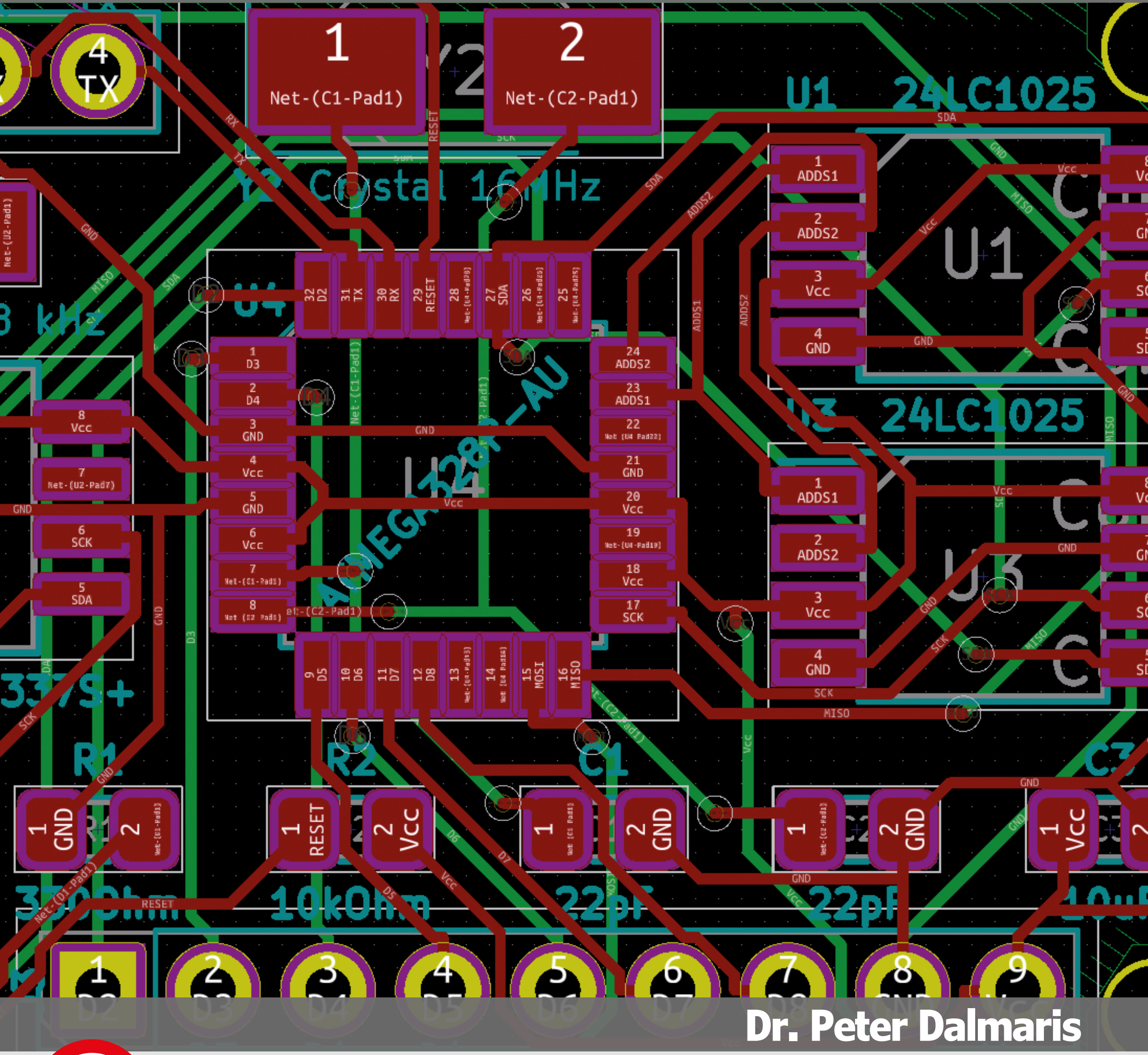
KiCad Like a Pro



Dr. Peter Dalmaris

elektor

LEARN DESIGN SHARE

LEARN • DESIGN • SHARE • LEARN • DESIGN • SHARE • LEARN • DESIGN • SHARE • LEARN • DESIGN • SHARE • LEARN • DESIGN • SHARE

Tech Explorations®

txplore.com

KiCad Like a Pro

●

Dr Peter Dalmaris

elektor LEARN DESIGN SHARE

● This is an Elektor Publication. Elektor is the media brand of
Elektor International Media B.V.
78 York Street, London W1H 1DP, UK
Phone: (+44) (0)20 7692 8344

● All rights reserved. No part of this book may be reproduced in any material form, including photocopying, or storing in any medium by electronic means and whether or not transiently or incidentally to some other sue of this publication, without the written permission of the copyright holder except in accordance with the provisions of the Copyright Designs and Patents Act 1988 or under the terms of a licence issued by the Copyright Licencing Agency Ltd., 90 Tottenham Court Road, London, England W1P 9HE. Applications for the copyright holder's permission to reproduce any part of the publication should be addressed to the publishers.

● Declaration
The author and publisher have used their best efforts in ensuring the correctness of the information contained in this book. They do not assume, or hereby disclaim, any liability to any party for any loss or damage caused by errors or omissions in this book, whether such errors or omissions result from negligence, accident or any other cause.

● British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

● Disclaimer
The material in this publication is of the nature of general comment only, and does not represent professional advice. It is not intended to provide specific guidance for particular circumstances and it should not be relied on as the basis for any decision to take action or not take action on any matter which it covers. Readers should obtain professional advice where appropriate, before making any such decision. To the maximum extent permitted by law, the author and publisher disclaim all responsibility and liability to any person, arising directly or indirectly from any person taking or not taking action based on the information in this publication.

● **ISBN 978-1-907920-74-5**

© Copyright 2018: Tech Explorations™
Prepress Production: D-Vision, Julian van den Berg
First published in Australia 2018
Printed in Germany by mediaprint

Elektor is part of EIM, the world's leading source of essential technical information and electronics products for pro engineers, electronics designers, and the companies seeking to engage them. Each day, our international team develops and delivers high-quality content - via a variety of media channels (e.g., magazines, video, digital media, and social media) in several languages - relating to electronics design and DIY electronics. **www.elektor.com**

LEARN DESIGN SHARE

Did you find an error?

Please let us know.

Using any web browser, go to txplo.re/KiCadbook, and fill in the form.

We'll get it fixed right away.

About the author

Dr. Peter Dalmaris is an educator, an electrical engineer, electronics hobbyist, and Maker. Creator of online video courses on DIY electronics and author of several technical books. Peter has recently released his book 'Maker Education Revolution', a book about how Making is changing the way we learn and teach in the 21st century.

As a Chief Tech Explorer since 2013 at Tech Explorations, the company he founded in Sydney, Australia, Peter's mission is to explore technology and help educate the world.

Tech Explorations offers educational courses and Bootcamps for electronics hobbyists, STEM students, and STEM teachers.

A lifelong learner, Peter's core skill lies in explaining difficult concepts through video and text. With over 15 years of tertiary teaching experience, Peter has developed a simple yet comprehensive style in teaching that students from all around the world appreciate.

His passion for technology and the world of DIY open-source hardware, has been a dominant driver that has guided his personal development and his work through Tech Explorations.

From the back cover

Printed circuit boards (PCB) are, perhaps, the most undervalued component of modern electronics. Usually made of fibreglass, PCBs are responsible for holding in place and interconnecting the various components that make virtually all electronic devices work.

The design of complex printed circuit boards was something that only skilled engineers could do. These engineers used to expensive computer-aided design tools. The boards they designed were manufactured in exclusive manufacturing facilities in large numbers.

Not anymore.

During the last 20 years, we have seen high-end engineering capabilities becoming available to virtually anyone that wants them. Computer-aided design tools and manufacturing facilities for PCBs are one mouse click away.

KiCad is one of those tools. Perhaps the world's most popular (and best) computer-aided design tool for making printed circuit boards, KiCad is open source, fully featured, well-funded and supported, well documented. It is the perfect tool for electronics engineers and hobbyists alike, used to create amazing PCBs. KiCad has reached maturity and is now a fully featured and stable choice for anyone that needs to design custom PCBs.

This book will teach you to use KiCad. Whether you are a hobbyist or an electronics engineer, this book will help you become productive quickly, and start designing your own boards.

Are you a hobbyist? Is the breadboard a bottleneck in your projects? Do you want to become skilled in circuit board design? If yes, then KiCad and this book are a perfect choice. Use KiCad to design custom boards for your projects. Don't leave your projects on the breadboard, gathering dust and falling apart.

Complete your prototyping process with a beautiful PCB and give your projects a high-quality, professional look.

Are you an electronics engineer? Perhaps you already use a CAD tool for PCB design. Are you interested in learning KiCad and experience the power and freedom of open-source software? If yes, then this book will help you become productive with KiCad very quickly. You can build on your existing PCB design knowledge and learn KiCad through hands-on projects.

This book takes a practical approach to learning. It consists of four projects of incremental difficulty and recipes.

The projects will teach you basic and advanced features of KiCad. If you have absolutely no prior knowledge of PCB design, you will find that the introductory project will teach you the very basics. You can then continue with the rest of the projects. You will design a board for a breadboard power supply, a tiny Raspberry Pi HAT, and an Arduino clone with extended memory and clock integrated circuits.

The book includes a variety of recipes for frequently used activities. You can use this part as a quick reference at any time.

