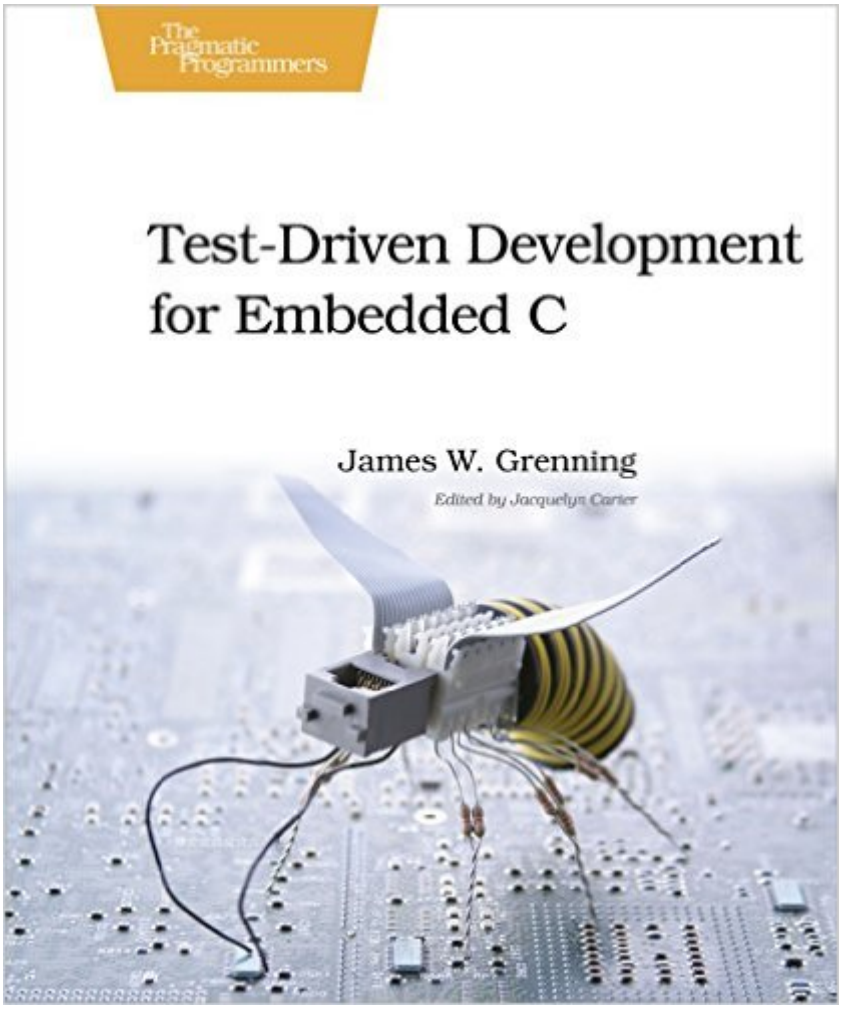
Test Driven Development For Embedded C (Pragmatic Programmers)



Synopsis

Another day without Test-Driven Development means more time wasted chasing bugs and watching your code deteriorate. You thought TDD was for someone else, but it’s not! It’s for you, the embedded C programmer. TDD helps you prevent defects and build software with a long useful life. This is the first book to teach the hows and whys of TDD for C programmers.TDD is a modern programming practice C developers need to know. It’s a different way to program---unit tests are written in a tight feedback loop with the production code, assuring your code does what you think. You get valuable feedback every few minutes. You find mistakes before they become bugs. You get early warning of design problems. You get immediate notification of side effect defects. You get to spend more time adding valuable features to your product.James is one of the few experts in applying TDD to embedded C. With his 1.5 decades of training,coaching, and practicing TDD in C, C++, Java, and C# he will lead you from being a novice in TDD to using the techniques that few have mastered.This book is full of code written for embedded C programmers. You don’t just see the end product, you see code and tests evolve. James leads you through the thought process and decisions made each step of the way. You’ll learn techniques for test-driving code right nextto the hardware, and you’ll learn design principles and how to apply them to C to keep your code clean and flexible.To run the examples in this book, you will need a C/C++ development environment on your machine, and the GNU GCC tool chain or Microsoft Visual Studio for C++ (some project conversion may be needed).

Book Information

Series: Pragmatic Programmers

Paperback: 352 pages

Publisher: Pragmatic Bookshelf; 1 edition (May 5, 2011)

Language: English

ISBN-10: 193435662X

ISBN-13: 978-1934356623

Product Dimensions: 7.5 x 0.8 x 9.2 inches

Shipping Weight: 1.3 pounds (View shipping rates and policies)

Average Customer Review: 4.7 out of 5 starsÂ Â See all reviewsÂ (28 customer reviews)

Best Sellers Rank: #212,717 in Books (See Top 100 in Books) #20 inÂ Books > Computers & Technology > Hardware & DIY > Microprocessors & System Design > Embedded Systems #83 inÂ Books > Computers & Technology > Programming > Languages & Tools > C & C++ > C #89

inÂ Books > Computers & Technology > Programming > Software Design, Testing & Engineering > Testing

