

new and revised  
edition 2022

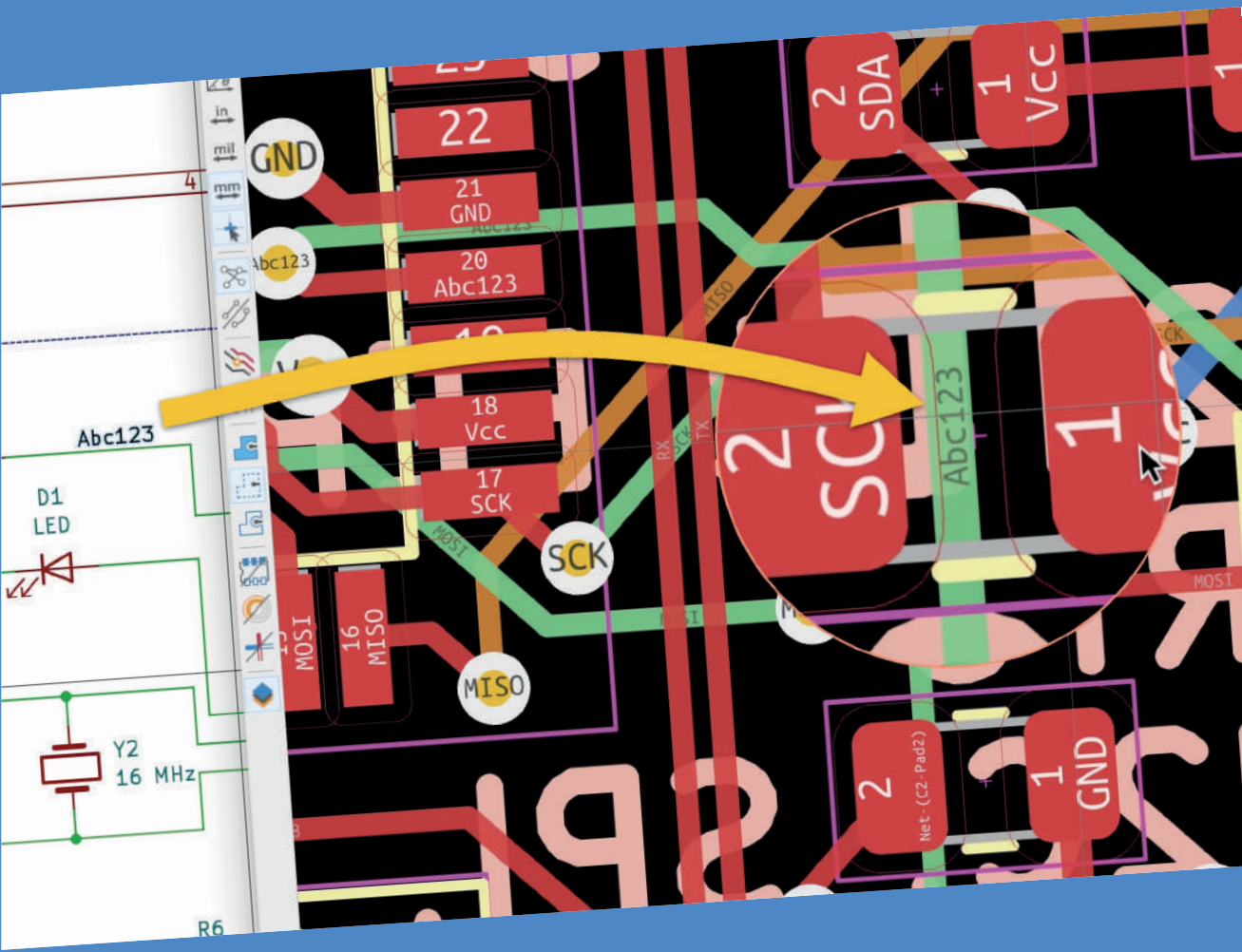
eLektor books

LIKE  
A PRO

# KiCad 6

## Projects, Tips, and Recipes

Mastering PCB design with  
real-world projects



Peter Dalmaris

eLektor  
design > share > learn



---

# KiCad 6 Like A Pro

## Projects, Tips and Recipes



Dr. Peter Dalmaris

● This is an Elektor Publication. Elektor is the media brand of Elektor International Media B.V.  
PO Box 11, NL-6114-ZG Susteren, The Netherlands  
Phone: +31 46 4389444

● 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 use 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 Licensing 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, and 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.

All the programs given in the book are Copyright of the Author and Elektor International Media. These programs may only be used for educational purposes. Written permission from the Author or Elektor must be obtained before any of these programs can be used for commercial purposes.

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

● This third edition of **KiCad Like a Pro** consists of two books:

- **KiCad 6 Like A Pro | Fundamentals and Projects**

Getting started with the world's best open-source PCB tool

548 pages

ISBN **978-3-89576-496-7** print

ISBN **978-3-89576-497-4** ebook

- **KiCad 6 Like A Pro | Projects, Tips, and Recipes**

Mastering PCB design with real-world projects

400 pages

ISBN: **978-3-89576-498-1** print

ISBN: **978-3-89576-499-8** ebook

● © Copyright 2022: Elektor International Media B.V.  
First published in Australia: 2021  
Prepress Production: D-Vision, Julian van den Berg  
Printed in the Netherlands by Ipskamp Printing, Enschede

## Contents

<b>Did you find an error?</b> .....	<b>10</b>
<b>About the author</b> .....	<b>10</b>
<b>About Tech Explorations.</b> .....	<b>11</b>
<b>From the back cover</b> .....	<b>13</b>
<b>How to read this book</b> .....	<b>14</b>
<b>Requirements</b> .....	<b>15</b>
<b>Foreword by Wayne Stambaugh.</b> .....	<b>16</b>
<b>The book web page</b> .....	<b>17</b>
<b>An introduction: Why KiCad?</b> .....	<b>18</b>
<b>Part 11 : Project - MCU datalogger.</b> .....	<b>22</b>
11.1. Project - Introduction .....	22
11.2. Create the new project and Git repository .....	25
11.3. Schematic design .....	28
11.3.1. Schema 1 - Setup .....	28
11.3.2. Schema 2 - Symbols .....	30
11.3.3. Schema 2 - Sheet two .....	31
11.3.4. Schema 3 - Arrange, Annotate .....	34
11.3.5. Edit component values .....	36
11.3.6. Schema 3 - Associate .....	37
11.3.7. Schema 4 - Wiring of sheet 1 .....	39
11.3.8. Schema 4 - Wiring of sheet 2 .....	42
11.3.9. Schema 5 - Nets .....	46
11.3.10. Schema 6 - Electrical Rules Check .....	48
11.3.11. Schema 7 - Comments .....	48
11.4. Create the 2-layer branch in Git .....	50
11.5. Layout design. ....	51
11.5.1. Layout 1 - Setup .....	51
11.5.2. Layout 2 - Outline and constraints .....	53
11.5.3. Layout 3 - Place components. ....	54
11.5.4. Layout 2 - Outline refinement .....	55

