to multiple gigabits per second. This book will show you how to use Field Programmable Gate Arrays (FPGAs) and high-speed digital circuit design to create your own cutting-edge digital systems. Architecting High-Performance Embedded Systems takes you through the fundamental concepts of embedded systems, including real-time operation and the Internet of Things (IoT), and the architecture and capabilities of the latest generation of FPGAs. Using powerful free tools for FPGA design and electronic circuit design, you'll learn how to design, build, test, and debug high-performance FPGA-based IoT devices. The book will also help you get up to speed with embedded system design, circuit design, hardware construction, firmware development, and debugging to produce a high-performance embedded device – a network-based digital oscilloscope. You'll explore techniques such as designing four-layer printed circuit boards with high-speed differential signal pairs and assembling the board using surface-mount components. By the end of the book, you'll have a solid understanding of the concepts underlying embedded systems and FPGAs and will be able to design and construct your own sophisticated digital devices.What you will learn Understand the fundamentals of real-time embedded systems and sensors Discover the capabilities of FPGAs and how to use FPGA development tools Learn the principles of digital circuit design and PCB layout with KiCad Construct high-speed circuit board prototypes at low cost Design and develop high-performance algorithms for FPGAs Develop robust, reliable, and efficient firmware in C Thoroughly test and debug embedded device hardware and firmware Who this book is for This book is for software developers, IoT engineers, and anyone who wants to understand the process of developing high-performance embedded

systems. You'll also find this book useful if you want to learn about the fundamentals of FPGA development and all aspects of firmware development in C and C++. Familiarity with the C language, digital circuits, and electronic soldering is necessary to get started.

## Open Sources

Develop Linux device drivers from scratch, with hands-on guidance focused on embedded systems, covering key subsystems like I2C, SPI, GPIO, IRQ, and DMA for real-world hardware integration using kernel 4.13 Key Features Develop custom drivers for I2C, SPI, GPIO, RTC, and input devices using modern Linux kernel APIs Learn memory management, IRQ handling, DMA, and the device tree through hands on examples Explore embedded driver development with platform drivers, regmap, and IIO frameworks Book DescriptionLinux 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).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 Who this book is for This book is ideal for embedded systems developers, engineers, and Linux enthusiasts who want to learn how to write device drivers from scratch. Whether you're new to kernel development or looking to deepen your understanding of subsystems like I2C, SPI, and IRQs, this book provides practical, real-world instructions tailored for working with embedded Linux platforms. Foundational knowledge of C and basic Linux concepts is recommended.

## Architecting High-Performance Embedded Systems

The book starts with the basics, explaining how to compile and run your first program. First, each concept is explained to give you a solid understanding of the material. Practical examples are then presented, so you see how to apply the knowledge in real applications.

## Linux Device Drivers Development

There are many books on project management and many on embedded systems, but few address the project management of embedded products from concept to production. Project Management of Complex and Embedded Systems: Ensuring Product Integrity and Program Quality uses proven Project Management methods and elements of IEEE embedded software develop

## Beginning Linux?Programming

As the standard for KDE desktop environment, Trolltech's Qt is a necessary basis for all programmers who want to develop cross-platform applications on Windows, Mac OS, Linux, and FreeBSD. A multitude of popular applications have been written in Qt, including Adobe Photoshop Elements, Google Earth, Perforce Visual Client, and Skype. Foundations of Qt Development is based on Qt 4.2, and is aimed at C++ programmers who want to become proficient using this excellent toolkit to create graphical applications that

can be ported to all major platforms. The book is focused on teaching you to write your own code in addition to using existing code. Common areas of confusion are identified, addressed, and answered.

