嵌入式系统开发（影印版）

Making Embedded Systems

O'REILLY®
東南大學出版社

Elecia White 著

嵌入式系统开发（影印版）

Making Embedded Systems

Elecia White 著

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

O'Reilly Media, Inc.授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

嵌入式系统开发：英文/（美）怀特（White, E.）著. —影印本. —南京：东南大学出版社，2012.6

书名原文：Making Embedded Systems

ISBN 978-7-5641-3450-1

I. ①嵌… II. ①怀… III. ①微型计算机－系统设计－英文 IV. ① TP360.2

中国版本图书馆 CIP 数据核字（2012）第 084977 号

江苏省版权局著作权合同登记

图字：10-2012-160 号

嵌入式系统开发（影印版）

出版发行：东南大学出版社
地　　址：南京四牌楼 2 号　　邮编：210096
出 版 人：江建中
网　　址：http://www.seupress.com
电子邮件：press@seupress.com
印　　刷：扬中市印刷有限公司
开　　本：787 毫米 × 980 毫米　16 开本
印　　张：20.5
字　　数：401 千字
版　　次：2012 年 6 月第 1 版
印　　次：2012 年 6 月第 1 次印刷
书　　号：ISBN 978-7-5641-3450-1
定　　价：58.00 元（册）

本社图书若有印装质量问题，请直接与营销部联系。电话（传真）：025-83791830

O'Reilly Media, Inc.介绍

O'Reilly Media通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自1978年开始，O'Reilly一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了Make杂志，从而成为DIY革命的主要先锋；一如既往地通过多种形式缔结信息与人的纽带。O'Reilly的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly现在还将先锋专家的知识传递给普通的计算机用户。无论是书籍出版、在线服务还是面授课程，每一项O'Reilly的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar博客有口皆碑。”

——Wired

“O'Reilly凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——Business 2.0

“O'Reilly Conference是聚集关键思想领袖的绝对典范。”

——CRN

“一本O'Reilly的书就代表一个有用、有前途、需要学习的主题。”

——Irish Times

“Tim是位特立独行的商人，他不光放眼于最长远、最广阔的视野，并且切实地按照Yogi Berra的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去，Tim似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——Linux Journal

出版说明

随着计算机技术的成熟和广泛应用，人类正在步入一个技术迅猛发展的新时期。计算机技术的发展给人们的工业生产、商业活动和日常生活都带来了巨大的影响。然而，计算机领域的技术更新速度之快也是众所周知的，为了帮助国内技术人员在第一时间了解国外最新的技术，东南大学出版社和美国 O'Reilly Meida, Inc. 达成协议，将陆续引进该公司的代表前沿技术或者在某专项领域享有盛名的著作，以影印版或者简体中文版的形式呈献给读者。其中，影印版书籍力求与国外图书“同步”出版，并且“原汁原味”展现给读者。

我们真诚地希望，所引进的书籍能对国内相关行业的技术人员、科研机构的研究人员和高校师生的学习和工作有所帮助，对国内计算机技术的发展有所促进。也衷心期望读者提出宝贵的意见和建议。

最新出版的影印版图书，包括：

- 《Hbase 权威指南》（影印版）
- 《社交应用编程》（影印版）
- 《深入浅出 jQuery》（影印版）
- 《深入浅出 HTML5 编程》（影印版）
- 《创新的神话 第二版》（影印版）
- 《HTML5 应用编程》（影印版）
- 《HTML5 Cookbook》（影印版）
- 《Perl 语言编程 第四版》（影印版）
- 《深入浅出移动互联网》（影印版）
- 《易读代码的艺术》（影印版）
- 《嵌入式系统开发》（影印版）
- 《iOS 5 编程 Cookbook》（影印版）
- 《高性能 MySQL 第三版》（影印版）
- 《iOS 应用安全攻防》（影印版）

Preface

I love embedded systems. The first time a motor turned because I told it to, I was hooked. I quickly moved away from pure software and into a field where I can touch the world. Just as I was leaving software, the seminal work was done on design patterns.* My team went through the book, discussing the patterns and where we'd consider using them. As I got more into embedded systems, I found compilers that couldn't handle C++ inheritance, processors with absurdly small amounts of memory in which to implement the patterns, and a whole new set of problems where design patterns didn't seem applicable. But I never forgot the idea that there are patterns to the way we do engineering. By learning to recognize the patterns, we can use the robust solutions over and over. So much of this book looks at standard patterns and offers some new ones for embedded system development. I've also filled in a number of chapters with other useful information not found in most books.