11.5.5. Layout 4 - Route . . . . .	57
11.5.6. Layout 4 - Copper fills . . . . .	61
11.5.7. Layout 4 - Routing improvements . . . . .	66
11.5.8. Layout 5 - Silkscreen . . . . .	68
11.5.9. Layout 4 - Routing violations and complete silkscreen . . . . .	72
11.5.10. Layout 6 - Design Rules Check. . . . .	77
11.5.11. Layout 7 - Manufacture. . . . .	79
11.6. 3D shapes . . . . .	83
11.7. Merge 2-layer branch to main . . . . .	86
11.8. Design 4 Layer PCB in new Git branch. . . . .	88
11.9. Four-layer PCB routing . . . . .	89
11.10. Four-layer PCB manufacturing . . . . .	92
11.11. Updating layout from changes to the schematic with Git . . . . .	94
11.12. Finding and correcting a design defect. . . . .	97
11.12.1. Fix the schematic. . . . .	99
11.12.2. Fix the 2 layer PCB layout . . . . .	105
11.12.3. Fix the 4 layer PCB layout . . . . .	108
<b>Part 12: Project - An ESP32 clone . . . . .</b>	<b>110</b>
12.1. Project - Introduction . . . . .	110
12.2. Schematic design . . . . .	115
12.2.1. Schema 1 - New KiCad project and Schematic Setup . . . . .	115
12.2.2. Schema 2 - Symbols . . . . .	118
12.2.3. Schema 3 - Annotate and set component values . . . . .	120
12.2.4. Schema 3 - Arrange . . . . .	123
12.2.5. Schema 3 - Associate . . . . .	125
12.2.6. Schema 4 - Wiring . . . . .	126
12.2.7. Schema 5 - Nets and Net Classes. . . . .	129
12.2.8. Schema 6 - Electrical Rules Check . . . . .	130
12.2.9. Schema 7 - Comments . . . . .	132
12.3. Layout design. . . . .	135
12.3.1. Layout 1 - Setup . . . . .	136

---

12.3.2. Layout 2 - Outline and constraints . . . . .	139
12.3.3. Layout 3 - Place components. . . . .	142
12.3.4. Layout 2 supplemental - refine outline . . . . .	148
12.3.5. Layout 4 - Route . . . . .	154
12.3.6. Layout 4 - Copper fills and keep out areas . . . . .	159
12.3.7. Layout 5 - Silkscreen . . . . .	164
12.3.8. Layout 4 - Routing improvements . . . . .	165
12.3.9. Layout 6 - Design Rules Check . . . . .	167
12.3.10. Layout 7 - Manufacture. . . . .	167
12.4. 3D shapes . . . . .	170
12.5. Finding and correcting a design defect. . . . .	172
12.5.1. Fix the schematic. . . . .	173
12.5.2. Fix the layout . . . . .	174
<b>Part 13: Recipes . . . . .</b>	<b>177</b>
13.1. Create a custom silkscreen or copper graphic. . . . .	177
13.2. Change a symbols and footprints in bulk . . . . .	187
13.2.1. Change a symbol in bulk. . . . .	187
13.2.2. Change a footprint in bulk. . . . .	191
13.3. Interactive delete . . . . .	195
13.4. Find and Replace (Eeschema). . . . .	197
13.5. Edit Text & Graphics Properties. . . . .	198
13.6. Edit Track & Via Properties (Pcbnew). . . . .	202
13.7. Text variables. . . . .	205
13.8. Board Setup - pre-defined sizes for tracks and vias. . . . .	210
13.9. Board Setup - Design rules violation severity . . . . .	213
13.10. Board Setup - Custom design rules . . . . .	217
13.11. Schematic Setup - Electrical Rules and violation severity . . . . .	220
13.12. Schematic Setup - Electrical Rules and Pin conflicts map . . . . .	223
13.13. Field name templates . . . . .	226
13.14. Bill of Materials. . . . .	231
13.14.1. Build-in BOM in Pcbnew . . . . .	231

13.14.2. Build-in BOM in Eeschema . . . . .	234
13.14.3. A plug-in for BOM. . . . .	236
13.15. Import components from Snapeda . . . . .	243
13.16. The FreeRouting autorouter . . . . .	252
13.16.1. Install and start FreeRouting on MacOS . . . . .	254
13.16.2. Install and start FreeRouting on Linux Kubuntu . . . . .	260
13.16.3. Install and start FreeRouting on Windows . . . . .	263
13.16.4. How to use the FreeRouting autorouter 2-layer example . . . . .	266
13.16.5. How to use the FreeRouting autorouter 4-layer example . . . . .	275
13.17. Pcbnew Inspection menu . . . . .	279
13.18. Single track and differential pair routing . . . . .	287
13.19. Track length tuning . . . . .	291
13.20. Differential pair skew tuning. . . . .	294
13.21. Interactive router modes . . . . .	296
13.22. The footprint wizard . . . . .	300
13.23. Pin and wire highlighter tool . . . . .	305
13.24. Pcbnew Origins . . . . .	307
13.25. KiCad project management with Git . . . . .	313
13.25.1. Install Git . . . . .	318
13.25.2. Git configuration . . . . .	318
13.25.3. Create a new KiCad project Git repository . . . . .	319
13.25.4. How to ignore files . . . . .	323
13.25.5. Basic Git commands: add, commit . . . . .	324
13.25.6. Basic Git commands: branch . . . . .	329
13.25.7. Basic Git commands: merge . . . . .	332
13.26. Sharing your KiCad project on GitHub . . . . .	334
13.27. Customize the editor color scheme . . . . .	343
13.28. Import an EAGLE, Altium, or Cadstar project . . . . .	347
13.29. The circuit simulator . . . . .	353
13.29.1. Prepare the circuit for simulation . . . . .	355
13.29.2. Configure the simulator. . . . .	360



---

13.29.3. Simulate . . . . .	363
13.30. Import a KiCad 5 project . . . . .	367
13.31. KiCad project templates. . . . .	372
13.31.1. Using a system project template . . . . .	373
13.31.2. Create a user project template . . . . .	376
13.32. Archive/unarchive and share a project. . . . .	378
13.33. Buses . . . . .	381
13.34. Calculate the width of a trace. . . . .	386
13.35. Design a custom schematic sheet . . . . .	387
<b>Index . . . . .</b>	<b>396</b>

## Did you find an error?

Please let us know.