## Project Management of Complex and Embedded Systems

The MSP430 microcontroller family offers ultra-low power mixed signal, 16-bit architecture that is perfect for wireless low-power industrial and portable medical applications. This book begins with an overview of embedded systems and microcontrollers followed by a comprehensive in-depth look at the MSP430. The coverage included a tour of the microcontroller's architecture and functionality along with a review of the development environment. Start using the MSP430 armed with a complete understanding of the microcontroller and what you need to get the microcontroller up and running! - Details C and assembly language for the MSP430 - Companion Web site contains a development kit - Full coverage is given to the MSP430 instruction set, and sigma-delta analog-digital converters and timers

## Foundations of Qt Development

A guide to using Linux on embedded platforms for interfacing to the real world. \"Embedded Linux\" is one of the first books available that teaches readers development and implementation of interfacing applications on an Embedded Linux platform.

## MSP430 Microcontroller Basics

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.

## Embedded Linux

Learn how to write high-quality kernel module code, solve common Linux kernel programming issues, and understand the fundamentals of Linux kernel internals Key Features Discover how to write kernel code using the Loadable Kernel Module framework Explore industry-grade techniques to perform efficient memory allocation and data synchronization within the kernel Understand the essentials of key internals topics such as kernel architecture, memory management, CPU scheduling, and kernel synchronization Book

DescriptionLinux Kernel Programming is a comprehensive introduction for those new to Linux kernel and module development. This easy-to-follow guide will have you up and running with writing kernel code in next-to-no time. This book uses the latest 5.4 Long-Term Support (LTS) Linux kernel, which will be maintained from November 2019 through to December 2025. By working with the 5.4 LTS kernel throughout the book, you can be confident that your knowledge will continue to be valid for years to come. You'll start the journey by learning how to build the kernel from the source. Next, you'll write your first kernel module using the powerful Loadable Kernel Module (LKM) framework. The following chapters will cover key kernel internals topics including Linux kernel architecture, memory management, and CPU scheduling. During the course of this book, you'll delve into the fairly complex topic of concurrency within the kernel, understand the issues it can cause, and learn how they can be addressed with various locking technologies (mutexes, spinlocks, atomic, and refcount operators). You'll also benefit from more advanced material on cache effects, a primer on lock-free techniques within the kernel, deadlock avoidance (with lockdep), and kernel lock debugging techniques. By the end of this kernel book, you'll have a detailed understanding of the fundamentals of writing Linux kernel module code for real-world projects and products.What you will learn Write high-quality modular kernel code (LKM framework) for 5.x kernels Configure and build a kernel from source Explore the Linux kernel architecture Get to grips with key internals regarding memory management within the kernel Understand and work with various dynamic kernel memory alloc/dealloc APIs Discover key internals aspects regarding CPU scheduling within the kernel Gain an understanding of kernel concurrency issues Find out how to work with key kernel synchronization primitives Who this book is for This book is for Linux programmers beginning to find their way with Linux kernel development. If you're a Linux kernel and driver developer looking to overcome frequent and common kernel development issues, or understand kernel intervals, you'll find plenty of useful information. You'll need a solid foundation of Linux CLI and C programming before you can jump in.

## Embedded System Design

Famed author Jack Ganssle has selected the very best embedded systems design material from the Newnes portfolio. The result is a book covering the gamut of embedded design, from hardware to software to integrated embedded systems, with a strong pragmatic emphasis.

## Linux Kernel Programming

Linux Application Development, Second Edition, is the definitive reference for Linux programmers at all levels of experience, including C programmers moving from other operating systems. Building on their widely praised first edition, leading Linux programmers Michael Johnson and Erik Troan systematically present the key APIs and techniques you need to create robust, secure, efficient software or to port existing code to Linux. Linux Application Development is divided into four parts. Part 1 introduces you to Linux(the operating system, licenses, and documentation. Part 2 covers the most important aspects of the development environment(the compilers, linker, loader, and debugging tools. Part 3-the heart of the book-describes the interface to the kernel and to the core system libraries, including discussion of the process model, file handling, directory operations, signal processing (including the Linux signal API), job control, the POSIX(termios interface, sockets, and the Linux console. Part 4 describes important development libraries with interfaces more independent of the kernel. The source code from the book is freely available at http://www.awl.com/cseng/books/lad.

