The Pragmatic Programmers

Programming Your Home

Automate with Arduino, Android, and Your Computer

Mike Riley

Edited by Jacquelyn Carter

Praise for *Programming Your Home*

Mike has a broad technology experience base that puts all the pieces of some remarkable projects together. It's amazing that he makes it all so easy and affordable. Don't miss all that can be learned from this gem.

➤ **Michael Bengtson, Consultant**

The Web-Enabled Light Switch project gave my family convenience and security options and enhanced my knowledge of RS-232 communications. It is nice to be able to switch on lights from my favorite chair. And the Tweeting Bird Feeder project has opened my eyes to the uses of radio communications around the home for things besides Wi-Fi, and it will help in my work to contribute to the preservation of bird species that are struggling for food and habitat.

➤ **Bob Cochran, Information Technology Specialist**

With this book, Mike Riley celebrates the Arduino microcontroller in a way that both beginning and advanced home automation hobbyists will enjoy.

➤ **Sven Davies, Vice President of Applications**

This is an outstanding reference that should be on the desk of every DIYer. In much the same way that software engineers mention "The Gang of Four Patterns Book," I predict this text will eventually be referred to as "The Riley Book of Home Automation."

➤ **Jon Kurz, President, Dycet, LLC**

Every technology is only as exciting as the things you do with it. Mike takes a few cheap electronics parts, an Arduino, and a bit of code and turns your home into a much more exciting and enjoyable place. His easy-to-follow instructions make every single one of these projects both fun and useful.

➤ **Maik Schmidt, Software Developer, Author of *Arduino: A Quick-Start Guide***

I've had more fun learning new languages, systems, and gadgets with this book than any other book I've read!

➤ **James Schultz, Software Developer**

Home automation is great fun, and *Programming Your Home* by Mike Riley will get you started right away. By leveraging this book and the easily available free/inexpensive hardware and software, anyone can tackle some great projects.

➤ **Tony Williamitis, Senior Embedded Systems Engineer**

This is a fun and enthusiastic survey of electronic devices that can interact with the real world and that starts in your own home!

➤ **John Winans, Chief Software Architect**

Programming Your Home

Automate with Arduino, Android, and Your Computer

Mike Riley

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at *http://pragprog.com*.

The team that produced this book includes:

Jackie Carter (editor)
Potomac Indexing, LLC (indexer)
Molly McBeath (copyeditor)
David J Kelly (typesetter)
Janet Furlow (producer)
Juliet Benda (rights)
Ellie Callahan (support)

Printed in the United States of America.
ISBN-13: 978-1-93435-690-6
Printed on acid-free paper.
Book version: P1.0—February 2012

This book is dedicated to Bill, Eileen, and Josie.

Contents

Acknowledgments xi

Preface xiii

Part I — Preparations

1. Getting Started 3
   - 1.1 What Is Home Automation? 3
   - 1.2 Commercial Solutions 4
   - 1.3 DIY Solutions 5
   - 1.4 Justifying the Investment 5
   - 1.5 Setting Up Your Workbench 6
   - 1.6 Sketching Out Your Ideas 7
   - 1.7 Writing, Wiring, and Testing 8
   - 1.8 Documenting Your Work 9

2. Requirements 11
   - 2.1 Knowing the Hardware 12
   - 2.2 Knowing the Software 17
   - 2.3 Be Safe, Have Fun! 18

Part II — Projects

3. Water Level Notifier 23
   - 3.1 What You Need 23
   - 3.2 Building the Solution 26
   - 3.3 Hooking It Up 26
   - 3.4 Sketching Things Out 27
   - 3.5 Writing the Web Mailer 34
   - 3.6 Adding an Ethernet Shield 36

3.7 All Together Now 40
3.8 Next Steps 41

4. Electric Guard Dog 45
4.1 What You Need 46
4.2 Building the Solution 47
4.3 Dog Assembly 48
4.4 Dog Training 52
4.5 Testing It Out 55
4.6 Unleashing the Dog 56
4.7 Next Steps 57