Using any web browser, go to [txplo.re/kicadr](https://txplo.re/kicadr), 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.

## About Tech Explorations

Tech Explorations creates educational products for students and hobbyists of electronics who rather utilize their time making awesome gadgets instead of searching endlessly through blog posts and Youtube videos.

We deliver high-quality instructional videos and books through our online learning platform, [txplore.com](https://txplore.com).

Tech Explorations courses are designed to be comprehensive, definitive and practical. Whether it is through video, ebook, blog or email, our delivery is personal and conversational.

It is like having a friend showing you something neat... the «AHA» moments just flow!

Peter left his career in Academia after his passion for electronics and making was rekindled with the arrival of his first Arduino. Although he was an electronics hobbyist from a young age, something the led him to study electrical and electronics engineering in University, the Arduino signalled a revolution in the way that electronics is taught and learned.

Peter decided to be a part of this revolution and has never looked back.

We know that even today, with all the information of the world at your fingertips, thanks to Google, and all the components of the world one click away, thanks to eBay, the life of the electronics hobbyist is not easy.

Busy lifestyles leave little time for your hobby, and you want this time to count.

We want to help you to enjoy your hobby. We want you to enjoy learning amazing practical things that you can use to make your own awesome gadgets.

Electronics is a rewarding hobby. Science, engineering, mathematics, art, and curiosity all converge in a tiny circuit with a handful of components.

Our courses have been used by over 70,000 people across the world.

From prototyping electronics with the Arduino to learning full-stack development with the Raspberry Pi or designing professional-looking printed circuit boards for their awesome gadgets, our students enjoyed taking our courses and improved their making skills dramatically.

Please check out our courses at [techexplorations.com](https://techexplorations.com) and let us be part of your tech adventures.

## About KiCad 6

KiCad 6 is the world's best open-source and free-to-use Printed Circuit Board tool. Its latest iteration, version 6, is packed with features usually found only in expensive commercial CAD tools.

KiCad 6 is a modern, cross-platform application suite built around schematic and design editors, with auxiliary applications: a custom symbol and footprint creator, calculators, a Gerber file viewer, and an image converter for customizing graphics in silkscreen or copper. KiCad 6 is a stable and mature PCB tool, a perfect fit for electronic engineers and hobbyists. With KiCad 6, you can create PCBs of any complexity and size without the constraints associated with the commercial packages.

Here are some of the most significant improvements and features in KiCad 6, both over and under the hood:

- Modern user interface, completely redesigned from earlier versions
- Improved and customizable electrical and design rule checkers
- Theme editor allowing you to fully customize the look of KiCad on your screen
- Ability to import projects from Eagle, CADSTART, and more
- Enhanced bus handling
- Full control over the presentation of information by the layout editor: set the visibility, color, and attribute of any board element, and create presets
- Use of Filters to define which elements of a layout are selectable – an essential feature for complex boards
- Enhanced interactive router helps you draw single tracks and differential pairs, and define their attributes (length, gaps, angles, etc.) with precision
- New or enhanced tools to draw tracks, measure distances, tune track lengths, etc.
- Enhanced tool for creating filled zones
- Data exchange with other CAD applications facilitated by a customizable coordinate system
- Realistic ray-tracing capable 3D viewer

## About this book

Printed circuit boards (PCBs) 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 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 resources. Signup to receive assistance and updates.

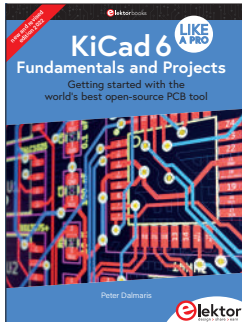
## Requirements

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

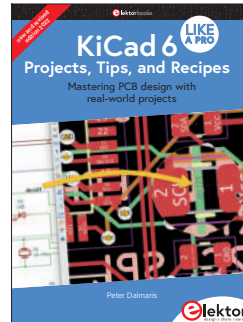
- A computer running Windows, Mac OS or Linux.
- Access to the Internet.
- A mouse with at least two buttons and a scroll wheel. I use a Logitech MX Master 2S mouse (see <https://amzn.to/2ClySq0>).
- Ability to install software.
- Time to work on the book, and patience.

## How to read KiCad 6 Like A Pro

I wrote this third edition of **KiCad Like A Pro** so that you can use it as a learning guide and as a reference source. It now consists of two separate but complementary books : **Fundamentals and Projects | Getting started with the world's best open-source PCB tool** (548 pages), and **Projects, Tips, and Recipes | Mastering PCB design with real-world projects** (400 pages).



**Parts 1 to 10**  
548 pages



**Parts 11, 12, 13**  
400 pages

The book you are holding now!

All examples, descriptions and procedures are tested on the nightly releases of KiCad 6 (also known as KiCad 5.99) and in KiCad 6 RC1.

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 to 8 because those will give you 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 zoom through the first chapters, and then proceed to the projects in Parts 9 and 10.

Throughout this work, you will find numerous figures that contain screenshots of KiCad. To create these screenshots, I used KiCad 5.99 and KiCad 6.0 RC1 running on Mac OS. If you are using KiCad under Windows or Linux, 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 where 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 work as a text book and companion. Whenever you read it, have KiCad open on your computer and follow along with the instructions.

This work has a web page with resources designed to maximize 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, you may be interested in the video course version of this book. This course spans over 25 hours of high-definition video, with detailed explanations and demonstrations of all projects featured in the book. The video lectures capture techniques and procedures that are just not possible to do so in text.

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.

## 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 [txplo.re/kicadr](https://txplo.re/kicadr).

The two available resources are:

1. **Photos and schematics.** Get high-res copies of the photos, schematics, and layouts that appear in the book.
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 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.