About This Book

After seeing embedded systems in medical devices, race cars, airplanes, children's toys, and gunshot location systems, I've found a lot of commonalities. There are a lot of things I wish I knew then on how to go about designing and implementing software for an embedded system. This book contains some of what I've learned. It is a book about good software design in resource-constrained environments.

It is also a book about understanding what interviewers look for when you apply for an embedded systems job. Each section ends with an interview question. These are generally not language specific; instead, they attempt to divine how you think. Good interview questions don't have a single correct answer. Instead of trying to document all the paths, the notes after each question provide hints about what an interviewer might look for in your response. You'll have to get the job (and the answers) on your own merits.

* Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides, 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston: Addison-Wesley.

One note, though: my embedded systems don't have operating systems. The software runs on the bare metal. When the software says "turn that light on," it says it to the processor without an intermediary. This isn't a book about an embedded operating system (OS). But the concepts translate to processors running OSs, so if you stick around, you may learn about the undersides of OSs too. Working without one helps you really appreciate what an OS does.

This book describes the archetypes and principles that are commonly used in creating embedded system software. I don't cover any particular platform, processor, compiler, or language, because if you get a good foundation from this book, specifics can come later.

About the Author

In the field of embedded systems, I have worked on DNA scanners, inertial measurement units for airplanes and race cars, toys for preschoolers, a gunshot location system for catching criminals, and assorted medical and consumer devices.

I have specialized in signal processing, hardware integration, complex system design, and performance. Having been through FAA and FDA certification processes, I understand the importance of producing high-quality designs and how they lead to high-quality implementations.

I've spent several years in management roles, but I enjoy hands-on engineering and the thrill of delivering excellent products. I'm happy to say that leaving management has not decreased my opportunities to provide leadership and mentoring.

Acknowledgments

This book didn't happen in a vacuum. It started with a colleague who said, "Hey, do you know a book I can give to one of my junior engineers?" From that planted seed came months of writing; I learned to really appreciate understanding (and encouraging) friends. Then there were the engineers who gave their time to look at the technical material (any remaining issues are my fault, not theirs, of course). Finally, O'Reilly provided tremendous support through the whole process.

Thanking each person as they properly deserve would take pages and pages, so I will just go through them all in one breath, in no particular order: Phillip King, Ken Brown, Jerry Ryle, Matthew Hughes, Eric Angell, Scott Fitzgerald, John Catsoulis, Robert P. J. Day, Rebecca Demarest, and Jen Costillo. These folks made a difference in the flavor of this book. There are additional thank-yous spread throughout the book, where I got help from a particular person on a particular area, so you may see these names again (or a few different ones).

Two people are left out of that list. Andy Oram is an excellent editor at O'Reilly, and I was lucky to get him. He's made a big difference to this book and to my writing.

Finally, authors always give gushing thanks to their spouses; it is a cliché. However, having written a book, I see why. Christopher White, my favorite drummer, physicist, and embedded systems engineer, thank you most of all. For everything.

Organization of This Book

I read nonfiction for amusement. I read a lot more fiction than nonfiction, but still, I like any good book. I wrote this book to be read almost as a story, from cover to cover. The information is technical (extremely so in spots), but the presentation is casual. You don't need to program along with it to get the material (though trying out the examples and applying the recommendations to your code will give you a deeper understanding).

This isn't intended to be a technical manual where you can skip into the middle and read only what you want. I mean, you can do that, but you'll miss a lot of information with the search-and-destroy method. You'll also miss the jokes, which is what I really would feel bad about. I hope that you go through the book in order. Then, when you are hip-deep in alligators and need to implement a function fast, pick up the book, flip to the right chapter, and, like a wizard, whip up a command table or fixed point implementation of variance.

Or you can skip around, reading about solutions to your crisis of the week. I understand. Sometimes you just have to solve the problem. If that is the case, I hope you find the chapter interesting enough to come back when you are done fighting that fire.

The order of chapters is:

Chapter 1, Introduction
: What is an embedded system? How is development different from traditional software?

Chapter 2, Creating a System Architecture
: How to create (and document) a system architecture.

Chapter 3, Getting Your Hands on the Hardware
: Hardware/software integration during board bring-up can be scary, but there are some ways to make it smoother.

Chapter 4, Outputs, Inputs, and Timers
: The embedded systems version of "Hello World" is making an LED blink. It can be more complex than you might expect.

Chapter 5, Managing the Flow of Activity
: This chapter describes how to set up your system, where to use interrupts (and how not to), and how to make a state machine.