5. Tweeting Bird Feeder 59
5.1 What You Need 59
5.2 Building the Solution 62
5.3 The Perch Sensor 63
5.4 The Seed Sensor 67
5.5 Going Wireless 70
5.6 Tweeting with Python 75
5.7 Putting It All Together 83
5.8 Next Steps 84

6. Package Delivery Detector 87
6.1 What You Need 88
6.2 Building the Solution 90
6.3 Hardware Assembly 91
6.4 Writing the Code 92
6.5 The Package Delivery Sketch 92
6.6 Testing the Delivery Sketch 94
6.7 The Delivery Processor 95
6.8 Creating the Delivery Database 95
6.9 Installing the Package Dependencies 97
6.10 Writing the Script 98
6.11 Testing the Delivery Processor 102
6.12 Setting It Up 104
6.13 Next Steps 105

7. Web-Enabled Light Switch 107
7.1 What You Need 107
7.2 Building the Solution 110
7.3 Hooking It Up 111

7.4 Writing the Code for the Web Client 114
7.5 Testing Out the Web Client 116
7.6 Writing the Code for the Android Client 117
7.7 Testing Out the Android Client 121
7.8 Next Steps 124

8. Curtain Automation 127
8.1 What You Need 127
8.2 Building the Solution 130
8.3 Using the Stepper Motor 131
8.4 Programming the Stepper Motor 132
8.5 Adding the Sensors 133
8.6 Writing the Sketch 134
8.7 Installing the Hardware 139
8.8 Next Steps 142

9. Android Door Lock 143
9.1 What You Need 143
9.2 Building the Solution 146
9.3 Controlling the Android Door Lock 150
9.4 Writing the Android Server 154
9.5 Writing the Android Client 166
9.6 Test and Install 170
9.7 Next Steps 171

10. Giving Your Home a Voice 173
10.1 What You Need 173
10.2 Speaker Setup 175
10.3 Giving Lion a Voice 177
10.4 Wireless Mic Calibration 179
10.5 Programming a Talking Lion 181
10.6 Conversing with Your Home 190
10.7 Next Steps 191

Part III — Predictions

11. Future Designs 195
11.1 Living in the Near 195
11.2 The Long View 198
11.3 The Home of the Future 200

12. More Project Ideas 203
12.1 Clutter Detector 203
12.2 Electricity Usage Monitor 204
12.3 Electric Scarecrow 204
12.4 Entertainment System Remote 204
12.5 Home Sleep Timer 205
12.6 Humidity Sensor-Driven Sprinkler System 205
12.7 Networked Smoke Detectors 205
12.8 Proximity Garage Door Opener 206
12.9 Smart HVAC Controller 207
12.10 Smart Mailbox 207
12.11 Smart Lighting 207
12.12 Solar and Wind Power Monitors 207

Part IV — Appendices

A1. Installing Arduino Libraries 211
A1.1 Apple OSX 211
A1.2 Linux 212
A1.3 Windows 212

A2. Bibliography 213

Index 215

Acknowledgments

I have been a lifelong tinkerer. My earliest recollection of dissecting my father's broken tape recorder instilled an appreciation for the technology that drove it. From there, erector sets, model railroads, and programmable calculators led to personal computers, mobile devices, and microcontrollers. Over the years, this passion for learning not only how stuff works but also how technical concepts can be remixed with surprising, often highly satisfying results has been liberating. That's why this book was such a joy for me to write.

Helping others to see what's possible by observing their surroundings and having the desire to take an active role in making their lives easier with technology while having fun is this book's primary goal. Yet without others helping me distill my ideas into what you are reading now, this book would not have been possible. It is to them that I wish to express my deepest gratitude for their support.

A boatload of thanks goes to the book's editor, Jackie Carter, who spent countless hours ensuring that my words were constructed with clarity and precision. Copy editor Molly McBeath did a fantastic job catching hidden (from my view anyway) typos and grammatical misconstructions. Big thanks to Susannah Pfalzer for her infectious enthusiasm and boundless boosts of encouragement and to Arduino expert and fellow Pragmatic author Maik Schmidt, whose own success helped pave the way for a book like this.