## Foreword by Wayne Stambaugh

In 1992 Jean-Pierre Charras started the KiCad project. From its humble beginnings as one man's goal to provide an electronics design application for his students to a full fledged open source project with a significant number of contributors, KiCad has become the choice for users who prefer an open source solution for electronics design. As the feature parity gap from proprietary EDAs continues to close, KiCad will continue to become more widely accepted and influential.

One of the enduring hallmarks of a successful open source software project is the publication of a «how to» book. With the publication of «KiCad Like a Pro», the KiCad project joins the other well known open source projects with that distinction. Whether you are a beginner or a seasoned professional user, this book has something for everyone. From properly configuring and using KiCad to design your first printed circuit board to advanced topics such as using git for version control this book is a valuable resource for any KiCad user.

Wayne Stambaugh  
KiCad Project Leader

## An introduction: Why KiCad?

Since KiCad first appeared in the PCB CAD world in 1992, it has gone through 6 major versions and evolved into a serious alternative to commercial products. I have been using KiCad almost daily since version 4 when I published the first edition of KiCad Like a Pro.

Once thought clunky and barely usable, it is now a solid, reliable CAD application. KiCad has been consistently closing the feature and performance gap against its commercial competitors. It has made leaps in adding powerful features and has significantly improved its stability.

Combined with the benefits of [free and open-source software](#)<sup>1</sup>, I believe that KiCad is simply the best PCB CAD software for most use cases.

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 organizations like CERN, the Raspberry Pi Foundation, Arduino LLC, and Digi-Key Electronics. The community is also active in contributing funds to cover development costs. Since joining the Linux Foundation, the KiCad project has received around \$90,000 in donations. The project used this money to buy development time and funding developer conference travel and meet-ups. To a large extent, this alone guarantees that KiCad's development will accelerate and continue to in the future.

Supporting the KiCad core team is the KiCad community. The community consists of over 250 thousand people worldwide that have downloaded a copy. These people support the KiCad project in various ways: they write code, create and share libraries, and help others learn. They write documentation, record videos, report bugs, and share hacks. During the KiCad 6 development cycle, the KiCad repository had around 14600 commits from the community. Based on this number, KiCad 6 is the most significant KiCad version ever in terms of changes. Another signal of the strength of the KiCad community is that KiCad 6 includes completed or nearly completed translations to nearly 20 languages. No other CAD software that I am aware of can boast this.

PCB manufacturers have also taken notice. Many of them now publish KiCad-specific tutorials, explaining how to order your boards. Some have made it possible to upload the KiCad native layout file from your project instead of generating multiple Gerber files.

And finally, KiCad is part of an expanding CAD ecosystem. You will find KiCad-compatible component libraries on the Internet's major repositories, such as [Snapeda](#)<sup>2</sup> and [Octopart](#)<sup>3</sup>, as well as native support in PCB project version control software for teams, such as [CADLAB.io](#).

KiCad's development and prospects have never been brighter than now. KiCad's [roadmap](#) has exciting new features and capabilities such as grouping board objects into reusable snippets and a stable Python API.

Why do I use KiCad? Because it is the perfect PCB software for my use case.

I am an electrical engineer with a background in electronics and computer engineering. But, above all, I am a technology educator and electronics hobbyist. The majority of my PCB projects eventually find themselves in my books and courses. My projects are very

---

1 [https://en.wikipedia.org/wiki/Free\\_and\\_open-source\\_software](https://en.wikipedia.org/wiki/Free_and_open-source_software)

2 <https://Snapeda.com>

3 <https://Octopart.com>

similar to those of other hobbyists in terms of complexity and size. I make things for my Arduino and Raspberry Pi courses. As a hobbyist, KiCad proved to be the perfect tool for me. Your use case may be different. You may be a university student completing an engineering degree. You may be a hobbyist or solo developer working in a startup company. You may be part of a team working on commercial projects that involve highly integrated multi-layer PCBs.

To help you decide whether KiCad is right for you, I have compiled a list of 12 KiCad Benefits. This list contained ten items in the second edition of the book. I added the last two items to highlight additional benefits brought about with KiCad 6.

Here they are:

**Benefit 1:** KiCad is open source. This is very important, especially as I spend 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 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 KiCad, I take it.

**Benefit 2:** It is free! This is particularly important for hobbyists. CAD tools can be expensive. This is worsening with most CAD software companies switching to a subscription-based revenue model. When you are a hobbyist or student or bootstrapping for a startup, regular fees do add up. Not to mention that most of us would not be using even half of the features of commercial CAD software. It is hard to justify spending hundreds of dollars on PCB software when there is KiCad. This brings me to Benefit 3

**Benefit 3:** KiCad is unlimited. There are no "standard", "premium" and "platinum" versions to choose from. It's a single download, and you get everything. While there are commercial PCB tools with free licensing for students or hobbyists, there are always restrictions on things like how many layers and how big your board can be, what you can do with your board once you have it, who can manufacture your board, and much more. And there is always the risk that the vendor may change the deal in the future where you may have to pay a fee to access your projects. I'll say again: KiCad is unlimited and forever! This is so important that I choose to pay a yearly donation to CERN 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, multi-sheet schematics, configurable rules checker, and differential routing are professional-grade. While you may not need to use some of them right away, you will use them eventually. You can add new features through third-party add-ons. The external 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 & Society Foundation](#)<sup>4</sup> became involved in their current capacity, I have seen a very aggressive and successfully implemented roadmap. When I wrote the first version of this list (August 2018), KiCad 5 was about one month old. The funding for KiCad 6 was already complete, and the road map living document was published. Three years later, KiCad 6 was delivered with promises fulfilled. Now, with KiCad 6 published, the [road map for the future](#)<sup>5</sup> looks just as exciting.

---

4 <https://cernandsocietyfoundation.cern/projects/kicad-development>

5 <https://gitlab.com/kicad/code/kicad/-/wikis/KiCad-Future-Versions-Roadmap>

**Benefit 6:** KiCad's clear separation of schematics and layout is a bonus to learning and using it. Users of other PCB applications often find this confusing, but I believe that it is an advantage. Schematic design and layout design are indeed two different things. Schematic symbols can be associated with different footprints that depend on the project requirements. You can use the schematic editor independently of the layout editor or in sync. I often create schematic diagrams for my courses that I have no intention of 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 online 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.

**Benefit 8:** KiCad works anywhere. Whether you are a Mac, Windows, or Linux person, you can use KiCad. I use it on all three platforms. I can take my KiCad 6 project from the Mac and continue working on Windows 10 without worrying about any software or project files glitches.