The book is supported by the author via a page that provides access to additional resourc-es. Signup to receive assistance and updates.

How to read this book

I designed this book to be used both to learn how to use KiCad, and as a reference.

If you have never used KiCad and have little or no experience in PCB design, I recommend you read it in a linear fashion. Don't skip the early chapters in parts 1, 2 and 3, because those will set the fundamental knowledge on which you will build your skill later in the book. If you skip those chapters, you will have gaps in your knowledge that will make it harder for you to progress.

If you have a good working knowledge of PCB design, but you are new to KiCad, you can go straight to Part 2, zoom through it very quickly, and then proceed to the projects in Part 4. Once you have the basic KiCad concepts and skills confidently learned, you can use the recipes in Part 5 as a resource for specific problems you need solved. These recipes are useful on their own. Throughout the text, you will also find prompts to go to a particular recipe in order to learn a specific skill needed for the projects.

Images: Throughout this book, you will find numerous figures that contain screenshots of KiCad. To create these screenshots, I used KiCad 5 running on Linux Ubuntu. If you are using KiCad under Windows or Mac OS, do not worry: KiCad works the same across these platforms, and even looks almost the same.

Although I took care to produce images that are clear, there are cases were this was not possible. This is particularly true in screenshots of an entire application window, meant to be displayed in a large screen. The role of these images is to help you follow the instructions in the book as you are working on your computer. There is no substitute to experimenting and learning by doing, so the best advice I can give is to use this book as a text book and companion. Whenever you read it, have KiCad open on your computer and follow along with the instructions.

This book has a web page with resources designed to maximise the value it delivers to you, the reader. Please read about the book web page, what it offers and how to access it in the section 'The book web page', later in this introductory segment.

Finally, if you learn best by watching and following a video presentation, you may be interested in the video course version of this book. I estimate that this course will be ready by the end of 2018. Please check in the book web page for updates on this project. Be sure to subscribe to the Tech Explorations email list so I can send you updates.

Requirements

To make the most out of this book, you will need a few things. You probably already have them:

- A computer running Windows, Mac OS or Linux.
- Access to the Internet.
- A mouse with at least two buttons and a scroll wheel. I use a Logitech MX Master 2S mouse (see https://amzn.to/2ClySq0).
- Ability to install software.
- Time to work on the book, and patience.

The book web page

As a reader of this book, you are entitled access to its online resources.

You can access these resources by visiting the book's web page at http://txplo.re/klpr.

The two available resources are:

1. The book forum. This is a place where you can ask book-related questions and have a conversation about your projects. I will be spending time in the forum weekly, answering questions and participating in discussions.

2. An errata page. As I correct bugs, I will be posting information about these corrections in this page. Please check this page if you suspect that you have found an error. If you an error you have found is not listed in the errata page, please use the error report form in the same page to let me know about it.

3. Chapters that didn't fit in the book. I was not able to fit everything I wanted in this book. Deadlines and other constraints result in decisions about what can go in and what can't. I plan to post those chapters that didn't make it in the book, online.

From time to time, I will be posting additional KiCad resources and updates in that page, so please check regularly. By subscribing to the Tech Explorations email list, your'll be sure to receive my regular book updates and news. The subscription form is in the book page.

An introduction: Why KiCad?

Since KiCad first appeared in the PCB CAD world in 1992, it has gone through 5 major versions, and evolved into a serious alternative to commercial products. Once thought clunky and barely usable, it is now a solid, reliable CAD application. While it is true that KiCad is still behind its commercial competitors in specific areas,[1]

I believe that the benefits we get from truly free ('free'' as 'free and open source[2]') software are worth the trade-off in polish and finish.

One of those benefits is KiCad's very active and growing community of users and contributors. KiCad has a dedicated developer team, supported by contributing organisations such as CERN, the Raspberry Pi Foundation, Arduino LLC, and Digi-Key Electronics. The community is also active in contributing funds to cover development costs. A fund-raising campaign covered the requested amount by 160%, ensuring 600 hours of development towards KiCad version 6. These alone, to a large extent, guarantee that KiCad's development will accelerate, and will continue to in the future.

Next to the core team, are the people that make the KiCad community. These people support the KiCad project in various ways: writing code, sharing libraries, helping others learn. Over the last five years, I have seen an explosion of interest in KiCad. As a consequence, the Internet is flooded with relevant resources: guides, tutorials, libraries, scripts. Manufacturers have also taken notice. Many of them now publish tutorials, explaining how to order your boards. Some have even made it possible to do so by uploading a single file from your KiCad project instead of having to generate multiple Gerber files, making you prone to make simple errors because of the multiple export options of this process.

Why do I use KiCad? I'm glad you asked. First, let's look at my background. I am an electrical engineer with a background in electronics and computer engineering. Above all, I am an educator and electronics hobbyist. The majority of my PCB projects eventually find themselves in my books and courses. My projects are very similar to those of other hobbyists, in terms of complexity and size. I make things for my Arduino and Raspberry Pi courses. It could be an Arduino clone, or shield, a Raspberry Pi HAR, or a stand-alone relay board, power supply or motor controller. Nothing I would brag about. As a hobbyist, KiCad proved to be the perfect tool to me. But I do plan to bigger and better boards.

This is why I decided not to use another excellent tool, Fritzing. In KiCad, I saw a lot of benefits, without any show-stopping problems. I will list and briefly discuss my top 10 KiCad benefits here.

1 An example is KiCad's library manager, traditionally the topic of complaints and the reason of a lot of headaches. With version 5, however, library management is far better than it used to be.
Another area when KiCad is still behind some of the alternatives is in collaborative tools and workflows for teams. With source control systems like Github, however, KiCad collaborative projects are possible, to an extend.

2 Learn about Free and Open Source Software: https://en.wikipedia.org/wiki/Free_and_open-source_software

Benefit 1: KiCad is open source. To me, this is very important, especially as I find myself spending more time creating new and more complicated boards. Open source, by definition, means that the code base of the application is available for anyone to download and compile on their computer. It is why technologies such as Linux, Apache, and Wordpress essentially run the Internet (all of them open source). While I am not extreme in my choices between open source and closed source software, whenever a no-brainer open source option does appear, like Ki-Cad, I take it.

Benefit 2: It is free! This is particularly important for hobbyists. CAD tools can be expensive. Without a revenue resulting from the hobby capable of supporting the licensing fees, it is hard to justify hundreds of dollars spent, especially when there are viable alternatives. Which brings me to Benefit 3…

Benefit 3: KiCad is unlimited. There are no 'standard', 'premium' and 'platinum' versions to choose from. It's just a download, and you get everything. While there are many free commercial PCB tools, there are always restrictions on things like how many layers and how big your board can be, what can you do with your board once you have it, who can manufacture your board, and much more. I'll say again: KiCad is unlimited! This is so important, that I choose to pay a yearly donation to CERN & Society Foundation that is higher than the cost of an Autodesk Eagle license to do my part in helping to maintain this.

Benefit 4: KiCad has awesome features. Features such as interactive routing, length matching, and differential routing professional-grade. While you may not need to use some of them right away, you will use them eventually. Features that are not included 'in the box' can be added through third-party add-ons, one of the benefits of open source. The autorouter is one example. The ability to automate workflows and extend capabilities through Python scripts is another.

Benefit 5: KiCad is continually improved. Especially since CERN because involved in their current capacity, I have seen a very aggressive and successfully imple-mented roadmap. At the time of writing this, KiCad 5 is about one month old (it was released in early August, 2018). The funding for KiCad 6 is complete, and the road map living document published. When I look at this roadmap, I get very excited: an improved and modernised user interface, improvement in the schematic editor and the electrical rules checker (hopefully, with better error messages), better net highlighting, and much more, are in the works right now.

Benefit 6: KiCad's clear separation of schematic and layout is a bonus to learning and using it. Users of other PCB applications often find this confusing, but I really believe that it is an advantage. Schematic design and layout design are truly two different things. You can use them independently. I often create schematic dia-grams for my courses that I have no intention in converting into PCBs. I also often create multiple versions of a board, using the same schematic. This separation of roles makes both scenarios easy.

Benefit 7: I can make my boards anywhere: I can upload my project to any on-line fabricator that accepts the industry-standard Gerber files; I can upload it to an increasing number of fabricators that accept the native KiCad layout file; and, of course, I can make them at home using an etching kit (I do not cover this option in this book).

Benefit 8: KiCad works anywhere. Whether you are a Mac, Windows or Linux person, you can use KiCad. I actually use it on all three platforms.

Benefit 9: KiCad is very configurable. You can assign your favourite keyboard hotkeys and mapping, and together with the mouse customisations, you can fully adapt it to you preferences.

Benefit 10: If you are interested in creating analog circuits, you will be happy to know that KiCad now has integration with SPICE. You can draw the schematic in Eeschema, and then simulate it in SPICE, without leaving KiCad. When I need to simulate an analog circuit, I normally use iCircuit, an excellent desktop app. But I do plan to start using KiCad and Spice for this kind of work.

These are the ten most important reasons for which I have chosen KiCad as my tool of choice for designing PCBs. These reasons might not be right for you, but I hope that you will consider reading this book first before you make your own decision.

In this book, I have packed almost everything I have learnt as a KiCad user over the last four years. I have organised it in a way that will make learning KiCad quick. The objective of this book is to make you productive by the time you complete the first project, in part 4. If you come from another PCB CAD tool and are already experienced in designing PCBs, I only ask that you have an open mind. KiCad is most certainly very different to the tool that you are used to. It looks different, and it behaves differently. It will be easier to learn it if you consciously put aside your expectations, and look at KiCad like a beginner would. As per the Borg in Star Trek, 'resistance is futile', and in learning, like in so many other aspects of life, you are better off if you just go with the flow.