Many thanks also go to John Winans, tech wiz extraordinaire, who refactored the state machine code used in several of the projects, as well as to Sven Davies, Mike Bengtson, Jon Bearscove, Kevin Gisi, Michael Hunter, Jerry Kuch, Preston Patton, and Tony Williamitis for helping to make this book as technically accurate and complete as it is. Shout-outs also go to Jon Erikson and Jon Kurz for their enthusiastic encouragement. I also want to thank Bob Cochran and Jim Schultz for providing wonderfully helpful feedback during the book's beta period. Thanks also go to Philip Aaberg for filling my ears with

music to code by. And to the makers of and contributors to the Arduino and Fritzing projects, you people have changed the world for the better.

I am most grateful to my wife, Marinette, and my family for allowing me to tunnel away for months in my mythical man cave to complete this book. And I can't gush enough over the wonderful pencil illustrations that my daughter drew for the book. I am so proud of you, Marielle!

Finally, I am sincerely thankful to Dave Thomas and Andy Hunt for their passion and vision. You're the best.

Mike Riley
`mailto:mike@mikeriley.com`
Naperville, IL, December 2011

Preface

Welcome to the exciting, empowering world of home automation! If you have ever wanted your home to do more than just protect you against the outside elements and want to interface it to the digital domain, this book will show you how. By demonstrating several easy-to-build projects, you will be able to take the skills you learned from this book and expand upon and apply them toward custom home automation projects of your own design.

The book's primary objective is to get you excited about the broader possibilities for home automation and instill the confidence you need to ultimately build upon these and your own ideas. The projects also make great parent-child learning activities, as the finished products instill a great sense of accomplishment. And who knows? Your nifty home automation creations may even change the world and become a huge new business opportunity for other homeowners actively seeking an automation solution that saves them time and money.

Who Should Read This Book

Programming Your Home is best suited to DIYers, programmers, and tinkerers who enjoy spending their leisure time building high-tech solutions to further automate their lives and impress their friends and family with their creations. Essentially, it is for those who generally enjoy creating custom technology and electronics solutions for their own personal living space.

A basic understanding of Arduino and programming languages like Ruby and Python are recommended but not required. You will learn how to combine these technologies in unique configurations to resolve homemaker annoyances and improve home management efficiencies.

In addition to the inclusion of Python scripts and Ruby on Rails-based web services, several of the projects call upon Google's Android platform to help enhance the data event collection, visualization, and instantiation of activities.

A basic familiarity with the Android SDK will be beneficial so that the projects that make use of the Android OS can offer a more mobile reach.

If you're the type of person who prefers to build versus buy your home accessories, this book will further motivate you to use what you learned in the book as a starting point to expand upon and optimize them in various ways for their environment. Even though some of the topics deal with multiple software- and hardware-based solutions, they are easy to follow and inexpensive to build. Most of all, they show how a few simple ideas can transform a static analog environment into a smart digital one while having fun.

What's in This Book

After a basic introduction to home automation and the tools of the trade, this book will teach you how to construct and program eight unique projects that improve home utility and leisure-time efficiencies. Each project incorporates a variety of inexpensive sensors, actuators, and microcontrollers that have their own unique functions. You will assemble the hardware and codify the software that will perform a number of functions, such as turning on and off power switches from your phone, detecting package deliveries and transmitting emails announcing their arrival, posting tweets on Twitter when your bird feeder needs to be refilled, and opening and closing curtains depending on light and temperature, and more.

Because the recommended skill set for building these solutions includes some familiarity with programming, this book builds upon several previously published Pragmatic Bookshelf titles. If you would like to learn more about programming Arduinos or writing Ruby or Python scripts, I strongly recommend checking out the books listed in Appendix 2, *Bibliography*, on page 213.