**Benefit 9:** KiCad is very configurable. You can assign your favorite keyboard hotkeys and mapping, and together with the mouse customizations, you can fully adapt it to your preferences. With the additions of the [plugin system](#)<sup>6</sup> and the Python API, it will be possible to extend your instance of KiCad with the exact features you need (or write them).

**Benefit 10:** If you are interested in creating analog circuits, you will be happy to know that KiCad ships with SPICE. You can draw the schematic in Eeschema and then simulate it in SPICE without leaving KiCad. This integration first appeared in KiCad 5, and it is now a stable feature.

**Benefit 11:** In the past, KiCad's release cycle was somewhat chaotic. New major versions would come out every two or three years, but no one knew ahead of time. In the future, KiCad will operate in a yearly release cycle. This is good for two reasons: One, commercial users who can now better predict how the software they depend on will change and when. Two, as KiCad users, all of us will be able to expect a reliable development schedule that prioritizes reliability. KiCad is now mature enough to be able to evolve predictably.

**Benefit 12:** KiCad is now a serious productivity tool for businesses. If you are an electronics engineer, you can proudly list it in your resume. If you are using it in your business, you can contract the KiCad Services Corporation, to customize the software to your exact requirements. I am talking about deep customization, not just changing the theme and the menu bars. This means that KiCad can fit precisely with your business. As far as I know, no commercial CAD application can do that. For the non-business users among us, we can expect many of these business-led improvements to flow into future software versions in the tradition of open-source software.

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

---

6 <https://techexplorations.com/blog/kicad/jon-evans-answers-kicad-6-questions/>

I have been using KiCad since version 4. I have packed almost everything I have learned as a KiCad user in this book. I have organized 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 four.

If you come from another PCB CAD tool and have experience designing PCBs, I only ask that you have an open mind. KiCad is most certainly very different from your current PCB tool. 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 go with the flow.

Let's begin!

**Note:** Parts 1 to 10 of the third edition of **KiCad Like a Pro** can be found in the book **KiCad 6 Like A Pro | Fundamentals and Projects**

Getting started with the world's best open-source PCB tool

548 pages

ISBN 978-3-89576-496-7 print

ISBN 978-3-89576-497-4 ebook

## Part 11 : Project - MCU datalogger

### 11.1. Project - Introduction

Welcome to Part 11. In this Part, you will design a printed circuit board for a microcontroller data logger. The data logger is based on an Atmega 328P-AU microcontroller and is supported by two EEPROMs and a real-time clock. Additional components on the board, such as status LEDs with their supporting resistors, two crystal oscillators, connectors, and capacitors.

You will use SMD packages for most components on a rectangular two-layer board with mounting holes on the four corners.

The project highlights are:

1. You will use Git to capture the history of the project's development.
2. You will design two versions of the PCB: one with two layers and one with four layers. Both will use data from the same schematic design.

KiCad, on its own, does not allow the creation of more than one layout for a schematic. Git makes this possible with the use of branches. This project will allow you to practice this aspect of Git-powered PCB design with KiCad.

The schematic design contains components distributed across two sheets. You can see the final schematic below (sheet 1):

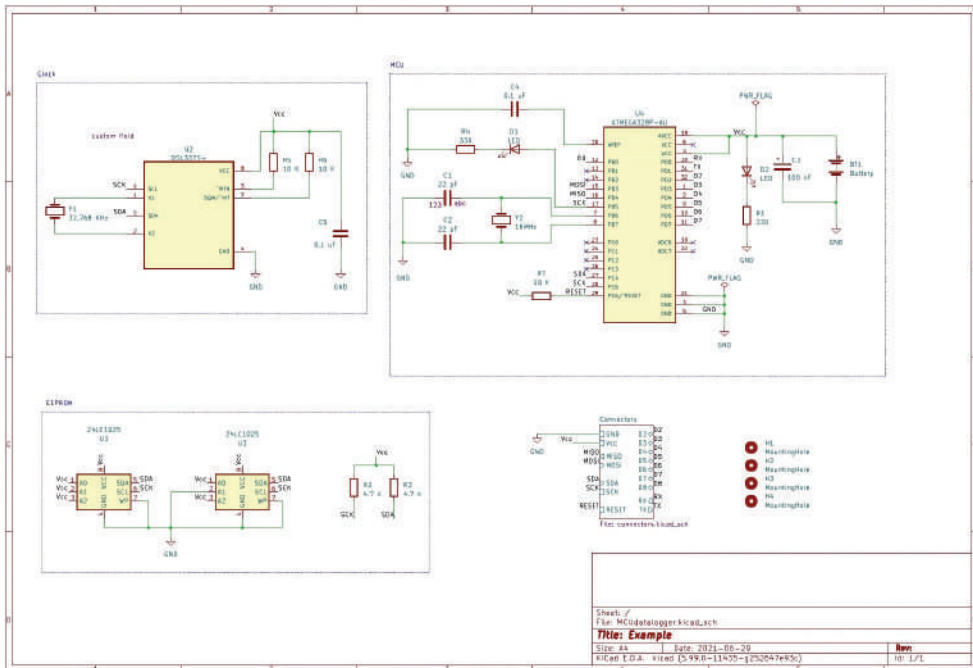


Figure 11.1.1: Sheet 1 of the project's final schematic design.

In Sheet 1, I have placed the main components of the board. Sheet 2 contains the connectors:

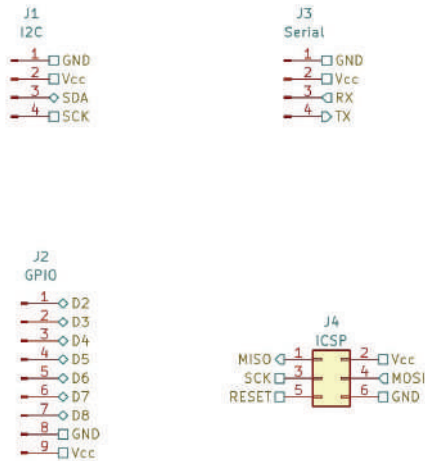


Figure 11.1.2: Sheet 2 of the project's final schematic design.