Customer Reviews

As the other reviews have stated, this is a very good book. I had been looking for a book like this for a while, so I first picked this up in a beta version from the Pragmatic Programmers website.The two first section give a wonderful introduction to TDD in C. By the end of the second section, Grenning has covered the reasons for doing TDD, looked at available tools, and introduced various methods (spies, test doubles, mocks) for breaking module dependencies during testing. Lots of code examples are included throughout. These two sections were by far the most useful to me. Having been a programmer for a number of years without doing TDD, I needed some convincing, so the "Yeah, but..." chapter was spot on.The third section (Design and Continuous Improvement) feels a little bit more unfocused. It covers three rather large topics (SOLID design, refactoring, and working with legacy code) that all deserve (and have) whole books dedicated to them. It may be intended as further examples of how to apply TDD, and it does do a fine job of that.In short, I think this book serves as a very good introduction to the topic. That does not mean, however, that it answered all my questions about TDD. Most of these question revolve around how these techniques scale up to larger projects and teams.Two examples: * In Chapter 10 it is stated that "Mocks enforce a strict ordering of interactions, which can lead to fragile tests ...". I would have loved to read some thoughts on when this is likely to occur, possible solutions, etc. * The LED driver example is a good example, but it isn't immediately obvious how this approach would scale to larger hardware blocks (say, a co-processor).

For those who don't want to go through the entire review, here's the summary: Despite including "Embedded C" in the title, this book does not include anything at all particular to embedded programming. This is an introduction to TDD book, and in my opinion not such a good one either.Here's the breakdown:I'm a firmware developer so I picked this book up because, (1) I wanted to learn TDD and, (2) I wanted to learn how to apply it to embedded programming. So I thought I can kill two birds with one stone buying this book. Sadly this book does a very poor job at both. In my opinion, anybody picking up a book on TDD is not a beginner in programming. This is a place the book gets things wrong first. It is unnecessarily overly verbose at times, explaining simple things duplicating before and after code snippets. On the other hand, some important points are not explaining enough. For example, the only two points I found useful in this book was link-time

substitution and function pointer use. These are not new to a programmer, but I felt are very useful when applying TDD, especially when working with existing legacy code (which most of us will have to work on one time). But the book doesn't explain them in detail (as it does other very trivial topics).And for the biggest dissapointement, this book has nothing special for embedded programming. The closest the author gets to an embedded system are the two exaplme projects he presents in the book, the LedDriver and the LightScheduler. These two are very simple to qualify as an embedded system, because usually an embedded system is much more complex than turning on an LED at the given time.An embedded system program differs from a normal program in many ways.

Test-Driven Development for C does exactly what the title promises you. It describes how to do Test-Driven Development in the C programming language. People have argued that Agile development is for modern projects, but not embedded ones. Test-Driven Development can work in Object-Oriented languages but not in programming languages like C. James proves this wrong by showing how you can test-drive you code in C.The book consists of 4 different parts of which the last part are the appendices, which I'll skip in this review. The first part covers the basics of TDD, the second part discusses how to test a module that has dependencies with other modules. The third part discusses the design aspects of TDD.The first chapter introduces the concept of test-driven development after which the author continues introducing the two unit test frameworks used in the book: Unity and CppUTest. In the third chapter, the LED example is introduced and used to clarify TDD. The fifth chapter dives in the embedded space and discusses dual targeting and other embedded C techniques. The first part ends with an summary of objections that people typically have against TDD and an counter argument for each other them.The second part continues with a more complicated example (light automation system). This system has multiple modules and thus each of the modules need to be separated to be able to test it. Chapter 8 discusses link-time substitution and chapter 9 then dives into how to do this at run-time. Chapter 10 introduces Mock objects by first writing one by hand, and then introducing CppUTest mocking and CMock.The last part dives into design. In the end, TDD is a design technique, so a TDD book couldn't do without diving deeper into design.

Release It!: Design and Deploy Production-Ready Software (Pragmatic Programmers) OpenGL ES 2 for Android: A Quick-Start Guide (Pragmatic Programmers) Language Implementation Patterns: Create Your Own Domain-Specific and General Programming Languages (Pragmatic Programmers) Debug It!: Find, Repair, and Prevent Bugs in Your Code (Pragmatic Programmers) Practical Vim: Edit Text at the Speed of Thought (Pragmatic Programmers) Good Math: A Geek's Guide to the Beauty of Numbers, Logic, and Computation (Pragmatic Programmers) Android 6 for Programmers: An App-Driven Approach (Deitel Developer Series) iOS 8 for Programmers: An App-Driven Approach with Swift (3rd Edition) (Deitel Developer Series) DSP Software Development Techniques for Embedded and Real-Time Systems (Embedded Technology) Private Pilot Test Prep 2017: Study & Prepare: Pass your test and know what is essential to become a safe, competent pilot — from the most trusted source in aviation training (Test Prep series) Remote Pilot Test Prep — UAS: Study & Prepare: Pass your test and know what is essential to safely operate an unmanned aircraft – from the most trusted source in aviation training (Test Prep series) Commercial Pilot Test Prep 2017: Study & Prepare: Pass your test and know what is essential to become a safe, competent pilot — from the most trusted source in aviation training (Test Prep series) Instructor Test Prep 2017: Study & Prepare: Pass your test and know what is essential to become a safe, competent pilot — from the most trusted source in aviation training (Test Prep series) Applied Control Theory for Embedded Systems (Embedded Technology) Design Patterns for Embedded Systems in C: An Embedded Software Engineering Toolkit Analog Interfacing to Embedded Microprocessor Systems, Second Edition (Embedded Technology Series) Real-Time UML Workshop for Embedded Systems, Second Edition (Embedded Technology)