Each project begins with a general introduction and is followed by a What You Need section that lists the hardware parts used. This is followed by a section called Building the Solution that provides step-by-step instructions on assembling the hardware. *Programming Your Home* will call upon the Arduino extensively for most (but not all) of the projects. Once the hardware is constructed, it can be programmed to perform the automation task we built it to do. Programs can range from code for Arduino microcontrollers to scripts that execute on a computer designed to control, capture, and process the data from the assembled hardware elements.

The book concludes with a chapter on future projections in home automation and a chapter filled with idea starters that reuse the hardware and software approaches demonstrated in the eight projects.

Arduinos, Androids, and iPhones, Oh My!

With the meteoric rise of mobile device proliferation, the post-PC moniker has made its way into the tech world's vocabulary. I am a big proponent of technology shifts, but I am also old enough to have lived through three major computing revolutions (the shift from mainframes to PCs, the rise of the Internet, and the shift from PCs to mobile devices) and know that change isn't as fast as people say it is. Until mobile applications can be developed on mobile devices the way PC applications can be developed on PCs, a Linux, Windows, or Mac computer will be a central requirement for developing mobile apps. The same holds true for Arduino programming.

That said, the times are indeed a-changing. Microsoft Research was one of the first major phone OS providers to attempt to create native mobile applications directly on the mobile device with their release of TouchStudio. Google engineer Damon Kohler created the Scripting Layer for Android (SL4A) that gives Android users the ability to write fairly sophisticated programs using a text editor on their phone. Coupled with Sparkfun's IOIO ("yo-yo") board, we're already seeing early glimpses of what could replace the PC for some of the scripts created for this book.

Since you will need a Mac, Linux, or Windows computer to program the Arduinos and mobile apps in this book, this computer will also be the machine that runs the server-side programs that interpret and extend information out to your mobile devices. Of course, if you only have one computer and it's a laptop that travels with you, consider purchasing a cheap Linux or Mac to run as your home server. Not only will you benefit from having a dedicated system to run the monitoring apps 24/7/365, but it can also serve as your home Network Attached Storage (NAS) server as well.

I am a believer in open source hardware and software. As such, the projects in the book depend upon these. I am also technology-agnostic and rarely have any overriding devotion to one hardware supplier or programming language. Code for this book could have been presented just as easily in Mono-based C# and Perl, but I opted for Ruby and Python because of their portability and multiparty open source support. I could have used a Windows or Linux machine as the server and development system but chose Mac for the book because Ruby and Python are preinstalled with the OS, thereby eliminating the time and space required to install, configure, and troubleshoot the operating environment.

In accordance with this open source philosophy, I also opted to demonstrate the mobile application examples exclusively for the Android OS. While I

personally prefer iOS devices as the platform of choice for my mobile lifestyle, the overhead associated with writing applications for iOS is a hassle. In addition to learning Objective-C and the various frameworks as well as dealing with the burden of memory management, deploying iOS apps requires either a jailbroken device or the legitimate purchase of an annual membership to Apple's iPhone developer network. Conversely, Android's SDK and application deployment is free and open. Android programs can also multitask better than iOS programs. Of course, these two advantages also bring on greater security and resource utilization risks. That said, I encourage readers who prefer the mobile demos to run on non-Android devices to port the simple client programs presented in this book to their favorite mobile OS and share these conversions with the Programming Your Home community.

Another term that is gaining a foothold in the tech press is the "Internet of Things." This phrase refers to the idea that with the proliferation of network-connected microcontrollers, Internet-based communication between such small devices will eventually outnumber people surfing the Web. While that may be the case for submitting data upstream, reaching such a device from the Internet is still a hassle. Besides the technical knowledge required to set up a dynamic DNS and securely configure port forwarding to easily reach the device, ISPs may block outbound ports to prevent end consumers from setting up dedicated servers on popular network ports like FTP, HTTP/S, and SMTP.

The projects in this book should work perfectly fine in a home local area network. However, obtaining sensor data outside of this local network is a challenge. How do you check on the status of something like a real-time temperature reading without going through the hassles of opening and forwarding ports on your router (not to mention the potential security risks that entails)?