In the schematic design, I have used a combination of line wires and net labels. Other than the distribution of the components across the two sheets, the techniques I have used to draw the schematic should be familiar to you from previous projects. The most exciting aspect of this project is the layout design: you will design two versions of the PCB. A two-layer and a four-layer version. You can see the final version of the two-layer PCB layout below:

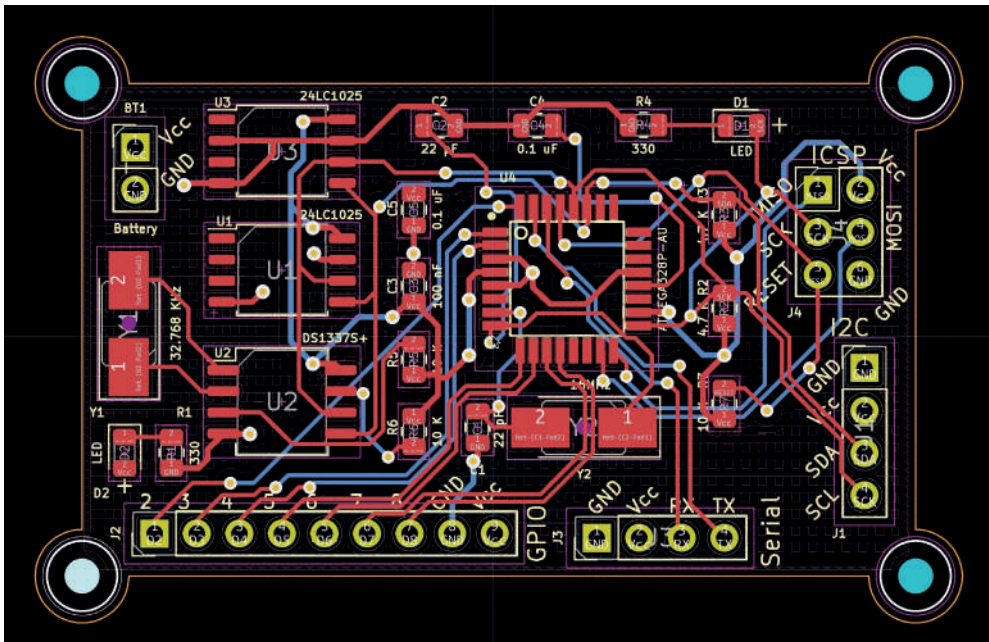


Figure 11.1.3: The project's final layout design (two layers).



You can see the final four-layer PCB below:

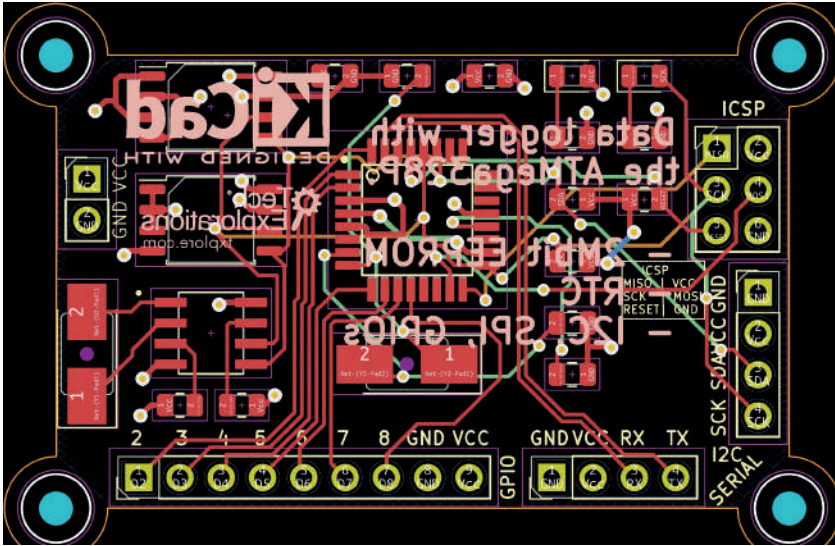


Figure 11.1.4: The project's final layout design (four layers).

The main objectives of this project are:

1. To help you practice skills you acquired in previous projects.
2. To use Git in a non-trivial project to extend KiCad's use cases in a single-schematic and multi-layout project.
3. To gain experience in creating multi-layer PCBs.

Below you can see the Bill of Materials for this project, as I have extracted it from the KiCad project (learn how later in this book):

Reference	Value	Footprint
BT1	Battery	Connector_PinHeader_2.54mm:PinHeader_1x02_P2.54mm_Vertical
C1, C4	0.1uF	Capacitor_SMD:C_0805_2012Metric
C2, C3	22pF	Capacitor_SMD:C_0805_2012Metric
C5	100nF	Capacitor_SMD:C_0805_2012Metric
D1, D2	LED	LED_SMD:LED_0805_2012Metric
H1-H4	MountingHole	MountingHole:MountingHole_2.1mm
J2	Conn_01x09_Male	Connector_PinHeader_2.54mm:PinHeader_1x09_P2.54mm_Vertical
J1, J3	Conn_01x04_Male	Connector_PinHeader_2.54mm:PinHeader_1x04_P2.54mm_Vertical



J4	Conn_02x03_Odd_Even	Connector_PinHeader_2.54mm:PinHeader_2x03_P2.54mm_Vertical
R1, R2, R6	10K	Resistor_SMD:R_0805_2012Metric
R3, R4	4.7K	Resistor_SMD:R_0805_2012Metric
R5, R7	330	Resistor_SMD:R_0805_2012Metric
U2	DS1337S+	Footprints:SOIC127P600X175-8N
U1, U3	24LC1025	Package_SO:SOIC-8_5.23x5.23mm_P1.27mm
U4	ATMEGA328P-AU	Footprints:QFP80P900X900X120-32N
Y1	32.768 KHz	Crystal:Crystal_SMD_5032-2Pin_5.0x3.2mm_HandSoldering
Y2	16 MHz	Crystal:Crystal_SMD_5032-2Pin_5.0x3.2mm_HandSoldering

Table 11.1.1: The Bill of Materials for this project.

I used Snapeda to find the symbol-footprint pairs for U4. You should be able to find all other symbols and footprints in KiCad's libraries.

In the next chapter, you will begin work on this project by creating a new KiCad project and Git repository.

## 11.2. Create the new project and Git repository

A primary objective of this project is to help you learn how to use Git within a KiCad project. If you are not familiar with Git, please consider reading the chapter "13.25. KiCad project management with Git" in the Recipes part of this book before continuing with the project. To keep the project flowing, I will not explain the meaning of each Git command that I use. Instead, I assume that you already know the basics of Git, and in the chapters that follow, I will focus on showing you how to use this knowledge in the context of a complete KiCad project.