## Embedded Systems: World Class Designs

Expand Raspberry Pi capabilities with fundamental engineering principles Exploring Raspberry Pi is the innovators guide to bringing Raspberry Pi to life. This book favors engineering principles over a 'recipe' approach to give you the skills you need to design and build your own projects. You'll understand the fundamental principles in a way that transfers to any type of electronics, electronic modules, or external peripherals, using a \"learning by doing\" approach that caters to both beginners and experts. The book begins

with basic Linux and programming skills, and helps you stock your inventory with common parts and supplies. Next, you'll learn how to make parts work together to achieve the goals of your project, no matter what type of components you use. The companion website provides a full repository that structures all of the code and scripts, along with links to video tutorials and supplementary content that takes you deeper into your project. The Raspberry Pi's most famous feature is its adaptability. It can be used for thousands of electronic applications, and using the Linux OS expands the functionality even more. This book helps you get the most from your Raspberry Pi, but it also gives you the fundamental engineering skills you need to incorporate any electronics into any project. Develop the Linux and programming skills you need to build basic applications Build your inventory of parts so you can always \"make it work\" Understand interfacing, controlling, and communicating with almost any component Explore advanced applications with video, audio, real-world interactions, and more Be free to adapt and create with Exploring Raspberry Pi.

## Linux Application Development

With a mixture of theory, examples, and well-integrated figures, Embedded Software for the IoT helps the reader understand the details in the technologies behind the devices used in the Internet of Things. It provides an overview of IoT, parameters of designing an embedded system, and good practice concerning code, version control and defect-tracking needed to build and maintain a connected embedded system. After presenting a discussion on the history of the internet and the word wide web the book introduces modern CPUs and operating systems. The author then delves into an in-depth view of core IoT domains including: Wired and wireless networking Digital filters Security in embedded and networked systems Statistical Process Control for Industry 4.0 This book will benefit software developers moving into the embedded realm as well as developers already working with embedded systems.

## Exploring Raspberry Pi

Nowadays, embedded systems - the computer systems that are embedded in various kinds of devices and play an important role of specific control functions, have permitted various aspects of industry. Therefore, we can hardly discuss our life and society from now onwards without referring to embedded systems. For wide-ranging embedded systems to continue their growth, a number of high-quality fundamental and applied researches are indispensable. This book contains 19 excellent chapters and addresses a wide spectrum of research topics on embedded systems, including basic researches, theoretical studies, and practical work. Embedded systems can be made only after fusing miscellaneous technologies together. Various technologies condensed in this book will be helpful to researchers and engineers around the world.

## Embedded Software for the IoT

Master the art of developing customized device drivers for your embedded Linux systemsKey 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 practicesBook DescriptionLinux 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 WatchdogWho this book is forThis 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.

## Embedded Systems

LINUX DRIVER DEVELOPMENT FOR EMBEDDED PROCESSORS - SECOND EDITION - The flexibility of Linux embedded, the availability of powerful, energy efficient processors designed for embedded computing and the low cost of new processors are encouraging many industrial companies to come up with new developments based on embedded processors. Current engineers have in their hands powerful tools for developing applications previously unimagined, but they need to understand the countless features that Linux offers today. This book will teach you how to develop device drivers for Device Tree Linux embedded systems. You will learn how to write different types of Linux drivers, as well as the appropriate APIs (Application Program Interfaces) and methods to interface with kernel and user spaces. This is a book is meant to be practical, but also provides an important theoretical base. More than twenty drivers are written and ported to three different processors. You can choose between NXP i.MX7D, Microchip SAMA5D2 and Broadcom BCM2837 processors to develop and test the drivers, whose implementation is described in detail in the practical lab sections of the book. Before you start reading, I encourage you to acquire any of these processor boards whenever you have access to some GPIOs, and at least one SPI and I2C controllers. The hardware configurations of the different evaluation boards used to develop the drivers are explained in detail throughout this book; one of the boards used to implement the drivers is the famous Raspberry PI 3 Model B board. You will learn how to develop drivers, from the simplest ones that do not interact with any external hardware, to drivers that manage different kind of devices: accelerometers, DACs, ADCs, RGB LEDs, Multi-Display LED controllers, I/O expanders, and Buttons. You will also develop DMA drivers, drivers that manage interrupts, and drivers that write/read on the internal registers of the processor to control external devices. To easy the development of some of these drivers, you will use different types of Frameworks: Miscellaneous framework, LED framework, UIO framework, Input framework and the IIO industrial one. This second edition has been updated to the v4.9 LTS kernel. Recently, all the drivers have been ported to the new Microchip SAMA5D27-SOM1 (SAMA5D27 System On Module) using kernel 4.14 LTS and included in the GitHub repository of this book; these drivers have been tested in the ATSAMA5D27-SOM1-EK1 evaluation platform; the ATSAMA5D27-SOM1-EK1 practice lab settings are not described throughout the text of this book, but in a practice labs user guide that can be downloaded from the book ?s GitHub.