Let's begin!

Part 1: A quick introduction to PCB design

Chapter 1 • What is a PCB?

As a child, I remember that my interest in electronics grew from admiration of what these smart engineers had come up with, to curiosity about how these things worked. This curiosity led me to use an old screwdriver that my dad had left in a drawer (probably after fixing the hinges on a door), to open anything electronic with a screw large enough for the screwdriver to fit in.

A record player, a VCR, a radio. All became my victims. I am still amazed that a charged capacitor didn't electrocute me. At least, I had the good sense to unplug the appliances from the mains. Inside those devices, I found all sorts of amazing things. Resistors, transformers, integrated circus, displays.

All of those things were fitted on green boards, like the one in Figure 1.1. This is an example of a printed circuit board, or PCB, for short.

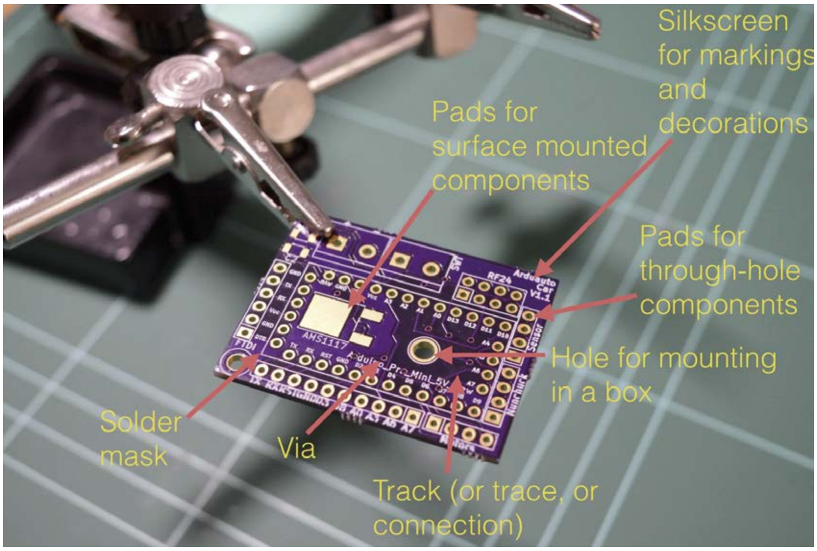

Silkscreen for markings and decorations
Pads for surface mounted components
Pads for through-hole components
Hole for mounting in a box
Solder mask
Via
Track (or trace, or connection)

Figure 1.1: The top side of a printed circuit board.

Let's have a look at the components of a PCB, what a PCB looks like and the terminology that we use. The example PCB is one I made for one of my courses (Figure 1.1).

The top side of the PCB is the side where we place the components. We can place components on the bottom side too; however this is unusual.

In general, there are two kinds of components: through hole or surface mounted components. Through hole components, are attached on the PCB via inserting the leads or the pins through small holes, and using hot solder to hold them in place. In the example pictured in Figure 1.1, you can see several holes into which you can insert the through-hole component pins. The holes extend from the top side to the bottom side of the PCB, and plated with a conductive material, such as tin, or in this case, gold. We use solder to attach and secure a component through its lead onto the pad that is surrounding the hole (Figure 1.1).

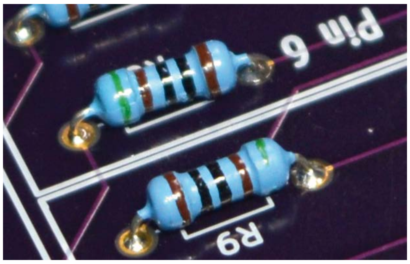


Figure 1.1: A through-hole component attached to a PCB.

If you wish to attach a surface-mounted component, then instead of holes, you attach the component onto the surface of the PCB using tin-plated pads. You will use just enough solder to create a solid connection between the flat connector of the component and the flat pad on the PCB (Figure 1.2).

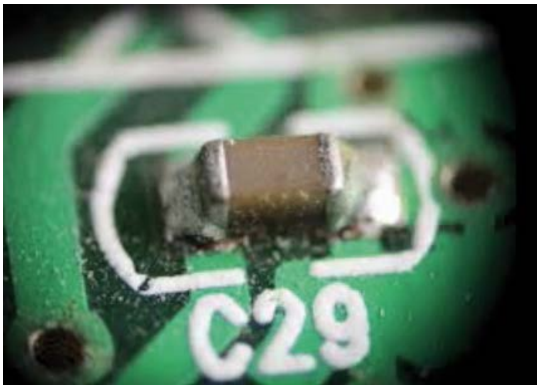


Figure 1.2: A surface-mounted component attached to a PCB.

Next, is the silkscreen. We use the silkscreen for adding text and graphics. The text can provide useful information about the board and its components. The graphics can include logos, other decorations, and useful markings.

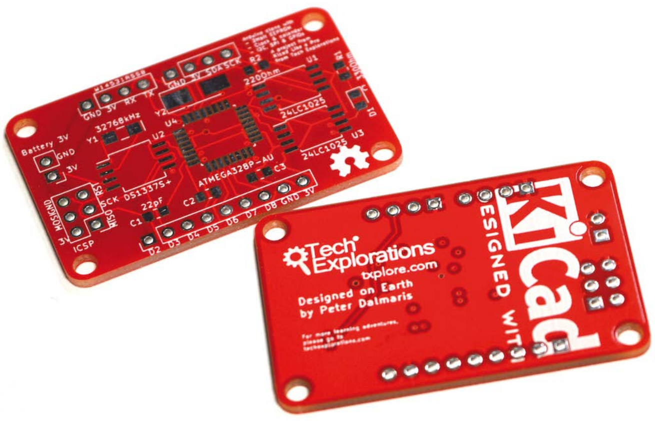


Figure 1.3: The white letters and lines is the silkscreen print on this PCB.

In Figure 1.3, you can see here that I've used white boxes to indicate the location of various components. I've used text to indicate the names of the various pins, and I've got version numbers up there. It's a good habit to have a name for the PCB and things of that sort. Silkscreen goes on the top or the bottom of the PCB.

Sometimes, you may want to secure your PCB onto a surface. To do that, you can add a mounting hole. Mounting holes are similar to the other holes in this board, except that they don't need to be tinned. You can use a screw and a nut and bolt to the other side so that the PCB is secured inside, for example, a box.

Next are the tracks. In this example (Figure 1.4), they look red because of the color of the masking chemical used by the manufacturer.

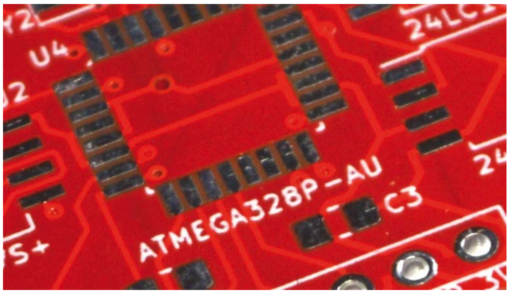


Figure 1.4: The bright red lines connecting the holes are tracks.

Tracks are made of copper, and they electrically connect pins or different parts of the board. You can control the thickness of a track in your design. Tracks can also be referred to as 'traces'.

Notice the small holes that have no pad around them? These are called 'vias.' A via looks like a hole but is not meant to be used to mount a component on it. A via is used to allow a track to continue its route to a different layer. If you're using PCBs that have more than two layers, then you can use vias to connect a track from any one of the layers to any of the other layers. Vias are very useful for routing your tracks around the PCB.

The purple substance that you see on the PCB is the solder mask. It does a couple of things. It prevents the copper on the PCB from being oxidised over time. The oxidisation of the copper tracks negatively affects their conductivity. The solder mask prevents oxidisation.

Another thing that the solder mask does is to make it easier to solder by hand. Because pads can be very close to each other, soldering would be very difficult without the solder mask. The solder mask prevents hot solder from creating bridges between pads because it prevents it from sticking on the board. (Figure 1.5). The solder mask prevents bridges because solder cannot bond with it.

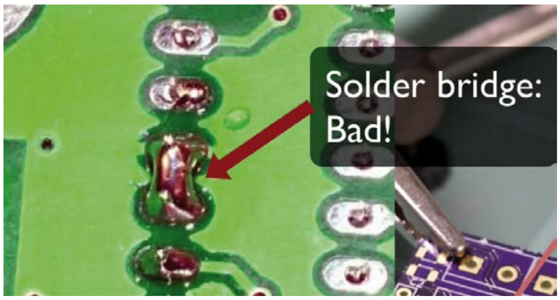


Figure 1.5: A solder bridge like this one is a defect that solder mask helps in preventing.

Often, the tip of the solder, the soldering iron is almost as big or sometimes as bigger than the width of the pads, so creating bridges in those circumstances is very easy and solder mask helps in preventing that from happening.

In Figure 1.6 you can see an example of the standard 1.6mm thick PCB.

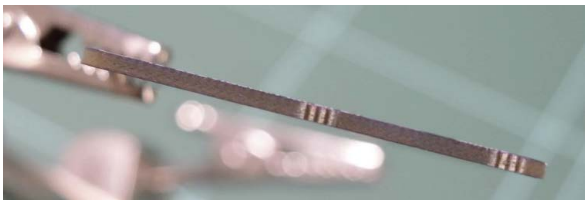


Figure 1.6: This PCB has a thickness of 1.6mm, and is made of fiberglass.