Start KiCad, and create a new KiCad project. I have named my instance of the project "MCU Datalogger."

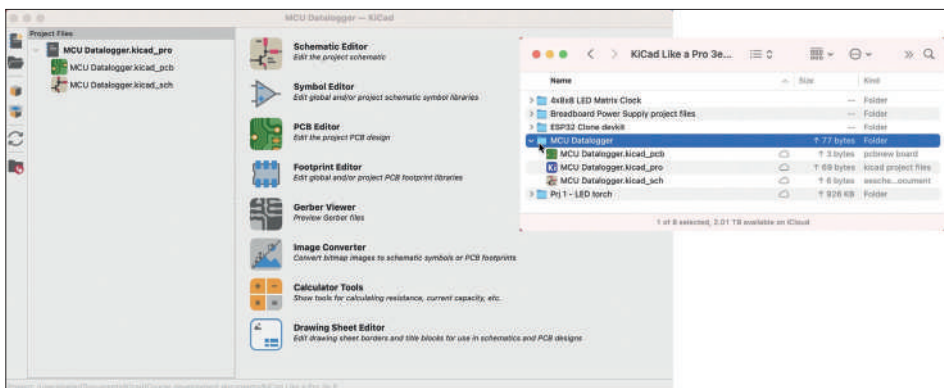
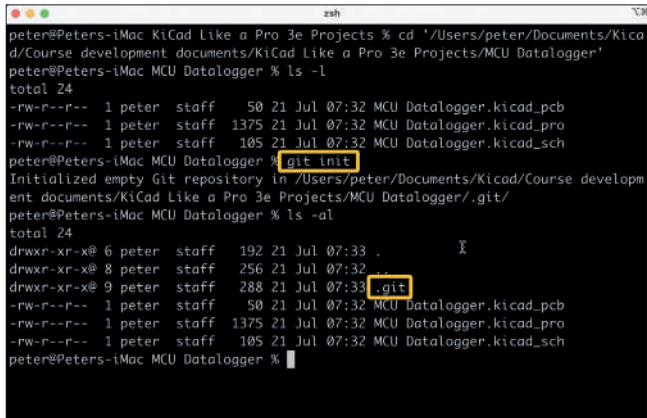


Figure 11.2.1: Starting a new project.

Now that you have a project directory for the project files create a new Git repository. Open a terminal window and navigate to the project directory. Issue the "init" command to initialize a new repository:

```
% git init
```

Below is my command line session. The second rectangle points to the new «.git» hidden directory, which will contain the project's history.



```

zsh
peter@Peters-iMac KiCad Like a Pro 3e Projects % cd ~/Users/peter/Documents/KiCad/Course development documents/KiCad Like a Pro 3e Projects/MCU Datalogger/
peter@Peters-iMac MCU Datalogger % ls -l
total 24
-rw-r--r--  1 peter  staff   50 21 Jul 07:32 MCU Datalogger.kicad_pcb
-rw-r--r--  1 peter  staff 1375 21 Jul 07:32 MCU Datalogger.kicad_pro
-rw-r--r--  1 peter  staff  105 21 Jul 07:32 MCU Datalogger.kicad_sch
peter@Peters-iMac MCU Datalogger % git init
Initialized empty Git repository in /Users/peter/Documents/KiCad/Course development documents/KiCad Like a Pro 3e Projects/MCU Datalogger/.git/
peter@Peters-iMac MCU Datalogger % ls -al
total 24
drwxr-xr-x@ 6 peter  staff  192 21 Jul 07:33 .
drwxr-xr-x@ 8 peter  staff  256 21 Jul 07:32 ..
drwxr-xr-x@ 9 peter  staff  288 21 Jul 07:32 .git
-rw-r--r--  1 peter  staff   50 21 Jul 07:32 MCU Datalogger.kicad_pcb
-rw-r--r--  1 peter  staff 1375 21 Jul 07:32 MCU Datalogger.kicad_pro
-rw-r--r--  1 peter  staff  105 21 Jul 07:32 MCU Datalogger.kicad_sch
peter@Peters-iMac MCU Datalogger %

```

Figure 11.2.2: Creating a new Git repository.

The new Git repository is ready. Let's start tracking the project files. Get the status of the repository to see what is not being tracked:

```

% git status
On branch master

No commits yet

Untracked files:
  (use «git add <file>...» to include in what will be committed)

    MCU Datalogger.kicad_pcb
    MCU Datalogger.kicad_pro
    MCU Datalogger.kicad_sch

nothing added to commit but untracked files present (use «git add» to track)
%

```

The repository is not tracking any files yet, and there are no commits. The working branch is «master.» Three files are not being tracked. Let's track them. Use the «add» command:

```
% git add .
```

You can use "git status" again to confirm that the three files are in the staging area but not yet committed. Go ahead and commit them using the "commit" command:

```
% git commit -am "First commit of new project."

[master (root-commit) c268ef2] First commit of new project.
3 Files changed, 68 insertions(+) E
create mode 100644 MCU Datalogger.kicad_pcb
create mode 100644 MCU Datalogger.kicad_pro
create mode 100644 MCU Datalogger.kicad_sch
%
```

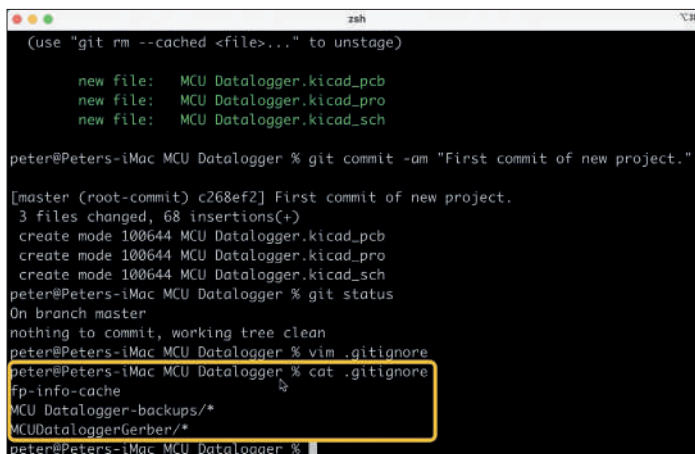
Use the "status" command to confirm that the working tree is clean:

```
% git status
On branch master
nothing to commit, working tree clean
%
```