## Mastering Linux Device Driver Development

Write software that makes the most effective use of the Linux system, including the kernel and core system libraries. The majority of both Unix and Linux code is still written at the system level, and this book helps you focus on everything above the kernel, where applications such as Apache, bash, cp, vim, Emacs, gcc, gdb, glibc, ls, mv, and X exist. Written primarily for engineers looking to program at the low level, this updated edition of Linux System Programming gives you an understanding of core internals that makes for better code, no matter where it appears in the stack. You'll take an in-depth look at Linux from both a theoretical and an applied perspective over a wide range of programming topics, including: An overview of Linux, the kernel, the C library, and the C compiler Reading from and writing to files, along with other basic file I/O operations, including how the Linux kernel implements and manages file I/O Buffer size management, including the Standard I/O library Advanced I/O interfaces, memory mappings, and

optimization techniques The family of system calls for basic process management Advanced process management, including real-time processes File and directories-creating, moving, copying, deleting, and managing them Memory management—interfaces for allocating memory, managing the memory you have, and optimizing your memory access Signals and their role on a Unix system, plus basic and advanced signal interfaces Time, sleeping, and clock management, starting with the basics and continuing through POSIX clocks and high resolution timers

## Linux Driver Development for Embedded Processors - Second Edition

Linux Kernel Module Programming Guide is for people who want to write kernel modules. It takes a hands-on approach starting with writing a small \"hello, world\" program, and quickly moves from there. Far from a boring text on programming, Linux Kernel Module Programming Guide has a lively style that entertains while it educates. An excellent guide for anyone wishing to get started on kernel module programming. *** Money raised from the sale of this book supports the development of free software and documentation.

## Linux System Programming

\"BeagleBone Systems and Applications\" \"BeagleBone Systems and Applications\" is a comprehensive and authoritative guide to the architecture, programming, and practical deployment of BeagleBone hardware for modern embedded systems. Starting with an in-depth exploration of the diverse BeagleBone family—including Black, Green, Blue, AI, and industrial variants—the book meticulously covers the AM335x SoC, board design, expansion capabilities, and robust power management. Readers are provided with a solid technical foundation in hardware interfacing, networking, and capes, ensuring a strong grasp of the platform’s core capabilities. Delving into the world of embedded software, the book guides engineers and developers through OS customization, real-time Linux, secure boot processes, and advanced kernel development. It presents best-in-class methods for device driver programming, hardware abstraction with device trees, and integration of modern languages such as Rust, Python, and C/C++. With a practical focus, the text thoroughly addresses containerization, CI/CD pipelines, and scalable deployment strategies for both edge and industrial IoT scenarios, while also highlighting security, resilience, and lifecycle maintenance for long-term reliability. Beyond technical essentials, \"BeagleBone Systems and Applications\" empowers professionals to build sophisticated solutions in edge computing, AI, automation, robotics, and large-scale field operations. Detailed sections guide readers through industrial networking, machine learning acceleration, robotics algorithms, and fleet management for mass production. From the prototyping bench to factory floors and autonomous robotic systems, this book is an indispensable reference for those seeking robust, secure, and scalable embedded system design with BeagleBone platforms.

## The Linux Kernel Module Programming Guide

BeagleBone Systems and Applications