Typically, PCBs, are made of fiberglass. The typical thickness of the PCB is 1.6 millimeters. In this close-up view of a PCB picture (Figure 1.7), you can see the holes for the through-hole components. The holes for the through hole components are the larger ones along the edge of the PCB. Notice that they are tined in the inside, electrically connecting the front and back.

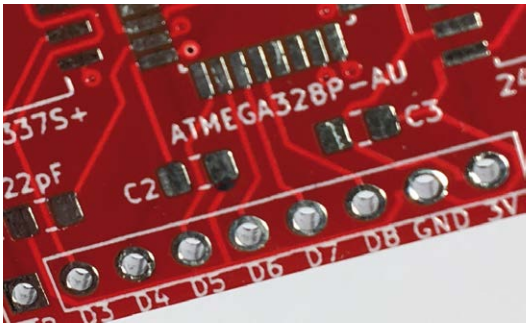


Figure 1.7: A closeup view of the top-layer.

In Figure 1.7 you can see several vias (the small holes) and tracks, the purple solder mask, and the solder mask between the pads. In this close up, you can also see the detail of the silkscreens. The white ink is what you use in the silkscreen to create the text and graphics. Figure 1.8 is interesting because it shows you a way to connect grounds and VCC pads to large areas of copper which is called the copper fill.

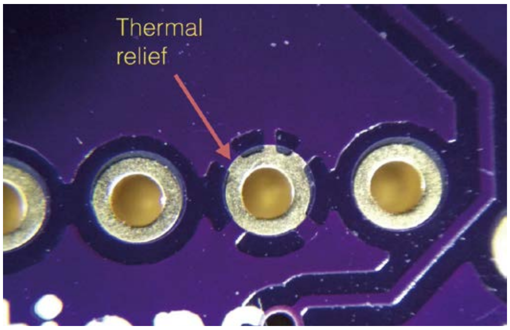


Figure 1.8: Thermal relief connects a pad to a copper region.

In Figure 1.8, the arrow points to a short segment of copper that connects the pad to a large area of copper around it. We refer to this short segment of copper as a 'thermal relief.' Figure 1.9 gives a different perspective that allows to appreciate the thickness of the tracks.

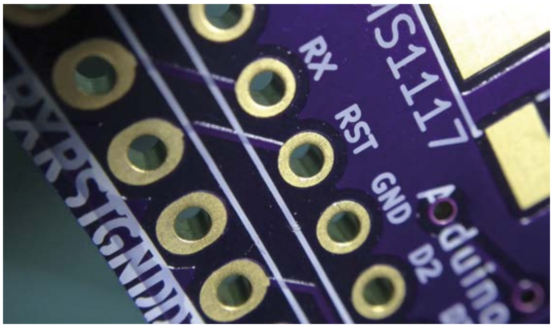


Figure 1.9: The plating of the holes covers the inside of the hole, and connects that front end with the back end.

Notice the short track that connects the two reset holes (RST)? The light that reflects off the side of the track gives you an idea of the thickness of that copper which is covered by the purple solder mask.

In this picture, you can also see a very thin layer of gold that covers the hole and the pad and how that also fills the inside of the hole. This is how you electrically have both sides of the hole connected.

Instead of gold plating, you can also use tin plating in order to reduce the manufacturing costs.

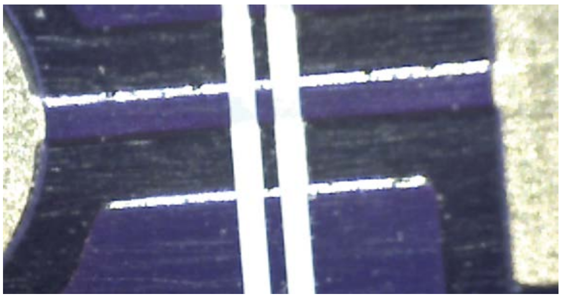


Figure 1.10: A details of this example board at 200 times magnification.

The image in Figure 1.10 is at 200 times magnification. You can see a track that connects two pads, and the light that reflects off one side of the track.

Chapter 2 •The PCB design process

To design a printed circuit board, you have to complete several steps, make decisions, and iterate until you are satisfied with the result. Because you are creating something that must be able to perform a specific operation (or multiple operations) well, your design work must be of high quality, safe, and manufacturable. There is no point designing a PCB that can't be manufactured.

Apart from the practical considerations of designing a PCB, there are also the aesthetic ones. You want your work to look good, not just to function well. Designing a PCB, apart from being an engineering discipline, is also a form of art.

In this book, you will learn about the technical elements of designing a PCB in KiCad, but I am sure that as you start creating your PCBs, your artistic side will emerge. Over time, your PCB will start to look uniquely yours.

PCB design is concerned with the process of creating the plans for a printed circuit board. It is different from PCB manufacturing. In PCB design, you learn about the tools, process, and guidelines useful for creating such plans. In PCB manufacturing, on the other hand, you are concerned about the process of converting the plans of a PCB into the actual PCB. As a designer of printed circuit boards, it is useful to know a few things about PCB manufacturing, though you surely do not need to be an expert. You need to know about the capabilities of a PCB manufacturing facility so that you can ensure that your design does not exceed those capabilities and that your PCBs are manufacturable.

As a designer, you need to have an understanding of the design process, and the design tools. To want to design PCB, I assume that you already have a working knowledge of electronics. Designing a PCB, like much else in engineering, is a procedural and iterative process that contains a significant element of personal choice. As you build up your experience and skills, you will develop your unique designing style and process.

While a personalised design process contains unique elements, it still follows the generic process you will learn about in this book. I distilled this process by drawing from my own experience and learning from other people's best practices. I also tried to simplify this process and make it suitable for people new to PCB design.

Since, in this book, you will be using KiCad, I have created a schematic that shows the PCB design process using KiCad terms and tools. You can see it in Figure 2.1.

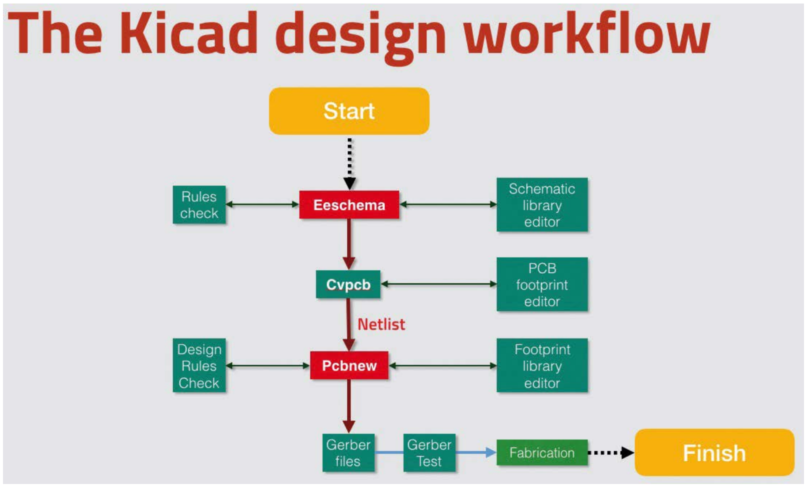


Figure 2.1: The KiCad design workflow.

From a very high-level perspective, the PCB design process only has two major steps:

1. it starts with a schematic diagram;
2. it ends with the layout.

The goal of the design is to create the layout. The layout is a file that contains information about your board, which the manufacturer can use to create the board. The layout must contain information about the size and shape of the board; its construction (such as how many layers it must have); the location of the components on the board, the location of various board elements, like pads, holes, traces and cutouts; the features of these elements (such as the sizes of holes and traces); and much more (which you will learn in detail later in this book).

When we bring this generic process to KiCad, we can map it to the elements that you see in Figure 2.1.

The process begins with Eeschema. In Eeschema you create the electrical schematic that describes the circuit that eventually will be manufactured into the PCB. You draw the schematic by selecting symbols from the library and adding them to the schematic sheet. If a component that you need doesn't exist in the library, you can create it using the schematic library editor.

Running regular electrical rules checks helps to detect defects early. Eeschema has a built-in checker utility for this purpose, as has Pcbnew, the layout editor. These utilities help to produce PCBs that have a low risk to contain design or electrical defects.

There are two things that you need to do before we go to Pcbnew:

1. associate the components in Eeschema with footprints;
2. create the netlist file, which contains information that Pcbnew needs to set up the layout sheet. The netlist file is what connects Eeschema and Pcbnew.

A symbol is a graphical representation of a real component in the schematic; it does not have a physical counterpart. However, in the layout editor (Pcbnew), everything is real in the sense that it has a real-world counterpart. Therefore, you, as the designer, must associate the symbol with a footprint. The footprint is a real thing; it represents resistors, switches, and pads on the PCB. This allows us to match schematic components with footprint modules.[3]

Once you have created the associations between symbols and modules, you will then export the Netlist file from Eeschema. Then, you'll import the netlist into Pcbnew, and all the footprints that you associated in Eeschema will appear in a new sheet so you can start working on the layout.

You use Pcbnew to position the footprints on the sheet and wire them. Wiring can be very time-consuming, especially for large boards. It is possible to use tools that do the wiring automatically, a capability that can significantly reduce the layout time.

Once you have your PCB laid out and have its traces completed, you can go ahead and do the design rules check. This check looks for defects in the board, such as a trace that is too close to a pad or two footprints overlapping.