KiCad automatically creates backup and cache files that don't have to be captured in the repository. I also want to exclude the contents of the Gerbers directory since I can always generate them at will. To exclude those files, use your text editor to edit the «.gitignore» file. Below you can see the contents of «.gitignore»:

```
fp-info-cache
MCU Datalogger-backups/*
MCUDataloggerGerber/*
```

You can see the contents of my ".gitignore" file below:



```

(use "git rm --cached <file>..." to unstage)
new file:   MCU Datalogger.kicad_pcb
new file:   MCU Datalogger.kicad_pro
new file:   MCU Datalogger.kicad_sch

peter@Peters-iMac MCU Datalogger % git commit -am "First commit of new project."

[master (root-commit) c268ef2] First commit of new project.
3 files changed, 68 insertions(+)
create mode 100644 MCU Datalogger.kicad_pcb
create mode 100644 MCU Datalogger.kicad_pro
create mode 100644 MCU Datalogger.kicad_sch
peter@Peters-iMac MCU Datalogger % git status
On branch master
nothing to commit, working tree clean
peter@Peters-iMac MCU Datalogger % vim .gitignore
peter@Peters-iMac MCU Datalogger % cat .gitignore
fp-info-cache
MCU Datalogger-backups/*
MCUDataloggerGerber/*
peter@Peters-iMac MCU Datalogger %
```

Figure 11.2.3: The contents of the ".gitignore" file.

At this point, you have a new KiCad project, and you are tracking it using Git. Let's continue with the schematic design workflow.

### 11.3. Schematic design

In this chapter, you will complete the schematic design of this PCB by following the schematic design workflow. You learned about this workflow in Part 6 of the book.

#### 11.3.1. Schema 1 - Setup

In this chapter, you will set up your schematic editor. Begin by opening the Preference window (KiCad → Preferences). I have kept the defaults, except for the following settings:

- Schematic Editor
  - Display Options
    - Grid thickness: 1 px
    - Min Grid spacing: 15 px
  - Colors: Using a custom theme with white background.
  - Field Name Templates: A new field named "Purpose"; visible.
- Schematic Setup
  - Project
    - Net Classes: create a new net class with the name "Power."
    - Text Variables: created a new variable:
      - Variable Name: project\_name
      - Text Substitution: MCU Datalogger with memory and clock
  - Page Settings:
    - Issue Date: copied today's date from the date field.
    - Revision: 1
    - Title: \${project\_name}

The setup is complete. Don't forget to save the project.

Let's continue with updating the Git repository. Use the "status" command to get an update:

```
% git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

   modified: MCU Datalogger.kicad_pro
   modified: MCU Datalogger.kicad_sch

Untracked files:
  (use «git add <File>...» to include in what will be committed)

   .DS_Store
   .gitignore
   MC Datalogger.kicad_pr1
```

```
no changes added to commit (use «git add» and/or «git commit -a»)
%
```

Git reports that there are changes in the project and schematic files and three new untracked files. I do not want to track the Mac OS system file «.DS\_Store» so I will add it to the «.gitignore» file. The other two files should be tracked. Here is my updated «.gitignore» file:

```
.DS_Store
fp-inFo-cache
MCU Datalogger-backups/*
MCUDataloggerGerber/*
```

Add the new files to Git (using the file name instead of the wildcard «.» character), and commit them:

```
% git add .gitignore
% git add "MCU Datalogger.kicad_prl"
% git commit -am "Setup of EEschema for new project."
[master ddb72ca] Setup of EEschema for new project.
4 files changed, 391 insertions(+), 9 deletions(-)
create mode 100644 .gitignore
create mode 100644 MCU Datalogger.kicad_prl
%
```

Use the "log" command to see Git's recent history:

```
% git log
commit ddb72ca7e5e8778Fcb7c0b1e8bf480f262635a6a (HEAD -> master)
Author: Peter Dalmaris <peter@txplore.com>
Date: Wed Jul 21 07:51:26 2021 +1000

    Setup of EEschema for new project.

commit c268ef258069718c44511d36a8ed03d4038570c9
Author: Peter Dalmaris <peter@txplore.com>
Date: Wed Jul 21 07:35:12 2021 +1000

    First commit of new project.
%
```

The log shows the two commits done so far, along with their commit IDs and other information.

In the previous chapter, where I created the new Git repository, I forgot to change the name of the «master» branch into «main.» The naming convention for the primary branch of Git repositories is now «main.» Online repositories, like Github, also follow this convention. If you plan to share your KiCad project with other people using Github, you should consider changing the primary branch of your project repository to «main» to avoid compatibility issues. I will make the name change now, so I don't forget again:

```
% git branch
* master
% git branch -m master main
% git branch
* main
%
```

In the session above, I used the «branch» command to check the name of the current working branch. The response was «master.» In the third line, I used «branch -m» to rename the «master» branch into «main.» In the fourth line, I confirmed that the renaming worked.

If (like me) you tend to forget to remake your Git repository's default branch, you can set it in the Git configuration like this:

```
% git config --global init.defaultBranch main
```

In the next segment of this chapter, you will add the schematic symbols into the editor.

### 11.3.2. Schema 2 - Symbols

Let's add the schematic symbols to the editor sheet. Use the BoM table from the introduction chapter to help you find the required symbols. Add all symbols except for the headers. I plan to add the symbols in a second sheet in the next segment of this chapter.

Download the files from Snapeda (all three types: symbols, footprints, and 3D models if available), and store them in your project folder. Install them as I describe in the relevant chapter from Part 7. You should be able to find all symbols in KiCad's symbol libraries, except for those for the [microcontroller](https://www.snapeda.com/parts/ATMEGA328P-AU/Microchip%20Technology/view-part/?ref=search&t=atmega328P-AU)<sup>7</sup> ("U4") and the [real-time clock](https://www.snapeda.com/parts/DS1337S/Maxim%20Integrated/view-part/?ref=search&t=DS1337S)<sup>8</sup> ("U2"). I installed these libraries in my Project Specific Libraries tab:

---

7 <https://www.snapeda.com/parts/ATMEGA328P-AU/Microchip%20Technology/view-part/?ref=search&t=atmega328P-AU>

8 <https://www.snapeda.com/parts/DS1337S/Maxim%20Integrated/view-part/?ref=search&t=DS1337S>