Fortunately, several companies have begun to aggressively offer platforms accessible via simple web service APIs to help overcome these hassles. Three of these gaining momentum are Pachube, Exosite, and Yaler.[1] Configuring and consuming their services is a fairly straightforward process. I encourage you to visit these sites to learn more about how to incorporate their messaging capabilities into your own projects.

1. http://www.pachube.com, http://www.exosite.com, and http://www.yaler.org, respectively.

Code Examples and Conventions

The code in this book consists of C/C++ for Arduino, Java for Android, Ruby for web middleware, and Python for desktop scripts. Most of the code examples are listed in full, except when burdened by external library overhead (such as in the case of Android and Ruby on Rails program listings). Syntax for each of these languages is highlighted appropriately, and much of the code is commented inline along with bullet markings to help bring attention to the big ideas in the listings.

Highlights and sidebars are used sparingly in the book in an effort to keep information moving at a quick yet manageable clip.

Online Resources

Visit the book's website at http://pragprog.com/titles/mrhome, where you can download the code for all the projects, participate in the book's discussion forum, ask questions, and post your own home automation ideas. Bugs, typos, omissions, and other errors in the book can be found on the book's errata web page.

Other popular website resources include the popular DIY websites Makezine, and Instructables,[2] where participants share a wide variety of home-brewed creations with their peers.

There are also several IRC channels on freenode.net and SIG forums on Google Groups dedicated to the subject, with many focused on singular aspects of DIY gadget design, home automation, and hardware hacking.[3]

OK, enough with the preamble. Let's get ready to build something!

2. http://www.makezine.com and http://www.instructables.com, respectively.
3. http://groups.google.com/group/comp.home.automation/topics

Part I

Preparations

CHAPTER 1

Getting Started

Before we start wiring up hardware and tapping out code, let's lay down the foundation, starting with what exactly we mean by home automation, what's been available in the consumer space in the past, and why building our own solutions makes sense today and in the future.

We will also review a couple of design and construction best practices that will be put to good use when assembling the projects in this book.

We'll start by defining what we mean by home automation. Next we'll consider some of the prepackaged commercial solutions on the market, and then we'll take a quick snapshot of some of the more popular custom automation hardware and software projects. The chapter will conclude with some of the tools and practices that have helped me quite a bit when building the projects in this book as well as with other projects beyond the home automation category.

1.1 What Is Home Automation?

So what exactly does the term home automation mean? At its most basic level, it's a product or service that brings some level of action or message to the home environment, an event that was generated without the homeowner's direct intervention. An alarm clock is a home automation device. So is a smoke alarm. The problem is, these stand-alone devices don't use a standard network communication protocol, so they can't talk to one another the way that networked computers can.

One of my earliest memories of home automation was when the Mr. Coffee automatic drip coffee machine came out in the early 1970s. The joy this simple kitchen appliance brought my coffee-drinking parents was genuine. They were so pleased to know that when they woke up in the morning a

freshly brewed pot of coffee would be waiting for them. Who would have thought that such a simple concept as a coffee maker combined with an alarm clock would change their world?

Now that we're in the twenty-first century, rudimentary coffee makers are getting a makeover by tinkerers bolting network adapters, temperature sensors, and microcontrollers to make the brew at the right time and temperature and to send a text message alert that the beverage is ready for consumption. It's only a matter of time before manufacturers incorporate inexpensive electronics into their appliances that do what tinkerers have been doing with their home electronics for years. But a standard communication protocol among such devices remains elusive. Nevertheless, efforts are afoot by a number of home automation vendors to address that problem.

1.2 Commercial Solutions

The number of attempts to standardize home automation communication protocols has been ongoing nearly as long as Mr. Coffee has been in existence. One of the earliest major players was X10, a company that still offers basic and relatively inexpensive home automation solutions today. X10 takes advantage of existing electrical wiring in the home. It uses a simple pulse code protocol to transmit messages from the X10 base station or from a computer connected to an X10 communication interface. But problems with signal degradation, checksums, and return acknowledgments of messages, as well as X10's bulky hardware and its focus on controlling electrical current via on/off relay switches, have constrained X10's broader appeal.