When you are finished working on the layout, you can continue with the last step which involves exporting the layout information in a format that is compatible with your board manufacturer's requirement. The industry standard for this is a format called 'Gerber.' Gerber files contain several related files, with one Gerber file per layer on your PCB, and contain instructions that the fabrication house needs to manufacture your PCB.

Let's move on to the next chapter where we'll talk about fabrication.

3 In KiCad, a schematic contains symbols, and a layout contains footprints.

Chapter 3 • Fabrication

Imagine that you have finished with laying out your board in KiCad and you're ready to make it. What are your options? One option is to make your PCBs at home. There's a guide available on the Fritzing website, at:

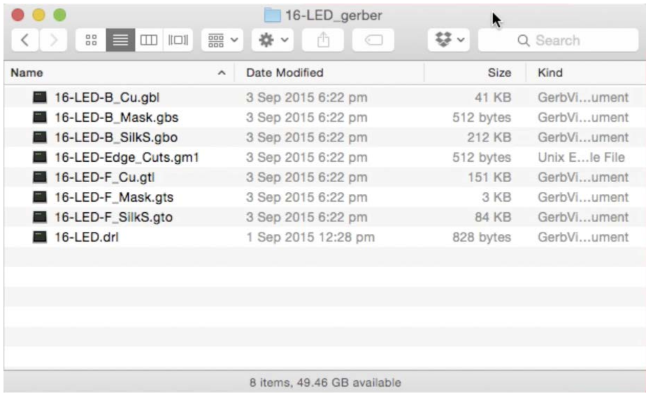
http://fritzing.org/learning/tutorials/pcb-production-tutorials/diy-pcb-etching/

The process described in the Fritzing guide is called etching. It involves the use of various chemicals, in chemical baths. Some of these chemicals are toxic. You have to have special safety equipment, and keep your children and pets away. The process emits smelly and potentially dangerous fumes. Once you have your board etched, you still need to use a drill to make holes and vias, and then figure out a way to connect your top and bottom layers.

If this sounds like not your kind of thing (I'm with you!), then you can opt for a professional PCB manufacturer service. PCBWay, OSH Park and other manufacturers like ExpressPCB and Seeed Studio are very good at what they offer.

You can get a professionally made PCB around $15 for several copies, and without danger to yourself as well. I've used OSHPark (great for beginners thanks to its straightforward user interface) and PCBWay (great for more advanced projects that need a large array of manufacturing options) extensively. I'm always happy with the result. Using an online manufacturer does take a little bit of planning because once you order your PCBs it can take up to several weeks for them to be delivered. If you're in a hurry there are options to expedite the process if you are willing to pay a premium.

The typical small standard two-layer order costs around $10 for a two square inch board; you get three copies of that. This price works out to around $5 per square inch. The pricing is consistent in the industry, where the main cost factor is the size of the PCB. There is a strong incentive to make your PCBs as small as possible. Be aware of this when you design your layout.

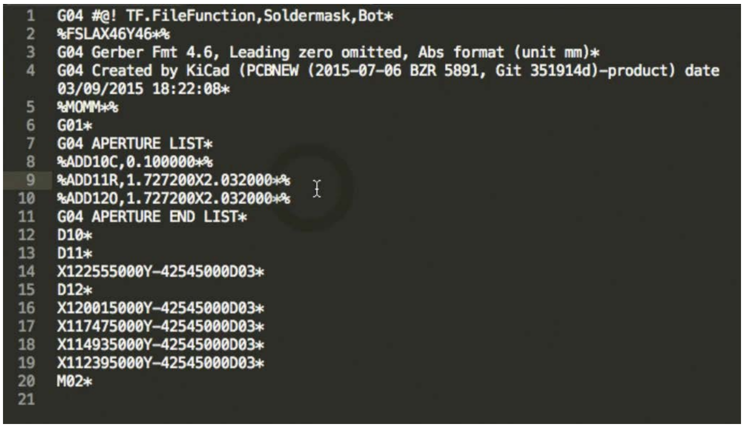


Name	Date Modified	Size	Kind
16-LED-B_Cu.gbl	3 Sep 2015 6:22 pm	41 KB	GerbVi...ument
16-LED-B_Mask.gbs	3 Sep 2015 6:22 pm	512 bytes	GerbVi...ument
16-LED-B_SilkS.gbo	3 Sep 2015 6:22 pm	212 KB	GerbVi...ument
16-LED-Edge_Cuts.gm1	3 Sep 2015 6:22 pm	512 bytes	Unix E...le File
16-LED-F_Cu.gtl	3 Sep 2015 6:22 pm	151 KB	GerbVi...ument
16-LED-F_Mask.gts	3 Sep 2015 6:22 pm	3 KB	GerbVi...ument
16-LED-F_SilkS.gto	3 Sep 2015 6:22 pm	84 KB	GerbVi...ument
16-LED.drl	1 Sep 2015 12:28 pm	828 bytes	GerbVi...ument

8 items, 49.46 GB available

Figure 3.1: An example of the Gerber files that the manufacturer will need in order to make your PCB.

Now, let's turn our attention to the files that you need to upload for these services — and the files are Gerber files. Each layer on your PCB has its own Gerber file which is simply a text file. Figure 3.2 shows the contents of an example Gerber file.



```
G04 #@! TF.FileFunction,Soldermask,Bot*
%FSLAX46Y46*%
G04 Gerber Fmt 4.6, Leading zero omitted, Abs format (unit mm)*
G04 Created by KiCad (PCBNEW (2015-07-06 BZR 5891, Git 351914d)-product) date
03/09/2015 18:22:08*
%MOMM*%
G01*
G04 APERTURE LIST*
%ADD10C,0.100000*%
%ADD11R,1.727200X2.032000*%
%ADD12O,1.727200X2.032000*%
G04 APERTURE END LIST*
D10*
D11*
X122555000Y-42545000D03*
D12*
X120015000Y-42545000D03*
X117475000Y-42545000D03*
X114935000Y-42545000D03*
X112395000Y-42545000D03*
M02*

```

Figure 3.2: Gerber files contain text

You can see that this is just a text-based file that contains instructions. An advantage of this text format is that you can use a version control system like Git and keep your projects stored in repositories like Github.com.

The Gerber files system and standard has been designed by Ucamco. They make equipment and write software for PCB manufacturers — things like a PreCAM software, PCB CAM, laser photoplotters and direct imaging systems. If you're curious about how to read these Gerber files then you can look up the specification of the Gerber format specification on Ucamco's web site. Beware, it's a huge file.

Chapter 4 • Installation

You can install KiCad on Windows, Mac OS and several flavours of Linux using operating system-specific installers. Its sources code is also available, so you can download it and compile it yourself. You can find the version of the installer for your OS at the KiCad download page: http://KiCad-pcb.org/download.

I suggest you install the stable version of KiCad unless you feel compelled to use the cutting-edge releases which contain the latest features (and bugs). In this case, you can download the latest nightly build. You can find information about these builds in the KiCad's download page for your operating system (Windows and OS X). If you are using Linux, you can install KiCad from the command line using tools like apt-get (Debian, Ubuntu) and dnf (Fedora).

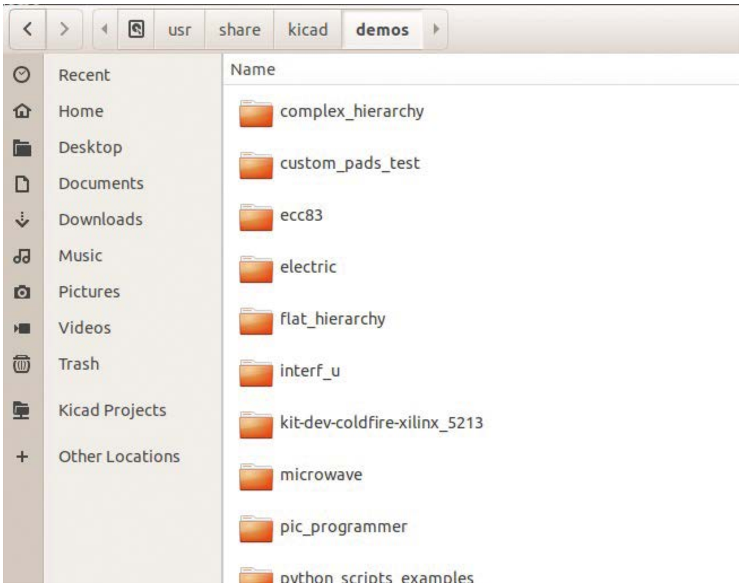
You can find detailed instructions on how to do the installation on the KiCad web site.

Please install your copy now before you continue with the next chapter. I also recommend that you install the demo projects because they provide multiple examples of design best practices. I have learned a lot about KiCad by browsing and studying these examples.

In Ubuntu, you can do this by running this command:

```
$ sudo apt install KiCad-demo
```

The demos are installed in /usr/share/KiCad/demos, from where you can copy them in your working folder (Figure 4.1).



complex_hierarchy
custom_pads_test
ecc83
electric
flat_hierarchy
interf_u
kit-dev-coldfire-xilinx_5213
microwave
pic_programmer
python_scripts_examples

Figure 4.1: The KiCad 'demos' directory in Ubuntu.

In Mac OS, the demos are packaged with the installed, but you have to copy them separately to the KiCad apps, help, and libraries files (Figure 4.2).