Chapter 6, Communicating with Peripherals
: Different serial communication forms rule embedded systems (UART, SSP, SPI, I^2C, USB, etc.). Networking, bit-bang, and parallel buses are not to be discounted.

Chapter 7, Updating Code
: When you need to replace the program running in your processor, you have a few options ranging from internal updaters to building your own solution.

Chapter 8, Doing More with Less
: This covers methods for reducing consumption of RAM, code space, and processor cycles.

Chapter 9, Math
: Most embedded systems need to do some form of analysis. Understanding how mathematical operations and floating points work (and don't work) will make your system faster.

Chapter 10, Reducing Power Consumption
: From reducing processor cycles to system architecture suggestions, this chapter will help you if your system runs on batteries.

The information is presented in the order that I want my engineers to start thinking about these things. It may seem odd that architecture is first, considering that most people don't get to it until later in their careers. However, I want the people I work with to be thinking about how their code fits in the system long before I want them to worry about optimization.

Conventions Used in This Book

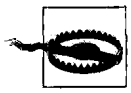
Typography

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, data types, and keywords.

This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.

Terminology

A *microcontroller* is a processor with onboard goodies like RAM, code space (usually flash), and various peripheral interfaces (e.g., I/O lines). Your code runs on a processor, or *central processing unit* (CPU). A *microprocessor* is a small processor, but the definition of "small" changes.

A DSP (*digital signal processor)* is a specialized form of microcontroller that focuses on signal processing, usually sampling analog signals and doing something interesting with the result. Usually a DSP is also a microcontroller, but it has special tweaks to make it perform math operations faster (in particular, multiply and add).

As I wrote this book, I wanted to use the correct terminology so you'd get used to it. However, with so many names for the piece of the system that is running your code, I didn't want to add confusion by changing the name. So, I stick with the term *processor* to represent whatever it is you're using to implement your system. Most of the material is applicable to whatever you actually have.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Making Embedded Systems* by Elecia White (O'Reilly). Copyright 2012 Elecia White, 978-1-449-30214-6."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at *http://my.safaribooksonline.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://shop.oreilly.com/product/0636920017776.do

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our website at:

http://www.oreilly.com

About the Author

Elecia White has worked on DNA scanners, inertial measurement units for airplanes and race cars, toys for preschoolers, a gunshot location system for catching criminals, and assorted other medical and consumer devices. She is the founder of Logical Elegance, an embedded systems consulting company based in San Jose. Elecia has developed strong skills in signal processing, hardware integration, complex system design, and performance. Having been through FAA and FDA certification processes, she understands the importance of producing quality designs and how they lead to quality implementations.

Elecia has spent several years in management roles but enjoys hands-on engineering and the thrill of delivering excellent products. While continuing to provide leadership and mentoring, she prefers to focus on the technical aspects of a project. A graduate of Harvey Mudd College in Claremont, CA, Elecia enjoys sharing her passion for science, engineering, and interesting gizmos, particularly how these things can make the world a better place.

Colophon

The animal on the cover of *Making Embedded Systems* is a great-eared goatsucker.

Great-eared goatsuckers are members of the family *Caprimulgidae*, informally referred to as "nightjars." The term "goatsucker" stems from an inaccurate belief that the birds drink milk from goats; it is not to be confused with "chupacabra," the fabled cryptid believed by some to inhabit the Americas and drink the blood of goats. The name "great-eared nightjar" is more commonly used today.

Great-eared nightjars inhabit subtropical or tropical moist lowland forests in areas of Southeast Asia, including India, Bangladesh, Myanmar, Thailand, Malaysia, Indonesia, Laos, Cambodia, Vietnam, the Philippines, and China. They are crepuscular, meaning mostly active at dusk and at night, during which time they satisfy their diets of flying insects and moths and sound their distinctive calls: a sharp "tissk," followed by a two-syllable "ba-haw."

Great-eared nightjars are characterized by pronounced ear tufts, which, in addition to their average length of 16 inches, make them more conspicuous among nightjars. Their soft gray and brown plumage resembles their preferred habitat, leaf litter and bracken. They lay their eggs either directly on bare ground or in leaf-litter nests on the ground. Nightjar nestlings have been observed to be completely silent and motionless, which, in addition to their leaf-colored plumage, may help protect them from danger while nesting.

The cover image is from Wood's *Animate Creation*. The cover font is Adobe ITC Garamond. The text font is Linotype Birka; the heading font is Adobe Myriad Condensed; and the code font is LucasFont's TheSansMonoCondensed.

Table of Contents