Other residentially oriented attempts at standards, such as CEBus and Insteon, have been made, but none have attained broad adoption in the home. This is partly due to the chicken-and-egg problem of having appliance and home electronics manufacturers create devices with these interfaces and protocols designed into their products.

Most recently, Google has placed its bet on the Android operating system being embedded into smart devices throughout the home. Time will tell if Google will succeed where others have failed, but history is betting against it.

Rather than wait another twenty years for a winning standard to emerge, embedded computing devices exist today that employ standard TCP/IP to communicate with other computers. This hardware continues to drop to fractions of the prices they cost only a few years ago. So while the market continues to further commoditize these components, the time is now for

software developers, home automation enthusiasts, and tinkerers to design and implement their own solutions. The lucky few will uncover and market a cost-effective, compelling solution that will one day catch on like wildfire and finally provide the impetus to forever change our domestic lives.

1.3 DIY Solutions

The Do-It-Yourself category in home automation is more active today than ever before. The combination of inexpensive electronics with low-cost networked computers make this option extremely attractive. There's other reasons that make DIY an ideal pursuit. Unlike proprietary commercial offerings, the projects you build are not mysterious black boxes. You have the source code. You have the knowledge. You have the measurements, the metrics, and the methods.

Not only will you know how to build it, you will know how to troubleshoot, repair, and enhance. None of the commercial solutions can match exactly what you may need. Home automation vendors have to generalize their products to make them appeal to a large consumer base. By doing so, they don't have the luxury of creating one-off solutions that exactly match one customer's specific needs. But with some rudimentary knowledge and project construction experience, you'll gain the confidence to create whatever design matches your situation.

For example, the first project in this book builds a sump pit notifier that emails you when water levels exceed a certain threshold. While commercial systems have audible alarms, none that I have found at the local hardware store have the means to contact you via such messaging. And should you need to modify the design (add a bright flashing LED to visually broadcast the alert, for example), you don't need to purchase a whole new commercial product that includes this feature.

Walk around your house. Look for inefficiencies and repetitive tasks that drive you crazy the way George Bailey was with pulling off the loose finial on his staircase's newel post. Take note of what can be improved with a little ingenuity and automation. You may be surprised at just how many ideas you can quickly come up with.

1.4 Justifying the Investment

Let's be honest. Spending more money on parts that may or may not work well together versus buying a cheaper purpose-built device that meets or

exceeds the functionality of a homegrown solution is simply not a good investment. Sure, there may be some value derived from the knowledge gained from the design experience, the pleasure of building the solution, and the satisfaction of seeing your creation come to life. But justifying such an investment to a budget-conscious spouse, for example, may deflate whatever gains you have made in the satisfaction department.

When considering any new design approach, strive for a scenario where you will maximize your time, equipment investment, and learning potential. You may have to try several experiments and iterations before the hardware and software come together and work the way you envisioned. But if you keep at it, you will be well rewarded for your persistence. Not only will you achieve high points for devising a low-cost solution, but such constraints will help drive creativity to even higher levels. That's why I have tried my best to keep all the projects in this book within a reasonable budget, and I encourage reuse of old electronic parts and materials as much as possible.

Do your homework. Research online to see who may have attempted to build what you have in mind. Did they succeed? Was it worth the money and time they invested? Is there a commercially viable alternative?

If you determine that your idea is unique, put together an estimate of the expenses in terms of your time and of the materials you need to purchase. Remember to also include the cost of any tools you need to buy to construct and test the project's final assembly. This added expense is not negligible, especially if you're just starting down the DIY road. As you get more involved with hardware projects, you will quickly find that your needs will expand from an inexpensive soldering iron and strands of wire to a good quality multimeter and perhaps even an oscilloscope. But the nice thing about building your own solutions is that you can build them at your own pace. You will also find that as your network of DIYers grows, your opportunities for group discussion, equipment loans, insightful recommendations, and encouragement will grow exponentially.