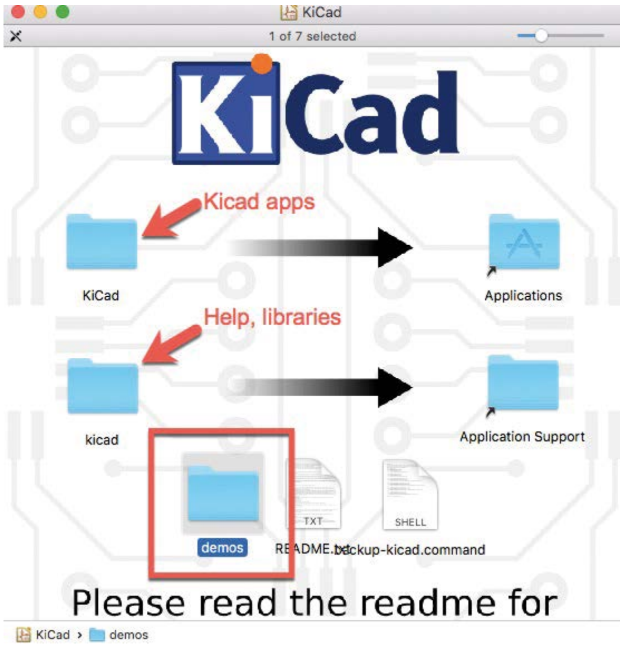


Figure 4.2: The Mac OS installer contains the demo projects in a seperate folder.

Copy the 'demos' folder in your Documents directory, and the other two folders as instructed ('KiCad' to Applications, and 'KiCad' to Application Support). You can see the contents of this folded in Figure 4.3. The original name of this folder is 'demos,' but I have renamed it to 'KiCad demos' to make it easier to find among the other contents of my Documents folder.

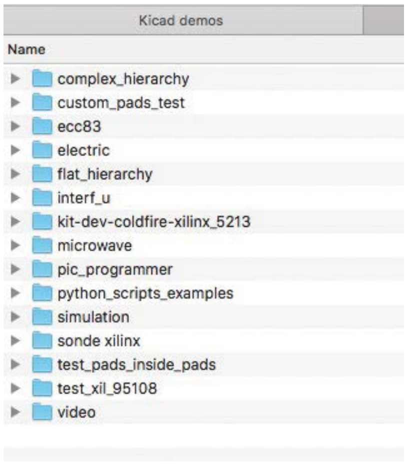
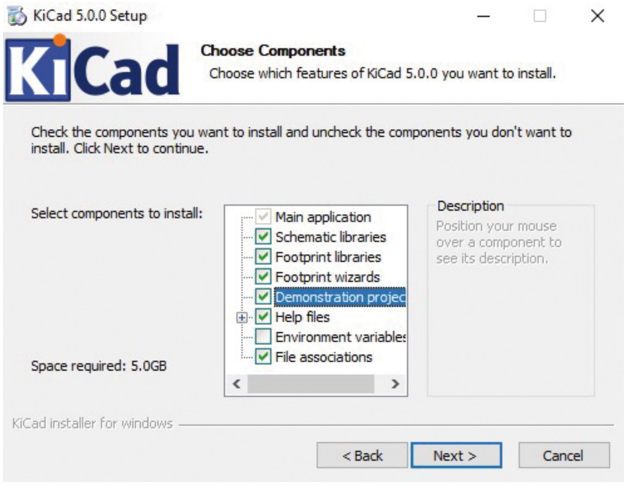


Figure 4.3: The demo folder in my ~/Documents/KiCad demos folder (renamed from 'demos').

In the Windows installer, the demos are available as an option (Figure 4.4).


KiCad 5.0.0 Setup

Choose Components

Choose which features of KiCad 5.0.0 you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- [✓] Main application
- [✓] Schematic libraries
- [✓] Footprint libraries
- [✓] Footprint wizards
- [✓] Demonstration projec
- [✓] Help files
- [ ] Environment variable:
- [✓] File associations

Description

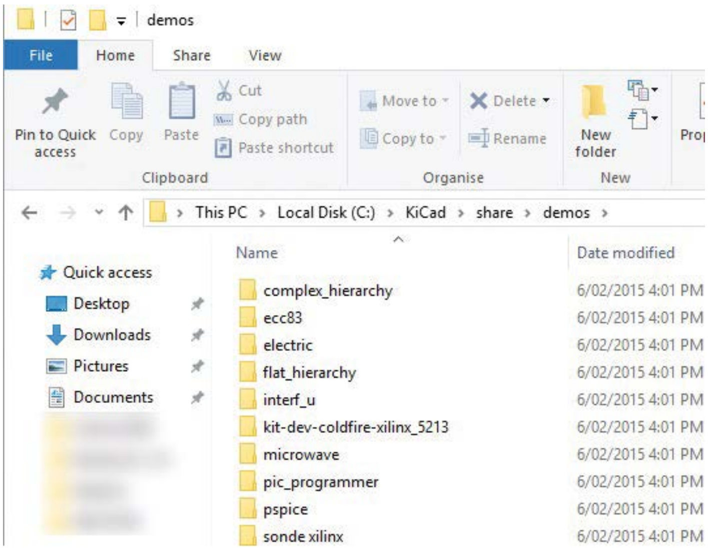
Position your mouse over a component to see its description.

Space required: 5.0GB

KiCad installer for windows

Figure 4.4: The Windows installer contains the demo projects as an optional component.

The default location of the KiCad demos in Windows is at C:\KiCad\share\demos.



This PC > Local Disk (C:) > KiCad > share > demos >

Name	Date modified
complex_hierarchy	6/02/2015 4:01 PM
ecc83	6/02/2015 4:01 PM
electric	6/02/2015 4:01 PM
flat_hierarchy	6/02/2015 4:01 PM
interf_u	6/02/2015 4:01 PM
kit-dev-coldfire-xilinx_5213	6/02/2015 4:01 PM
microwave	6/02/2015 4:01 PM
pic_programmer	6/02/2015 4:01 PM
pspice	6/02/2015 4:01 PM
sonde xilinx	6/02/2015 4:01 PM

Figure 4.5: The default demo directory in Windows.

With KiCad and its demos installed, you can continue with the next chapter, where you will take a look at one of the demo projects.

Chapter 5 • Examples of KiCad projects

Now that you have installed your instance of KiCad let's start your familiarisation with it by looking at one of the examples that come with it. Browse to the KiCad demos folder, and access the one titled 'pic_programmer' (Figure 5.1).

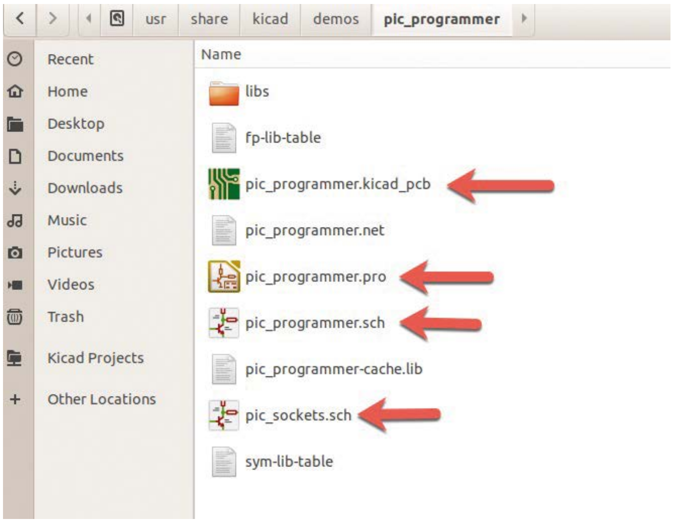
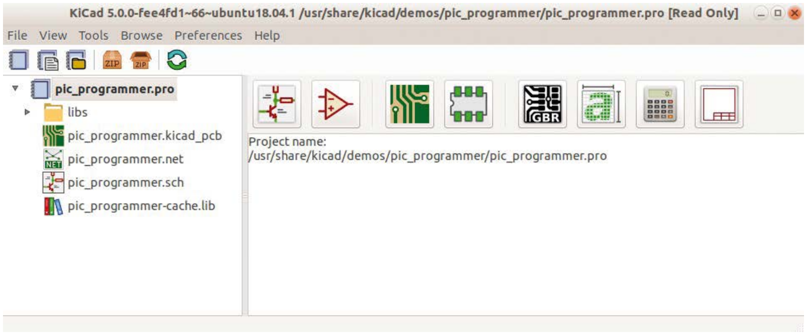


Figure 5.1: The contents of the 'pic_programmer' demo project folder.

The demo project folder contains several files that make up the project. The ones to focus on for now have the extensions 'pro,' 'KiCad_pcb' and 'sch.' The file with the 'pro' extension contains project information. The 'KiCad_pcd' file contains layout information. The files with the 'sch' extension contain schematic information. There are two 'sch' files because this project includes two schematics.

Double-click on the 'pro' (project) file. The main KiCad window will appear. This window is the launch pad for the other KiCad apps, like Eeschema (the schematic editor) and Pcbnew (the layout editor). You can see the main KiCad window in Figure 5.2.



Project name:
/usr/share/kicad/demos/pic_programmer/pic_programmer.pro

Figure 5.2: The main KiCad window.

The main KiCad window shows the project files in the left pane, the various app buttons in the top right pane, and various status messages in the bottom right pane. Let's explore the schematic of this demo project. In the top right pane, click on the first button from the left. This button will start the Eeschema app, the schematic layout editor. You should see the editor as in the example in Figure 5.3.

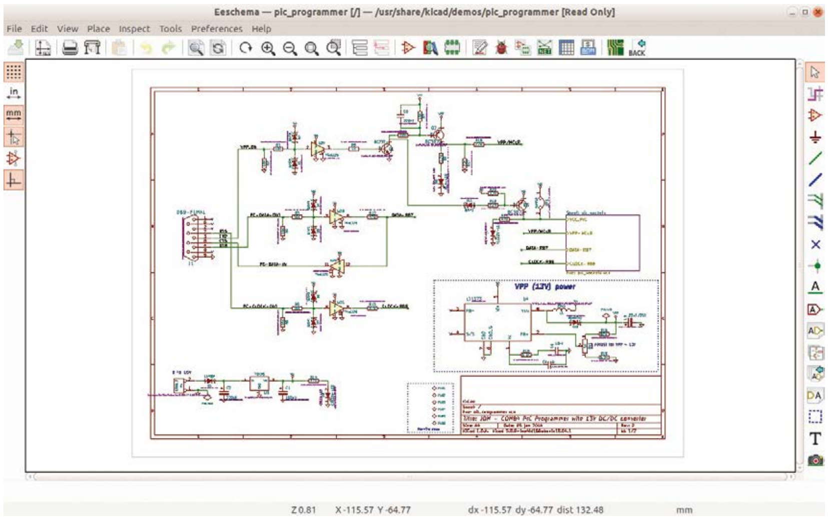

Eeschema — plc_programmer [/] — /usr/share/kicad/demos/plc_programmer [Read Only]
File Edit View Place Inspect Tools Preferences Help
Z 0.81 X -115.57 Y -64.77 dx -115.57 dy -64.77 dist 132.48 mm
VPP (13V) power

Figure 5.3: The schematic editor.

There are a few things going on here. At first, this window might seem overwhelming. Don't worry about the various buttons and menus, just concentrate on the schematic itself. Look at the various symbols, like those for the diodes, the transistors, and the operational amplifiers. There are symbols for resistors, and connectors, with green lines connecting their pins. Notice how text labels give names to the symbols, but also to the wirings between pins. Notice how even the mounting holes, at the bottom right side of the schematic, have names. Even though these mounting holes are not electrically active, they are depicted in the schematic. The values of the capacitors and resistors are noted, and any pins that are not connected to other pins are marked with an 'x'.

In the right side of the schematic, there is a rectangular symbol with the title 'Sheet: plc_sockets' (Figure 5.4).

Double click on it. What happened?

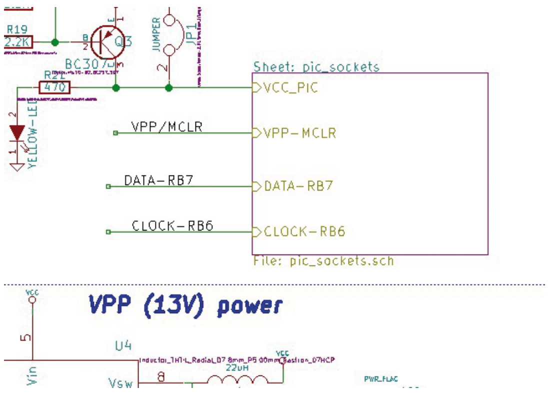
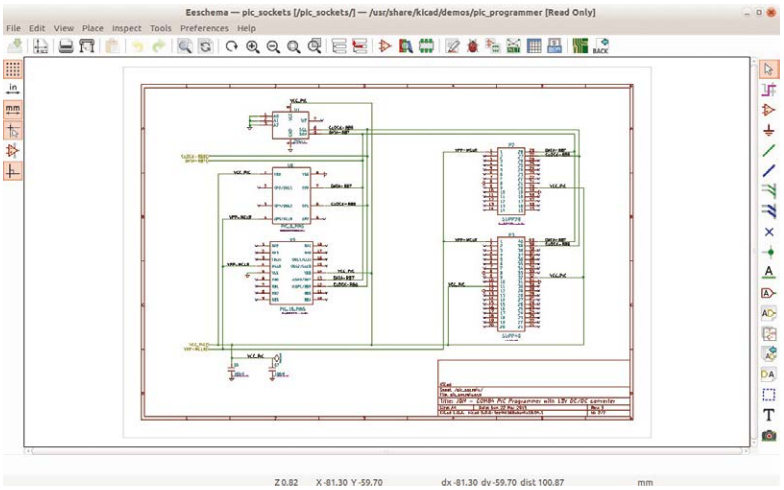


Figure 5.4: A link to another sheet.

This symbol is a link to another sheet, which contains additional symbols that are part of the same schematic. It looks like the example in Figure 5.5.


Eeschema — pic_sockets [/pic_sockets/] — /usr/share/kicad/demos/pic_programmer [Read Only]

File Edit View Place Inspect Tools Preferences Help
Z 0.82 X -81.30 Y -59.70 dx -81.30 dy -59.70 dist 100.87 mm

Figure 5.5: KiCad's schematics can span over multiple sheets.

KiCad's schematics can span over multiple sheets. If your schematic is too large to comfortably fit in one sheet, just add more (you will learn how to do this in this book).

I encourage you to spend a bit of time to study this schematic. You can learn a lot about how to draw good schematic diagrams by studying… good schematic diagrams, just like you can learn programming by studying good open source code.

Go back to the main KiCad window. Click on the third button from the left, the one that looks like a PCB. This will launch Pcbnew, the layout editor. The window that appears will look like the example in Figure 5.6.

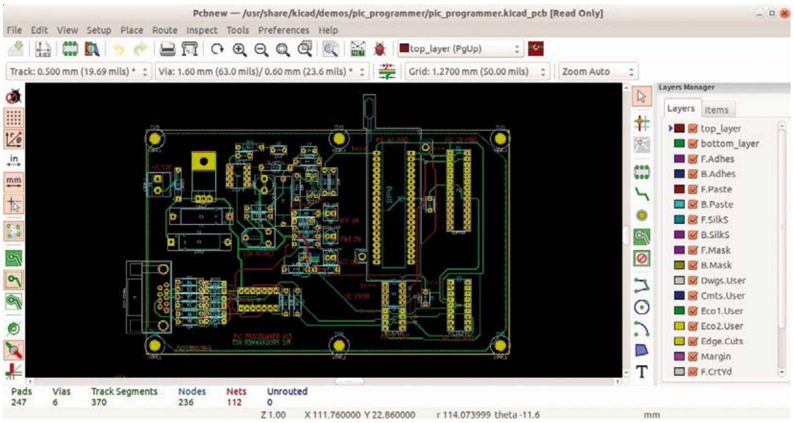


Figure 5.6: Pcbnew, the layout editor.

Again, don't worry about the various buttons and menus, just concentrate on the layout inside the sheet. Use the scroll wheel of your mouse to zoom in and out, and the Alt+right mouse button to pan. Zoom in and look at some of the layout details, such as the pads, how they are connected to traces, the names that appear on the pads and traces, and the colours of the front copper and back copper layer traces.

Also, compare how a footprint in the layout compares to the symbol in the schematic. You can see a side-by-side comparison in Figure 5.7.

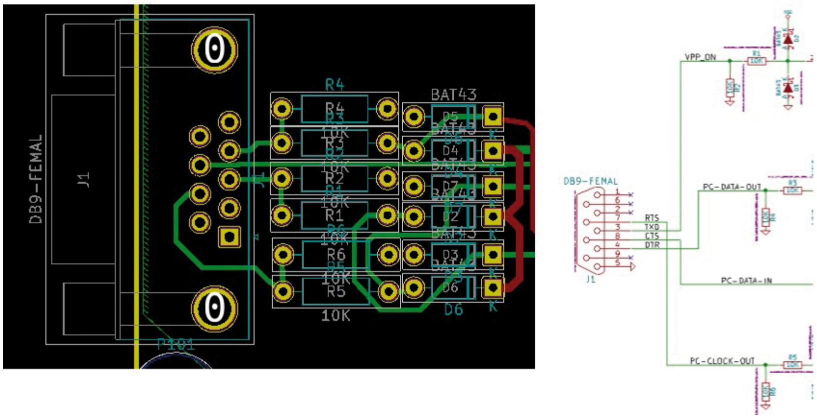


Figure 5.7: A side-by-side comparison of a footprint (left) and its schematic symbol (right).

Associated symbols and footprints have the same designator (J1, in this example), and the same number of pins. The layout shows the traces that correspond to the wires in the schematic.

Everything you see here is configurable: the width of the traces, which layer they belong to, the shape, size, and configuration of the pads. You will learn all of this in this book. In the layout, zoom in the J1 connector to see one of its details: the name of the trace that connects pad 7 of J1 to pad 1 of R5. Traces, like everything else in KiCad, have names. The names of everything that you see in Pcbnew are defined (manually or automatically) in Eeschema.

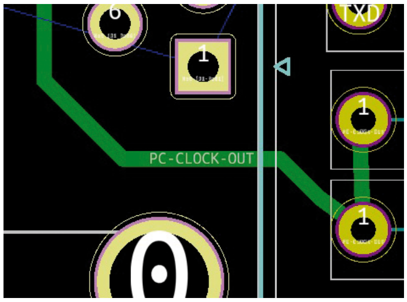


Figure 5.8: Traces have names.