1.5 Setting Up Your Workbench

Good assembly follows good design. Building these projects in a frustration-free environment will help keep your procedures and your sanity in check.

Work in a well-lit, well-ventilated area. This is especially important when soldering. Open a window and use a small fan to push the fumes outside. Use a soldering exhaust fan if an open window isn't an option.

If your work space can afford it, have a large table to spread out your electronic parts. Keep it close to power outlets and have a power strip on the table for easy access.

Organize your components with small craft containers, baby food jars, pill boxes, Altoids tins—anything that helps keep the variety of capacitors, resistors, LEDs, wires, shields, motors, and sensors sorted will make it much easier to keep track of your parts inventory.

Have your computer stationed near or on the work space. This is a no-brainer if it's a laptop. If it's a desktop, minimize its table footprint by only placing a monitor, mouse, and keyboard (both preferably wireless) on the table to leave as much unobstructed working space as possible.

Keep clutter away from underneath and around the table. Not only does this aid fire prevention, but doing so will also make it far easier to find that elusive component when it rolls off the table and bounces toward the unknown.

Lastly, keep the work space dedicated to project work. Some projects can be like building a jigsaw puzzle. You need a place for the half-assembled pieces to sit while life goes on. Being able to sit down and start working, rather than start unboxing and repackaging a fur ball of wires and parts, makes building projects a joy instead of a chore.

1.6 Sketching Out Your Ideas

When inspiration strikes, nothing beats old-fashioned pencil and paper to quickly draw out your ideas. For those who prefer to brainstorm their designs on a computer, several free, open source, cross-platform tools have helped me assemble my ideas and document my work:

- Freemind is great for organizing thoughts, objectives, and dependencies.[1] This mature mind-mapping application helps you make sense of a brain dump of ideas and see the links between them. This will save you time and money because you will be able to spot key ideas, eliminate redundancies, and prioritize what you want to accomplish.
- Fritzing is a diagraming application specifically designed for documenting Arduino-centric wiring.[2] Unfortunately, it's still a work in progress and is rough around the edges. It also doesn't have a number of the popular sensors iconically represented yet, but the object library is growing as

1. http://freemind.sourceforge.net
2. http://fritzing.org/

more people contribute to the project. I use this application exclusively for documenting my Arduino-based projects, which is why the wiring diagrams in this book were generated by Fritzing.

- Inkscape is an easy-to-use vector-based drawing program that helps sketch out ideas beyond the Arduino-centricity of Fritzing.[3] While Inkscape is mainly intended for graphic artists, it has accurate measurement tools that are great for scoping out bracket and enclosure ideas for your projects.

Going beyond the desktop, tablets are rapidly taking over the role that were once the domain of traditional paper uses. In fact, it wouldn't surprise me if you're reading this book on an iPad or a Kindle right now. Beyond just reference lookups, tablets are excellent for brainstorming ideas and creating initial sketches of preliminary project designs. An iPad (or Android tablet, for that matter) combined with a sturdy stand also makes for a handy electronic reference. Load up your sketches, track your progress, reorder priorities, and make notes along the way.

My current favorite iPad apps for my projects include the following:

- Elektor Electronic Toolbox is an electronic parts reference with a variety of helpful calculators and conversion tools.[4]
- iCircuit is a electronic circuit simulator that makes building and understanding circuits far easier than static diagrams on a printed page.[5]
- iThoughts HD is a mind-mapping application compatible with importing and exporting Freemind files.[6]
- miniDraw is a vector-based drawing program that can export to SVG format, perfect for importing your sketches into Inkscape.[7]

In addition to designing and documenting your projects, well-executed projects also rely on taking accurate measurements and running tests to validate your work.

1.7 Writing, Wiring, and Testing

Unfortunately, no good software emulator exists yet for the Arduino; fortunately, programs for this platform are usually small and specific enough such

3. http://inkscape.org
4. http://www.creating-your-app.de/electronic_toolbox_features.html?&L=1
5. http://icircuitapp.com/
6. http://ithoughts.co.uk
7. http://minidraw.net/