Try one more thing: In Pcbnew, click on the View menu, and then choose the 3D Viewer. The 3D viewer will show you a three-dimensional rendering of the PCB, with remarkable detail. You can zoom and turn the board around to see it from any angle you want (Figure 5.9). Many of the components are populated, like the LED, resistors and some of the integrated circuits. For the rest, you can still see their pads and outlines on the board.

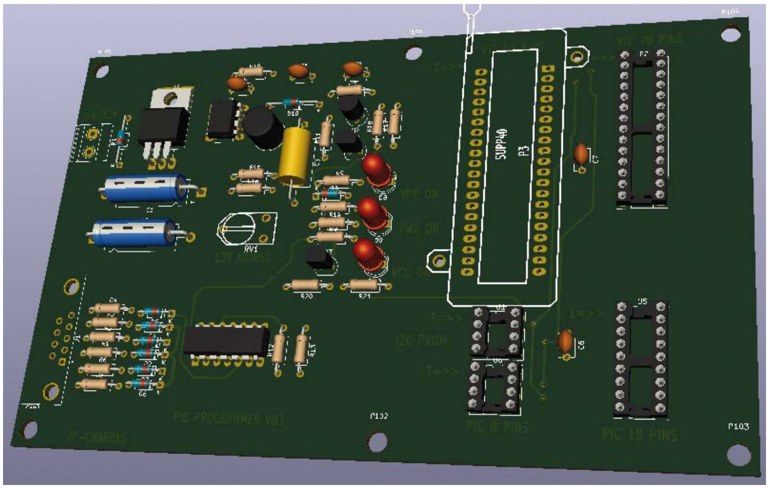


Figure 5.9: The 3D viewer will give you a realistic rendering of your board that you can examine in 3D.

As with the schematic editor, I encourage you to spend a bit of time to study the layout of this demo project. Later in this book, you will learn about the most important layout guidelines that will help you to design well-functioning and elegant PCBs.

Apart from the demo projects that KiCad ships with, you should also have a look at some of the very impressive showcased projects of boards designed using KiCad. For example, the

CSEduino is a 2-layer PCB that contains an Atmega328P microcontroller and implements a simple Arduino clone. You will be able to easily create a board like this by the time you finish this book.

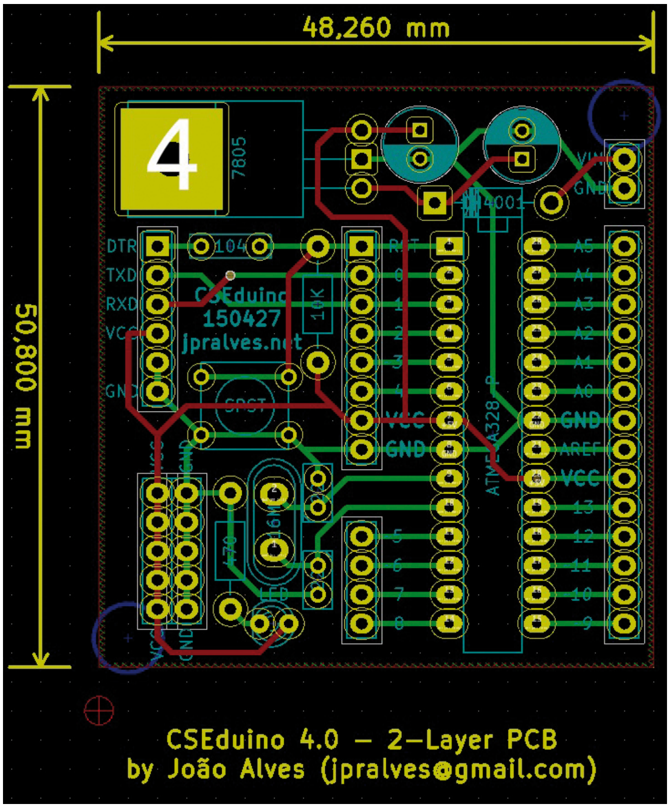

48,260 mm
50,800 mm
CSEduino 4.0 – 2-Layer PCB
by João Alves (jpralves@gmail.com)
CSEduino
150427
jpralves.net
7805
ATMEGA328-P

Figure 5.10: Featured board 'Made with KiCad': CSEduino.

Another featured board is Anavi Light, a HAT board for the Raspberry Pi. This is also a 2-layer board that allows you to control a 12V LED strip and get readings from sensors.

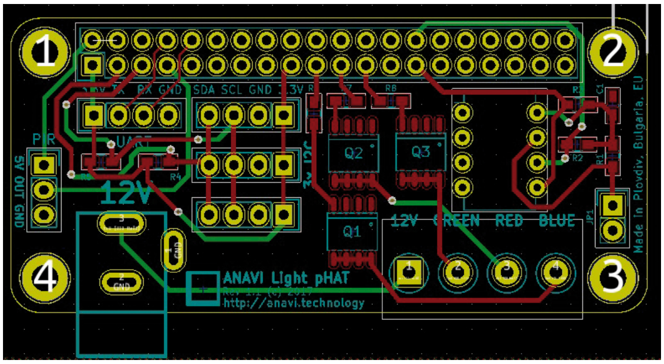


Figure 5.11: Featured board 'Made with KiCad': Anavi Light.

Finally, a truly impressive board made with KiCad is Crazyflie (Figure 5.12). Crazyflie is a dense 4-layer PCB with a rather elaborate shape. The board implements a the flight controller of a tiny drone. The shape is specifically designed to implement the drone's body and arms. You will also learn how to create PCBs with complicated shapes in this book.

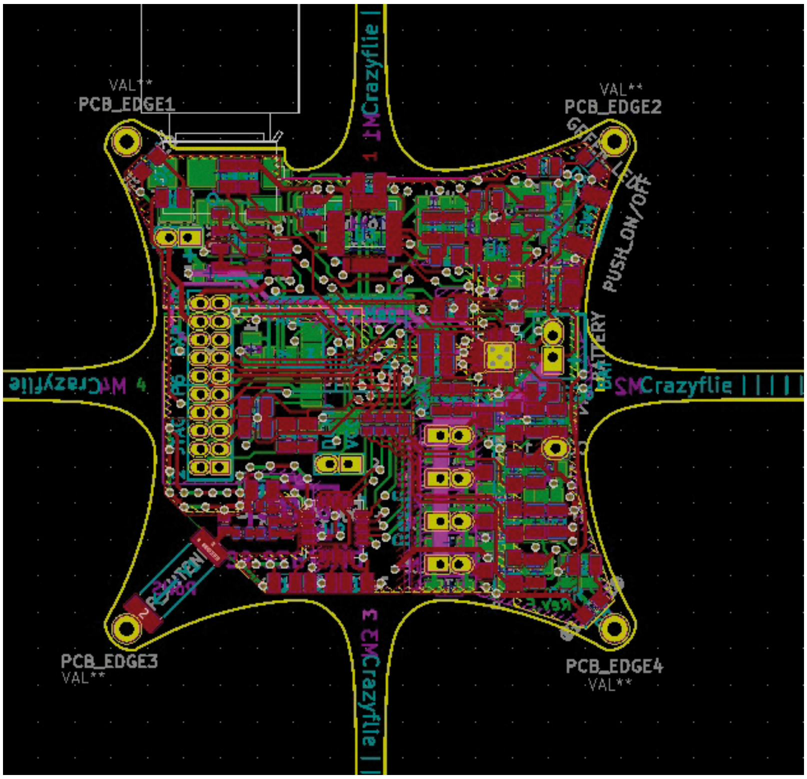


Figure 5.12: Featured board 'Made with KiCad': Crazyflie.

With this chapter complete, you should have a better understanding of the kinds of projects that people use KiCad for. These are also the kinds of boards that you will be able to design by the time you complete this book. Let's get straight into the first project so that you can start discovering this amazing tool by doing.

KiCad Like a Pro

Dr. Peter Dalmaris is an educator, an electrical engineer and Maker. Creator of online video courses on DIY electronics and author of several technical books. As a Chief Tech Explorer since 2013 at Tech Explorations, the company he founded in Sydney, Australia, Peter's mission is to explore technology and help educate the world.

Printed circuit boards (PCB) are, perhaps, the most undervalued component of modern electronics. The design of complex printed circuit boards was something that only skilled engineers could do. During the last 20 years, we have seen high-end engineering capabilities becoming available to virtually anyone that wants them. CAD tools and manufacturing facilities for PCBs are one mouse click away.

KiCad is one of those tools. Perhaps the world's most popular (and best) computer-aided design tool for making printed circuit boards, KiCad is open source, fully featured, well-funded and supported, well documented. It is the perfect tool for electronics engineers and hobbyists alike, used to create amazing PCBs. KiCad has reached maturity and is now a fully featured and stable choice for anyone that needs to design custom PCBs.

This book will teach you to use KiCad. Whether you are a hobbyist or an electronics engineer, this book will help you become productive quickly, and start designing your own boards. This book takes a practical approach to learning. It consists of four projects of incremental difficulty and recipes.

The projects will teach you basic and advanced features of KiCad. If you have absolutely no prior knowledge of PCB design, you will find that the introductory project will teach you the very basics. You can then continue with the rest of the projects. You will design a board for a breadboard power supply, a tiny Raspberry Pi HAT, and an Arduino clone with extended memory and clock integrated circuits.

The book includes a variety of recipes for frequently used activities. You can use this part as a quick reference at any time. The book is supported by the author via a page that provides access to additional resources.

ISBN 978-1-907920-74-5

9 781907 920745

Elektor International Media BV

www.elektor.com

elektor

LEARN DESIGN SHARE